



**REPUBLIC OF TURKEY  
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY  
UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
DEPARTMENT OF ELECTRICAL AND ELECTRONIC  
ENGINEERING**

**INTELLIGENT CONTROL OF THE 3D AUTONOMOUS CARS: FROM  
THEORY TO APPLICATION**

**ÖMER CAN TOLUN  
MASTER OF SCIENCE**



**REPUBLIC OF TURKEY  
ADANA ALPARSLAN TÜRKESİ SCIENCE AND TECHNOLOGY  
UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
DEPARTMENT OF ELECTRICAL AND ELECTRONIC  
ENGINEERING**

**INTELLIGENT CONTROL OF THE 3D AUTONOMOUS CARS: FROM  
THEORY TO APPLICATION**

**ÖMER CAN TOLUN  
MASTER OF SCIENCE**

**SUPERVISOR  
Assoc. Prof. Dr. ÖNDER TUTSOY**

**ADANA 2022**



I hereby declare that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all information that is not original to this work.

[Signature]

Ömer Can TOLUN

# **ABSTRACT**

## **INTELLIGENT CONTROL OF THE 3D AUTONOMOUS CARS: FROM THEORY TO APPLICATION**

Ömer Can TOLUN

Master of Science, Department of Electrical and Electronic Engineering

Supervisor: Assoc. Prof. Dr. Önder TUTSOY

February 2022, 180 pages

The rapid increase in the world population and the inadequacy of the existing transportation system have pushed scientists and researchers to develop another option. Consequently, development of the autonomous vehicles (AVs) has begun, which is an energy-efficient and intelligent technology that will not disrupt the existing transportation system, but rather enhance it. One of the essential purposes of this dissertation is to examine the components of the AVs. Among these are the vehicle's battery, Battery Management System (BMS), the mathematical model of the vehicle, and the controller of the vehicle. In order to construct the simulation for the Electrical Vehicle (EV), these components must be examined step by step. MATLAB/Simulink and MapleSim have been used to simulate the EV. The driving range (DR) estimate is based on physical parameters such as vehicle speed, motor angular speed, motor voltage, State of Charge (SoC), and the position of the constructed EV model. The DR estimate has been performed using the Multi-Layer Perceptron with Back-Propagation (MLP-BP) technique. Finally, the dissertation focuses on intelligent control of AV. Python and OpenCV have been used while constructing the lane line/vehicle detection and tracking algorithms. The method for recognizing lane lines in road images or videos utilizes the Canny Function and the Hough Transform. Feature extraction, Support Vector Machines (SVMs), and sliding windows techniques have been used to detect and track moving vehicles on the road.

**Keywords:** Battery models, battery management systems, electric vehicles modelling, artificial neural networks, back-propagation algorithm, driving range estimation, intelligent control of the autonomous vehicles

# ÖZET

## 3B OTONOM ARAÇLARIN AKILLI KONTROLÜ: TEORİDEN UYGULAMAYA

Ömer Can TOLUN

Yüksek Lisans, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Önder TUTSOY

Şubat 2022, 180 sayfa

Dünya nüfusunun hızla artması ve mevcut ulaşım sisteminin yetersizliği, bilim insanlarını ve araştırmacıları başka bir seçenek geliştirmeye itmiştir. Bunun sonucu olarak, mevcut ulaşım sistemini bozmayacak, aksine geliştirecek, enerji verimliliği odaklı ve akıllı bir teknoloji olan otonom araçlar (AV'ler) üzerinde gelişme başlamıştır. Bu tezin temel amaçlarından biri, AV'lerin bileşenlerini incelemektir. Bunlar arasında aracın aküsü, Akü Yönetim Sistemi (BMS), aracın matematiksel modeli ve aracın kontrolörü bulunmaktadır. Elektrikli Araç (EV) için ana simülasyonu belirlemek amacıyla bu bileşenlerin tek tek incelenmesi gerekmektedir. EV'yi simüle etmek için MATLAB/Simulink ve MapleSim kullanılmıştır. Sürüş mesafesi (DR) tahmini, araç hızı, motor açısal hızı, motor voltajı, Şarj Durumu (SoC) ve EV modelinin konumu gibi fiziksel parametrelere dayanır. DR tahmini, Geri Yayılımlı Çok Katmanlı Algılayıcı (MLP-BP) tekniği kullanılarak yapılmıştır. Son kısımda tez, AV'nin akıllı kontrolüne odaklanmaktadır. Şerit çizgisi ve araç tespit-takip algoritmaları oluşturulurken Python ve OpenCV kullanılmıştır. Yol görüntülerinde veya videolarında şerit çizgisini tanıma yöntemi, Canny fonksiyonunu ve Hough Dönüşümünü kullanır. Yolda hareket eden araçları tanıma ve takip etme yöntemi, özellik çıkarma, Destek Vektör Makineleri (SVM'ler) ve sürgülü pencereleri kullanır.

**Anahtar Kelimeler:** Batarya modelleri, batarya yönetim sistemleri, elektrikli araç modelleme, yapay sinir ağları, geri yayılım algoritması, otonom araçların akıllı kontrolü, sürüş mesafesi tahmini



*Dedication*

*I would like to dedicate my dissertation to my beloved uncle.*

## ACKNOWLEDGEMENTS

I would like to thank my esteemed advisor Assoc. Prof. Dr. Önder TUTSOY for his continuous support during this dissertation. His enthusiasm, inspiration, dedication, and rigorous attitude on research work deeply encourage me. His excellent guidance, strong assistance, and critical remarks helped me to make this thesis.

I would also like to thank my committee members, Asst. Prof. Dr. Kemal BALIKÇI and Asst. Prof. Dr. Davood Asadi HENDOUSTANI for letting my defense be an enjoyable moment, and for their brilliant comments and suggestions.

I would like to thank my friends, Res. Asst. Deniz Furkan KANBAK, Res. Asst. Ömer AYANA, Dilara NİHADİOĞLU, and Esila SUCU for their valuable discussion and spiritual help.

Finally, I owe a debt of gratitude to my father Mehmet TOLUN, to my mother Güler TOLUN for their all manner of support, prayers, and spiritual support, and to my sister Vahide Nur TOLUN who always provided me with unfailing support throughout my life. I would like to thank my dear wife Res. Asst. Gülizar Gizem TOLUN has motivated me with patience, understanding, and support. I would also like to thank and pray to my Uncle Cengiz TOLUN whom I recently lost for his all manner of support and guidance that shapes my life with his precious engineering knowledge.

This thesis was financially supported by Adana Alparslan Türkeş Science and Technology University Scientific Research Projects with project number of 21303004.

# TABLE OF CONTENTS

|                                                   |      |
|---------------------------------------------------|------|
| TABLE OF CONTENTS.....                            | viii |
| LIST OF FIGURES .....                             | xii  |
| LIST OF TABLES .....                              | xvi  |
| NOMENCLATURE .....                                | xvii |
| 1. INTRODUCTION.....                              | 1    |
| 1.1. The Objective of the Dissertation .....      | 1    |
| 1.2. The Outline of the Dissertation.....         | 2    |
| 2. LITERATURE REVIEW .....                        | 4    |
| 2.1. Battery Modelling .....                      | 4    |
| 2.1.1. Basis of the Batteries .....               | 4    |
| 2.1.2. Working Principle of the Battery .....     | 6    |
| 2.1.3. Major Parameters of the Battery.....       | 7    |
| 2.1.4. Battery Necessities for the EVs .....      | 7    |
| 2.1.5. Electrochemical Models.....                | 9    |
| 2.1.6. Mathematical Models.....                   | 9    |
| 2.1.6.1. Peukert's Law Model.....                 | 9    |
| 2.1.6.2. Rakhmatov & Vrudhula Model.....          | 11   |
| 2.1.7. Physical Models .....                      | 11   |
| 2.1.8. Electrical Equivalent Circuit Models ..... | 12   |
| 2.2. Battery Management Systems.....              | 16   |
| 2.2.1. Main Functions of the BMS.....             | 17   |
| 2.2.1.1. Data Collection .....                    | 17   |
| 2.2.1.2. State Monitoring .....                   | 17   |
| 2.2.1.3. Safety Protection .....                  | 17   |
| 2.2.1.4. Charging Control .....                   | 18   |
| 2.2.1.5. Energy Management.....                   | 18   |
| 2.2.1.6. Equalization management.....             | 18   |
| 2.2.1.7. Thermal management.....                  | 18   |
| 2.2.1.8. Information Management .....             | 18   |
| 2.3. Modelling of the Vehicles .....              | 19   |

|          |                                                                              |    |
|----------|------------------------------------------------------------------------------|----|
| 2.3.1.   | Longitudinal Vehicle Dynamic Model.....                                      | 19 |
| 2.3.2.   | Lumped Parameter Model .....                                                 | 22 |
| 2.3.3.   | Virtual Prototyping Model .....                                              | 24 |
| 2.4.     | AVs Technologies .....                                                       | 25 |
| 2.4.1.   | Review of the Sensing, Perception, and Decision .....                        | 28 |
| 3.       | MATERIALS AND METHODS.....                                                   | 34 |
| 3.1.     | The Modelling of the Battery .....                                           | 34 |
| 3.1.1.   | The Simulation of the Battery in the Simulink.....                           | 34 |
| 3.1.1.1. | Charge and Discharge Characteristics .....                                   | 36 |
| 3.1.1.2. | Parameterization.....                                                        | 37 |
| 3.1.1.3. | Equation of the Battery .....                                                | 39 |
| 3.1.2.   | The Simulation of the Battery in the MapleSim.....                           | 39 |
| 3.1.2.1. | The Maplesim Environment .....                                               | 39 |
| 3.1.3.   | The Simulation of the Battery in the Matlab .....                            | 40 |
| 3.2.     | The Battery Management Systems .....                                         | 42 |
| 3.2.1.   | Topology of the BMS.....                                                     | 42 |
| 3.2.2.   | Construction Process of the BMS .....                                        | 46 |
| 3.3.     | The EVs Modelling in the Matlab .....                                        | 48 |
| 3.3.1.   | The Linear Model of theVehicle .....                                         | 48 |
| 3.3.2.   | The Linear Model of theVehicle with DC Servo Motor .....                     | 52 |
| 3.3.3.   | The Nonlinear Model of theVehicle with DC Motor.....                         | 55 |
| 3.3.4.   | The Motion Analysis in the Dynamical Environments .....                      | 56 |
| 3.3.4.1. | The Puddles on the Road.....                                                 | 56 |
| 3.3.4.2. | The Bump on the Road.....                                                    | 58 |
| 3.3.4.3. | The Intersection on the Road.....                                            | 59 |
| 3.3.5.   | Determining the Controller Regions for General Motion of the AVs .....       | 60 |
| 3.3.5.1. | The Stabiliy Region for the Linear Model by Using PD Controller.....         | 60 |
| 3.3.5.2. | The Stabiliy Region for the Nonlinear Model by Using PD Controller.....      | 62 |
| 3.3.6.   | Performance Based Controller Design for Autonomous Motion and Departing..... | 64 |
| 3.4.     | The EVs Modelling in the Simulink .....                                      | 66 |
| 3.4.1.   | The architecture of the Vehicle Modelling.....                               | 67 |
| 3.4.2.   | Construction of the EVs Simulation .....                                     | 68 |

|          |                                                                          |     |
|----------|--------------------------------------------------------------------------|-----|
| 3.4.2.1. | Driving Cycle.....                                                       | 68  |
| 3.4.2.2. | Driver Model .....                                                       | 69  |
| 3.4.2.3. | The Battery of the Vehicle.....                                          | 72  |
| 3.4.2.4. | The Motor of the Vehicle.....                                            | 72  |
| 3.4.2.5. | Vehicle Dynamics .....                                                   | 77  |
| 3.5.     | The EVs Modelling in the MapleSim .....                                  | 82  |
| 3.6.     | Artificial neural networks .....                                         | 82  |
| 3.6.1.   | Generalized the BP Algorithm .....                                       | 84  |
| 3.6.2.   | Pseudo Code: Delta Learning Rule for $m$ Inputs and $n$ Outputs NNs..... | 90  |
| 3.7.     | Intelligent Control of the AVs.....                                      | 91  |
| 3.7.1.   | The Lane Line Detection System .....                                     | 92  |
| 3.7.1.1. | The Edge Detection .....                                                 | 93  |
| 3.7.1.2. | The Region of Interest .....                                             | 100 |
| 3.7.1.3. | The Hough Transform.....                                                 | 104 |
| 3.7.2.   | The Vehicle Detection and Tracking System.....                           | 112 |
| 3.7.2.1. | Support Vector Machines (SVM) .....                                      | 113 |
| 3.7.2.2. | Datasets of the Algorithm.....                                           | 115 |
| 3.7.2.3. | Feature Extraction .....                                                 | 117 |
| 3.7.2.4. | Sliding Window Technique .....                                           | 121 |
| 4.       | RESULTS AND DISCUSSION.....                                              | 122 |
| 4.1.     | The Simulation Results of the Battery in the Simulink .....              | 122 |
| 4.2.     | The Simulation Results of the Battery in the MapleSim .....              | 124 |
| 4.3.     | The Simulation Results of the Battery in the Matlab .....                | 129 |
| 4.4.     | The Simulation Results of the BMS in the Matlab/Simulink.....            | 130 |
| 4.5.     | The Results of the EVs Modelling in the Matlab .....                     | 133 |
| 4.6.     | The Simulation Results of the EVs in the Simulink.....                   | 134 |
| 4.7.     | The Simulation Results of the EVs in the MapleSim .....                  | 137 |
| 4.8.     | Driving Range (DR) Estimation by Using MLP-BP .....                      | 139 |
| 4.8.1.   | Construction of the BP Algorithm.....                                    | 139 |
| 4.8.2.   | Vehicle Speed.....                                                       | 140 |
| 4.8.2.1. | Training Part of the Algorithm .....                                     | 141 |
| 4.8.2.2. | Testing Part of the Algorithm .....                                      | 143 |
| 4.8.3.   | Motor Angular Speed.....                                                 | 143 |

|          |                                                        |     |
|----------|--------------------------------------------------------|-----|
| 4.8.3.1. | Training Part of the Algorithm .....                   | 143 |
| 4.8.3.2. | Testing Part of the Algorithm .....                    | 146 |
| 4.8.3.3. | Comparison of the Two Inputs .....                     | 146 |
| 4.8.4.   | Motor Voltage .....                                    | 147 |
| 4.8.4.1. | Training Part of the Algorithm .....                   | 147 |
| 4.8.4.2. | Testing Part of the Algorithm .....                    | 150 |
| 4.8.4.3. | Comparison of the Three Inputs .....                   | 150 |
| 4.8.5.   | State of Charge (SoC) .....                            | 151 |
| 4.8.5.1. | Training Part of the Algorithm .....                   | 151 |
| 4.8.5.2. | Testing Part of the Algorithm .....                    | 154 |
| 4.8.5.3. | Comparison of the Four Inputs .....                    | 154 |
| 4.9.     | The Optimization of the Detected Lines .....           | 155 |
| 4.10.    | The Results of the Vehicle Detection and Tracking..... | 158 |
| 5.       | CONCLUSIONS.....                                       | 164 |
| 6.       | RECOMMENDATIONS.....                                   | 165 |
|          | REFERENCES .....                                       | 166 |
|          | CURRICULUM VITAE .....                                 | 180 |

## LIST OF FIGURES

|                                                                                                                                                                 |    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| <b>Figure 2.1.</b> Depicting the general battery types utilized in the EVs – Nickel-Cadmium (Ni-Cd) – Nickel-Metal Hydride (Ni-MH) (Hannan et al., 2018). ..... | 5  |
| <b>Figure 2.2.</b> a) Discharge of the Li-ion battery b) Charge of the Li-ion battery.....                                                                      | 6  |
| <b>Figure 2.3.</b> An example of a piecewise constant discharge profile. ....                                                                                   | 10 |
| <b>Figure 2.4.</b> The simplified circuit model: (a) ideal voltage source; (b) real voltage source. ..                                                          | 13 |
| <b>Figure 2.5.</b> The Thevenin equivalent circuit model of Li-ion battery. ....                                                                                | 13 |
| <b>Figure 2.6.</b> The extracted Thevenin equivalent circuit model of the Li-ion battery. ....                                                                  | 14 |
| <b>Figure 2.7.</b> Randles circuit-based impedance electric model. ....                                                                                         | 15 |
| <b>Figure 2.8.</b> Structure of the BMS (Xiong, 2020).....                                                                                                      | 16 |
| <b>Figure 2.9.</b> Forces applied on the vehicle's body. ....                                                                                                   | 20 |
| <b>Figure 2.10.</b> (a) 2DOF (lateral, yaw) model (Rajamani, 2011), (b) 3DOF (longitudinal, lateral, yaw) model (Huh, Kim, and Hong 2000).....                  | 23 |
| <b>Figure 2.11.</b> The one-dimensional (quarter vehicle) model (Cao et al., 2008). ....                                                                        | 24 |
| <b>Figure 2.12.</b> The structure of the AVs. ....                                                                                                              | 28 |
| <b>Figure 3.1.</b> The system of the battery. ....                                                                                                              | 35 |
| <b>Figure 3.2.</b> The equivalent circuit of the block. ....                                                                                                    | 35 |
| <b>Figure 3.3.</b> Typical discharge characteristics. ....                                                                                                      | 36 |
| <b>Figure 3.4.</b> Typical charge characteristics. ....                                                                                                         | 37 |
| <b>Figure 3.5.</b> The ability to function in the low temperatures. ....                                                                                        | 38 |
| <b>Figure 3.6.</b> Structure of centralized BMS topology. ....                                                                                                  | 43 |
| <b>Figure 3.7.</b> Structure of distributed BMS topology. ....                                                                                                  | 45 |
| <b>Figure 3.8.</b> The fundamental construction process of the BMS. ....                                                                                        | 47 |
| <b>Figure 3.9.</b> The forces affecting the movement of the vehicle. ....                                                                                       | 48 |
| <b>Figure 3.10.</b> The impulse response of the linear systems. ....                                                                                            | 52 |
| <b>Figure 3.11.</b> DC servo motor model. ....                                                                                                                  | 53 |
| <b>Figure 3.12.</b> The step response of the linear system with DC motor. ....                                                                                  | 55 |
| <b>Figure 3.13.</b> The response of the system against to the puddles. ....                                                                                     | 57 |
| <b>Figure 3.14.</b> The response of the system against to the bump.....                                                                                         | 59 |
| <b>Figure 3.15.</b> The block diagram of the closed-loop system. ....                                                                                           | 61 |

|                                                                                                   |     |
|---------------------------------------------------------------------------------------------------|-----|
| <b>Figure 3.16.</b> The stability region of the linear system.....                                | 61  |
| <b>Figure 3.17.</b> Step response of the closed-loop system. ....                                 | 62  |
| <b>Figure 3.18.</b> The stability region of the nonlinear system.....                             | 63  |
| <b>Figure 3.19.</b> The difference between step and impulse response of the closed-loop system. . | 64  |
| <b>Figure 3.20.</b> Step response of the closed-loop system. ....                                 | 66  |
| <b>Figure 3.21.</b> The architecture of the EVs. ....                                             | 67  |
| <b>Figure 3.22.</b> The simulation of the EVs in the Simulink.....                                | 68  |
| <b>Figure 3.23.</b> Inside of the driving cycle sub-system.....                                   | 69  |
| <b>Figure 3.24.</b> Inside of the driver model sub-system. ....                                   | 70  |
| <b>Figure 3.25.</b> Inside of the motor sub-system. ....                                          | 72  |
| <b>Figure 3.26.</b> Up counter PWM. ....                                                          | 73  |
| <b>Figure 3.27.</b> Down counter PWM. ....                                                        | 74  |
| <b>Figure 3.28.</b> Up-down counter PWM.....                                                      | 74  |
| <b>Figure 3.29.</b> The structure of the H-bridge.....                                            | 75  |
| <b>Figure 3.30.</b> The two basic states of the H-bridge.....                                     | 76  |
| <b>Figure 3.31.</b> Inside of the vehicle dynamics sub-system.....                                | 77  |
| <b>Figure 3.32.</b> The longitudinal vehicle model. ....                                          | 78  |
| <b>Figure 3.33.</b> The forces affecting the tire. ....                                           | 81  |
| <b>Figure 3.34.</b> The simulation of the EVs in the MapleSim.....                                | 82  |
| <b>Figure 3.35.</b> The illustration of the BP algorithm. ....                                    | 83  |
| <b>Figure 3.36.</b> Block diagram of the neuron without hidden layer. ....                        | 84  |
| <b>Figure 3.37.</b> Block diagram of neuron with hidden layer and BP.....                         | 87  |
| <b>Figure 3.38.</b> The image of the road. ....                                                   | 92  |
| <b>Figure 3.39.</b> The intensity of the pixels.....                                              | 93  |
| <b>Figure 3.40.</b> The numeric values of the pixels. ....                                        | 93  |
| <b>Figure 3.41.</b> Types of the gradient. ....                                                   | 94  |
| <b>Figure 3.42.</b> The gradient of the image. ....                                               | 94  |
| <b>Figure 3.43.</b> The grayscale of the road image. ....                                         | 95  |
| <b>Figure 3.44.</b> The blurred of the road image.....                                            | 97  |
| <b>Figure 3.45.</b> The corresponding image of the road. ....                                     | 97  |
| <b>Figure 3.46.</b> Definition of low and high threshold in the canny function. ....              | 98  |
| <b>Figure 3.47.</b> Applying the canny function to the road image.....                            | 99  |
| <b>Figure 3.48.</b> The canny image in the mathplot.....                                          | 100 |

|                                                                                                                            |     |
|----------------------------------------------------------------------------------------------------------------------------|-----|
| <b>Figure 3.49.</b> The masked area of the image. ....                                                                     | 101 |
| <b>Figure 3.50.</b> The working principle of the bitwise_and. ....                                                         | 102 |
| <b>Figure 3.51.</b> The pixel values after applying the bitwise_and.....                                                   | 102 |
| <b>Figure 3.52.</b> The cropped image of the canny image. ....                                                             | 103 |
| <b>Figure 3.53.</b> The line parameters in (a) 2D space and (b) Hough space. ....                                          | 104 |
| <b>Figure 3.54.</b> The line parameters with several lines in (a) 2D space and (b) Hough space. .                          | 105 |
| <b>Figure 3.55.</b> The intersection of the two points in (a) 2D space and (b) Hough space. ....                           | 106 |
| <b>Figure 3.56.</b> The intersection of the four points in (a) 2D space and (b) Hough space. ....                          | 107 |
| <b>Figure 3.57.</b> The relationship between Cartesian and Polar coordinates. ....                                         | 108 |
| <b>Figure 3.58.</b> The intersection of the three points in (a) Polar space and (b) Hough space....                        | 109 |
| <b>Figure 3.59.</b> The intersection of the three points in (a) Polar space and (b) Hough space (in the grid method). .... | 110 |
| <b>Figure 3.60.</b> The lines detection of the gradient image. ....                                                        | 111 |
| <b>Figure 3.61.</b> The blended image of the two images. ....                                                              | 112 |
| <b>Figure 3.62.</b> The labelled data. ....                                                                                | 113 |
| <b>Figure 3.63.</b> The simply a line for the best hyperplane. ....                                                        | 114 |
| <b>Figure 3.64.</b> Vehicle and non-vehicle images. ....                                                                   | 116 |
| <b>Figure 3.65.</b> Scaling down the resolution of the image by using OpenCV. ....                                         | 117 |
| <b>Figure 3.66.</b> The histogram and colour feature of the vehicles. ....                                                 | 118 |
| <b>Figure 3.67.</b> The histogram and colour feature of the non-vehicles.....                                              | 119 |
| <b>Figure 3.68.</b> The HOG features extraction from the vehicle and non-vehicle. ....                                     | 120 |
| <b>Figure 4.1.</b> The input and outputs of the Battery with step input.....                                               | 122 |
| <b>Figure 4.2.</b> The input and outputs of the Battery with pulse inputs. ....                                            | 123 |
| <b>Figure 4.3.</b> The input and outputs of the Battery with sine input.....                                               | 124 |
| <b>Figure 4.4.</b> The modelling of the battery in the MapleSim. ....                                                      | 125 |
| <b>Figure 4.5.</b> The input and outputs of the battery with step input. ....                                              | 126 |
| <b>Figure 4.6.</b> The input and outputs of the battery with pulse input. ....                                             | 127 |
| <b>Figure 4.7.</b> The input and outputs of the battery with sine input. ....                                              | 128 |
| <b>Figure 4.8.</b> The comparison of the SoC with different $\alpha$ . ....                                                | 129 |
| <b>Figure 4.9.</b> The Modelling of the BMS. ....                                                                          | 130 |
| <b>Figure 4.10.</b> The outputs of the BMS simulation. ....                                                                | 133 |
| <b>Figure 4.11.</b> Reference versus real vehicle speed. ....                                                              | 134 |
| <b>Figure 4.12.</b> Reference versus real vehicle positions. ....                                                          | 135 |

|                                                                                                    |     |
|----------------------------------------------------------------------------------------------------|-----|
| <b>Figure 4.13.</b> The SoC of the battery, the voltage, and the current drawn from the battery... | 136 |
| <b>Figure 4.14.</b> Reference versus real vehicle speeds. ....                                     | 137 |
| <b>Figure 4.15.</b> Reference versus real vehicle positions. ....                                  | 138 |
| <b>Figure 4.16.</b> The SoC of the battery, the voltage, and current drawn from the battery.....   | 139 |
| <b>Figure 4.17.</b> The structure of the BP algorithm.....                                         | 140 |
| <b>Figure 4.18.</b> The input of the algorithm for the vehicle speeds.....                         | 141 |
| <b>Figure 4.19.</b> Mean error of the output for the vehicle speed. ....                           | 142 |
| <b>Figure 4.20.</b> The estimated position of the vehicle with the trained data.....               | 142 |
| <b>Figure 4.21.</b> Tested vehicle position for four different driving cycles.....                 | 143 |
| <b>Figure 4.22.</b> The input of the algorithm for the motor angular speeds.....                   | 144 |
| <b>Figure 4.23.</b> Mean error of the output for the motor angular speed. ....                     | 145 |
| <b>Figure 4.24.</b> The estimated position of the vehicle with the trained data.....               | 145 |
| <b>Figure 4.25.</b> Tested vehicle position for four different driving cycles.....                 | 146 |
| <b>Figure 4.26.</b> The error between estimated and real outputs. ....                             | 147 |
| <b>Figure 4.27.</b> The input of the algorithm for the motor voltage. ....                         | 148 |
| <b>Figure 4.28.</b> Mean error of the output for the motor voltage. ....                           | 149 |
| <b>Figure 4.29.</b> The estimated position of the vehicle with the trained data.....               | 149 |
| <b>Figure 4.30.</b> Tested vehicle position for 4 different driving cycles. ....                   | 150 |
| <b>Figure 4.31.</b> The error between estimated and real outputs. ....                             | 151 |
| <b>Figure 4.32.</b> The input of the algorithm for the SoC.....                                    | 152 |
| <b>Figure 4.33.</b> Mean error of the output for the SoC.....                                      | 153 |
| <b>Figure 4.34.</b> The estimated position of the vehicle with the trained data.....               | 153 |
| <b>Figure 4.35.</b> Tested vehicle position for four different driving cycles.....                 | 154 |
| <b>Figure 4.36.</b> The error between estimated and real outputs. ....                             | 155 |
| <b>Figure 4.37.</b> The optimization of the detected lines in the road image. ....                 | 157 |
| <b>Figure 4.38.</b> The lines detection in the various road images. ....                           | 158 |
| <b>Figure 4.39.</b> The process of the sliding window and heat map. ....                           | 160 |
| <b>Figure 4.40.</b> The result of the vehicle detection. ....                                      | 160 |
| <b>Figure 4.41.</b> The pipeline sample on the test images. ....                                   | 162 |

## LIST OF TABLES

|                                                                                           |     |
|-------------------------------------------------------------------------------------------|-----|
| <b>Table 2.1.</b> Comparison of commercially usable batteries .....                       | 8   |
| <b>Table 2.2.</b> The levels of autonomous vehicle .....                                  | 26  |
| <b>Table 3.1.</b> A123 Li-ion temperature-dependent battery model from the datasheet..... | 38  |
| <b>Table 3.2.</b> The parameters of the vehicle.....                                      | 50  |
| <b>Table 3.3.</b> The parameters of the DC servo motor .....                              | 54  |
| <b>Table 3.4.</b> The variables of the equations .....                                    | 71  |
| <b>Table 3.5.</b> The operation of the MOSFETS.....                                       | 76  |
| <b>Table 3.6.</b> The paramaters of the longitudinal model.....                           | 80  |
| <b>Table 3.7.</b> The parameters of tire model.....                                       | 81  |
| <b>Table 3.8.</b> The pseudo code of the kernel convolution.....                          | 95  |
| <b>Table 3.9.</b> The pseudo code of the edge detection.....                              | 99  |
| <b>Table 3.10.</b> The pseudo code of the region of interest .....                        | 103 |
| <b>Table 3.11.</b> The pseudo code of the Hough transform .....                           | 110 |
| <b>Table 3.12.</b> The parameters of the Linear SVM .....                                 | 120 |
| <b>Table 3.13.</b> The parameters of the Linear SVM .....                                 | 121 |
| <b>Table 4.1.</b> The pseudo code of the synchronization algorithm.....                   | 131 |
| <b>Table 4.2.</b> The pseudo code of the control algorithm of the charging profile .....  | 132 |
| <b>Table 4.3.</b> The pseudo code of the optimization.....                                | 156 |
| <b>Table 4.4.</b> The SVM classifier testing results.....                                 | 159 |
| <b>Table 4.5.</b> The parameters of the sliding window .....                              | 159 |

# NOMENCLATURE

|           |                                                           |
|-----------|-----------------------------------------------------------|
| AV        | : Autonomous Vehicle                                      |
| BMS       | : Battery Management System                               |
| EV        | : Electrical Vehicle                                      |
| DR        | : Driving Range                                           |
| SoC       | : State of Charge                                         |
| MLP-BP    | : Multi-Layer Perceptron with Back-Propagation            |
| ITS       | : Intelligent Transport System                            |
| SVM       | : Support Vector Machine                                  |
| SoH       | : State of Health                                         |
| SoP       | : State of Power                                          |
| RUL       | : Remaining Useful Lifetime                               |
| VCU       | : Vehicle Control Unit                                    |
| SoE       | : State of Energy                                         |
| SAE       | : Society of Automotive Engineers                         |
| DoF       | : Degree of Freedom                                       |
| FE        | : Finite Element                                          |
| FVP       | : Functional Virtual Prototyping                          |
| ABS       | : Antilock Brake System                                   |
| ADAS      | : Advanced Driver Assistance System                       |
| GPS       | : Global Positioning System                               |
| HD        | : High-Definition                                         |
| MLP-SARSA | : Multi-Layer Perceptron-State Action Reward State Action |
| NMPC      | : Nonlinear Model Predictive Control                      |
| RL        | : Reinforcement Learning                                  |
| RRT       | : Rapidly Random Trees                                    |
| HIL       | : Hardware-in-the-Loop                                    |
| DoD       | : Depth of Discharge                                      |
| Li-ion    | : Lithium-ion                                             |
| DE        | : Differential Equation                                   |

|     |                                  |
|-----|----------------------------------|
| BMC | : Battery Monitoring Circuit     |
| BCU | : Battery Control Unit           |
| PCB | : Printed Circuit Board          |
| CAN | : Controller Area Network        |
| DC  | : Direct Current                 |
| AC  | : Alternative Current            |
| PD  | : Proportional and Derivative    |
| PI  | : Proportional and Integral      |
| PWM | : Pulse Width Modulation         |
| CG  | : Center of Gravity              |
| ANN | : Artificial Neural Network      |
| BP  | : Back Propagation               |
| HOG | : Histogram of Oriented Gradient |

# 1. INTRODUCTION

As the world's population and transportation system continue to grow at an alarming rate, the need for managing them grows at the same time. The world is becoming increasingly populous. As a result, the number of machines, including vehicles of all kinds, increased at the same time. Nevertheless, new concerns such as traffic, accidents, and several other difficulties must be addressed in order to be effectively handled. One of these difficulties is traffic on highways and in cities. Many methods, such as traffic lights, signs, and other devices, have been used to cope with this phenomenon. It appears that these solutions are either insufficient or inefficient on their own. Technology advancements, such as object detection and tracking, are being developed in order to use automated camera surveillance to provide data that can be interpreted and used in the process of making decisions in real time. Object identification and tracking is one of the many components of the Intelligent Transport System (ITS), which is a new trend in transportation. This technology is used to identify cars, lanes, traffic signs, and the movement of vehicles on the road. The capacity to recognize and categorize vehicles provides us with the opportunity to enhance traffic flow and road conditions, avoid accidents, and track down and register traffic offenses and traffic infractions.

Until now, humans have been responsible for the vast majority of vehicle operations. In the meantime, recent advancements in sensing technology, computing power, data processing, and connectivity have made it possible to begin developing vehicles that are autonomous, meaning they can perform transportation tasks without the need for human intervention. Therefore, a variety of research is being conducted in the area of the AVs. Many studies have shown the feasibility of autonomous driving in real-world situations, and various teams across the globe are working to make significant progress on the subject. The studies that compose the content of this dissertation have been presented to the literature with the aim of contributing to the development process of AVs technology.

## 1.1. The Objective of the Dissertation

One of the main objectives of this dissertation is to perform an analysis of the components that comprise the AVs. These components are the vehicle's battery, BMS, the mathematical model of the vehicle, and the controller of the vehicle. The main purpose of explaining these

components is to consider them one by one with the aim of detecting the main simulation created for the EVs. The EV has been simulated using the MATLAB/Simulink and MapleSim software. All the mentioned components have been combined in the simulation environment, and the modelling of the EV has been constructed. After this process, physical parameters such as vehicle speed, SoC, engine angular speed, motor voltage, and position obtained from EV modelling are used as input and output in the DR estimation. The MLP-BP algorithm has been used to perform the DR estimation and in the relevant part of the dissertation, it is explained step-by-step in detail.

The last topic covered in the dissertation is the intelligent control of the AVs. The algorithms used to perform the operations at this stage have been created using Python and OpenCV. These algorithms have been used to perform lane line/vehicle detection and tracking functions, which can be considered the basis for the vehicle's gaining autonomous features. The algorithm, which combines the Canny Function and the Hough Transform, has been designed to detect lane lines in a road image or video. A gradient image of the road has been created by performing operations on the image with the Canny Function. The strip in the region of interest of the vehicle on the gradient image has been detected with the Hough Transform. Additionally, an algorithm consisting of feature extraction, SVM, and sliding-window techniques has been constructed to detect and track vehicles on the road. Since the concentrated information in the raw data contains a significant amount of unnecessary information, the feature extraction method has been applied to the data to obtain and process only certain and important information about the object. The information obtained by feature extraction has been categorized with the SVM classifier. In the last part, the sliding window technique has been applied to detect and track the vehicles in the image or video.

## **1.2. The Outline of the Dissertation**

This dissertation is organized as follows:

- In Chapter 2, a literature review of the battery modelling approaches, battery management systems, vehicle modelling approaches, and AVs technologies is performed.
- In Chapter 3, the modelling of the battery, the topology and modelling of BMSs for the EVs, and the modelling of the EVs are introduced. In addition, the artificial neural

network algorithm that is the MLP-BP is presented in detail. Furthermore, the use of the MLP-BP to predict the DR of a vehicle is handled in this dissertation. Finally, the intelligent control methods, such as lane line and vehicle detection and tracking that are created for the AVs using Python are explained in depth.

- In Chapter 4, extensive discussion is held on the findings of Chapter 3, which includes battery modelling and BMS modelling for the EVs, the modelling of the EVs, vehicle DR estimation, and intelligent control approaches for the AVs.
- Finally, the conclusion is given in Chapter 5.



## **2. LITERATURE REVIEW**

This chapter of the dissertation reviews the battery modelling approaches, battery management systems, vehicle modelling approaches, and AVs technologies.

