UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO DE JOINVILLE

POS-GRADUATION PROGRAM OF ENGINEERING AND MECHANICAL SCIENCES

MAIKOL FUNK DRECHSLER

ANTI-SLIP CONTROL WITH AN ACTOR-CRITIC REINFORCEMENT LEARNING ALGORITHM

JOINVILLE

2019

Maikol Funk Drechsler

ANTI-SLIP CONTROL WITH AN ACTOR-CRITIC REINFORCEMENT LEARNING ALGORITHM

Dissertation submitted to the Pos-Graduation Program of Engineering and Mechanical Sciences of Federal University of Santa Catarina to obtain the title of Master in Engineering and Mechanical Sciences.

Advisor: Prof. Dr. Thiago Antonio Fiorentin
Co-advisor: Prof. Dr. Ing. Harald Göllinger

Joinville

2019

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

Drechsler, Maikol Funk
Anti-slip control with an actor-critic reinforcement learning algorithm / Maikol Funk Drechsler ; orientador, Thiago Antonio Fiorentin, coorientador, Harald Göllinger, 2019.
136 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. Reinforcement Learning. 3. Anti-slip control. 4. Electrical vehicles. 5. Vehicle dynamics. I. Fiorentin, Thiago Antonio. II. Göllinger, Harald. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia e Ciências Mecânicas. IV. Título.

Maikol Funk Drechsler

Anti-Slip Control with an Actor-Critic Reinforcement Learning Algorithm

The present Master level work was evaluated and approved by the jury composed by the following members:

Prof. Werner Huber, Dr.
Technische Hochschule Ingolstadt

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

Prof. Roberto Simoni, Dr.
Universidade Federal de Santa Catarina

We certified that this is the **original and final version** of the concluding work, which was judged appropriate to obtain the title of master of engineering and mechanical sciences.

Documento assinado digitalmente
Regis Kovacs Scalice
Data: 27/01/2020 09:25:58-0300
CPF: 163.286.178-01

Prof. Dr. Régis Kovacs Scalice
Program coordinator

Documento assinado digitalmente
Thiago Antonio Fiorentin
Data: 27/01/2020 00:21:53-0300
CPF: 041.053.219-35

Prof. Dr. Thiago Antonio Fiorentin
Advisor

Joinville, 24 November 2019.

ACKNOWLEDGEMENTS

Firstly, I would like to sincerely thank my wife by the incentive and assistance in the participation of the Double Master degree, always understanding the time and financial limitations and standing by me during the change of country and life.

I also specially thank Professor Göllinger and Professor Fiorentin not just for the orientation on the development of this research, but also by the support on the participation of the Double Degree, clarifying my doubts about the curse, the documentation and the professional life.

I acknowledge Frau Ebenbeck for the support on the developing of this research, always supporting me in the laboratory activities and clarifying me the Germany culture, language and bureaucracy.

I also would like to thank in special the professionals of the AWARE of the THI that support me during the search of the scholarships and during the registration and acquiring of my and my wife's residence permits.

Finally, I thank AWARE, FAPESC and the Bavarian state for the financial support in Brazil and in Germany.

RESUMO

A implementação de controladores eletrônicos no setor automotivo melhorou significativamente a segurança, conforto e consumo de combustível dos veículos. Contudo, com a chegada de novas tecnologias como veículos elétricos e autônomos a demanda por controladores aumentou, ampliando desafios ainda presentes como controladores para sistemas completamente não lineares, a necessidade de dados dificilmente medidos ou dificuldade de definir modelos que representam o ambiente. Para avaliar as possibilidades de superar algumas destas restrições a presente pesquisa implementa diferentes algoritmos baseados em Machine Learning com Ator-Crítico para controlar o escorregamento do pneu de um kart elétrico de tração traseira. O controlador baseado em Machine Learning aprende por benefícios e punições o melhor comportamento a seguir e duas redes neurais são implementadas para julgar e controlar o sistema sem a necessidade de modelá-lo. Dois tipos diferentes de processo de aprendizado foram avaliados, incluindo algoritmos de aprendizagem iterativa e direta. O algoritmo proposto chamado Directly Trained Network Actor-Critic (DTNAC) simplificou o processo de aprendizado, permitindo a coleta de dados do ambiente em uma única vez e a realização do treinamento em uma estação off-line. Ambos controladores, cíclico e proposto foram avaliados nos estados nos quais foram treinados e em ambientes com diferentes pisos e manobras. A necessidade de aplicar a velocidade do veículo como uma entrada do controlador também foi analisada. Os controladores sem a velocidade do veículo como um dado de entrada apresentaram mais sensibilidade à qualidade dos dados usados no treinamento, enquanto o algoritmo que leva a velocidade em consideração apresentou um comportamento mais robusto. Todos os controladores avaliados apresentaram adequado controle do veículo nos estados treinados, contudo algumas limitações ocorrem quando a direção do veículo é simulada em manobras não treinadas. O algoritmo DTNAC com a velocidade do veículo como um dado de entrada do controlador, apresentou uma aplicação próspera. Quando treinado em solos de gelo e asfalto seco, o controlador foi capaz de governar o veículo na neve e no asfalto molhado em diferentes manobras, apresentando sucesso no controle de sistemas não-lineares sem a necessidade de modelar o sistema. Nos casos onde estão disponíveis dados que representem todos os estados, a remoção da velocidade do veículo como uma entrada também pode ser aplicada para reduzir a quantidade de dados medidos.

Palavras-Chave: Aprendizado de Máquina. Redes Neurais. Dinâmica veicular. Controlador. DTNAC.

RESUMO EXPANDIDO

INTRODUÇÃO

O sistemas de assistência ao motorista exercem um importante papel de auxiliar o condutor no controle do veículo. Estes sistemas possuem componentes eletrônicos, os quais, através de sensoriamento, consideram o veículo, o motorista e restrições de tráfego. (Braess & Seiffert, 2005). A aplicação destes controladores permite o aumento do conforto e segurança a bordo além de reduzir o consumo de combustível e a poluição ao meio ambiente através de sistemas de tração mais eficientes (Rathmann, 2007). Recentemente, a combinação de controladores eletrônicos com técnicas de aprendizado de máquina tem-se apresentado como uma ótima oportunidade de melhoria dos controladores atuais. Pesquisas aplicando estas técnicas tem sido realizadas em direção autônoma (Jaritz, et al., 2018), (El Sallab, et al., 2016), (El Sallab, et al., 2017), gerenciamento de baterias (Hsu, et al., 2010) e sistemas ativos de segurança (Radac & Precup, 2018), (de Amaral, 2018). Quando aplicado em combinação com redes neurais o aprendizado de máquina proporciona a possibilidade de aprender a controlar um sistema por tentativa e erro, obtendo sucesso em diversas atividades como em jogos virtuais (Mnih, et al., 2013) ou controle de hardware (Taitler & Shimkin, 2017). No ramo de segurança automotiva esta técnica foi implementada por de Amaral, et al. (2018) que avaliou um controle de estabilidade eletrônico em ambiente simulado com aplicação da técnica *Neural Fitted Q-interection,* a qual apresentou resultados inadequados para aplicação em tempo real. Por outro lado, os testes de bancada com sistemas ABS (*Anti-lock Brake System*) apresentaram sucesso na aplicação desta técnica (Radac & Precup, 2018). A implementação de controladores de tração, possui uma importante função de evitar o escorregamento dos pneus sob tração, contudo seu desenvolvimento possui limitações devido a não-linearidade dos pneus e a necessidade de controle em tempo real (Borrelli, et al., 2006). Controladores de aprendizado por reforço apresentam uma grande possibilidade de satisfazer estes requisitos, permitindo o uso se modelos que compreendem não linearidades e que atuam com um resposta adequada no tempo (Radac & Precup, 2018). Além disto, a aplicação de aprendizado por reforço requer de maneira geral um baixo investimento pois os sensores necessários para capturar dados para este controlador são os mesmos já empregados nos controles de tração convencionais. Desta forma, a presente pesquisa visa a avalição da viabilidade desta aplicação em um ambiente simulado, avaliando metodologias para treinamento e as características dos dados que são necessárias para um adequado treinamento do controlador. Além disso um método off-line de treinamento é proposto para facilitar o processo em casos nos quais dados reais são implementados para treinar as redes neurais.

OBJETIVOS

O objetivo principal desta pesquisa consiste na implementação de um algoritmo de aprendizado por reforço denominado Deep Actor-Critic, em tradução livre, Ator-Crítico Profundo, para controlar a tração de um veículo elétrico de tração traseira. A implementação em ambiente virtual tem como intuito evitar o alto escorregamento dos pneus em casos de baixo atrito do contato pneu-pista. Entre os objetivos específicos é possível destacar a validação do Deep Actor-Critic como um controlador adequado para implementações automotivas, compreender a influência dos parâmetros de treinamento e da presença da velocidade do veículo na desempenho do controle, comparar a aplicação dos treinamentos cíclico e direto nos resultados do controlador, analisar o comportamento do controlador sob mudanças de cenário incluindo aqueles não abordados durante o treinamento do mesmo e avaliar a possibilidade de aplicar este controlador quando os sensores possuem diferentes taxas de aquisição.

METODOLOGIA

O processo de treinamento da rede neural responsável por controlar o veículo pode ser dividida em três grandes áreas: Exploração do ambiente, processamento de dados e o treinamento propriamente dito. Para a exploração do ambiente o veículo elétrico disponível para testes foi modelado em um ambiente virtual, o qual leva em conta a dinâmica longitudinal do veículo, o contato pneu-pista e a dinâmica do sistema de tração, incluindo o motor elétrico o qual propulsiona o veículo. Durante o processamento dos dados, os dados obtidos são normalizados entre 0 e 1 para evitar a desigualdade das grandezas físicas, facilitando o treinamento da rede neural. A partir destes dados são calculados os benefícios e punições que o sistema irá receber para indicar os comportamentos adequados e inadequados apresentados pelo controlador. Estes benefícios e punições levam em conta se a posição virtual do pedal do acelerador proveniente do controlador se encontra dento do intervalo adequado, se a saída do controlador é inferior à aceleração desejada pelo motorista, se o escorregamento está dentro da faixa desejada. Quando todos estes fatores são atendidos o benefício é obtido pela diferença entre o a posição do pedal do acelerador pressionado pelo usuário e a posição do pedal de acelerador virtual gerado pelo controlador. A arquitetura de ator e crítico empregam duas redes neurais, o crítico avalia a qualidade das ações empregadas em cada estado enquanto o ator realiza o mapeamento entre estados e ações. No presente estudo uma rede neural com duas camadas ocultas e 20 nós em cada uma das camadas somos aplicadas. Esta possui de cinco a seis entradas dependendo da inclusão da velocidade do veículo e apenas uma saída a qual representa as punições ou benefícios esperados para o futuro, chamado função de valor. A rede que representa o ator possui arquitetura semelhante, porém possui apenas 12 nós em cada camada oculta e uma entrada a menos que o crítico. Convencionalmente o treinamento é realizado de forma iterativa, onde com base nos erros e acertos obtidos o algoritmo tem a oportunidade de melhorar seu desempenho. Desta mesma forma um aprendizado cíclico foi implementado no presente estudo. Neste caso, dados gerados no ambiente virtual são salvos em um banco de dados e usados no treinamento em mini lotes, os quais correspondem a uma porcentagem de todos os dados disponíveis. Com a combinação de estados, ações e função de valor, o crítico é treinado através da optimização de Levenberg-Marquart. Posteriormente a ação que maximiza a função de valor é determinada e o ator é treinado de maneira semelhante ao crítico. Este processo é repetido até que ocorra a convergência dos pesos das redes neurais. Posteriormente o comportamento do veículo é novamente testado no ambiente virtual e caso ocorram comportamentos inadequados o processo é reiniciado com nova coleta de dados. Contudo esta forma de treinamento requer um grande investimento de tempo na coleta de dados e simultâneo upgrade da rede neural treinada. Assim, um novo método off-line foi proposto, no qual uma maior quantidade de dados é coletada somente uma vez e as redes neurais são treinadas usando os dados coletados sem a influência do controlador. Neste caso, uma inserção de dados randômicos é necessária para simular a posição do pedal do acelerador enquanto esta é usada como saída do controlador. Esta adequação foi necessária pois a ausência do ator impede a geração de dados pelo mesmo. Para definição dos melhores parâmetros de treinamento ambas as metodologias foram avaliadas com o uso da técnica de *Design of Experiments* (DOE) com planejamento de dois níveis e k fatores. A influência dos fatores foi avaliada sobre o tempo de treinamento e o benefício médio apresentado pelo controlador quando submetido aos cenários nos quais a rede neural foi treinada. Definidos os melhores controladores para os treinamentos cíclico e direto e também para controladores com e sem a velocidade do veículo como entrada, estes foram comparados em diferentes cenários, incluindo aqueles usados no treinamento, diferentes manobras e diferentes condições de pista.

RESULTADOS E DISCUSSÃO

Avaliando as redes neurais obtidas constata-se que o método de treinamento cíclico apresenta instabilidade na convergência dos valores e baixa repetitividade. O treinamento cíclico da rede neural que inclui a velocidade do veículo como uma entrada não apresenta fatores que influenciam significativamente o tempo de treinamento. Enquanto isso, somente a quantidade de dados usados para o treinamento influencia a qualidade do controlador, onde menores quantidades de dados geram uma rede neural com melhor desempenho. Quando o treinamento cíclico é avaliado sem a presença da velocidade do veículo como entrada do controlador o fator de desconto apresenta-se como único que influencia significativamente o desempenho do controlador. Este fator é responsável por determinar a proporção na qual o treinamento tomará em conta os futuros benefícios esperados pelo sistema. De acordo com o DOE realizado, os controladores que não realizam observações futuras possuem melhor desempenho. Altos valores do fator de desconto e maiores quantidades de dados também levam a maiores tempo de treinamento durante o treinamento cíclico com velocidade do veículo como entrada do controlador. Quando avaliados os controladores treinados de forma não cíclica, aqueles que consideram a velocidade do veículo apresentam um treinamento mais robusto sem grande influencia da qualidade dos dados usados no treinamento. Por outro lado, quando a velocidade do veículo não é considerada, as características dos dados usados influenciam significativamente os controladores obtidos. Em ambos os casos com e sem a velocidade do veículo como entrada, a quantidade de dados usada para o treinamento influencia no tempo de treinamento. Todos os controladores apresentam desempenho adequado quando analisados nos cenários usados para o treinamento do controlador. Quando diferentes padrões de aceleração são analisados somente o controlador incluindo a velocidade do veículo e treinado de forma direta apresenta comportamento adequando enquanto os outros controladores apresentam oscilações principalmente quando o veículo encontra-se em baixas velocidades. Durante as avaliações em diferentes condições de pista o controlador cíclico com a velocidade do veículo apresenta melhores resultados. Contudo, o controlador treinado diretamente e com a velocidade do veículo apresenta funcionamento adequado apenas em solos que apresentam um coeficiente de atrito entre os empregados para o treinamento. Por outro lado quando pistas com menores coeficientes de atritos são empregas o controlador não apresenta resultados plausíveis. Visto que este controlador possui o melhor desempenho nos demais testes um novo treinamento foi realizado utilizando as pistas de asfalto seco e gelo para geração do banco de dados. A rede neural treinada nestas condições apresenta uma capacidade de interpolar todas as pistas avaliadas dentro deste intervalo de coeficiente de atrito, incluindo neve fresca e asfalto molhado. Por fim os resultados obtidos com esta rede neural para tempos de aquisição diferentes entre os sensores não apresentam resultados adequados. Para solução desta limitação é indicada a implementação de uma rede neural que inclua diferentes passos de tempo nas entradas para que seja possível a interpretação das diferentes taxas de aquisição.

CONSIDERAÇÕES FINAIS

A aplicação de aprendizado por reforço com redes neurais de ator e crítico apresenta-se promissora para o controle de sistemas não lineares e necessidades de aplicação em tempo real. Ambas as técnicas de treinamento aplicadas apresentam resultados adequados quando todas os cenários de teste e de treinamento coincidem, contudo melhores resultados são obtidos pelo controlador treinado diretamente e incluindo a velocidade do veículo nas entradas. Este controlador permite a interpolação dos dados, sendo possível controlar o veículo em situações não apresentadas durante o processo de treinamento. Esta possibilidade é vantajosa para campos como o automotivo onde todas os cenários possíveis não podem ser cobertos durante o treinamento. O controlador obtido pelo treinamento direto também apresenta a facilidade de

possuir uma coleta única de dados, evitando a necessidade de dirigir o carro mantendo pausas para treinamento ou de incluir um uma máquina com elevada potência computacional no veículo para realizar o treinamento on-line. A implementação de todos os parâmetros que descrevem fisicamente o sistema apresenta melhores resultados visto que os controladores que não empregam a velocidade do veículo como uma entrada não apresentam resultado tão satisfatório como aqueles que tem acesso a velocidade do veículo. O controlador proposto apresentou uma característica de interpolação, permitindo treinar apenas usando dados que representem condições extremas do ambiente. Esta característica apresenta grande potencial na redução da quantidade de dados necessários para o treinamento dos controladores, sendo indicado a avaliação desta característica em outras aplicações. Para trabalhos futuros indica-se a construção de uma rede neural que considere mais passos de tempo para a análise de diferente tempos de aquisição dos sensores e uma avaliação mais aprofundada do efeito dos dados de treinamento nos resultados dos controladores.

Palavras-Chave: Aprendizado de Máquina. Redes Neurais. Dinâmica veicular. Controlador. DTNAC.

ABSTRACT

The implementation of electronic controllers in the automotive sector significantly improved vehicle safety, comfort and fuel consumption. However, with new technologies as electric and autonomous driving, the demand by controllers increase significantly and some challenges are still present, as the controller of a completely non-linear system, the necessity of data hardly measured or difficulty to define models that represent the environment. To evaluate the possibility of overcoming some of these restrictions, the present research implemented different Reinforcement Learning Actor-Critic algorithms to control the wheel slip of a rear traction electrical go-kart. These Machine Learning based controllers learn by rewards and punishments the best behaviour to follow and two deep networks are implemented to judge and control the system without the necessity of environment modelling. Two different types of learning process were evaluated, including iterative and direct learning algorithms. The proposed Directly Trained Network Actor-Critic (DTNAC) simplifies the learning process and permits to collect data from the environment a single time, realizing the training process in an off-line station. Both cyclic and proposal controllers were evaluated on the trained states and in distinct environments as varied grounds and maneuvres. The necessity of the vehicle velocity as an input of the controller was also analyzed. The controllers without the vehicle velocity as an input showed more sensibility to the training data quality, while the algorithm that took the vehicle velocity into account had more robust behaviours. All the evaluated controllers presented an adequate control of the vehicle on the trained states, however, some limitation occurs when the vehicle driving is simulated on non-trained manoeuvres. The DTNAC algorithm with vehicle velocity presents itself as a prosperous application. When trained in ice and dry-asphalt, the controller was able to deal with snow and wet asphalt floors in different manoeuvres showing success in non-linear conditions without the necessity of modelling the system. In cases where the data that represent all the possible states are available, the removal of the vehicle velocity also can be applied to reduce the number of measured variables.

Key-words: Machine learning. Neural network. Vehicular dynamics. Controller. DTNAC.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABS	Antilock Braking System
AC	Alternated Current
ACO	Ant Colony Optimization
AI	Artificial Intelligence
ASC	Antislip Control
CG	Center of Gravity
DC	Direct Current
DDPG	Deep Deterministic Policy Gradient
DFQ	Deep Fitted Q Interaction
DOE	Design of Experiments
DOF	Degrees of Freedom
DPG	Deterministic Policy Gradient
DQN	Deep Q Network
DTNAC	Directly Trained Neural Actor-Critic
ECU	Engine Control Unit
ESC	Electronic Stability Control
IMC	Internal Model Control
MDP	Markov Decision Process
MF	Magic Formula
MFOC	Model-Free Optimization Control
MTTE	Maximum Transmissible Torque Estimation
NFQ	Neural Fitted Q Interaction
NFQCA	Neural Fitted Q Interaction with Continuous Actions
PID	Proportional, Integral and Derivative controller
PWM	Pulse Width Modulation
RL	Reinforcement Learning
SAE	Society of Automotive Engineering
SGD	Stochastic Gradient Descent
TCS	Traction Control System

LIST OF SYMBOLS

A	Factor effect	
a	Action	
a_x	Longitunidal acceleration	[m/s²]
B	Stiffness factor of the Magic Formula	
b	Bias of the neural network	
C	Shape factor of the Magic Formula	
c	Number of interactions of the DQN algorithm	
C_{Fk}	Tire stiffness	[N/m]
D	Peak factor of the Magic Formula	
d	Descendent direction	
D_A	Aerodynamic force	[N]
E	Curvature factor of the Magic Formula	
e	Error	
F_x	Longitudinal tractive force	[N]
F_{xf}	Longitudinal tractive force on the front wheel	[N]
F_{xr}	Longitudinal tractive force on the rear wheel	[N]
F_y	Lateral tractive force	[N]
GP	Driver throttle pedal position	
h	CG height	[m]
H	Hessian matrix	
h_A	Height of aerodynamic force actuation	[m]
I	Identity matrix	
i_a	Armature current	[A]
i_k	Motor-Wheel gear ratio	
I_m	Rotational Inertia of the motor	[kg/m²]
I_w	Rotational Inertia of the wheel	[kg/m²]
J	Performance objective	
J_R	Jacobian Matrix	

K	longitudinal slip	[m]
K_{gT}	Motor Torque constant	[Nm/A]
K_{gv}	Motor speed constant	[rad/V s]
L	Wheelbase	[m]
L_a	Motor armature inductance	[H]
L_I	Distance between the front axle and the CG	[m]
L_{II}	Distance between the rear axle and the CG	[m]
M_z	Aligning moment	[Nm]
P	Probability	
p	Preference	
Q	State-action value function	
r	Reward	
R_a	Motor armature resistance	[Ω]
R_{hx}	Trailer towing force in the horizontal direction	[N]
R_{hz}	Trailer towing force in the vertical direction	[N]
r_w	Dynamic radius of the wheel	[m]
R_{xf}	Resistance force on the front wheel	[N]
R_{xr}	Resistance force on the rear wheel	[N]
s	State	
S_h	Horizontal shift of the Magic Formula	
S_v	Vertical shift of the Magic Formula	
t	Timestep	
T	Probability of state transition	
T_a	Torque in the motor armature	[Nm]
T_m	Motor Torque	[Nm]
V	State value Function	
v_a	Motor armature voltage	[V]
v_i	Induced voltage	[V]
v_r	Wheel velocity	[m/s]
v_x	Vehicle velocity	[m/s]

W	Vehicle weight	[N]
w	Weights of the neural network	
W_f	Normal force on the frontal wheel	[N]
W_r	Normal force on the rear wheel	[N]
X	Longidutinal displacement	[m]
x	Node output	
Y	Lateral displacement	[m]
Y^Q	Target value	
$\bar{y}_{A^+}$	Result average with high level	
$\bar{y}_{A^-}$	Result average with low level	
Z	Vertical displacement	[m]
α	lateral slip angle	[rad]
α_m	Motor rotational acceleration	[rad/s²]
α_v	Level of significance	
α_w	Wheel rotational acceleration	[rad/s²]
β	Learning rate	
γ	Discount factor	
ϵ	Probability of exploration	
ε	Levenberg-Marquardt optimization parameter	
ζ	Target network	
θ	Ground inclination	[rad]
μ	Friction coefficient	
μ_x	Longitudinal friction coefficient	
μ_y	Lateral friction coefficient	
ξ	Update size	
π	Deterministic policy	
ρ	Policy performance	
φ	Activation function	
ψ	Policy parameter	
ω_m	Rotational speed of the motor	[rad/s]

SUMMARY

1 INTRODUCTION

A stable behaviour of lateral and longitudinal accelerations of the vehicle is necessary to an efficient vehicle movement prediction by the driver, since the driver commands the vehicle movements by your own movements combined with control systems (steering, acceleration, brake, etc.) (Dixon, 2007). To assist the driver during this task, these control systems have electronical components which consider the overall vehicle, driver and traffic-relevant constrain resulting in a better behaviour of the automobile (Braess & Seiffert, 2005).

The use of electronic systems in the control of motor vehicles is increasing faster since the beginning of the eighties. The electronic increase vehicle comfort and safety and reduce consumption due to better efficiency on powertrains systems. On the other side, the electronic control systems permit to attend the actual legislation about safety and environmental pollution (Rathmann, 2007).

In the safety area, electronic active safety systems as Antilock Braking System (ABS), Antislip Control (ASC) and Electronic Stability Control (ESC) help to prevent accidents and reduce traffic risks. Due to this safety improvement, since the end of nineties, these systems are applied in all vehicles in production in Germany, at least as an optional (Robert Bosch GmbH, 1998). After that, predictive safety systems as collision warning and emergency braking were also developed (Rathmann, 2007) as first steps of vehicle automation, following to recent studies about autonomous driving (Naujoks, et al., 2016).

Recently, the combination of electronic systems and reinforcement learning method presents itself a good opportunity for improvement in vehicles tasks. The researches using these technics are implemented in autonomous drive (Jaritz, et al., 2018), (El Sallab, et al., 2016), (El Sallab, et al., 2017), battery management (Hsu, et al., 2010) and safety active systems (Radac & Precup, 2018), (de Amaral, 2018).

Reinforcement Learning is a topic of the computer sciences area, in which intelligent programs, called agents, work in an environment in constant cyclic interaction. This relationship between environment and agent permit the system adaptation and learning through positive or negative feedbacks called rewards and punishments respectively (Nandy & Biswas, 2018).

The earliest use of Reinforcement Learning (RL) framework was Samuel's checkers in 1959. In this research, a computer learns to play checkers just based on the

game rules, a sense of direction and parameters that show the targets of the game (Samuel, 1959). After that, RL was improved aiming the learning of the system control with the prediction of future behaviour based on past experience (Sutton, 1988). On the automotive area, the RL was recently applied in battery management also showing good results. The RL implementation allows that the vehicle learns the best way to control the energy, aiming adequate comfort to the user and maximum displacement (Hsu, et al., 2010).

The RL combined with neural networks, named neural fitted Q interaction showed promising with the possibility of excellent performance in game playing (Mnih, et al., 2013) and systems control (Taitler & Shimkin, 2017). The recent researches using neural fitted Q interaction in the automotive area are focused on autonomous driver, usually using race game environments to evaluate the vehicle response to an RL control without risks (Jaritz, et al., 2018), (El Sallab, et al., 2016), (El Sallab, et al., 2017). De Amaral, et al. (2018), evaluated an ESC strategy with Neural fitted Q interaction with simulations in CarMaker environment, but the results showed inadequate learning times to real applications. On the other hand, the test of an ABS system in benches, showed adequate results with Neural fitted Q interaction controls (Radac & Precup, 2018).

In the vehicle safety area, the Traction Control exerts an important function of avoiding the wheel slipping during acceleration, improving acceleration and cornering. In 1998, the Traction Control of electrical vehicles had been already researched due to the easy control of electric motors and the benefit of use low-drags tires to improve the battery autonomy (Hori, et al., 1998).

Traction control systems development presents challenges due to the nonlinearities of tires and simplicity needed to the real-time application (Borrelli, et al., 2006). Besides that, describing the behaviour of the tire is a very complex task, usually applying empirical or semi-empirical equations as the so-called Magic Formula (Pacejka, 2002). Thus, the use of model-free control techniques, which do not need the model of the system, can be a good opportunity for improvement (Radac & Precup, 2018).

The RL methods show a good possibility to fulfil these requirements, allowing the use of non-model techniques and applying adequate time response to control the system (Radac & Precup, 2018). A successful implementation can provide safety improvement to the automotive sector, due to better control of tire slips and avoidance of vehicle uncontrollability. The consumption can be also reduced by the use of low-drags tires (Hori, et al., 1998).

Moreover, to the impletation of the Traction Control based on Reinforcement Learning, low or no cost with sensor development is needed, since the necessary data is already collected to the conventional Traction Control systems. In this case, the costs are the algorithm and methodology development and any increase on the computation power of the control processor.

In this way, the present research aims to develop and investigate a Traction Control system to a rear-wheel driven electric vehicle, based on an actor-critic reinforcement learning. The evaluation takes into account the longitudinal behaviour of the vehicle and permits to understand the possible implementation of reinforcement learning methods in automotive controls.

To this task, the algorithm receives data from the vehicle sensors and the driver intention of acceleration from the throttle pedal position. Based on the actual state of the vehicle and drive intention the controller generates a new optimized virtual throttle pedal position that controls the electric motor.

During the research, two different training algorithms were evaluated. The first implemented a cyclic approach which uses the last trained controller to collect new data in each training cycle, due to the difficulty in real applications a second trained process was proposed using a single data collection without the interaction of the controller.

1.1 OBJECTIVES

The main objective of the research is to implement a Deep Actor-Critic Reinforcement Learning algorithm to control the traction of a rear-wheel driven electric vehicle. The implamentation is done in a simulated environment, aiming to avoid the inadequate slip of the wheels.

Among the specific objectives of the research can be highlighted:

- Validate the Deep Actor-Critic algorithm as an adequate controller to automotive implementations;
- Understand the training parameter influence and the vehicle velocity presence on the performance of the controller;
- Compare the cyclic and the direct training process by the vehicle response when the RL controller is implemented;
- Analyze the controlled vehicle behaviour with scenarios changing, including different tracks and maneuvres;

- Evaluate the operation of the controller when the sensors have different acquisition times.

2 LITERATURE REVIEW

This section is divided in the well stabilished physical background necessary to understand the current research and the state-of-art of the related areas including anti-slip control systems, RL and neural network applications.

2.1 PHYSICAL BACKGROUND

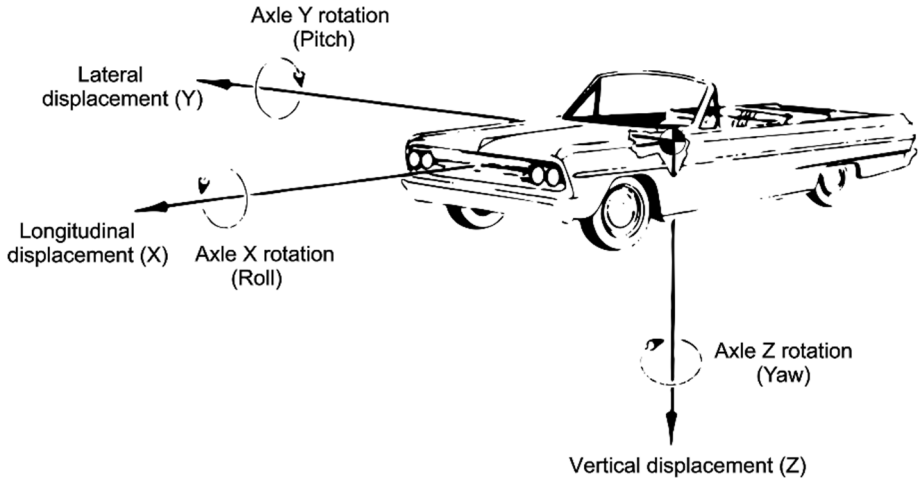
The physical background sub-section include the main physical and mathematical methodologies applied during the research. Inicially the physical behaviour of the vehicle is explained to support the understantding of the environment simulation. After that, the basis of Anti-slip control are clarified to combustion and electrical powertrains. To finish, the components and the mathematical development of the RL and neural network are exposed.

2.1.1 Vehicle Dynamics

The vehicle is composed of many individual components with different positions inside the vehicle, however, the whole vehicle moves together. In this way, in a simplified method, all the accelerations can actuate on a concentrated mass, positioned in the so-called Center of Gravity (CG) (Gillespie, 1992).

To create a standard according to the coordinates position on the vehicles CG, The Society of Automotive Engineering (SAE) (1976) defined it in the J670e standard as presented in Figure 1.

Figure 1 - Vehicle Coordinate System.

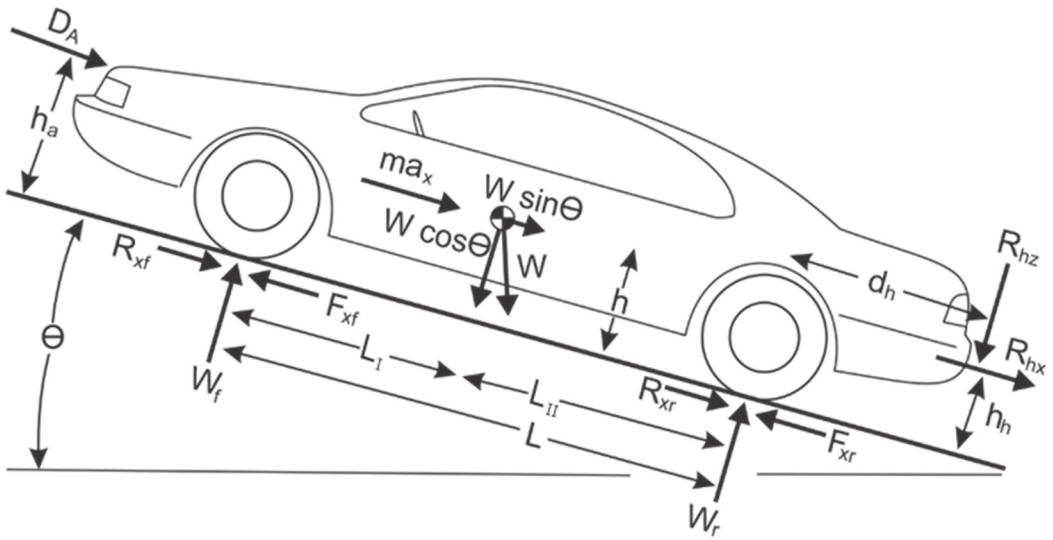


Source: J670e standard SAE (1976).

In Figure 1 the coordination system follows the right-hand rule, with the Z axle pointing down. The rotation on X, Y and Z axes are respectively Roll, Pitch and Yaw.

The evaluation of the dynamic behaviour of the vehicle on X axle (longitudinal direction), is possible by using Newton's Second Law. In Figure 2 is presented the main forces that actuated in a passenger car (Gillespie, 1992).

Figure 2 - Forces acting on a vehicle.



Source: Gillespie (1992, p. 11).

In Figure 2, W is the weight of the vehicle acting on vehicle CG. When the vehicle is on an inclination, this force presents two components, one perpendicular to the ground (WcosΘ) and another parallel to the road (WsinΘ).

The normal forces from the road are shown in Figure 2 to front and rear wheels as W_f and W_r respectively. Parallel with the road, are presented tractive (F_{xf}, F_{xr}) and resistive (R_{xf}, R_{xr}) forces. When the vehicle is towing a trailer, there are also the forces R_{hz} and R_{hx}, which represent the vertical and longitudinal interaction with the attached trailer (Gillespie, 1992).

The "d'Alembert force" denoted by the vehicle mass times the acceleration acting at the vehicle CG can represent the Inertia of vehicle movement. This force always acts in the opposite direction of the acceleration or braking force applied on the vehicle (Den Hartog, 1948).

The aerodynamic force D_A acts on the centre of pressure of the vehicle generating a longitudinal resistance force. In addition, there is a distance between the centre of pressure and the vehicle CG. This distance generates a pitch moment on the vehicle. The generated moment directly depends on the difference between the CG height (h) and the D_A actuation height (h_a). The D_A force depends on geometric parameters as vehicle frontal area, drag coefficient and the square of vehicle velocity. The square dependence on velocity means that the force increase fast with the velocity increasing, so the aerodynamical force only has a significant influence at velocities over 40km/h (Hucho, 1998).

The forces transmission from the vehicle to the ground is directly depended on the tires distortions and slip. Several types of mathematical models were developed to describe the relationship between the longitudinal force, side force and moment with the tire deformation. However, some limitations are still present in describing the dynamical behaviour of the tyre (Pacejka, 2002).

2.1.2 Tyre Model

According to Pacejka (2002), when evaluating just the longitudinal behaviour the tire can be described as a spring with stiffness C_{Fk}. In this case, the correlation between the longitudinal F_x and the longitudinal slip ratio K is given by Equation (1).

$$F_x = C_{Fk}K \tag{1}$$

However, with Equation (1) is it possible to evaluate the tire behaviour just in the linear range, while the real force presents a nonlinear shape. To overcome the

simplifications new equations were developed including the description of peak accelerations in determined slip ratios. Among them it is possible to highlight the semiempirical Magic Formula (MF) (Pacejka, 2002).

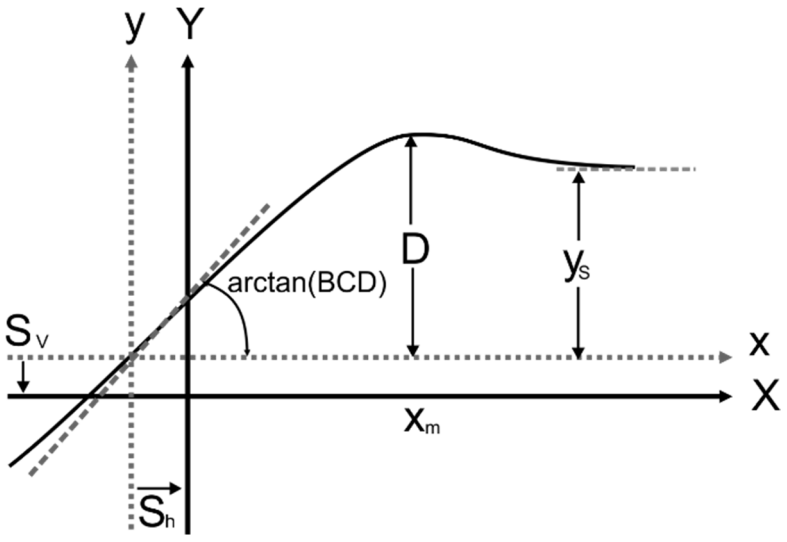
Widely used in vehicle dynamics studies, this formula is a combination of physical evaluations and experimental results. Based on experimental tests of the tire, the constants of the equations can be obtained and the model is used to simulate or estimate the forces based on the slip. The model is governed by Equation (2) (Pacejka, 2002):

$$y = D \sin [C \arctan \{Bx - E(Bx - \arctan Bx)\}] \quad (2)$$

where: $Y(x) = y(x) + S_V$ and $x = X + S_h$

Y is the output variable: longitudinal force F_x, side force F_y or aligning moment Mz. X is the input parameter: it can be tangent of lateral slip angle α or longitudinal slip K. B, C, D and E refer to stiffness, shape, peak and curvature factors respectively. S_h and S_v refer to horizontal and vertical shift respectively. In Figure 3 is presented the curve produced by Equation (2) and the meaning of curve parameters.

Figure 3 - Curve produced by the Magic Formula Equation to the longitudinal force.



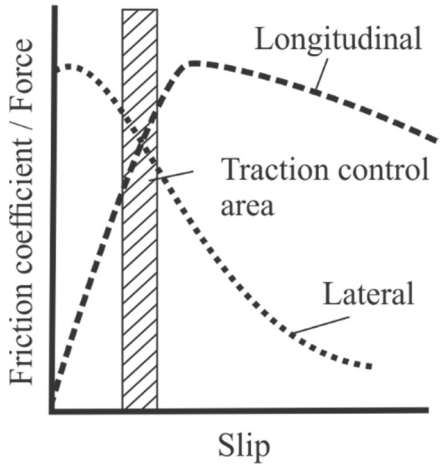
Source: Pacejka (2002, p. 173).

In addition, the traction force is correlated with the normal force on the wheel by the friction coefficient μ as shown in Equation (3).

$$F_{x(F,R)} = \mu W_{F,R} \quad (3)$$

In this way, adapting the MF parameters, longitudinal (μ_x) and lateral (μ_y) friction coefficients can be output parameters as well (Ružinskas & Sivilevičius, 2017). Comparing lateral and longitudinal coefficients based on tires slip it is possible to obtain Figure 4 (Hori, et al., 1998).

Figure 4 - Friction coefficient correlation with slip.



Source: Adapted from Hory, et al. (1998, p. 1131).