### **2.1. Battery Modelling**

Modelling is an essential tool for the engineering applications. For this purpose, battery modelling, which is mostly used for the simulations of the EVs, plays an important role in developing advanced control approaches for them. The modelling of the batteries is critical for predicting their SoC and State of Health (SoH), accurately. Numerous battery models and parameter estimation approaches have been suggested, in the literature. Because of the rising interest in this area, improving the accuracy of the battery models, particularly those using Lithium-ion (Li-ion) ones, has become a critical goal. This is why there are various ways to represent the battery dynamics with the models of varying degrees of complexity. Because the battery is a nonlinear system due to nonlinear load, it can be divided into four types of categories commonly used in electric vehicles: electrochemical models, mathematical models, physical models, and electrical equivalent circuit models. The subsections below summarize the researches performed on the battery's structure, working principle, major material components, and required properties of the batteries for the electric vehicles. In the continuation of these subsections, the battery modelling methods are briefly reviewed.

#### **2.1.1. Basis of the Batteries**

There are a variety of batteries, and considering the impact of fossil fuels on climate change, they are being used to solve the challenge of electric transportation. Figure 2.1 depicts a graph the most commonly used battery technology (Hannan et al., 2018).

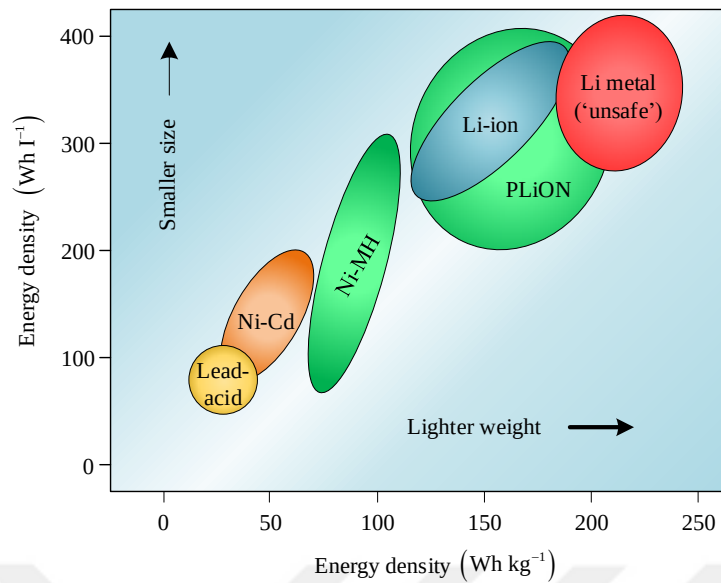


**Figure 2.1.** Depicting the general battery types utilized in the EVs – Nickel-Cadmium (Ni-Cd) – Nickel-Metal Hydride (Ni-MH) (Hannan et al., 2018).

Early EV applications relied on Gaston Planté's rechargeable lead-acid battery, which he invented in 1859. Jungner invented the nickel-cadmium battery in 1899, which improved storage capacity significantly yet had certain downsides, with the inclusion of a voltage-suppressing problem known as the memory effect that develops as the battery ages (Chen et al., 2019). Although research proceeded during the first half of the twentieth century, Li-ion batteries were not developed until 1985. They were commercialized after another 6 years of development. Meanwhile, EVs using ZEBRA and Nickel-Metal Hydride batteries have been constructed (Sudworth, 2001). Among the developed battery types, the Li-ion battery is the most commonly used energy storage technology for the EVs.

Batteries essentially consist of two electrodes in an electrolyte that act as a storage medium. This electrolyte affords a channel for ion exchange, which generates energy. In Figure 2.1, each of the batteries has its own set of benefits and drawbacks, but Li-ion batteries have risen to the top of the market for usage in most handheld and portable gadgets, as well as electric vehicles, thanks to recent developments. This is owing to their high efficiency, energy density, increased nominal voltages, lower specific weight, long life cycle, faster and more efficient charging, and smaller size. Along with these advantages, there are also disadvantages, such as the hefty price tag and the requirement for complicated security and monitoring methods (Baboselac, Hederić, and Benšić, 2017; Gandoman et al., 2019).

The structure of the Li-ion batteries is a power storage instrument that withstands infusion processes among both the electrodes and lithium ions moving as the charging bearer. Considering this wide description, the Li-ion battery family encompasses a variety of cell chemistries. The negative electrode in most Li-ion batteries is constructed mostly of carbon (e.g., graphite) or lithium titanate ( $\text{Li}_4\text{Ti}_5\text{O}_{12}$ ), with some new elements under research, such as Li metal and Li (Si) compounds. The electrolyte utilized alters depending on the electrode substances chosen. However, it is generally made up of a combination of lithium salts (e.g.,  $\text{LiPF}_6$ ) and a natural dissolvent (e.g., diethyl carbonate) to provide ion diffusion. A separating membrane is utilized to allow lithium ions to flow between the electrodes while avoiding an inner short circuit (Wen, Yu, and Chen, 2012).

### 2.1.2. Working Principle of the Battery

The battery is a device made up of one or more electrochemical cells that transform the chemical energy stored in its active materials into the electrical energy through an oxidation-reduction (redox) process that involves the transfer of electrons from one material to another. When a battery is discharged, it works as shown in Figure 2.2 (a). The operation of Li-ion batteries can be expressed as the battery being used as a galvanic device; electrons flow from the negative electrode to a current collector, then to the load, and finally to the positive electrode through a second current collector. To maintain electro neutrality,  $\text{Li}^+$  ions move from the negative electrode via the electrolyte to the positive electrode at the same time.

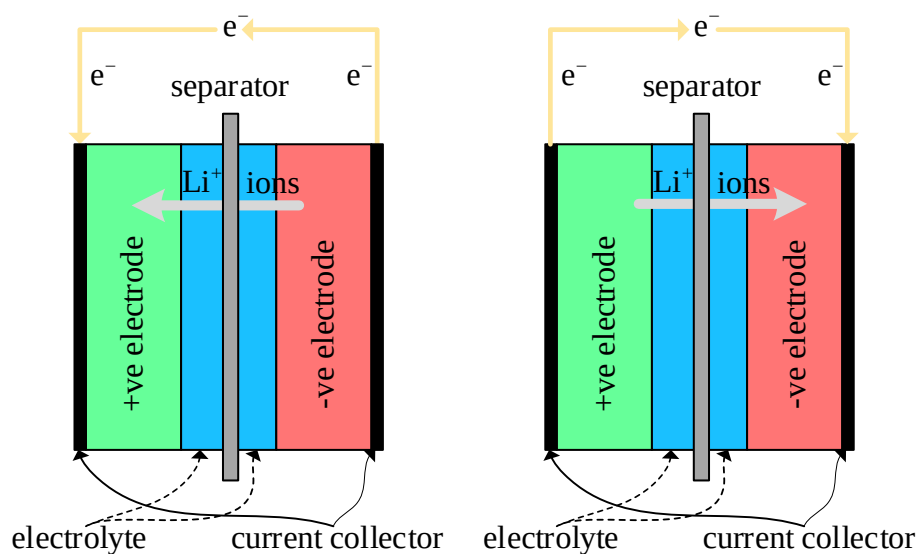


**Figure 2.2.** a) Discharge of the Li-ion battery b) Charge of the Li-ion battery.

Figure 2.2 (b) depicts the functioning of the battery during charging. The current flow has now been reversed, with reduction occurring at the negative electrode and oxidation occurring at the positive electrode. Finally, it's worth noting that the battery's functionality is dependent on the employment of two metals capable of transferring electrons (Petrovic, 2021).

### 2.1.3. Major Parameters of the Battery

The parameters defining a general battery model are as follows (Lu et al., 2013):

a. **Available power** ( $P = V \times I$ ):

It's calculated by multiplying the battery voltage ( $V$ ) by the maximum current it can handle ( $I$ ). To deliver energy throughout the working range, this useable power must be at least equal to the peak power.

b. **Accessible energy** ( $Kwh$ ):

Energy is a unit of measurement for the amount of fuel contained in or used by anything over a given length of time. A kilowatt-hour applies the same things that are known about a kilowatt to a steady load for a specified length of time. The capacity in ampere-hour ( $Ah$ ) and the voltage of the battery are used to calculate the battery's energy.

c. **State of Charge** (SoC):

The ratio of the remaining charge of the battery to the total charge when the battery is completely charged under the same standard condition might be one definition of the SoC. The SoC is frequently represented as a percentage, with 100% indicating completely charged and 0% indicating drained.

By using these important parameters, battery necessities for the EVs can be constructed in the following section.

### 2.1.4. Battery Necessities for the EVs

When choosing a battery, weight and volume are two of the most significant considerations. EVs must have a battery that, among other things, satisfies the following specifications (Julien et al., 2016):

- The amount of energy stored per mass unit ( $\text{Wh kg}^{-1}$ ) is good mass-energy. It enables the battery's autonomy to be defined.
- A good power-to-weight ratio ( $\text{W kg}^{-1}$ ) is the amount of power provided per unit of battery mass.
- The number of cycles (charge and discharge) that a battery can support determines its maximum lifespan. The number of times a battery may be recharged till it reaches a level of energy more than 80% of its nominal energy is described as its lifespan.
- Less maintenance
- A good level of autonomy.
- Availability
- A constant voltage that produces consistent results.

Lead-acid, NiCad, NiMH, sodium metal chloride (Zebra), and Li-ion are among the batteries described in this chapter that are currently commercially available for use in the EVs. For comparison reasons, these batteries are included in Table 2.1.

**Table 2.1.** Comparison of commercially usable batteries.

| Battery          | Specific energy<br>( $\text{Wh kg}^{-1}$ ) | Energy density<br>( $\text{Wh l}^{-1}$ ) | Specific power<br>( $\text{W kg}^{-1}$ ) | Current cost |
|------------------|--------------------------------------------|------------------------------------------|------------------------------------------|--------------|
| <b>Lead-acid</b> | 30                                         | 75                                       | 250                                      | 0.5          |
| <b>NiCad</b>     | 50                                         | 80                                       | 150                                      | 1.5          |
| <b>NiMH</b>      | 65                                         | 150                                      | 200                                      | 1.0          |
| <b>Zebra</b>     | 100                                        | 150                                      | 150                                      | 2.0          |
| <b>Li-ion</b>    | 150                                        | 150                                      | 300                                      | 3.0          |
| <b>Zinc-air</b>  | 230                                        | 270                                      | 105                                      | ?            |

Lead-acid is a low-cost battery that is ideal for applications that do not require a great quantity of energy storage. For example, there is no better battery for short-range vehicles like golf carts and wheelchairs that can be charged overnight. When range and performance are required, NiMH is an excellent alternative. It can also be recharged relatively quickly, resulting in a smaller and less expensive battery pack than a lead-acid battery in situations where the car is often charged. Therefore, it would be useful for hybrid vehicles or vehicles such as commuter buses or trams that regularly stop. When compared to other battery types,

the Li-ion battery provides several benefits, such as good specific energy, energy density, and specific power. Therefore, it has become one of the most popular battery types for the EVs.

#### **2.1.5. Electrochemical Models**

It is probable to obtain all the basic responses of the battery with high precision by using electrochemical models. In addition, as illustrated in Figure 2.2, they are useful for understanding distributed electrochemistry processes in electrodes and electrolytes. Nevertheless, they use a high number of partial differential equations (PDEs) with a huge number of unknown parameters to explain the battery chemistry charge and discharge carrier processes. These equations must be solved at the same time, incurring a high computational cost and a considerable memory demand. Furthermore, because of their low model robustness during extrapolation, they commonly have over-fitting issues and prevent them from being used in real-time online control (MapleSoft, 2021; Bester, El Hajiaji, and Mabwe, 2015).

#### **2.1.6. Mathematical Models**

Mathematical models identify the battery at a quite advanced level of abstraction compared to the electrochemical and circuit models. The foundation features of the battery are modelled by utilizing only a few equations. There are several mathematical models, such as Peukert's law and the Rakhmatov & Vrudhula model, that are also mentioned as follows (Jongerden and Haverkort, 2009).

##### **2.1.6.1. Peukert's Law Model**

It is the easiest model for estimating battery SoC, considering some of the nonlinear properties of the battery. It deals with the nonlinear correlation between the rate of discharge and the SoC of the battery, but it does not include modelling the recovery effect. In accordance with Peukert's Law, the battery SoC ( $L$ ) can be derived by:

$$L = \frac{a}{I^b} \quad (2.1)$$

where  $I$  is the discharge current,  $a$  and  $b$  are constants that are achieved from experiments. In ideal situations,  $b$  is equal to 1 and  $a$  is equal to the battery capacity. On the other hand,

$b$  is a value that is greater than one and  $a$  is a number that is close to the capacity of the battery. The number  $b$  would be between 1.2 and 1.7 for most batteries.

With one of the simplest mathematical models mentioned above, Peukert's Law, the battery capacity estimation can be obtained by the equation (2.2).

$$C = I^b t \quad (2.2)$$

where  $C$  is the battery capacity and  $t$  is the time.

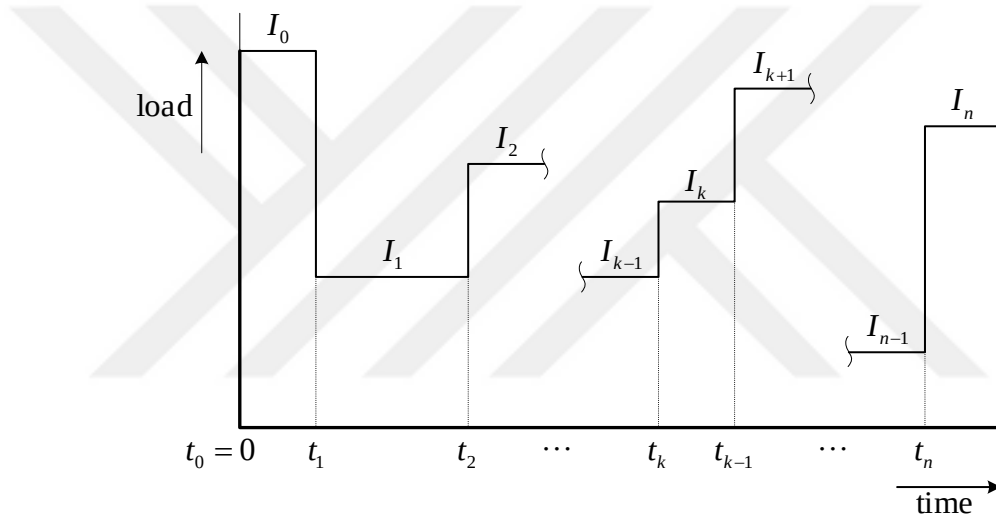


**Figure 2.3.** An example of a piecewise constant discharge profile.

For constant continuous loads, the result of the equation, which is derived by using Peukert's Law for estimating battery SoC is quite good. However, the model does not handle variable or interrupted loads. In order to overcome this situation, the renewed version of Peukert's Law is found by Rakhmatov and Vrudhula. Therefore, the equation yields:

$$L = \frac{a}{\left[ \frac{\sum_{k=1}^n I_k (t_k - t_{k-1})}{L} \right]^b} \quad (2.3)$$

From the equation (2.3),  $I$  is the change by average current up to  $t = L$ . As shown in Figure 2.3, the current profile is a piecewise constant discharge with  $t_k$  the points in time.

#### **2.1.6.2. Rakhmatov & Vrudhula Model**

They presented a Peukert model extension that works even when the discharge currents are not constant and a new analytical battery model based on the active material's diffusion into the battery. This suggested model is also appropriate for optimization methods that attempt to maximize the discharging process. The diffusion method of the active substance in the battery is characterized by the mentioned model. The model is represented by differential equations and boundary conditions using Laplace transforms. When the extended Peukert model is compared to the suggested model, the latter produces more accurate outcomes in terms of maximum and average errors (Barcellona and Piegari, 2017). The findings acquired using the Dualfoil battery simulation program were used as a baseline for this comparison (Rakhmatov, Vrudhula, and Wallach, 2002).

#### **2.1.7. Physical Models**

Physical models are created using descriptions of real-world physical events that occur in batteries. This is the most complicated type because it uses nonlinear equations to represent the electrochemical dynamics of the system in a precise and detailed manner. The lumped parameters technique, which is based on a set of differential-algebraic equations, is one of the most significant approaches for modelling the electrochemical processes of batteries. Modelling lead-acid and NiMH batteries are successfully using this method. On the other hand, the more complicated electrochemical behaviour of Li-ion batteries is not appropriate for reproduction (Barcellona and Piegari, 2017).

The porous electrode theory, which was established, set a precedent for modelling Li-ion batteries (Doyle et al., 1996). In reality, the principles of porous electrodes and concentration solutions are used in the majority of today's physical electric models. Other more recent research has attempted to decrease the order of these models using well-known model reduction approaches to create models that are less computationally. For instance, several approaches for model reduction are described, including perturbation techniques and heuristic simplifications. The primary disadvantage of these approaches is that they require the system

to be pre-processed to learn how it behaves under specific situations (Subramanian, Boovaragavan, and Diwakar, 2007).

In another study, the authors utilized the Chebyshev polynomial approach, which involved a linear combination of multiple Chebyshev polynomials to approximate the behaviour of the state variables (Bhikkaji and Söderstöm, 2001). The residue-grouping method, which was mentioned in different studies, is a one-dimensional battery model based on a different approach to model reduction (Smith, Rahn, and Wang, 2008).

According to another study, analytical transfer functions have been used to define the electrolyte concentration in addition to the residue-grouping method (Lee, Chemistruck, and Plett, 2012). Furthermore, a discrete-time model based on transcendent transfer functions has been developed to transform it into a state-space model. These models have an important drawback because they work only for a fixed point in the SoC (Lee, Chemistruck, and Plett, 2012). Contrarily, the authors of the previous study improved a new model based on the mixing method that can operate in a broad variety of temperature and SoC circumstances (Lee et al., 2014). The appropriate orthogonal decomposition technique, the Padè approximation approach, and the Galerkin method are examples of other reduction methods available in the literature (Forman et al., 2010) (Cai and White, 2008) (Dao, Vyasarayani, and McPhee, 2012). The reformulation of the Li-ion battery equations into a state-space model was a more novel method utilized to decrease the order of the models (Hu et al., 2012).

#### **2.1.8. Electrical Equivalent Circuit Models**

Electrical equivalent circuit models are a good method in terms of balancing between models, considering the other mentioned modelling techniques. In particular, they are less complex than other models and have a medium degree of accuracy. Furthermore, they are straightforward to construct for real-time applications.

As shown in Figure 2.4 (a), the most basic model consists of an ideal voltage source that ignores the inherent characteristics of the battery. As shown in Figure 2.4 (b), it is connected to a resistor in series with the voltage generator to demonstrate the internal voltage drop. These two characteristics are independent of the SoC and are constant. On the other hand, the charge and discharge voltage patterns are poorly described by these models. In addition, these

models can only be utilized in scenarios where the battery's energy is assumed to be unlimited (Kim and Ha, 1997).

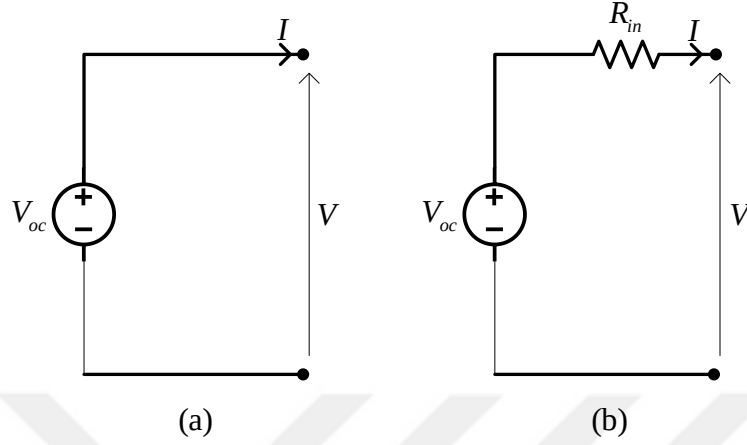


**Figure 2.4.** The simplified circuit model: (a) ideal voltage source; (b) real voltage source.

Heretofore, static models have occurred because these circuit models do not contain parameters to consider the dynamic behaviour of the battery. In addition, the so-called Thevenin model is another significant type of electric model that takes into account dynamic behaviour. The Thevenin equivalent circuit model is a battery equivalent circuit model based on the Thevenin theorem and the resistive and capacitive characteristics of batteries. As shown in Figure 2.5, the resistor  $R$  and the capacitor  $C$  can be used to model the capacitive impedance caused by polarization processes.

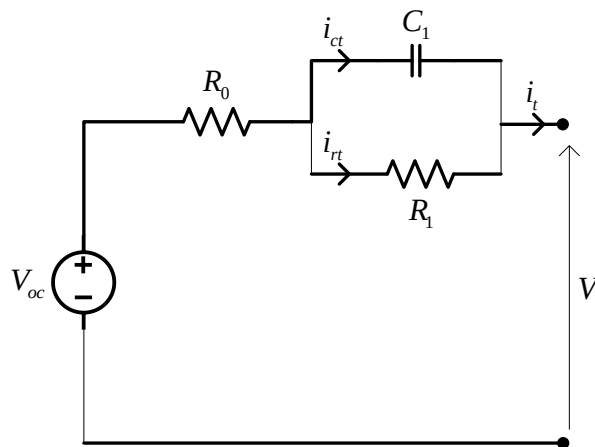


**Figure 2.5.** The Thevenin equivalent circuit model of Li-ion battery.

where  $R_0$  is the internal resistance,  $R_1$  and  $C_1$  represents the dynamics of the battery.

The following equations may be derived using the Thevenin equivalent circuit model:

$$V_{oc} = R_0 i_t + R_1 i_{rt} \quad (2.4)$$

$$i_t = i_{ct} + i_{rt} \quad (2.5)$$

$$\frac{i_{ct}}{C_1} = R_1 \frac{di_{rt}}{dt} \quad (2.6)$$

where  $V_{oc}$  is the open-circuit voltage of the battery,  $i_t$  is the operating current (when discharging it is negative when charging it is positive).

The Thevenin model has a straightforward structure, with parts that have obvious physical meaning and can properly explain the battery's operation. However, the model's flaw is that the open-circuit voltage  $E$  is fixed, but the open-circuit voltage  $E$  of the battery will fluctuate during the actual work process. As a result, as illustrated in Figure 2.6, the Thevenin model needs to be expanded with numerous more capacitors and resistors (Haizhou, 2017). For instance, the impact of the SoC is taken into account in another study (Berrueta et al., 2018).

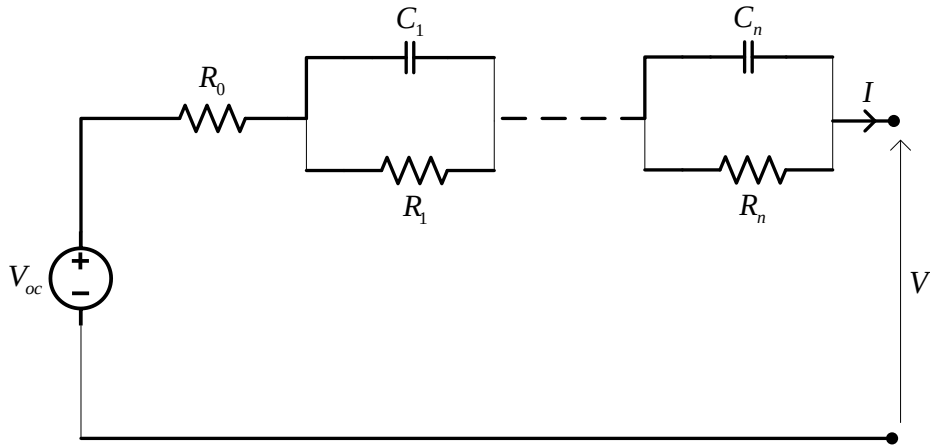


**Figure 2.6.** The extracted Thevenin equivalent circuit model of the Li-ion battery.

All of these models may be classified as a single group, with Figure 2.6 being the most generic instance. It is possible to select a different number of RC branches depending on the accuracy and the application's goal. One of the two models illustrated in Figure 2.4 is employed when the application only wants to describe steady-state behaviour. In any instance, the number of RC branches must accurately match the transient behaviour. The correctness of

the output battery voltage's behaviour is determined by the factors it is dependent on. To put it another way, it is dependent on the model variables. As a result, depending on the parameter dependency, the same model might be more or less complicated. In reality, the parameters of the same model can be influenced by one or more of the following variables: current, temperature, SoC, SoH( $K$ ), etc., as shown in the following equation.

$$p_i = f_i(I, T, L, K) \quad (2.7)$$

where  $p_i$  is the  $i$ -th parameter of the model ( $V_{oc}, R_o, R_1, C_1, \dots$ ).

In the literature, impedance models, which are another type of electrical equivalent circuit model, may be found. These models are quite similar to Thevenin models, but the method for estimating the parameters differs. Nonetheless, the well-known Randles equivalent circuit model provides the basis for one of the most significant so-called impedance models (Andre et al., 2011). The Randles equivalent circuit model is displayed in Figure 2.7. The model is constructed of a resistor  $R_{in}$  that is connected to the electrodes' internal resistance;  $C_d$ , which accounts for the effect of double-layer capacitance;  $R_{SEI}$  and  $C_{SEI}$ , which are the resistance and capacitance of the solid-state interface layer;  $R_{ct}$ , which is related to the charge transfer effect; and, the Warburg impedance  $Z_W$ , which considers the diffusion process.

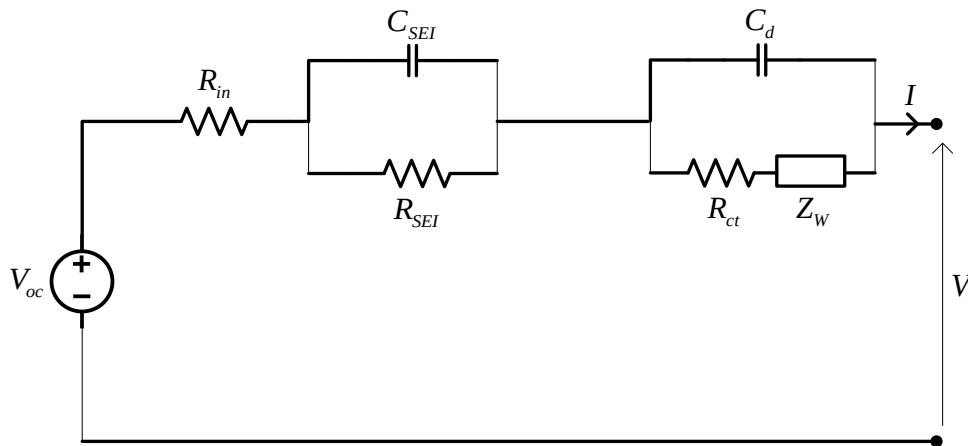


**Figure 2.7.** Randles circuit-based impedance electric model.

The Warburg impedance is defined as follows:

$$Z_w = R \frac{\tanh(\sqrt{j\omega C})}{\sqrt{j\omega C}} \quad (2.8)$$

where  $R$  and  $C$  are the two parameters characterizing the Warburg impedance itself.

## 2.2. Battery Management Systems

BMSs are products or technologies that manage and regulate batteries in some way.

Figure 2.8 depicts the structure of a typical BMS, which includes numerous sensors, actuators, algorithm controllers, signal lines, and other components.

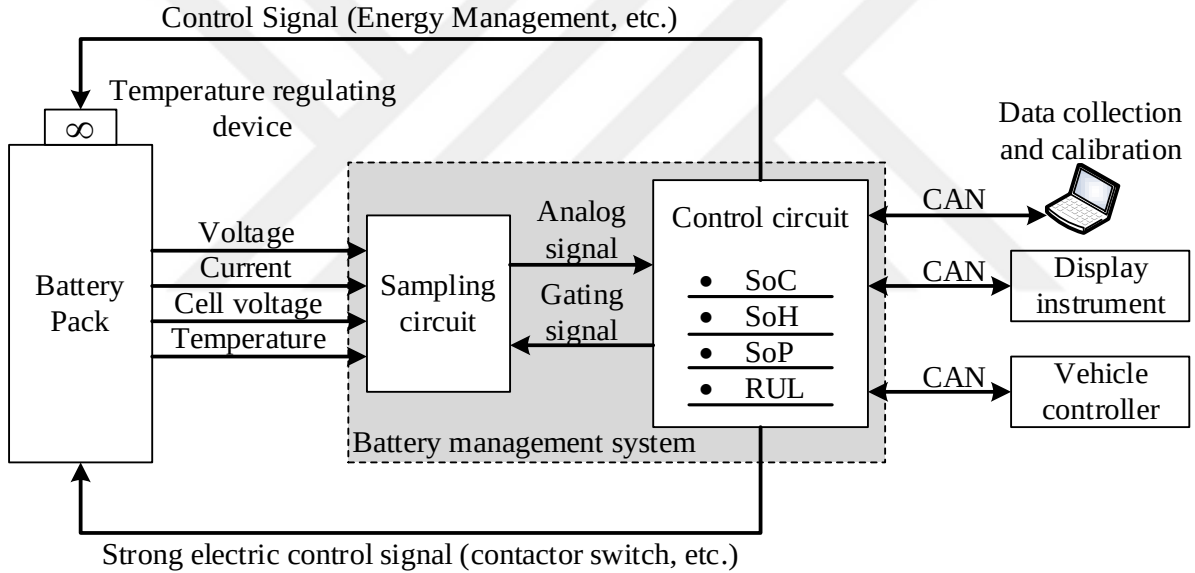


**Figure 2.8.** Structure of the BMS (Xiong, 2020).

Its primary responsibility is to ensure the safety and dependability of the battery system, to provide the status information required for vehicle control and energy management, and to intervene appropriately on the battery system in the event of an abnormal situation. Furthermore, the sampling circuit inside of the BMS is meant to gather the terminal voltage, current, temperature, and other information of the pack and each cell in real-time. It estimates the battery SoC, SoH, State of Power (SoP), and Remaining Useful Lifetime (RUL) using embedded algorithms and methods, then sends the results to the Vehicle Control Unit (VCU). Consequently, it ensures the foundation for electric vehicle energy management and power distribution control (How et al., 2019).

### **2.2.1. Main Functions of the BMS**

Data collection, state monitoring, safety protection, charging control, energy management, equalization management, thermal management, and information management are the major tasks of the BMS (Xiong, Li, and Tian, 2018; Xiong et al., 2014; Ma et al., 2018).

#### **2.2.1.1. Data Collection**

The operating environment of the battery in an EV is quite complex. The EVs must adapt to a complicated and changing environment, which means the battery must perform year-round in a complex and changeable humidity environment. Furthermore, the battery must be adaptable to the rapidly changing load as road conditions and drivers change. The BMS must gather the terminal voltage, current, temperature, and other information of the pack and each cell in real-time through the sampling circuit to properly determine the battery's working state and better apply control countermeasures.

#### **2.2.1.2. State Monitoring**

The battery is a complicated, nonlinear, time-varying system with several states that alter in real-time. The key to the battery cell and its system management and the energy management and control of the EVs is proper and yielding battery condition monitoring. As a result, the BMS must use embedded algorithms and methods to predict the states of the pack based on real-time acquired battery data. The mentioned embedded algorithms should be constructed to acquire battery states at any time, including SoC, SoH, SoP, and State of Energy (SoE), to aid in the study of the battery's real-time condition.

#### **2.2.1.3. Safety Protection**

The battery safety protection department primarily assists the battery system's online defect diagnostics and safety control. The problem type detected by the defect diagnostic algorithm based on the gathered sensor data is referred to as the battery's online defect diagnostic. Overvoltage (overcharge), undervoltage (over-discharge), smog, overcurrent, high temperature, short-circuit, connection loss, decreased insulation, and electrolyte leakage are among the battery management defects to be diagnosed, as are defects in electronic components such as sensors, actuators, and controllers. After identifying the problem types,

the BMS must issue alerts and take appropriate action to guarantee the safety of the EVs as soon as feasible.

#### **2.2.1.4. Charging Control**

Because the charging operation has such a direct impact on the battery's lifetime and safety, the BMS generally includes a charging management module that regulates the battery's charging based on real-time characteristics, battery temperature, and charger power level.

#### **2.2.1.5. Energy Management**

The driving circumstances for the EVs are complicated. Random driving maneuvers such as sudden acceleration, abrupt stopping, and uphill and downhill driving can result in complex and varied dynamic loads. The BMS must appropriately manage the energy output of the battery and the energy recovery of regenerative braking based on the acquired battery data and real-time condition information to maintain the vehicle's safe and economical operation.

#### **2.2.1.6. Equalization management**

Incoherencies across the cells are unavoidable due to the accumulation of errors in the manufacturing operation, transport and storage, and electrical elements. To effectively use the energy contained in the cells while also ensuring the pack's safety, the BMS must use an active or passive equalization approach based on the information provided by the cells to reduce cell incoherence.

#### **2.2.1.7. Thermal management**

The battery is influenced not only by the environmental temperature but also by the heat generated during charge and discharge in the normal process. Herewith, the BMS must provide the battery thermal management module with the ability to calculate the density of active heating and cooling based on temperature dispersion data and charge and discharge necessities in the pack. As a result, batteries may operate within a safe temperature range, maximizing performance and extending battery life.

#### **2.2.1.8. Information Management**

Multiple functional modules must be integrated into the BMS, and communication between modules must be coordinated. The BMS must handle and filter battery data, store essential

data, and connect with network nodes such as vehicle controllers due to the huge quantity of data flowing. With the advancement of big data technologies, the BMS must also communicate in real-time with the cloud platform to better address battery management issues and enhance management standards.

### **2.3. Modelling of the Vehicles**

In the literature, there is an extended variety of mathematical representations of vehicle dynamics, varying in the complexity of the models with different degrees of freedom. The fundamental dynamic models derived from the basics of physics can mathematically depict the force and moment dynamics (Ehsani, Wang, and Brosch, 2013). Nevertheless, in the case of important vehicle handling maneuvers, it is necessary to examine a more detailed model that takes into account the cross-coupling effects between several vehicle sub-components, kinematic restrictions, and geometrical nonlinearities. McHenry (1984) was the first to use multi-body dynamics in the study of vehicle handling issues. Afterward, it was eventually superseded by several sophisticated multibody dynamics models (Blundell and Harty, 2004). Korte (1993) presented a full multi-body model of the vehicle, including suspension geometry and tire properties, based on the Society of Automotive Engineers (SAE). Several multi-Degree of Freedom (DoF) nonlinear multi-body dynamic models that can properly describe virtually all physical features of the vehicle, inclusive suspension, and tire dynamics, have been suggested concurrently (Shabana, 2020). The following subsections include vehicle modelling methods used in studies in the literature.

#### **2.3.1. Longitudinal Vehicle Dynamic Model**

The modelling of the dominant dynamics of a generic vehicle is the first step in modelling a vehicle. Based on basic physics concepts, vehicle behavior may be quantitatively modeled. Representing a vehicle, a complicated system with many subsystems necessitates, a high level of mechanical and mathematical expertise. The basic vehicle dynamic model may be created by using fundamental mechanical concepts, particularly Newton's laws of motion. An examination of the forces operating on the vehicle body in the direction of the velocity can thoroughly characterize the vehicle's condition. The dynamic model takes into account the vehicle's progress up a hill; the forces emerging along the vehicle's body are shown in Figure 2.9 (Jazar, 2017).

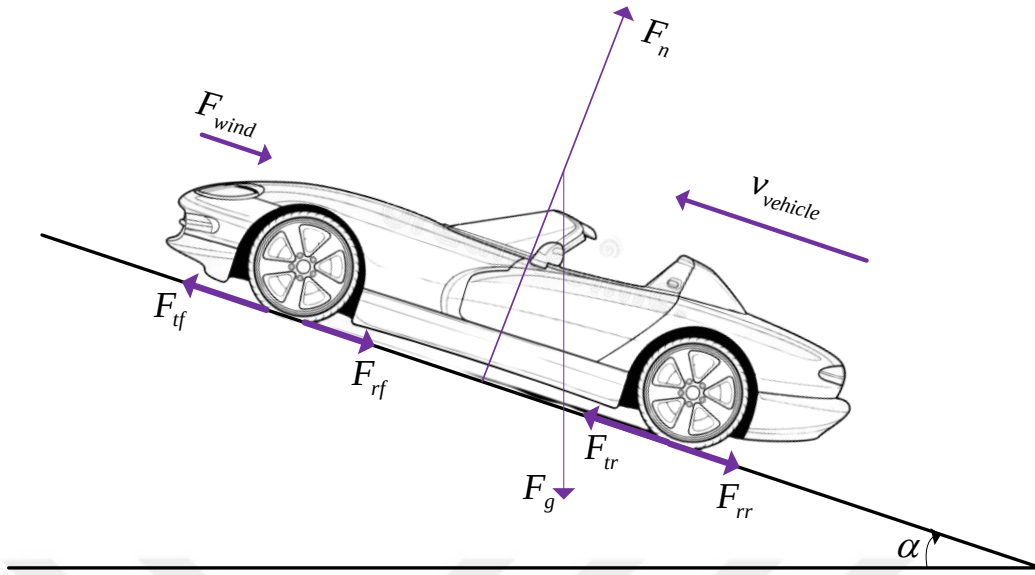


**Figure 2.9.** Forces applied on the vehicle's body.

Knowing the physical resistances of all subsystems that affect a vehicle's driving dynamics is vital for vehicle movement. The force required for the movement of the vehicle must exceed the physical resistance of the system. For the vehicle to overcome these forces, the propulsion system must accommodate the necessary force. Therefore, the required driving force for the vehicle can be calculated as follows.

$$\sum F = ma \quad (2.9)$$

$$(F_{tf} + F_{tr}) - (F_{rr} + F_{rf} + F_{wind} + F_g) = m \frac{dV}{dt} \quad (2.10)$$

where  $F_{tf}$  and  $F_{tr}$  are the tractive forces and they are positive when the vehicle direction is forward, negative when it is reverse.