Evaluating Figure 4, the lateral friction coefficient is maximum near to zero and when the slip increase, the friction between rubber and track decrease quickly. This behaviour can occur due to high acceleration or suddenly steering by the driver. The low friction permits an unstable behaviour of the vehicle including drift out or spin (Robert Bosch GmbH, 1998)

By contrast, the longitudinal coefficient increase with the slip increment until a maximum point where the friction decrease slowly. However, the use of MF just is possible in steady-state applications where the experimental data is collected. Due to this limitation, it is difficult to implement MF equation in onboard systems since the vehicle load, tire type, tire pressure and track conditions change frequently (Braess & Seiffert, 2005).

To control the traction force of the vehicle and keep a stable behaviour usually are implemented traction control systems, which keeps the vehicle-tire velocity difference in an adequate ratio (Robert Bosch GmbH, 1998).

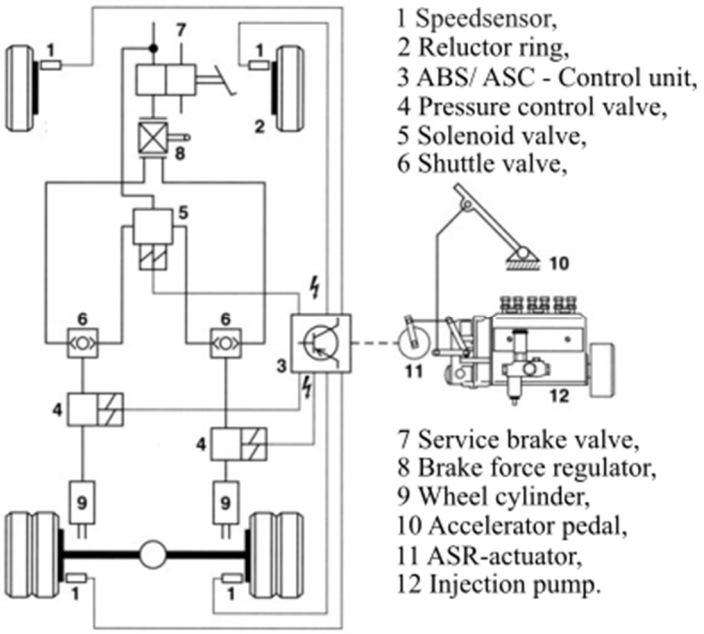
2.1.3 Intern Combustion Engine Traction Control

The Traction Control System (TCS) or Antislip Control (ASC) suppresses tire slip when the vehicle accelerates in a low friction surface as the icy or muddy road (Hori, et al., 1998). By maintaining the traction forces at an adequate level, this system assurances vehicle stability, increase acceleration performance and reduce the tire wear. A brake ASC and an engine/motor ASC can compose the control system (Braess & Seiffert, 2005).

The brake ASC actuate using the Antilock Brake System (ABS). Based on wheel speed sensors, the system brakes the wheel that present high slip compared with the vehicle average velocity. The benefits of this composition are the possibility to control the wheels individually and the transference of torque to the free wheel by the differential. On the other hand, the brake cannot be pressed for a long time, due to the brake overheating (Braess & Seiffert, 2005).

To reduce the brake use, the motor or engine ASC can be applied in spite of not permit the individual wheel control. The motor torque reduction can generate vehicle stability in manoeuvres which both wheels present a low friction coefficient, as a linear acceleration on ice. Due to this characteristic, the implementation of engine ASC avoids dangerous manoeuvre during acceleration (Braess & Seiffert, 2005).

Figure 5 - Brake and motor ASC to a commercial vehicle.



Source: Adapted from Robert Bosch GmbH (1998, p. 198).

On this approach, the ABS/ASC control unit receives from the Engine Control Unit (ECU) a driver request through the throttle pedal position. This data is compared with the brake and slip scenarios to return to the ECU a desired fuel injection to the engine. In some cases, the ASC unit directly actuated on the engine providing mechanical retarder in exhausts gases (Robert Bosch GmbH, 1998). In Figure 5 is possible to identify a Brake and Engine ASC scheme to a commercial vehicle.

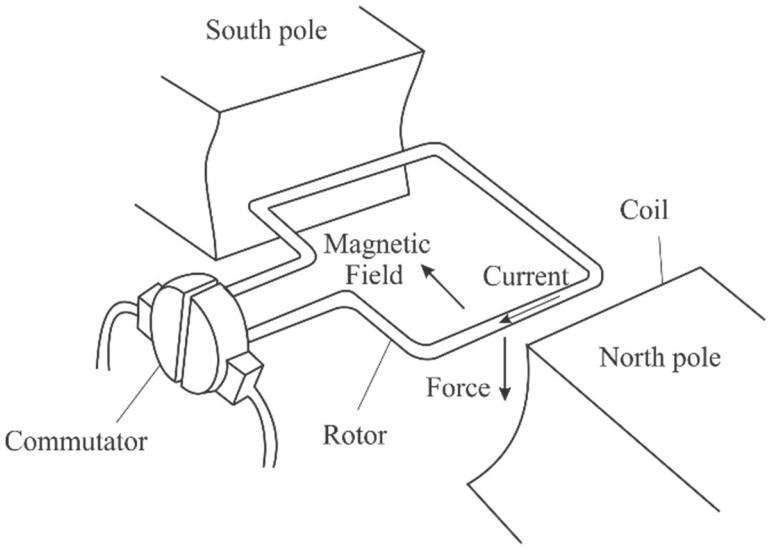
The ASC was showed at first time in 1986 as an extension possibility to the already installed ABS system. According to Gerstenmeier (1986), using small hardware or software improvement would be possible to implement the ASC in the vehicle.

The suceeding researches in ASC area focused on the control strategy, usually using the same actuators and sensors. Among the control strategies, it is possible to highlight the control method of sliding mode, widely used due to the robustness in non-linear dynamic systems (Kabganian & Kazemi, 2001).

2.1.4 Electric Motor Traction Control

The first electrical motors implemented in vehicles was the direct current (DC) motors due to the easy control and connection with the DC batteries. The DC motor works based on three main components: coil, rotor and a commutator as showed the simplified scheme in Figure 6 (Khajepour, et al., 2014).

Figure 6 - Schematic of a brushed DC motor.



Source: Khajepour, Fallah and Goodarzi (2014, p. 55).

As shown in Figure 6, a set of coils mounted inside the motor generates a magnetic field that combined with the current on the rotor create the torque. The commutator act switching the supply voltage to the revolving rotor from the stationary brush making the rotor turn. In this way, changing the voltage and consequent current level in the rotor the final torque is easily controllable.

However, the DC motor has disadvantages about its weight, efficiency, reliability and high maintenance with the brushes wear when compared with the alternated current (AC) motors. On the other hand, the AC motors have more complex control systems correlated with the frequency of the current waves (Khajepour, et al., 2014).

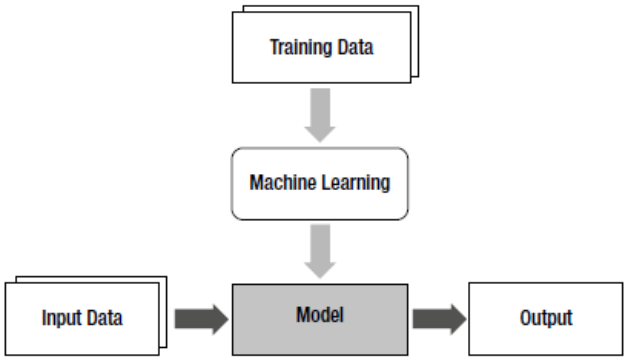
2.1.5 Machine Learning

The Machine Learning is a field of computer science, considered from some authors as a kind of AI (Kim, 2017) and for others as a field that is closely related with the AI, as well as it is related with pattern recognition and computational statistics (Paluszek & Stephanie, 2017). However, what really defines the Machine Learning field is the ability to use existing data to predict future data.

Usually implemented to recognition and statistics tasks, where it is not feasible to write algorithms, machine learning has the ability to learn directly from the environment data (Paluszek & Stephanie, 2017). Based on captured data the algorithm generates a model as a final product, permitting predict the future data based on the model generated by the old data. (Kim, 2017).

In these cases, the model suffers adaptations without the direct human intervention, allowing the solution of problems for which analytical models are hardly available. The algorithm learns by already known input-output correlations that permit the model update as a fitting process. In Figure 7 it is indicated the relation between the training process and the model application (Kim, 2017).

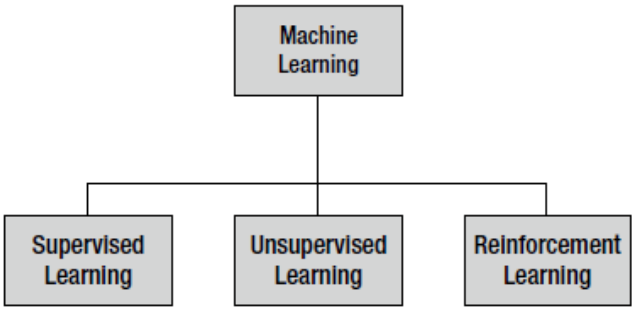
Figure 7 - Training and input data.



Source: Kim (2017, p. 5).

As showed in Figure 7, some training data is used by the machine learning process to obtain the model that represents the input-output relation (Paluszek & Stephanie, 2017). When the model is already learned, the input data can be used to obtain the waited output results. The learning process can be divided into three principal types as showed in Figure 8 (Kim, 2017).

Figure 8 - Three types of machine learning techniques.



Source: Kim (2017, p. 12).

The supervised learning means that a specific training set is applied to the system. In this part, a human gives a correct combination of input and output to the system. It is not necessary that the human actively validate the results, but the system needs to know specifically which output is expected from each input (Paluszek & Stephanie, 2017).

During the training, the correct choice of data is very important, in the way that the data need to be widespread enough to permit the input comprehension returning a correct output (Kim, 2017).

The unsupervised learning acts just with input data, without a correct output to learn. In this case, the main principle is usually to investigate the data with the possibility of finding hidden structures. For example, it is possible to evaluate a face identification data to understand what could be a correct recognition output (Paluszek & Stephanie, 2017).

Reinforcement learning (RL) act based on input, some output and a grade for the output. This type of machine learning is widely used for classification because, in the place of the correct output, classification of the analyzed data is learned. The RL is used in more applications then unsupervised and supervised learning (Kim, 2017).

2.1.6 Reinforcement Learning

The RL permits a continuous learning process based on the environment relation and feedbacks that consider the environment and system interactions (Nandy & Biswas, 2018). Therefore, the RL presents itself as a good tool to improve vehicle control as autonomous driven or simpler systems like ASC or ABS.

The system that executes the action receives the name of agent. The agent work in a known or unknown environment constantly adapting and learning based on given points. The feedback may be positive, known as reward, or negative called punishment. The agent and the environment interaction is called state. Based on the state, the agent chooses the next action aiming to maximize the reward or minimize the punishment (Nandy & Biswas, 2018).

One of the pioneer use of Reinforcement Learning (RL) framework was Samuel's checkers in 1959. In this research, a computer learns to play checkers just based on the game rules, a sense of direction and parameters that show the targets of the game (Samuel, 1959). After that, RL was improved aiming the learning of the system control with the prediction of future behaviour based on past experience (Sutton, 1988).

2.1.6.1 Elements of reinforcement learning

Beyond the agent and the environment, it is possible to identify sub-elements of RL system: policy, reward function, value function and optionally a model of the environment (Sutton & Barto, 2017).

A deterministic policy π is a mapping from the agent-environment interaction to the action to be taken. The policy can be very simple equation, or in some cases a complex search process. In general, policies may also be stochastic and can change during the time due to the learning process (Sutton & Barto, 2017), (Wiering & van Otterlo, 2012).

The reward function defines the goal of the system. It maps the state or state-action to a reward or a punishment, composed by a value, defining how good or bad is the actual state. The main objective of the agent is to maximize the reward or minimize the punishment in the long run. The reward function is defined by the programmer and cannot be changed by the agent. However, it may serve as a basis to change the policy (Sutton & Barto, 2017).

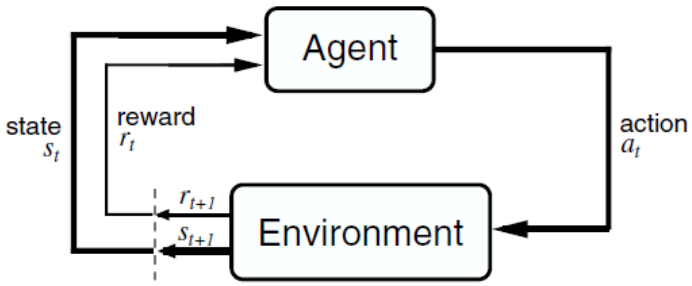
While the reward function indicates how good is a state in an immediate sense, the value function makes the same to the long run. The value function can be explained as the amount of reward that the agent can expect to the future starting from the actual state. This function avoids that the agent chooses an action that gives an immediate great reward but is preceded by low rewards (Sutton & Barto, 2017).

Finally, the model of the environment mimics the behaviour of the environment. The model permits to predict the next state and reward based on simulated actions, aiming to choose the best action before taking it. The use of models in machine learning is relatively new, and bring the possibility of planning the action rather than act exclusively with try and error learners (Sutton & Barto, 2017).

2.1.6.2 The Agent-Environment Interface

The RL problem consists of learning from interaction until achieving a goal. The learning and decision-maker is the agent, everything that is not controllable by the agent is called environment, which the agent interacts. The agent and the environment interact cyclically as shown in Figure 9. The agent select actions and the environment respond to these actions presenting a new state. In this step, a reward also can be calculated based on the new state and the taken action (Sutton & Barto, 2017).

Figure 9 - The agent-environment interaction in reinforcement learning.



Source: Sutton and Barto (2017, p. 52).

The agent and the environment interact in discrete time steps. At each step t, the agent receives some representation of the state s_t and choose a possible action a_t to this state. The next state s_{t+1} is a result of the later action and based on these factors a reward r_{t+1} is calculated. State and reward are sent to the agent that chooses a new action. The agent always tries to maximize the reward, choosing the action that returns the best value during the time (Sutton & Barto, 2017).

2.1.6.3 Online versus offline learning

Online learning performs the learning directly on the problem instance, while off-line learning uses a simulator of the environment as a safe and fast way to train the policy. The online learning is difficult applicable, due to the time consuming and the safety of use simulation environments mainly in arbitrary training situations (Wiering & van Otterlo, 2012).

Sometimes an adequate policy can be created in a simulated environment and then, fine-tuned in the real task. This learning process permit that the agent makes dangerous errors in the simulation and tune the behaviour taking into account the real physical behaviour of the system (Wiering & van Otterlo, 2012).

2.1.6.4 The Exploration-Exploitation Trade-off

The most important characteristic that differentiates the RL from the other learning algorithms is the evaluation of the actions taken rather than the use of instructs by giving correct actions. This creates a necessity of trial-and-error search to find the best policy, this search received the name of exploration. On the other hand, at any time the

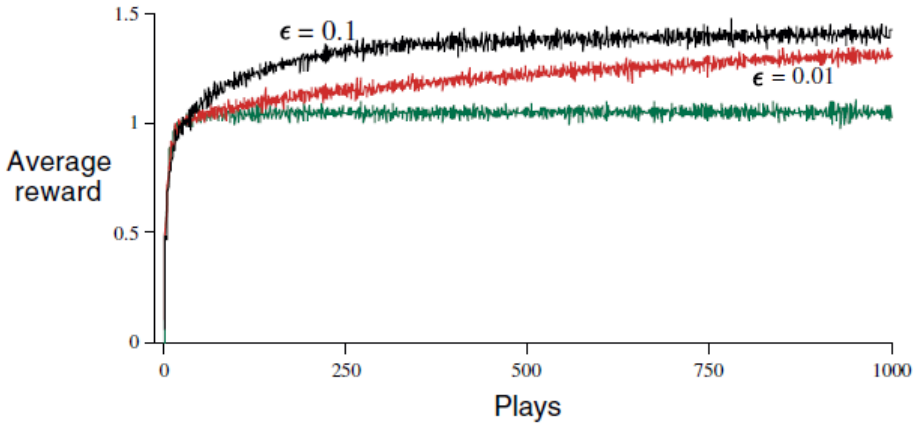
agent can choose the action the return the best reward, which is called greedy action. The choice of the greedy action receive the name of exploitation (Sutton & Barto, 2017).

Exploiting maximize the expected reward in one play, but exploration can produce a greater reward in the long run. This occurs because when the agent chooses the actual greater reward, the other actions are not evaluated to verify if there is a better action to this state. The balancing of these two applications is called exploration-exploitation problem (Sutton & Barto, 2017).

Whether it is better to explore or exploit depends in a complex way on the precise of the values estimates, uncertainties and the number of plays remaining. However, some simplified methods can be applied to overcome this difficulty. A simple alternative is to exploit the most part of the time, but with a small probability ϵ choose a random action in the possible actions. This type of method is named ϵ-greedy (Sutton & Barto, 2017).

Although simple, the ϵ-greedy method permits the convergence of all rewards to the maximum value, as all the action will be tried many times when the number of plays increase. In Figure 10 is presented the average reward of ϵ-greedy with three different ϵ values: 0, 0.01 and 0.1 (Sutton & Barto, 2017).

Figure 10 - Average performance of the ϵ-greedy method.



Source: Sutton and Barto (2017, p. 29).

In Figure 10 is possible to verify that ϵ = 0 increase faster at the very beginning, but levelled off at a lower level, achieved around 70% if the possible reward. This behaviour is waited due to the high possibility to get stuck in suboptimal performance, not exploring better rewards (Sutton & Barto, 2017).

On the other hand, the ϵ = 0.1 show worse rewards at the very beginning but achieve a higher reward due to the high exploration. The ϵ = 0.1 actives earlier the optimal

action, but 10% of the time the agent exploring among the possible actions. The $\epsilon = 0.01$ increase slowly but with more plays could pass the $\epsilon = 0.1$, as long as the $\epsilon = 0.01$ is exploring just 1% of the time. In a more robust algorithm is also possible to reduce the ϵ over time to optimize the ϵ-greedy method (Sutton & Barto, 2017).

2.1.6.5 Markov Decision Process

The environment in time t+1 responds to an action taken at time t usually taking into account all prior states and actions. When the system is Markovian or has a Markov property, the response of the system in time t+1 depends only on the state and action at the time t. In Equation (4) is showed the relation (Wiering & van Otterlo, 2012).

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1},\ldots s_0, a_0) = P(s_{t+1}|s_t,a_t) \quad (4)$$

According to the Equation (4), the probability of having a state s_{t+1} given all the prior states and actions is the same as having a state s_{t+1} given just the last action and state. It means that the last state and action have all the important information to know which new state the system will active applying a determined action in the actual state (Wiering & van Otterlo, 2012).

Another important way to interpret Equation (4), is that taking an action in the same state, the probability distribution to active the next state is the same at all the times that the system makes this decision. This property makes the learning process possible (Sutton & Barto, 2017).

The Markov Decision Process (MDP) also uses the reward function defined as $R(s_t)$, which the reward is given based on the actual state. Depending on the used algorithm the reward can be defined as $R(s_t, a_t)$, which is given based on the actual state and action or $R(s_t, a_t, s_{t+1})$ which depends on state transition. The last one is usually implemented in model-free algorithms, permitting to correlate start and resulting states (Sutton & Barto, 2017).

2.1.6.6 Value Functions

The value function estimates how good is to the agent to be in a given state or taken an action in a given state. This “how good” notion is defined in terms of future

rewards that can be expected and consequently depends on the future states and actions. In this way, being the policy π the mapping from state to actions, the value function V^{π} takes into account that starting in a taken state the agent will follow the policy π thereafter (Sutton & Barto, 2017).

A state-value function for policy π of an MDP can be defined in terms of the so-called Bellman Equation as present in Equation (5) (Wiering & van Otterlo, 2012).

$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s') \left(r(s, a, s') + \gamma V^{\pi}(s') \right) \tag{5}$$

In Equation (5) the term $T(s,\pi(s),s')$ correspond to the probability of transition from the state s_t to the state s' or s_{t+1} and the term γ is the discount factor that permits to operate in infinity horizons. The discount factor is always between 0 and 1 and makes that the latest rewards obtained, have a more significant influence than rewards obtained in the past. The greater the γ more important are the older values, in the case that the $\gamma = 0$ the agent is called myopic and just take into account the most recent reward (Wiering & van Otterlo, 2012).

Evaluating the Equation (5), it denotes that the expected value of a state is defined in terms of the immediate reward and values of possible next states weighed by the transition probabilities and a discount factor (Wiering & van Otterlo, 2012).

In the same way, it is possible to define a state-action-value Q function that follows the policy π as shown in Equation (6) (Wiering & van Otterlo, 2012).

$$Q^{\pi}(s, a) = \sum_{s'} T(s, \pi(s), s') \left(r(s, a, s') + \gamma Q^{\pi}(s', a') \right) \tag{6}$$

The goal of the RL algorithm is to find the best policy, which receives more rewards. The optimal policy is named π^* and occurs when the optimal value function $V^*(s)$ is bigger or equal than the policy value function $V^{\pi}(s)$ to all the states and policies. The optimal state-value function is given by Equation (7) (Wiering & van Otterlo, 2012).

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') \left(r(s, a, s') + \gamma V^*(s') \right) \tag{7}$$

The Equation (7) is named Bellman optimality equation and present that the value of a state under the optimal policy is equal to the expected return for the best action in that state. In this case, Equation (8) can be applied (Wiering & van Otterlo, 2012).

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s, a, s') \left(r(s, a, s') + \gamma V^*(s) \right) \tag{8}$$

The policy of the Equation (8) is called greedy policy because it greedily selects the best action using the state-value function. In the same way that the Equation (7) the optimal action-state-value function can be obtained as showed in the Equation (9) (Wiering & van Otterlo, 2012).

$$Q^*(s, a) = \sum_{s'} T(s, a, s') \left(r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right) \tag{9}$$

To find the optimal policy, the action-state-value function no needs the transition function and no forward-reasoning step is needed. This is an advantage of the Q-functions instead of V-functions because Q-function makes the learning process of model-free systems easier. The optimal action selection can be present as in Equation (10) (Wiering & van Otterlo, 2012).

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} Q^*(s, a) \tag{10}$$

Based on Equation (10) it is possible to define a greedy policy related to the Q-function. Different from the greedy policy related to the V-value in the Q based there is no need to consult the model of the MDP (Wiering & van Otterlo, 2012).

2.1.6.7 Model-free

As opposed to the Dynamic Programming models where a model of the environment is necessary, the RL uses model-free algorithms that permit to obtain the policy from sampling and exploration. The RL does not present a priori known of transition and reward models, what makes necessary to explore the environment by doing

actions and observing the states to estimate the state-action value functions (Wiering & van Otterlo, 2012).

However, most of the model-free methods are focused on direct estimation of action values. In this approach transition and reward, functions cannot appear in the update rules. In this case, the action is based on the actual state and on the value function Q or V. These RL algorithms are called temporal difference learning and Q-learning is one of the examples of this algorithm (Wiering & van Otterlo, 2012).

2.1.6.8 Q-Learning

Christofer Watkins developed the Q-learning method in 1989 as an algorithm that does not need a model of the system (Watkins, 1989). The algorithm converges when evaluating a discrete case, with a finite number of actions and states (Harmon, et al., 1996).

The Q-learning algorithm was applied with success in a differential game in 1996. The game evaluated a strategy of aeroplane and missile simulation, which the aeroplane avoid the missile and the missile pursues the aeroplane. Applying the Q-learning combined with residual-gradient technic, the authors achieve an excellent result, where the aeroplane and the missile learn to achieve low levels of reward in a moment to increase significantly the reward during the next steps (Harmon, et al., 1996).

The Q-learning strategy uses the Q-value improvement to learn from a sequence of experiences in which every action is tried (Harmon, et al., 1996). In the Q-learning, the value function is changed incrementally after each state transition based on the rewards obtained by the last state-action combination. Initially, the Q-values are set randomly and updated to achieve the optimal Q-value by Equation (11). This Equation is derived from the Bellman Equation (Harmon, et al., 1996).

$$Q(s,a)=\left(r(s,a)+\gamma^{\Delta t}\max_{a} Q(s',a)\right) \tag{11}$$

The unique solution to Equation (11) is the optimal Q-function. The policies implied by the optimal Q-function return the best action for each state, permitting to find the optimal actions to the system (Harmon, et al., 1996).

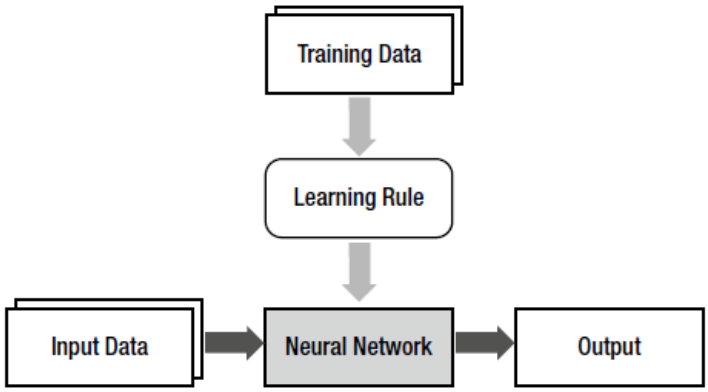
The complexity of one interaction of the Equation (11) does not depend on the number of states. However, when the Q-values are saved in a looked up table the

complexity depends linearly with the number of actions, due to the time to find the action that returns the maximum Q-value. The Q-learning algorithm also presents a restriction with short time steps, which increase significantly the time required to train the bigger number of data. In continuous-time, it is impossible to work with Q-learning based algorithms (Harmon, et al., 1996).

2.1.6.9 Neural Network in Reinforcement Learning

To overcome the number of states restriction, the neural networks may be applied. In this case, a neural network replaces the machine learning model as shown in Figure 11. The training data is replaced by a learning rule that uses already explored data to create the correct policy between input and output data (Kim, 2017).

Figure 11 - Relation between machine learning and neural network.

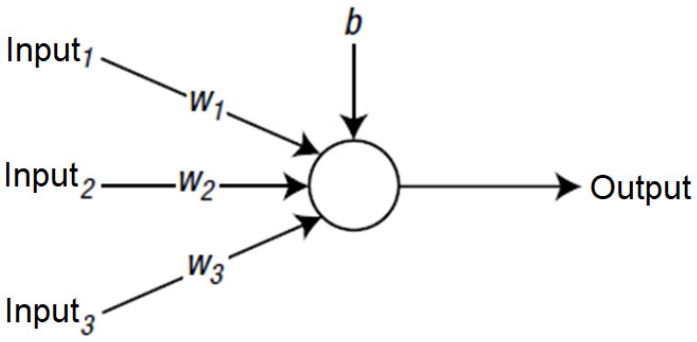


Source: Kim (2017, p. 12).

2.1.6.9.1 Neural network

The neural network mimics the mechanism of the brain, using a network connection of nodes in the place of neurons and the connection between the “neurons” is made by weight values. Figure 12 represents one node of the neural network (Wiering & van Otterlo, 2012).

Figure 12 - Node, three inputs, bias and output.



Source: Kim (2017, p. 20).

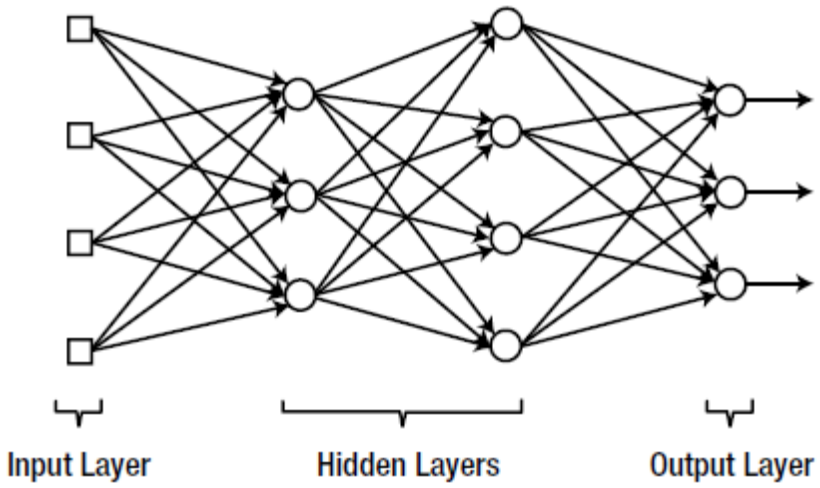
The node receives three inputs that are multiplied by the weights (w_1, w_2, w_3) respectively. Lastly, in the sum of the products is added the bias b that also contributed to the final output result. Using this methodology the information on the neural network is saved in weights and bias, permitting the input-output relation similar to an equation (Kim, 2017).

Before the final output, an activation function determinate the behaviour of the node, usually including a non-linearity. The most common activation equation is the sigmoid function presents in Equation (12). This function guarantees that all the outputs maintain their values between zero and one (Kim, 2017).

$$\varphi(x)=\frac{1}{1+e^{-}} \tag{12}$$

Like the brain, the neural network is composed of a large number of nodes connected to each other as present in Figure 13. The squared nodes represent the input ones, which do not compute any calculation, working just as passage nodes. The group of rightmost nodes are called output layer and from the output of these nodes are obtained the results of the neural network. The nodes between input and output are named hidden layers. These nodes are not accessible from the outside of the network (Kim, 2017).

Figure 13 - Layered structure of nodes.



Source: Kim (2017, p. 22).

The neural network can be classified according to the number of hidden layers. The first developed network had just input and output layers, which receive the name of single-layer neural network. This type of network has a limitation about the data that is possible to evaluate and learn due to the low number of layers. In the case of complex classification rules, the number of layers needs to be increased (Kim, 2017).

When hidden layers are applied the network receives the name of multi-layered network. The multi-layered network that has just one hidden layer is designated shallow or vanilla neural network. On the other hand, the multi-layered networks with more than two hidden layers are called deep neural network (Kim, 2017).

The correct weights and bias are learned in the network by data containing known inputs and outputs. Using the error between the output calculated by the network and the expected output, the weights and bias are updated until obtained a corrected relation between input-output (Kim, 2017). The update is realized based on Equation (13) (Kim, 2017).

$$w_{ij} = w_{ij} + \beta \, \varphi_i'(\acute{\eta}_i) \, e_i \, x_j \tag{13}$$

Where w_{ij} is the weight between the output of the node i and input in the node j, β is the learning rate, which is between 0 and 1 and determines how fast the weight is updated, $\varphi_i'(\acute{\eta}_i)$ is the derivative of the activation function φ of the output node i evaluated at the weighted sum of the output node i, e_i is the error of the output node i and x_j is the output from the input node j (Kim, 2017).

The learning process may be realized in three different ways, Stochastic Gradient Descent (SGD), batch and minibatch (Kim, 2017):

- The SGD calculates the error for each training data and adjusts the weight immediately.
- The batch method use averages of errors based on all training data, updating the error just once. The possible large number of data evaluated to update the weight make the batch slower than SGD, but the use of average values permit a more stable learning process.
- The minibatch is a mix between SGD and batch. In this approach, the algorithm selects out an arbitrary part off the total training data, permitting to increase the velocity when a large training data is used (Kim, 2017).

The learning process in single-layer networks is easily realized using one of the mentioned methods. However, in multi-layered networks, the errors of the hidden layers cannot be calculated because the expected results are not available. In this way, the use of the backwards propagation method needs to be applied (Kim, 2017).

Usually, the data travel in the network from the input to the output, what is called forward propagation. Introduced in 1986 (Rumelhart, et al., 1986), the backpropagation consist of making the "travel" of the final error from the output value in the direction of the input value. The layer transition is made using the same weights of the forward propagation, and the error in each hidden node is updated using the same relations of the output error calculation.

*2.1.6.9.2 N*eural Fitted Q Iteration

The Neural Fitted Q Interaction (NFQ) use a network to update the Q value, based on the Bellman's equation. In this case, the network starts with random weights $Q(s, a, w_0)$, where w_0 are the initial weights. Alternating between exploration and neural update, the Q-values at the k^{th} interaction is updated towards the target value as shown in Equation (14) (Francois-Lavet, et al., 2018).

$$Y_k^Q. = r(s,a,s') + \gamma \max_{a'} Q^*(s',a',w_k) \tag{14}$$

The term γ is the discount factor that permits to operate in infinity horizons, r is the reward function and Q^* is calculated based on the last Q network available.

The network is updated using stochastic gradient descent to minimize the square loss between the network value Q(s, a, w_k) and the target value Y_k^Q. Thus, the weights upgrade is given by the Equation (15) where β is the learning rate. (Francois-Lavet, et al., 2018) (Riedmiller, 2005).

$$w_{k+1} = w_k + \beta \left[Y_k^Q - Q(s, a, w_k) \right] \nabla_{w_k} Q(s, a, w_k) \quad (15)$$

Based on Equation (15), the weights are updated by the last weight plus the multiplication of the error and the gradient of the Q value. The learning rate β is applied to permit the programmer to control the rate of change in the network weights (Francois-Lavet, et al., 2018).

The very straight-forward application of NFQ and the possibility of work in continuous states makes the algorithm widely applicable in many areas. The use in control tasks is a very interesting example since the NFQ permit the autonomously learn of near-optimal controls. Besides that, the use of batch has contributed to a major use of NFQ in control area due to the data efficiency (Wiering & van Otterlo, 2012).

However, the NFQ as the , the network update may propagate errors and present a slow convergence or no convergence as showed experimentally by Riedmiller (2005). Due to the instability of the algorithm, specific care has to be taken to obtain good results (Francois-Lavet, et al., 2018). To overcome these convergence restrictions, other algorithms were developed, as for example Deep Q Network.

When the neural fitted Q iteration uses a deep network it is called deep fitted Q interaction (DFQ), a method inside the Deep RL area. With this strategy, application which require more computational power can be executed just with one single network. For example, in this case it is possible to control more than one policy or learn directly from images using just one network (Wiering & van Otterlo, 2012).

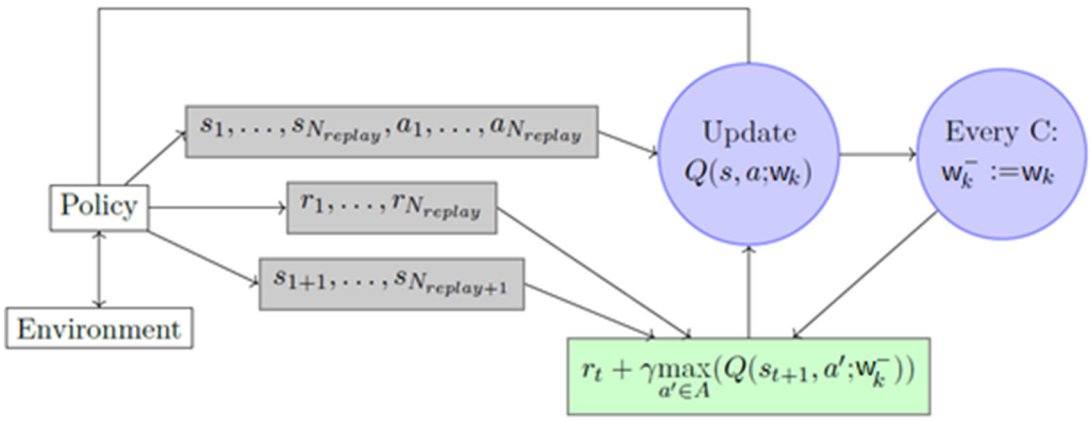
2.1.6.9.3 Deep Q Network

Improving the NFQ ideas, the Deep Q Network (DQN), was proposed by Mnih, et al. (2013) and Mnih, et al. (2015). The application was related to game playing, especially due to the algorithm possibility of learns to play a large number of different games using the same architecture. Based on the input raw pixels, the algorithm was able

to learn how to measure and obtain rewards, applying a variant of the Q-Learning algorithm.

In DQN the target Q-network in Equation (14) is replaced by Q(s', a'; w_k^-) where the weights w_k^- are updated only after a specific number of iterations c, avoiding the risk of divergences. Using a replay memory, all the explored data, collected by a ϵ-greedy policy in N-replays are saved. The update is made using a mini-batch containing data from all replays. (Francois-Lavet, et al., 2018) (Mnih, et al., 2015). In Figure 14 is shown a sketch of the DQN algorithm.

Figure 14 - DQN algorithm sketch.



Source: François-Lavet et al. (2018, p. 29).

Another important heuristic to DQN keeps the rewards between -1 and 1 which limits the generation of errors. The use of normalizes inputs from -1 to 1 also makes it easier to operate different magnitudes in the neural network without converging problems.

2.1.6.10 Reinforcement Learning in Continuous State-Action

Deep Q-learning algorithms permit the evaluation of continuous states spaces but still work with discrete action states. In the real world, the use of discrete action maybe not enough to control a complete system. In this way, the use of algorithms that can choose actions from an infinitely large action range is a good opportunity of improvement (El Sallab, et al., 2017) (Wiering & van Otterlo, 2012).

In problems of control, the aim is an approximation of the optimal policy π^* that maximizes the value function Q^* for each state. Therefore, the π^* estimation may be made

directly or, based in Q^*estimation. This relation leads to three general methodologies of solution (Wiering & van Otterlo, 2012):

- Model approximation: This kind of algorithms approximate the MDP and compute the desired policy on this approximate MDP. Based on s, a, γ the algorithm may learn the T and r to construct π^* and Q^*. However, the extraction of all states is usually very difficult or impossible.
- Policy approximation: Policy-approximation algorithms store a policy directly and update this policy to found the optimization. This type of algorithm is also called direct policy-search or actor-only algorithms.
- Value approximation: In value-approximation algorithms, the actions, states and rewards are used to directly update a value function. Many algorithms use this procedure, including on-line, off-line, on-policy and off-policy algorithms as SARSA, Q-Learning and DQN.

2.1.6.10.1 Stochastic Policy Gradient

Policy gradient is a very common continuous action reinforcement-learning algorithm based on policy approximation. (Francois-Lavet, et al., 2018).

In this algorithm, the policy gradient method updates the policy parameters ψ in the direction of the greater performance measure of the corresponded policy ρ^{π}. The performance measure may be for example an average reward per step. The application of this method permits to find a locally optimal policy based on the maximum performance objective J(π). Equation (16) shows the gradient calculation (Francois-Lavet, et al., 2018) (Silver, et al., 2014) (Sutton, et al., 1999).

$$\begin{aligned}\nabla_{\psi} J(\pi_{\psi}) &= \int_{S}^{\cdot} \rho^{\pi} \int_{A}^{\cdot} \nabla_{\psi} \pi_{\psi}(a|s) Q^{\pi}(s,a) da ds \\ &= E_{s\sim\rho^{\pi},\ a\sim\psi}\left[\nabla_{\psi} \log \pi_{\psi}(a|s) Q^{\pi}(s,a)\right]\end{aligned} \tag{16}$$

The state-action value function $Q^{\pi}(s,a)$ in the Equation (16) needs to be estimated. One approach is to use the actual rewards as an approximation as show Equation (17) and Equation (18) (Sutton, et al., 1999).

$$\rho(\pi) = E\left\{\sum_{t=1}^{\infty} \gamma^{t-1} r_t \,\middle|\, s_0, \pi\right\} \tag{17}$$

$$Q^{\pi}(s,a) = E\left\{\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \,\middle|\, s_t = s, a_t = a, \pi\right\} \tag{18}$$

This leads to the function gradient estimator known as REINFORCE algorithm (Francois-Lavet, et al., 2018) (Sutton, et al., 1999) (Williams, 1992). The use of this algorithm permits network implementation as the policy function. In the network, the weights represent the policy parameters ψ, In this algorithm, the learning process uses as input the states and as output the action distribution (Sutton, et al., 1999).

2.1.6.10.2 Deterministic Policy Gradient (DPG)

The majority of model-free reinforcement learning methods are based on generalized policy interaction, interleaving policy evaluation and policy improvement. While the evaluation estimates Q, policy improvement uses the estimated Q to improve the policy. Considering π as a deterministic policy, the most common approach to evaluation is the greedy maximization of Q as shown in the Equation (19) (Silver, et al., 2014).

$$\pi^{k+1}(s) = \underset{a}{\operatorname{argmax}}\, Q^{\pi^k}(s,a) \tag{19}$$

In continuous action-spaces is computationally difficult to solve Equation (19), but is possible to move the policy in the direction of the gradient of the Q value (Silver, et al., 2014).

Based on this, the DPG is underpinned in Stochastic Policy Gradient, but the transition between state and action is directly given, without a probabilistic distribution. This methodology permits that DPG evaluates the integration only on the state space as present in the Equation (20). On the other hand, the stochastic police gradient needs the state and action integration, which increases the implementation challenge (Silver, et al., 2014).

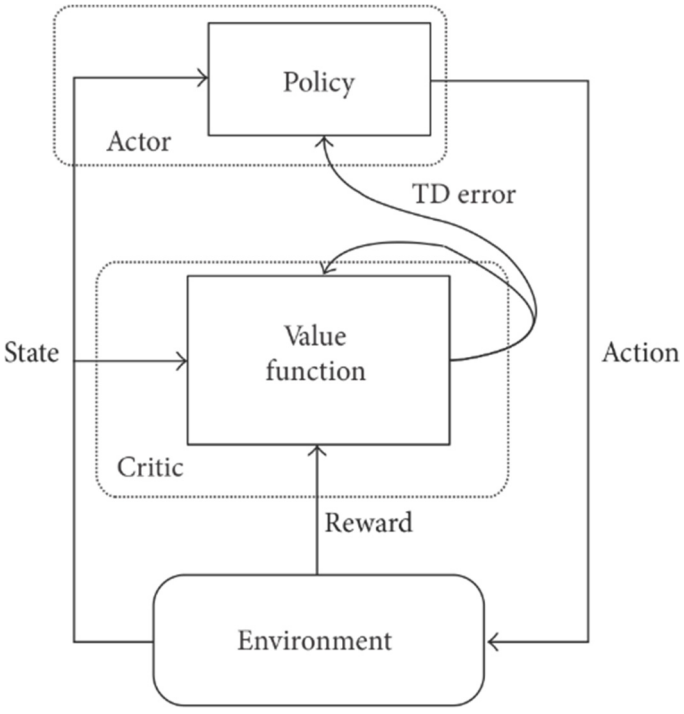
$$\begin{aligned} \nabla_{\psi} J(\pi_{\psi}) &= \int_{S} \rho^{\pi}(s)\, \nabla_{\psi} \pi_{\psi}(s)\, \nabla_{a} Q^{\psi}(s,a)\big|_{a=\pi_0(s)} ds \\ &= E_{s\sim\rho^{\pi}}\left[\nabla_{\psi} \pi_{\psi}(s)\, \nabla_{a} Q^{\pi}(s,a)\big|_{a=\pi_0(s)}\right] \end{aligned} \tag{20}$$

This modification permits a reduction in the data quantity since it is not necessary to evaluate a large number of actions. However, to ensure that DPG has a satisfactory exploration it is necessary to use off-policy methods. In this instance, the actions are chosen according to a stochastic behaviour policy, but the learning process uses a deterministic target policy (Silver, et al., 2014).

2.1.6.11 Actor-Critic algorithms

The Actor-Critic algorithm is a combination of policy approximation and value function approximation. This method uses an explicit approximation of the state value function V^{π} or action-state value function Q^{π} to obtain the policy approximation. This value function approximation is called critic. The policy that generates the relation state-action based on value function optimization is called actor (Francois-Lavet, et al., 2018). Figure 15 presents the actor-critic algorithm architecture (Uc-Cetina, 2013).

Figure 15 - The actor-critic architecture.



Source: Uc-Cetina (2013, p. 3).

The earliest actor-critic algorithm had its value function based on the TD (0) algorithm, permitting to obtain the TD-error δ_t by the Equation (21) (Wiering & van Otterlo, 2012).

$$\delta_t = r+\gamma V(s') - V_t(s) \tag{21}$$

Applying the Equation (21), the critic analyses the selection of the last action in the last state, evaluating its strengthen or weaken. A preference for an action in a specific state is given by the Equation (22), where p is the preference and ξ is the update size (Wiering & van Otterlo, 2012).

$$p(s_t,a_t) \leftarrow p(s_t,a_t)+\xi\delta_t \tag{22}$$

The first advantage of the actor-critic algorithm is the possibility to work with a large number of actions since there is no need to consider all actions to select one of them. The second advantage is the possibility to learn stochastic policies from scratch, permitting to apply the same algorithm to different physical systems (Wiering & van Otterlo, 2012). Due to this, the actor-critic algorithm show successful applications in the control area, as robot movement (Wang, et al., 2015), adaptive cruise control (Zhao, et al., 2013) (Zhao, et al., 2017) and autonomous driving (Jaritz, et al., 2018) (El Sallab, et al., 2016).

Several new versions of the actor-critic algorithms were developed based on Q-learning (Silver, et al., 2014), DQN (El Sallab, et al., 2017), SARSA (Uc-Cetina, 2013), NFQ (Radac & Precup, 2018) (Hafner & Riedmiller, 2011), among others specific value approximations (Jaritz, et al., 2018) (Wiering & van Otterlo, 2012) (Francois-Lavet, et al., 2018) (Zhao, et al., 2017).

2.1.6.11.1 Deep Actor-Critic

The use of the DPG as policy approximation in actor-critic algorithms, permit the use in off-policy applications when the Q-learning algorithm is used as value function approximation. This combination permit efficient model-free learning of different tasks as board game playing or arm robot controller (Silver, et al., 2014).

To take all the advantages of the NFQ and DQN algorithms, combined with the possibility of continuous actions the Neural Fitted Q Interaction with Continuous Actions (NFQCA) (Hafner & Riedmiller, 2011) and the Deep Deterministic Policy Gradient (DDPG) (Lillicrap, et al., 2016) algorithms were developed.

The NFQCA uses two neural networks to control the policy and the value function. Using the off-policy method all the collect data is saved in a dataset with the inputs (s, a, s'). The policy is represented by a neural policy function, $\pi(s)$, with weights w_π. Considering that the policy represents the Q greedy of the state-action value function, the Equation (23) may be formulated (Hafner & Riedmiller, 2011).

$$Q_{k+1}(s,a) = r(s,a) + Q_k\left(s',\pi_k(s')\right) \tag{23}$$

The Equation (23) is computed for each transition sample of the dataset. Employing the batch approach, the weights of the critic network are updated using backpropagation of the Q error. Using the already learned critic the actor network has its weights updated by the gradient of the Q-value, aiming to find the optimum action. (Hafner & Riedmiller, 2011).

The application of NFQCA exhibit good control possibilities in simulation and real application. The RL algorithm present results similar to optimal controllers, with fast learning time and the possibility to control continuous and non-linear systems (Hafner & Riedmiller, 2011).

The more recent Deep Deterministic Policy Gradient (DDPG) has a similar approach; using DQN instead of NFQ to found Q_π. Based on this, a second deep network is used to found the policy π (Lillicrap, et al., 2016).

The DDPG has a similar architecture to the NFQCA but implements some upgrades that guarantee better behaviour on the learning process. To avoid the instability during the networks update the algorithm uses copies of the critic and actor networks on the calculations to smoothly update the final networks (Lillicrap, et al., 2016).

The copies are designated as Q' and π' to the critic and actor respectively. At the end of the calculations, the copies are updated by the slow target network ζ given by the Equation (24) to critic and Equation (25) to actor weights (Lillicrap, et al., 2016).

$$w^{Q'} \leftarrow \zeta w^{Q} + (1-\zeta)\, w^{Q'} \tag{24}$$

$$w^{\pi'} \leftarrow \zeta w^{\pi} + (1-\zeta)\, w^{\pi'} \quad (25)$$

The networks copies were already proposed in the DQN algorithms but without the soft update of the weights. In DQN the weights were directly changed (Mnih, et al., 2013).

Another important improvement already present in DQN is the use of mini-batch that permit to evaluate a large number of data on the replay memory without a significant loss of performance (Mnih, et al., 2013) (Lillicrap, et al., 2016).

To improve the learning time, normalization of the collect data was also applied as already proposed by Ioffe and Szegedy (2015). When working with low dimensional feature vectors the large difference between physical units can difficult the learning efficiency in the network. The use of batch normalization normalizes each dimension across samples in minibatch to have unit mean and variance (Lillicrap, et al., 2016).

The DDPG implementation presented a good result in learning and controlling diverse tasks. The algorithm was able to actuate using low-dimensional feature vectors and high-dimensional pixel inputs, due to the use of deep networks. The DDPG presented better results than the NFQCA mainly due to the use of target networks, which improve considerably the control performance (Lillicrap, et al., 2016).

2.1.7 Design of Experiments

According to Montgomery (2009a) the implementation of designed experiments is a powerful methodology to verify the influence of the inputs of a system on its outputs. The evaluation occurs by the analyses of the outputs according to a controlled variation of the inputs. The Design of Experiments (DOE) represents the guidelines and standards to be followed during the execution of the tests.

Before the execution of the experiment, a planning phase include the choice of factors and levels, where factor represents each one of the inputs that will be variated and the levels indicate the magnitudes of the inputs that will be evaluated. When a significant number of factors is applied, a factorial design can be applied. In this case, all factors variate together permitting the evaluation of all possible combinations of factor levels. (Montgomery, 2009a).

The effect of a factor is measured by the response change due to a change in the levels of the factor. When just two levels are applied to a factor, they are usually named

high (+) and low (-) levels. In this case, the effect is calculated by the difference of the average from all results with low levels and the average of all results with high levels as presented in Equation (26) where A represents the effect of a factor, $\bar{y}_{A+}$ is the average of all results where the level of the factor is high and $\bar{y}_{A-}$ is the average of all results which have low levels of the factor (Montgomery, 2009a).

$$A = \bar{y}_{A+} - \bar{y}_{A-} \tag{26}$$

To deal with the replications of the experiment and the statistical validation of the results, the analyses of variance (ANOVA) is applied. Is this case the hypotheses of no significant effect from each factor is tested with a level of significance α_v. In this way, the effect of the factor just is considered significant when it can be confirmed with more than 100 *(1- α_v) % of certain according to the available data. So, in case that the data present high variability or a small quantity of data is collected, the effect cannot be statistically confirmed (Montgomery, 2009a).

In statistical tools, the significance of each factor effect on the results is graphically plotted in a Pareto graph. In this graph is plotted the correlation of the calculated t-value for each factor correlating it with a red line, which is drawn on the quantile of a t-distribution with degrees of freedom (DOF) equal to the DOF of the DOE error. The calculation of the DOF considers the number of levels, number of factors and number of evaluated interactions of the experiment. In the final plot the factor which advances the red line shows significant influence on the results (Minitab, 2019).

With the corresponding tools and methodologies, complex analysis of the system can be realized. However, to preliminary studies, it is very common the application of the 2^k factor study. In this methodology, each factor applies two levels and a k number of factors can be evaluated. This experiment permits to variate all factor together, reducing the number of tests to find the best solution in a reduced time (Montgomery, 2009b).

2.2 STATE OF THE ART

This sub-section includes the manly researches in the traction control and RL with neural network areas. This sub-topic expose the studies used as base to the development of the present research.

2.2.1 Traction Control

After the proposal of the Traction Control System by Gerstenmeier (1986), the researches are focused mainly in the development of the algorithms that control the system. As an example, Engbom, et al. (2004) overcame the limitations of tuning the controller to new vehicles, implementing the Internal Model Control (IMC). The benefit of this type of controller is the changeability of the control between vehicles, adjusting measurable vehicle parameters in the controller.

More recently Kirchner and Southward (2013) applied an adaptive gradient ascent algorithm to find the best controller. In this research, the author also evaluates new possibilities of sensors, taking the vehicle accelerations as possible input data to the controller. The results present the possibility of implement this type of algorithm with significant advantages when compared with the traditional ones. The use of acceleration as input shows better results in situations where the performance is the main objective while using the traditional input data prevents excessive tire slip.

Jin, et al. (2018) presented the possibility of optimizing a common Proportional, Integral and Derivative (PID) controllers using the Ant Colony Optimization (ACO) algorithm. This combination permits to found an optimized PID controller using statistical algorithms, based on ants behaviour. The results present a good possibility of improving in the ASC when compared with the normal PID controllers.

Due to the easier controller of electrical motor, the torque available in low speeds and the tendencies of electrical vehicles the recent researches are focused mainly on electrical motors traction control. However the early researches in traction control of electric vehicles were from the seventies. Using simple DC motors, Bose & Steigerwald (1978) emphasizes the necessity of smooth acceleration of the vehicle and the possibility of vehicle boost controlling the thermal range of the motor.

Twenty years after, Hori, et al. (1998) realized new researches using DC motors now implemented in prototype vehicles. The main emphasis was the quick control due to the fast response of torque in electrical vehicles, showing a possible advantage against the combustion engines.

More recently, many controllers types have been applied usually based on the wheel velocity and motor data (Lee & Tomizuka, 2003), (Khatun, et al., 2003), (Sampaio, et al., 2012), (Hu, et al., 2012), (Gasbaoui, et al., 2017). The main strategy applied is the Fuzzy controller (Lee & Tomizuka, 2003), (Khatun, et al., 2003), but it is also possible

to find a large number of other strategies as PID, H-infinity (Sampaio, et al., 2012) and some specific controllers to traction control as Maximum Transmissible Torque Estimation (MTTE) (Hu, et al., 2012) and Direct torque control based on space vector modulation (SVM-DTC) (Gasbaoui, et al., 2017).

The specific controllers usually are developed to solve problems found in the conventional controllers. For example, MTTE was implemented by Hu, et al. (2012) to avoid vehicle velocity evaluation, parameter that is difficult to be appropriately measured, mainly in 4WD vehicles. According to the author in vehicles where the traction occurs in all wheels, the data is collected by accelerometers, causing deviations and errors.

The use of electrical motors also brought the possibility of control the wheel individually without brakes system actuation when the vehicle has one electric motor for each wheel (Hu, et al., 2012), (Gasbaoui, et al., 2017). This characteristic permits more control freedom on the vehicle behaviour, in curves or wheel actuating in different friction coefficients.

Some author as Sampaio, et al. (2012), Kirchner & Southward (2013) and Jin, et al. (2018) starts to use optimization techniques to find the best performance of the Anti-Slip controllers. The successful application of Artificial Intelligence (AI)-based neuro-fuzzy controllers (Sampaio, et al., 2012), controller optimization using statistics algorithms as ant colony (Jin, et al., 2018) and adaptive gradient ascent algorithm implementation (Kirchner & Southward, 2013) show a possible application of Machine Learning to the control of wheel slip.

2.2.2 Neural Reinforcement Learning

Reinforment Learning algorithms combined with Neural Networks are applied with success in many complex tasks. Riedmiller, et al. (2009) implement NQF controllers for soccer-playing robots, which participate in a RoboCup competition. In this competition, the robot is able to evaluate the environment by image processing, realize dribbles against the other competitors and intercept the ball during the game.

With the application of Deep Neural Networks with a DQN algorithm, Taitler and Shimkin (2017) controlled an air hockey striking. After the learning process the RL algorithm was able to make points based on the first instruction of strike the disk and rewards by achived goals. The system was also able to learn how to use the table rebound angle, winning a human player.

In the automotive area, the Neural RL algorithms present a good possibility of success in applications related to autonomous driven and control improving. Riedmiller et al. (2007), still using an NFQ, controlled a vehicle steering, obtaining a good result in the vehicle behaviour. The vehicle was able to learn the correct steering in 20 minutes of driving

In the same way, de Amaral et al. (2018) developed a torque-vectoring algorithm using deep RL algorithm. The policy was learned in CarMaker environment, which permits to simulate the vehicle environment. The controller showed a good ability to avoid dangerous manoeuvres, however, the authors highlight the great time take to learn and control, what could difficult the real implementation.

The use of DQN on automation drive was applied in a simulated environment using game engines. The possibility of learning in this kind of environment permit assimilate the policy from scratch in a safe way. Good results were presented using an open-source Race simulator, where the vehicle learn to keep the central area of the street, based on a deep RL algorithm (El Sallab, et al., 2016), (El Sallab, et al., 2017).

The combination of Actor and Critic Neural Networks presents excellent performance in some automotive applications, highlighting Zhao, at al. (2017) adaptive cruise control research. Using an actor-critic algorithm named model-free optimal control (MFOC), was possible to control the gas pedal position and brake pressure of the vehicle. The MFOC is based on Q-learning rules with two networks to criticize and controlling of the system respectively. The actor and the critic were trained alternatively until convergence. The converged actor network was used to control the vehicle in a hardware in the loop simulation. The results present a better control behaviour than the widely used PID controller (Zhao, et al., 2017).

The DDPG method proposed by Lallicrap, at al. (2016) was also evaluated in automotive area in Torcs simulated environment. Torcs permits the vehicle control based on video input or on the vehicle physical parameter as position, velocity, steer and acceleration. The results presented good behaviour of the vehicle mainly when physical parameters were used, showing a possible use of DDPG in autonomous driving.

The use of Actor-Critic controller in the Torcs environment was also applied by El Sallab et al. (2016). The authors use an algorithm similar to DDPG, with two networks, based on DQN. The results were also very satisfactory, with the actor-critic being able to control in a smoothly way the vehicle steering (El Sallab, et al., 2016) (El Sallab, et al., 2017)**.**

Unfortunately, few studies test the Actor-Critic controller in real environments. In the automotive area is possible to highlight the Radac and Precup (2018) research that uses an NFQ value function approximation combined with DPG policy approximation to control an ABS system. The tests and data collection was realized in a bench, showed adequate results on slip avoidance. The authors indicate the algorithm as a good opportunity of control to be implemented in the automotive industry, with adequate time cycle to control real time systems (Radac & Precup, 2018).

Despite the good results obtained by Radac and Precup (2018), some authors emphasise the challenging of implement RL algorithms directly in the final hardware (Heim, et al., 2018). A possible option to overcome this difficulty is learning the first policy in simulation and after that tune-up the policy in the real hardware (Wiering & van Otterlo, 2012).

3 METHODOLOGY

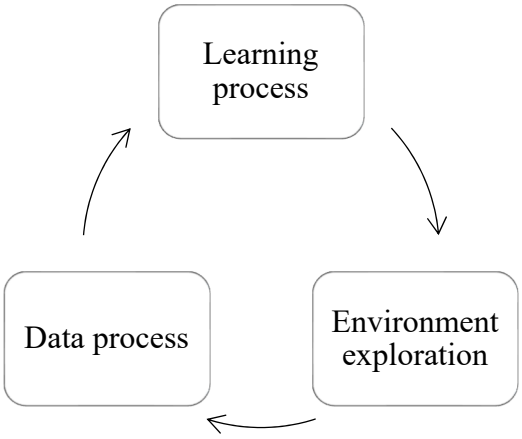
Based on the already made researches, the main objective of this work is evaluating the possibility of implementation of a Deep Actor-Critic Algorithm as a controller in a rear-wheel driven electrical vehicle. The vehicle system behaves as a Markovian Decision problem, which the next state depends only on the actual state-action combination, permitting the RL controller.

The implementation of actor-critic RL algorithms as DDPG and NFQAC usually is given by an iterative learning and exploration principle. In this case, simulated data of the vehicle behaviour is generated and the data saved in the database. This data is treated and used to train the networks in small minibatches.

To controlling application, the trained actor network is used as a controller in the environment, generating new data that is saved together with the old data in the database. All the database is once again used to learn the networks and generate a new controller, keeping the cycle until achieving the desired behaviour.

This implementation may be divided into three main steps that work in a cyclic process as showed in Figure 16:

Figure 16 - Steps of the implementation.



Source: Elaborated by the author.

- Environment exploration: Is the place where the data is collected and the controller evaluated. As proposed by Wiering and Otterlo (2012), an initial simulated environment is used to avoid dangerous behaviour in the real world;

- Data treatment: All the data collected are saved and needed treatment. As proposed by Ioffe and Szegedy (2015) a normalization was realized to make the network learning process easier. During this step, the reward was also calculated;
- Learning process: This step consists of actor and critic networks training by the update of the weights.

The Learning algorithm implemented in the cyclic learning process is based on NFQCA (Hafner & Riedmiller, 2011) and DDPG (Lillicrap, et al., 2016) methods, with some updates to permit the use of MATLAB functions. The vehicle behaviour is evaluated at each step of the learning process, training the controller until no bad behaviours are founded or the maximum number of cycles is achieved.

Despite being long applied in simulations and virtual platforms, the iterative learning process presents a restriction in the applicability in real environments due to the necessity of controller update and new data collection at each learning step. Therefore, a second learning process was proposed which a greater amount of data is simulated or measured without the influence of the controller. The obtained data is treated and the actor and critic networks are directly trained based on these data.

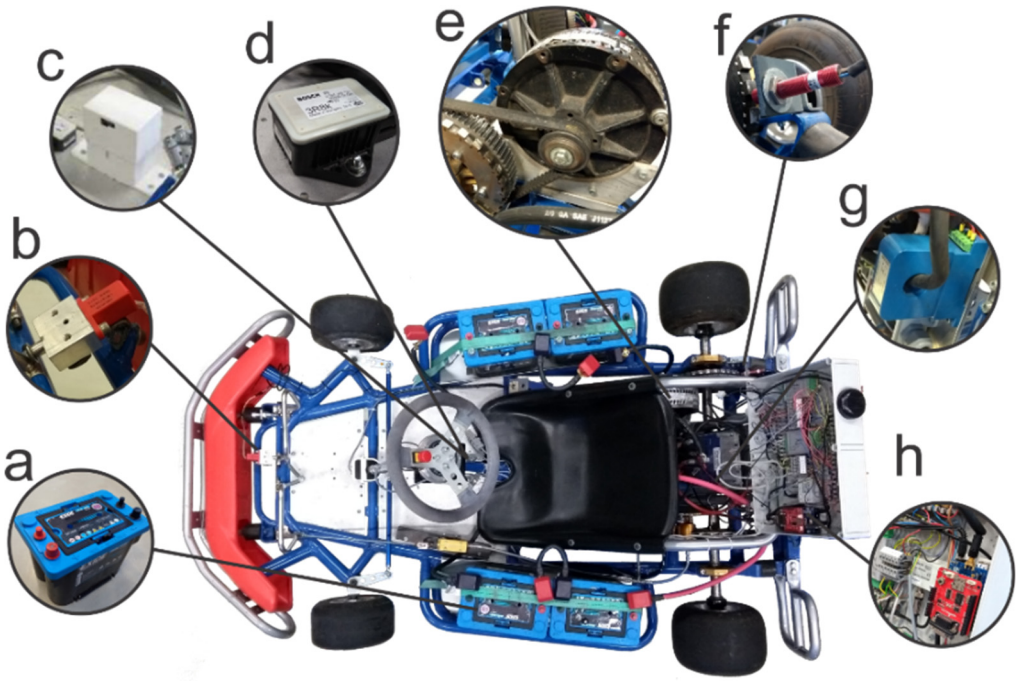
The next sub-sections present each one of the steps that comprise the complete research, including the environment, treatment of the data and learning process in both implemented algorithms.

3.1 EVALUATED VEHICLE

The simulated vehicle is an electrical Go-kart with rear traction. The rear axle is rigid, demanding the same torque/rotation to both wheels, what permit a simplification in the ASC research. Figure 17 shows the vehicle and the main components that influence the research.

The vehicle has a combination of four 12V batteries connected in series to provide 48V that run the electrical powertrain as highlighted in Figure 17 a). The accelerator pedal position is measured by a potentiometer showed in Figure 17 b). This sensor sends a signal with its position to the Millipak 4QPM controller that uses a Pulse-width modulation (PWM) signal to control the motor (Sevcon, 2002).

Figure 17 - Electric Go-kart components.



Source: Elaborated by the author.

The wheel slip angle is given by the difference between vehicle velocity and wheel rotation. In this way, to evaluate the slip in the vehicle tire the real vehicle velocity may be estimated by an optical sensor fixed at the vehicle body (Raoul, et al., 2004).

The sensor showed in Figure 17 c) is an Optical Flow low-cost sensor proposed by Schraufstetter (2018). This sensor permits estimate the real velocity of the vehicle by the calculation of successively recorded images of the ground (Schraufstetter, 2018).

The automotive yaw rate sensor of Bosch Company present in Figure 17 d) measures the vehicle longitudinal acceleration. To measure the wheel rotation a gear tooth speed sensor model GS1005 from Cherry is applied at the rear axle as shown in Figure 17 f). The last data used in the training process is the current of the motor measured by the transducer of LEM, model HTA 400-S, shown in Figure 17 g).

The vehicle powertrain is composed by a 48V Lynch LEM 200-127 brushed DC motor, shown in Figure 17 e), which produces a rated torque of 31.5 Nm and rated power of 8.55 kW or 11.5 hp.

The data of all sensors are sent via CAN bus communication to the vehicle controller, which is connected to a telemetry system. A Holybro V.3 radio module compose the telemetry showed in Figure 17 h).

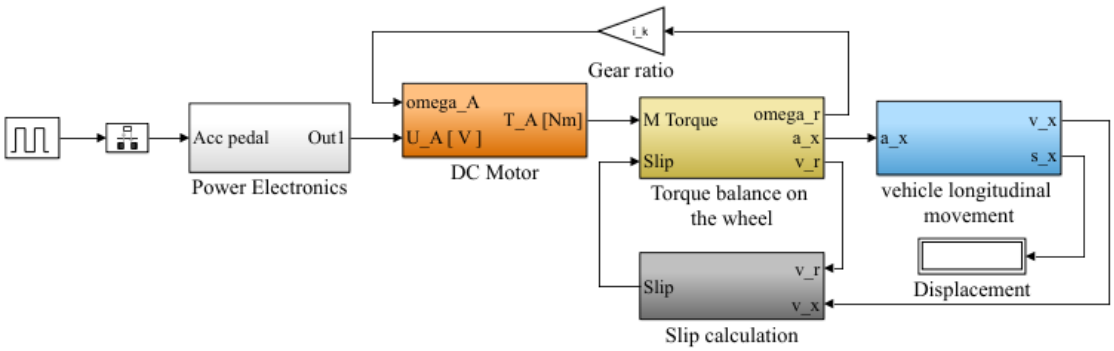
More technical information of the actuators and sensors of the vehicle are available in APPENDIX A. All the available data in the real vehicle is used to create the

generation of data in the simulated environment to the training process. The next subsection presents the simulated environment and the variables save in database.

3.2 SIMULATED ENVIRONMENT

To allow an iterative training implementation a simulated environment is implemented to the execution of data collection and controller evaluation. The simulation consists of a Simulink/Matlab based algorithm that takes into account vehicle longitudinal behaviour, powertrain system and tire behaviour. In Figure 18 is shown the Simulink algorithm.

Figure 18 - Simulated vehicle longitudinal behaviour.



Source: Elaborated by the author.

In Figure 18 omega_A is the motor rotational velocity, U_A is the voltage from the power electronics, T_A and M_torque are the torque generated in the electric motor, Slip represents the slip ratio, a_x, v_x and s_x are respectively longitudinal acceleration, velocity and displacement, v_r is the linear velocity of the surface of the tyre in contact with the rouad and omega_r is the rotational velocity of the wheel. Each component of the environment shown in Figure 18 is explained in more details in the next subsections.

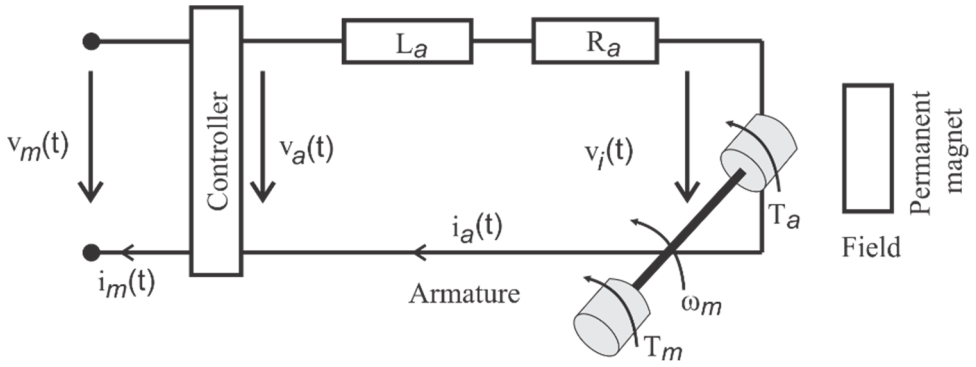
3.2.1 Accelerometer pedal signal and Power Electronics

Initially, a signal generator produces an accelerator pedal signal, which depends on the desired manoeuvre. The signal is sent in a 20 ms time rate, updating the acceleration pedal position in a discrete way as occurs in the real vehicle. The signal is multiplied in the box representing the Power electronics, which change the range from 0-5V to a range of 0 up to 48V.

3.2.2 DC motor

The block representing the DC electrical motor calculates motor current and torque based on the motor velocity and on voltage originated at the power electronics. In Figure 19 is presented the equivalent electric circuit of the DC motor, that is used to implement the code.

Figure 19 - Equivalent circuit of a DC motor.



Source: Adapted of Khajepour, et al. (2014, p. 201).

In Figure 19 $i_a(t)$ is the armature current, $v_i(t)$ is the induced voltage, R_a is the armature resistance, L_a is the armature inductance, $v_a(t)$ is the armature voltage, ω_m is the speed of the motor, T_m is the motor torque and T_a is the torque of the armature. Based on the equivalent circuit the calculation is made by the Kirchhoff's law obtaining the Equation (27) (Khajepour, et al., 2014).

$$v_a(t)=R_a i_a(t)+L_a \frac{di_a(t)}{dt}+v_i(t) \tag{27}$$

Due to the constant magnetic field originating from the permanent magnet a linear relation between ω_m and $v_i(t)$ may be obtained as showed the Equation (28), where K_{gv} is the speed constant (Khajepour, et al., 2014).

$$\omega_m=K_{gv} v_i(t) \tag{28}$$

Solving Equation (27) and Equation (28) where K_{gv} = 54 rpm/V, L_a = 23E-6 H and R_a = 0.0225 Ohm are constants provided by the supplier (LEM, 2019), the motor current is directly obtained.

The linear relation also occurs between $T_a(t)$ and $i_a(t)$ that is given by Equation (29) (Khajepour, et al., 2014).

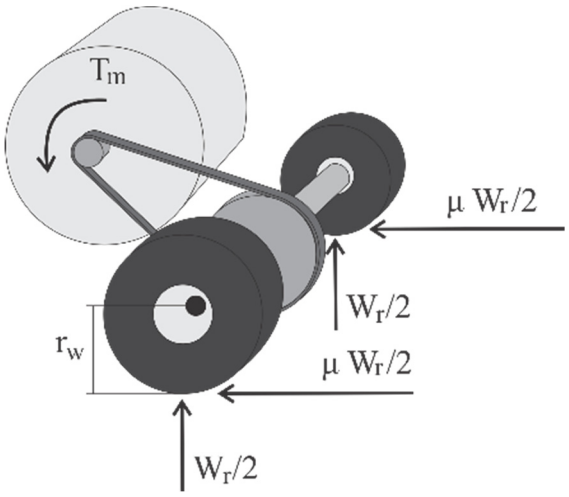
$$T_a(t)=K_{gT}i_a(t) \tag{29}$$

The K_{gT} = 0.15 Nm/A is the torque constant, also provided by the supplier. Therefore, the motor torque may be obtained from the motor current.

3.2.3 Torque Balance on the wheel

The torque balance on the vehicle wheel is based on the torque originating from the motor T_m, the tractive force μW_r and geometric relations of the vehicle as rotational inertia and the dynamic radius of the wheel r_w (Gillespie, 1992). In Figure 20 is shown the balance of forces acting on the wheel and motor.

Figure 20 - Torque balance on the wheel.



Source: Elaborated by the author.

Evaluating Figure 20 and realizing the sum of moments, the torque on the wheel T_a may be obtained by Equation (30) (Gillespie, 1992).

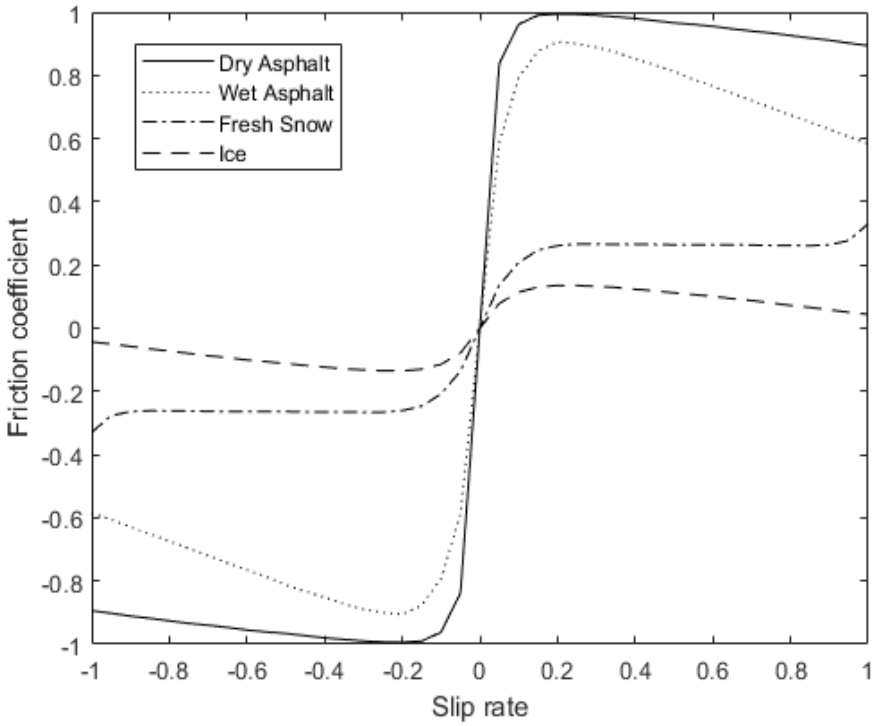
$$T_a = \mu W_r r_w + I_w \alpha_w = (T_m + I_m \alpha_m) i_k \quad (30)$$

In Equation (30) I_w is the rotational inertia of the wheel, calculated with base in the axle and wheel dimensions and mass. The rotational inertia of the motor $I_m = 0.0236$ is provided by the manufacturer (LEM, 2019), ik is the gear ratio between the motor and the wheel axle in this case equal to 4, finally α_w and α_m are the wheel and motor rotational acceleration, which can be related by Equation (31).

$$\alpha_m = \alpha_w i_k \quad (31)$$

To obtain the motor rotational acceleration by Equation (30), firstly the friction coefficient μ need to be estimated. To evaluate different track coefficients, the correlation curve between the friction coefficient and slip is taken from the literature (Braess & Seiffert, 2005). In Figure 21 is shown the curves implemented in the simulations.

Figure 21 - Friction coefficient vs.slip rate.

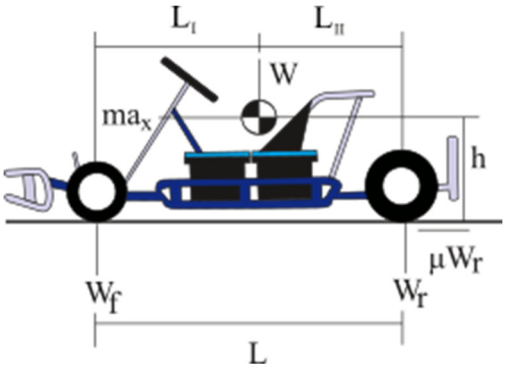


Source: Adapted from Braess, et al. (2005, p. 410).

The slip rate is easily calculated by the relation of wheel and vehicle speeds, as showed in the next subsection. The friction coefficient is directed obtained by the interpolation of the curve showed in Figure 21.

The vehicle dynamics is given by a simplified evaluation of Figure 2, where the vehicle is in a surface without inclination, the velocity is low enough to reject aerodynamic forces, there is no trailer attached and the tire losses are rejected. The used forces and dimensions are shown in Figure 22.

Figure 22 - Simplified forces on Go-kart.



Source: Elaborated by the author.

By the sum of forces on the longitudinal axle is possible to obtain the Equation (32).

$$ma_x = \mu W_r \tag{32}$$

Where the rear normal force is given by the Equation (33) when considering the load transfer between the axles during acceleration. In Equation (33) represents the gravity acceleration.

$$W_r = W\left(\frac{L_I}{L} + \frac{a_x h}{gL}\right) \tag{33}$$

Substituting Equation (33) in the Equation (32) and rearranging, the acceleration of the vehicle can be directed calculated by Equation (34).

$$a_x = \frac{\frac{L_I\, g}{L}}{\frac{}{\mu} - h} \tag{34}$$

During the simulation, the friction coefficient is estimated from the slip angle and the Equation (34) are used to obtain the vehicle acceleration. With the friction

coefficient and the normal load on the rear axle, the Equation (30) is solved to obtain the motor rotational acceleration. Integrating the acceleration it is possible to obtain the motor rotational velocity.

3.2.4 Slip calculation

The calculation of the slip rate is directly made by the normalized difference between wheel velocity and vehicle velocity (Radac & Precup, 2018). The algorithm takes into account two conditions:

a) When the vehicle is accelerating the wheel turns faster than the vehicle velocity the slip is calculated by the Equation (35).

$$\text{Slip} = \frac{v_r - v_x}{v_r} \tag{35}$$

b) when the vehicle is decelerating the wheel turns slower than the vehicle velocity and the slip may be calculated by the Equation (36).

$$\text{Slip} = \frac{v_r - v_x}{v_x} \tag{36}$$

In the algorithm a value of 10^{-16} is summed in the divisor to avoid divisions by zero.

3.2.5 Vehicle longitudinal movement

The block “vehicle longitudinal movement” in Figure 18 realize the integration of the vehicle acceleration originated from the “torque balance on the wheel” block. Therefore, vehicle velocity and vehicle displacement are obtained.

3.2.6 Display and save

Finally, the data is saved in an array. The data saved consist in accelerator pedal position, vehicle and wheel velocity, slip angle, vehicle acceleration and motor current.

3.3 DATA PROCESSING

This step of the implementation consists of the creation of adequate data to the learning process. All collected data is normalized and the reward is calculated to permit the training of the network. All the data collected in both environments are treated and saved in a database inside an array containing: actual state, actual action, next state and reward.

Each one of the procedures to normalize and to calculate the cited parameters are described in the next subsections.

3.3.1 States and actions

The states and actions are directly measured from the environment and saved in the database. To accelerates the network training, data is normalized as proposed by Ioffe and Szegedy (2015). The authors proposed a normalization based on the variance and average of each minibatch.

However, to keep the correct correlation between the variables the normalization of the data is realized directly by the maximum and minimum expected values. This procedure reduces the difference between the physical units keeping the characteristics of the raw data.

All the received data is normalized in a range from 0 to 1. The calculation is made by the Equation (37) where x_n is the normalized value, x is the measured value and x_{max} and x_{min} are the maximum and minimum possible values respectively.

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{37}$$

The input states for actor and critic networks are vehicle velocity, motor current, vehicle acceleration, accelerator pedal position and wheel velocity. However, two approaches are taken, with and without the vehicle velocity. Since the real velocity of the vehicle is difficult to be measured, it can be obtained just during the training process with optical measurement of the vehicle velocity. Without the vehicle velocity as an input of the controller the trained algorithm can be applied in other vehicles without the necessity of measuring the real velocity of the vehicle.

The action used as input in the critic network and as output of the actor is the desired position of the acceleration pedal that keep the slip of the wheels in the desired range.

3.3.2 Reward function

The reward function is responsible to indicate how "good" is take an action in the actual state. Considering a Markovian Decision problem, the actual state and action can be directly evaluated by the next state.

The adequate acceleration of the vehicle should keep the wheel slip in an adequate range to obtain a greater friction coefficient (Hori, et al., 1998). In Figure 4 is presented this range as proposed by Hory, et al. (1998).

However, the most important relation is that the desired accelerator position never can be higher than the accelerator position chosen by the driver. This behaviour can cause a vehicle over acceleration placing the driver in a dangerous situation.

The application of this controller may use a product result of the driver acceleration and the network output to avoid this dangerous manoeuvre. However, this operation would make impossible to fully understand the limitations during the application of the network in automotive controllers.