Rolling resistance moments are produced by the strength provided by tires when running on hard terrain, as a result of the unsymmetrical ground response force distribution induced by hysteresis in tire materials. Reaction forces  $F_{rr}$  and  $F_{rf}$  act on the center of the wheel, counter the tractive force acting in the direction of motion, to stabilize this moment and protect the wheel from the rolling. The rolling response force can be calculated using the equation (2.11).

$$F_r = mgf_r \cos \alpha \quad (2.11)$$

where  $f_r$  is the rolling resistance coefficient,  $g$  is the gravity, and  $\alpha$  is the road angle.

Aerodynamic drag is the counter force imposed on a vehicle moving at a given velocity in the air. It can be calculated as follows:

$$F_{wind} = \frac{1}{2} \rho A_f C_D (V + V_w)^2 \frac{(V + V_w)}{|V + V_w|} \quad (2.12)$$

where  $A_f$  is the frontal area of the vehicle,  $\rho$  is the air density,  $C_D$  is the aerodynamic coefficient, and  $V_w$  is the wind velocity.

Grading resistance refers to the grading force created by a vehicle's weight as it travels up or down a slope. This force, which always operates in the downward direction with gravity and has a component counter to the tractive force, can be represented as in the following equation.

$$F_g = mg \sin \alpha \quad (2.13)$$

Several studies using the longitudinal modeling methods are available in the literature. In the first of these studies, in addition to the vehicle model, the electric motor model has been added to the system. The final model obtained has been transferred to the MATLAB/Simulink environment as sub-systems and one and two closed-loop control methods have been integrated separately. The different control structures used for the system have been designed to cope with different demands such as reference signal change, distortions that will occur in the total mass, changes in the road surface, and smooth driving for comfortable driving (Salem, 2013).

Hitesh and Sunanda (2020) have constructed the dynamic model of the EV using the MATLAB/Simulink. The ranges and energy consumption values of the EV have been determined for the various driving cycles found in the program. The effects of the rated torque and power of the engine, the voltage and energy capacity of the battery, and the transmission

gear ratio, which are the parameters of the designed vehicle, on the vehicle performance and energy consumption have been observed.

In another study, the dynamic model of the vehicle has been built using the MATLAB/Simulink environment to determine the energy consumption of the EV. The effects of important parameters of the model on vehicle performance and energy consumption have been observed. Concerning the NEDC, the vehicle consumes 15.82 kWh of energy per 100 kilometers and has a range of 177 kilometers as a conclusion of testing. In addition, WLTP, which has a different driving range apart from NEDC, has been used, and the vehicle has consumed 17.93 kWh of energy per 100 kilometers. It has a total range of 157 kilometers in this driving range, and thanks to regenerative braking, 8% of energy savings are achieved in 100 kilometers (Kiyakli and Solmaz, 2018).

Awasthi (2020) has created a vehicle model by modeling the various engine models, battery models, power models, and regenerative systems as sub-systems in MATLAB/Simulink. With the help of the designed model and the driving cycles added as an input to the system, various parameters such as engine torque, vehicle speed, and battery charge status have been obtained. The different observable factors have been examined to improve the model.

### **2.3.2. Lumped Parameter Model**

The mass, spring, and damper components compose the finite DoF model of a vehicle system in the lumped parameter modeling approach. Examples include a quarter vehicle model with two DoF, a half vehicle model with four or five DoF, a full vehicle model with seven or eighteen DoF, and so on. The number of DoF has a significant place for vehicle usage stability studies. In addition, the number of DoFs can be 2-DoF (lateral, yaw), 3-DoF (longitudinal, lateral, and yaw), 4-DoF (longitudinal, lateral, yaw, and roll), or more. As illustrated in Figure 2.10, a 2-DoF vehicle model is often used when the vehicle's lateral acceleration is less than 0.3 g (Rajamani, 2011).

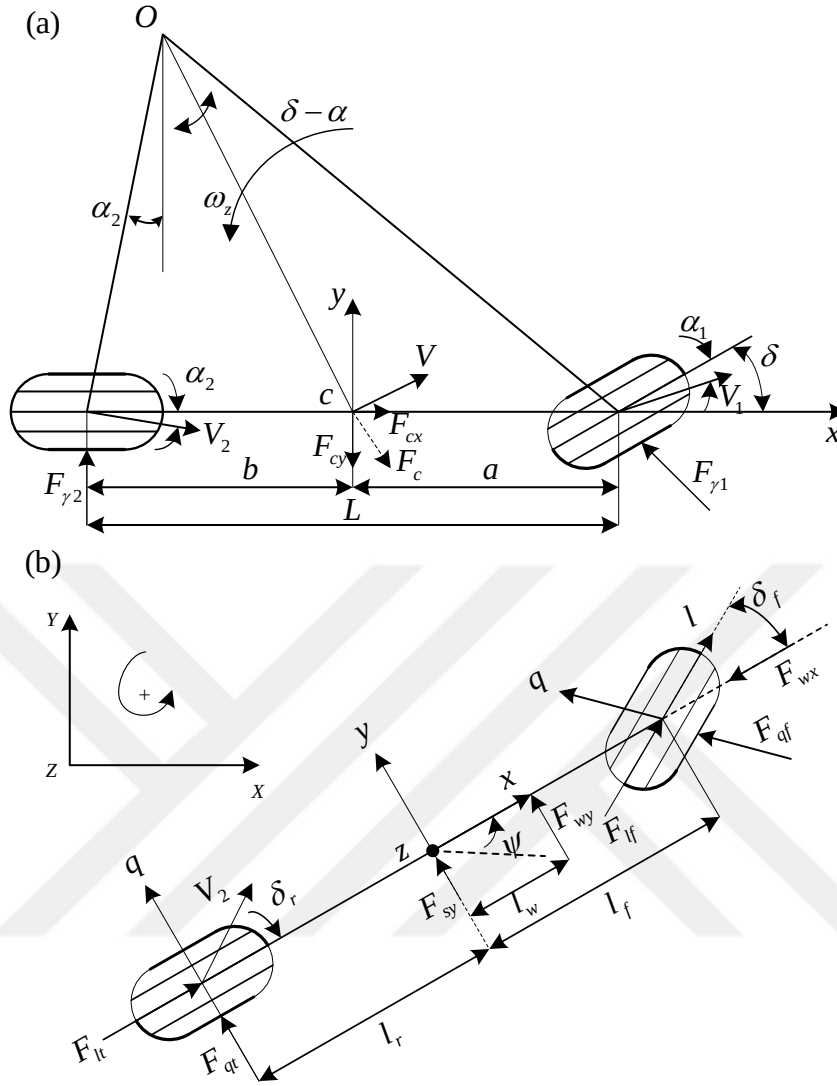


**Figure 2.10.** (a) 2DOF (lateral, yaw) model (Rajamani, 2011), (b) 3DOF (longitudinal, lateral, yaw) model (Huh, Kim, and Hong 2000).

If the cornering characteristics of the vehicle's tire are in the linear range, it has the capacity to make the vehicle's lateral, longitudinal, and yaw movements with the 3-DoF model. The model is also extensively used to investigate the dynamics of the vehicle while avoiding emergency collisions (Macek et al., 2007). Huh, Kim, and Hong (2000) have suggested an 18-DoF 6-axis MATLAB/Simulink heavy-duty vehicle with non-linear suspension characteristics and tire properties. The impact on the yawing angular speed and lateral acceleration by the middle steering axle cannot be ignored.

There is a quarter vehicle model with 2-DoF that implies the front and rear axles of the vehicle vibrate separately for the vehicle dynamics. The one-dimensional model is the most

basic quarter vehicle model that is shown in Figure 2.11. It is frequently used in the study and design of active and semi-active suspensions in vehicles (Cao et al., 2008). The lumped mass vehicle with 7, 8, or more DoF is typically three-dimensional, capable of analyzing the vehicle's vertical, pitch, and roll vibrations (Kim and Ro, 2002).

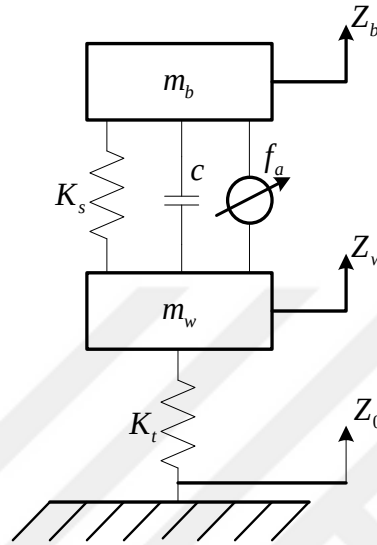


**Figure 2.11.** The one-dimensional (quarter vehicle) model (Cao et al., 2008).

### 2.3.3. Virtual Prototyping Model

The finite element (FE) requires significantly more time to calculate. The FE model is used largely to design complicated pieces like the body of the vehicle, the mounting of engines, etc. The mechanical technique and the modal synthesis method are both parts of the dynamical substructure approach. The modal synthesis approach has effectively been used to investigate the vibration noise mechanism for vehicles and the vibration transmission features of the chassis and frames. The dynamic modelling analysis of the vehicle is currently generally coupled with the lumped mass procedure, the FE technique, and the modal synthesis method (Dieter, Hiller, and Baradini, 2018).

With the advancement of computer technology and application software, Functional Virtual Prototyping (FVP) technology produces more sophisticated models for the vehicle. The main substance of the FVP is the theory of multi-organism kinematics and dynamics. In recent times, several scientists have started building the vehicle model and utilizing FVP software to study its dynamic behavior. Examples of typical multi-body dynamics software include ADAMS/Car, TruckSim, and MATLAB/Simulink (Simscape). Pappalardo et al. (2020) have

developed a three-degree-of-freedom electromechanical model for an L7 class electric vehicle using Simscape, a MATLAB/Simulink library. The forward advancement, the vertical displacement, and the angular pitch are the main degrees of freedom for the developed mechanical model. Li, Liangi, and Wang (2015) have created a virtual prototype model of a vehicle using ADAMS/Car software to examine vehicle handling stability. In addition, by applying the simulations of single lane change at different speeds and steering angles, the vehicle track, the righting rate, the lateral acceleration, and the side slip angle curve were obtained. Odhams et al. (2008) have built a complicated model of high load-bearing trucks using TruckSim software and have studied trailer safety under the active steering system. Parametric modelling is the main advantage of the multi-body model. A few test simulations of the vehicle are required to determine which vehicle parameters (vehicle geometric position, suspension stiffness, and damping characteristics) have the greatest impact on vehicle performance.

#### **2.4. AVs Technologies**

We characterize "autonomy" in a vehicle as the vehicle settling on driving choices without the intercession of a human being. All things considered, autonomy exists in different parts of a vehicle today; cruise control and Antilock Brake System (ABS) are likewise perfect representations of autonomous conduct. A few systems that were previously available in a few models—advanced cruise control, lane departure avoidance, and obstacle avoidance systems—are now fully autonomous. Close term advancements that we expect, with some first showing up as notice gadgets, are convergence impact cautioning, path change cautioning, backup parking, parallel parking aids, and transport accuracy docking. These abilities show either autonomous conduct or can be autonomous with the straightforward addition of the operation.

Zhao et al. (2018) discussed the effects of driverless, automatic cars within the framework of facilitating human life in their study. Four basic technologies in autonomous vehicles are discussed. Navigation systems, car control, environmental perception, and road planning technologies are examined. In addition, the development trend prediction of autonomous vehicles and the latest situations in this technology are clarified in this paper. Piao and McDonald (2008) investigated the development of longitudinal control assistance systems such as forward collision warning, avoidance, and adaptive cruise control based on Advanced

Driver Assistance System (ADAS). They discussed the use of ADAS functions and the issue of furthering this potential. The purpose of ADAS technology is to support drivers by automating some manual controls while driving.

Autonomous vehicles sense their environment by combining various sensors such as radar (radio detection and ranging), lidar (light detection and ranging), sonar (sound navigation and ranging), Global Positioning System (GPS), odometry, and inertial measurement units. In addition, advanced control systems identify obstacles and relevant traffic signs as well as suitable navigation routes with the help of data from sensors (Taeihagh, Araz, and Hazel, 2019; Junyan et al., 2020). Experiments with autonomous vehicles date back to the 1920s and the beginning of R&D activities in the last forty years. The first self-contained and truly autonomous vehicles were created in the 1980s by Navlab from Carnegie Mellon University. Also, the autonomous land vehicle project emerged with Mercedes-Benz in 1984. Since the 2000s, important studies on autonomous vehicles have started to be conducted. Since then, many companies and research institutions, including Tesla, Volvo, Nissan, Bosch, Toyota, Audi, Mercedes-Benz, General Motors, Continental Automotive Systems, Autoliv, Vislab (Parma University), Oxford University, and Google, have developed autonomous vehicles. In addition, the electric vehicle produced by Turkey has the autonomous driving feature (Level 3). The SAE, an automobile standardization agency, has determined levels for autonomous vehicles to prevent possible classification errors that may occur as a result of these researches and studies. This community has divided a vehicle's autonomous driving capability into six levels, from the most basic systems to 100% autonomous driving. These levels, shown in Table 2.2, are criteria that help to measure how advanced the technology of a particular autonomous vehicle is (SAE International, 2014).

**Table 2.2.**The levels of autonomous vehicle.

| SAE Level                                           | Name                 | Defining Attributes                                                                                                                                | Steering and execution of acceleration/deceleration | Monitoring the driving environment | The return performance of a dynamic driving task | System capability (drive modes) |
|-----------------------------------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|------------------------------------|--------------------------------------------------|---------------------------------|
| <i>The driver monitors the driving environment.</i> |                      |                                                                                                                                                    |                                                     |                                    |                                                  |                                 |
| 0                                                   | Not Autonomous       | The full-time performance of the driver in all aspects of the dynamic driving task is required when "enhanced by warning or intervention systems". | Driver                                              | Driver                             | Driver                                           | Disable                         |
| 1                                                   | driving assistant    | "Steering or acceleration/deceleration" provides driving mode-specific application by the driver assistance system.                                | driver and system                                   |                                    |                                                  | Some driving modes              |
| 2                                                   | partially autonomous | It provides drive mode-specific execution by one or more driver assistance                                                                         | System                                              |                                    |                                                  |                                 |

|                                                                       |                          |                                                                                                                                       |                                                                                                                                            |        |        |        |                    |
|-----------------------------------------------------------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|--------|--------|--------|--------------------|
|                                                                       |                          | systems for both steering and acceleration/deceleration.                                                                              |                                                                                                                                            |        |        |        |                    |
| <i>The automatic driving system monitors the driving environment.</i> |                          |                                                                                                                                       |                                                                                                                                            |        |        |        |                    |
| 3                                                                     | Conditionally autonomous | It provides driving mode-specific performance with an automatic driving system that includes all aspects of the dynamic driving task. | It is expected that the driver will respond appropriately to the request for intervention.                                                 | System | System | Driver | Some driving modes |
| 4                                                                     | highly autonomous        |                                                                                                                                       | If a driver does not respond appropriately to an intervention request, they can steer the vehicle over to the system and safely pull over. |        |        | System | Many driving modes |
| 5                                                                     | fully autonomous         |                                                                                                                                       | Its driving assist features are advanced enough that it can perform all driving actions no matter the conditions.                          |        |        |        | All driving modes  |

According to these levels, numerous areas of research and development are emerging for autonomous vehicles in various fields. In parallel to these areas, the processes and procedures (function requirements and regulations) established to test the functions of the autonomous vehicle have also been developed and established over the last fifteen years. However, in such a technological environment, it is vital to consider the possibility of mechanical, electronic, and software failures that could cause an accident, especially when considering the safety of people. To prove that the autonomous vehicle is safe and reduces costs, different scenarios need to be modeled and tested. Therefore, systematic testing of autonomous vehicles can be done in simulation or in the physical world. Physical tests provide test cases, and engineers can use real tools instead of models. The types of features and systems to be tested are to be considered in this direction. As displayed in

Figure 2.12, autonomous driving isn't one single innovation, but instead, an exceptionally complex system that comprises many sub-systems.

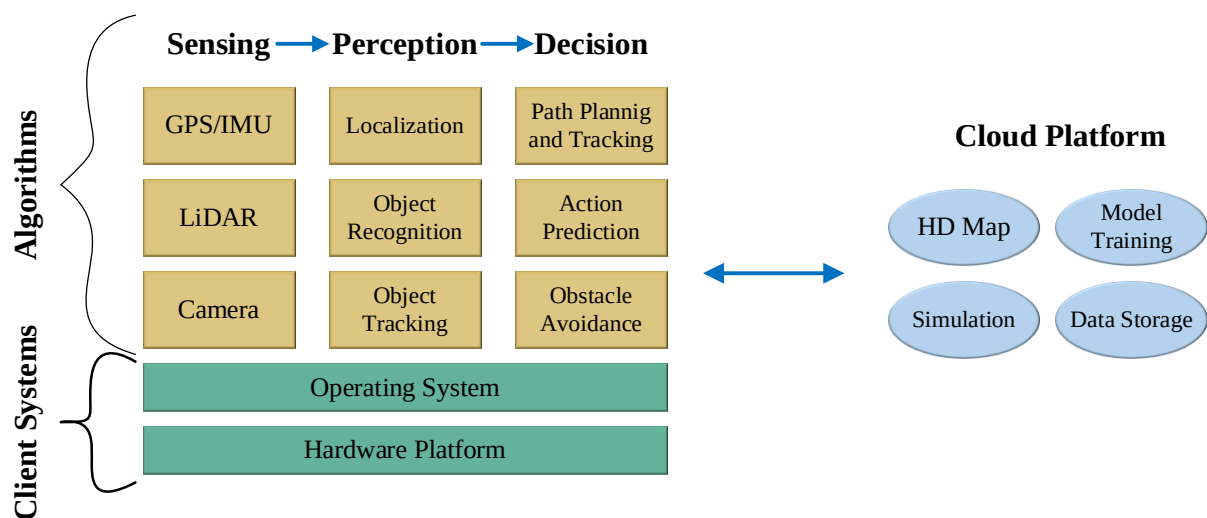


**Figure 2.12.** The structure of the AVs.

As seen in

Figure 2.12, the structure of the AVs is divided into three main components. The first of these algorithms includes sensing, perception, and decision. In addition, client systems consist of operating systems and hardware platforms, while cloud platforms consist of High-Definition (HD) maps, deep learning model training, simulation, and data storage. The algorithm subsystem removes significant data from sensor crude information to comprehend its current circumstances and to settle on choices about its future activities. Client systems combine these methods to fulfill real-time and reliability necessities. For instance, if the camera creates information at 60 Hz, the client systems need to ensure that the longest phase of the handling pipeline takes under 16 ms to finish. The cloud platform enables offline processing and storage abilities for self-driving automobiles. We can test new algorithms, update HD maps, and train improved recognition, tracking, and decision models using the cloud platform (Guruswamy et al., 2018). In addition, there are several studies about the AVs technologies in the literature that are reviewed as following in this section.

#### **2.4.1. Review of the Sensing, Perception, and Decision**

Babak et al. (2017) examined the sensor technology used in autonomous vehicles in detail. In this study, the developments in embedded software methodologies that define the process between sensory information and actuation decisions, together with the standard sensor fusion method recently discovered, are examined. Bacha et al. (2017) examined the methods used for vehicle navigation and the basic studies used in autonomous vehicle modelling in their study. In modeling the vehicle, geometric, dynamic, or kinematic approaches are considered. Based on the selected model, a control technique will be used to design a controller with high performance in terms of stability and path tracking.

Trajectory tracking and planning approaches developed for autonomous overtaking systems were examined by Dixit et al. (2018). A comparison of trajectory planning techniques was carried out based on aspects such as computational requirements and feasibility in real-world scenarios. As a result of this review, the importance of trajectory planning in high-speed overtaking was expanded to include vehicle dynamics and environmental constraints; secondly, it is characterized by accurate information about the obstacles in the environment.

The overall aim of this article is to improve the performance of advanced control methods but to demonstrate that results are valid only under well-regulated conditions.

In this study, path tracking control, strategies used in track tracking control, and criteria considered to evaluate the controller's performance are generally discussed in terms of the basic vehicle model used by Amer et al. (2017). This vehicle model has been divided into categories depending on the approach to road tracking control, and this article includes a critical review of each of these categories by examining their advantages and disadvantages. In this way, it is thought that this study can be used as a preliminary idea for the selection of controllers in the development of track tracking control by detailing the difficulties in the field of track tracking control and the situations that researchers may encounter. Several promising improvements have been proposed and developed in a nonlinear vehicle model that can be tested with hardware-in-the-loop (HIL).

Schwarting et al. (2018) discussed smart and autonomous vehicles in detail in their article. Although the latest developments in autonomous vehicle technology have made significant progress with the utilization of a few prototypes on the roads, safety challenges remain in all driving conditions. The purpose of this study is to highlight recent approaches for sensing and planning, often based on machine learning. Finally, the difficulties encountered in the management of autonomous vehicle fleets are discussed. In this article, a control method named the Feedback Local Optimality Principle is proposed for autonomous vehicles by Antonelli et al. (2019). This proposed method is based on Pontryagin's theory. At the conclusion of the article, the theory is applied to the steering and traction control of a two-wheeled vehicle. It demonstrates a certain trajectory and the ability to avoid obstacles in a complex environment.

In order for AVs to make human-like judgments, Li et al. (2018) developed a driving system that resembles that of a human. To identify, recognize, and intangibly store the data in the driving environment acquired by the onboard sensors, they employ a convolutional neural network. Abstractions are used to determine the particular instructions needed to control the vehicles with a decision-making system. As a result, we have developed a decision-making system that works effectively in a real-world environment where there are many human drives. Furthermore, their perception system using only the depth information has been

developed, rather than the unstable RGB data. The empirical findings offer a solid illustration of the effectiveness and resilience of the suggested technique.

Interactive behaviors may be generated and reacted to using a game-theoretic framework introduced by Isele (2019). As shown in this article, game-tree decision-making may be conducted by an AV, incorporating approximations and reasoning that make the tree-search computationally tractable. Stochastic rule-based traffic agents are also being developed as a benchmark for simulating merging traffic participants in busy traffic situations to evaluate the model developed by the researchers.

Liu et al. (2020) have constructed the overtaking strategy on a three-lane road for AVs depending on the heuristic planning reinforcement learning approach. The suggested decision-making controller is concerned with ensuring that the AVs continue to operate safely and efficiently. It is used to make highway overtaking decisions using Dyna-H, a modified Q-learning algorithm with a heuristic planning strategy. Three alternative heuristic procedures have been developed to increase learning efficiency and compare performance. In an unpredictable environment, a vehicle's lane shift and speed selection are determined by this algorithm. When it comes to the autonomous overtaking situation, Dyna-H is evaluated by comparing it to the reference and conventional learning techniques. A driving dataset is used to test Dyna-decision-making H's algorithms in an open-source environment. Convergence rate and control are shown to be better with the suggested decision-making technique.

Baras et al. (2019) demonstrate their self-driving car implementation, which uses a Raspberry Pi and a LIDAR module for indoor navigation to provide a robust answer to this challenge. Using their car, they are able to navigate in unfamiliar terrain and avoid obstacles. Instead of using Computer Vision (CV) algorithms for obstacle identification, their car uses a single LIDAR sensor to travel in low-light conditions. They ran a number of tests and were able to confirm that our implementation was a success. Arvind and Senthilnath (2019) suggested a reinforcement learning approach based on Multi-Layer Perceptron-State Action Reward State Action (MLP-SARSA) for dynamic obstacle identification and avoidance in AV navigation. The pygame library is used to train an MLP-SARSA system in a dynamic urban simulation environment. When it comes to AV navigation in dynamic environments, MLP-SARSA is more effective than traditional Q-learning and SARSA reinforcement algorithms.

According to Wang et al. (2019), an enhanced artificial potential field might be used to provide a collision-free route for AVs. To create this algorithm, it is necessary to first develop a safety model of obstacle avoidance behavior by examining the obstacle avoidance behavior of a human driver. The artificial potential field approach is then developed on the basis of the safety model, and the repulsive field range of barriers is reconstructed using the safety model. Finally co-simulation as well as real-world vehicle testing, are carried out to evaluate the performance of the suggested method. Roads, kinematics, and dynamics are all satisfied by a route provided by the algorithm. The suggested route planning technique for obstacle avoidance situations is also tested in real-time to determine its real-time performance, efficacy, and feasibility, among other things.

A HD map-based motion planning system was suggested by Jian et al. (2019). OpenStreetMap's HD map allows the local motion planner to learn about traffic conditions and plan and optimize routes appropriately. Additionally, they suggest an inertia-like route planning technique to increase both the safety and comfort of obstacle avoidance. For the 2018 Intelligent Vehicles Future Challenge, they tested their suggested solution on their self-designed autonomous driving experimental platform "Pioneer". Without operator assistance, the "Pioneer" finished all the races and won the title. Considering the three levels of perception, route selection, and guidance control, Laghmara et al. (2019) developed a global architecture. An evidentiary occupancy-grid-based technique is employed for dynamic obstacle identification at the level of perception of the surrounding environment. Objects' positions are taken into account while generating trajectories. A Sigmoid smooth trajectory function is used in the development of the later model. Once this obstacle avoidance trajectory has been determined, it is used by the control guidance to create the proper steering angle. The whole method has been tested on their experimental test vehicle. The experimental results suggest that the proposed method is successful.

For nonlinear systems with restrictions, Sanchez et al. (2021) created a comprehensive model predictive path-following control formulation that included an obstacle avoidance method. Nonlinear Model Predictive Control (NMPC) is used in this study to concurrently optimize route following and obstacle avoidance in a single optimization problem. The fundamental concept is to expand current NMPC controllers with an auxiliary trajectory that keeps the

following optimization problems possible even when the reference curve is impractical or may be discontinuous. To demonstrate the effectiveness of the suggested control technique, simulations are run using an AV subjected to input and state limitations.

Cooperative integration of Reinforcement Learning (RL) with supervised learning for AVs in dynamic and uncertain environments was studied by Bin Issa et al. (2021). Unhindered AV maneuvering has been made possible by the Faster RCNN, a supervised learning method that recognizes the ambient environmental barriers. In contrast, using a Reinforcement Learning technique called Double Deep Q-Learning, the autonomous agent may learn to make good navigation choices by observing the constantly changing surroundings. For the most part, the model under consideration was evaluated in a simulation of the current world. It has shown the overall efficiency and efficacy of AVs of maneuvers.

AV obstacle avoidance was enhanced by Yan and Lin's (2021) work on safe route planning. An enhanced technique based on the Rapidly Exploring Random Trees (RRT) algorithm, which integrates route pruning, smoothing, and optimization with geometric collision detection, has been shown to increase the efficiency of planning. The enhanced RRT algorithm has predicted a route for an AV to follow in the experimental analysis using both a pure pursuit steering controller and a proportional-integral speed controller. With an average tracking control deviance of 5.2 percent of the vehicle's width, the vehicle is able to effectively trace the course and safely arrive at its destination. Additionally, lane changes have been subjected to route planning, which ensures that the average departure from the lane during and after lane changes stays within 8.3 percent of the vehicle width.

Arvind and Senthilnath (2020) were able to make a choice utilizing RL methods obtained from the sensor data. For efficient vehicle future action prediction, the policy-free, model-free Q-learning-based RL algorithm with MLP has been implemented and trained depending on the current state of the vehicle. The number of ultrasonic radar sensors used to identify barriers in a simulated metropolitan environment was studied. In the experiment, it was shown that MLP Q-learning with ultrasonic sensors was more accurate than traditional Q-learning methods with ultrasonic sensors.

Using a model predictive controller (MPC) model, Leman et al. (2019) explored creating monitoring route controllers based on highway situation avoidance obstacles with varying speeds. The controller must be able to automatically avoid static road barriers and trace the intended course by managing the vehicle's front steering angle. A controller architecture was built to provide steering input to the vehicle dynamic system, depending on X, Y, global position, and yaw rate. The Stanley controller was compared to the MPC nonlinear controller in low and medium-speed situations for the route tracking approach. The findings of the simulation reveal that the MPC controller has the benefit of a tracking route that is effective at mid and high-speed operations.

On a straight road or in an unknown environment, Feng et al. (2020) suggested a method for avoiding obstacles that can be avoided online. To determine the vehicle's needed steering angle, the algorithm makes use of an extended circle notion. A lateral controller is needed to maintain the required steering angle. To ensure that the vehicle maintains its position after the obstacle avoidance manipulation, a restriction on the terminal point pose has been introduced. Online obstacle avoidance is guaranteed by the method, which is computationally intensive. It has been tested in a laboratory environment, and the findings demonstrate that the suggested algorithm is more efficient than dynamic window approach and even better than genuine human drivers.

### **3. MATERIALS AND METHODS**

This chapter of the dissertation introduces the modelling of the battery, the topology and modelling of BMSs for the EVs, and the modelling of the EVs by using Matlab/Simulink and MapleSim software. In addition, artificial neural networks are presented in detail to estimate the DR of the vehicle. Finally, the last subsection of this chapter defines intelligent control methods that are developed for the AVs using Python.

#### **3.1. The Modelling of the Battery**

In this subsection, the modelling of the battery has been constructed in Matlab/Simulink and Mapelsim. The input of the battery contains three different inputs that are step, pulse, and sinusoidal. The outputs of several software programs as a consequence of the inputs supplied to the model are compared in order to demonstrate the accuracy of the simulated battery.

##### **3.1.1. The Simulation of the Battery in the Simulink**

Li-ion battery which is the most popular in both EVs and renewable energy systems is used in this simulation. The crucial advantages of the Li-ion battery against other battery types are greater efficiency and energy density, increased nominal voltages, lower specific weights, increased lifetime, faster and more efficient charging, smaller size, no need for maintenance, and greater resistance to external conditions. Therefore, accurate battery models are vital to the design and simulation of hybrid/electric vehicle propulsion systems. The inside of the battery of the vehicle is shown in the following figure.

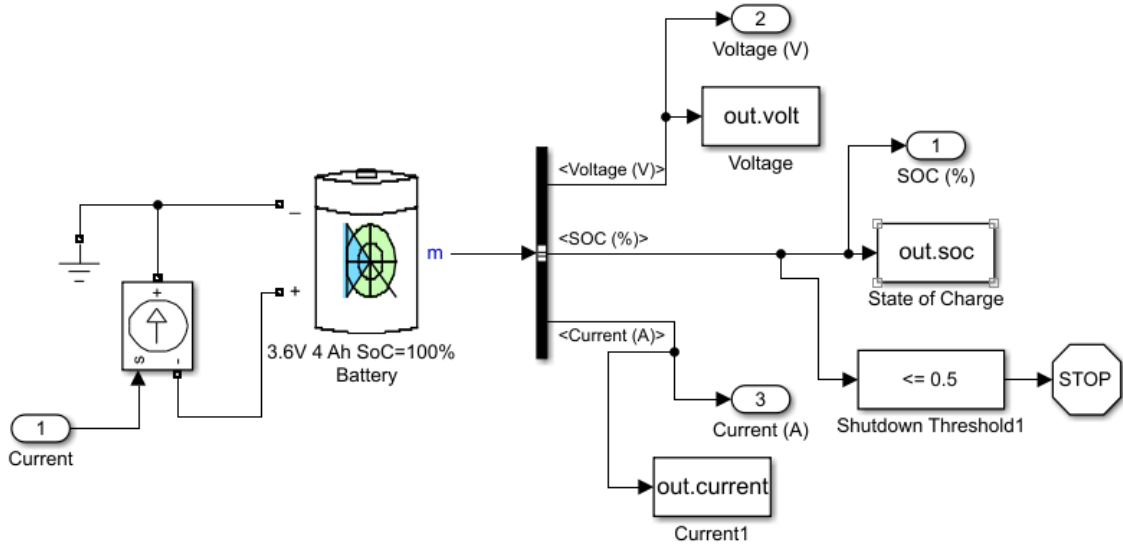


**Figure 3.1.** The system of the battery.

Using a general dynamic model, the battery block simulates the most common rechargeable battery kinds.

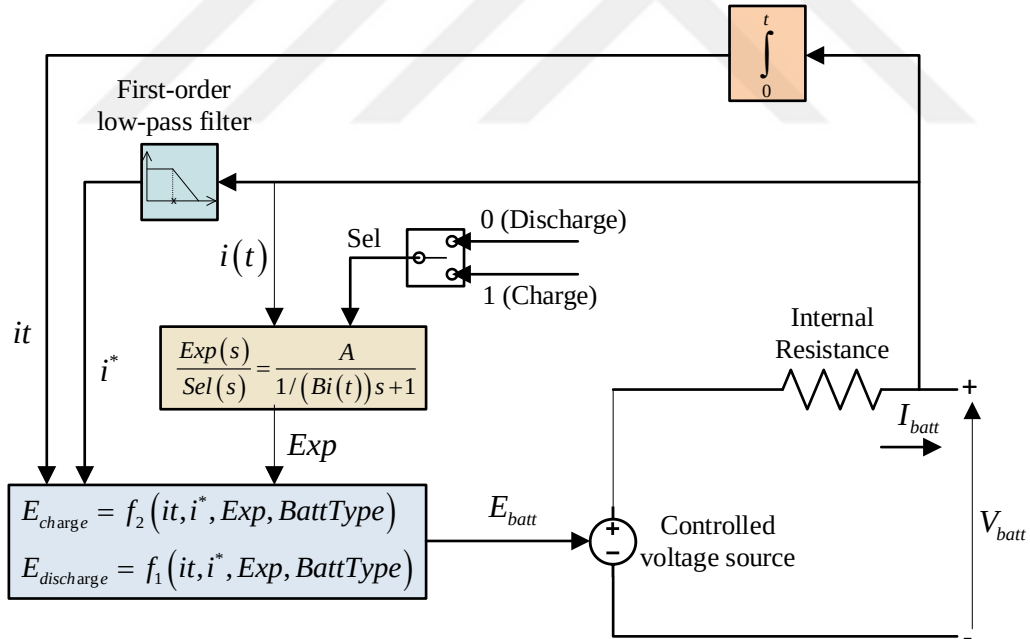


**Figure 3.2.** The equivalent circuit of the block.

The figure above displays the analogous circuit that the block models. The following section contains the charge and discharge characteristics of the Li-ion cell.

### 3.1.1.1. Charge and Discharge Characteristics

The parameters of the circuit may be changed to reflect a certain battery type and its discharge characteristics. A typical discharge curve has three portions, as seen in the figure below.