The reward function is calculated based on the database using the Equation (38), where GP is the acceleration pedal position chosen by the driver, a is the controller output action and v_x' and slip' are the vehicle velocity and wheel slip ratio in the next state.

$$r(s,a)=\begin{cases} 0 \text{ if } 0 > a > 1 \\ 0 \text{ , if } GP + 0.05 < a \\ 0, \text{ if } |slip'| > 0.2 \\ 1- |a - GP|, \text{ if } |slip'| \leq 0.2 \text{ or } (v_x' \leq 0.05 \text{ and } v_r' \leq 0.0625) \end{cases} \quad (38)$$

As shown by Equation (38) null reward was applied when the action is out of the range, the action is greater than the desired acceleration or the wheel slip ratio in the next step exceeds 0.2, changing to an unstable situation.

When the wheel slip is kept in the desired area, the vehicle can be in the required velocity or can be kept stationary for example. To avoid this event the reward is calculated by the difference between the throttle pedal position of the driver and the output action.

Evaluating the Equation (36), very small velocities of the vehicle present an inconsistency on the slip calculation. When the vehicle is stopped (vx=0), any wheel velocity presents an infinity wheel slip. To avoid this inconvenience and permit the controller to learn how to start from rest a maximum wheel velocity of 6,25% of the final value is consider as good behaviour when the vehicle velocity is smaller than 5% of the final velocity of the car.

The next subsection explains the training process to obtain the final network responsible to control the system.

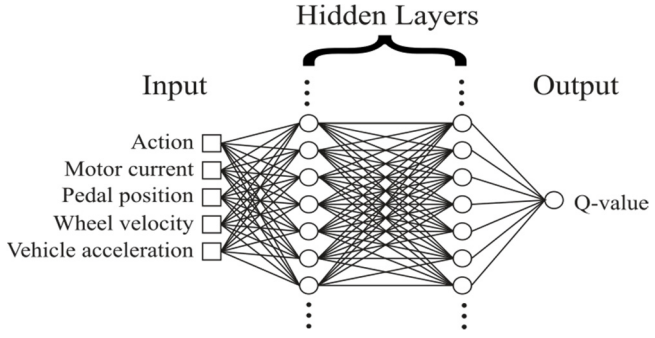
3.4 TRAINING PROCESS

Two different training processes were applied, one of these keeps the iterative learning way proposed by DDPG and NFQAC, while the new proposed algorithm named Directly Trained Neural Actor-Critic (DTNAC), takes the treated data obtained and directly trains the controller only once. The networks and both training algorithms are thoroughly explained in the next subsections.

3.4.1 Actor and critic networks

Both training processes used the same networks just changing the mode as the weights are learned. In this way, the characteristic of the network can be explained just once. The critic is a deep network with two hidden layers with twenty nodes each layer. The input number is six or five depending on the inclusion of the vehicle velocity in the inputs. The output consists of one node that represents the value function Q. Figure 23 shows a representation of the critic network without the vehicle velocity as an input.

Figure 23 - Critic network representation.



Source: Elaborated by the author.

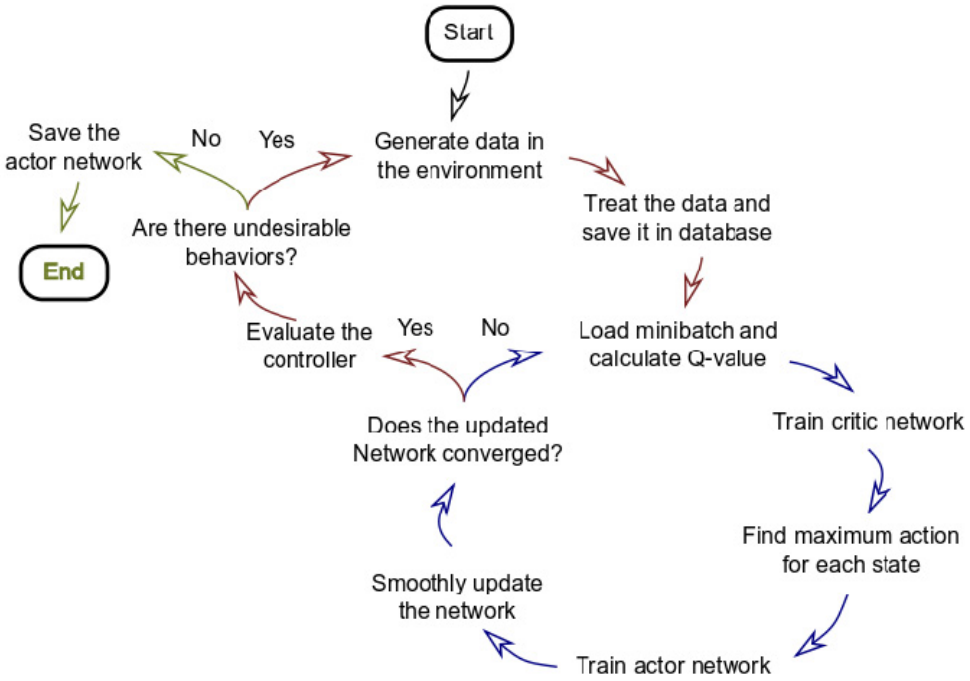
In Figure 23 the number of hidden nodes is resumed easing the comprehension. The critic network used in the hidden layers a Tangent-Sigmoid activation function to permit non-linear behaviours and a completely linear function was used in the output layer to permit a large range of outputs values. (Kim, 2017).

The actor-network used two hidden layers with twelve nodes each. The inputs were the same used in the critic network excluding the action that is the output of the network. The actor-network applies the same activation functions as the critic networks.

3.4.2 **Cyclic learning process**

Based on the algorithm proposed by Hafnet, et al. (2011) and Lillicrap, et al. (2016), the used cyclic algorithm suffer some updates to permit the application in our problem and the use of functions already available at MATLAB. The algorithm runs the training and treatment part on MATLAB and the simulation of the environment is given by the SIMULINK routine shown in subsection 3.2.

Figure 24 - Cyclic training process.



Source: Elaborated by the author.

Figure 24 shows in a schematic way the learning process in the cycle training of the networks. As presents in the scheme, the learning process occurs in two cycles inside each other. The path highlighted in red consist of the collection and treatment of the data

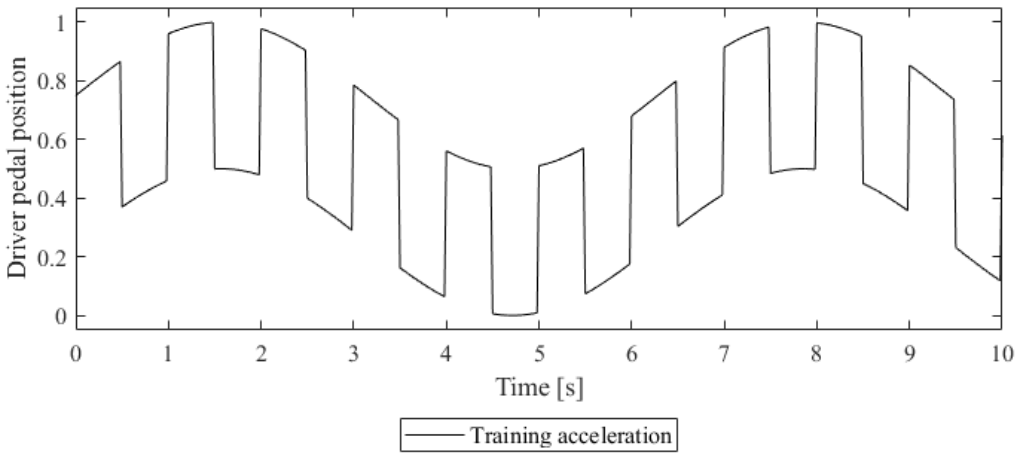
and the evaluation of the obtained controller. When applying this process in a real environment it occurs outside the main algorithm, driving the vehicle and evaluating the obtained controller.

In Figure 24, the blue path represents the training part where the networks are updated and the best actions are estimated to each collected state. In the next subsections, the scheme is explained in details.

3.4.2.1 Generate data in the environment

In the simulation environment, the desired manoeuver is given by the signal representing the driver accelerator pedal. In the evaluated case, this signal is a combination of a sinus wave, steps and a random signal that is included in a percentage of the data. This combination permits to vary the vehicle velocity and acceleration during the data generation. The input data without the randomization is shown in Figure 25.

Figure 25 - Input acceleration used during environment simulation.



Source: Elaborated by the author.

The trained network is implemented as a controller and the output is given directly by the weights of the network. At the first run, instead of using a completely random network to control the system as proposed by other authors, the controller is ignored, considering the output equal to the input. This tactic is used to create a “hint-to-go”, which shows to the network that to keep the output similar to the input in some cases is a “good” behaviour.

3.4.2.2 Treat the data and save in database

The environment simulation is run for 10 seconds in each cycle generating 500 points of data. Subsequently, the data is normalized and the rewards calculated as present in subsection 3.3.1. Subsequently, the data is saved in the database. The database archives a maximum value of data and start to replace the older one, always keeping the first "hint-to-go" simulation.

3.4.2.3 Load minibatch and calculate Q-value

To the training process, a minibatch corresponding to a percentage of the total database is selected randomly. This procedure avoids the loss of important data collected in the old tests and reduce the time of the learning process, as long as less data is needed for the learning process.

The first step after choosing the data is to calculate the value function Q, using the most recent networks available. The state action value function also named Q-value evaluates the expected reward on the next steps. Therefore, the Q of the actual state and action is based on the actual reward plus the Q waited to the next state, when the greedy policy is followed.

The next Q value can be easily obtained by the last trained networks. The actor evaluated at the next state, directly returning the next best action. The critic network, with the next state and action as input, result in the next Q-value. The Q-value is updated by the Equation (39) where γ, called discount factor, permit to operate in infinity horizons.

$$Q_{k+1}(s,a) = r(s,a)+\gamma\, Q_k\left(s', \mu_k(s')\right) \tag{39}$$

The Q-value is updated at each time that a new network is calculated and a new minibatch is loaded. The Q-value is calculated to the entire minibatch and is used to train the critic.

3.4.2.4 Train Critic Network

With the collected states, actions and calculated Q-values the critic network can be trained to permit the correlation between the inputs (states and action) and the output

(Q-value). The training of the critic network is made by the minimization of the normalized mean square error of the output.

The chosen method for minimization of the error is a Levenberg-Marquardt optimization. This method is a trust region method which consists of a combination of simplified Newton and Gradient descendent methods (MathWorks, 2019).

Both Newton and Gradient descendent methods move iteratively the calculated position inside the function in the descent direction until finding a minimum value that attends the convergence tolerance. The difference between the methods is given by the equation used to calculate the descendant direction. Newton descendent uses Equation (40), while Gradient descendent applies Equation (41) (Quarteroni, et al., 2014).

$$d^{(k)} = - \left[\mathbf{H}\left(x^{(k)}\right)\right]^{-1} \nabla f\left(x^{(k)}\right) \tag{40}$$

$$d^{(k)} = - \nabla f\left(x^{(k)}\right) \tag{41}$$

In Equation (40) and Equation (41) $d^{(k)}$ is the descent direction at the interaction k, H is the Hessian matrix, $x^{(k)}$ is the point evaluated, f is the function and ∇ is the gradient vector.

Due to the implementation of the Hessian matrix, the Newton method converges in quadratic order but is more sensitive to the chosen initial values. On the other hand, the gradient descendent permit a global converge, that do not depend on the initial values. However, its converges linearly. (Quarteroni, et al., 2014).

Another restriction of Newton descendent method is the necessity of calculating the Hessian matrix "**H**" to find the descent direction. The Hessian matrix is composed of second-order derivates and its calculation takes large computational power and time consumption (Quarteroni, et al., 2014).

The applied Levenberg-Marquardt method overcomes the Hessian and the gradient calculation by changing it by the Jacobian matrix as shown Equation (42) and Equation (43). In Equation (42) and Equation (43) J_R is the Jacobian matrix that contains first derivates of the network errors with respect to weights and bias and $e^{(k)}$ is the vector of network errors (MathWorks, 2019).

$$H\left(x^{(k)}\right) = J_R{}^T\left(x^{(k)}\right) J_R\left(x^{(k)}\right) \tag{42}$$

$$\nabla f\left(x^{(k)}\right) = J_R{}^T\left(x^{(k)}\right) e^{(k)} \quad (43)$$

The trust-region methods also apply a different approach when compared with descendent methods. In this case, instead of evaluating the descendant direction, the next point inside a specific region is directly calculated. According to the position inside the region, the next evaluation area can change de centre position and radius until the optimization of the answer. (Quarteroni, et al., 2014).

Therefore, the calculation of the next weights and bias of the network is given by Equation (44) (MathWorks, 2019).

$$x^{(k+1)} = x^{(k)} - \left[J_R{}^T\left(x^{(k)}\right) J_R\left(x^{(k)}\right) + \varepsilon I\right]^{-1} J_R{}^T\left(x^{(k)}\right) e^{(k)} \quad (44)$$

In Equation (44) I correspond to the Identity matrix and ε is a scalar that permits change between gradient and Newton methods. When ε is null the optimization behaves as a Newton descendent algorithm, however, when ε is large the behaviour is similar to a simple Gradient descendent method. In this way, ε decrease at each step that the algorithm presents a reduction of the error and increases when the error increase. These implementations permit adequate behaviour in large size problems due to the gradient descendent behaviour and a faster and precise definition of the minimum value when a small evaluation area is achieved. Due to these, it is defined as the fastest method for training moderate-sized feedforward neural networks (MathWorks, 2019).

Table 1 - Training stop criteria.

Maximum number of epochs	1000
Minimum gradient	1 e-7
Maximum ε	1 e10
Maximum validation failure	6

Source: Elaborated by the author.

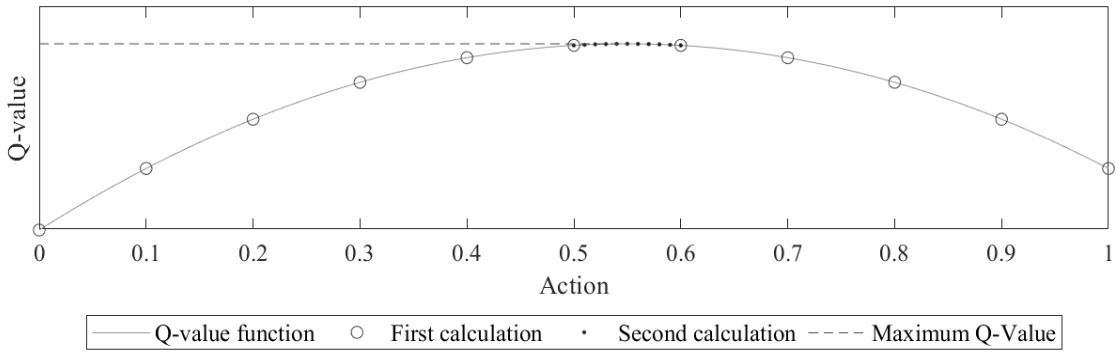
The criteria used to stop the training is given in Table 1 where each epoch is counted when all the data pass the network once and the validation failure consists in the number of consecutive times that the validation performance degrades.

3.4.2.5 Find maximum action

To find the correct correlation between states and actions each state of the minibatch needs a correspondent best action to permit the actor training. In this process, each one of the states has the action varied inside the action range until finding the best Q-value and the correspondent best action.

Using the last trained critic network the states are fixed and the action variated in 11 equal spaced points as shown the first calculation in Figure 26 After that, the actions that result in first and second-best Q-values are used as limits to a new division of 11 equal spaced points. The action that returns the best Q-value inside this range is saved together with the correspondent state.

Figure 26 - Chosen of the best action.



Source: Elaborated by the author.

3.4.2.6 Train Actor Network

Based on the pairs of states and best actions available at the actual minibatch, the actor-network is trained using the same optimization method and stop criteria present in the Critic-Network training available in subsection 3.4.2.4. In actor-network case, the states are used as input and the best actions are applied as the output of the network.

3.4.2.7 Smoothly Update of the Network

As proposed by Lillicrap et al. (2016) a copy of the networks is used to improve the algorithm convergence. After the training of the network, new actor and critic

networks receive part of the weights and bias from the last calculated networks and keep part of the old values.

The weights and bias of the copy networks are updated with a slow target networks change given by the Equation (24) to critic and Equation (25) to the actor-network (Lillicrap, et al., 2016).

3.4.2.8 Convergence criteria

The blue cycle in Figure 24 is considered converged when the calculated and the updated actor networks achieve a similar output to the same inputs. To calculate this difference all the states from the minibatch are used to generate an array of outputs in both networks. The mean square error of the output arrays is calculated and when it achieves a minimal tolerance the learning process is considered converged and the algorithm follow to the next step.

3.4.2.9 Controller Evaluation

To evaluate the performance of the updated networks the simulated environment is run again on the trained floors. This time the input signal is the same used in the training process, present in Figure 25, but without implementing the random pedal position.

Based on the rewards received in this simulation is possible to evaluate the occurrence of ‘bad behaviours’. When all the rewards are greater than 0.2, no actions out of range or high slip ratios are found in the simulation. The minimum of 0.2 also guarantees that the action is not kept in 0 when the drive desired is to accelerate the vehicle. Besides the greater reward condition, a maximum number of 200 runs has been set to prevent non-convergent models from maintaining the cycle infinitely.

When the controller is considered adequate, the actor-network is saved, otherwise, the training process is continued restarting the simulation of the environment. The results of the simulation increase the amount of data or change it by a more actualized, always using the most recent controller available. With the new data, the cycle of Figure 24 continues by a chose of a new minibatch and training of new networks.

3.4.3 Proposed learning process

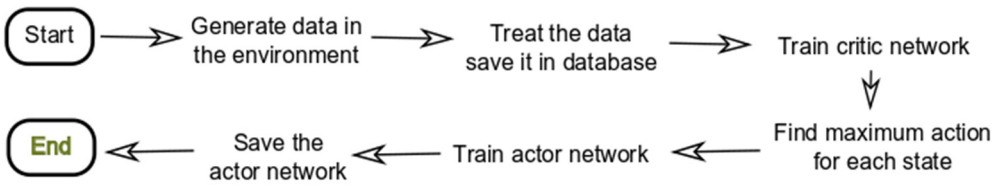
The proposed algorithm is similar to the cyclical, however, some simplification was applied due to the non-iterative implementation. The main reason for the application of cyclic algorithms is the necessity of improvement of the Q-value from a sequence of experiences. The Q-value or value function is changed incrementally after each cycle based on the rewards obtained by the last state-action combination and the last Q-value evaluated in the same state-action, as present in Equation (39).

To overcome this restriction a so-called myopic value function can be used. In this approach, the discount factor γ is null and the Q-value is equal to the reward and the last value function is ignored as present the Equation (45). The advantage of this learning process is the possibility to calculate the value function in only one cycle.

$$Q_{k+1}(s,a) = r(s,a) \tag{45}$$

With this simplification, some steps of the cyclic learning process can be removed, resulting in the learning process shown in Figure 27.

Figure 27- Non-cyclic learning process.



Source: Elaborated by the author.

The next subsections present in detail the topics of Figure 27.

3.4.3.1 Generate data in the environment

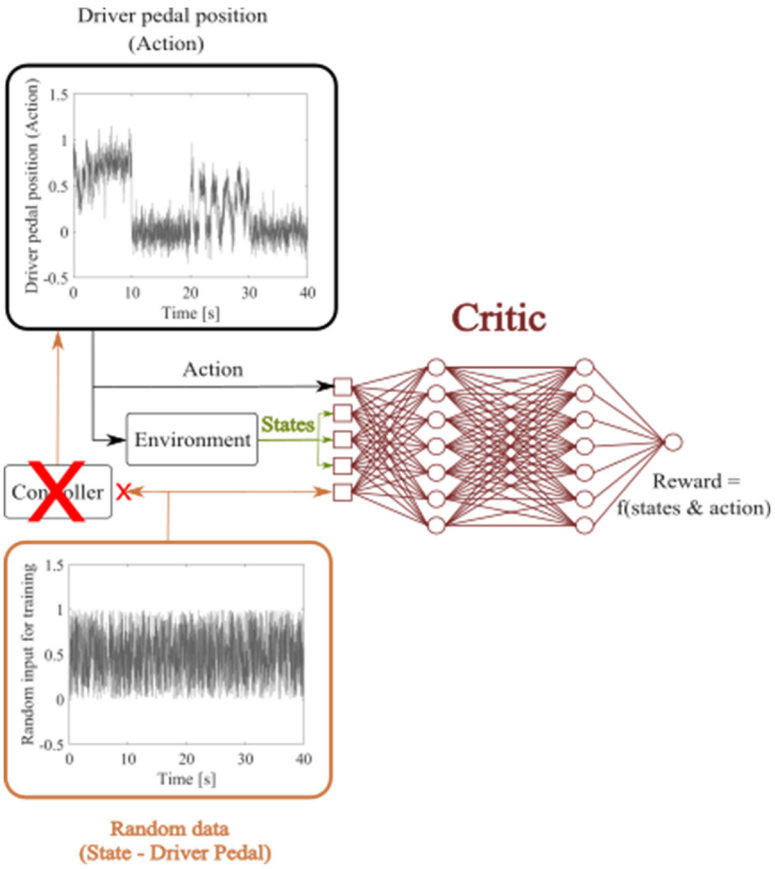
The generation of data is the part of the training process that suffered the greatest change inside the algorithm due to the necessity of having enough data to cover a large number of different state and actions. Instead of applying a known input as driver throttle pedal position, a combination of sinus and step in different phases and a small noise was applied to the input, permitting to vary all region of velocities and accelerations. In Figure

28 inside the “Drive pedal position (action)” box is shown part of the created driver pedal position.

As indicate in Figure 28, the second restriction is given by the absence of the controller, impeding the generation of actions calculated by its. To overcome it, the acceleration pedal position provided by the driver was directly used as the action. This procedure was made taking into account that just the accelerator pedal generated by the controller has a correlation with the environment to generate the next states. On the other hand, the accelerator position chosen by the driver has just a mathematical correlation inside the networks as presented in Figure 28.

Taking into account the necessity of evaluating a large number of states, a new completely random accelerator position was chosen as a state in the training process. It allows the network to understand if the chosen action and generated states are adequate in different accelerations desired by the driver. In Figure 28, is presented the random input used as accelerator pedal position in the training process inside the box named “Random data (State – Drive Pedal)”.

Figure 28 - Correlation between environment and input data.



Source: Elaborated by the author.

When applying the proposed algorithm in a real environment the measured driver pedal position can be used as environment input (action) in a way similar to the simulation, avoiding the necessity of cyclic training. To the training step, a completely random array can be used to simulated the driver pedal position.

3.4.3.2 Process the data and save in database

As opposed to the cyclical method, the data generation occurs just once. After it, all collected data is normalized and the rewards calculated as present in subsection 3.3.2. Subsequently, the data is saved in the database.

3.4.3.3 Train critic network

Critic-Network training occurs in a similar way that the training applied in the cyclic process. Using all obtained states, actions and calculated rewards, the critic can train the weights and bias to obtain the correct input-output correlation. The same optimization method and similar stop criteria as the cyclic training available in section 3.4.2.4 are adopted. The single stop criteria changed is the maximum validation failure which is increased to 20 to avoid the early ending of the training.

3.4.3.4 Find maximum action for each state

The process to find the best action is the same applied in the cyclic training and described in section 3.4.2.5.

3.4.3.5 Train and save actor-network

Based on the pairs of states and best actions available in the database, the actor-network is trained using the same optimization method and stop criteria present in the Critic-Network training available in section 3.4.3.3.

In the proposed algorithm no evaluation of the controller is realized. The final output is directly saved and applied in the environment as long as in the real implementation the controller need to be evaluated directly in the vehicle.

3.5 TRAINING EVALUATION

Before starting the evaluation of the trained controllers an adequate controller needed to be chosen and the influence of the input variables on the results understood. Therefore a factorial design of experiments (DOE) was applied at the cyclic and the proposed training algorithms, evaluating separately the controllers with and without the vehicle velocity as input.

To statistically determine if the factor produces a significant influence on the result, the 2^k analyses use a t-statistics method which tests the null hypothesis considering the effect on the result as null (Minitab, 2019). The level of significance applied in this study is 5%.

The DOE is run in the Minitab 18, evaluating the effects on the mean reward and on the time to train the network. The mean reward is calculated in 20 seconds of manoeuvre. The manoeuvre simulates the vehicle driver in two different floors using the trained throttle position as input.

The statistical effect on the results by each factor is evaluated by the Pareto graph. This graph plots the correlation of the t-value value for each factor correlating it with a red line, which is drawn on the quantile of a t-distribution with degrees of freedom (DOF) equal to the DOF of the DOE error. The DOF takes into account the number of levels, number of factors and number of evaluated interactions (Minitab, 2019).

To understand the correlation of the effects the main effect graphs are also plotted. In this case, the average output to high and low levels are plotted to evaluate the influence of the factors.

3.5.1 Identification of variables influence in the cyclic learning algorithm

The factorial DOE applied at the cyclic algorithm uses 5 factors. No replication or centre point is applied and a resolution V is implemented to reduce the number of runs from 32 to 16. According to Montgomery (2009b), this resolution allows the estimation of all main effects and two-factor interactions. How in this research just the mains effects are evaluated the resolution V is sufficient. In Table 2 is presented the factor and levels utilized during the training.

Table 2 - Factors of the DOE for the cyclical algorithm.

Factor	Low level	High Level
Target update	0.2	1
Discount Factor	0	0.9
Minibatch Size	0.25	1
Min number of training cycles	1	5
Data randomization	0.1	0.5

Source: Elaborated by the author.

In Table 2 the target update consists in the constant proposed by Lillicrap, et al (2016) that defines how smoothly is the update of the copied matrix as present in Equation (24) and Equation (25). The discount factor is the parameter in the value function shown in Equation (39) that define how much future behaviours influence the actual decision. The minibatch size corresponds to the percentage of data from the database which is selected to the training process. The minimum number of training cycles represents the minimum number of times that the blue cycle in Figure 24 is calculated. The data randomization gives the percentage of random data using in the data generation insert in the input of the controller.

3.5.2 Identification of variables influence on the proposed learning algorithm

The analyses applied to the proposed training process uses just 3 factors, without replication or centre point. In this case, all factors are related to data generation form, aiming to improve the reliability of the data and the possibility of evaluating a larger space of states and actions. Due to the small number of factors, a full resolution experiment is implemented, totalizing 8 runs. In Table 3 is presented the factor and levels utilized during the training.

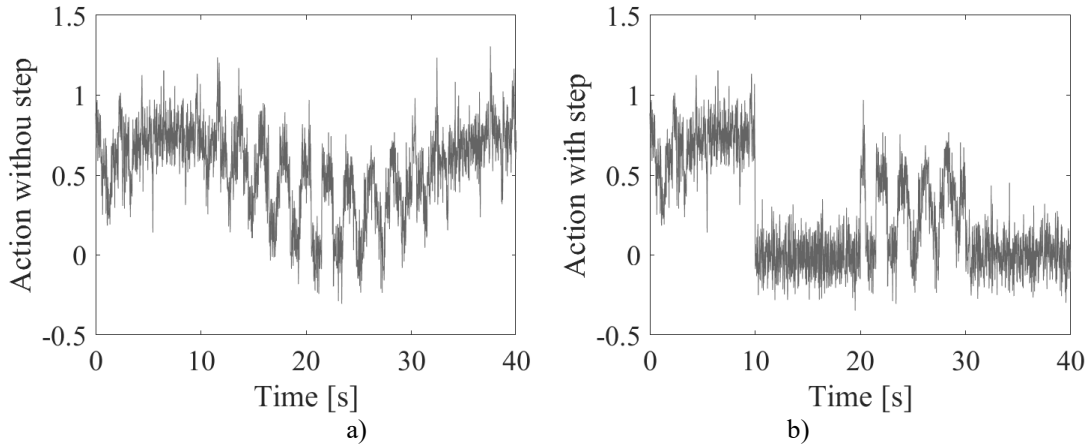
Table 3 - Factors of the DOE for the proposed algorithm.

Factors	Low level	High Level
Step	No	Yes
Reward	-1	0
Amount of data	25 min	50 min

Source: Elaborated by the author.

The step factor in Table 3 consists of the application of a unitary step with a period of 20 seconds and a pulse width of 50%. This signal multiplies the accelerator data before the noise application permitting to keep the pedal position around zero and reduce the velocity of the vehicle. This strategy is evaluated to permit the creation of data in the states where the vehicle velocity is slow. In Figure 29 a) is presented the accelerator signal without the step, while in Figure 29 b) is shown the signal with the step application.

Figure 29 - Signal influence by the step factor.



Source: Elaborated by the author.

The reward factor in Table 3 is related to the reward given when bad behaviour is computed. In the high level of this factor, the Equation (38) is kept the same, while in the low level all the 0 rewards of Equation (38) are changed to -1. This factor permit to understand how the learning process works with large and small differences between good and bad rewards.

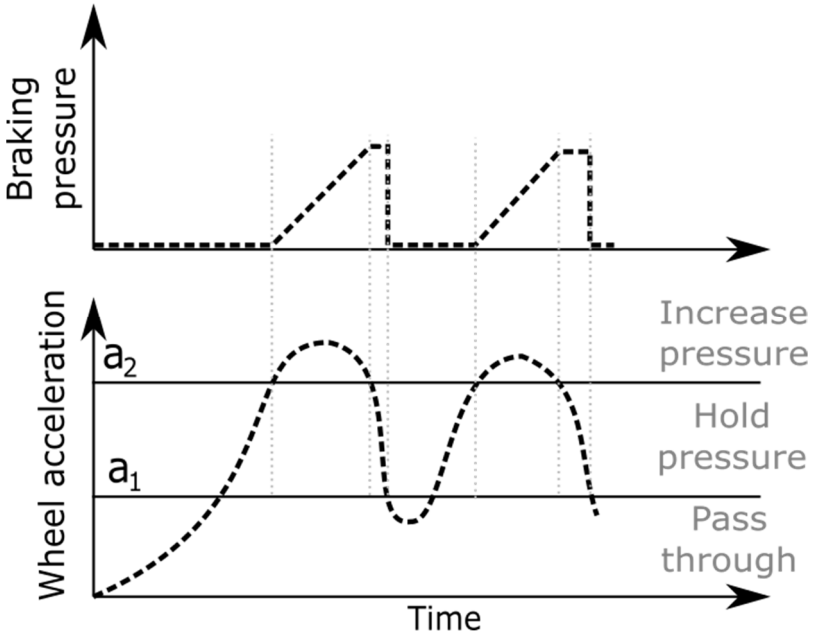
Finally, the factor amount of data is respective to the time which data is collected, aiming to understand how important is applied a large amount of data.

3.6 CLASSICAL CONTROLLER

According to Rajamani (2006), the actual controllers are mainly based on ABS actuation, which takes into account the wheel acceleration to suppress the rotational velocity by braking. In this approach, the wheel acceleration is used to determine unstable behaviours, applying two threshold values, a_1 and a_2. When the acceleration of the wheel reaches a_2 the brake pressure is increased until the acceleration of the wheels falls below a_2. After that, the pressure of the brake is kept until the wheel acceleration is lower than

a_1, where the brake pressure provided by the driver is again directly passed through the system. In Figure 30 is shown an example of controller behaviour.

Figure 30 - ABS based controller behaviour.



Source: Elaborated by the author.

Based on it, a simple classical controller is used to permit the understanding of classic controllers limitations and the benefits of using Actor-critic controllers. In the applied classical control, instead of controlling the brake pressure, the throttle position is administrated. In this architecture, the acceleration thresholds do not show adequate results and the controller is directly based on the last calculated slip ratio. However, this is an ideal application, as long as in a real environment the slip cannot be directly calculated, but with some accuracy, can be estimated by a correlation between wheel velocity and vehicle acceleration.

Based on the ABS controller, the proposed classical controller applied two slip ratio thresholds, the first at 0.15 and the second at 0.2. Each step that the slip ratio is higher than 0.2 the virtual pedal position is reduced in 0.1. Inside the region of slip ratio between 0.15 and 0.2, the virtual throttle is kept fixed on the last adequate value and in slip ratios slower than 0.15 the driver desired acceleration position is passing through.

In addition, the virtual throttle always should be small than the real acceleration pedal position and the output of the virtual throttle signal should always be kept inside the 0 to 1 range. The calculated output of the controller that is higher than the driver

desired value is changed to the desired value and calculations that return negative values are changed to 0.

The implemented classical controller generates a high-frequency shift of the desirable throttle position in a similar way that brake pressure oscillates in the ABS system. To understand the correlation of the acquisition rate on the slip behaviour, an initial experiment variating the simulated acquisition rate is applied to evaluate the mean reward received and the slip during the time. To this test, the input of Figure 25 is given as driver accelerator pedal position.

3.7 CONTROLLER TEST

After the optimization of the controllers, four Actor-Critic controllers are obtained by the combination of the cyclic or proposed algorithm and with or without vehicle velocity as input. Those controllers are compared in different situations to understand its limitations.

The next subsections present the different environments evaluated.

3.7.1 Trained conditions

The first evaluation of the results consists of the comparison of the different training process, including the cyclic and proposal trained networks with and without the vehicle velocity as an input. The results are compared with the classical algorithm on the trained grounds and acceleration signals. This first evaluation is important to understand the control possibility in scenarios included in the training process. This application also permits to compare the performance of the RL controllers with the system without a controller or with the classical controller.

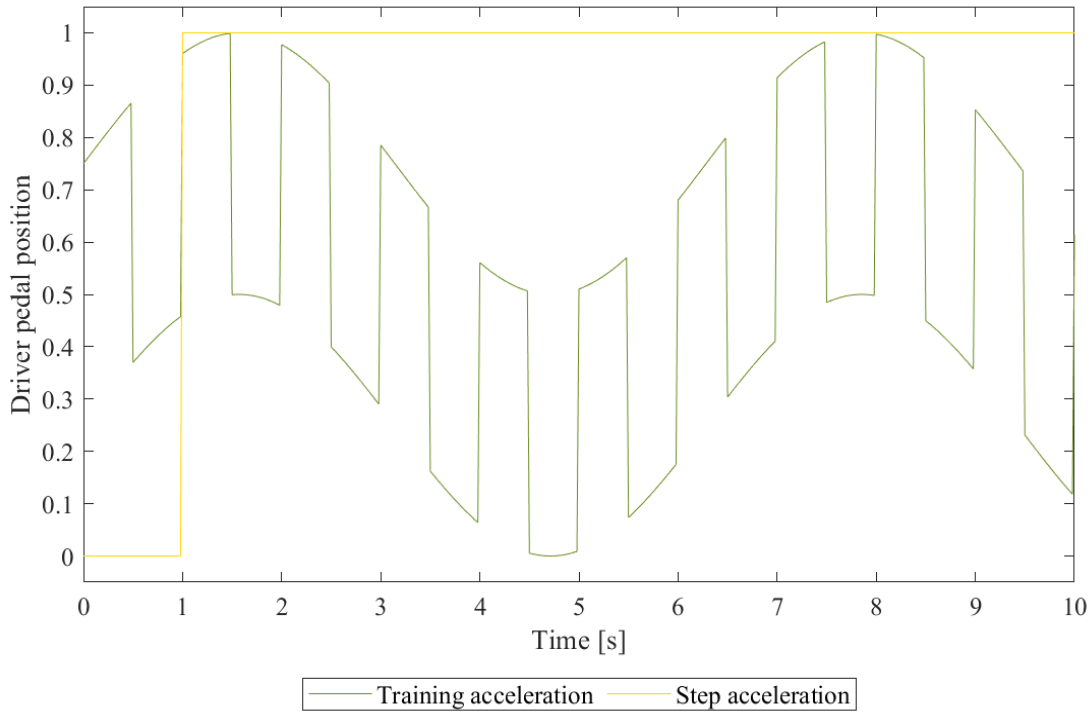
3.7.2 Different acceleration patterns

To verify if the trained networks can handle with no trained situations a different input signal is evaluated on the trained grounds. The results are compared with the system without a controller and with the classic controller.

In Figure 31 is presented the signal of the training process and the step signal chose to evaluate the controller behaviour in a different situation. The step signal is

chosen due to the fast and high change on the signal condition and because it is a very common behaviour of the electrical Go-kart drivers.

Figure 31 - Acceleration strategy during simulation.



Source: Elaborated by the author.

3.7.3 Different floor patterns

The controller behaviour in untrained tracks is realized by the changing of the friction coefficient vs slip rate curve in the simulation. In the first step, all the tracks available in Figure 21 are tested in all evaluated controllers using controllers trained in tracks of snow and dry asphalt. It permits the evaluation of extrapolation and interpolation possibility of the controllers, as long as ice represents the slippest floor, providing an extrapolation analysis while wet asphalt is a floor with characteristic between the snow and the dry asphalt, permitting the interpolation investigation.

In a second step, the proposed algorithm is trained in ice and dry asphalt with the velocity as input to verify how the controller works in a complete interpolation mode. To improve the controller performance different periods of low velocity in the training process are also evaluated.

3.7.4 Different acquisition rates

The real implementation of the controller on the vehicle presents limitations due to the sample time of the individual sensors. As explained in Appendix A, the vehicle velocity sensor has a sample time of 50 ms, while the wheel velocity has a sample time of 200 ms. The higher sample time occurs due to the calculation of the parameters that depend on the average of the displacement on the time, requiring more than one capture to process the velocity.

In all previous simulations, this factor is neglected, but a final test evaluates the proposed controller trained on ice and dry asphalt with vehicle velocity as input using data generated with different sample times, similar to the real vehicle.

4 RESULTS

This section presents the definition of the best controllers and the influence of training and generated data on the obtained controllers. In the second part, the chosen controllers are compared in different environments to comprehend its limitations.

4.1 TRAINING EVALUATION

The next subsection presents the influence of the training set and the collected data on the controller reliability in addition to the definition of the best training process to each controller. To facilitate the denomination in this and further subsections the algorithms with the vehicle velocity as a controller input are named 5 states, while the algorithms without the velocity of the vehicle as input are named 4 states.

4.1.1 Identification of variables influence in cyclic learning

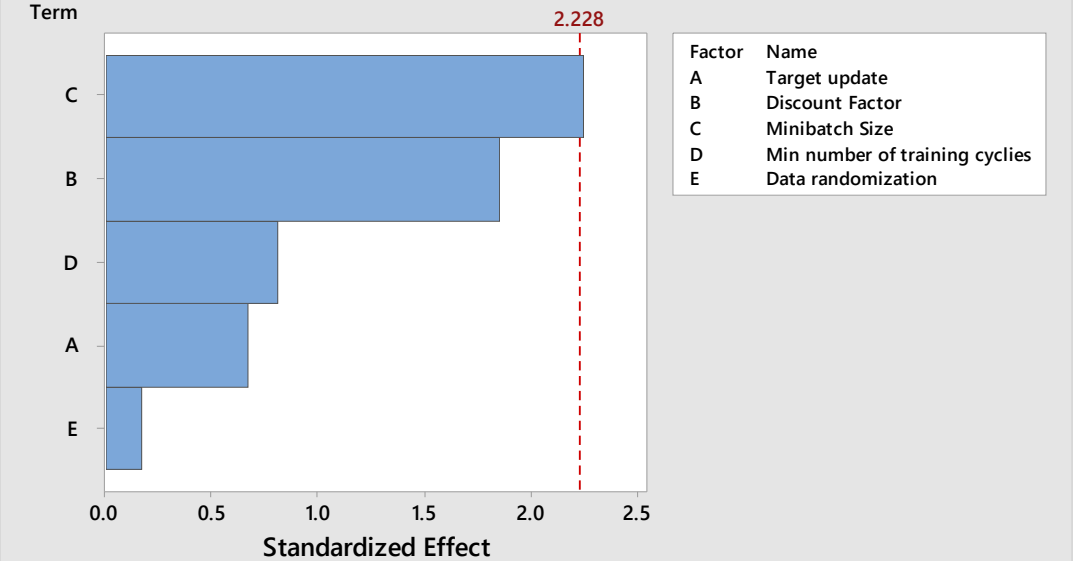
Due to the complete different behaviour and to easily comprehension the 4 and 5 states controllers are evaluated separately, allowing to obtain optimized controllers in both cases.