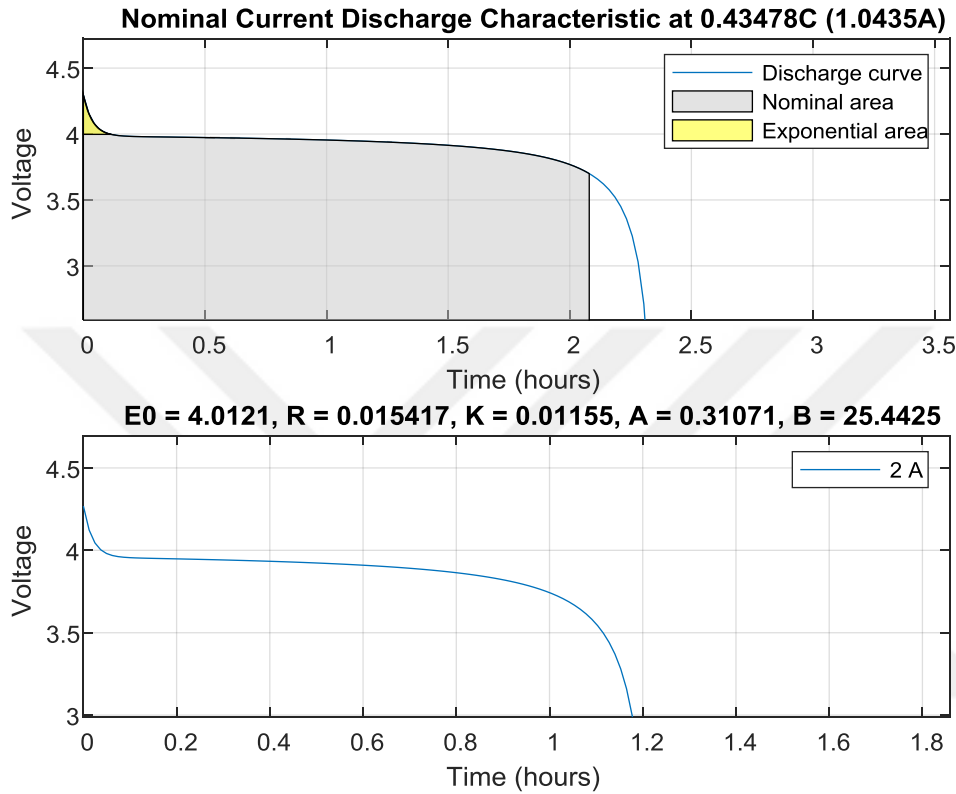


**Figure 3.3.** Typical discharge characteristics.

When the battery is charged, the first part illustrates the exponential voltage drop that occurs. The breadth of the drop is determined by the kind of battery used. Until the battery's voltage falls below its nominal value, the second section reflects the charge that can be taken from the battery. Subsequently, the battery's entire discharge is shown in the third part, which occurs when the voltage quickly declines. When the battery current is negative, the battery recharges in accordance with the charge characteristic of the battery that is shown in Figure 3.4.

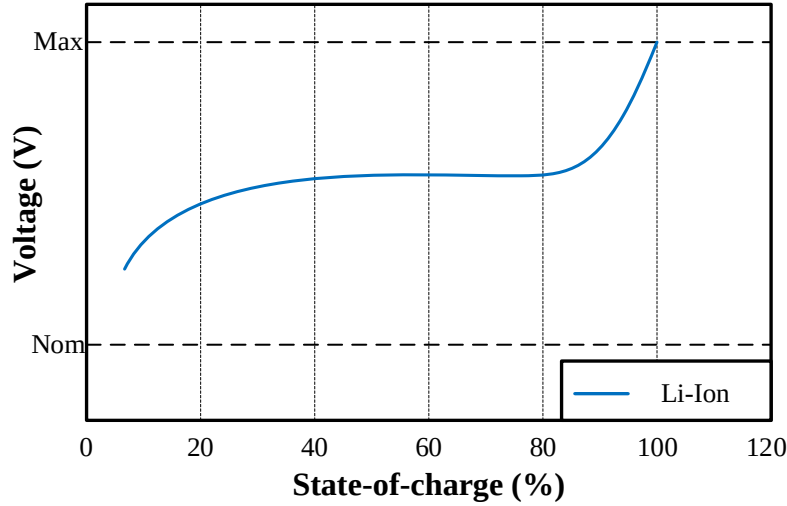


**Figure 3.4.** Typical charge characteristics.

The discharge characteristics serve as the basis for deriving the model parameters. The properties of discharging and charging are considered the same in both cases. The SoC for a battery is a measure of the battery's charge that is stated as a percentage of the battery's full charge. To put it another way, the SoC's numerical complement is called the Depth of Discharge (DoD), such that  $DoD = 100 - SoC$ . When the current is between 0 and 2 C, and when it is 0 to 5 C, the model's charge and discharge dynamics have a maximum inaccuracy of 5% (when SOC is between 10% and 100%).

### 3.1.1.2. Parameterization

The discharge curve at the surrounding temperature, which is distinct from the ideal temperature, as well as the thermal response characteristics of the Li-ion battery type are necessary to accurately reflect the temperature impacts of this battery type. In most cases, additional discharge curves are not available in the datasheet and may need experimental tests to be performed to access them. The ability to function in the low temperatures of the Li-ion battery is shown in Figure 3.5.

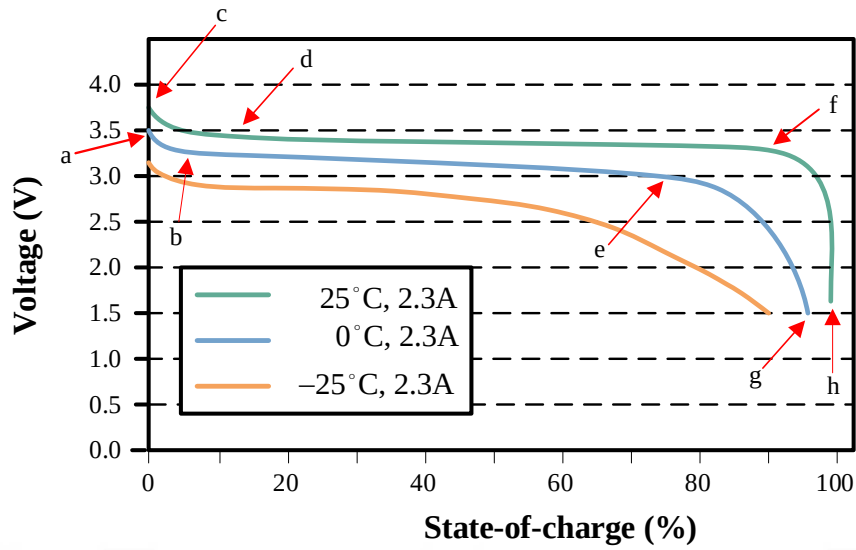


**Figure 3.5.** The ability to function in the low temperatures.

Using the A123 Li-ion battery datasheet as an example, the following parameters (Table 3.1) are based on the battery model's specifications from the datasheet.

**Table 3.1.** A123 Li-ion temperature-dependent battery model from the datasheet.

| Parameters                                         | Value             |
|----------------------------------------------------|-------------------|
| Nominal voltage (f)                                | 3.22 V            |
| Rated capacity                                     | 2.3 Ah            |
| Maximum capacity (h)                               | 2.3 Ah            |
| Fully discharge voltage (c)                        | 3.7 V             |
| Nominal discharge current                          | 2.3 A             |
| Internal resistance                                | 10 mΩ             |
| Capacity at a nominal voltage (f)                  | 2.07 Ah           |
| Exponential zone (d)                               | [3.4 V, 0.23 Ah]  |
| Nominal ambient temperature                        | 25 °C             |
| Second ambient temperature                         | 0 °C              |
| Maximum capacity at 0 °C (g)                       | 2.208 Ah          |
| Initial discharge voltage at 0 °C (a)              | 3.45 V            |
| Voltage at 90% maximum capacity at 0 °C (e)        | 2.8 V             |
| Exponential zone at 0 °C (b)                       | [3.22 V, 0.23 Ah] |
| Thermal resistance, cell-to-ambient (estimated)    | 0.6               |
| Thermal time constant, cell-to-ambient (estimated) | 1000              |

By using these parameters, the equation of the battery is constructed in the following section.

### 3.1.1.3. Equation of the Battery

The following equations are used in the model for Li-ion batteries.

For discharge model ( $i^* > 0$ )

$$f_1(it, i^*, i) = E_0 - K \frac{Q}{Q - it} i^* - K \frac{Q}{Q - it} it + A e^{(-B \cdot it)} \quad (3.1)$$

For charge model ( $i^* < 0$ )

$$f_2(it, i^*, i) = E_0 - K \frac{Q}{it + 0.1Q} i^* - K \frac{Q}{Q - it} it + A e^{(-B \cdot it)} \quad (3.2)$$

where  $E_0$  is the constant voltage (V),  $K$  is the polarization constant (Ohms),  $i^*$  is the low-frequency current dynamics (A),  $i$  is the battery current (A),  $it$  is the extracted capacity (Ah),  $Q$  is the maximum battery capacity (A),  $A$  is the exponential voltage (V) and  $B$  is the exponential capacity ( $\text{Ah}^{-1}$ ).

### 3.1.2. The Simulation of the Battery in the MapleSim

This study has been made possible by the modelling, simulation, and analysis tool MapleSim, which includes, among other things, a battery-specific library for use in the calculations. It offered us a diverse selection of battery models to choose from. This, together with the many tools for creating bespoke components, has enabled us to customize our models and simulations to meet our requirements and, as a result, achieve the objectives of this research.

#### 3.1.2.1. The Maplesim Environment

The combination of MapleSim and Maple provides us with more flexibility in the development of our models. Because of the flexibility of the system, we are no longer limited to the built-in components or analyses. Through the use of its comprehensive programming and analytic environment, users may run simulations, adjust analyses or script totally new ones, conduct optimizations, construct powerful symbolic control laws, and examine models in ways that are not feasible with other software packages. Moreover, it enables the creation of

component diagrams that describe physical systems in a graphical form. This may be of great assistance to many engineers who do not find many current simulation tools obvious when it comes to physically modelling systems. MapleSim automatically builds model equations from a component diagram and executes high-fidelity simulations using a combination of symbolic and mathematical techniques, according to the user's preference.

In order to reduce the problems that MapleSim has automatically generated, the program employs symbolic methods such as index reduction, differential elimination, separation of independent systems, and the removal of redundant systems. The following are the two most significant advantages of symbolic simplification:

- Symbolic simplification, which involves symbolically resolving algebraic loops and lowering the complexity of Differential Equations (DEs), makes many problems that were before unsolvable numerically manageable.
- The numerical solvers get the simplified equations in a form that is both computationally efficient and easy to read. In certain circumstances, this may result in a reduction of many orders of magnitudes in the entire simulation time.

For studies that need HIL simulation, such as the installation of batteries in electric vehicles, computational efficiency is especially significant since it allows for the development of higher fidelity models while maintaining accurate real-time performance.

### **3.1.3. The Simulation of the Battery in the Matlab**

We wanted to make certain that the characteristics of the electrical equivalent model were accurate once it was selected as the reference model from which our research could be constructed. Matlab's performance simulation tool, which we were planning on using, would enable us to have an exact depiction of the performance of our battery model. Using the electrical equivalent model that was created in Simulink and MapleSim, the parameters utilized in Matlab were derived from these models. When modelling a battery using the Matlab software, the goal is to build a first-order basic model of the battery. In addition, it is desired to observe that the output produced by the developed model is the same as the output produced by other programs that have been tested.

To obtain the first-order basic model of a battery, we must first investigate the relationship between the battery's charge, capacitance, and voltage parameters, as well as the connection between these parameters. This relationship is shown in the following equation.

$$Q(t) = C \cdot (v_i(t) - v_o(t)) \quad (3.3)$$

where  $v_i(t)$  is the input of the battery in the form of voltage,  $v_o(t)$  is the output of the battery in the form voltage,  $C$  is the capacitance of the battery,  $Q$  is the charge inside the battery.

Taking time derivative of both sides in the equation (3.3) yields:

$$\frac{1}{C} \frac{dQ(t)}{dt} = \frac{d(v_i(t) - v_o(t))}{dt} \quad (3.4)$$

Since change in input voltage will have direct proportional effect on the output voltage, the above relationship can be approximated as:

$$\frac{1}{C} \frac{dQ(t)}{dt} = v_i(t) - v_o(t) \quad (3.5)$$

Suppose the voltage is fixed to  $v_i(t) = \bar{v}$ , where  $\bar{v}$  is a fixed value. As a result, the change inside the battery will converge to  $\bar{Q}$ , so that the energy provided for the load will be  $v_i(t) = v_o(t) = \bar{v}$ . So, small variation in the voltage can be approximated as in the change stored in the battery and output energy.

$$v_o(t) - \bar{v} = \alpha [Q(t) - \bar{Q}] \quad (3.6)$$

Substitute the equation (3.6) into the equation (3.5) yields;

$$\frac{1}{C} \frac{dQ(t)}{dt} = v_i(t) - \bar{v} - \alpha [Q(t) - \bar{Q}] \quad (3.7)$$

Denote that  $Q(t) - \bar{Q} = y(t)$  and  $u(t) = v_i(t) - \bar{v}$  then:

$$\begin{aligned} \frac{1}{C} \frac{dy(t)}{dt} + \alpha y(t) &= u(t) \\ \frac{dy(t)}{dt} + \alpha C y(t) &= C u(t) \\ \frac{dy(t)}{dt} + a y(t) &= b u(t) \end{aligned} \tag{3.8}$$

where  $a = \alpha C$  and  $b = C$ .

At the steady state  $\frac{b}{a} = \frac{C}{\alpha C} = \frac{1}{\alpha}$  the capacitance does not have any impact on the system gain, because increase at the input leads to almost proportional increase at the output. The gain of the system is proportional to  $\alpha$ , if it gets large, then the gain of the system reduces, because the amount of the charge inside the battery reduces.

### 3.2. The Battery Management Systems

In this section of the dissertation, the topologies of the BMS, which have been discussed in detail in the previous part and are considered to be one of the most significant systems in the car, are described. It also discusses the BMS system that has been designed in the MATLAB/Simulink environment.

#### 3.2.1. Topology of the BMS

The EV's design must often fulfill specific dynamic performance criteria, such as acceleration, climbing ability, and top speed. Consequently, to satisfy the technological necessities of vehicle design, the cell is generally linked in series or parallel. For instance, the Tesla Model S utilizes Panasonic's NCA 18650 ternary nickel–cobalt–aluminum Li-ion batteries, which have a nominal capacity of 3,100 mAh. Over 7,000 cells were utilized in the series and parallel to construct a battery pack that was put on the underframe. For large-scale battery management, the topology of the BMS is critical (Tesla, 2021).

The topology of the BMS contains the battery monitoring circuit (BMC) and the battery control unit (BCU) to form the hardware circuit. The BMS may be split into centralized and distributed topologies based on the structural interaction between the BMC, BCU, and cells. In the centralized BMS architecture, the BMC and BCU are combined on a single printed circuit board (PCB) to perform data collection, computation, safety monitoring, switch management, charge and discharge control, and communication with the vehicle controller, among other tasks. The centralized BMS is frequently utilized in applications with limited capacity batteries, low total voltage batteries, and compact battery systems. As seen in Figure 3.6, the centralized BMS topology is expressed as transmitting the measured signals from each cell to a single PCB (Xiong et al., 2018).

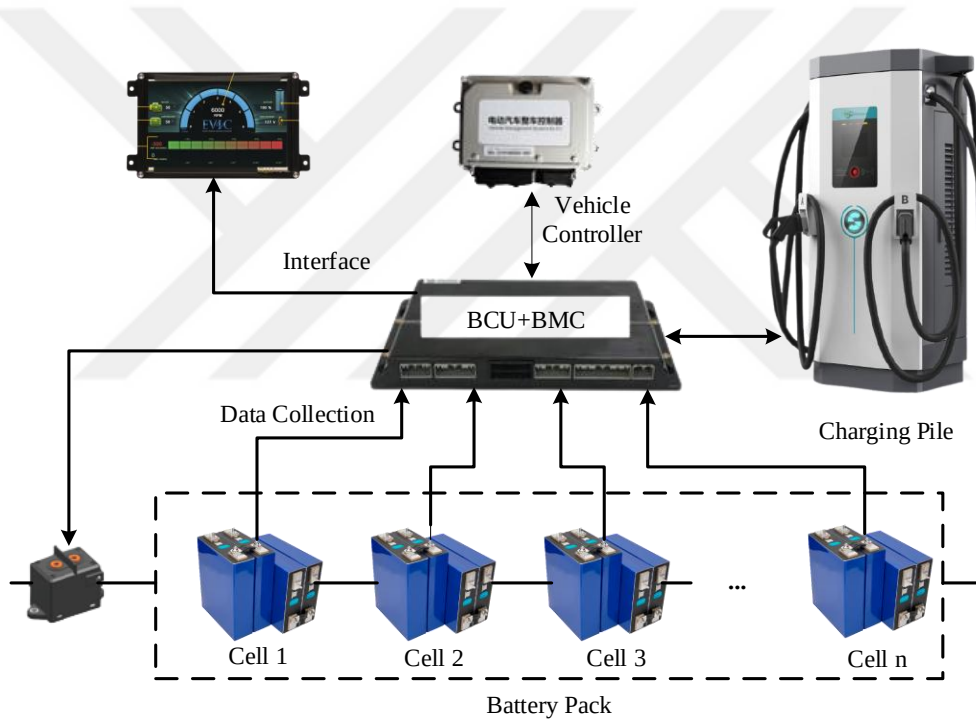


**Figure 3.6.** Structure of centralized BMS topology.

The following are some of the benefits of a centralized BMS topology:

1. To ensure synchronous data gathering, high-speed intra-board communication is advantageous.
2. It features a compact design and excellent anti-interference capabilities.
3. The cost is minimal, and the BMS's whole job may be done with just one package.

The centralized BMS topology, on the other hand, has the following drawbacks:

1. It includes a couple of complicated cables and connections.

2. When short-circuits and over-currents take place in various areas of the battery system, it is challenging to preserve the system.
3. Due to the high-voltage safety, a sufficient safety margin between separate channels must be left, resulting in an excessively large PCB.
4. Because all components are combined into a single PCB, scalability and maintainability are limited.

In contrast to the centralized architecture, the distributed BMS's BCU and BMC are constructed individually, as shown in Figure 3.7. The BCU is in charge of defect detection, predicting battery condition, switch control, charge and discharge control, and interaction with the vehicle controller. The BMC controls cell voltage, current, and temperature, as well as security and uniformity. A controller area network (CAN) bus connects the BCU with the BMC, allowing any BMC to interact with the BCU. Furthermore, each BMC PCB corresponds to a CAN bus node and is linked to the appropriate cell independently. As a result, communication between BMCs may be built (Jiang and Zhang, 2015).

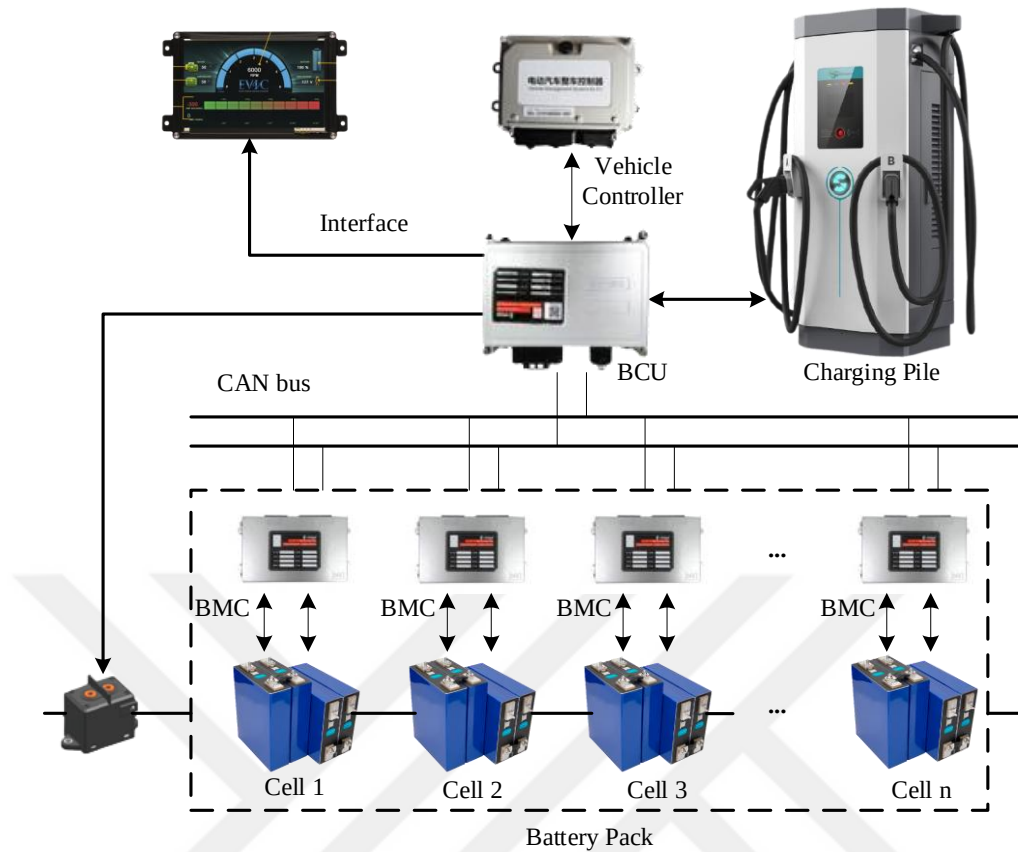


**Figure 3.7.** Structure of distributed BMS topology.

The following are some of the benefits of a distributed BMS topology:

1. Because the functions of data gathering and computation are separated, errors are easy to detect, and calculation efficiency is excellent.
2. The system's structure has been reduced, and the layout is adaptable and applicable.
3. The number of controlled batteries can be raised or lowered by relocating the BMC PCB near the relevant battery and connecting or disconnecting the reserved CAN bus interface.

The distributed BMS topology, on the other hand, has the following drawbacks:

1. Because there are so many components, there are more PCBs and installation, debugging, and disassembly processes.
2. The design criteria of the communication network are difficult to satisfy since network latency is easily formed and affects the synchronization of the gathered data.

At the moment, the distributed BMS architecture is the most often utilized in EVs. For instance, commercial EVs such as the Tesla Model S, BMW i3, Roewe eRX5, and BYD Qin have used this architecture.

### **3.2.2. Construction Process of the BMS**

Figure 3.8 depicts the fundamental growth phase of the BMS. It can be observed that the construction of both the battery and the BMS is based on the vehicle's power, energy, and other needs. To acquire the performance properties of the battery, it is required to first pick the battery and execute a series of feature tests and cycle life tests on the cell. The next step is to define the battery grouping technique, which includes the number of cells connected in series and parallel, as well as the exact layout. The next step is to define the battery grouping technique, which includes the number of cells connected in series and parallel, as well as the exact layout (Li et al., 2017).

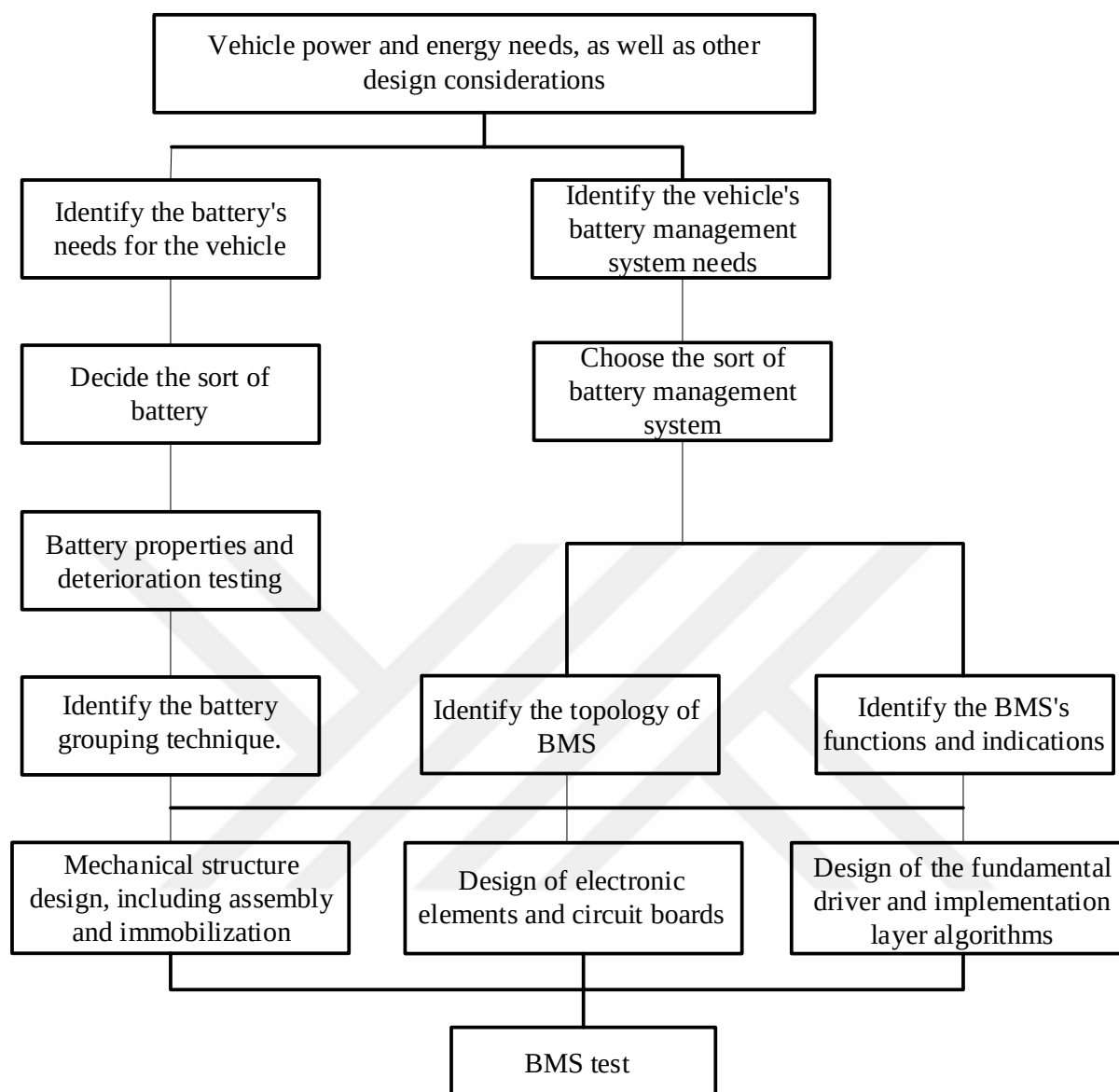


**Figure 3.8.**The fundamental construction process of the BMS.

In the construction of a BMS, the type must first be determined, followed by the system topology and the fundamental functions and indicators that must be satisfied, such as being water-resistant, dust-resistant, seismic, and so on. Following the determination of the battery grouping technique, it is necessary to accomplish the BMS topology and fundamental functions and indications, the mechanical structure design, electrical elements, and PCB design, and the system's underlying driver and application layer algorithm design. Before system development can be finished, several BMS tests must be performed to guarantee the entirety and security of the system design.

### 3.3. The EVs Modelling in the Matlab

In this section of the dissertation, the EVs' linear and nonlinear models are constructed by using Matlab. In addition, the modelled EVs are combined with the Direct Current (DC) servo motor and are analysed deeply. The aim of the subsection is to investigate the dynamics of the vehicle with respect to the various scenarios.

#### 3.3.1. The Linear Model of the Vehicle

For a vehicle to enforce its motion, the vehicle's propulsion system must overcome some physical forces: inertia force, rolling resistance, wind, and gravity. The forces acting on the vehicle are shown in Figure 3.9, which are dynamic vehicle parameters that affect the vehicle's motion.

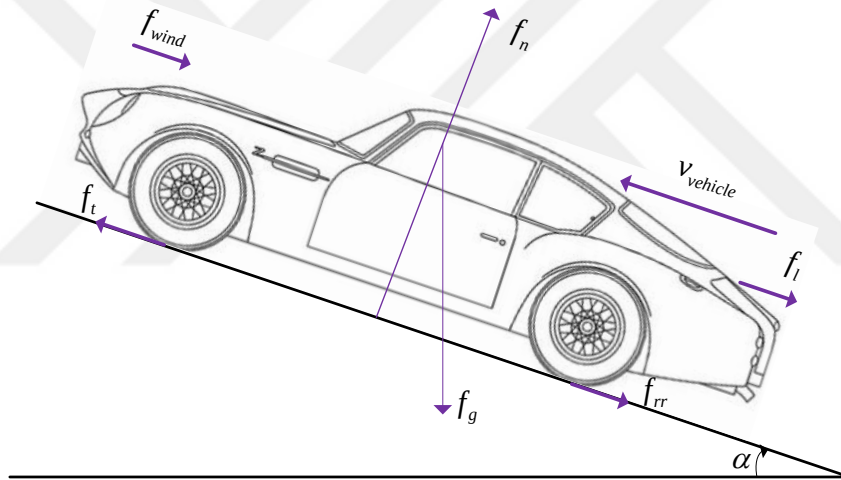


**Figure 3.9.** The forces affecting the movement of the vehicle.

Knowing the physical resistances of all subsystems that affect a vehicle's driving dynamics is vital for vehicle movement. The force required for the movement of the vehicle must exceed the physical resistance of the system. For the vehicle to overcome these forces, the propulsion system must accommodate the necessary force. Therefore, the required driving force for the vehicle can be calculated as follows:

$$F_T = F_l + F_g + F_{rr} + F_{ad} \quad (3.9)$$

where  $F_T$  is the driving force (N) of the vehicle,  $F_l$  is the vehicle inertia force (N),  $F_{rr}$  is the rolling resistance force (N),  $F_g$  is the force of gravity of the vehicle (N), and  $F_{ad}$  is the aerodynamic drag force (N).

The driving force is between the driven wheels and the road surface that is produced by the motor torque to the transmission, final drive to drive wheels. The resistance consist of rolling resistance, vehicle inertia, the gravity of the vehicle, and aerodynamic force, which can be shown in Figure 3.9. These forces are expressed by the following equations:

$$F_l = m_v \dot{v}_v \quad (3.10)$$

$$F_g = m_v g \sin(\alpha) \quad (3.11)$$

$$F_{rr} = m_v g C_{st} \cos(\alpha) \quad (3.12)$$

$$F_{ad} = \frac{1}{2} p_{air} C_d A_{ya} (v_v)^2 \quad (3.13)$$

where  $v_v$  is the speed of the vehicle (km/h),  $\dot{v}_v$  is the acceleration of the vehicle (km/h<sup>2</sup>),  $\alpha$  is the road slope (degrees),  $m_v$  is the mass of the vehicle (kg),  $g$  is the gravitational acceleration,  $p_{air}$  is the air density,  $C_d$  is the aerodynamic coefficient,  $C_{st}$  is the rolling resistance coefficient,  $A_{ya}$  is the surface area of the vehicle.

The torque transmitted to the wheels by the vehicle's mechanical systems can be calculated as shown in the equation(3.14). In addition to the torque equation, the equation of the driving power of the vehicle, the wheel torque, and the angular speed of the wheel are shown in the equation(3.15), (3.16), and(3.17).

$$\tau_T = F_T r_w \quad (3.14)$$

$$p_T = F_T v_v \quad (3.15)$$

$$\tau_w = \frac{\tau_T}{2} \quad (3.16)$$

$$\omega_w = \frac{v_v}{r_w} \quad (3.17)$$

where  $F_T$  the driving force of the vehicle (N),  $\tau_T$  is the torque (Nm),  $\tau_w$  is the wheel moment (Nm),  $\omega_w$  is the angular speed of the wheel (rad/s),  $r_w$  is the wheel radius (m), and  $p_T$  is the drive power (Watt).

Finally, if we apply the Newton's second law to the vehicle, the new equation will be constructed as shown in the following equation:

$$\sum F = (m_v + m(t) + J) \ddot{x}_v \quad (3.18)$$

The total  $F$  is shown in the equation (3.18). Therefore, we can find the vehicle mathematical model as follows:

$$F_T - F_g - F_{rr} - F_{ad} = (m_v + m(t) + J) \ddot{x}_v \quad (3.19)$$

The each forces on the vehicle is also found in the equation, (3.11), (3.12) and (3.13). Thus, the equation (3.20) will be reconstructed as follows:

$$F_T - m_v g \sin(\alpha) - m_v g C_{st} \cos(\alpha) - \frac{1}{2} p_{air} C_d A_{ya} (v_v)^2 - (b_a + b_r) \dot{x}_v - k_b x_v = (m_v + m(t) + J) \ddot{x}_v \quad (3.20)$$

The road slope, the rolling resistance, and aerodynamic coefficients are zero for the simulation. So, the linear equation (3.21) will be as below:

$$\underbrace{(m_v + m(t) + J)}_{a_2} \ddot{x}_v + \underbrace{(b_a + b_r)}_{a_1} \dot{x}_v + \underbrace{k_b}_{a_0} x_v = \tau \quad (3.21)$$

where  $m_v$  is the constant mass,  $m(t)$  is the maximum mass of the person,  $J$  is the inertia,  $b_a$  is the air friction,  $b_r$  is the road friction and  $k_b$  is the braking coefficient.

**Table 3.2.** The parameters of the vehicle.

| Symbols | Values                                      |
|---------|---------------------------------------------|
| $m_v$   | 1000 kg                                     |
| $J$     | 12 kg                                       |
| $m(t)$  | 80 kg an average per person, maximum 400 kg |
| $b_a$   | 18 kg.m/s <sup>2</sup>                      |
| $b_r$   | 12 kg.m/s <sup>2</sup>                      |
| $k_b$   | 22 kg.m/s                                   |

In addition to the equation(3.21), we can consider the mass of the passengers in the vehicle's modelling. According to

Table 3.2, the parameters of the vehicle is substituted in the equation(3.21). If we analyse the equation of the motion(3.21) with a various number of passangers, the roots of the characteristic equation will be as follows:

- a. With full passengers  
Roths:  $-0.0106 \pm j0.1244$   
Time Constant: 94.33
- b. With one passenger  
Roths:  $-0.0138 \pm j0.1416$   
Time Constant: 72.46
- c. Without braking forc with full passengers  
Roths: 0 and  $-0.0212$   
Time Constant: 47.16
- d. Without braking forc with one passenger  
Roths: 0 and  $-0.0276$   
Time Constant: 36

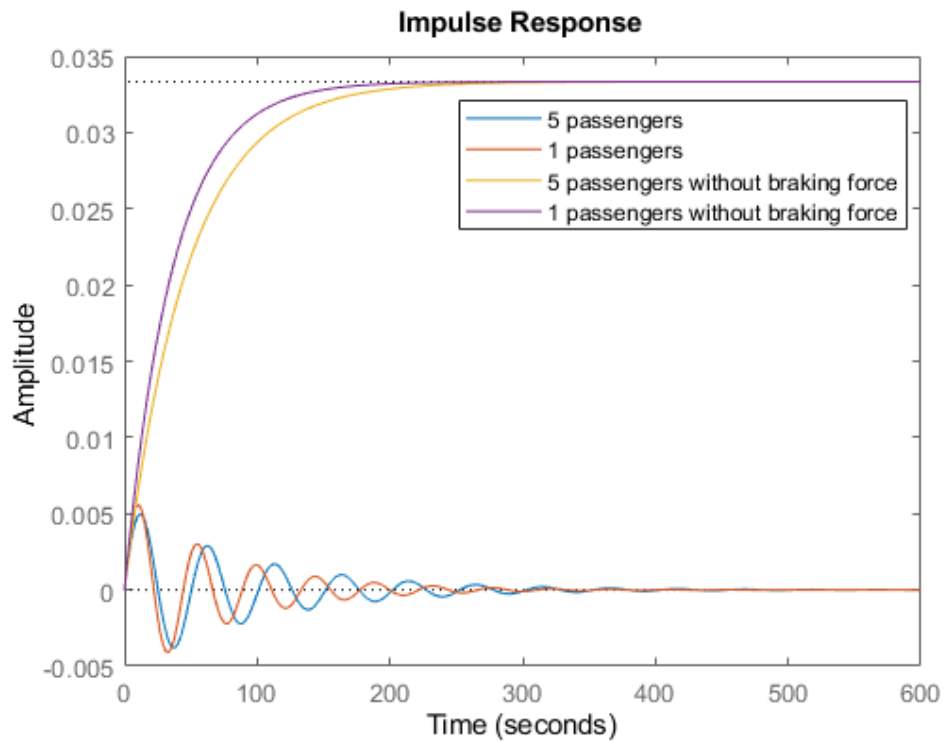


**Figure 3.10.** The impulse response of the linear systems.

As can be seen from the analysis and Figure 3.10, the braking force causes oscillation as it opposes the forward position change of the car. Also, the number of passengers affects the magnitude of the imaginary part since it acts as a damper. However, the number of people increases the settling time of the car as it reduces the natural frequency.

### 3.3.2. The Linear Model of the Vehicle with DC Servo Motor

The DC motor is a typical actuator in control systems. It offers direct rotational motion and, when used in conjunction with wheels or drums and cables, it may also produce translational motion. The electric equivalent circuit of the armature and the free-body diagram of the rotor are shown in the following figure.

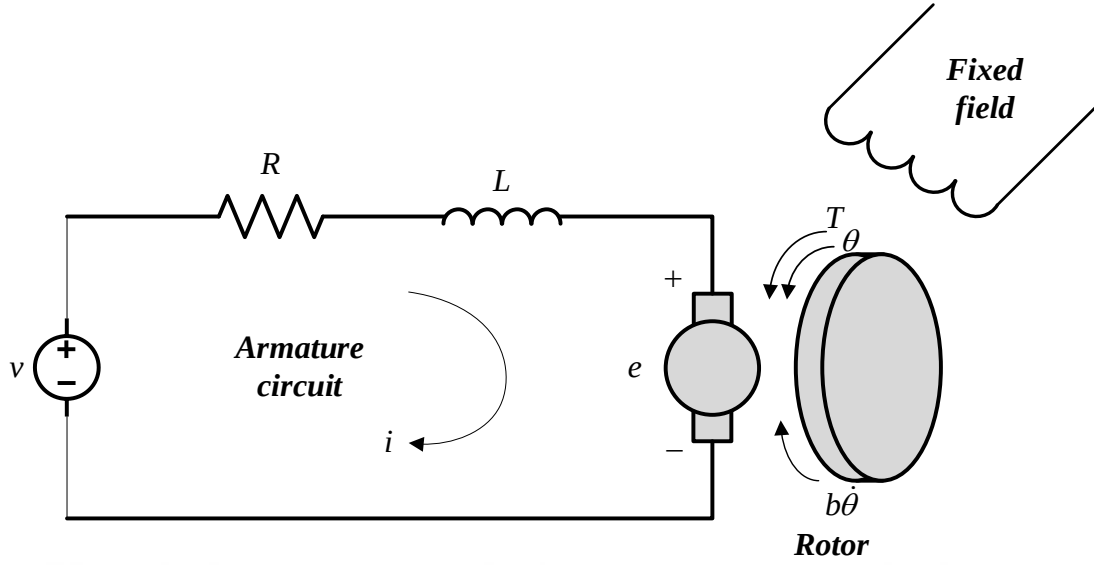


**Figure 3.11.** DC servo motor model.

For the sake of this example, we will suppose that the system's input is a voltage source ( $v$ ) delivered to the motor's armature and that the system's output is the rotational speed of the motor shaft ( $\dot{\theta}$ ). It is believed that the rotor and shaft are both rigid. In addition, we assume a viscous friction model, which means that the friction torque is proportional to the angular velocity of the shaft. There is a constant magnetic field in this case so the motor torque is directly proportional to the armature voltage, as demonstrated in this equation.

The DC motor torque is generally proportional to the current flowing through the armature as well as the intensity of the magnetic field created by it. There is a constant magnetic field in this case so the motor torque is directly proportional to the armature current  $i$  by a constant factor  $K_t$ , as demonstrated in this equation. An armature-controlled motor is what this is formally referred to as.

$$\tau = K_t i \quad (3.22)$$

As the shaft's angular velocity increases, the back emf  $e$  also increases by a constant factor.

$$e = K_e \dot{\theta} \quad (3.23)$$

Based on Kirchhoff's voltage law, we may obtain the following governing equations from the figure above.

$$L \frac{di}{dt} + Ri + K_e \dot{\theta} = u \quad (3.24)$$

If we consider the equation (3.21) and (3.22), then

$$a_2 \ddot{x} + a_1 \dot{x} + a_0 x = K_t i \quad (3.25)$$

Devided by  $K_t$  each sides of the equation (3.25), then

$$\frac{1}{K_t} (a_2 \ddot{x} + a_1 \dot{x} + a_0 x) = i \quad (3.26)$$

If we substitute the equation (3.26) into the equation(3.24), then

$$\frac{R}{K_t} (a_2 \ddot{x} + a_1 \dot{x} + a_0 x) + \frac{L}{K_t} (a_2 \ddot{x} + a_1 \dot{x} + a_0 \dot{x}) + K_e \dot{x} = u \quad (3.27)$$

If we rearrange the equation(3.27), then the final equation of the system is:

$$\underbrace{\frac{L}{K_t} a_2}_{b_3} \ddot{x} + \underbrace{\left( \frac{R}{K_t} a_2 + \frac{L}{K_t} a_1 \right)}_{b_2} \ddot{x} + \underbrace{\left( \frac{R}{K_t} a_1 + \frac{L}{K_t} a_0 + K_e \right)}_{b_1} \dot{x} + \underbrace{\frac{R}{K_t} a_0}_{b_0} x = u \quad (3.28)$$

**Table 3.3.**The parameters of the DC servo motor.

| Symbols | Definition                    | Values |
|---------|-------------------------------|--------|
| $K_e$   | The opposite EMF parameter    | 100    |
| $K_t$   | The produced torque parameter | 120    |
| $R$     | Armature resistance           | 200    |
| $L$     | Armature inductance           | 20     |

According to Table 3.3, the parameters of the vehicle are substituted in the equation (3.28). The roots of the characteristic equation will be as follows if we analyze the equation of motion (3.28) with a variable number of passengers:

$$\begin{aligned} & -5 + j0 \\ & -0.0106 \mp j0.1244 \end{aligned} \quad (3.29)$$

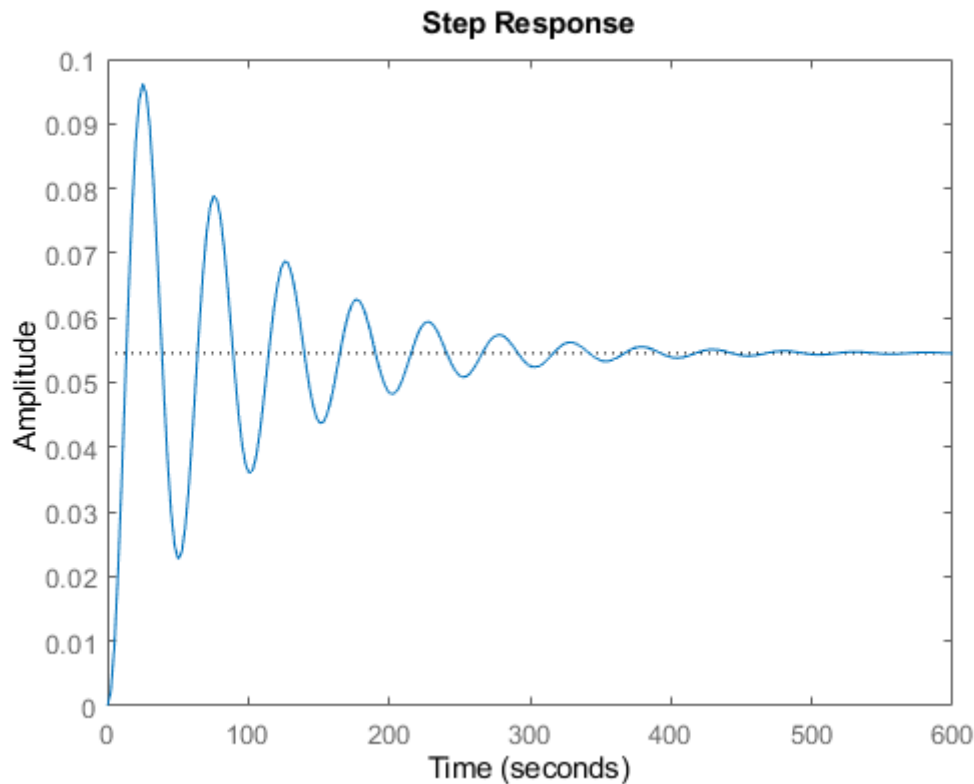


**Figure 3.12.** The step response of the linear system with DC motor.

As can be seen from the analysis and Figure 3.12, the dominant time constant of the system has not changed even though the system is now third order due to the added actuator. The selected actuator has added an insignificant real pole to the system.

### 3.3.3. The Nonlinear Model of the Vehicle with DC Motor

The nonlinear dynamic model of the automobile is quite intricate and requires a high degree of manoeuvrability. It is also important to note that the system is nonlinear and discontinuous. A linear vehicle model contains just the longitudinal dynamics of the vehicle and does not incorporate any other aspects of the vehicle. Consequently, Newton's Second Law may be used to easily deduce the mathematical model of the vehicle that has been

created. The nonlinear mathematical model of the vehicle, and the linear mathematical model of the vehicle are both described in this section. The derived model of the vehicle in the previous subsection, which is shown in the equation (3.28) becomes for the sloped case as follows:

$$b_3\ddot{x} + b_2\dot{x} + b_1x = u - mg \sin(\theta) \quad (3.30)$$

As can be seen from the analysis, the system is nonlinear, which causes direct torque reduction. However, the character of the system is the same as on the ground since the system is same system, but only the external force has changed. For the forced case, the nonlinear dynamics are injected due to slope.

### 3.3.4. The Motion Analysis in the Dynamical Environments

There are various uncertainties that the AVs may encounter in a dynamic environment, some of which are puddles and bumps. In addition, the angular velocity generated by AVs as the vehicle turns around the intersection is also an important dynamic factor.

#### 3.3.4.1. The Puddles on the Road

If we consider the puddles, we modify and analyse the AVs modelled in section 3.3.2 when it interacts with puddles having a height of 10 cm and a radius of 30 cm. Assume that the car has no initial velocity and no external forces.

During the falling without initial velocity, the equation will be as follows:

$$d = V_i t + \frac{1}{2} a t^2 \quad (3.31)$$

where  $d$  is the distance (height) and  $V_i$  is the initial velocity.

Therefore, the time is shown in the following equation:

$$10 = \frac{1}{2} * 9.81 * t^2 \quad (3.32)$$

$$t = 1.42 \text{ sn}$$

So the settling time or the steady state time is 1.42 sn and the negative force is mg. Thus, we can accept that the 5 time constants is 1.42 sn and one time constant is  $\tau = 0.28$ .

Hence the force of the system is shown in the following equation:

$$F_p = -mg(1 - e^{-t/\tau}) \quad (3.33)$$

Its behaviour is analysed through the Matlab and it is shown in the Figure 3.13.

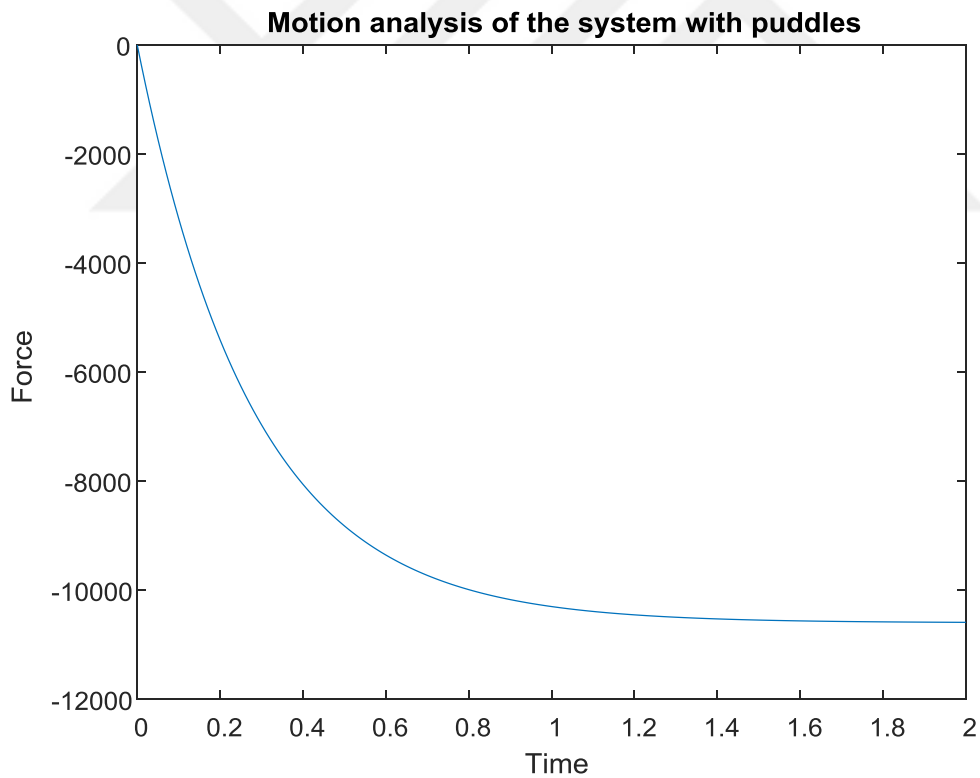


**Figure 3.13.** The response of the system against to the puddles.

If there are no initial conditions for the car, such as initial position and velocity, the only motion source will be the negative. A kind of impulse force will occur that is shown in Figure

3.13. This negative impulse force is in terms of the car's mass. Therefore, it represents the behaviour or response of the system.

### 3.3.4.2. The Bump on the Road

If we consider the bump, we modify and analyse the AVs modelled in section 3.3.2 when it interacts with a bump having a height of 15 cm and a radius of 20 cm. Assume that the car has no initial velocity and no external forces. During the climbing without initial velocity, the equation will be as follows:

$$\begin{aligned} d &= V_i t + \frac{1}{2} a t^2 \\ 20 &= \frac{1}{2} * 9.81 * t^2 \\ t &= 2.019 \text{ sn} \end{aligned} \tag{3.34}$$

This is the settling time which can be accepted as the five time constants. Therefore, the one time constant is  $\tau = 0.4039$ . In this case the force is:

$$F_b = mg(1 - e^{-t/\tau}) \tag{3.35}$$

Its behaviour is analysed through the Matlab and it is shown in the Figure 3.14.

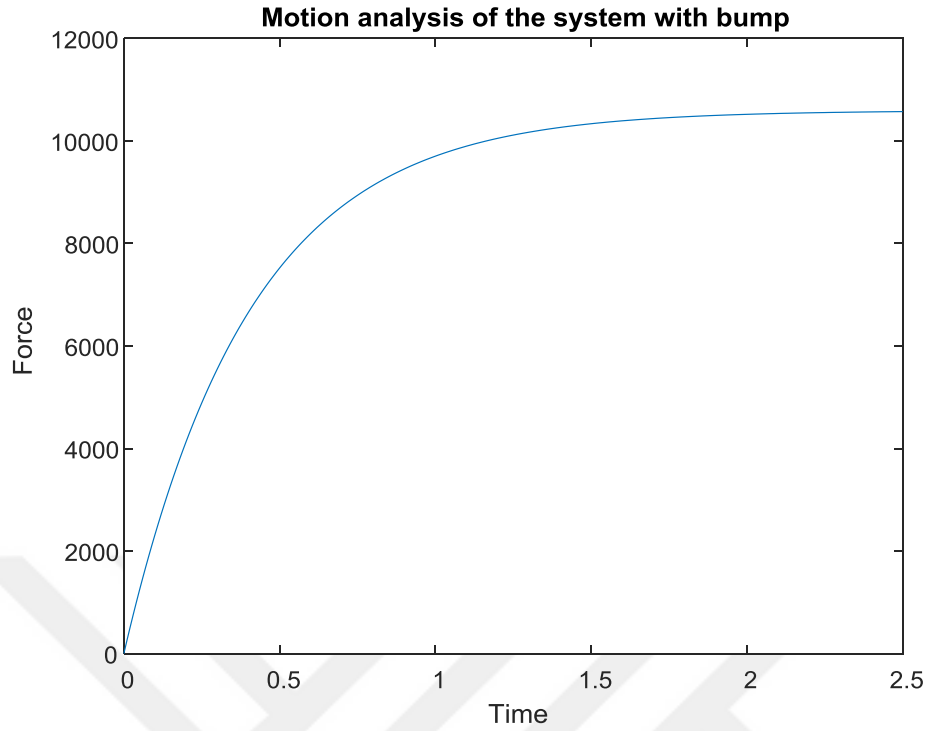


**Figure 3.14.** The response of the system against to the bump.

If there are no initial conditions for the car, such as initial position and velocity, the only motion source will be the positive. A kind of impulse force will occur that is shown in Figure 3.14. This positive impulse force is in terms of the car's mass. Therefore, it represents the behaviour or response of the system.

#### 3.3.4.3. The Intersection on the Road

If we consider the intersection on the road and determine the maximum angular velocity of the AVs, the following equations can be used to find the maximum angular velocity. In addition, the DC servo motor saturates at 40.000 N/m. If we remember the torque produced by the motor and armature current in the equations (3.22) and (3.24), the relationship between the torque and force can be defined as the following equation:

$$\begin{aligned}\tau &= r \times F \\ &= \|r\| \|F\| \sin \theta \quad \theta \rightarrow 90^\circ\end{aligned}\tag{3.36}$$

Thus,

$$\tau = 40000 = 10F \Rightarrow F = 4000N \quad (3.37)$$

Centrifugal force occurring on a circular motion of a mass is;

$$\begin{aligned} F &= mw^2r \\ w &= \sqrt{\frac{F}{m.r}} \\ &= 0.6114 \text{ rad / s} \end{aligned} \quad (3.38)$$

This is the maximum radial frequency that the car can have during a turn in a roundabout with a radius of 10 m. Other physical properties such as inertia and friction should be re-considered here as the motion is radial rather than translational.

### 3.3.5. Determining the Controller Regions for General Motion of the AVs

AVs controllers are constructed in this chapter, utilizing linear and nonlinear vehicle models generated in the previous chapters. When developing controllers, the most crucial priority to consider is determining the stability region of the model under consideration. A closed loop system with a PD (Proportional + Derivative) controller for the derived models is created in order to achieve this range of performance.

#### 3.3.5.1. The Stability Region for the Linear Model by Using PD Controller

There are two conditions in the controller design: the first one is the stability will be kept, which is intrinsically possible since the system is already stable, and the second one is that the overall closed loop system gain will be positive for the forward motion control.

Consider the equation (3.39) that is:

$$b_3\ddot{x} + b_2\dot{x} + b_1\dot{x} + b_0x = u \quad (3.39)$$

The PD controller is:

$$C(s) = k_p + k_d s \quad (3.40)$$

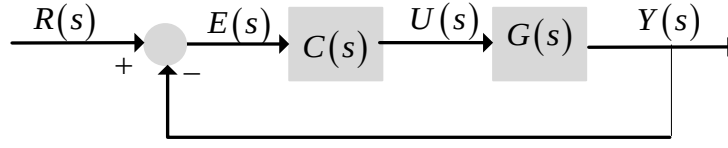


**Figure 3.15.** The block diagram of the closed-loop system.

The closed-loop system equation that is derived in Figure 3.15 is as follows:

$$\begin{aligned}
 G_{cl}(s) &= \frac{Y(s)}{U(s)} = \frac{G(s)C(s)}{1+G(s)C(s)} \\
 &= \frac{k_p + k_d s}{b_3 s^3 + b_2 s^2 + (b_1 + k_d)s + (b_0 + k_p)}
 \end{aligned} \tag{3.41}$$

The stability region of the system that is shown in the equation (3.41) is shown in the following figure. This region has been constructed by using the Matlab.

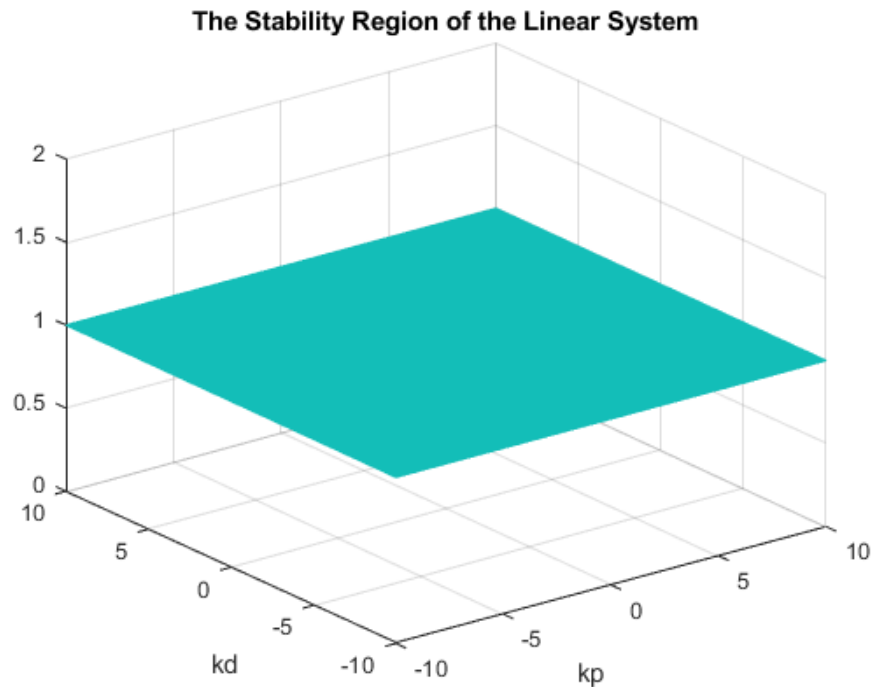


**Figure 3.16.** The stability region of the linear system.

In order to observe the step response of the closed-loop system, the closed-loop transfer function was derived by using the  $k_p$  and  $k_d$  values ( $k_p = 2.2$  and  $k_d = 0.5$ ) selected from the region obtained in Figure 3.16. Using the step input and applying it to this generated transfer function, the response shown in the following has been obtained.

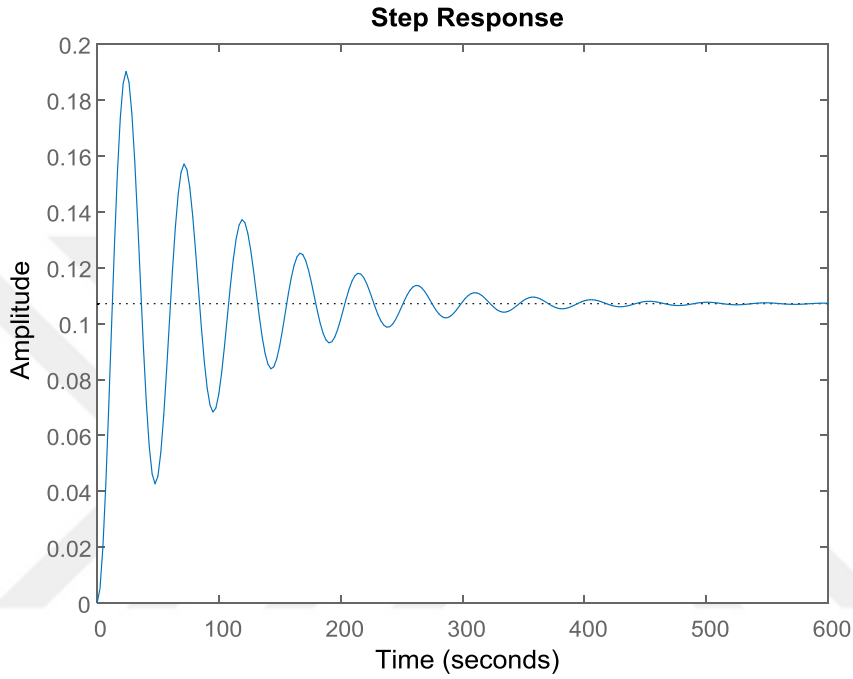


**Figure 3.17.** Step response of the closed-loop system.

As can be seen from the stability region in Figure 3.17, the selected control parameters region satisfies the conditions for the whole region. This is due to the dominant parameters of the car. This means the control parameters do not dominate the system's response. However, larger control parameters lead to much more aggressive controllers which basically suppress the dynamics of the car.

### 3.3.5.2. The Stability Region for the Nonlinear Model by Using PD Controller

When constructing the stability region for the nonlinear model, a closed-loop transfer function is obtained, similar to the procedure used to construct the stability region for the linear model. In addition, the PD controller is used to generate this transfer function. For the closed-loop system, the nonlinear model described in the equation(3.42) is utilized. The  $\theta$  value in this equation is determined as 0.15.

$$b_3\ddot{x} + b_2\dot{x} + b_1\dot{x} + b_0x = u - mg \sin \theta \quad (3.42)$$

The closed-loop transfer function obtained using the PD controller is shown in the equation below.

$$X(s) = \underbrace{\frac{k_p + k_d s}{b_3 s^3 + b_2 s^2 + (b_1 + k_d)s + (b_0 + k_p)}}_{\text{step}} R - \underbrace{\frac{mg \sin \theta}{b_3 s^3 + b_2 s^2 + (b_1 + k_d)s + (b_0 + k_p)}}_{\text{impulse}} \quad (3.43)$$

The stability region of the system that is shown in the equation (3.43) is shown in the following figure. This region has been constructed by using the Matlab.

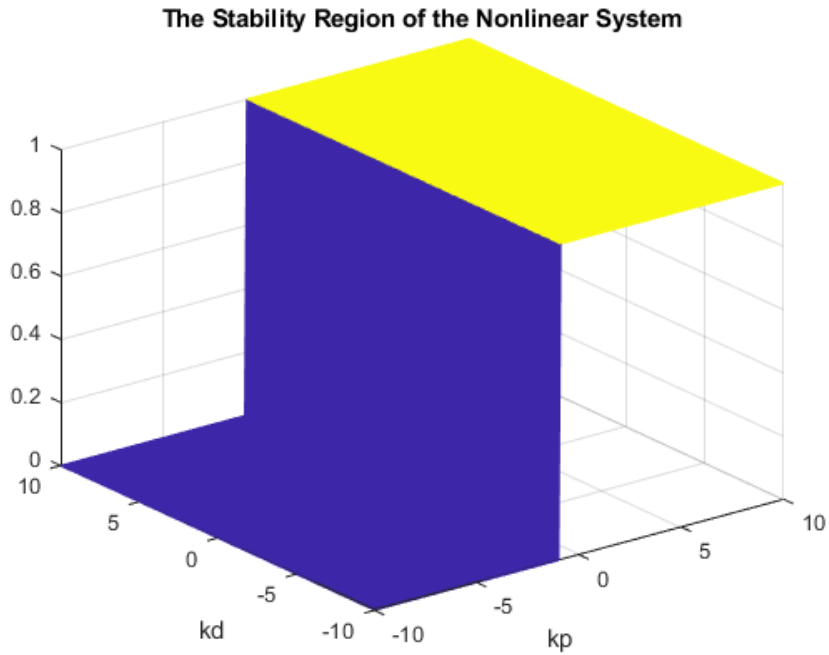


**Figure 3.18.** The stability region of the nonlinear system.

In order to observe the step response of the closed-loop system, the closed-loop transfer function was derived by using the  $k_p$  and  $k_d$  values ( $k_p = 4.2$  and  $k_d = 0.2$ ) selected from the region obtained in Figure 3.18. Using the step input and applying it to this generated transfer function, the response shown in the following has been obtained.

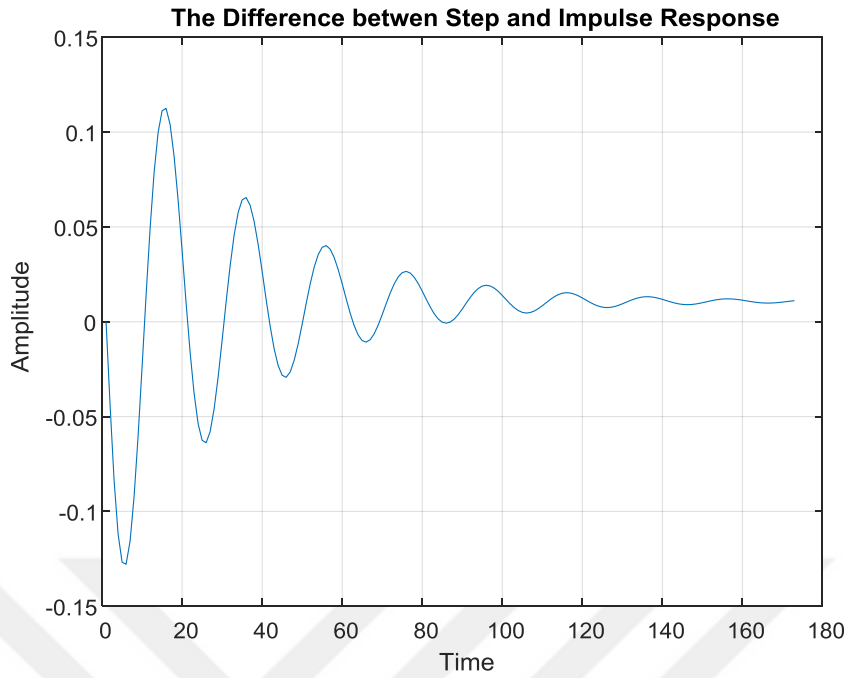


**Figure 3.19.** The difference between step and impulse response of the closed-loop system.

As can be seen from the results in Figure 3.19, a limited number of control parameters can yield stable forward motion. For departure control, the controller requires a larger region for freedom.

### 3.3.6. Performance Based Controller Design for Autonomous Motion and Departing

The goal of this part is to develop a Proportional and Derivative(PD) controller for the linear vehicle model that was previously acquired. It used a performance-based design while developing the PD controller. To put it another way, while creating the controller, a number of parameters are defined. For example, settling time, rising time, and maximum overshoot are all factors that are considered while building a controller for a certain application. When building the PD controller in this part, the maximum overshoot (15 percent) and settling time (200 seconds) are taken into consideration.

The damping ratio for 15% overshoot is shown in the following equation:

$$e^{-\frac{\zeta T}{\sqrt{1-\zeta^2}}} = 0.15 \Rightarrow \zeta = 0.51 \quad (3.44)$$

The natural frequency for 200 second settling time is:

$$t_s = \frac{4.6}{\zeta w_n} = 200 \Rightarrow w_n = 0.0445 \quad (3.45)$$

So that the standard second-order system with complex conjugate poles and without zeros is:

$$G_d(s) = \frac{w_n^2}{s^2 + 2\zeta w_n s + w_n^2} = \frac{0.002}{s^2 + 0.0454s + 0.002} \quad (3.46)$$

The closed-loop system with PD controller is:

$$G_{cl}(s) = \frac{(k_d s + k_p) / b_3}{s^3 + (b_2 / b_3) s^2 + ((b_1 + k_d) / b_3) s + (b_0 + k_p) / b_3} \quad (3.47)$$

To choose PD parameters, an extra pole that is significantly larger than the real part of the complex conjugate poles must be added to the second-order system. Therefore, the target system is:

$$\begin{aligned} G_d(s) &= \frac{0.002}{s^2 + 0.0454s + 0.002} \frac{1}{(s + 10)} \\ &= \frac{0.002}{s^3 + 10s^2 + 0.46s + 0.0198} \end{aligned} \quad (3.48)$$

Hence, the PD parameters are  $k_d = 79.58$  and  $k_p = -13.637$ .

Finally, the parameters of the PD controller acquired for the closed-loop were written in their proper locations, and a system analysis was carried out on the results. Figure 3.20 depicts the step response of the closed-loop transfer function that was calculated for system analysis and is used in the simulation.

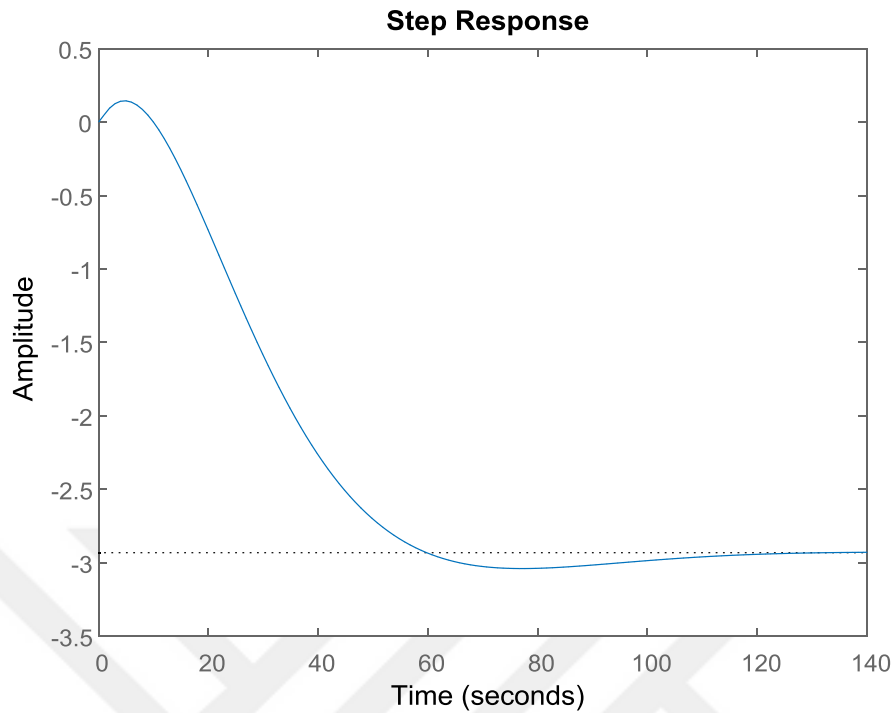


**Figure 3.20.** Step response of the closed-loop system.

If we analyse the closed-loop transfer that is shown in the equation(3.48), the roots of the characteristic equation will be as follows:

$$\begin{aligned} & -9.9540 + 0.0000j \\ & -0.0230 \mp 0.0382j \end{aligned} \quad (3.49)$$

It is important to note that the desired system does not have a zero, whereas the closed loop system has a zero at 0.1718 and this zero changes the system response significantly. Specifically, it changes the gain of the system, so that the overall response of the system is altered. This is again due to dominant system dynamics. To solve this problem, the size of the added third pole to the desired system must be increased, and the structure of the controller must be changed to control the more parameters of the system in order to gain freedom or flexibility.

### 3.4. The EVs Modelling in the Simulink

This subsection first performs the architecture of the vehicle modelling. In addition, other subsections of this chapter perform the construction of the EV simulation and identify each

simulation block, such as driving cycle, driver model, the battery of the vehicle, the motor of the vehicle, vehicle dynamics, and simulation results, which are shown in Figure 3.22.

### 3.4.1. The architecture of the Vehicle Modelling

Many researchers EVs to be the automobiles of the future since they are very efficient, do not generate local pollution, are silent, and may be utilized by the grid operator for the control of electricity. However, there are still major concerns with electric vehicles that must be addressed. The three primary difficulties are the restricted driving range, the lengthy charging period, and the exorbitant cost. The three primary tasks pertain to the car's battery pack. Batteries should have sufficient energy to have a specific driving range and should also have adequate power to accelerate and decelerate. To predict the energy consumption of EVs, a good vehicle model is highly necessary. The EV model is quite complex because it includes several components such as transmission, electric machine, power electronics, and battery. To achieve accurate conclusions, each component must be modelled correctly.

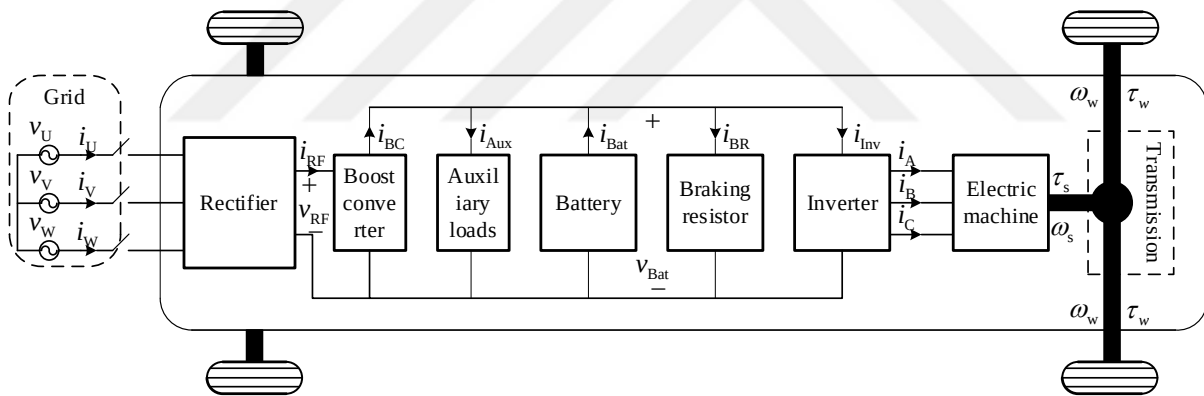


**Figure 3.21.** The architecture of the EVs.

There are many alternative EV designs as there are many possible options. For instance, 1 to 4 machines, DC or Alternative Current (AC), gearbox/subject gearboxes, high or low battery voltage, charging in one of three stages, etc. The following is a brief explanation of the functions of the various components in Figure 3.21: the three-phase electric machine provides traction power to the wheels. A differential with a gear ratio also allows the torque of the left and right wheels to adapt to the high speed of the machine shaft to the lower speed of the wheel. The inverter, which converts the battery voltage into a three-phase AC voltage that is suited for the electric machine, controls the torque and machine speed. When assessing the

electricity consumption of EVs, the losses from the components that are not part of the power chain from the grid to the wheels must also be included. The losses include the lighting, comfort and safety systems. These are sometimes referred to as auxiliary losses. The rectifier rectifies three-stage voltages, grid currents to DC levels, and the boost converter allows power transmission to the battery's high voltage side from the low voltage side of the rectifier.