4.1.1.1 Five states algorithm

The Pareto charts of the 5 states cyclic algorithm obtained by the influence on the average reward and on the time are shown in Figure 32 and Figure 33 respectively. This chart present, with 95% of confidence, which factor had an influence on the evaluated result. Effects that present a result higher than the red reference line, statistically are significant on the results influence.

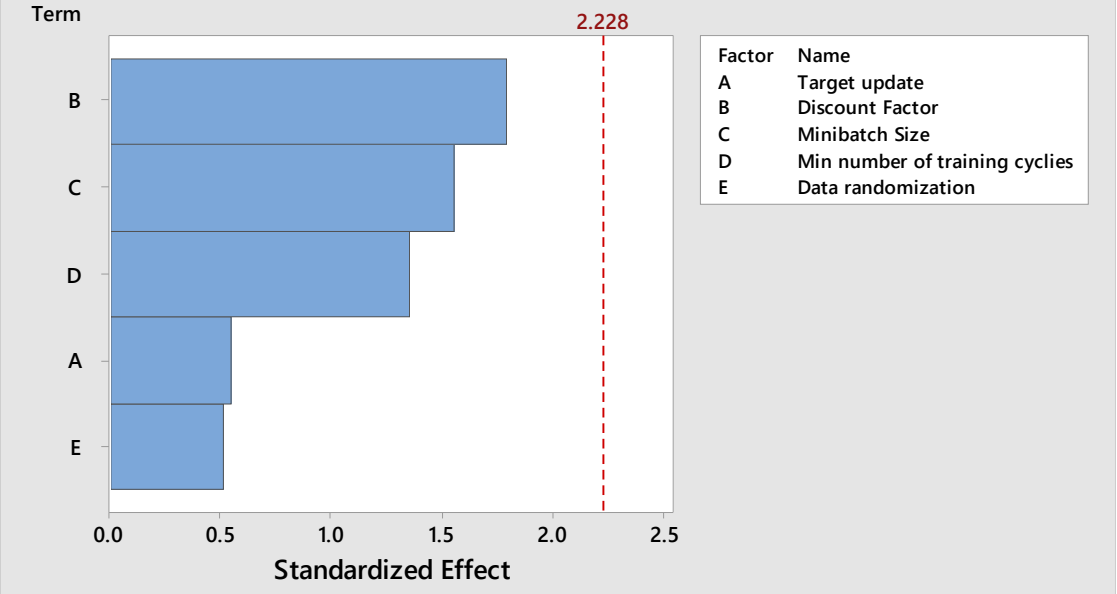
Evaluating the Pareto charts just the minibatch size influenced the results on the best average reward, while the evaluated factors do not influence the time. The low influence can occur by the small change of the results with the parameters change or by a great variation of the collected data. To understand the correlation, the main effects plots for best average reward and for time are shown in Figure 34 and Figure 35 respectively. In Figure 35 the time is given in minutes.

Figure 32 - Pareto Chart of the Standardized Effects on Best average time - 5 states, cyclic algorithm α = 0.05.



Source: Elaborated by the author.

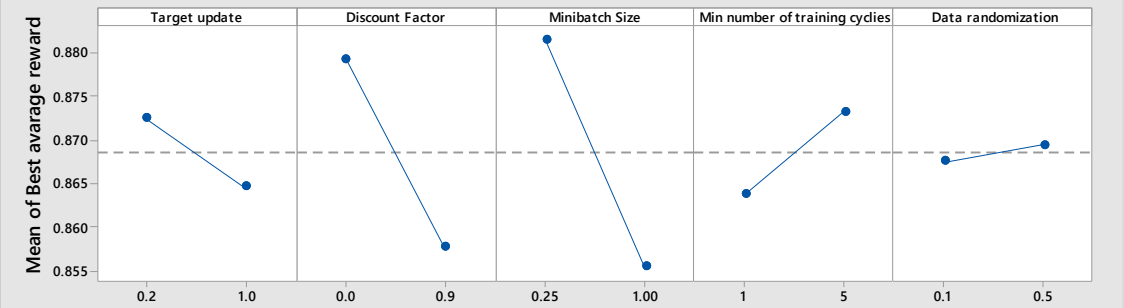
Figure 33 - Pareto Chart of the Standardized Effects on time - 5 states, cyclic algorithm α = 0.05.



Source: Elaborated by the author.

As shown in Figure 34, the average reward really suffers a small change with the variation of the parameter. This is an awakened outcome result, as long as the cyclic algorithm trains the network until the convergence of the controller, allowing high average rewards to all trained controllers.

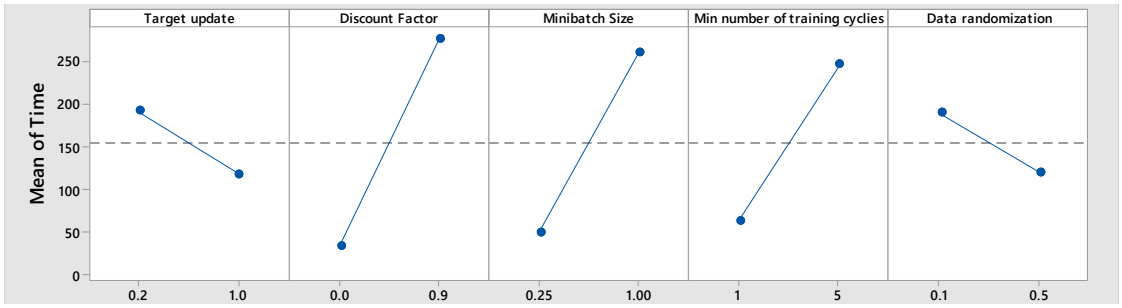
Figure 34 - Main effects plot for best average reward - 5 states, cyclic algorithm.



Source: Elaborated by the author.

In Figure 34 minibatches of 25% presents a better behaviour of the trained controller. Despite that the discount factor is not considered as a factor that influences the results, in this test, the null discount factor receives a better average reward when compared with the controller trained with discount factors of 0.9.

Figure 35 - Main effects plot for time - 5 states, cyclic algorithm.



Source: Elaborated by the author.

In Figure 35 a large change in the training time is presented, with averages from 30 to 280 minutes, which are large changes in the waiting time for training. In this way, the no influence of the factors on the time are derivated of the high variation of the training time, which do not present an adequate pattern. Thus, the application of the cyclic algorithm shows a limitation on its robustness, since is not possible to define a convergence time.

How it is not possible to really define a better training process and as all the trained networks were available, the network that returns the best average reward was chosen. The network used in the next subsections is trained with the factors shown in Table 4.

Table 4 - Training parameter of the 5 states cyclic algorithm.

Factor	Target update	Discount factor	Minibatch size	Minimum training cycles	Data randomization
Value	0.2	0.0	0.25	1.0	0.1

Source: Elaborated by the author.

4.1.1.2 Four states algorithm

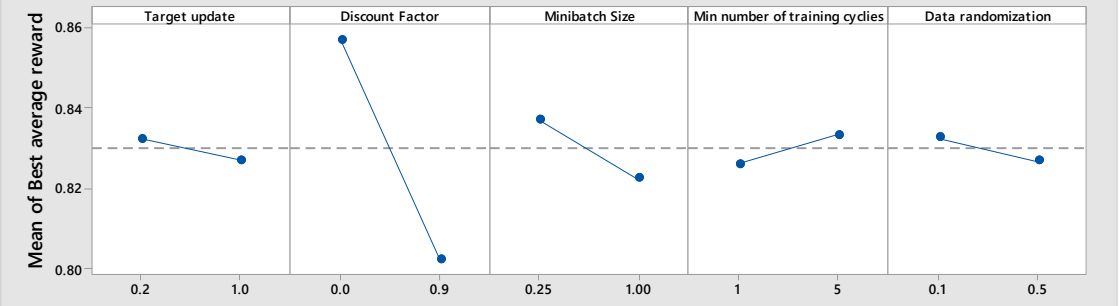
Figure 36 is shown the Pareto chart for the 4 states cyclic algorithm, evaluating the best average reward result. Among the factors, the discount factor is the only to show influence on the average reward.

Figure 36 - Pareto Chart of the Standardized Effects on best average reward - 4 states, cyclic algorithm α = 0.05.

Source: Elaborated by the author.

To identify how the influenceable factor interact with the levels the main effects plot of the cyclic 4 states algorithm is presented in Figure 37. Evaluating the plot, the discount factor presents a similar behaviour as the effects on the 5 states cyclic algorithm, returning better results when it is equal to 0 and does not take the future behaviours of the system into account. This presents itself as a good opportunity of success of the proposed algorithm, which ignores the future behaviour of the vehicle to permit the training in one single step.

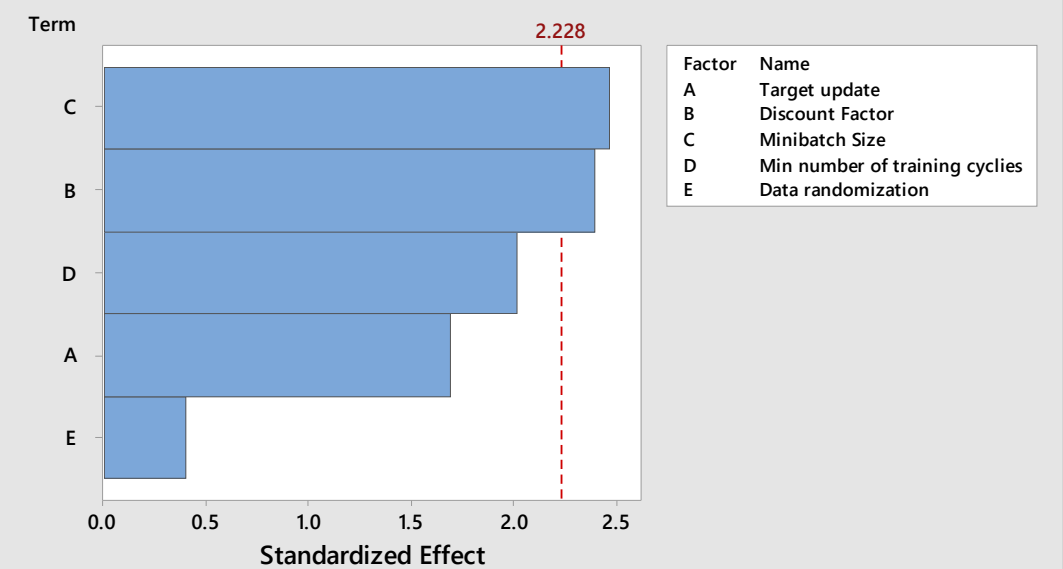
Figure 37 - Main effects plot for best average reward - 4 states, cyclic algorithm.



Source: Elaborated by the author.

In the Pareto chart of Figure 38, the discount factor and minibatch size influence the time of training while the minimum number of training cycles, the target update and the data randomization did not show significant influence on the time when four states are evaluated with the cyclical training.

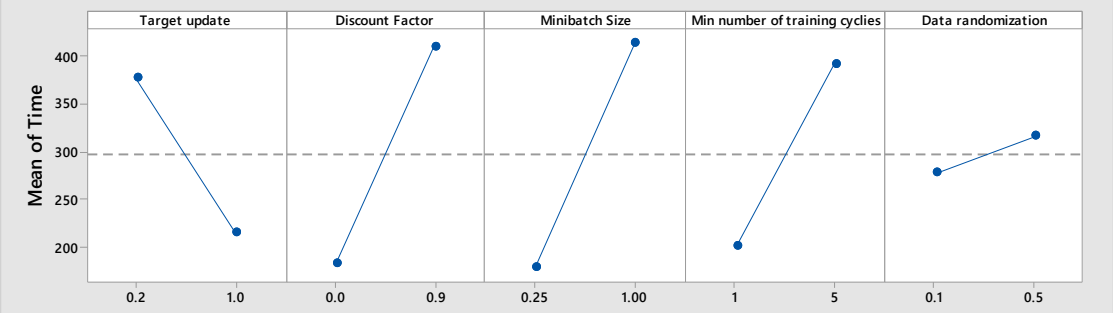
Figure 38 - Pareto Chart of the Standardized Effects on time - 4 states, cyclic algorithm α = 0.05.



Source: Elaborated by the author.

To understand the correlation between the factors and the time, the evaluation of the effects on time in the 4 states cyclic algorithm is shown in Figure 39. In this evaluation, again the results that do not present influence on the Pareto graph had a large variance of time, reaching 150 minutes of difference between the averages of the low and the high levels.

Figure 39 - Main effects plot for time - 4 states, cyclic algorithm.



Source: Elaborated by the author.

This behaviour reinforces the instability of the cyclic training, which with 4 or 5 states had large variations on the time processing which made impossible to comprehend the interaction between the factors and the time. It means that in the evaluated application, using the same parameters the training process could have very different training times depending on uncontrolled factors.

That is probably a result of the evaluation process, where the controller needs to reach a completely adequate behaviour. In the case that this perfect behaviour is not reached due to a small number of errors the controller can be overtrained and a larger time is necessary to reach the adequate behaviour in a second time.

Since the large variation did not permit do define the best controller, the trained network that returned the best average reward was chosen to the next evaluations. Despite that the fitted values of Figure 37 indicate 0.25 as the best target update, the individual controller that results in the best reward was the networks which apply a target update of 1. The difference between the statistical evaluation and the best result is derivated from the possibility of variation on the results and the small influence of this factor.

Table 5 indicates the train parameters used in the 4 states cyclic algorithm, which was applied as controllers in the next subsections. Most of the training parameters of the chosen controllers are the same between the 5 and 4 states cyclic algorithms, excluding the target updated which suffer changes.

Table 5 - Training parameter of the 4 states cyclic algorithm.

Factor	Target update	Discount factor	Minibatch size	Minimum training cycles	Data randomization
Value	1.0	0.0	0.25	1.0	0.1

Source: Elaborated by the author.

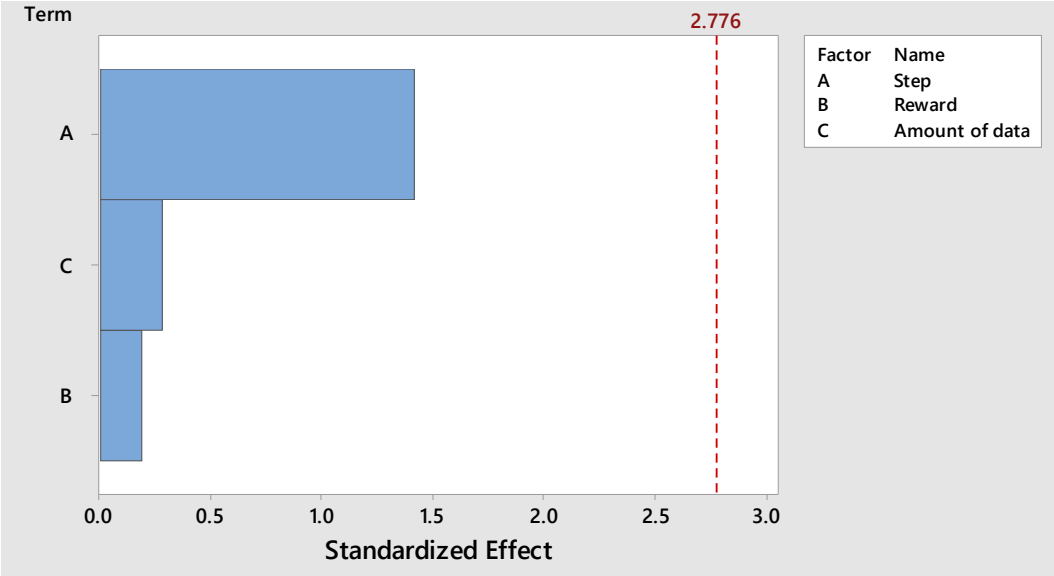
4.1.2 Identification of variables influence in proposal learning

In the same way that the cyclic algorithm, the proposed learning process was evaluated separately with and without the vehicle velocity as input, the next two subsection presents the factor effects and the optimized training factors.

4.1.2.1 Five states algorithm

The influence of the factors on the average reward of the proposed algorithm with the velocity as an input can be visualized in the Pareto chart in Figure 40.

Figure 40 - Pareto Chart of the Standardized Effects on average reward - 5 states, proposed algorithm α = 0.05.

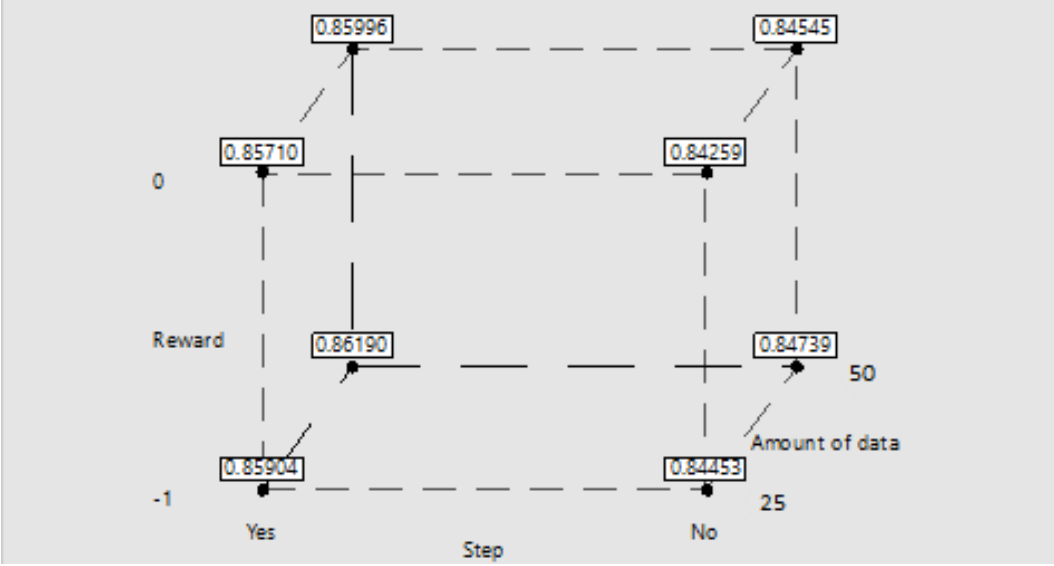


Source: Elaborated by the author.

Analysing the results of Figure 40, all the factors had no influence on the reward of the controller, in the same way, that in the cyclic evaluation, this behaviour can demonstrate a greater variance on the experiment or a really small change of the reward with the factors change.

Based on the cube plots all the fitted results can be displayed and a better understanding of the factors influence can be evaluated. In Figure 41 is presented the cube plot of the influence on the average reward.

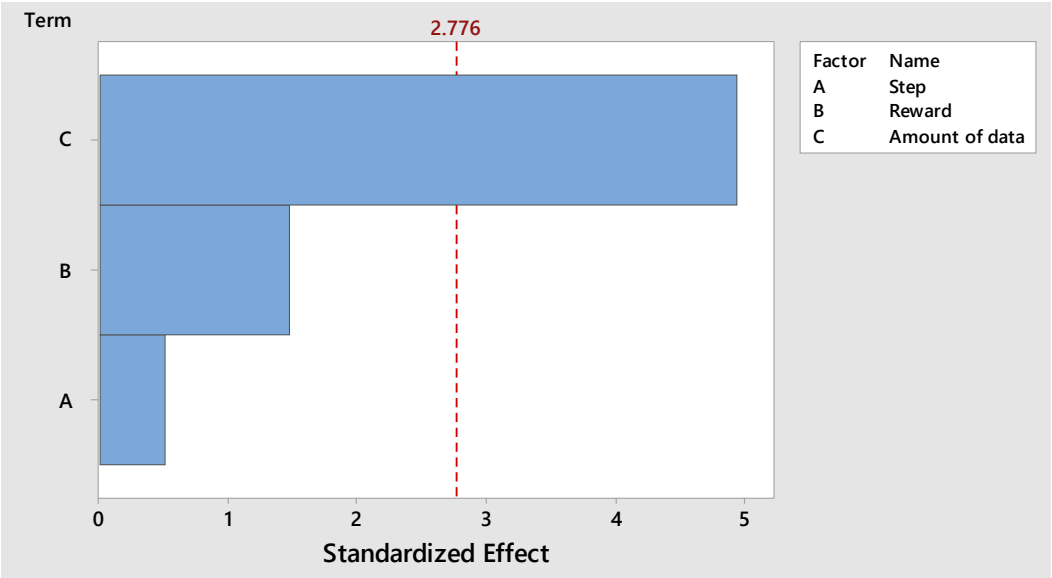
Figure 41 - Cube plot of the best average reward - 5 states, proposed algorithm α = 0.05.



Source: Elaborated by the author.

Evaluating Figure 41 all the results present similar values with small variance. This behaviour evidences that the evaluated factors really had no influence on the average reward of the controller. This result is a positive aspect of the training method, as long as the controller did not suffer the influence of the collected data, it shows itself as a robust training process, where no special care needs to be taken on the data collection.

Figure 42 - Pareto Chart of the Standardized Effects on the time - 5 states, proposed algorithm α = 0.05.

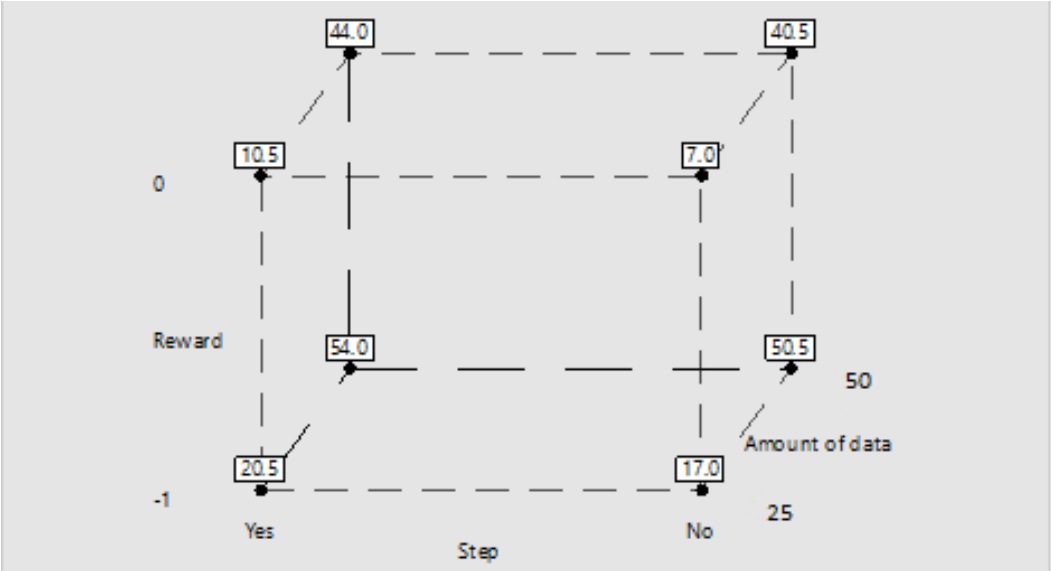


Source: Elaborated by the author.

On the other hand, as shown in Figure 42, the amount of data had a significant influence on the training process, result that is reinforced by the cube plot of the time dependence shown in Figure 43. In the plot, the amount of data generated an average increase of 40 minutes in the training process with the change from 25 to 50 minutes of data collected. The cube plot also presents a small influence of the step and the reward that can be ignored when compared with the amount of data influence.

The smallest time was present with the training process of no step, 0 of reward in bad situations and 25 minutes of data collection with a total of 7 minutes of training, while the opposite corner showed a training time of 54 minutes.

Figure 43 - Cube plot of the time - 5 states, proposed algorithm α = 0.05.



Source: Elaborated by the author.

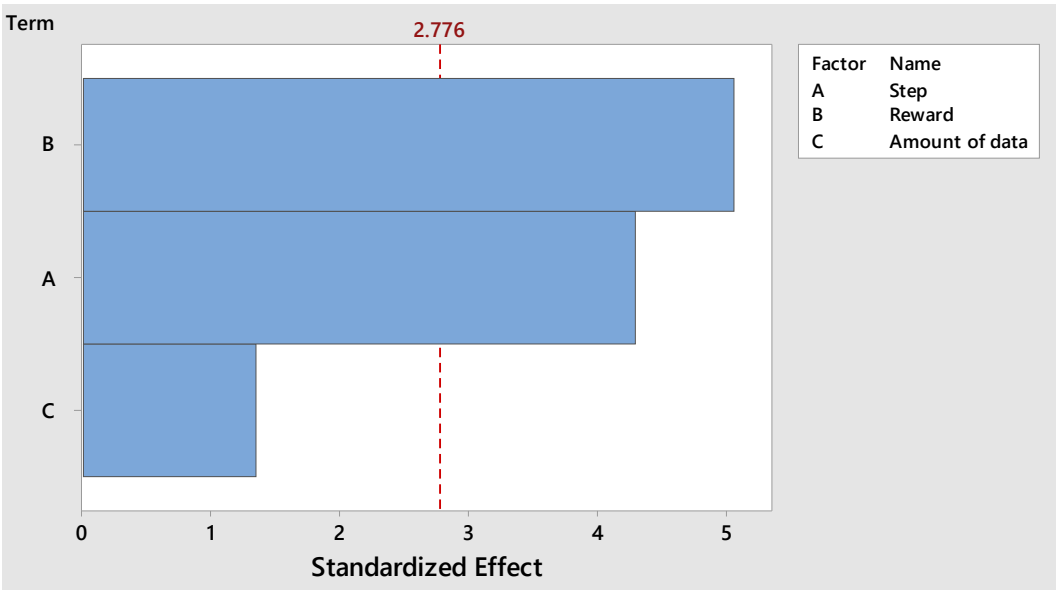
As long as all controllers are already trained and the controller performance is the most important variable in this study, the controller to the next evaluations is chosen just based on the best reward.

In the cube plot of the average rewards in Figure 41, the training that presents the best fitted average reward is the training with data composed by steps, with 50 minutes of data and punishment of -1 as value function. However, in the real data, the best result occurs with the controller trained with null rewards that result in an average reward of 0.8769. This behaviour happened due to the variability of the results and the small influence of the reward. How the trained controllers are already available the controller with reward 0 is chosen to the next evaluations.

4.1.2.2 Four states algorithm

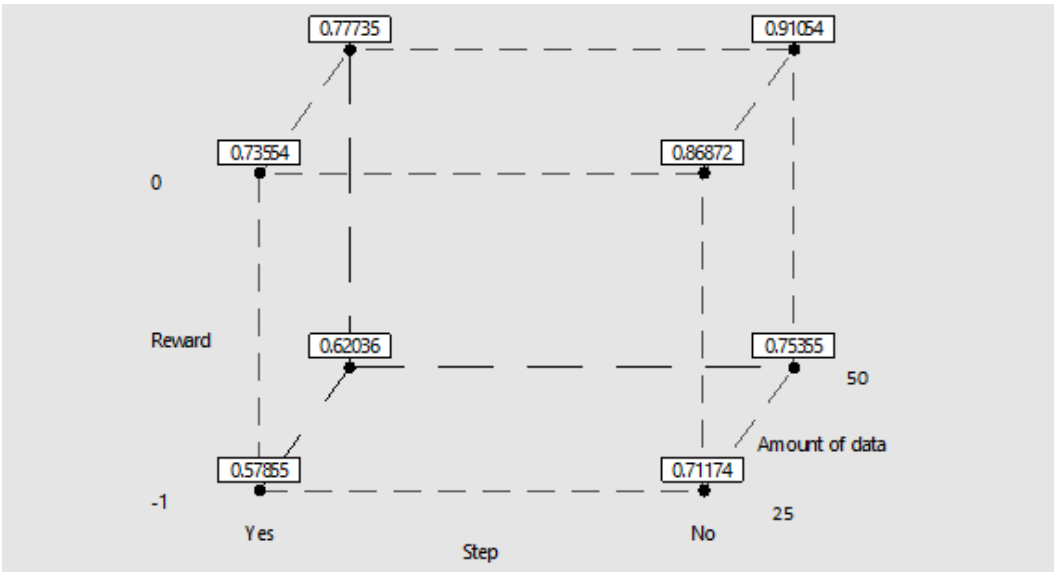
In Figure 44 is shown the Pareto chart of the effects on the average reward of the proposed controller trained without the vehicle velocity as an input. The results can also be confirmed in the Cube plot in Figure 45.

Figure 44 - Pareto Chart of the Standardized Effects on best average reward - 4 states, proposed algorithm α = 0.05.



Source: Elaborated by the author.

Figure 45 - Cube plot of the best average reward - 4 states, proposed algorithm α = 0.05.

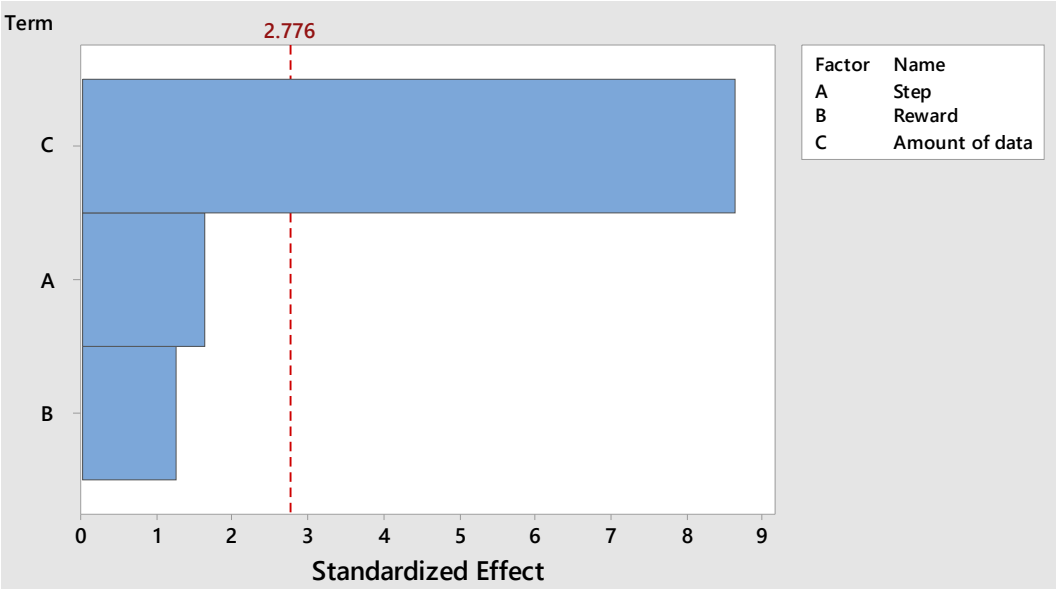


Source: Elaborated by the author.

The result of the Pareto chart showed a significant influence of the step and the reward on the performance of the controller. It indicates a higher dependence on the data quality in the 4 states controller when compared with the 5 states controller. Analyzing the cube plot, the influence indicated by the Pareto chart is reinforced and the best controller is given by the training process with a greater amount of data, null reward and data generated without the presence of steps in the throttle pedal position.

The difference of desirable data characteristic between the 5 and 4 states controllers indicate the necessity of redefining the data every time that the controller architecture is changed.

Figure 46 - Pareto Chart of the Standardized Effects on time - 4 states, proposed algorithm α = 0.05.



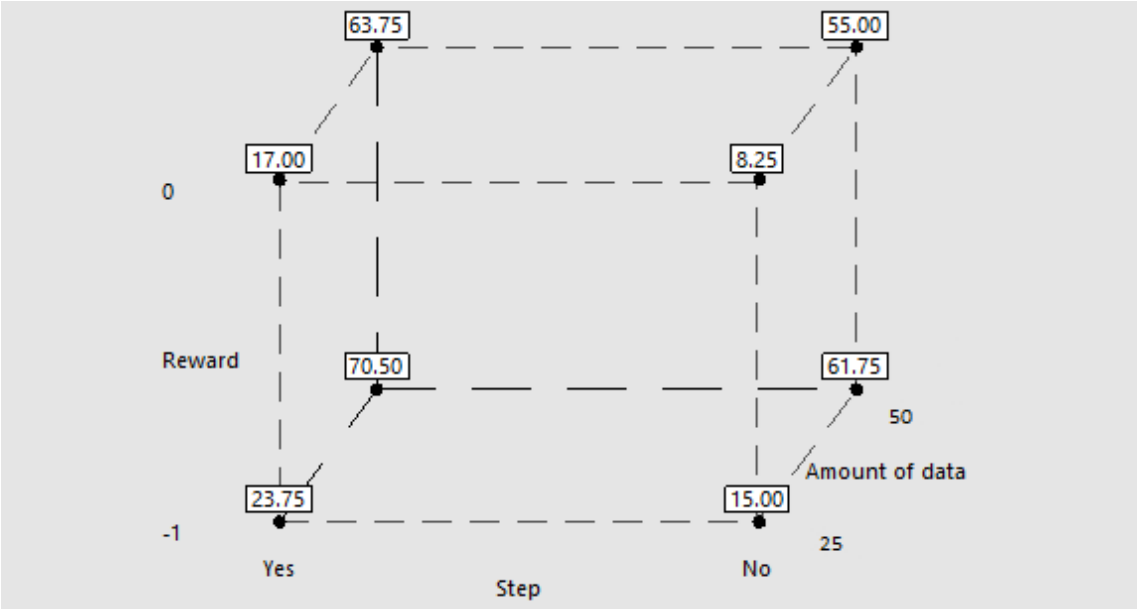
Source: Elaborated by the author.

The Pareto chart of the effects on time showed in Figure 46, exhibit the same correlations as the 5 states controller, in which the training time just suffers significant influence of the amount of data used to train the controller. Comparing the cube plots of 5 and 4 states in Figure 43 and Figure 47 respectively, the results are very similar. Both graphs present the fastest convergence with the training process without steps in the input, a smaller amount of data and null reward. In both cases, the slowest converge occurs in the opposite corner of the cube. The 4 states network also presented a slightly higher time to converge when compared with the 5 states algorithm.

As long as all the controllers are already trained and available, the training time was not taken into account to choose the controller to the next evaluations. In this way,

the applied controller is trained with 50 minutes of data collection, null reward and data generated without the presence of steps in the throttle pedal position.

Figure 47 - Cube plot of the time - 4 states, proposed algorithm α = 0.05.



Source: Elaborated by the author.

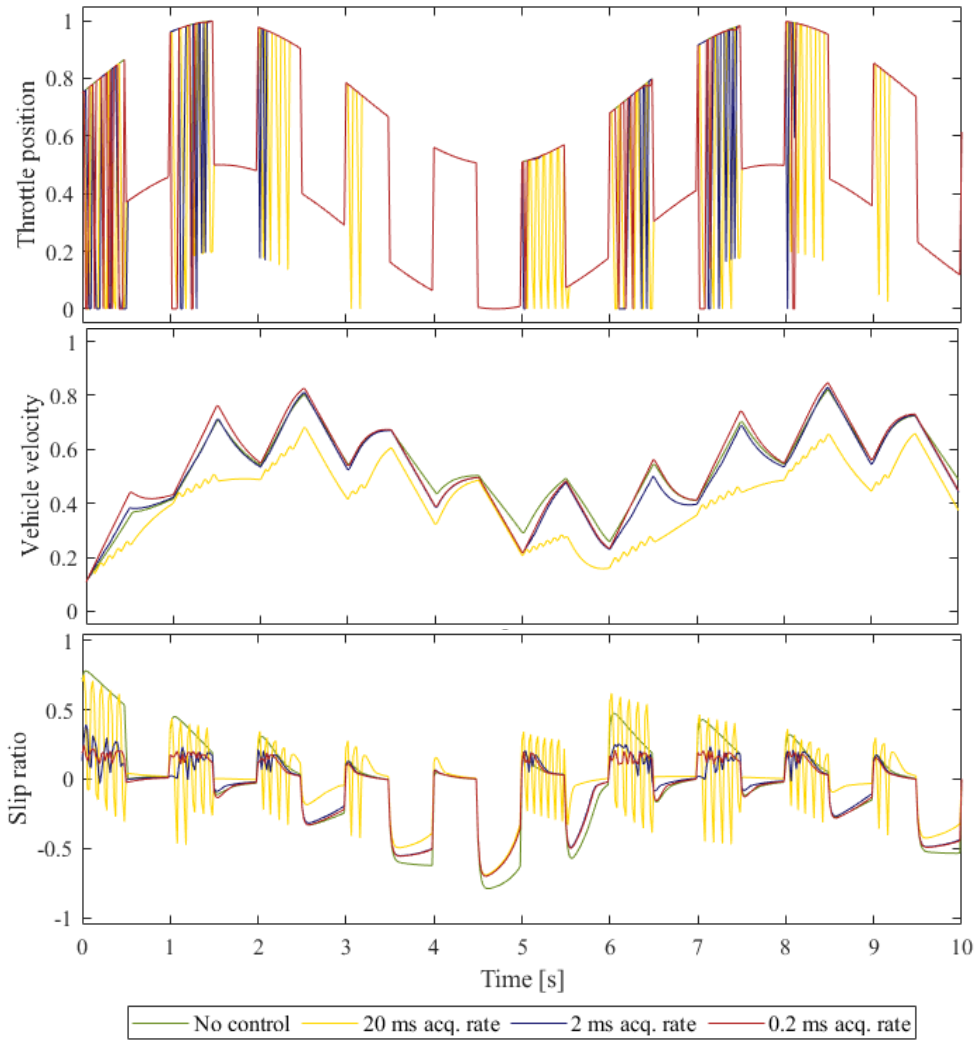
4.2 CLASSICAL CONTROLLER LIMITATIONS

The influence of the acquisition rate on the slip ratio of the system with the classical controller on a dry asphalt ground is displayed in Figure 48 together with the output signal of the controller and the vehicle velocity.

The long-time between the measurement and the action in the larger time step present a high variation of the slip ratio. In this condition, the slip ratio reached similar values as the no control condition indicating a poor behaviour of the control. Evaluating the velocity is possible to see that the vehicle acceleration is completely compromised by the controller, indicating that the classical controller can not work adequately in so large time steps.

With the reduction of the acquisition rate, the behaviour of the controller improves considerably, reaching very smoothly control of the slip ratio already with 2 ms of the acquisition rate. In 2 ms and 0.2 ms acquisition rates, the increase of the velocity also presents an improvement when compared with the system without control, indicating a good behaviour of the control and a consequent improvement of safety and comfort to the vehicle occupants.

Figure 48 - Influence of the acquisition rate in the classical control on the dry asphalt.

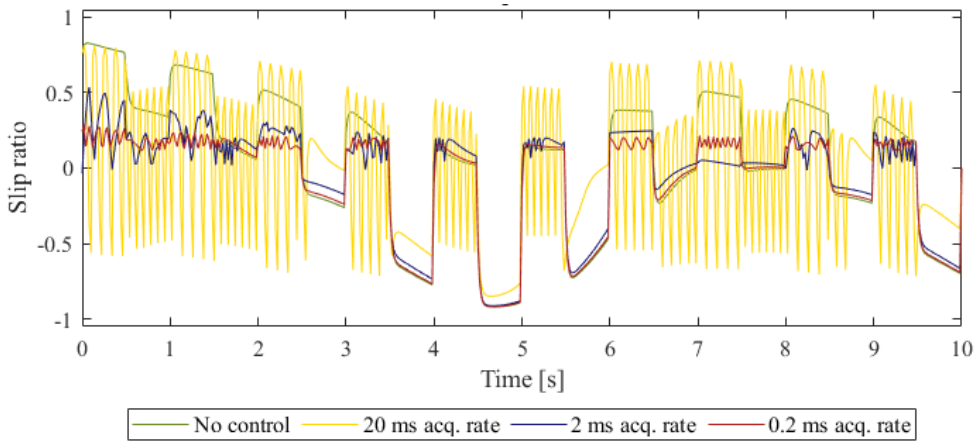


Source: Elaborated by the author.

In Figure 50 is shows the evaluation of the controller behaviour on the snow. In this case, the results are more problematic to the 20 ms acquisition rate due to the faster increase of the slip ratio in the very low friction surface. The reduction of the time step increases significantly the quality of the results reaching a good controller of the slip ratio mainly on the 0.2 ms of acquisition rate. However, the real implementation of sensors that processes the data in this time steps is improbable.

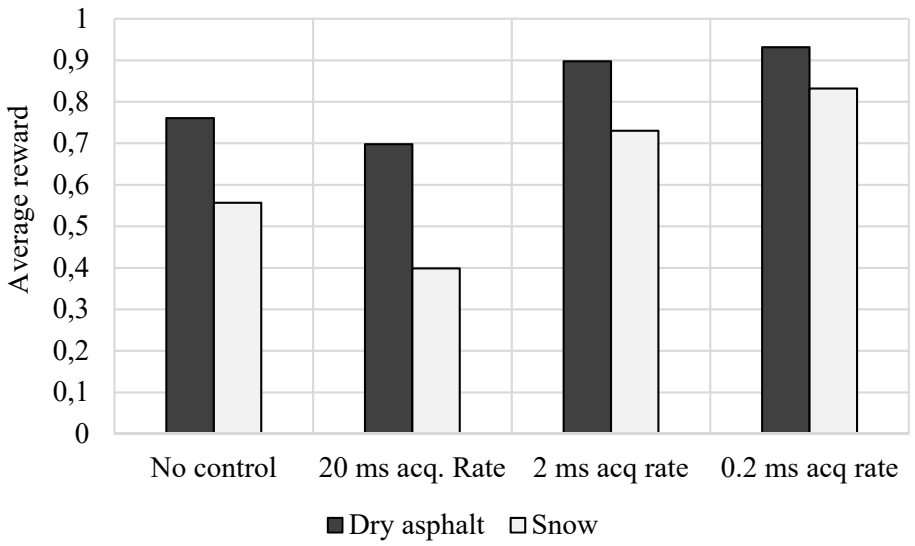
The average reward calculated over the data on both grounds is presented in Figure 50. As expected by the calculation form of the reward, the snow floor always has worse results than the asphalt. In both floors, the system without a controller shows a better performance than the classical controller with 20 ms of acquisition rate, while the controllers with smaller time steps present better average rewards.

Figure 49 - Influence of the acquisition rate in the classical control on the snow.



Source: Elaborated by the author.

Figure 50 - Influence of the acquisition rate on the reward.



Source: Elaborated by the author.

Despite the limitation of the classical controller to operate in 20 ms of acquisition rate, the available data on the real environment just can be run with this time step. In this way, the present evaluation is important to understand the control limitation, but the 20 ms controller is applied to the next studies.

4.3 TESTS OF THE CONTROLLERS

Defined the best parameters to each controller, this step of the work intended to compare the developed controllers in different environments, as different inputs and

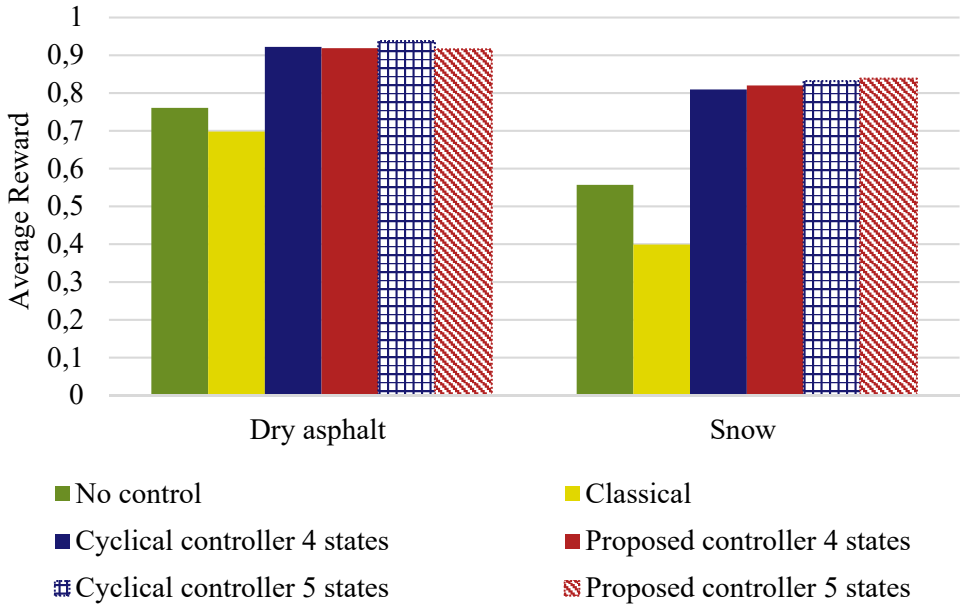
floors. The next subsection presents the results of the comparison of the controllers on the trained conditions, aiming to understand the applicability of the controllers when enough data is available to describe all possible situations of the environment.

To resume the work all the results were evaluated based on the mean reward, but the complete data of slip ratio, throttle pedal position, vehicle velocity and reward are available in Appendix C.

4.3.1 Trained conditions

On the trained conditions, all the reinforcement learning algorithms present better performance than the classical controller and the system without a controller. The controllers that take the velocity of the vehicle into account have slightly better performance as shown in Figure 51.

Figure 51 - Comparison of the average reward in trained conditions.

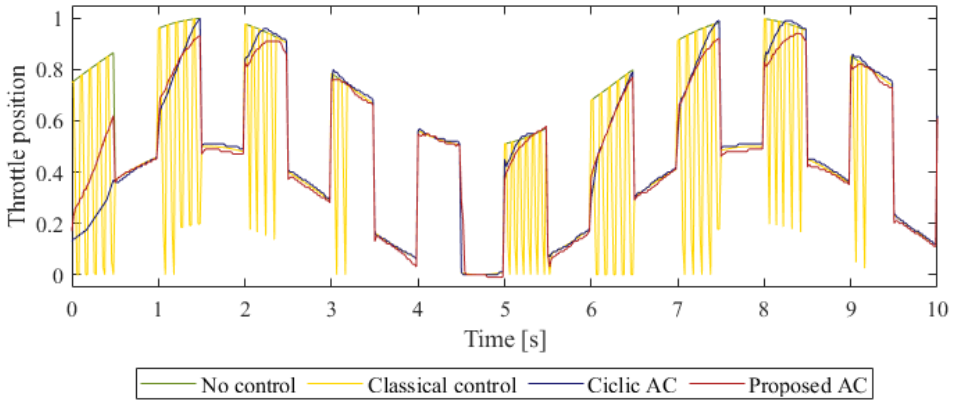


Source: Elaborated by the author.

The cyclical and proposed algorithms show very similar results alternating the better performance on different floors and number of states. This similar quality of proposed and cyclical algorithms show a successful application of the proposal controller in trained conditions, permitting to capture a large amount of data and train the network in an off-line station.

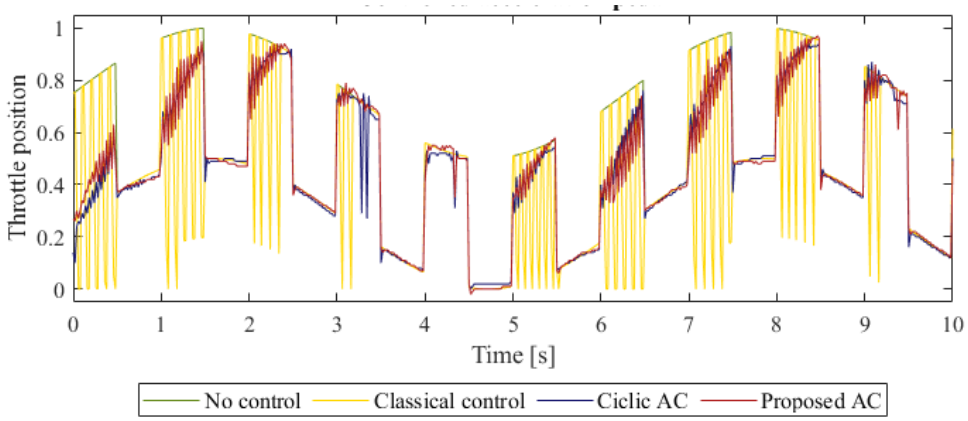
The outputs of the controllers that apply 5 states are shown in Figure 52, while the outputs obtained in the 4 states controllers are displayed in Figure 53. The slight difference between the 4 and 5 states performance can be explained by the high-frequency variation present on the 4 states controllers that punctually decrease the reward values. As can be compared in Figure 52 and Figure 53 the 5 states algorithm presents a very smoothly change of the desired acceleration pedal, while the 4 states controller of Figure 53 shows a small variation of the throttle position on both the cyclic and proposed training algorithms.

Figure 52 - Output signal of 5 states controller on dry asphalt.



Source: Elaborated by the author.

Figure 53 - Output signal of 4 states controller on dry asphalt.



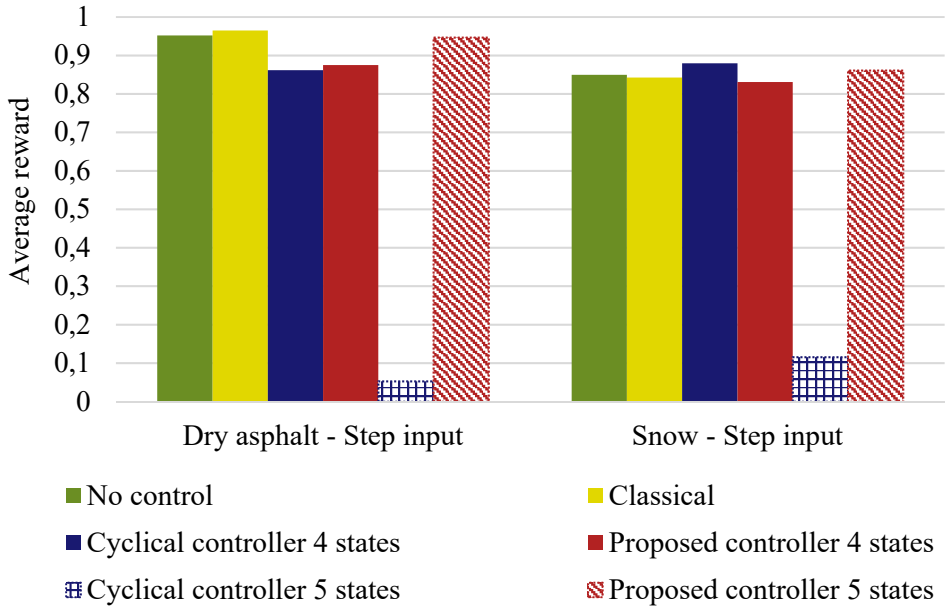
Source: Elaborated by the author.

The remainder captured data used to calculate the average rewards of this subsection are available in Appendix C in Figure 73, Figure 74, Figure 75 and Figure 76.

4.3.2 Different acceleration patterns

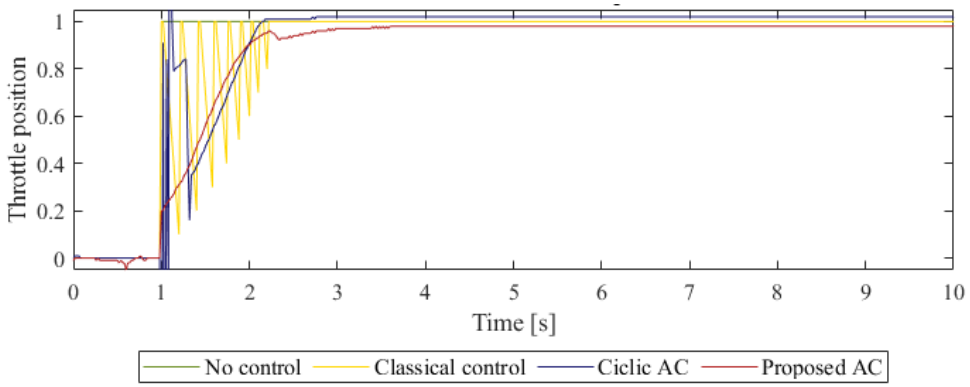
The results of the average reward on the controllers when a step input is given as driver accelerator pedal are shown in Figure 54. In the step manoeuvre, the classical controller and the system without control present good reward values, as long as a large part of the time the output is exactly the expected, in this case, 100% of the pedal position.

Figure 54 - Comparison of the average reward of different acceleration patterns.



Source: Elaborated by the author.

Figure 55 - Output signal of 5 states controller on dry asphalt with step signal in the accelerator pedal.



Source: Elaborated by the author

Despite the adequate average results of the 4 states algorithms, the single controller that presents an expected behaviour is the proposed algorithm with 5 states as shown Figure 55, all other algorithms show difficulty to handle with low velocities. This

behaviour is probably justified by the step multiplication applied just on this controller. The step multiplication changes the pattern of accelerator pedal position during the training, including a step to permit the generation of states with low velocities.

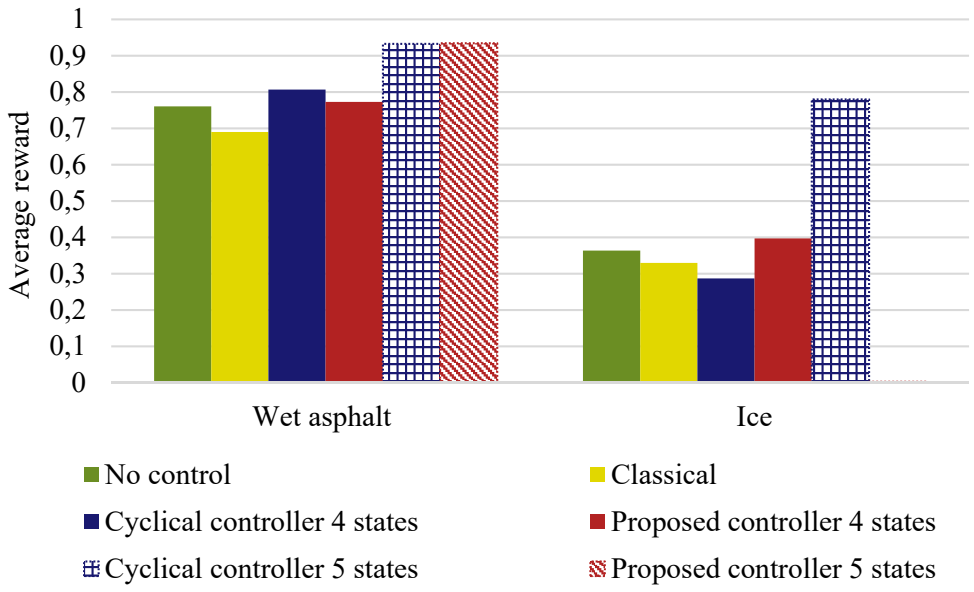
The very low average reward of the cyclic algorithm with 5 states can be justified by the great period of time that the output was bigger than the maximum value of the range. This problem and the small variation of the proposed controller in the region where the input acceleration pedal is null can be solved by the limitation of the range in a real application.

The remainder captured data used to calculate the average rewards of this subsection are available in Appendix C in Figure 77, Figure 78, Figure 79 and Figure 80.

4.3.3 Different floor patterns

In Figure 56 is compared the average reward of manoeuvres on wet asphalt and ice using controllers trained on dry asphalt and snow.

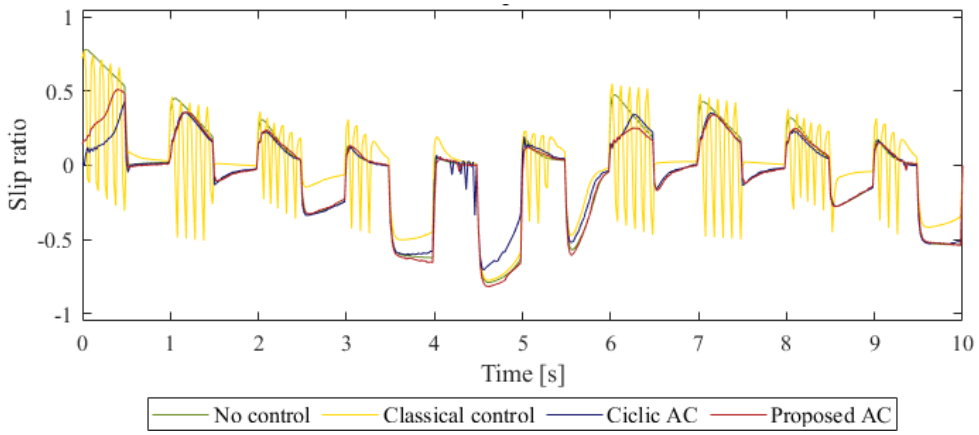
Figure 56 - Comparison of the average reward of different floors patterns.



Source: Elaborated by the author.

The results present a good behaviour of the controllers that take the vehicle velocity into account when the vehicle is driven on the wet asphalt, while the 4 states controller present high values of slip as shown in Figure 57.

Figure 57 - Slip ratio of 4 states controllers on wet asphalt.

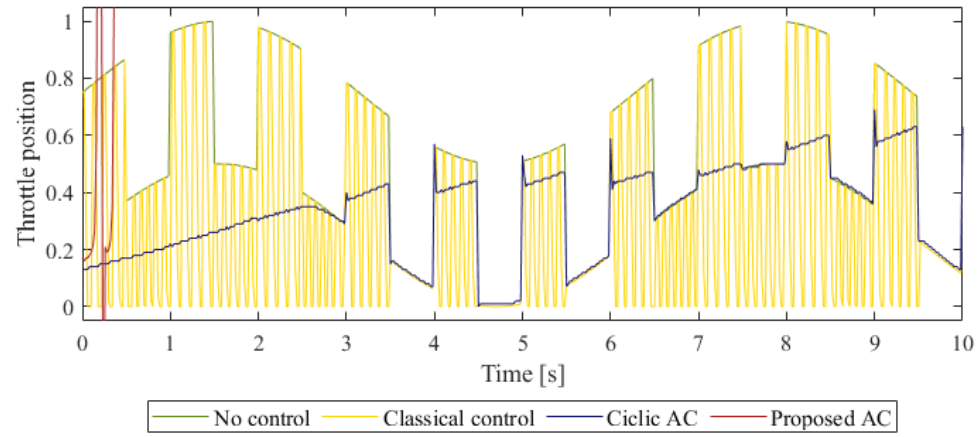


Source: Elaborated by the author.

On the ice ground it is possible to note the 5 states cyclic algorithm that reaches a very high average reward. However, the proposed algorithm seems to have a limitation to deal with behaviours that extrapolate the learning environment, as long as the wet asphalt has a friction coefficient between the dry asphalt and the snow, while the ice ground has friction out of this range.

In Figure 58 it is presented the exemplary behaviour of the cyclical controller on an ice surface, while the proposed algorithm presents a completely undesirable characteristic with out of range values most part of the time. This characteristic is probably originated from non-trained states, where the algorithm does not have a reference of the desired output.

Figure 58 - Output signal of 5 states controller on ice.



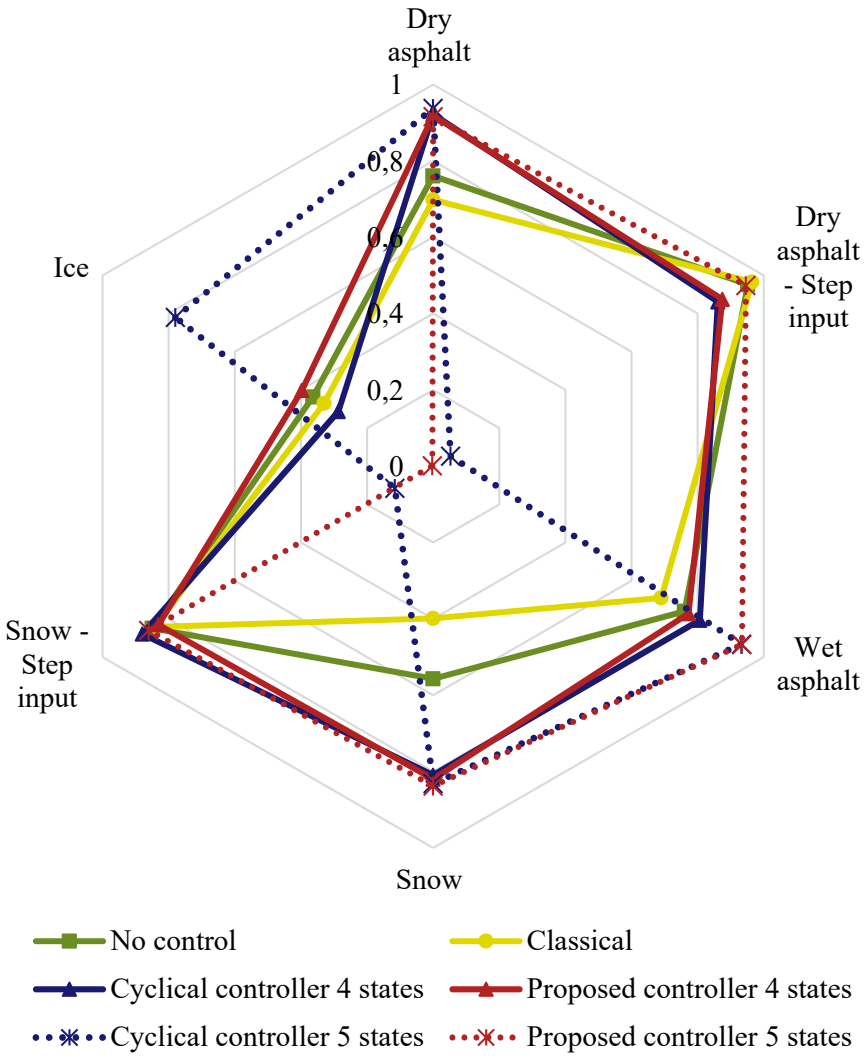
Source: Elaborated by the author.

The remainder captured data used to calculate the average rewards of this subsection are available in Appendix C in Figure 81, Figure 82, Figure 83 and Figure 84.

4.3.4 Final comparison of the controllers

To summarize all the comparison tests is presented in Figure 59 a compilation of all average rewards.

Figure 59 - Comparison of the average reward of all simulated conditions.



Source: Elaborated by the author.

Evaluating the graph, the step input returns better reward even in the classical control or in the algorithm without control, while the sinus signal combined with steps

presents a reduction of the reward with the decreasing of the friction. The classical controller always presents an inadequate response with similar or worse rewards as the system without a controller.

All the reinforcement learning algorithms presented an adequate behaviour on the trained floors, however, the 4 states controllers presented unsatisfactory results in the no trained scenarios as the dry asphalt with step input or the ice floor. The 5 states algorithms present better rewards, but both the cyclical and the proposal algorithms presented occasional problems.

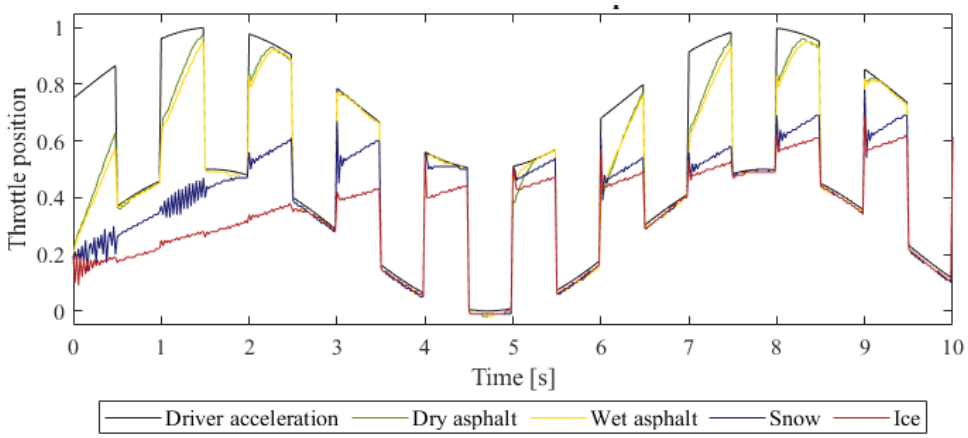
The cyclical controller with 5 states showed limitations on the step inputs, but the range problem showed by this algorithm can be easily solved by the application of a limit for the outputs. On the other hand, the proposed algorithm with 5 states presents a limitation on the extrapolation of the floor data.

As long as the 4 states algorithms showed limitations on the predictability and the cyclic algorithm cannot be applied on the real vehicle, the proposed algorithm with 5 states is evaluated during the next tests. Since this algorithm show limitation with the extrapolation, the next subsection presents a complete interpolated experiment.

4.3.5 Interpolation test

The evaluation of the proposed controller with 5 states trained in ice and dry-asphalt show very prosperous results with smoothly full control of the system as shown in Figure 60.

Figure 60 - Output of the 5 states controller trained on ice and dry asphalt.

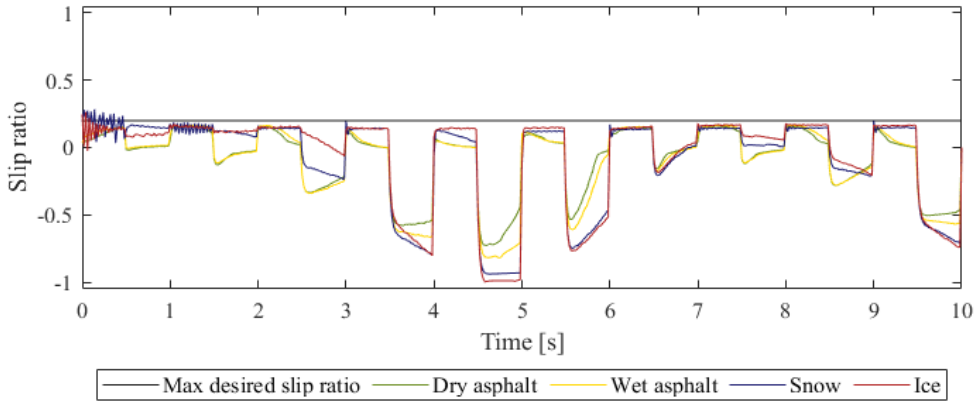


Source: Elaborated by the author.

The slip results present in Figure 61 indicate that in all floors the controller was able to keep the slip around the desired value with a small number of oscillations when the sinus with steps signal is evaluated. Comparing the output of the controller and the slip ratio is possible to check that the same network is able to change the inclination of the accelerator pedal curve to keep the slip ratio below the desired value.

This result indicates a prosperous implementation of the DTNAC, as long as just extreme characteristics need to be trained to generate a controller able to interpolate intermediary states.

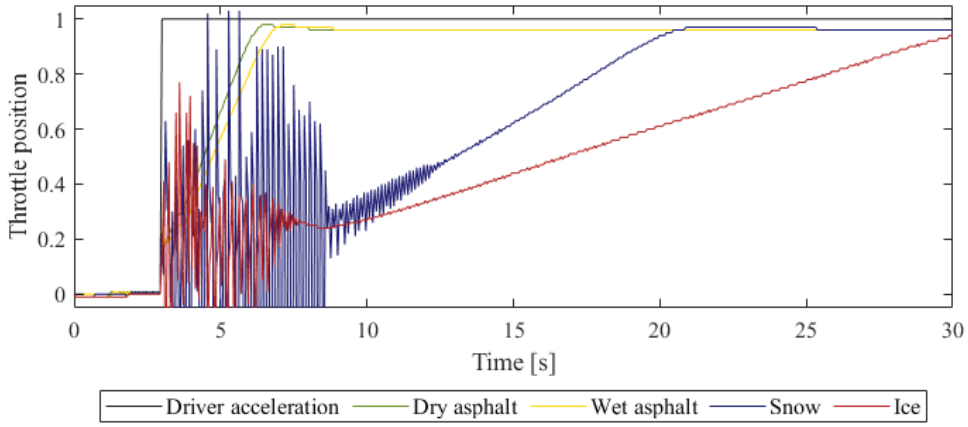
Figure 61 - Slip ratio of the 5 states controller trained on ice and dry asphalt.



Source: Elaborated by the author.

However, when the step input is evaluated, the snow and ice grounds presented an unexpected behaviour in low velocities as show in Figure 62.

Figure 62 - Output of the 5 states controller trained on ice and dry asphalt and simulated with step input.



Source: Elaborated by the author.

This oscillation behaviour may be explained by the physical limitation in low velocities where the difference between the vehicle velocity and the wheel velocity is

small. Other possibilities are a small number of states in low velocities or the restriction of the random input to describe this kind of behaviour.

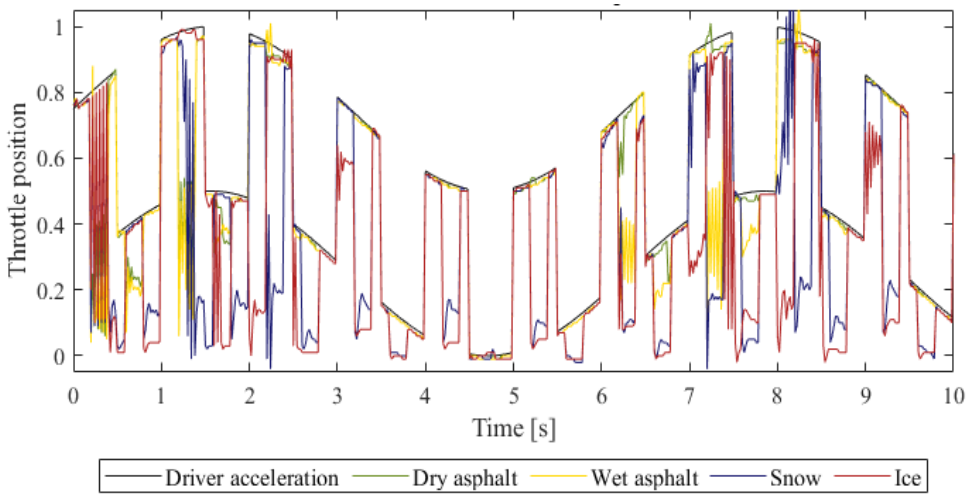
The larger time taken by the vehicle to decrease its velocity on the ice floor could be a possible limitation due to the smaller number of states in low velocity, which would impede the network from learning the correct behaviour in this state. However, the increasing of the multiplication step period in two times did not increase the data quality.

Considering just the sinus with steps as input, which did not present problems with the low-velocity conditions, the implementation of the controller in the real vehicle has a second limitation in terms of acquisitions rates.

4.3.6 Different acquisition rates

To apply the controller on the real vehicle it is necessary that the controller works with different acquisition rates, due to the sensors applied in the vehicle. The controller output with different acquisition rates and the respective wheel velocity and slip are shown in Figure 63 and Figure 64.

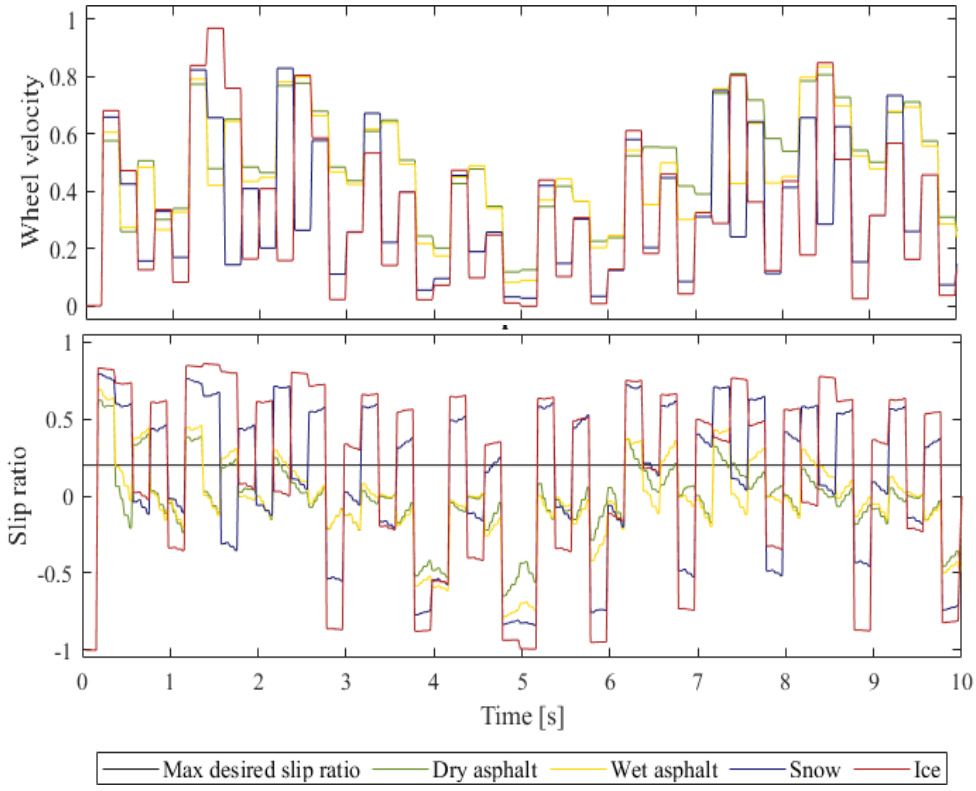
Figure 63 - Output of the 5 states controller trained on ice and dry asphalt with different acquisition rates.



Source: Elaborated by the author.

As given in Figure 63 the output of the controller had a high number of fluctuations that indicate an inadequate work of the controller. Evaluating the slip and the wheel velocity in Figure 64 it becomes clear that the large time step of the wheel velocity sensor implicates in a behaviour similar to the classical algorithm where the large time between the data update entail in the wrong decision by the controller.

Figure 64 - Velocity and slip of the 5 states controller trained on ice and dry asphalt with different acquisition rates.



Source: Elaborated by the author

When the average of the wheel velocity is low, the calculation results in a small slip ratio, that permits the increase of the output of the controller. However, the high accelerator pedal position increases the slip of the wheel, which is calculated just after 200 ms. Past this time the high slip ratio indicates the reduction of the throttle pedal position, thus the controller is not reading the real behaviour of the vehicle but a past velocity, impeding the correct correlation of slip and throttle.

When applying different acquisition rates the vehicle stops behaving like a Markov system impeding reinforcement learning algorithm to be applied. To the real implementation, an update of the kart hardware or the use of networks with a time-depended input may be possible solutions. In the case of time depending controller, the input of the last ten data points could provide to the network the necessary information about the vehicle behaviour. With this network update, the captured data should be enough to completely describe the vehicle state, returning the Markov characteristic.

CONCLUSIONS

The actor-critic reinforcement learning showed itself as a promising controller to many activities where the physical modelling is non-linear and complex. This algorithm is able to find the best behaviour by own and define a matrix controller. The obtained results proved that when the system scenario that needs to be controlled is incorporated in the training process both the proposed and the cyclical training process are able to generate an adequate controller. However, in vehicular applications, when the data that represent all the possible states are not available, some restriction can be found as the inaquete behavior of the trained controllers in novel states.

The cyclical training presented difficulties to real applications due to the necessity of executing the real test dozens of times. The obtained controller also had a restriction during the implementation in different scenarios. The proposed DTNAC training process, on the other hand, had a robust application with fewer parameters to be checked and the possibility of collecting the data only once, training the neural network in an off-line station.

With the implemented architecture, the cyclical training algorithm demonstrated better results with null discount factor, which makes the controller unable to predict future behaviours. This behaviour reinforces the possibility of simplifying the value function by the reward function as is made in the proposed algorithm.

The algorithms that implemented all the physical parameters necessary to calculate the slip behaviour, including the vehicle velocity, generated more robust results, which had small dependence of the data quality and quantity to generate an adequate controller. The 5 states algorithm is able to handle different scenarios with smoothly change of the output and small deviations from the desired behaviour. The 4 states algorithms, in contrast, presented a higher dependency on the data quality and quantity beside the high-frequency oscillation, usually present on the controller output. However, when implemented just on trained scenarios the algorithm without velocity work adequately and could simplify the sensor necessity.

By the comparison of all evaluated scenarios, all reinforcement learning controllers had an adequate response on the trained scenarios. However, the proposed controller with 5 states can be highlighted, because it is able to control appropriately the system in varied no-trained conditions as different grounds or accelerator pedal inputs, except on the ice ground. This characteristic is probably derived from the necessity of the

controller to have extreme points of reference, permitting to operate in an interpolation mode. It was proved by the training of a new controller based on ice and dry asphalt tracks to the training process, which was able to manage all the floors with the expected behaviour.

The limitation of all controllers is present in low velocities, where the controllers usually had a completely inadequate behaviour with high oscillations or unexpected increase of the output amplitude. This problem is partially solved on the proposed algorithm by the insertion of data corresponded to low velocity, but on ice and snow floors with step input, this event is still present. The proposed algorithm also could not handle with different acquisition rates impeding the implementation of the controller on the real vehicle without adaptations on the vehicle or controller.

Further researches consist of a deeper evaluation of the influence of the data on the controller, permitting clarify how the controller could be applied in all possible scenarios without the necessity of train the network in all possible states. This understanding would permit the proposed controller with 5 states implementation in different grounds and acceleration patterns without inadequate results.

The second research consists of the development of a time-dependent controller that take into account past data to enable the control of data with different time steps, as occur in the real vehicle. In this case, the last 10 points should be used as input in the network training and in the final controller. Finally, the obtained controller should be implemented on the real vehicle to understand the real physical limitation of the controller and the system.

As a suggestion for other researches, the proposed method can be implemented in other control applications as steer by wire, brake by wire, Automatic Cruise Control or inclusively in no automotive applications since that the implementation of the proposed algorithm is facilitated, permitting to directly train from real data.

REFERENCES

AB Elektronik GmbH, 2018. *Conductive Polymer Potentiometer.* [Online]
Available at: http://www.produktinfo.conrad.com/datenblaetter/450000-474999/453492-da-01-en-LIN__POSITIONSGEBER_LM_10___3M29___1_K.PDF
[Accessed 20 July 2019].

Borrelli, F., Bemporad, A., Fodor, M. & Hrovat, D., 2006. An MPC/Hybrid System Approach to a Traction Control. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY,* 14(3), pp. 541-552.

Bosch Engineering GmbH, 2016. *Yaw Rate Sensor YRS 3,* Abstatt: s.n.

Bose, B. K. & Steigerwald, R. L., 1978. A DC Motor Control System for Electric Vehicle Drive. *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS,* 14(6), pp. 565 - 572.

Braess, H. & Seiffert, U., 2005. *Handbook of Automotive Engineering.* 1st ed. Pennsylvania: SAE International.

Cherry, 2015. *Gear Tooth Speed Sensors.* [Online]
Available at: https://switches-sensors.zf.com/us/wp-content/uploads/sites/7/2012/05/Datasheet_GS1005-GS1007_EN_29_11_17.pdf
[Accessed 20 July 2019].

de Amaral, J. R., 2018. *Improvement of Vehicle Stability Using Reinforcement Learning,* Ingolstadt: Thesis (Master) - THI.

de Amaral, J. R., Göllinger , H. & Fiorentin , T. A., 2018. *Improvement of Vehicle Stability Using Reinforcement Learning.* São Paulo, Proceeding of XV Encontro Nacional de Inteligência Artificial e Computacional.

Den Hartog, J. P., 1948. *Mechanics,* Mineola: McGraw-HilI.

Dixon, J. C., 2007. *The Shock Absorber Handbook.* 1st ed. Chichester: John Wiley and Sons, Ltd.

El Sallab, A., Abdou, M., Perot, E. & Yogamani, S., 2016. *End-to-End Deep Reinforcement Learning for Lane Keeping Assist.* Barcelona, Proceedings of 30th Conference on Neural Information Processing Systems.

El Sallab, A., Abdou, M., Perot, E. & Yogamani, S., 2017. Deep Reinforcement Learning framework for Autonomous Driving. *Electronic Imaging*, 29 January, pp. 70-76.

Engbom, J. et al., 2004. Internal model control for traction control by fuel injection intervention. *IFAC Proceedings,* 37(22), pp. 601-606.

Exide, 2019. *EXIDE DUAL AGM.* [Online]
Available at: https://www.leab.eu/en/products/batteries/exide-dual-agm.html
[Accessed 22 02 2019].

Francois-Lavet, V. et al., 2018. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning,* 11(3-4), pp. 1-136.

Gasbaoui, B. et al., 2017. Behavior PEM fuel cell for 4WD electric vehicle under different scenario consideration. *International Journal of Hydrogen Energy,* Volume 42, pp. 535-543.

Gerstenmeier, J., 1986. *Traction Control (ASR) - An Extension of the Anti-lock Braking System (ABS),* Stuttgart: SAE.