### 3.4.2. Construction of the EVs Simulation

Vehicle driving tests and simulations are used to assist during the design process of the EVs and to identify whether the design is suitable for the application. The acquisition of the outputs which are used as the inputs of the algorithm, such as speed, position, SoC battery voltage, current, power, and the simulation part of the EV, is constructed in this section.

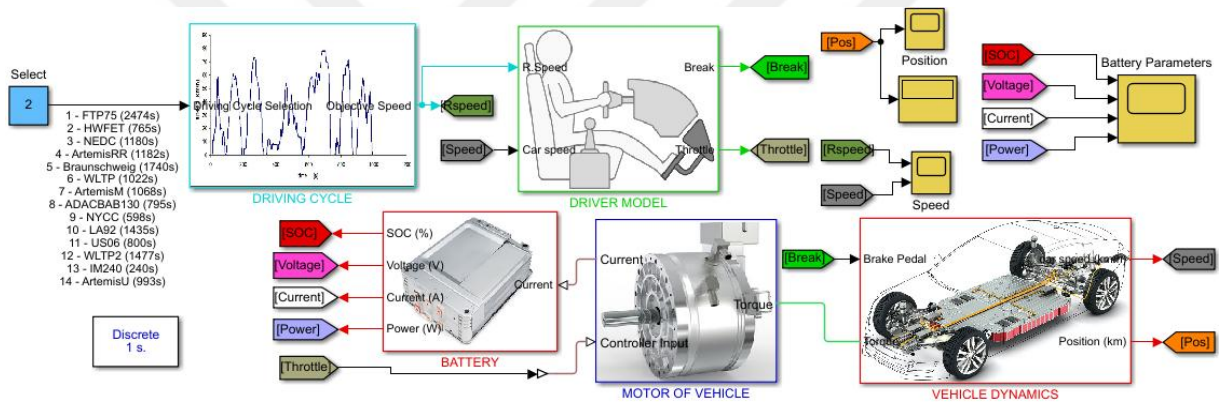


**Figure 3.22.** The simulation of the EVs in the Simulink.

The simulation of EV is shown above, and the following sub-sections are portrayed for each block in the MATLAB/Simulink simulation.

#### 3.4.2.1. Driving Cycle

The driving cycle plays a crucial role in the simulation of EV due to the necessity of the input, which is a second-by-second sequence of vehicle velocity values. The reasons for the requirement of the driving cycle during simulation include reducing the number of expensive road tests, reducing test time, and test engineer's fatigue. The driving cycles are incorporated to model the drive system and estimate its performance in the vehicle simulations. The drive cycle provides a longitudinal drive cycle that may be either standard or customized by the user. The output of the block is the given vehicle longitudinal speed, which may be used to perform the following operations:

- Using a gear change reference, calculate the amount of torque and fuel consumption a vehicle needs to reach the specified speed and acceleration in the given gear.
- Closed-loop accelerating and braking orders for vehicle control and plant models may be generated using accurate velocity and shift references.
- Over a period of time, analyze, adjust, and improve vehicle control, system performance, and system resilience.

MATLAB/Simulink has several standard driving cycles used for testing vehicles for fuel economy, driving range estimation, and other purposes. In our simulation, the driving cycle sub-system is constructed with 10 different driving paths that are shown in Figure 3.23.

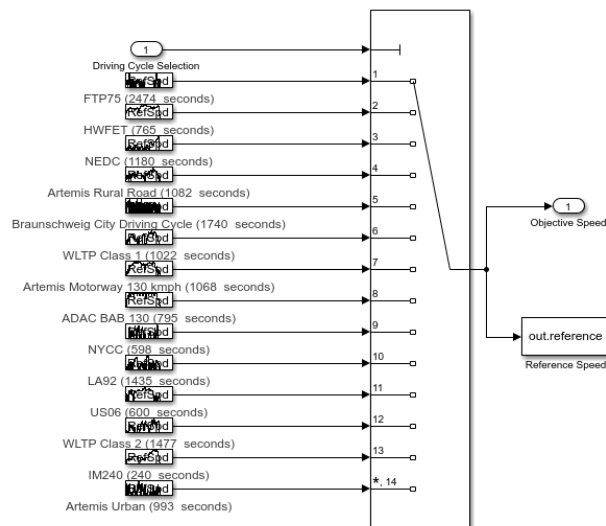


**Figure 3.23.** Inside of the driving cycle sub-system.

The input of the sub-system is selected as the driving path and the output of the sub-system is sent at an objective speed that is chosen in the input. The inside of the driving cycle sub-system is shown in the figure above.

#### 3.4.2.2. Driver Model

It aims to simulate the roles of the driver and the vehicle with this sub-system. Taking into account the objective speed, which is the output of the driving cycle, and the real-time instantaneous speed of the vehicle, these two speeds are controlled by the Proportioanl and Integral (PI) controller to compose a driver model aiming to develop the appropriate throttle

and brake commands. To allow the vehicle speed to follow the reference speed, the instantaneous speed of the vehicle is connected to a feedback control loop. The inside of the driver model sub-system is shown in the following figure.

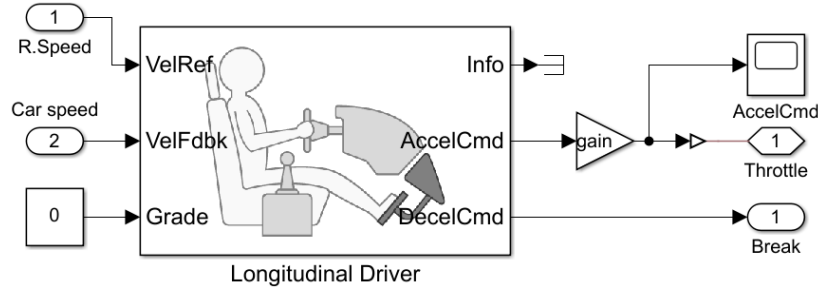


**Figure 3.24.** Inside of the driver model sub-system.

Implementation of a longitudinal speed tracking controller is provided by the Longitudinal Driver block. The block provides normalized accelerating and braking instructions that may range from 0 to 1 based on reference and feedback velocities. Modelling a driver's dynamic reaction may be done using the block, as well as generating the instructions needed to follow a drive cycle. Various configurations can be made in this block. For this simulation, the controller type of the block is set to proportional-integral (PI). The block provides PI control with tracking windup and feed-forward gains when the control type is changed to PI. The output of the speed control is determined by the following equations.

$$y = \frac{K_{ff}}{v_{nom}} v_{ref} + \frac{K_p e_{ref}}{v_{nom}} + \int \left( \frac{K_i e_{ref}}{v_{nom}} + K_{aw} e_{out} \right) dt + K_g \theta \quad (3.50)$$

where;

$$e_{ref} = v_{ref} - v \quad (3.51)$$

$$e_{out} = y_{sat} - y$$

$$y_{sat} = \begin{cases} -1 & y < -1 \\ y & -1 \leq y \leq 1 \\ 1 & 1 < y \end{cases} \quad (3.52)$$

This transfer function is used in the low-pass filter for velocity error that is shown below.

$$H(s) = \frac{1}{\tau_{err}s + 1} \quad \text{for } \tau_{err} > 0 \quad (3.53)$$

The following equations are used by the block to determine the acceleration and braking signals.

$$y_{sat} = \begin{cases} 0 & y_{sat} < 0 \\ y_{sat} & 0 \leq y_{sat} \leq 1 \\ 1 & 1 < y_{sat} \end{cases} \quad (3.54)$$

$$y_{dec} = \begin{cases} 0 & y_{sat} > 0 \\ -y_{sat} & -1 \leq y_{sat} \leq 0 \\ 1 & y_{sat} < -1 \end{cases} \quad (3.55)$$

The variables of the equations that are mentioned above is shown in Table 3.4.

**Table 3.4.** The variables of the equations.

|              |                                                          |
|--------------|----------------------------------------------------------|
| $v_{nom}$    | Nominal vehicle speed                                    |
| $K_p$        | Proportional gain                                        |
| $K_i$        | Integral gain                                            |
| $K_{aw}$     | Anti-windup gain                                         |
| $K_{ff}$     | Velocity feed-forward gain                               |
| $K_g$        | Grade angle feed-forward gain                            |
| $\theta$     | Grade angle                                              |
| $\tau_{err}$ | The error filter time constant                           |
| $y$          | Nominal control output magnitude                         |
| $y_{sat}$    | Saturated control output magnitude                       |
| $e_{ref}$    | Velocity error                                           |
| $e_{out}$    | Difference between saturated and nominal control outputs |
| $y_{acc}$    | Acceleration signal                                      |
| $y_{dec}$    | Braking signal                                           |
| $v$          | Velocity feedback signal                                 |
| $v_{ref}$    | Reference velocity signal                                |

### 3.4.2.3. The Battery of the Vehicle

In section 3.1, the battery of the vehicle has been introduced accurately. The mentioned information about the battery is used to develop this sub-system of the vehicle.

### 3.4.2.4. The Motor of the Vehicle

The Pulse Width Modulation (PWM) generator, H-bridge converter, and DC motor are contained in the motor of the vehicle subsystem. The PWM generator produces the signal to be used in the H-bridge according to the signal from the driver model. The H-bridge determines the voltage required for the DC motor according to the signal it receives from the PWM generator. The DC motor fulfills the required current from the battery with the voltage determined by the H-bridge, according to the objective speed of the vehicle. The torque provided by the electric motor is obtained by the throttle and brake commands, which is the output of the drive model, and it becomes the output of the sub-system as input to the vehicle dynamics sub-system. The inside of the motor of the vehicle sub-system is shown in the following figure.

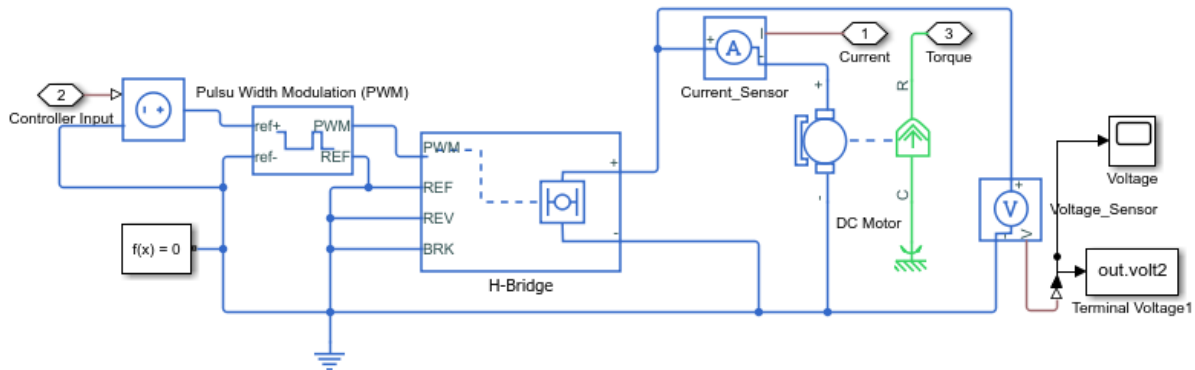


**Figure 3.25.** Inside of the motor sub-system.

#### a. PWM Generator

Controlling power transmission from one electrical component to another is accomplished using the PWM approach, which rapidly switches from full power transfer to none at all. This block's PWM generator outputs either 1 or 0, depending on whether or not the duty cycle is larger than the carrier counter. By setting the timer period  $T_{per}$ , we may control the number of times a cycle repeats. One of three kinds of carrier counters may be used to alter the beginning output or phase of the PWM output:

**Up counter:** At the beginning of each on-cycle, the PWM output signal is activated. The carrier counter signal and the accompanying PWM output are shown in the following figure.

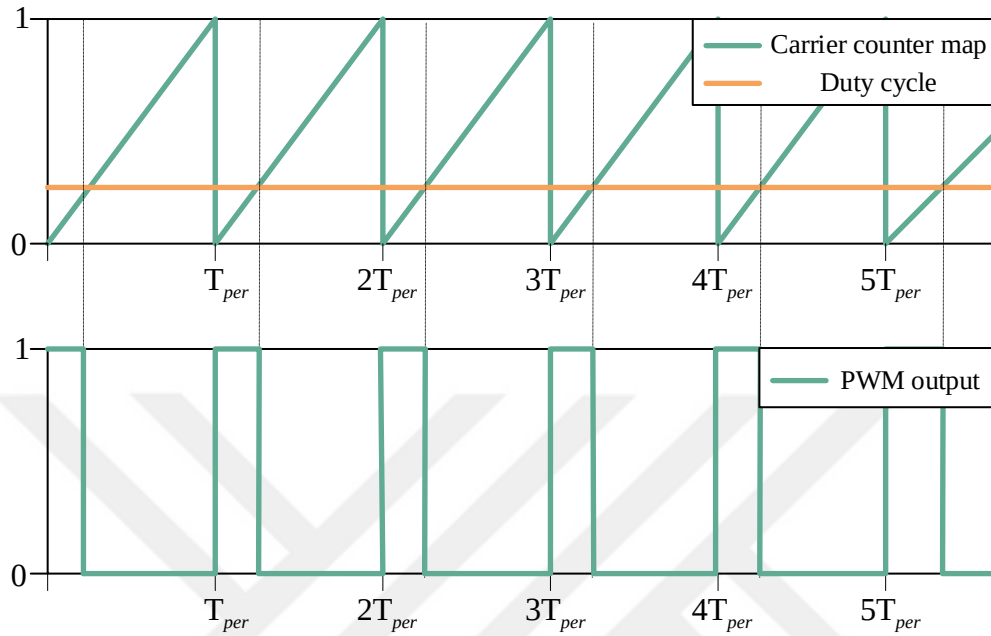


**Figure 3.26.** Up counter PWM.

**Down counter:** At the beginning of the off-cycle, the PWM output signal is initialized. The carrier counter signal and the accompanying PWM output are shown in the following illustration.

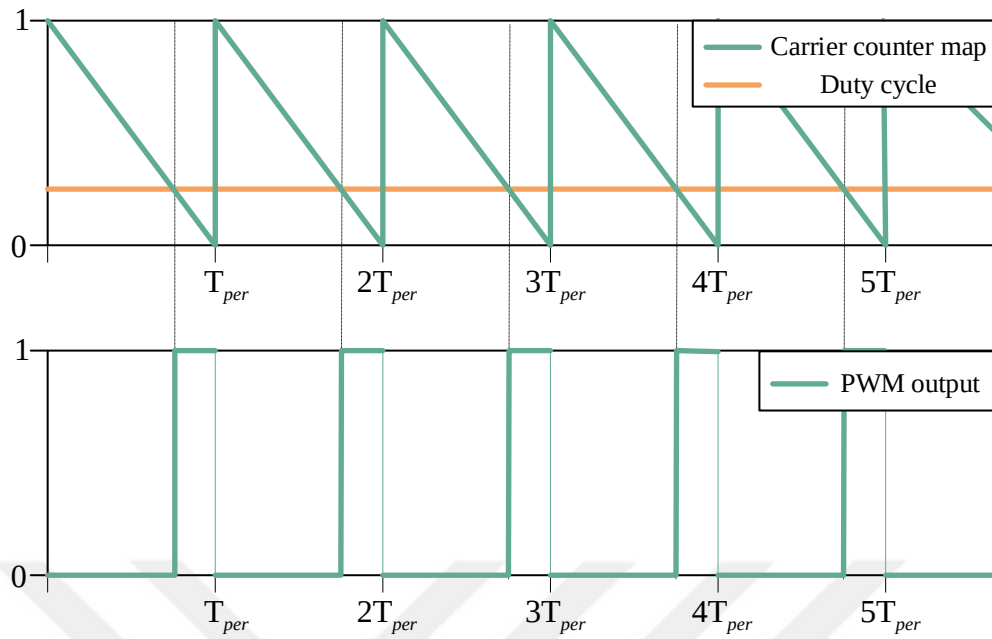


**Figure 3.27.** Down counter PWM.

**Up-down counter:** The initialization of the PWM output signal occurs around halfway through the on-cycle. The carrier counter signal and the accompanying PWM output are seen in the following illustration.

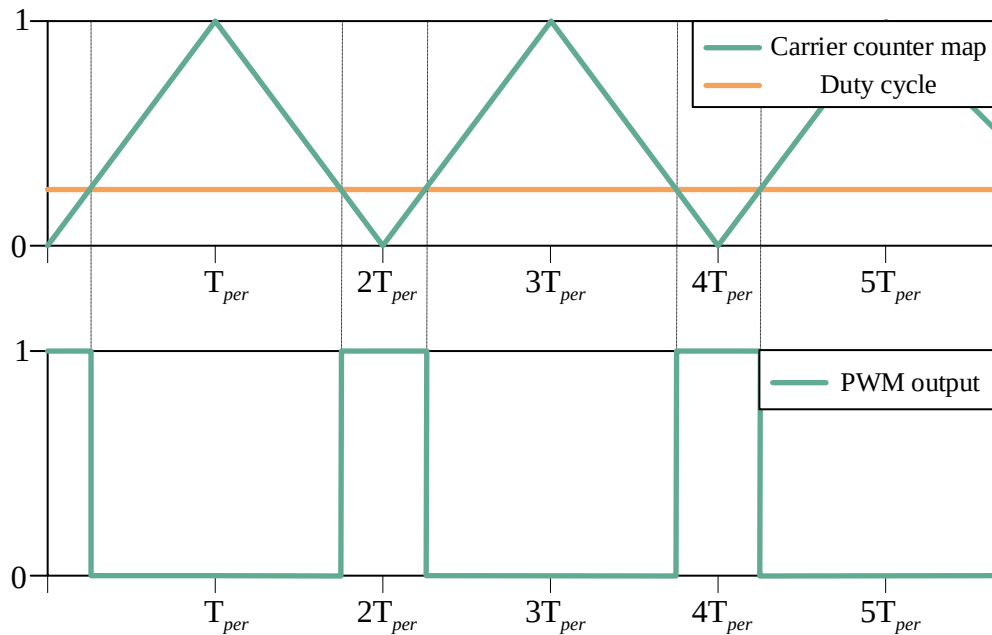


**Figure 3.28.** Up-down counter PWM.

## b. H-bridge

In electric-electronics engineering, an H-bridge is a kind of electronic circuit that changes the direction of the polarity of a voltage supplied to a load. Typically, these circuits are employed in robotics and other applications to enable DC motors to operate in either the forward or reverse direction. When it comes to power electronics, the H-bridge is a common component in many types of DC/AC converters, including power inverters, push-pull converters, and DC/DC converters. Motor controllers with two H-bridges are virtually often used to control bipolar stepper motors. The basic structure of the H-bridge is shown in the following figure.

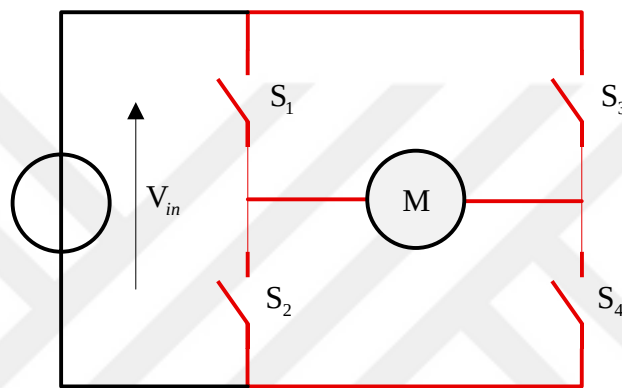


**Figure 3.29.** The structure of the H-bridge.

This circuit is known as the H-bridge because of the traditional graphical depiction as shown in Figure 3.29 (highlighted in red). Four switches are used to build an H-bridge (solid-state or mechanical). As shown in Figure 3.30 (a), when the switches  $S_1$  and  $S_4$  are closed, a positive voltage is provided across the motor while the switches  $S_2$  and  $S_3$  are open. This voltage may be reversed by opening the  $S_1$  and  $S_4$  switches and shutting the  $S_2$  and  $S_3$  switches, as illustrated in Figure 3.30(b), enabling the motor to operate in the opposite direction. A short circuit would occur if the switches  $S_1$  and  $S_2$  were closed at the same moment, since this would result in a loss of input voltage from the power supply.  $S_3$  and  $S_4$  switches are also subject to the same restrictions. Shoot-through is the term used to describe this phenomenon.

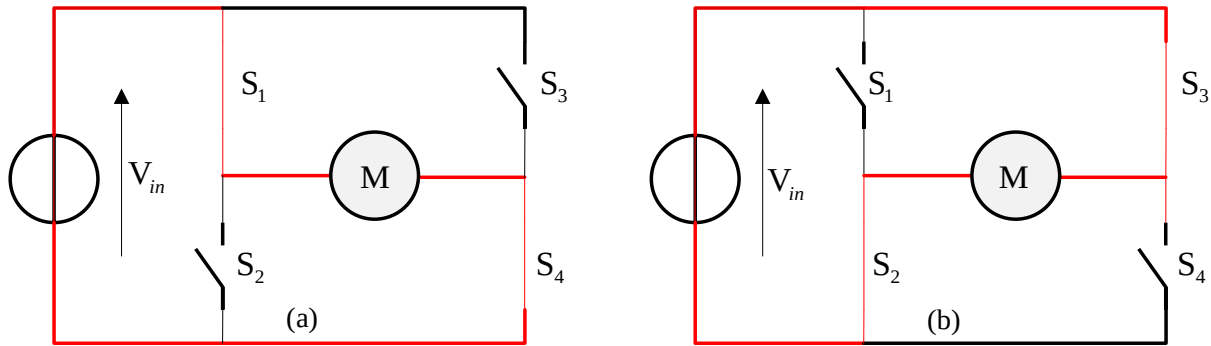


**Figure 3.30.** The two basic states of the H-bridge.

The H-bridge is commonly used to inverse the polarity or direction of a motor, but it can also be used to "brake" the motor, in which case the motor comes to a sudden stop as the motor's terminals are shorted, or to let the motor "free run" to a stop as the motor is effectively disconnected from the circuit. The operation is summarized in the following table, with the numbers S1-S4 matching the figure above.

**Table 3.5.** The operation of the MOSFETS.

| S <sub>1</sub> | S <sub>2</sub> | S <sub>3</sub> | S <sub>4</sub> | Results           |
|----------------|----------------|----------------|----------------|-------------------|
| 1              | 0              | 0              | 1              | Motor moves right |
| 0              | 1              | 1              | 0              | Motor moves left  |
| 0              | 0              | 0              | 0              | Motor coasts      |
| 1              | 0              | 0              | 0              |                   |
| 0              | 1              | 0              | 0              |                   |
| 0              | 0              | 1              | 0              |                   |
| 0              | 0              | 0              | 1              | Motor brakes      |
| 0              | 1              | 0              | 1              |                   |
| 1              | 0              | 1              | 0              |                   |
| x              | x              | 1              | 1              | Short circuit     |
| 1              | 1              | x              | x              |                   |

### c. DC Motor

In section 3.3, the DC motor of the vehicle has been introduced accurately. The DC motor mentioned above is used to develop this sub-system of the vehicle.

### 3.4.2.5. Vehicle Dynamics

The dynamic model of the vehicle required for longitudinal motion is presented in this section. The longitudinal dynamic model consists of two main elements: vehicle dynamics and powertrain dynamics. Among the factors that greatly affect the vehicle dynamic model are longitudinal tire forces, rolling resistance forces, aerodynamic pulling forces, and gravitational forces. The longitudinal powertrain system of the vehicle consists of an electric motor, a gearbox, a torque converter, and wheels. The inside of the vehicle dynamics sub-system is shown in the following figure.

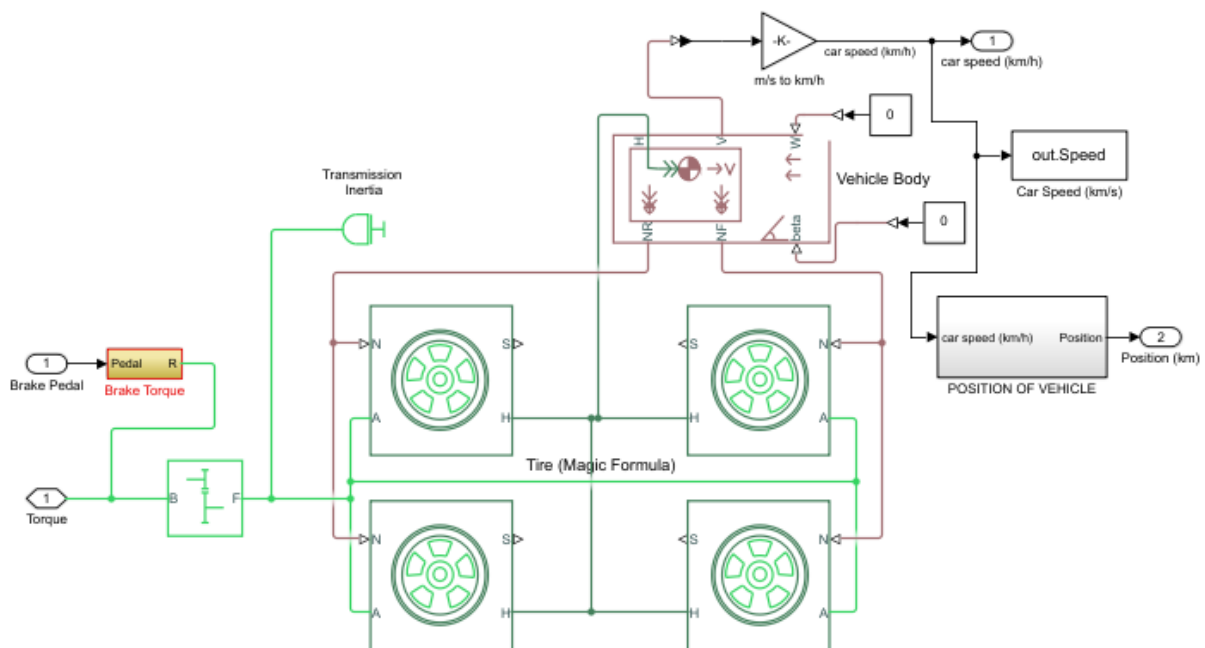


**Figure 3.31.** Inside of the vehicle dynamics sub-system.

Using the Vehicle Body block, we can simulate a two-axle vehicle body that is moving longitudinally. The number of wheels on each axle of the vehicle might be the same or vary. For instance, two wheels would be on the front axle and one wheel would be on the back axle. The size of the vehicle's wheels is presumed to be the same. The vehicle's Center of Gravity (CG) might alternatively be at or below the plane of motion, depending on the configuration. Body mass, aerodynamic drag, road inclination, and weight distribution between axles as a result of acceleration and road profile are all taken into consideration by this block. Pitch and suspension dynamics may be included as options. The vehicle does not change its vertical position in relation to the ground.

### a. Model of Vehicle Body

As seen in Figure 3.32, the 2-DoF vehicle longitudinal model is composed of 1-DoF for the longitudinal motion of the vehicle body and 1-DoF for the rotational motion of the wheels.

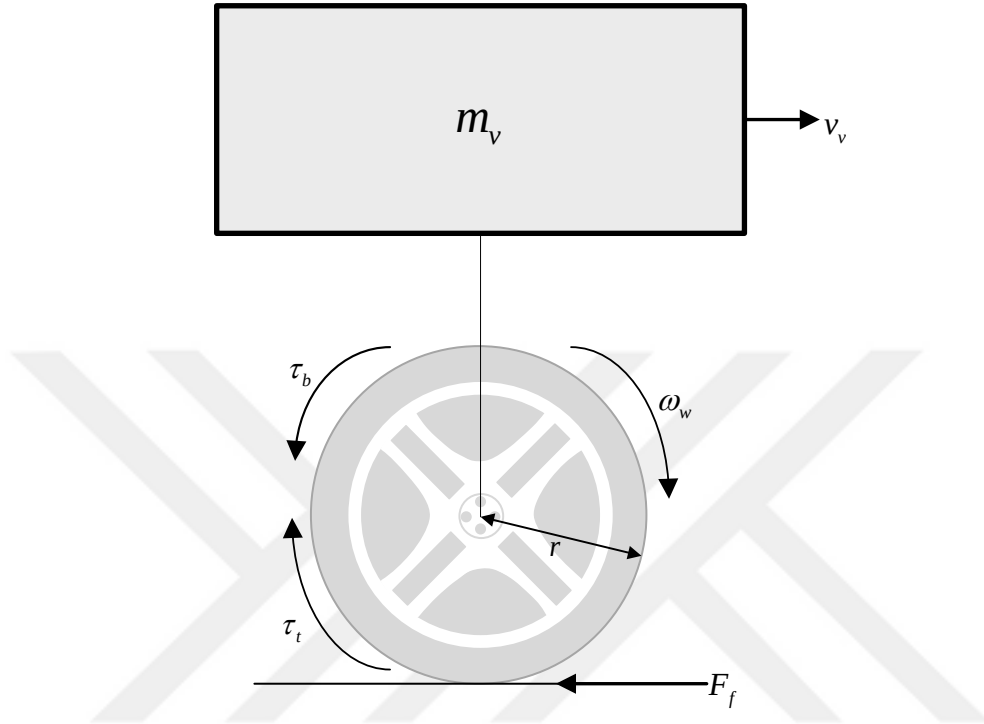


**Figure 3.32.** The longitudinal vehicle model.

The equation of the motion of first DoF that is rotational motion is shown as below:

$$\sum \tau_w = I_w a_w \quad (3.56)$$

The total torque of the wheel is shown in the equation (3.56). So, if we expand the total torque then

$$\tau_t - \tau_b = I_w a_w \quad (3.57)$$

And, tractive torque can be also expressed as:

$$\tau_t = F_f r \quad (3.58)$$

$$F_f r - \tau_b = I_w a_w \quad (3.59)$$

The road resistance of the system is

$$F_f = \mu N \quad (3.60)$$

So, the equation (3.59) will be as:

$$\mu Nr - \tau_b = I_w a_w \quad (3.61)$$

The normal force of the system is

$$N = m_v g \quad (3.62)$$

Thus, the equation (3.61) will be as below:

$$\mu m_v g r - \tau_b = I_w a_w \quad (3.63)$$

Then,

$$a_w = \frac{1}{I_w} (\mu m_v g r - \tau_b) \quad (3.64)$$

Longitudinal force is

$$F_f = F_l = m_v a_v = \mu m_v g \quad (3.65)$$

The longitudinal speed for vehicle and wheel are

$$\begin{aligned} v_v &= r \omega_v \\ v_w &= r \omega_w \end{aligned} \quad (3.66)$$

Then, the longitudinal slip is

$$s = \frac{v_v - v_w}{v_v} \quad (3.67)$$

The paramaters of the equations that are mentioned above is shown in Table 3.6.

**Table 3.6.** The paramaters of the longitudinal model.

| Symbol     | Description                | Unit               |
|------------|----------------------------|--------------------|
| $F_l$      | Longitudinal force         | N                  |
| $F_f$      | Road resistance            | N                  |
| $\omega_w$ | Wheel angular speed        | rad/s              |
| $v_v$      | Longitudinal vehicle speed | m/s                |
| $v_w$      | Longitudinal wheel speed   | m/s                |
| $a_v$      | Longitudinoal acceleration | m/s <sup>2</sup>   |
| $m_v$      | Vehicle mass               | kg                 |
| $r$        | Tire radius                | m                  |
| $\tau_t$   | Tractive torque            | Nm                 |
| $\tau_b$   | Brake torque               | Nm                 |
| $\tau_w$   | Wheel torque               | Nm                 |
| $I_w$      | Wheel moment of inertia    | kgm <sup>2</sup>   |
| $a_w$      | Wheel angular acceleration | rad/s <sup>2</sup> |
| $s$        | Longitudinal slip ratio    | constant           |
| $\mu$      | Coefficient of friction    | constant           |
| $g$        | Gravitational acceleration | m/s <sup>2</sup>   |
| $N$        | Wheel normal force         | N                  |

## b. Tire Model

Based on four fitting parameters, the Magic Formula that is shown in the equations below provides a model for tire longitudinal behavior. The tire dynamics block may be used to simulate tire dynamics under either constant or changeable pavement conditions. Using the tire block, the tire is represented as a stiff wheel-tire combination in contact with the road that is susceptible to slippage. Torque supplied to a wheel axle, the tire pushes the ground (while being subject to contact friction) and sends the ensuing reaction as a force back to the wheel itself. The wheel is propelled forward or backward by this movement. The forces acting on

the tire are shown in the following figure. The tire model variables are also defined in Table 3.7.



**Figure 3.33.** The forces affecting the tire.

The longitudinal force  $F_x$  on the tire is calculated using a tire model's steady-state tire characteristic function,  $F_x = f(k, F_z)$ , which is based on the vertical load ( $F_z$ ) and the wheel slip ( $k$ ). The Magic Formula is a special version of the tire characteristic function, which is defined by four dimensionless coefficients  $B$ ,  $C$ ,  $D$ , and  $E$ , which correspond to stiffness, shape, peak, and curvature, respectively, and are as follows:

$$F_x = f(k, F_z) = F_z \cdot D \cdot \sin\left(C \cdot \arctan\left(Bk - E\left[Bk - \arctan(Bk)\right]\right)\right) \quad (3.68)$$

where the slope of  $f$  at  $k=0$  is  $BCD \cdot F_z$ .

**Table 3.7.** The parameters of tire model.

| Symbol   | Description                     |
|----------|---------------------------------|
| $\Omega$ | Wheel angular velocity          |
| $r_w$    | Wheel radius                    |
| $V_x$    | Wheel hub longitudinal velocity |

|                             |                                                                     |
|-----------------------------|---------------------------------------------------------------------|
| $r_w \Omega$                | Tire tread longitudinal velocity                                    |
| $V_{sx} = r_w \Omega - V_x$ | Wheel slip velocity                                                 |
| $k = \frac{V_{sx}}{ V_x }$  | Wheel slip                                                          |
| $F_z$                       | Vertical load on the tire                                           |
| $F_x = f(k, F_z)$           | The longitudinal force is exerted on the tire at the contact point. |

### 3.5. The EVs Modelling in the MapleSim

This section performs the construction of the EV simulation by using MapleSim software. As shown in Figure 3.34, all of the components of the EV are linked to one another in a multibody configuration. As a result of the simulation, the real speed and position information of the vehicle and the outputs obtained in the Simulink environment are compared.

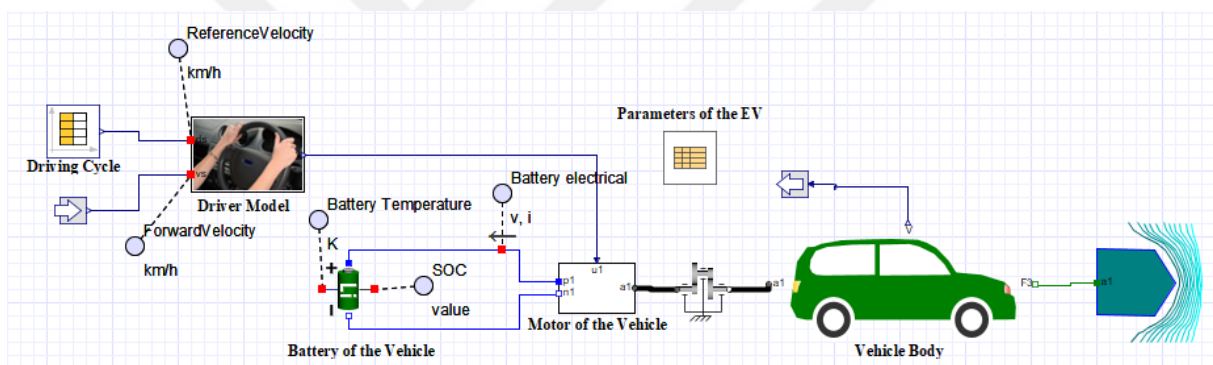


**Figure 3.34.** The simulation of the EVs in the MapleSim.

As seen in the figure above, the vehicle model consists of five submodels: driving cycle, driver model, battery of the vehicle, motor of the vehicle, and vehicle body. These submodels have been expressed in Section 3.4.