Gillespie, T. D., 1992. *Fundamentals of Vehicle Dynamics.* 1st ed. Warredale: Society of Automotive Engineers.

Hafner, R. & Riedmiller, M., 2011. Reinforcement learning in feedback control. *Machine Learning,* Volume 84, pp. 137-169.

Harmon, M. E., Klopf, A. H. & Baird, C. L., 1996. Reinforcement Learning Applied to a Differential Game. *Adaptive Behavior,* Volume 4, pp. 3-28.

Heim, S., Ruppert, F., Sarvestani, A. A. & Spröwitz, A., 2018. *Shaping in Practice: Training Wheels to Learn Fast Hopping Directly in Hardware.* Brisbane, Proceedings of IEEE International Conference on Robotics and Automation (ICRA).

Hori, Y., Toyoda, Y. & Tsuruoka, Y., 1998. Traction Control of electrical Vehicle: basic Experimental Results Using the Test EV "UOT Electric March". *IEEE Transactions on Industry Applications,* 34(5), pp. 1131-1138.

Hsu, R. C., Liu, C.-T., Lee, W.-M. & Chen, C.-H., 2010. *A Reinforcement Learning Based Power Assisted Method with Comfort of Riding for Light Electric Vehicle.* Taipei, Proceedings of IEEE 71st Vehicular Technology Conference.

Hucho, W.-H., 1998. *Aerodynamics of Road Vehicles.* 1st ed. Warrendale: Society of Automotive Engineers.

Hu, J.-S., Yin, D., Hori, Y. & Hu, F.-R., 2012. Electric Vehicle Traction Control. *IEEE Industry Applications Magazine,* 2014(8), pp. 23 - 31.

Ioffe, S. & Szegedy, C., 2015. *Batch normalization: Accelerating deep network training by reducing internal covariante shift.* Lille, Proceedings of 32nd International Conference on Machine Learning.

Jaritz, M. et al., 2018. *End-to-End Race Driving with Deep Reinforcement Learning.* Brisbane, Proceedings of IEEE International Conference on Robotics and Automation (ICRA).

Jin, L., Ling, M. & Li, J., 2018. Development of a new traction control system using ant colony optimization. *Advances in Mechanical Engineering,* 10(8), pp. 1 - 12.

Kabganian, M. & Kazemi, R., 2001. A New Strategy for Traction Control in Turning Via Engine Modeling. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY,* 50(6), pp. 1540 - 1548.

Khajepour, A., Fallah, S. & Goodarzi, A., 2014. *Electric and Hybrid Vehicles: Technologies, Modeling and Control: A Mechatronic Approach.* 1st ed. Chichester: Wiley.

Khatun, P., Bingham, C. M., Schofield, N. & Mellor, P. H., 2003. Application of Fuzzy Control Algorithms for Electric Vehicle Antilock Braking/Traction Control Systems. *Transactions on Vehicular Technology,* 52(5), pp. 1356 - 1364.

Kim, P., 2017. *MATLAB Deep Learning.* 1st ed. Seoul: Apress.

Kirchner, W. T. & Southward, S. C., 2013. Adaptive vehicle traction control: combined longitudinal and lateral motion. *International Journal of Dynamic Control,* 1(3), pp. 239-253.

LCM Limited, 2019. *MOTORS LEM-200.* [Online]
Available at: https://lynchmotors.co.uk/pdfs/lmc-lem-200.pdf
[Accessed 20 July 2019].

Lee, H. & Tomizuka, M., 2003. Adaptive Vehicle Traction Force Control for Intelligent Vehicle Highway Systems (IVHSs). *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,* 50(1), pp. 37 - 47.

LEM, 2019. *Current Transducer HTA 100 .. 1000-S.* [Online]
Available at: https://www.lem.com/sites/default/files/products_datasheets/hta_100-1000-s.pdf
[Accessed 20 July 2019].

Lillicrap, T. P. et al., 2016. *Continuous Control with Deep Reinforcement Learning.* San Juan, Proceedings of International Conference on Learning Representations.

MathWorks, 2019. *Levenberg-Marquardt backpropagation.* [Online]
Available at: de.mathworks.com/help/deeplearning/ref/trainlm.html
[Accessed 28 05 2019].

Minitab, 2019. *Effects plots for Analyze Factorial Design.* [Online]
Available at: https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/doe/how-to/factorial/analyze-factorial-design/interpret-the-results/all-statistics-and-graphs/effects-plots/
[Accessed 2019 06 06].

Minitab, 2019. *Methods and formulas for the effects plots in Analyze Factorial Design.* [Online]
Available at: https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/doe/how-to/factorial/analyze-factorial-design/methods-and-formulas/effects-plots/#lenth-s-pseudo-standard-error-pse
[Accessed 01 09 2019].

Mnih, V. et al., 2015. Human-level control through deep reinforcement Learning. *Nature,* 518(7540), pp. 529-543.

Mnih, V. et al., 2013. Playing Atari with Deep Reinforcement Learning. *arXiv,* pp. 1-9.

Montgomery, D. C., 2009a. *Introduction to Statistical Quality Control.* 6th ed. Jefferson City: John Wiley & Sons, Inc..

Montgomery, D. C., 2009b. *Design and Analysis of Experiments.* 8th ed. Singapore: John Wiley & Sons.

Nandy, A. & Biswas, M., 2018. *Reinforcement Learning.* 1st ed. New York: Apress.

Naujoks, F., Purucker, C. & Neukum, A., 2016. Secondary task engagement and vehicle automation – Comparing the effects of different automation levels in an on-road experiment. *Transportation Research Part F*, 6 February, pp. 67-82.

Pacejka, H. B., 2002. *Tyre and Vehicle Dynamics.* 1st ed. Oxford: Elvesier.

Paluszek, M. & Stephanie, T., 2017. *MATLAB Machine Learning.* 1st ed. New Jersey: Apress.

Quarteroni, A., Fausto, S. & Gervasio, P., 2014. *Scientific Computing with MATLAB and Octave.* 1st ed. Berlin: Springer.

Radac, M.-B. & Precup, R.-E., 2018. Data-driven model-free slip control of anti-lock braking systems using reinforcement Q-learning. *Neurocomputing,* January, 275(C), pp. 317-329.

Rajamani, R., 2006. *Vehicle dynamics and control.* 1st ed. Troy: Springer.

Raoul, X., Bosch, T., Plantier, G. & Servagent, N., 2004. A Double-Laser Diode Onboard Sensor for Velocity Measurements. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT,* 53(1), pp. 95-101.

Rathmann, S. a. F. R., 2007. Latest Trends in Automotive Electronic Systems - Highway Meets Off-Highway?. *Agricultural Engineering International,* Volume IX, pp. 7-12.

Riedmiller, M., 2005. Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In: *Machine Learning.* Berlin: Springer, pp. 317-328.

Riedmiller, M., Gabel, T., Hafner, R. & Sascha, L., 2009. Reinforcement learning for robot soccer. *Autonomous Robot,* Volume 27, pp. 55-73.

Riedmiller, M., Montermerlo, M. & Dahlkamp, H., 2007. *Learning to Drive a Real Car in 20 Minutes.* Jeju, Proceedings of the FBIT 2007 Conference.

Robert Bosch GmbH, 1998. *Fahrsicherheitssysteme.* 1st ed. Stuttgart: Vieweg.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J., 1986. Learning representations by back-propagating errors. *Nature,* 323(9), pp. 533-536.

Ružinskas, A. & Sivilevičius, H., 2017. Magic Formula Tyre Model Application for a Tyre-Ice Interaction. *Procedia Engineering,* Volume 187, p. 335 – 341.

SAE, 1976. *SAE J670e - Vehicle Dynamics Terminology,* Warrendale: Society of Automotive Engineers.

Sampaio, R. C. B. et al., 2012. A New Control Architecture for Robust Controllers in Rear Electric Traction Passenger HEVs. *TRANSACTIONS ON VEHICULAR TECHNOLOGY,* 61(8), pp. 3441 - 3453.

Samuel, A. L., 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal on Research and Development,* 3(3), pp. 211-229.

Schraufstetter, J., 2018. *Implementierung eines Sensors zur Geschwindigkeitsmessung eines Elektrokarts unter Verwendung des optischen Flusses,* Ingolstadt: Final Work (Bachelor) - THI.

Sevcon, 2002. *MILLIPAK 4QPM CONTROLLER MANUAL,* UK: s.n.

Silver, D. et al., 2014. *Deterministic Policy Gradient Algorithms.* Beijing, Proceedings of the 31 st International Conference on Machine Learning.

Sutton, R. S., 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning,* 1(3), pp. 9-44.

Sutton, R. S. & Barto, A. G., 2017. *Reinforcement learning: an introduction.* 1st ed. Cambridge: The MIT Press.

Sutton, R. S., McAllester, D., Singh, S. & Mansour , Y., 1999. Policy gradient methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems,* Volume 12, pp. 1057-1063.

Taitler, A. & Shimkin, N., 2017. *Learning Control for Air Hockey Striking using Deep Reinforcement Learning.* Prague, Proceedings of 2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO).

Uc-Cetina, V., 2013. A novel Reinforcement Learning Architecture for Continuous State and Action Spaces. *Advances in Artificial Intelligence,* Volume 2013, pp. 1-10.

Wang, J. et al., 2015. Temporal logic motion control using actor–critic methods. *The International Journal of Robotics Research,* 34(10), pp. 1329-1344.

Watkins, C. J. C. H., 1989. *Learning from delayed rewards,* Cambridge: Thesis (Doctor) University of Cambridge.

Wiering, M. & van Otterlo, M., 2012. *Reinforcement Learning State-of-the-Art.* 1st ed. Berlin: Springer.

Williams, R. J., 1992. Simple Statistical Gradient-Following Algorithms for connectionist Reinforcement Learning. *Machine Learning,* Volume 8, pp. 229-256.

Zhao, D., Wang, B. & Liu, D., 2013. A supervised Actor–Critic approach for adaptive cruise control. *Soft Comput,* 17(11), pp. 2089 - 2099.

Zhao, D., Xia, Z. & Zhang, Q., 2017. Model-free Optimal Control based Intelligent Cruise Control with Hardware-in-the-loop Demonstration. *IEEE Computational intelligence magazine,* 12(2), pp. 56 - 69.

APPENDIX A - Go-kart components

Batteries, Figure 17 a): 12V absorbent glass mat batteries. High-current batteries, which reduce the recharge time of up to 50%. The choice of this kind of battery also contributes to avoiding spilling and leaking during the vehicle vibration (Exide, 2019). In the vehicle, four batteries are connected in series to provide 48 V of electrical voltage.

Accelerator potentiometer, Figure 17 b): The identification of the driver acceleration intention is given by a conductive polymer potentiometer. The sensor used is an "LM10" model from the "ab" brand that presents a linear motion with 10 mm of effective travel (AB Elektronik GmbH, 2018). The sensor is connected to the accelerator pedal, and according to the pedal position send at each 20 ms a signal to the Millipak 4QPM controller that uses a Pulse-width modulation (PWM) signal to control the motor (Sevcon, 2002).

Vehicle velocity sensor, Figure 17 c): It is an Optical Flow low-cost sensor proposed by Schraufstetter (2018). In this sensor, an internal microcontroller based in successively recorded images of the ground calculates the vehicle speed. The longitudinal and lateral speeds are then transmitted via a CAN bus interface to the control of the kart. The optical flow sensor has a standard deviation smaller than 1 km/h for all range 0-50 km/h and permits a 50 ms sample time (Schraufstetter, 2018).

Longitudinal acceleration, Figure 17 d): Automotive yaw rate sensor of Bosch Company. The sensor has a measurement range of ± 4.1g and an absolute resolution of 0.01g. The sensor itself has a low-pass filter of 15 Hz (Bosch Engineering GmbH, 2016). The obtained data is sent by CAN communication to the controller system.

Table 6 - LEM 200-127 technical data.

Peak Power	Peak Efficiency	Peak Current	Rated Power	Rated Speed	Rated Current	Rated Torque
16.08 kW	89 %	400 A	8.55 kW	2592 Rpm	215 A	31.5 Nm

Source: LMC Limited (2019).

Vehicle motor, Figure 17 e): The vehicle powertrain is composed by a 48V Lynch LEM 200-127 brushed DC motor. The motor data is present in Table 6 (LCM Limited, 2019). The motor is connected with the wheel axle by a belt and pulleys with a

transmission rate of four times. The relation reduces the vehicle velocity to a safe area and improves the vehicle wheel torque.

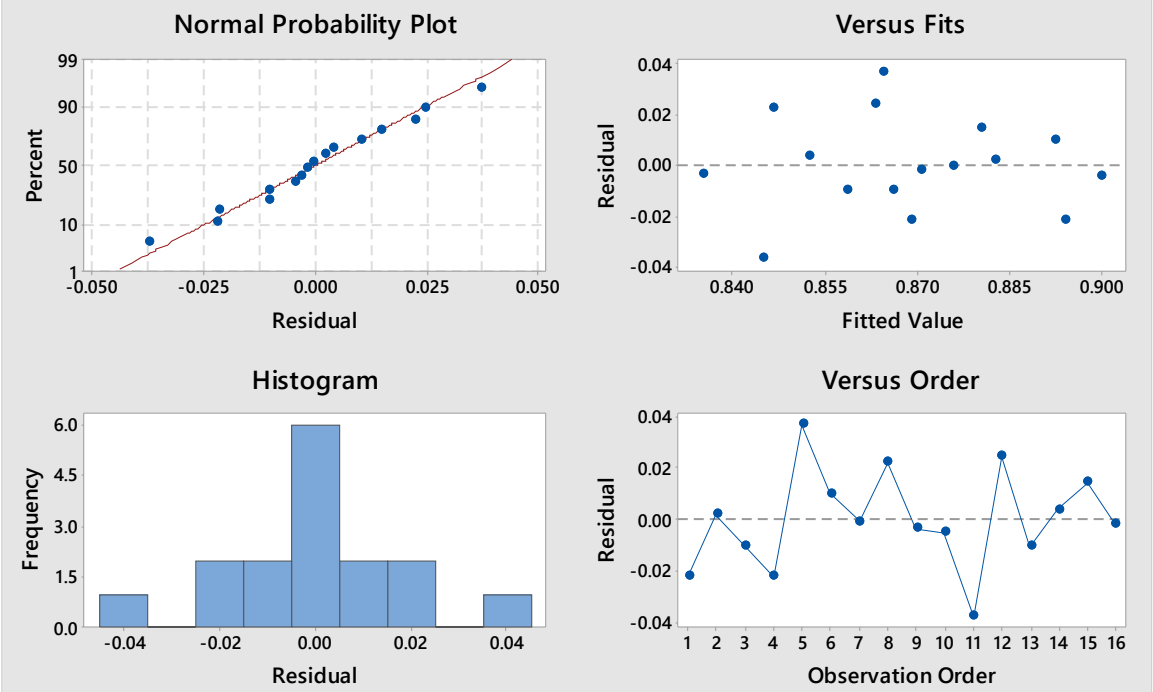
Wheel velocity, Figure 17 f): Gear tooth speed sensor model GS1005 from Cherry is applied at the rear axle of the vehicle to measure the wheel speed. The sensor uses the Hall Effect, permitting to measure from near zero velocities up to 15 kHz (Cherry, 2015). The gear has 29 teeth with tooth width of 10.9 mm and a distance between teeth of 10.4 mm. The wheel speed sensor has a sample time of 200 ms.

Motor current, Figure 17 g): Measured by a current transducer of LEM, model HTA 400-S. The sensor uses the Hall Effect to measure the motor current. The measurable range is ± 1000 A (LEM, 2019). The current transducer data is accessed at each 20 ms by the vehicle controller.

Telemetry System, Figure 17 h): All the data available on the vehicle is sent via CAN bus communication to the vehicle controller, which is connected to a telemetry system. A Holybro V.3 radio module composes the telemetry.

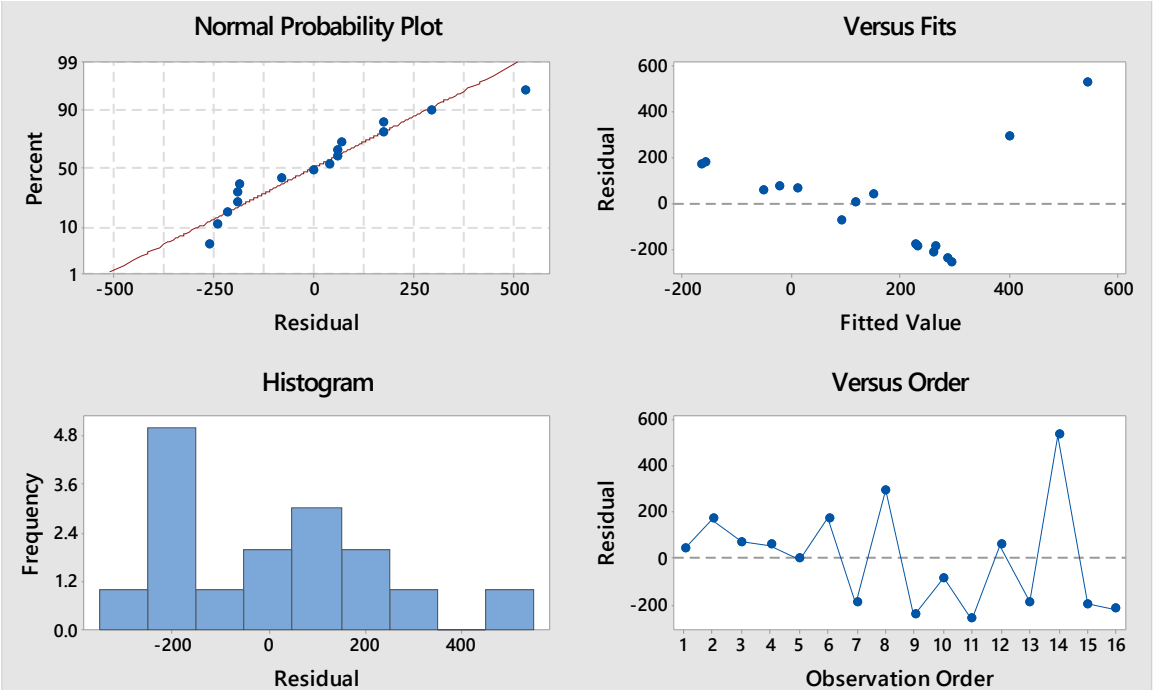
APPENDIX B - Residual Plots of DOE

Figure 65 - Residual plot for best average reward in 5 states cyclic algorithm.



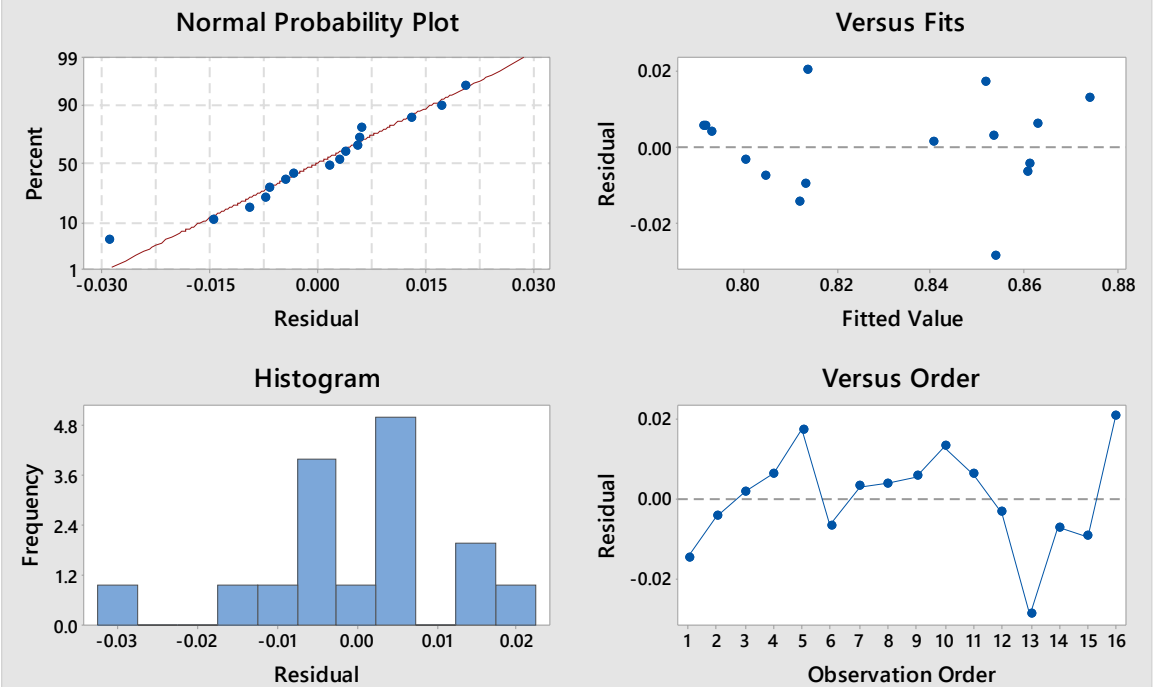
Source: Elaborated by the author.

Figure 66 - Residual plot for time in 5 states cyclic algorithm.



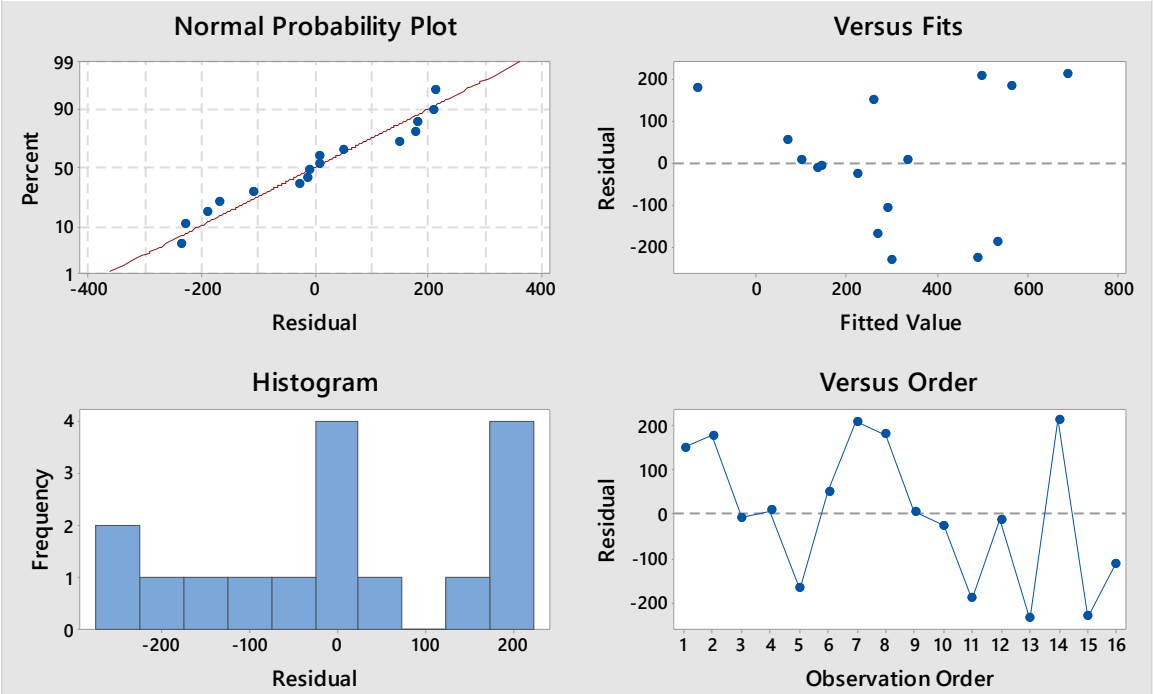
Source: Elaborated by the author.

Figure 67 - Residual plot for best average reward in 4 states cyclic algorithm.



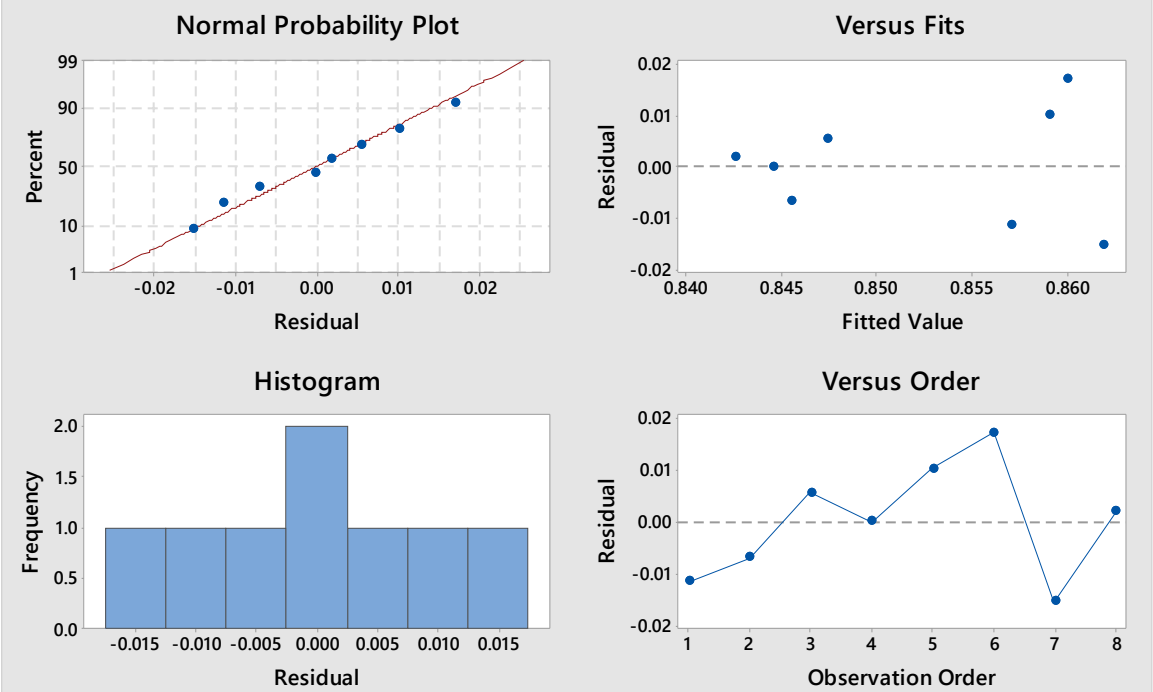
Source: Elaborated by the author.

Figure 68 - Residual plot for time in 4 states cyclic algorithm.



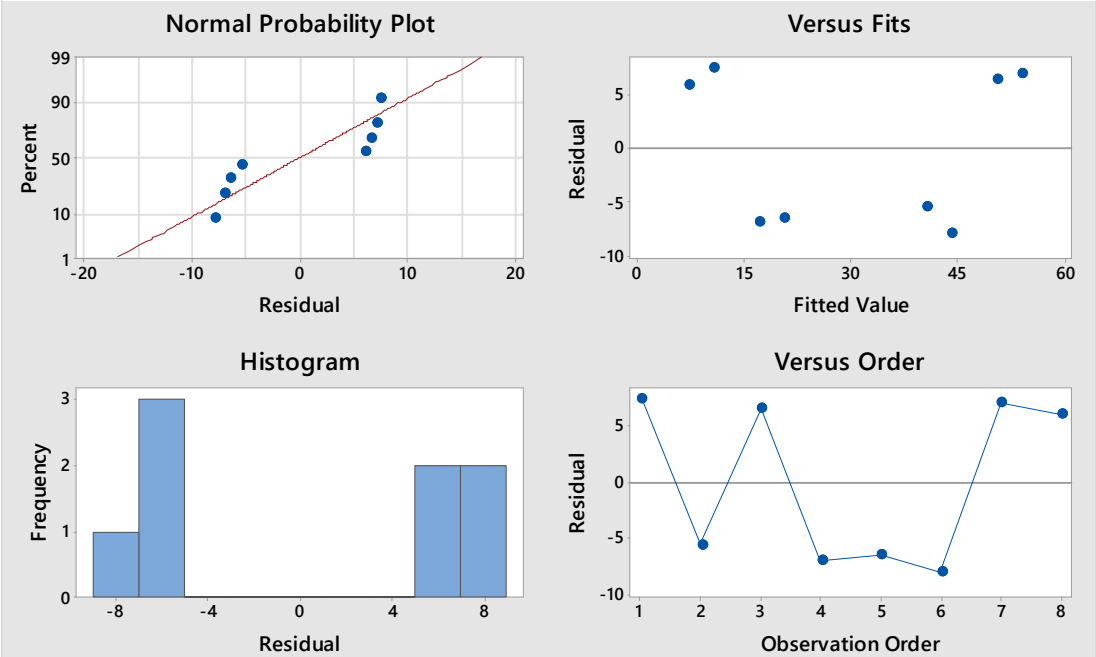
Source: Elaborated by the author.

Figure 69 - Residual plot for average reward in 5 states proposed algorithm.



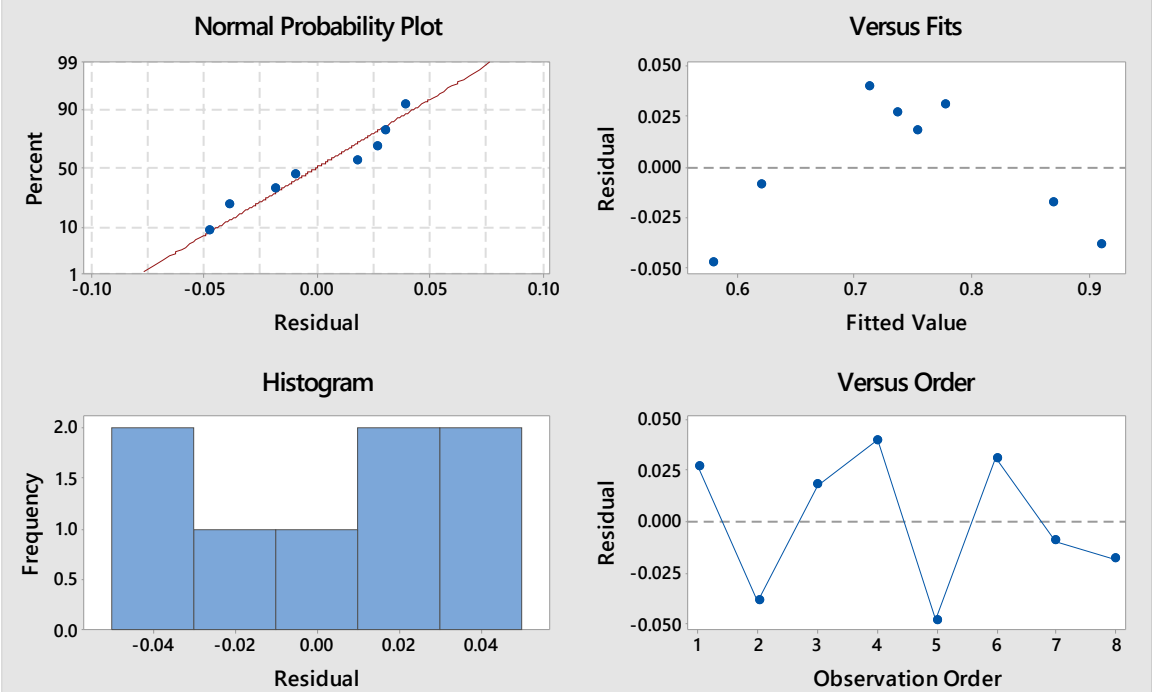
Source: Elaborated by the author.

Figure 70 - Residual plot for time in 5 states proposed algorithm.



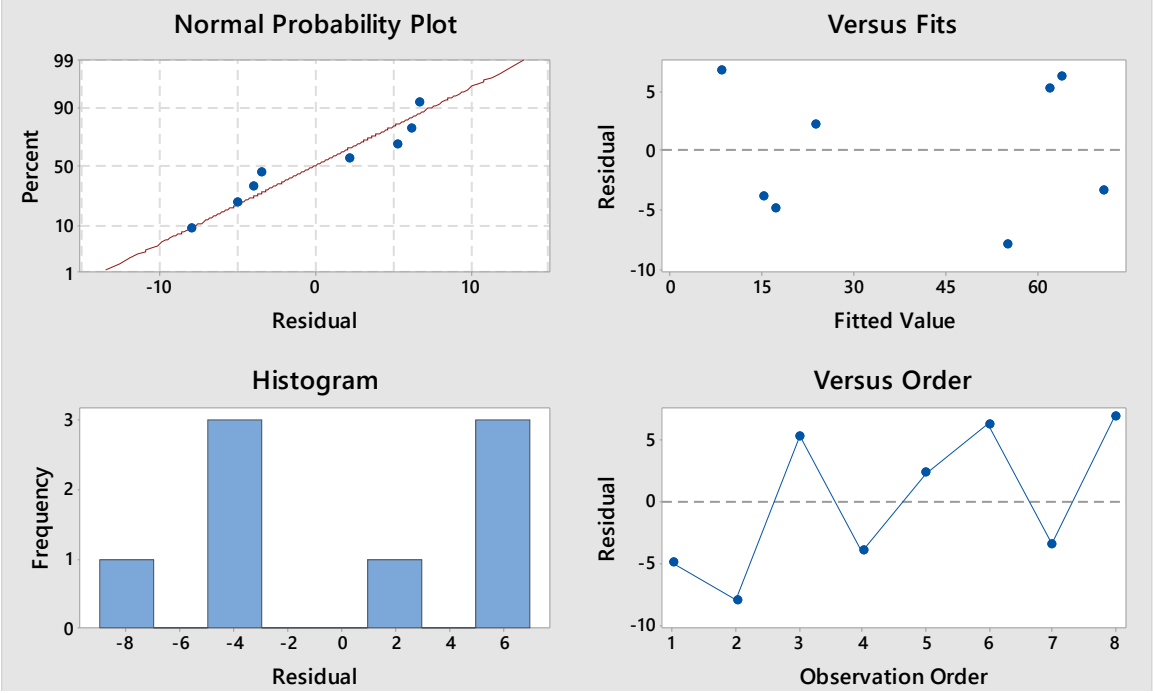
Source: Elaborated by the author.

Figure 71 - Residual plot for average reward in 4 states proposed algorithm.



Source: Elaborated by the author.

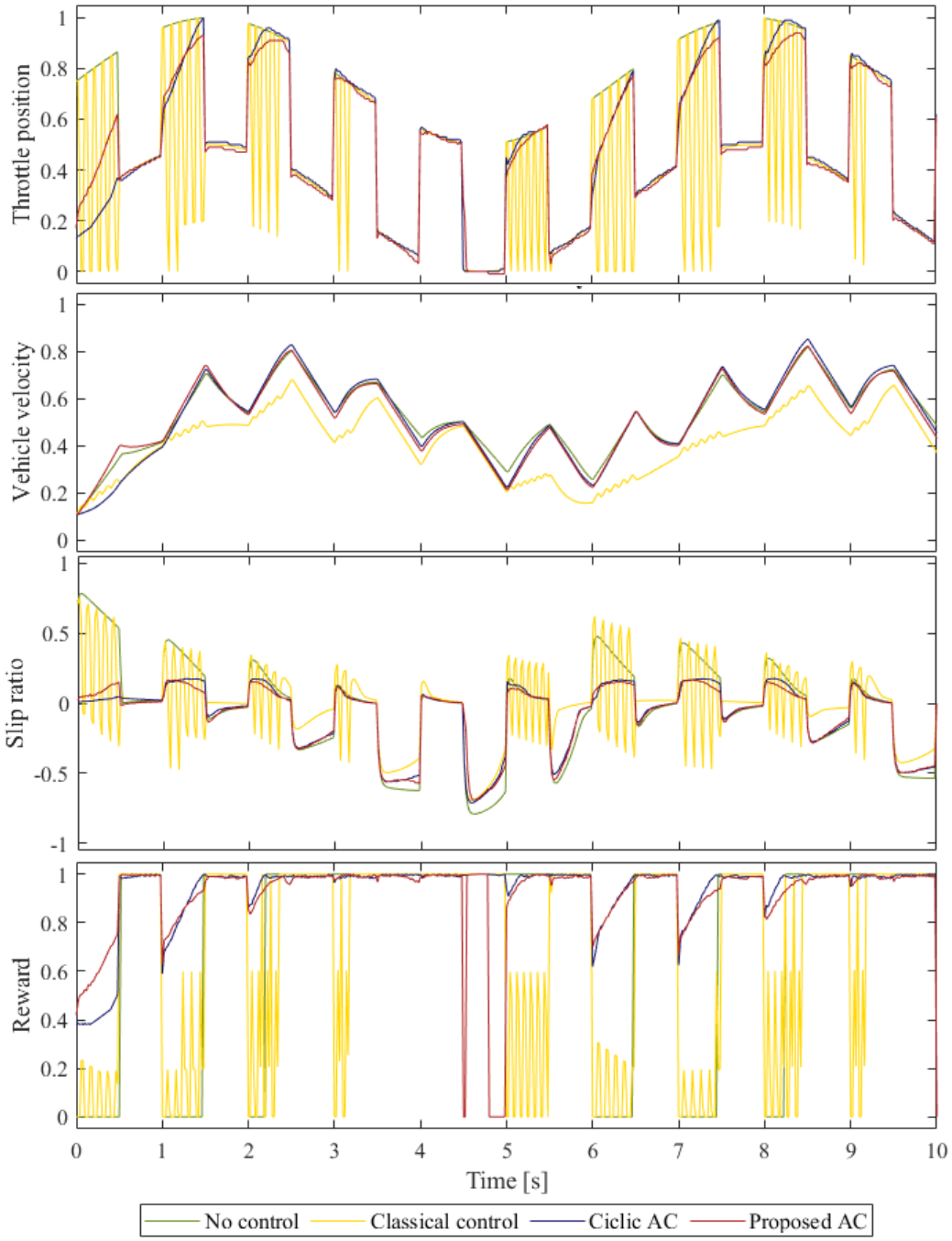
Figure 72 - Residual plot for time in 4 states proposed algorithm.



Source: Elaborated by the author.

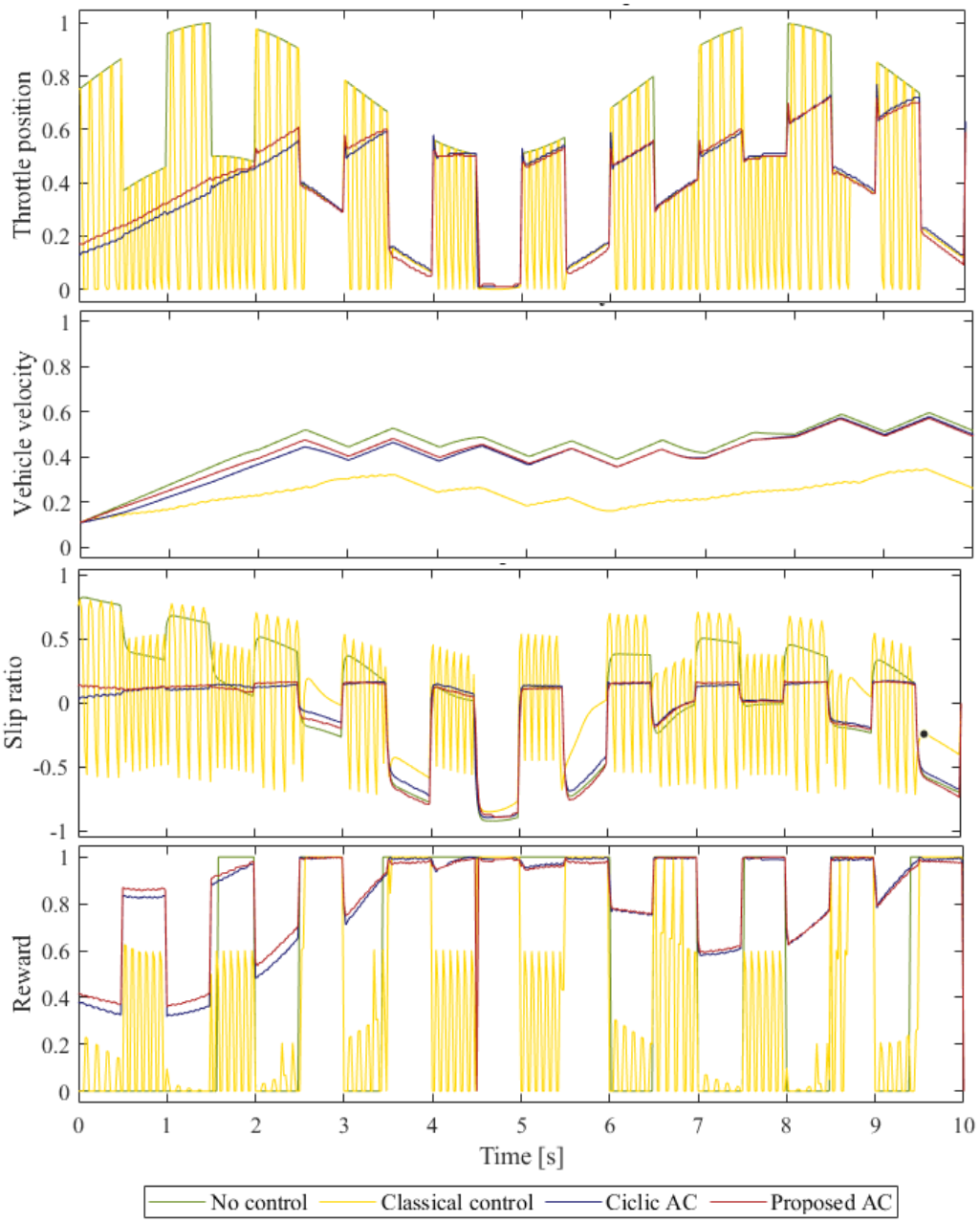
APPENDIX C - Algorithms comparison

Figure 73 - Simulation results - Dry asphalt - 5 states.



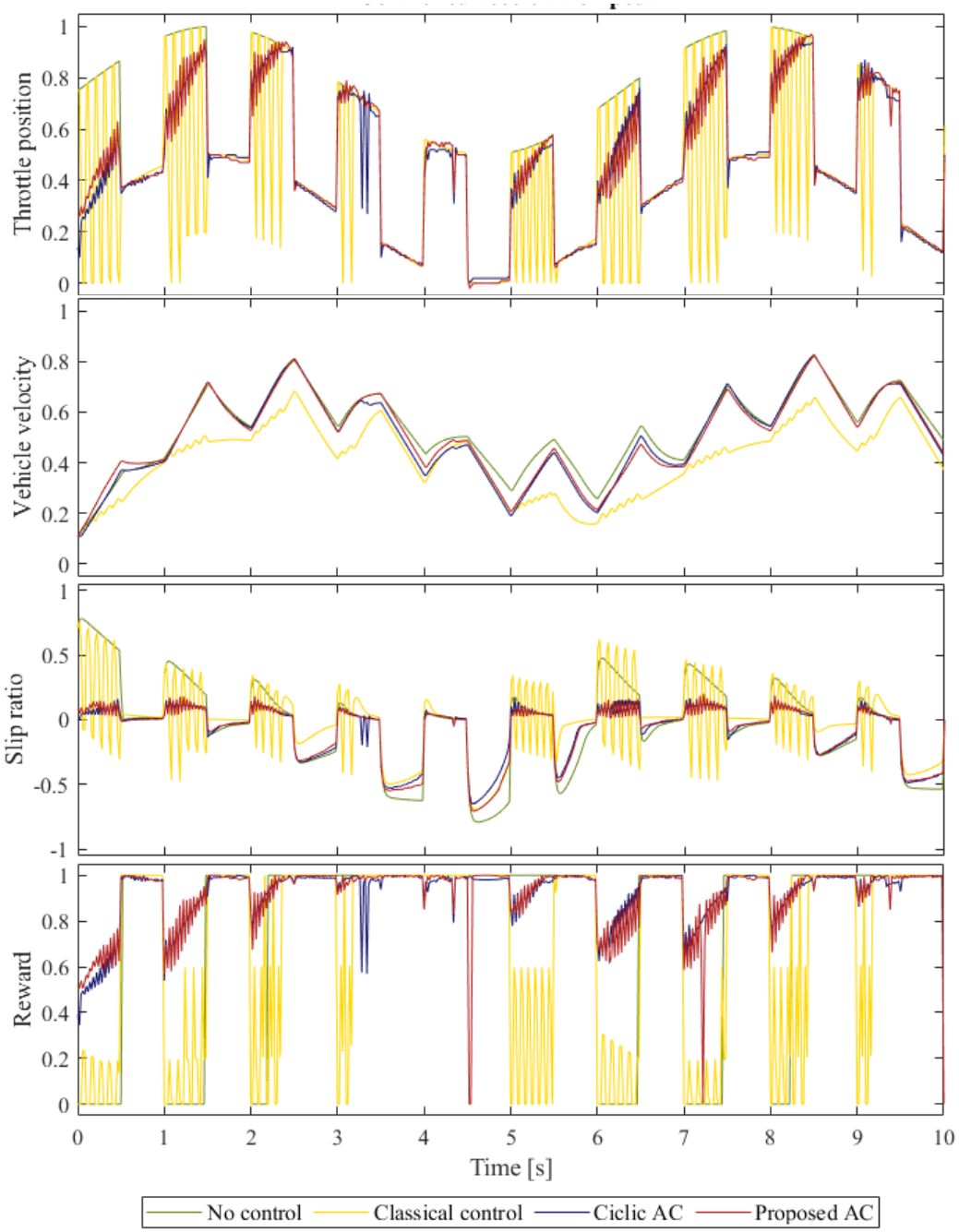
Source: Elaborated by the author.

Figure 74 - Simulation results - Snow - 5 states.



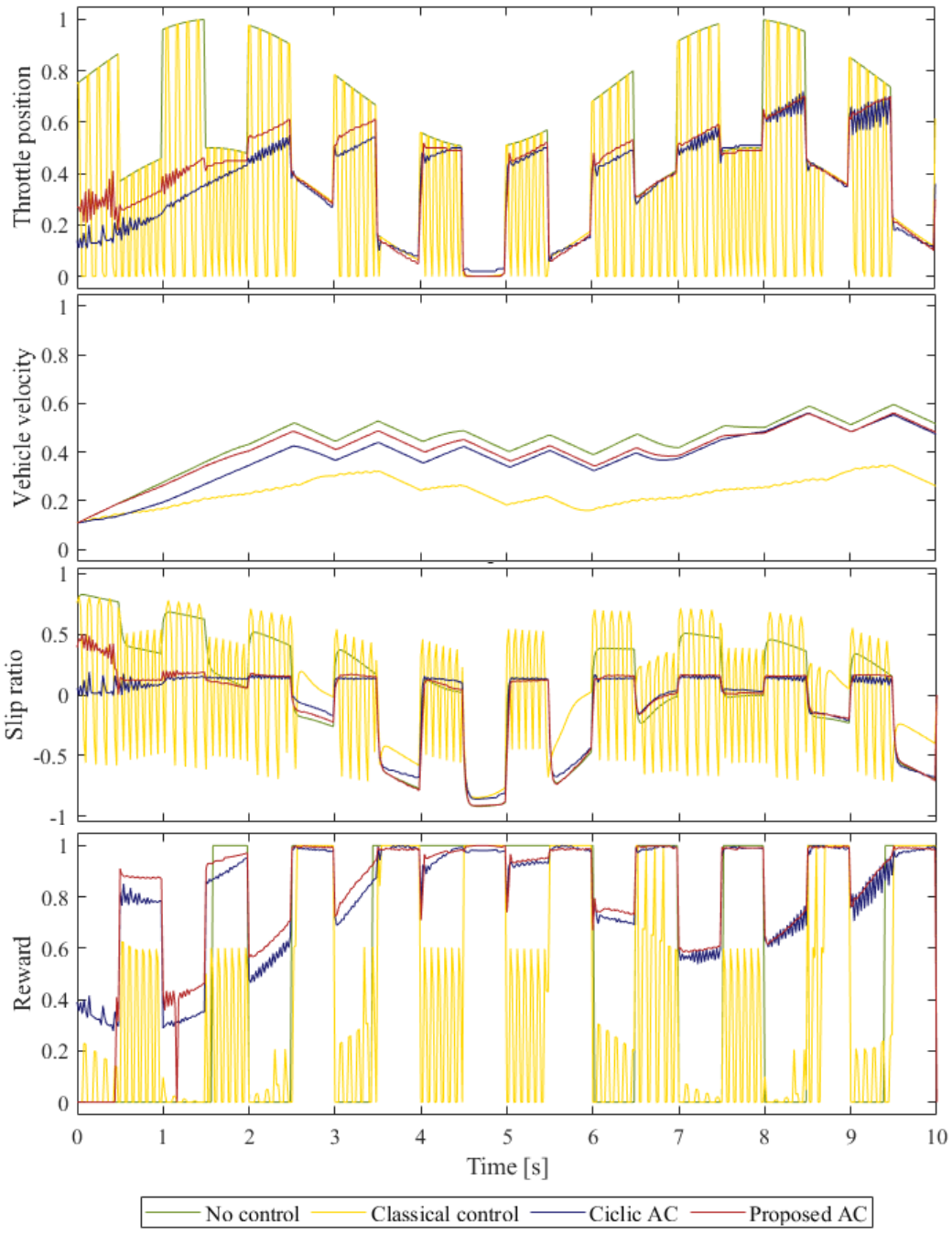
Source: Elaborated by the author.

Figure 75 - Simulation results - Dry asphalt - 4 states.



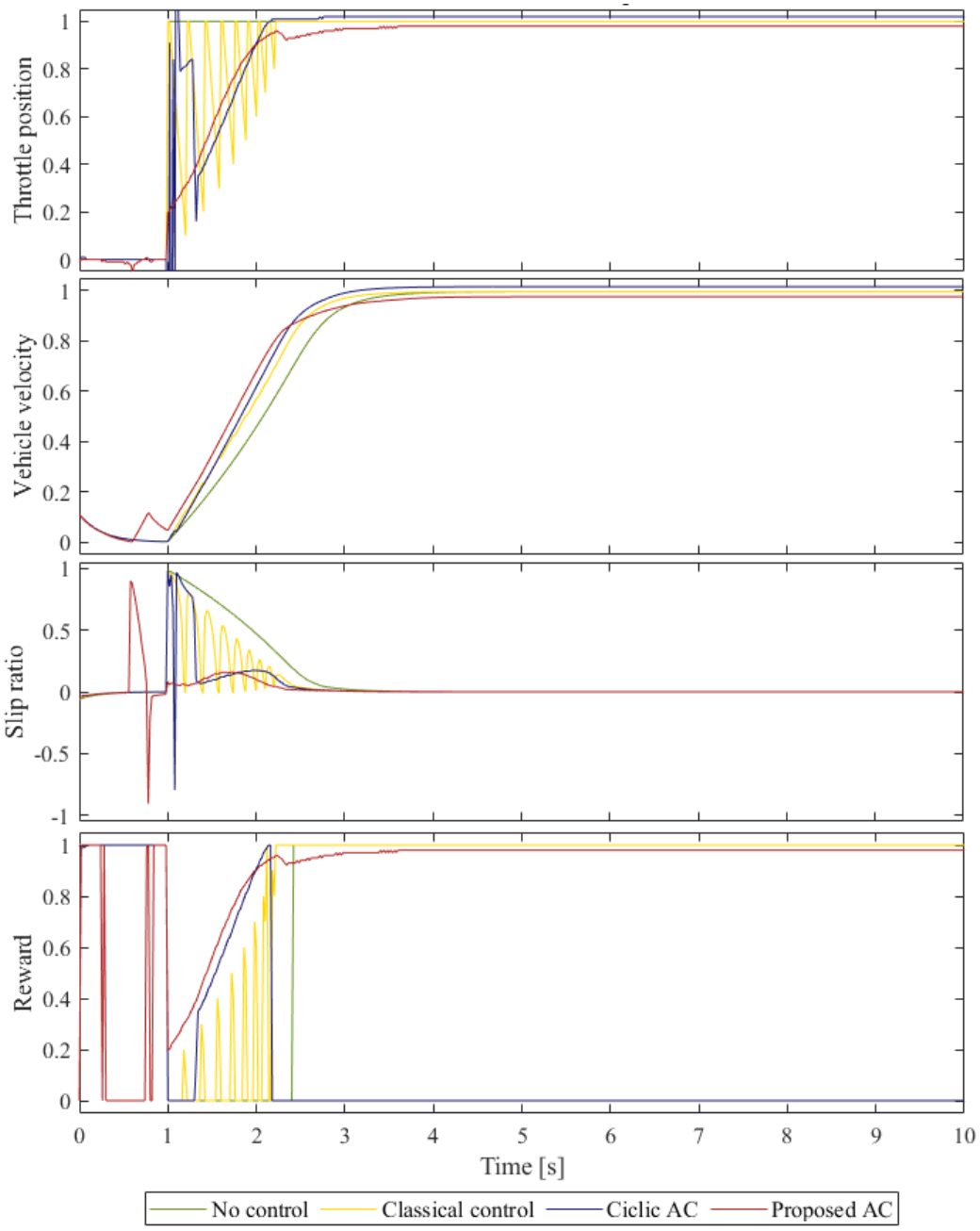
Source: Elaborated by the author.

Figure 76 - Simulation results - Snow - 4 states.



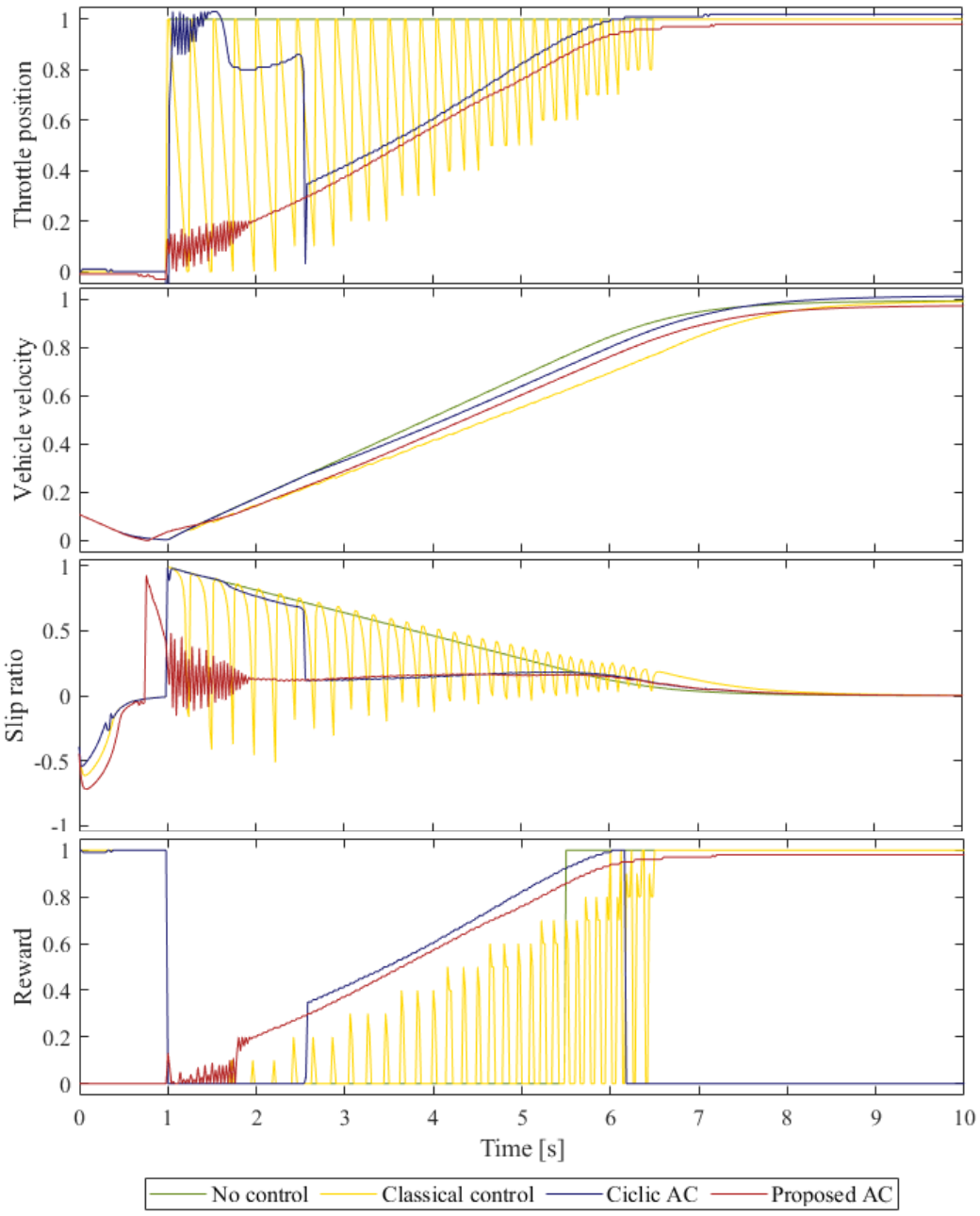
Source: Elaborated by the author.

Figure 77 - Simulation results - Dry asphalt with step input - 5 states.



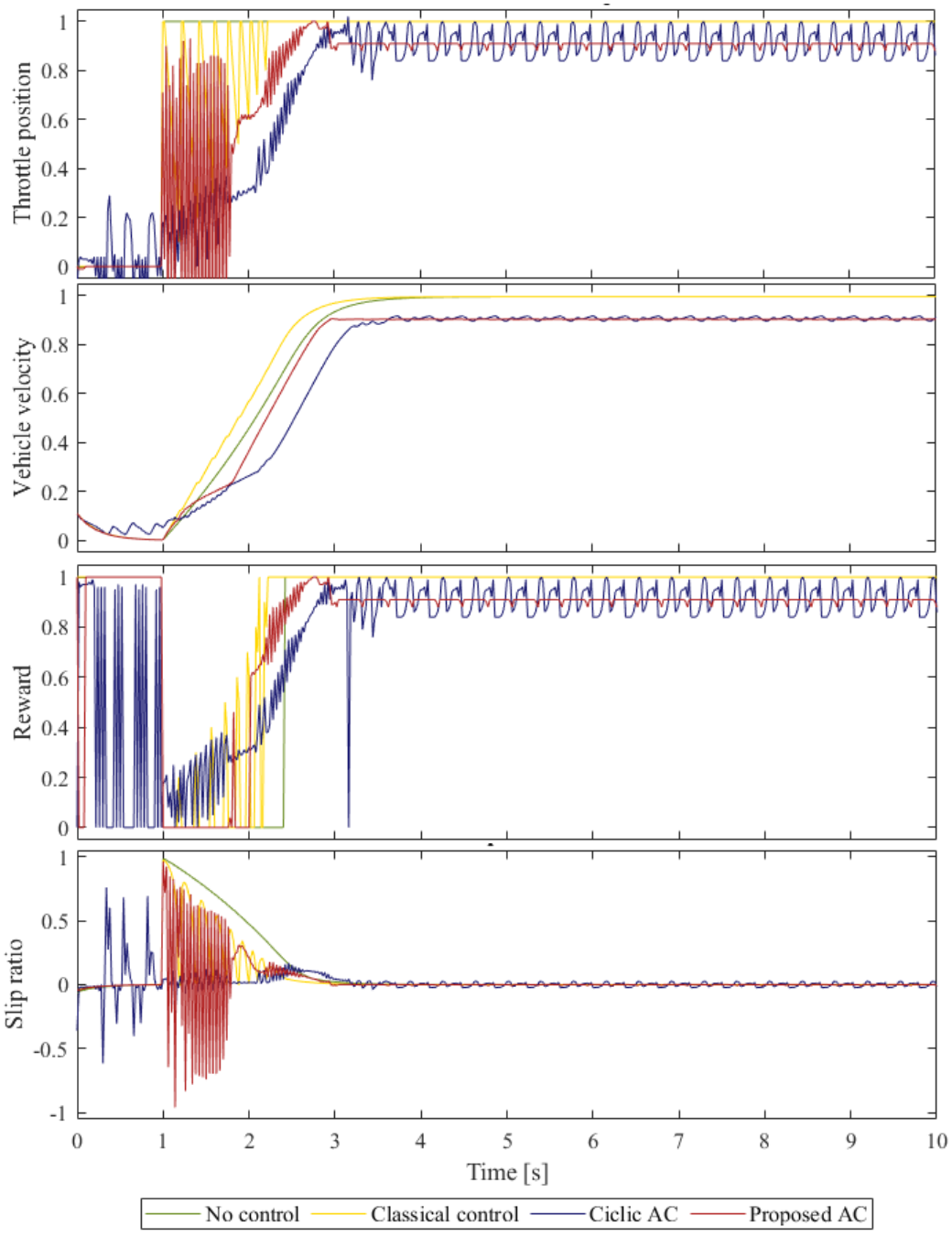
Source: Elaborated by the author.

Figure 78 - Simulation results - Snow with step input - 5 states.



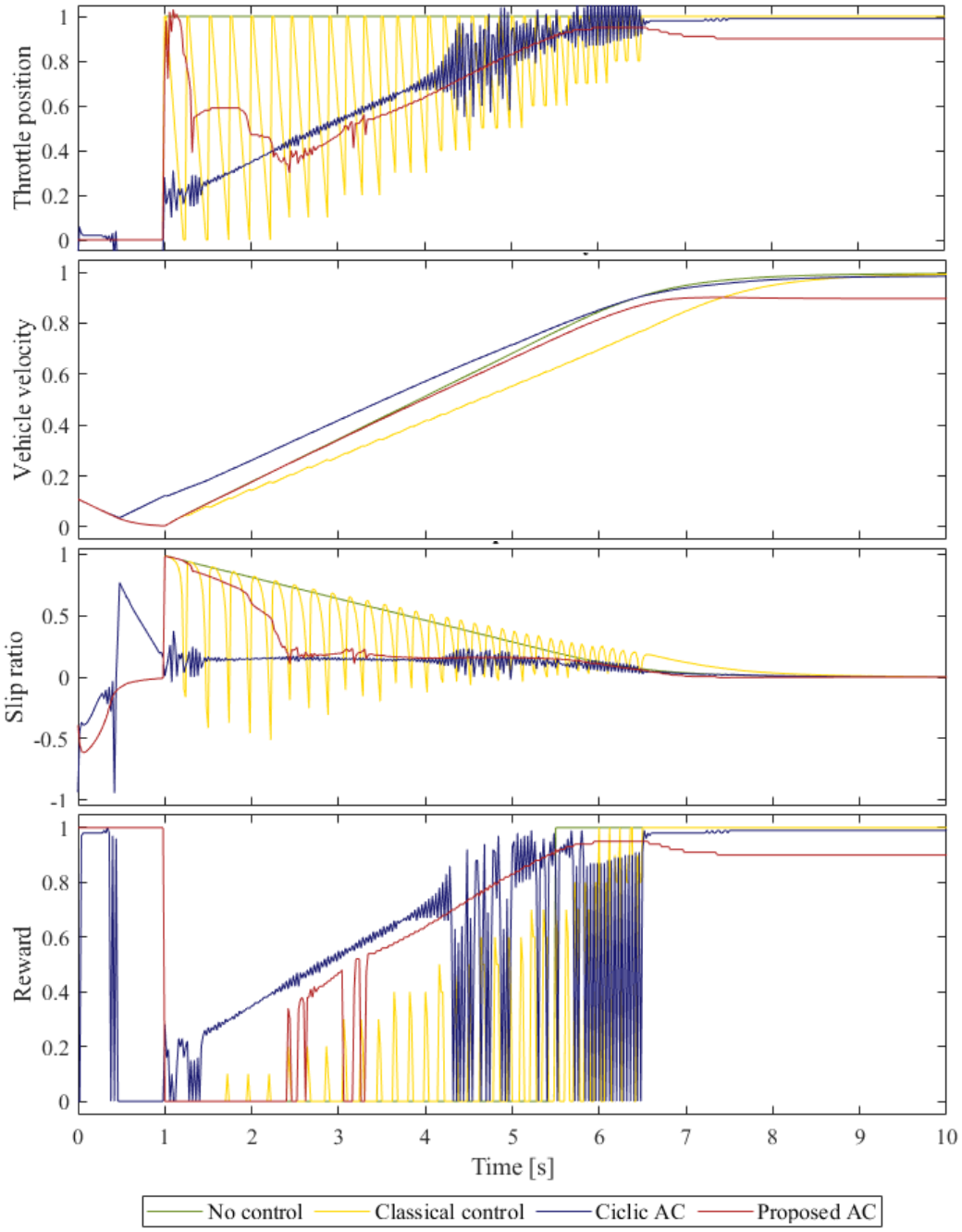
Source: Elaborated by the author.

Figure 79 - Simulation results - Dry asphalt with step input - 4 states.



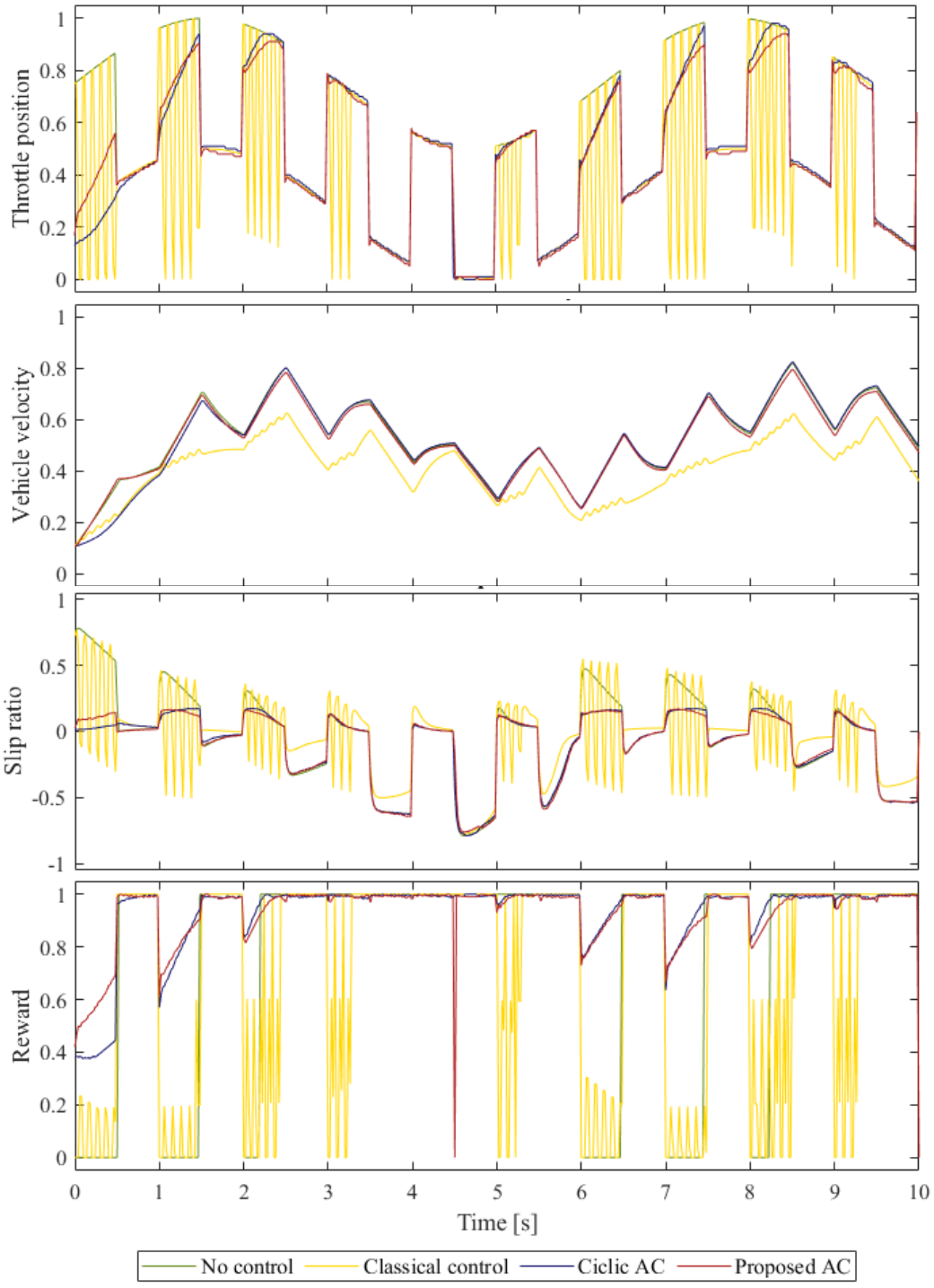
Source: Elaborated by the author.

Figure 80 - Simulation results -Snow with step input - 4 states.



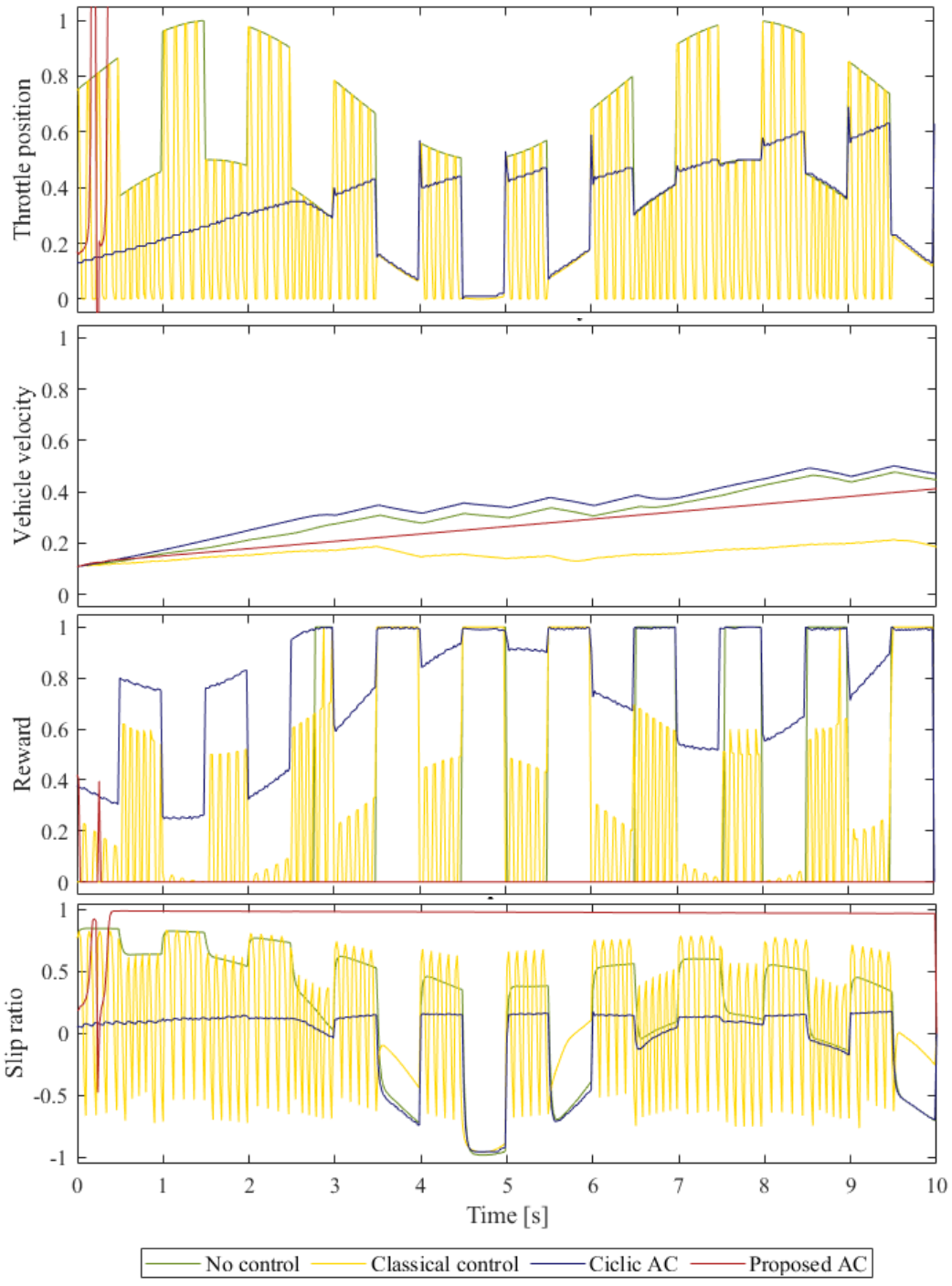
Source: Elaborated by the author.

Figure 81 - Simulation results - Wet asphalt - 5 states.



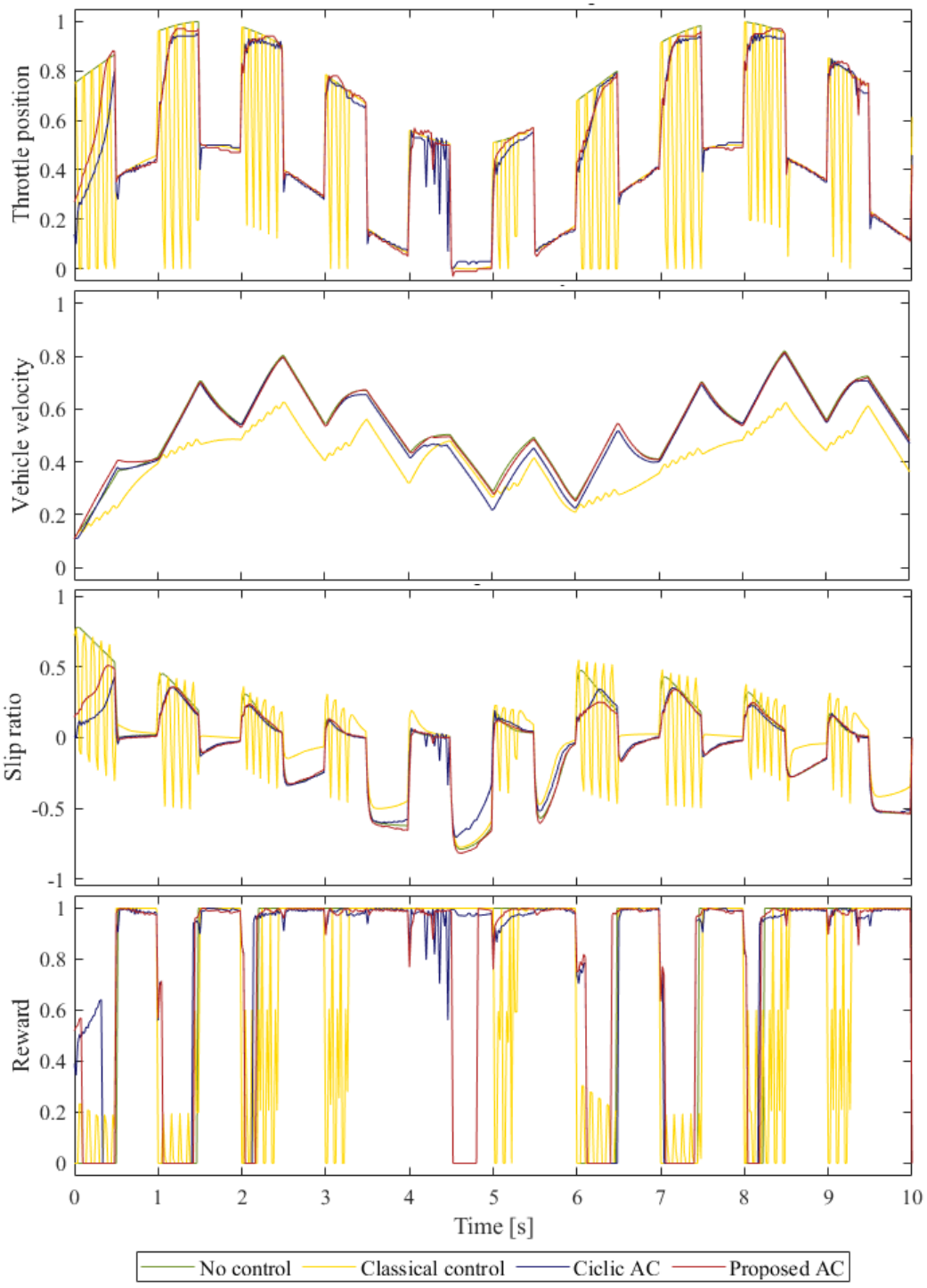
Source: Elaborated by the author.

Figure 82 - Simulation results - Ice - 5 states.



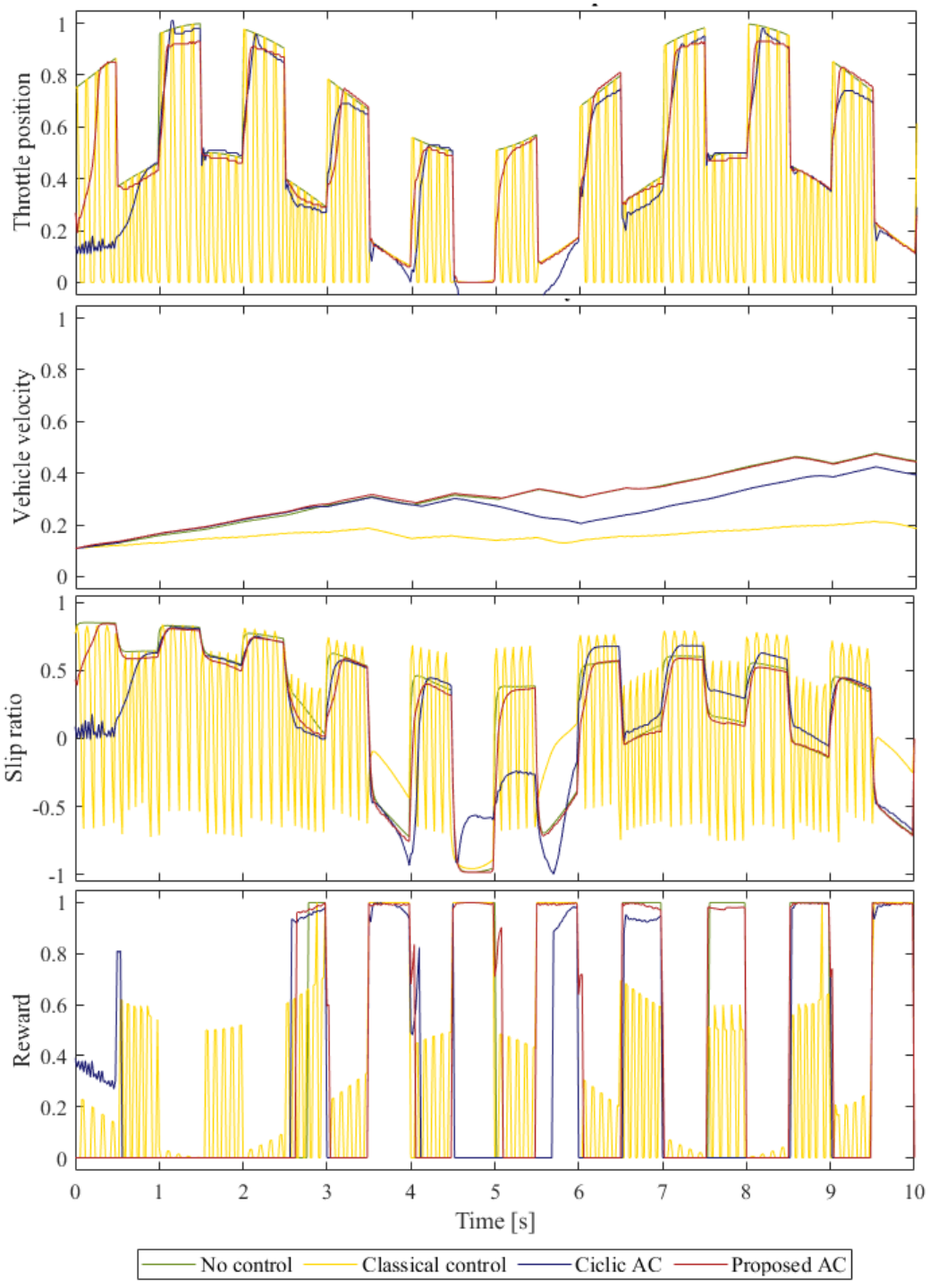
Source: Elaborated by the author.

Figure 83 - Simulation results - Wet asphalt - 4 states.



Source: Elaborated by the author.

Figure 84 - Simulation results - Ice - 4 states.



Source: Elaborated by the author.