### 3.6. Artificial neural networks

Artificial neural networks (ANNs) consist of processing units called neurons. The NNs try to replicate the structure and behaviour of the natural neuron. A neuron consists of inputs and one output. It is a computing system originating from a biological brain's structure, method of processing, and learning ability. In a commonly used NN architecture, the multi-layer NNs, the neurons are arranged into layers. One or more hidden layers are included in the developed system, but they are not part of the input or output of the system. Artificial neurons consist of an interconnected structure as represented in Figure 3.35 and have a processing node indicated

by circles. Sequential data from the simulation (vehicle speed, engine angular speed, engine voltage, and SoC) is assigned to the input layer. Each neuron that is placed on the input layer distributes the values among itself and the middle layer neurons. There is a connection weight between the layers to perform the multiplication between the middle neurons and the input neurons. The neurons in the middle layer are transferred to a function called the activation function, which is usually in logarithmic or sigmoid form, by taking the sum of the weight values coming to this layer. The result from the activation function becomes the output of that middle neuron. Each neuron in the middle layer is connected to an output neuron, and there is a connection weight for each neuron along with this connection. Finally, the output neuron calculates the weighted sum of the inputs, and the nonlinear function weighted sum is applied. As a result of this process, the output of the entire neural network structure is calculated.

Back-propagation (BP) is a systematic algorithm for training multi-layer NNs with error determination of the output node to the hidden nodes. Once the hidden layer errors are determined, the delta rule is applied to adjust the weights and unknown parameters with differentiable activation function units to learn a training set of input-output examples. The illustration of the BP algorithm is given in the following figure.

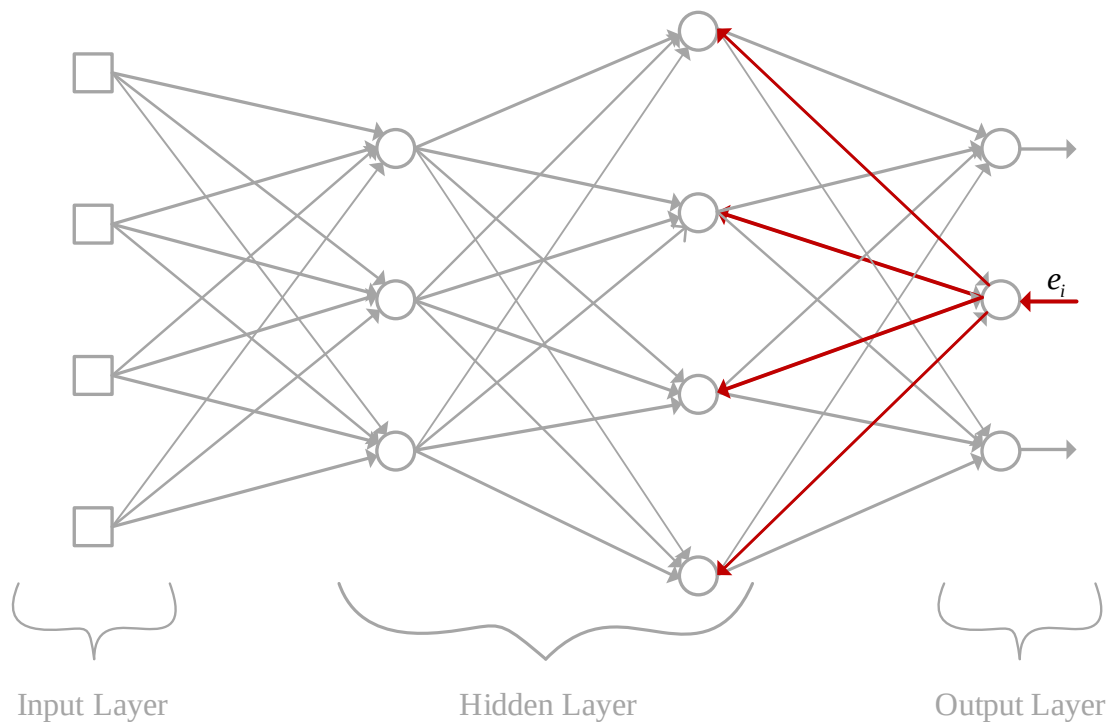


**Figure 3.35.** The illustration of the BP algorithm.

BP is a neural network operation that can be divided into two steps: feed forward and back propagation. In the feed forward part, a pattern applied from the input part is propagated layer by layer until the output is obtained. An error signal is calculated by comparing the output value of the neural network with the value intended to be obtained. To a certain extent, since hidden nodes contribute to this calculated error signal, output error signals are transmitted back from the output layer to each directly contributing node in the hidden layer. This process is repeated layer by layer until an error signal is obtained, detailing each node's contribution to the overall error signal.

After the error signal is determined for each node, the error values are used to update the coupling parameters until the parameter region is available for coding all training patterns. The BP algorithm searches the weight space until the error function reaches its minimum value, and this is called the delta rule or gradient descent. The weights that minimize the error function are then assigned as the solution to the learning problem.

### 3.6.1. Generalized the BP Algorithm

Let us assume that we have the neurons shown in Figure 3.36.

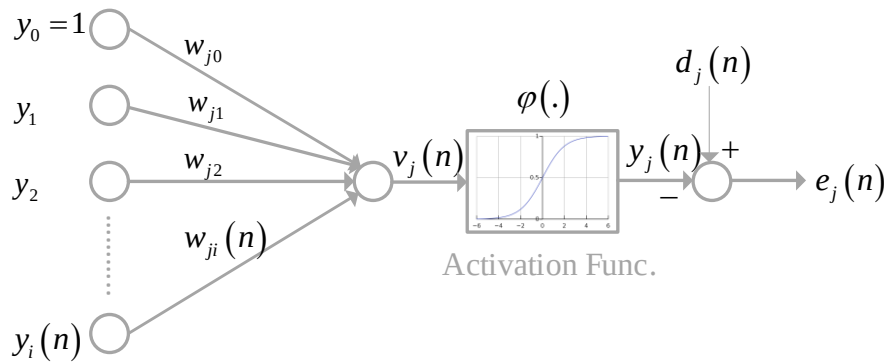


**Figure 3.36.** Block diagram of the neuron without hidden layer.

Error signal at the output of the neuron  $j$  is step  $n$  is equal to the difference between desired and obtained output:

$$e_j(n) = d_j(n) - y_j(n) \quad (3.69)$$

Mean square error in step  $n$ , in the output layer, is

$$E(n) = \frac{1}{N} \sum_{n=1}^N \left( \frac{1}{2} \sum_{j \in C} e_j^2(n) \right) \quad (3.70)$$

The goal of learning is to determine a set of weights that minimize the mean error  $E(n)$ . This goal is achieved using an algorithm similar to the Least Mean Square (LMS) learning algorithm.

The output of the neuron  $j$  is equal to

$$v_j(n) = \sum_{i=0}^p w_{ji}(n) y_i(n) \quad (3.71)$$

where  $p$  is the number of inputs into the neuron  $j$ .

Activation of the neuron  $j$  is equal to:

$$y_j(n) = \varphi_j(v_j(n)) \quad (3.72)$$

Error correction is performed similarly to the LMS algorithm in the direction of the negative gradient (steepest descent minimization method).

The gradient is calculated as follows:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \quad (3.73)$$

The partial derivative of  $E(n)$  with respect to the  $e_j(n)$ :

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \quad (3.74)$$

The partial derivative of  $e_j(n)$  with respect to the  $y_j(n)$ :

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \quad (3.75)$$

The partial derivative of  $y_j(n)$  with respect to the  $v_j(n)$ :

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \dot{\phi}_j(v_j(n)) \quad (3.76)$$

The partial derivative of  $v_j(n)$  with respect to the  $w_{ji}(n)$ :

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \quad (3.77)$$

Based on the equation (3.73) the equation is

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) \dot{\phi}_j(v_j(n)) y_i(n) \quad (3.78)$$

Error correction is defined as:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \quad (3.79)$$

Therefore, the equation (3.79) can be written as:

$$\Delta w_{ji}(n) = \eta e_j(n) \dot{\phi}_j(v_j(n)) y_i(n) = \eta \delta_j(n) y_i(n) \quad (3.80)$$

where  $\delta_j(n)$  is the so-called local gradient:

$$\delta_j(n) = e_j(n) \dot{\phi}_j(v_j(n)) \quad (3.81)$$

At last, the updated unknown parameter of the output layer equation is

$$w_{ji} = w_{ji} + \Delta w_{ji} \quad (3.82)$$

As can be seen, error correction depends on the error at the output of the neuron  $j$ . To calculate the error, we consider two cases that are

- The neuron  $j$  is an output neuron

If the neuron  $j$  is the output layer, then the error  $e_j(n)$  is easily calculated because we know the desired output  $d_j(n)$ . The local gradient  $\delta_j(n)$  is also calculated using the previously derived expression in the equation (3.81).

- The neuron  $j$  is hidden

Let us assume that the neuron  $j$  is a hidden layer connected to the output layer as shown in Figure 3.37.

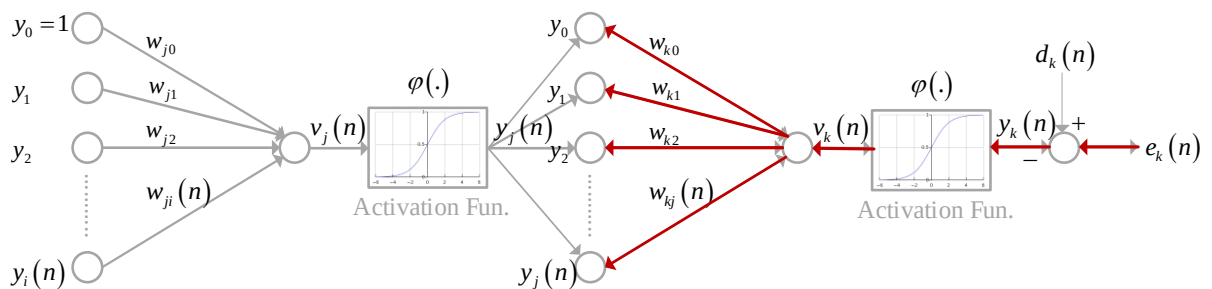


**Figure 3.37.** Block diagram of neuron with hidden layer and BP.

If the neuron  $j$  is in a hidden layer, then the desired output of this neuron is unknown. In this case, the error of a hidden can be estimated based on the error of neurons in the next layer to which the hidden is connected. For this case, the local gradient can be re-written as;

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \dot{\phi}_j(v_j(n)) \quad (3.83)$$

The partial derivative of the error with respect to the input can be calculated as follows. Let us start with the expression:

$$E(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \quad (3.84)$$

The partial derivative is then equal to:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \quad (3.85)$$

Two derivatives on the right hand side can be calculated as follows. Since,

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)) \quad (3.86)$$

Then,

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\dot{\varphi}_k(v_k(n)) \quad (3.87)$$

The activity of a neuron  $k$  is equal to:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \quad (3.88)$$

Based on the derived expressions, it is seen that:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n) \dot{\phi}_k(v_k(n)) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n) \quad (3.89)$$

Finally, it can be written that for hidden neuron  $j$  local gradient is equal to:

$$\delta_j(n) = \dot{\phi}_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \quad (3.90)$$

Where  $\delta_k(n) = e_k(n) \dot{\phi}_k(v_k(n))$  and using the local gradient error correction can be calculated for hidden neurons  $j$ .

At last, the updated unknown parameter of the hidden layer equation is

$$w_{kj} = w_{kj} + \Delta w_{kj} \quad (3.91)$$

- Let's summarize the BP algorithm

Correction  $\Delta w_{ji}(n)$  is calculated using the delta rule:

$$\begin{aligned} (\text{correction}) &= (\text{parametar}) \times (\text{local gradient of neuron } j) \times (i^{th} \text{ input into neuron } j) \\ \Delta w_{kj}(n) &= \eta \delta_j(n) y_i(n) \end{aligned} \quad (3.92)$$

If the neuron  $j$  is an output neuron then,

$$\delta_j(n) = e_j(n) \dot{\phi}_j(v_j(n)) \quad (3.93)$$

If the neuron  $j$  is hidden then,

$$\delta_j(n) = \dot{\phi}_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \quad (3.94)$$

### 3.6.2. Pseudo Code: Delta Learning Rule for $m$ Inputs and $n$ Outputs NNs

1. **Input:** Training input  $x \in \mathbb{R}^{m \times l}$ , where  $l$  is the length of each  $m$  number of the input data, labeled output  $y \in \mathbb{R}^{n \times l}$ , where  $n$  is the number of the output, randomly initialized unknown hidden parameter matrix/vector  $w_x \in \mathbb{R}^{h \times m}$  where  $h$  is the number of the input hidden node, randomly initialized unknown output hidden parameter matrix/vector  $w_y \in \mathbb{R}^{n \times h}$ , estimated output  $\hat{y}_o \in \mathbb{R}^{n \times l}$ , saturated and estimated  $\hat{y} \in \mathbb{R}^{n \times l}$ , the training error  $E \in \mathbb{R}^{n \times l}$ , the number of the multiple simulations iteration, store matrix  $w_s$ , store error  $e_s$ , store the estimated output  $\hat{y}_s$ .
2. **Output:** The terminal value of the trained parameters  $w_x$  and  $w_y$ , store learning error  $e_s$ , store output  $\hat{y}_s$ .
3. for  $i$  to iteration
4. for  $j$  to  $l$
5. Calculate the estimated current input hidden  $\hat{w}_x$
6.  $\hat{w}_x = w * x(:, j)$
7. Apply the activation function on the current input hidden  $\hat{w}_x$
8.  $\hat{w}_y = \varphi(w_x)$
9. Calculate the estimated current output  $\hat{y}_o$
10.  $\hat{y}_o = w_y * \hat{w}_y$
11. Apply the activation function on the current output
12.  $\hat{y}(:, j) = \varphi(\hat{y}_o)$
13. Determine the error
14.  $E(:, j) = y(:, j) - \hat{y}(:, j)$
15. Apply the delta rule for the hidden layer
16.  $\Delta = \hat{y}(:, j) * (1 - \hat{y}(:, j)) * E(:, j)$
17. Determine error for the output layer
18.  $E_1 = w_y^T * \Delta$
19. Apply the delta rule for hidden output
20.  $\Delta 1 = \hat{y}(:, j) * (1 - \hat{y}(:, j)) * E_1$

21. Update the learning rate
22. 
$$\mu(i, j) = 2 / \left( x(:, j)^T * x(:, j) \right)$$
23. Update the unknown parameter for the hidden layer
24. 
$$w_x = w_x + \mu(i, j) * \Delta 1 * x(:, j)^T$$
25. Update the unknown parameter for the output layer
26. 
$$w_y = w_y + \mu(i, j) * \Delta * \hat{w}_y^T$$
27. Store the unknown parameters,
28. 
$$w_{s1} = [w_{s1}; \text{reshape}(w_x, 1, [])]$$
29. 
$$w_s = [w_s; \text{reshape}(w_y, 1, [])]$$
30. Determine the Mean Error (ME)
31. 
$$\text{sumErr} = \text{sumErr} + 0.5 * E(:, j)$$
32. 
$$\text{MSE}(:, j) = \text{sumErr} / j$$
33. end  $j$
34. Store the error and the saturated output
35. 
$$e_s = [e_s; \text{reshape}(e, 1, [])]$$
36. 
$$\hat{y}_s = [\hat{y}_s; \text{reshape}(\hat{y}, 1, [])]$$
37. If  $e(:, j) < e_t$  then
38. break
39. end if
40. end  $i$

### 3.7. Intelligent Control of the AVs

In this section, lanes detection and object detection, which are the most significant control mechanisms for the AV, are constructed using Python and OpenCV. The Python programming language is implemented using the Anaconda and Atom software packages, while the image processing portion of the control system is implemented using the OpenCV toolbox. This section contains the two subsections to control the AVs, which is the lane line detection and vehicle detection & tracking.

### 3.7.1. The Lane Line Detection System

The purpose of this section is to construct a program that can identify lane lines in a picture or a video. When we drive a car, we can see where the lane lines are by using our eyes. However, vehicles do not have any eyes, and that's where computer vision comes in, which through complex algorithms helps the computer see the world as we do. A series of camera images are used for the AVs to identify the lane lines. If we can specify the lane lines from the image, then we can also recognize them from the video.

The initial stage of this section is to identify lane lines from the image. As shown in Figure 3.38, an image of a road is taken into account in order for the vehicle to be able to operate autonomously. This information is needed to calculate the lane lines on the portion of the road where the vehicle is traveling.



**Figure 3.38.** The image of the road.

To detect lane lines in a road image, several processes have to be done: edge detection, region of interest, and Hough transform. In this section, the processes required for lane line detection are applied step-by-step to the road image. In addition, these processes are explained in detail on the road image.

### 3.7.1.1. The Edge Detection

If we want to define the lane lines in the above road image, OpenCV, which is the computer vision library, is used to detect the lanes. First and foremost, it is important to import the OpenCV library in order to define the lane lines (`import cv2`). From this library, we can access functions that are `imread()` and `imshow()`. By using these functions, we can load the road image shown above so that it can be processed into the code. The `imread` function constructs the image as a multi-dimensional numpy array containing the relative intensities of each pixel. After this process, the edges in the picture are detected with the canny edge technique.

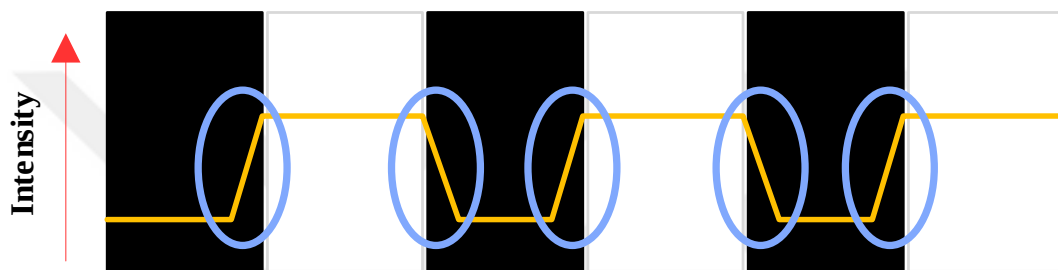


**Figure 3.39.** The intensity of the pixels.

The goal of edge detection is to identify the boundaries of objects within images. In other words, it defines sharp changes in intensity in adjacent pixels. It's important to recognize that an image can be read as a matrix of an array of pixels. A pixel contains the light intensity at some location in the image that is shown in Figure 3.40. The intensity of each pixel is denoted by a numeric value that ranges from 0 to 255 in the RGB color space. The following figure shows the numeric values of the pixels.

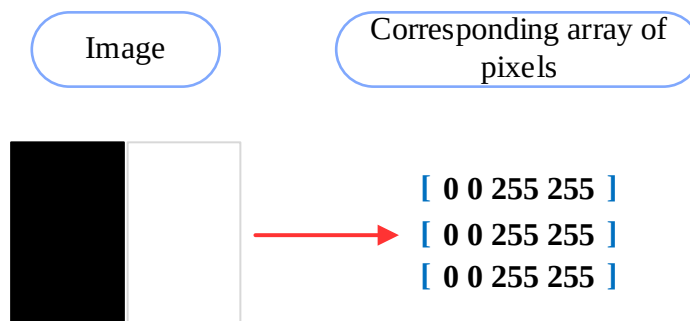


**Figure 3.40.** The numeric values of the pixels.

The intensity value of 0 indicates no intensity if something is completely black, whereas 255 represents the maximum intensity of something being completely white. That being said, a

gradient is a change in brightness over a series of pixels. As shown in Figure 3.41, a strong gradient indicates a steep change, whereas a small gradient represents a shallow change.

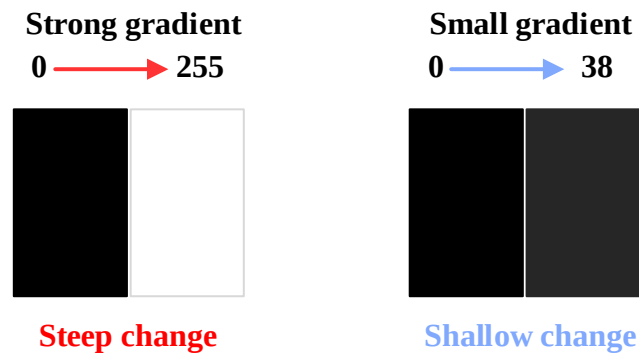


**Figure 3.41.** Types of the gradient.

Considering the ball in Figure 3.42, we are looking at the gradient of the soccer ball. The outlines of white pixels correspond to the discontinuity and brightness at the points of the strength ingredient. This helps us identify edges in our image since the edge is defined by the difference in intensity values in adjacent pixels. Wherever there's a sharp change in intensity, there's a rapid change in brightness. Furthermore, wherever there's a strong gradient, there is a corresponding bright pixel in the gradient image by tracing out all of these pixels. We obtain the edges by tracing out all of these pixels.

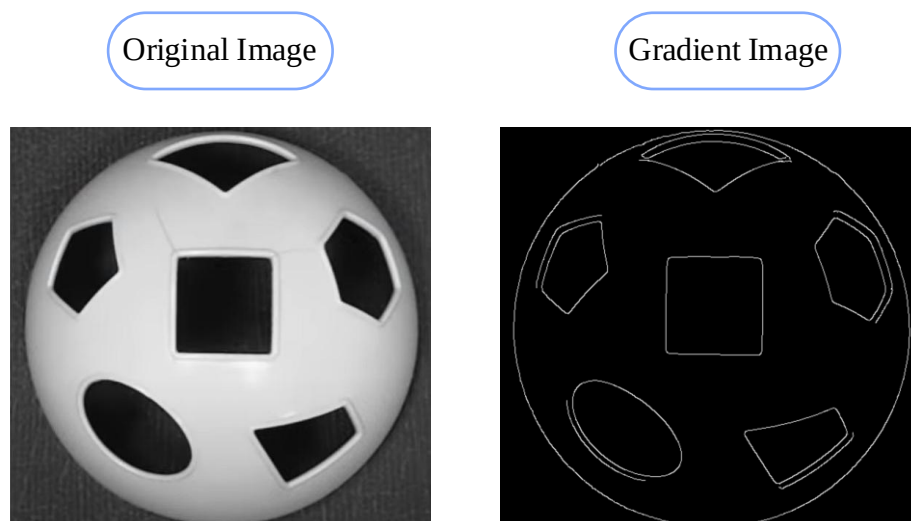


**Figure 3.42.** The gradient of the image.

This is a multi-step process. Step one is to convert our image to grayscale. A three-channel color image would have red, green, and blue channels for each pixel, a combination of three

intensity values, whereas a grayscale image only has one channel for each pixel, with only one intensity value ranging from 0 to 255. The point is that by using grayscale image processing, a single channel is faster than three-channel color images and less computationally expensive. The following figure shows the transition to grayscale of Figure 3.38.



**Figure 3.43.** The grayscale of the road image.

Step number two of edge detection is to apply Gaussian blur to the grayscale road image. In other words, Gaussian blur is used to reduce noise in the image when detecting edges. While it is important to accurately catch as many edges in the image as possible, image noise can create false edges and ultimately affect edge detection. Each of the pixels in a grayscale image is represented by a single number that describes the brightness of the pixel. The smoothening image would be modifying the value of a pixel with the average value of the pixel intensities around it. Averaging out the pixels in the image to reduce noise will be done with a kernel. Essentially, this kernel of normally distributed numbers is run across our entire image and sets each pixel value equal to the weighted average of its neighboring pixels. The pseudo code of the kernel convolution is shown in Table 3.8.

**Table 3.8.** The pseudo code of the kernel convolution.

---

1. **for each** image row **in** input image

```

2.  for each image row in input image
3.      set accumulator to zero
4.  for each kernel row in kernel
5.      for each element in kernel row
6.          If element position corresponding to pixel position then
7.              multiply element value corresponding to pixel value
8.              add result to accumulator
9.          endif
10. endfors

```

---

The method of image convolution is used in conjunction with a Gaussian Kernel (3x3, 5x5, 7x7, etc.). The size of the kernel is determined by the blurring effect that is intended. Generally speaking, the smaller the kernel size, the less obvious the blur. In this example, we will use a Gaussian kernel with dimensions of 5 by 5. The equation for a Gaussian filter kernel of size  $(2k+1)(2k+1)$  is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2} e^{\left( \frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2} \right)} \quad \text{where } 1 \leq i, j \leq (2k+1) \quad (3.95)$$

If we apply the Gaussian blur to the grayscale image with a 5x5 kernel, the blurred image is constructed. That means reducing the noise in the image. The image with Gaussian blur applied and noise reduced can be seen in Figure 3.44.



**Figure 3.44.** The blurred of the road image.

After obtaining the blurred image above, the next step is to apply the canny function to the final image. In order to use the canny function, the image must be represented by the two-dimensional coordinates  $x$  and  $y$ . As seen in Figure 3.45, the  $x$  axis traverses the image width, and the  $y$  axis goes along the image height. In addition, the height of the image represents the number of columns and the width of the image represents the number of rows. Therefore, the product of the width and height gives the total number of pixels in the image.

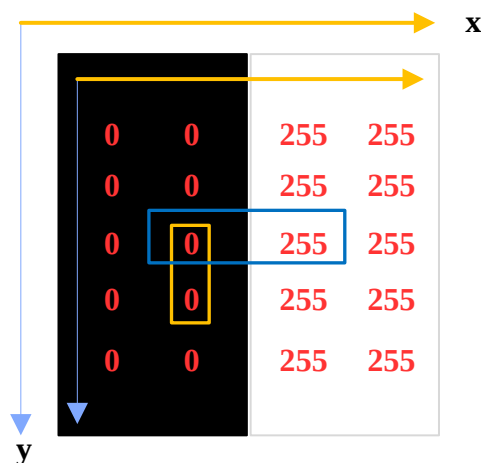


**Figure 3.45.** The corresponding image of the road.

The point is that we can look at our image not only as an array but also as a continuous function of  $x$  and  $y$  since it's a mathematical function. We can perform the mathematical operation  $f(x,y)$  to determine a rapid change in brightness in the image. So, the canny function performs the derivative on the function  $d(f(x,y))$  in both  $x$  and  $y$  directions by measuring the change in intensity with respect to adjacent pixels. As seen in the figure above, a small derivative is a small change (yellow square) in intensity, whereas a big derivative is a big change (blue square). We are computing the gradients by computing the derivative in all directions of the image. Since the gradient is the change in brightness over a series of pixels, canny function helps to compute the gradient in all directions of the blurred image. It then traces the strongest gradients as a series of white pixels. This function has low and high thresholds, so they allow to isolate the adjacent pixels that follow the strongest gradients.

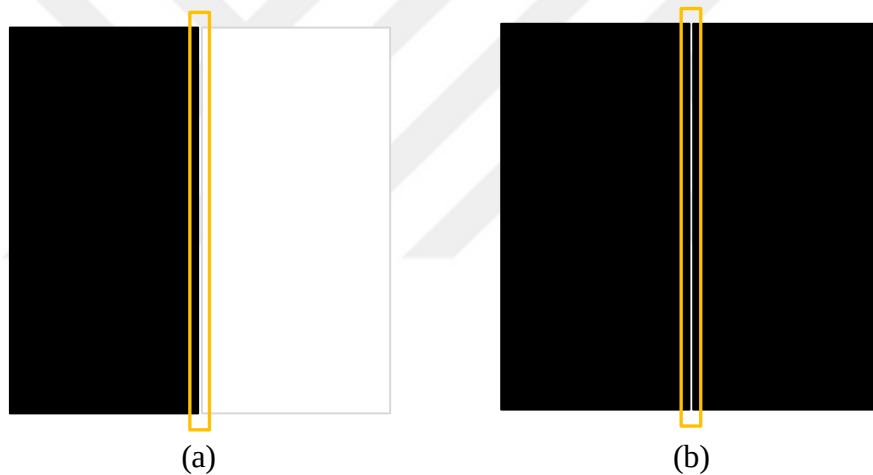


**Figure 3.46.** Definition of low and high threshold in the canny function.

As shown in Figure 3.46 (a), if the gradient is larger than the upper threshold, then it is accepted as an edge pixel. If it is below the lower threshold that is shown in Figure 3.46 (b) it is rejected. If the gradient is between the thresholds, then it will be accepted only if it is connected to a strong edge. If we define these thresholds, the best choice would be 50 for the low threshold and 150 for the high threshold. Therefore, if the canny function is applied to the blurred image, the edge detection of the road image is constructed as shown in the following figure.

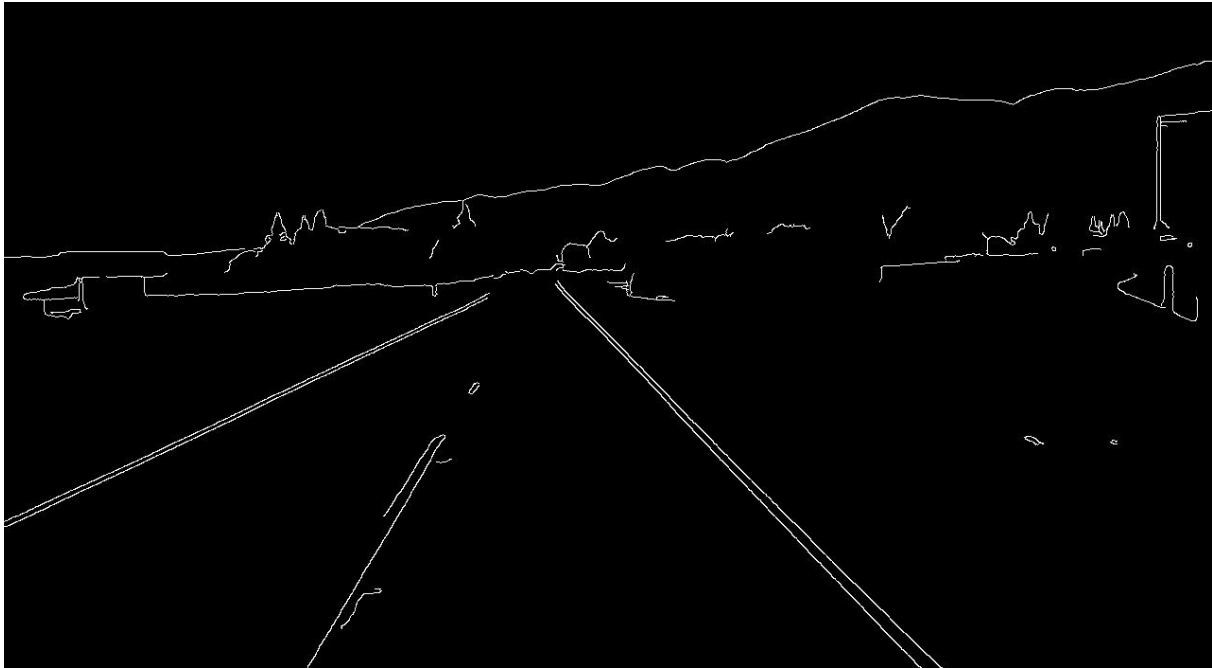


**Figure 3.47.**Applying the canny function to the road image.

The figure above is the gradient image that clearly traces an outline of the edges that correspond to the sharpest changes in intensity. The gradients that exceed the high threshold are traced as bright pixels identifying adjacent pixels in the image. The rapid or small changes in brightness are not traced. In other words, if the gradients are below the low threshold, then these pixels are changed to black. The pseudo code of the edge detection is shown in Table 3.9.

**Table 3.9.**The pseudo code of the edge detection.

|    |                                    |
|----|------------------------------------|
| 1. | <b>read</b> the image              |
| 2. | <b>copy</b> the image              |
| 3. | <b>function</b> canny(image)       |
| 4. | <b>rgb2gray</b> the copied image   |
| 5. | <b>gaussianblur</b> the gray image |
| 6. | <b>canny</b> the blurred image     |
| 7. | <b>return</b> canny                |
| 8. | <b>show</b> the canny image        |

The edges have been detected on the road image, as seen in the figure above. After this process, the vehicle has to deal with a certain part of the road while it is in motion. This part is called the region of interest. In the following section, this part of the road image is detected.

### 3.7.1.2. The Region of Interest

It is necessary to specify the region of interest that is shown in Figure 3.47 before we identify the lane lines in the image. For the purpose of selecting the area of interest in the road image, the matplotlib library is utilized. The road image is converted to the x-y axis, which represents two-dimensional space, with the help of the matplotlib library.

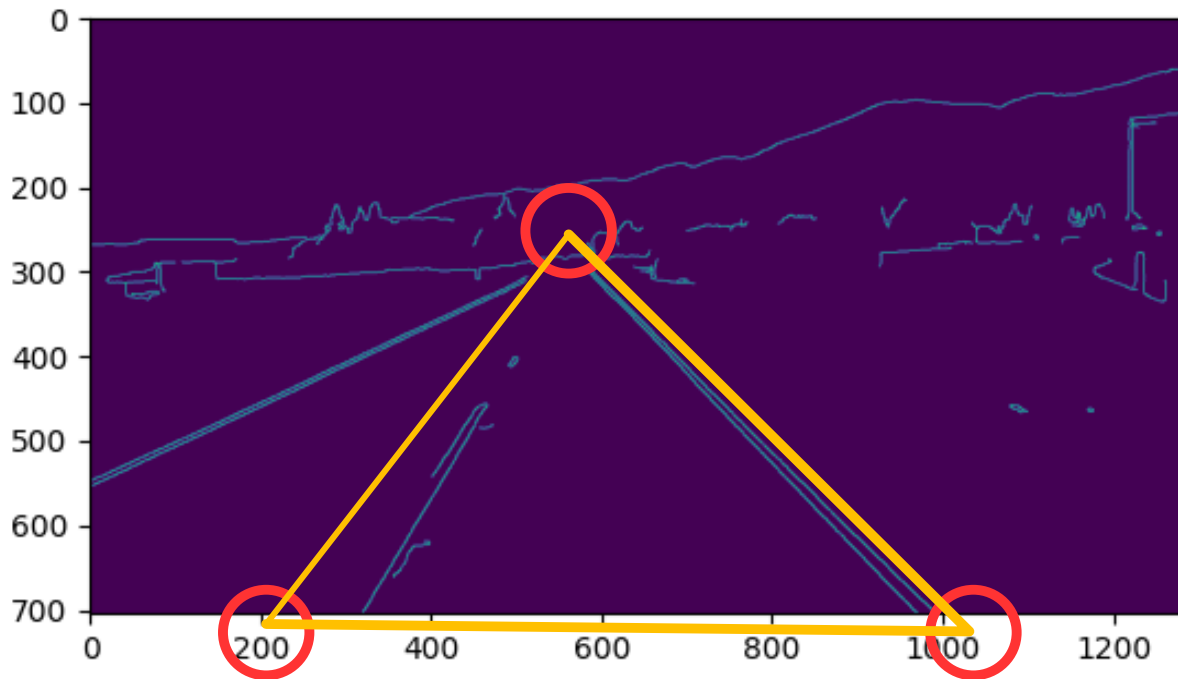


**Figure 3.48.** The canny image in the matplotlib.

As shown in Figure 3.48, we get the same image along with the x-axis and y-axis. The extent of the vehicle's field of view is limited according to the region of interest. If we want to determine the vehicle's "region of interest," we may designate this region as a triangle as shown in the figure above. The three points that make up this triangle have the coordinates (200,700), (1100,700), and (550,250), respectively. The goal of this section is to create an image that is completely black except for this determined triangle (the inside of the triangle is white) with the same dimensions as the road image. In other words, a black mask is applied with the same dimension as the road image. Afterwards, using the triangle whose borders are defined above, a completely white mask is applied to the area bounded by the triangle. Finally, the following figure shows the region of interest the vehicle.

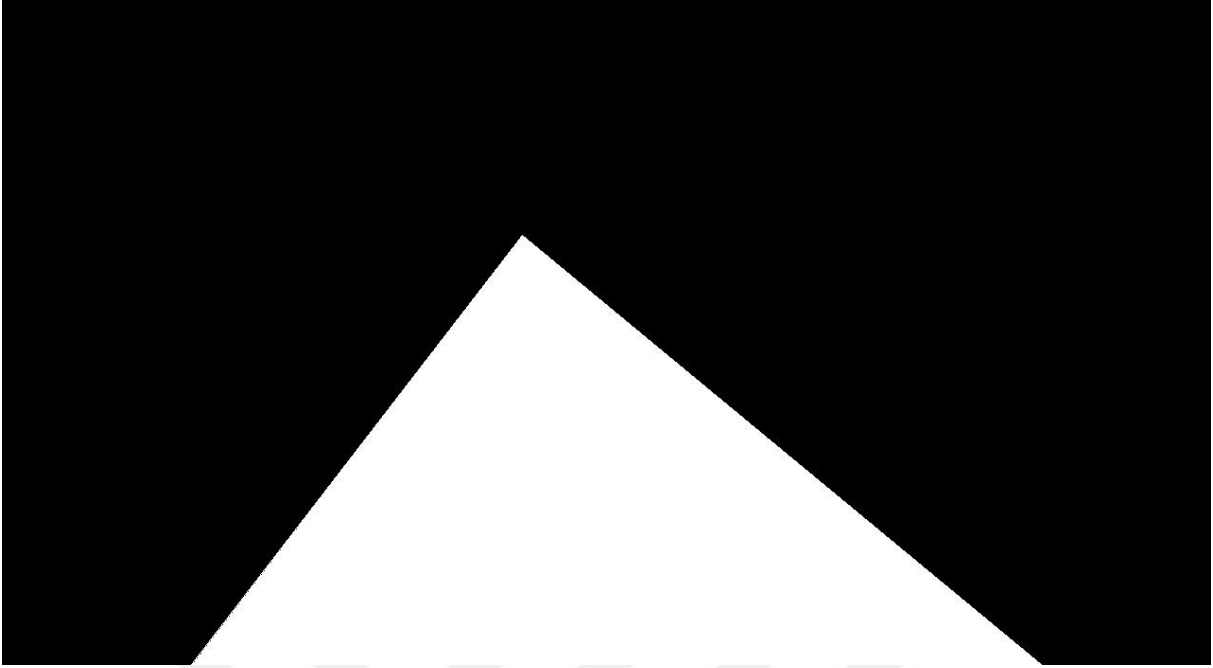


**Figure 3.49.** The masked area of the image.

It is necessary to utilize the triangle image acquired before in order to mask the gradient image. It is required, however, to be familiar with the fundamentals of binary numbers in order to complete this process. Binary numbers are expressed in the base-2 numeral system, which uses only two symbols, typically zeros and ones. If we give an example of the binary number, it is shown in the following equation.

$$(23)_{10} = (10111)_2 \quad (3.96)$$

By using binary numbers, it is desired to combine the gradient image obtained in Figure 3.47 with the triangle image obtained in Figure 3.49. First, the RGB value of the white part in the triangle image is 255 for all pixels, while the RGB value of the pixels in the black parts is 0. If we express the values 0 to 255 as binary, they are 0000 and 11111111 respectively. Therefore, the surrounding region is completely black, which means the binary representation of each pixel intensity in that region would be zeros. On the other hand, for the triangle contour of the image, whose region is completely white, which means the binary representation of each pixel intensity in that region would be one. It is important because this mask is applied to our gradient image to ultimately only show the region of interest by applying the bitwise\_and operation between the two images. The bitwise\_and operation occurs element-wise between the two images or between two arrays of the pixels. Both of these images have the same array

shape, the same dimensions, and the same number of pixels by applying the bitwise\_and. The working principle of the bitwise\_and can be expressed in an example that is shown below.

$$(011001)_2 = (25)_{10} \quad (3.97)$$

$$(110010)_2 = (50)_{10} \quad (3.98)$$

The equations (3.97) and (3.98) are the standard examples, but regardless, let's take their bitwise\_and to explain this process. Therefore, an operation is applied to the between equations (3.97) and (3.98) that means the operation puts a zero unless both pairs are ones. Thus, the following figure shows the working principle of the bitwise\_and operation.

$$\begin{array}{c} (00011001)_2 = (25)_{10} \\ \text{\textcolor{red}{\&}} \\ (00110010)_2 = (50)_{10} \\ \text{\textcolor{red}{\|}} \\ (00010000)_2 = (16)_{10} \end{array}$$

**Figure 3.50.** The working principle of the bitwise\_and.

According to Figure 3.47 and Figure 3.49, the black region whose pixels have intensity values of zeros. By taking the bitwise\_and operation between two images, the black region of Figure 3.47 absorbs the corresponding pixels of the gradient image. In other words, the value of all pixels corresponding to the black region in the gradient image becomes zero. In addition, the pixels corresponding to the white region (triangle contour) in Figure 3.49 in the gradient image remain at the same values since all the values in this white region are ones. The examples of these operations are shown in the figure below.

$$\begin{array}{cc} (00011001)_2 = (25)_{10} & (00011001)_2 = (25)_{10} \\ \text{\textcolor{red}{\&}} & \text{\textcolor{red}{\&}} \\ (00000000)_2 = (0)_{10} & (11111111)_2 = (255)_{10} \\ \text{\textcolor{red}{\|}} & \text{\textcolor{red}{\|}} \\ (00000000)_2 = (0)_{10} & (00011001)_2 = (25)_{10} \\ \text{(a)} & \text{(b)} \end{array}$$

**Figure 3.51.** The pixel values after applying the bitwise\_and.

The pseudo code of the region of interest is shown in Table 3.10.

**Table 3.10.**The pseudo code of the region of interest.

|    |                                                                                                   |
|----|---------------------------------------------------------------------------------------------------|
| 1. | <b>function</b> region_of_intereset                                                               |
| 2. | <b>height</b> ofthe image                                                                         |
| 3. | <b>define</b> the triangle on the image                                                           |
| 4. | <b>mask</b> the image with black                                                                  |
| 5. | <b>fill</b> the triangle with white on the masked image                                           |
| 6. | apply the <b>bitwise_and</b> to the cropped image (difference between the canny and masked image) |
| 7. | <b>show</b> the cropped image                                                                     |

Finally, the following figure shows the gradient image of the road after applying the bitwise\_and operation.

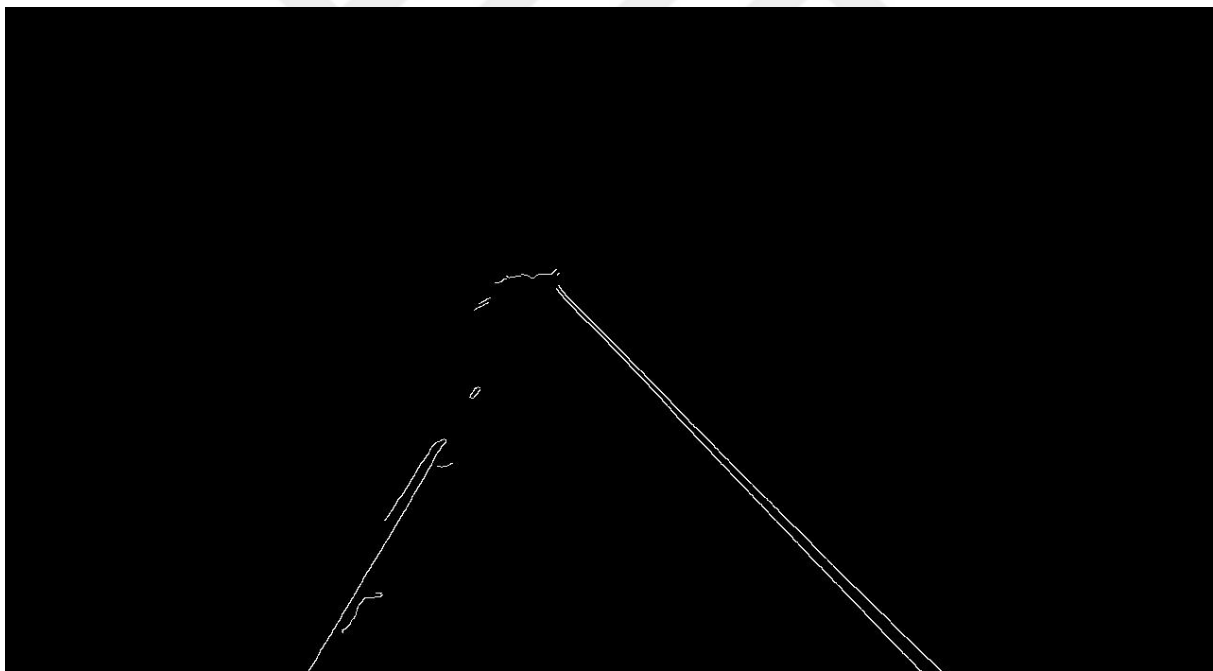


**Figure 3.52.** The cropped image of the canny image.

So far, the edge detection and the isolated region of interest in the gradient image have been identified. The final step is to apply the Hough transform technique to detect straight lines in the region of interest.

### 3.7.1.3. The Hough Transform

After the region of interest is determined for the vehicle, the lane lines of the road are detected by using the Hough transform. As seen in Figure 3.52, 2D coordinate space and inside of it a straight line is constructed. The straight line is represented by the equation (3.99).

$$y = mx + b \quad (3.99)$$

This line equation (3.99) has two parameters that are  $m$  and  $b$ . It is plotting as a function of  $x$  and  $y$  but it can also be represented in the parametric space which is called Hough space as  $b$  versus  $m$ . The  $y$  intercept of the line is 2 that is the value of  $b$  and the slope of the line  $m$  is found in figure (a) below that is 3.

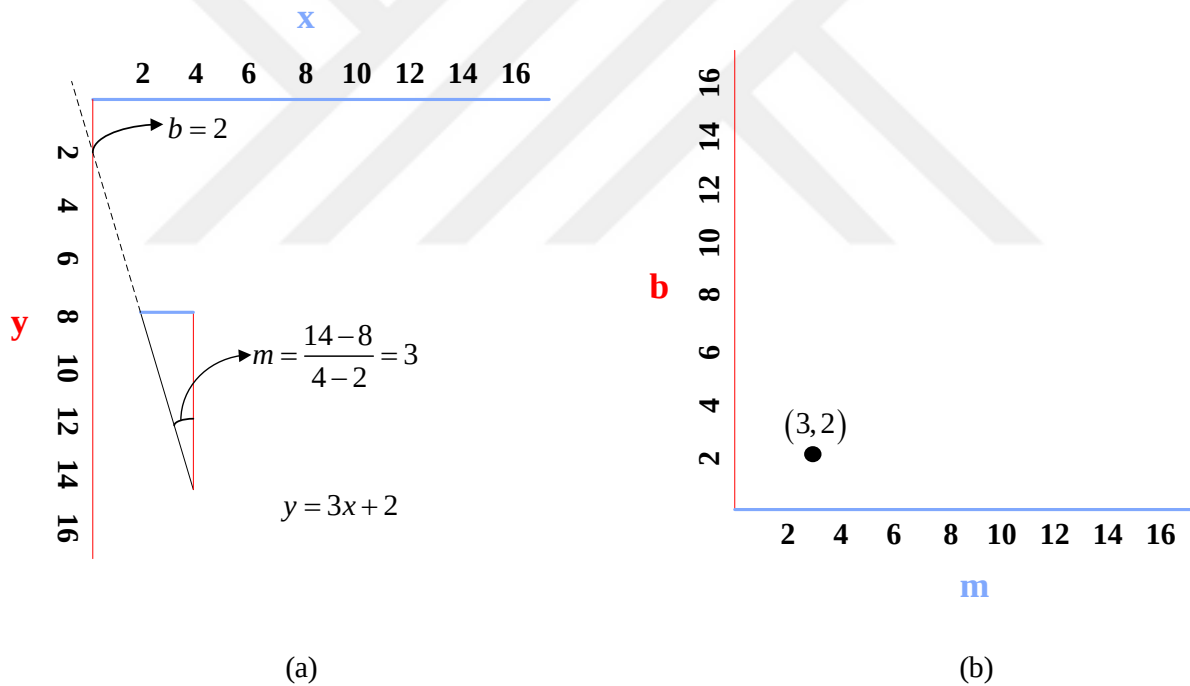


**Figure 3.53.** The line parameters in (a) 2D space and (b) Hough space.

As seen in

Figure 3.53 (b), the line has been transferred into the Hough space that is represented as a single point. In addition, there are many possible lines by considering the single point in 2D space and it has different values for  $m$  and  $b$ . As shown in Figure 3.54(a), the single point is located at the coordinates (2,12) and the line parameters  $m$  and  $b$  that can pass through this

dot can be  $(2,8)$ ,  $(3,6)$ ,  $(4,4)$ ,  $(5,2)$ ,  $(6,0)$ , and so on. It is noticed that a single point in 2D space is represented by a line in Hough space as seen in Figure 3.54(b).

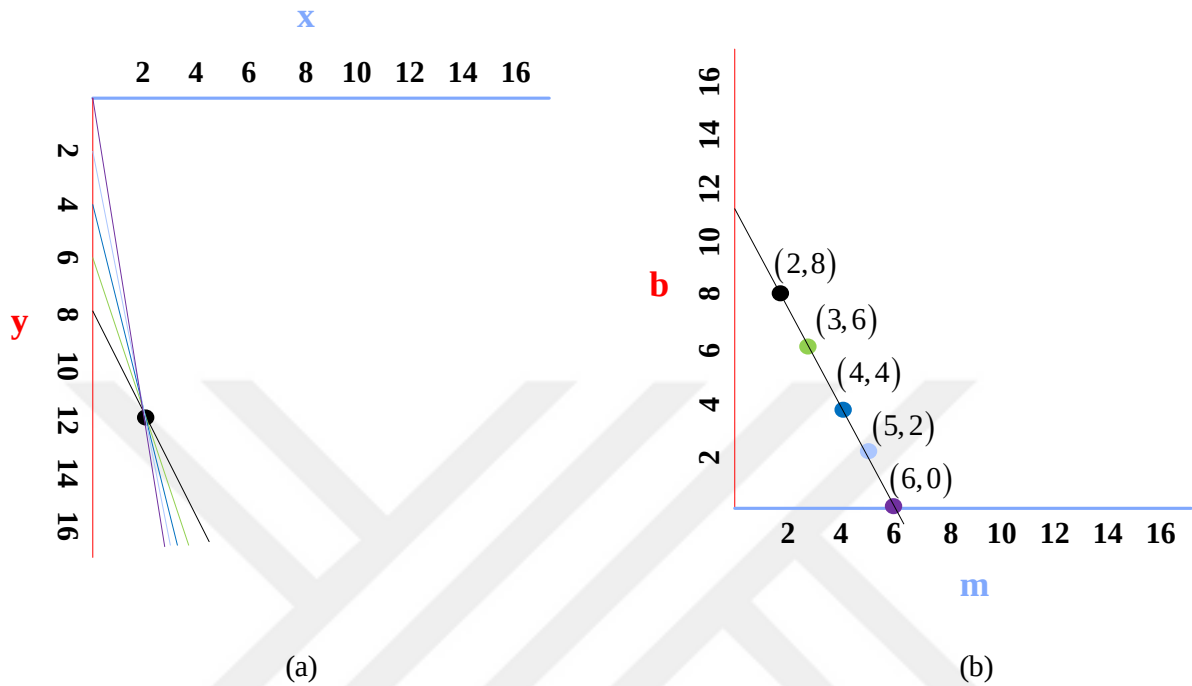


**Figure 3.54.** The line parameters with several lines in (a) 2D space and (b) Hough space.

In other words, by plotting the family of lines that go through our points each line with its own distinct  $m$  and  $b$  value pair, this produces an entire line of  $m$  and  $b$  value pairs in Hough space. By taking into account the point  $(1,8)$  in the 2D space, there are also many lines that can cross this point (each line has different  $m$  and  $b$  values). On the other hand, this point  $(1,8)$  is connected to the other point  $(2,12)$  with one line that is consistent with both points. As shown in Figure 3.55(b), these points have an intersection point in the Hough space because that point of intersection represents the  $m$  and  $b$  values of the line crossing both points. In this case, these two points' parameters  $m$  and  $b$  are equal to 4. If there is one more point that is  $(3,16)$  then this point is also represented by the different lines in 2D space. However, there is another intersection at the same point in the Hough space, which means the line with the following slope and y-intercept  $(m,b)$  with  $(4,4)$  crosses all three points.

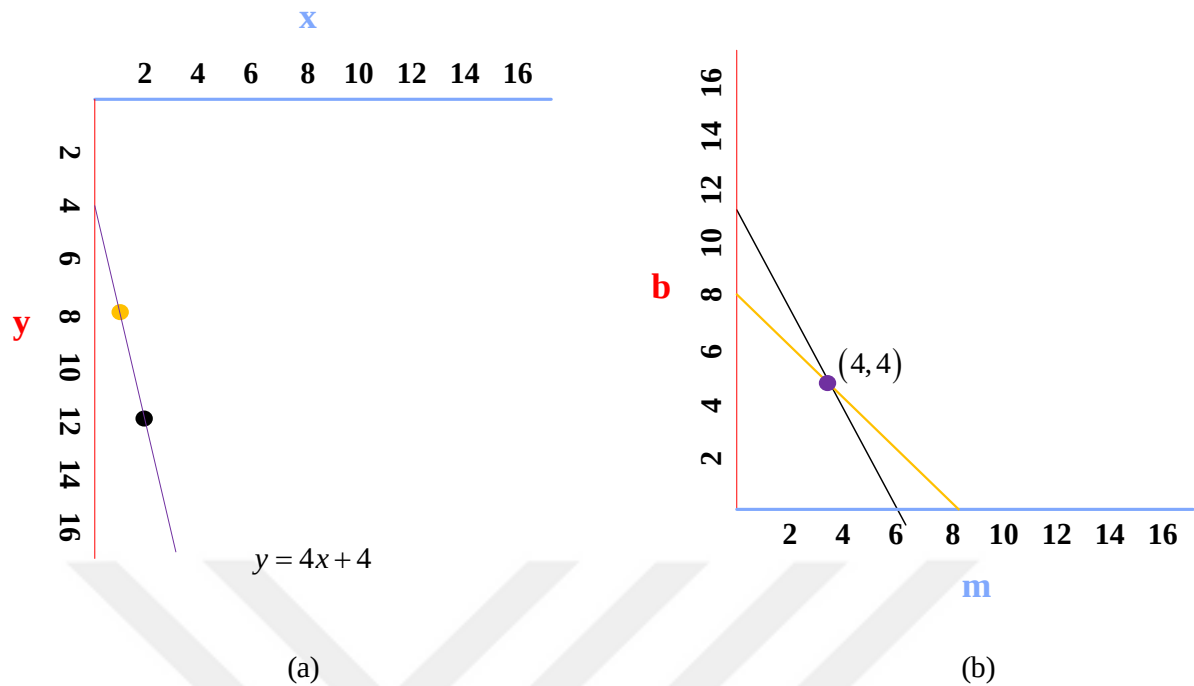


**Figure 3.55.** The intersection of the two points in (a) 2D space and (b) Hough space.

This process is important to find the lane lines in the gradient image because the gradient image has a series of white points which represent the edges in the road image. However, it is necessary to identify the parameters of these points. In order to determine these parameters, four points have been taken, as shown in figure (a) below. In addition, these points correspond to the Hough space shown in figure (b) below.

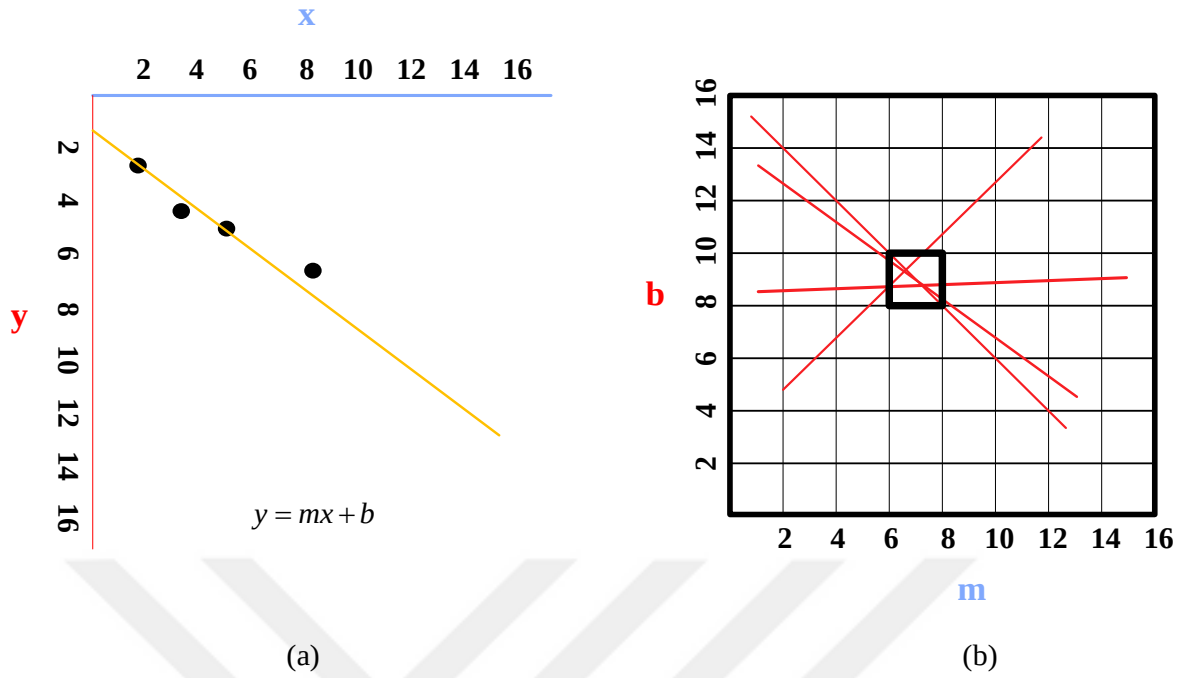


**Figure 3.56.** The intersection of the four points in (a) 2D space and (b) Hough space.

The Hough space is divided into a grid, as shown in Figure 3.56 (b), and each box within the grid corresponds to the slope and y-intercept values of the candidate lines. If we consider these points in the figure above (a), there are several intersection points, as shown in the figure above (b). These points are inside of a single box. For every point of intersection, the box with the highest number of points is the  $m$  and  $b$  values of the line formed in 2D space. Therefore, the best fit parameters of the line in these four points are constructed by using this technique. However, if we take into account the vertical line, the slope of it goes to an infinity that cannot be represented in Hough space. Hence, it is necessary to represent this line with a polar coordinate system. The parameters of the line can be written as  $\rho$  and  $\theta$  instead of  $m$  and  $b$  that is shown in Figure 3.57.  $\rho$  is the perpendicular distance from the origin to this line and  $\theta$  indicates the angle of inclination of the normal line for the x-axis, which is measured in radians clockwise with respect to the positive x-axis.

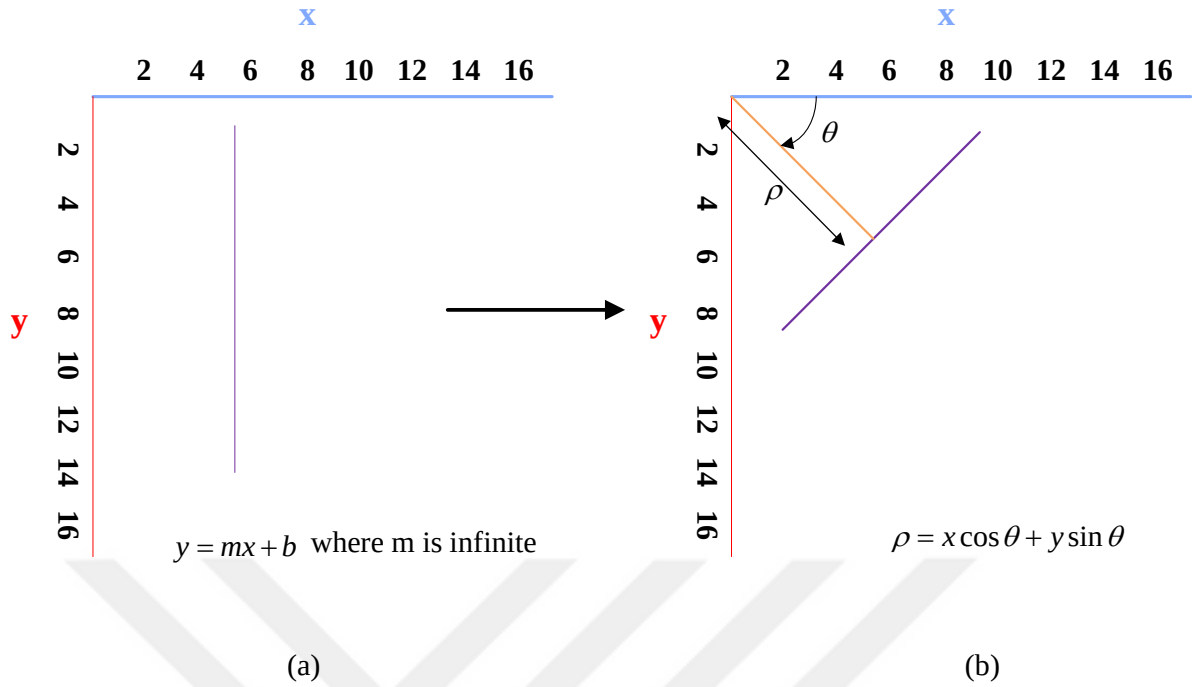


**Figure 3.57.** The relationship between Cartesian and Polar coordinates.

If we consider the examples that the point is  $(5, 2)$ , many lines can pass through this point including a vertical line. We can represent this point based on  $\rho$  and  $\theta$  that is shown in the following equation.

$$\rho = 5 \cos \theta + 2 \sin \theta \quad (3.100)$$

In the equation above, if  $\theta$  are  $0$ ,  $\frac{\pi}{2}$ , and  $\frac{\pi}{4}$  then  $\rho$  are  $5$ ,  $2$ , and  $4.9$ , respectively. The goal of all this information is that the point in polar coordinates is represented with sinusoidal in the Hough space. If the curves of different points intersect in the Hough space, then these points belong to the same line characterized by some  $\theta$  and  $\rho$ . As seen in Figure 3.58, the points in the cartesian coordinates are represented as sinusoidal curves in the Hough space. All three lines intersect at the same points characterized by a  $\theta$  value of  $0.92$  radians and perpendicular distance  $\rho$  of about  $9.66$ .

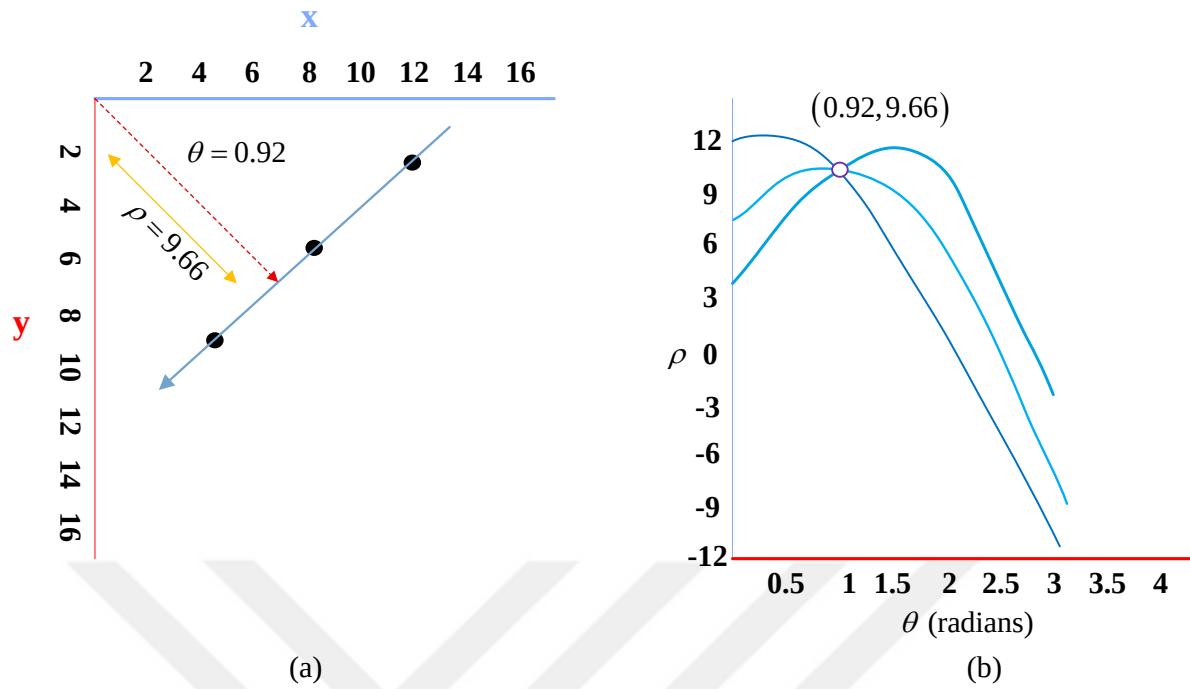


**Figure 3.58.** The intersection of the three points in (a) Polar space and (b) Hough space.

Therefore, the process shows the line that best fits the data. In addition, we can apply this concept to the grid method in the Hough space. As shown in Figure 3.59, the box that is specified by the green rectangle has the maximum number of intersections. In other words, the maximum number of intersections defines the parameters that are the best fit values of the points.

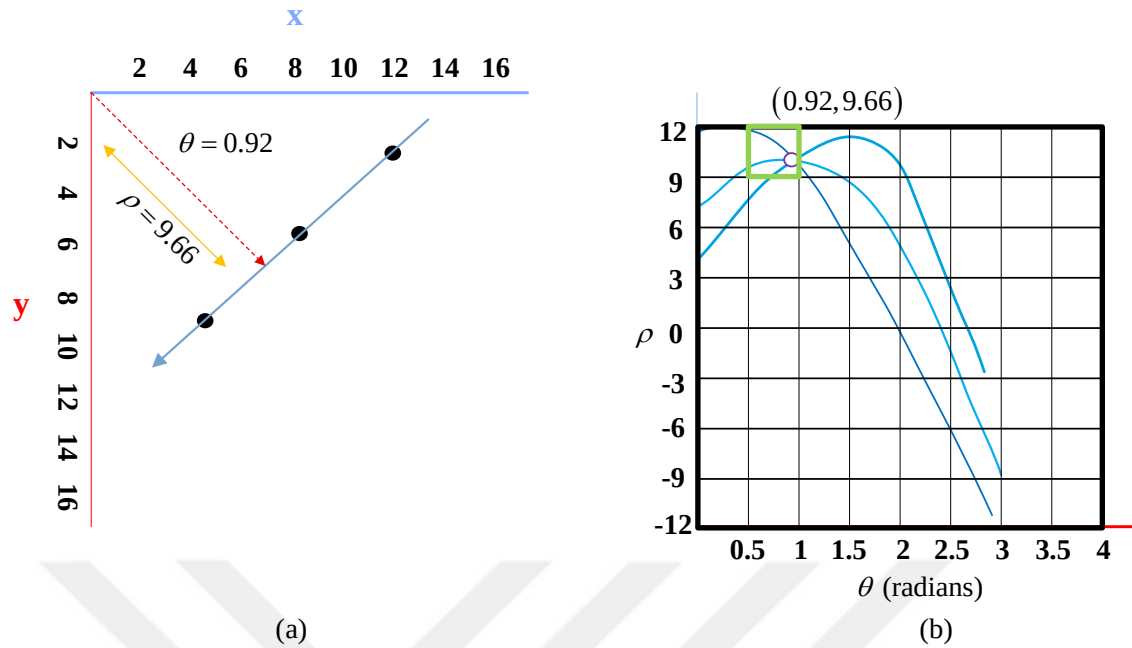


**Figure 3.59.** The intersection of the three points in (a) Polar space and (b) Hough space (in the grid method).

By using the Hough transform that is mentioned above in detail the lane lines in the gradient image are defined in the best way. It is important to specify the size of the box.  $\rho$  is the distance resolution of the accumulator in pixels and  $\theta$  is the angle resolution of the accumulator in radians. If the boxes are large, it means less precision. It is necessary to define the boxes as being sufficiently small. In addition, the boxes cannot be too small since this can also result in inaccuracies and can take longer time to run the program. The pseudo code of the Hough transform is shown in Table 3.11.

**Table 3.11.** The pseudo code of the Hough transform.

|     |                                                       |
|-----|-------------------------------------------------------|
| 1.  | apply the <b>HoughLinesP</b> to the cropped image     |
| 2.  | <b>function</b> display_lines(image, lines)           |
| 3.  | line_image is <b>zeros_like</b> lane_image            |
| 4.  | <b>if</b> lines is not empty                          |
| 5.  | <b>for</b> line in lines                              |
| 6.  | <b>shape</b> x1, y1, x2, y2 with respect to the line  |
| 7.  | apply the <b>line</b> (line_image)                    |
| 8.  | <b>endfor</b>                                         |
| 9.  | <b>endif</b>                                          |
| 10. | apply the <b>addWeighted</b> (lane_image, line_image) |
| 11. | <b>show</b> the combo_image                           |

The following figure shows the detected lines in the cropped gradient image. However, this step is not the final step of the line detection because the lines are detected in the gradient image that is the black image of the road.

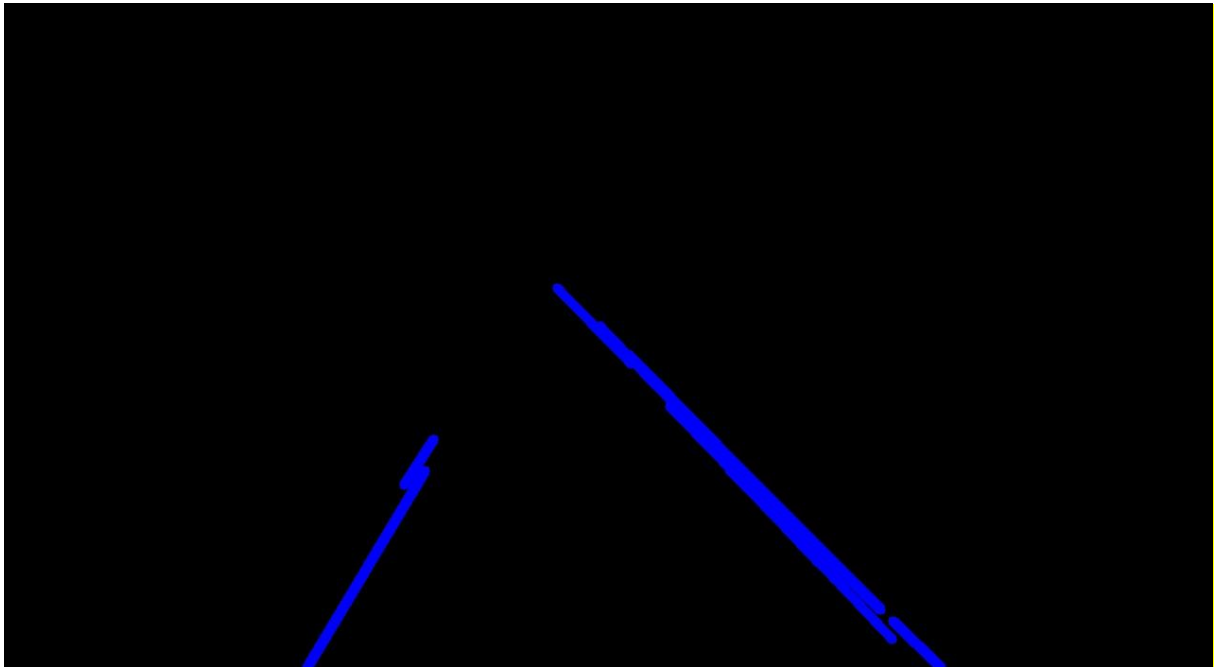


**Figure 3.60.** The lines detection of the gradient image.

The final step of the line detection is to blend the figure above into the original color image so that the lines show up on the lanes instead of the black screen. To find the final image, Figure 3.60 is added with the original image because the background of the line image is completely black. In other words, the black pixels do not affect the original image, and the detected lines in the figure above remain with a different color. The final line detection image is shown in Figure 3.61.

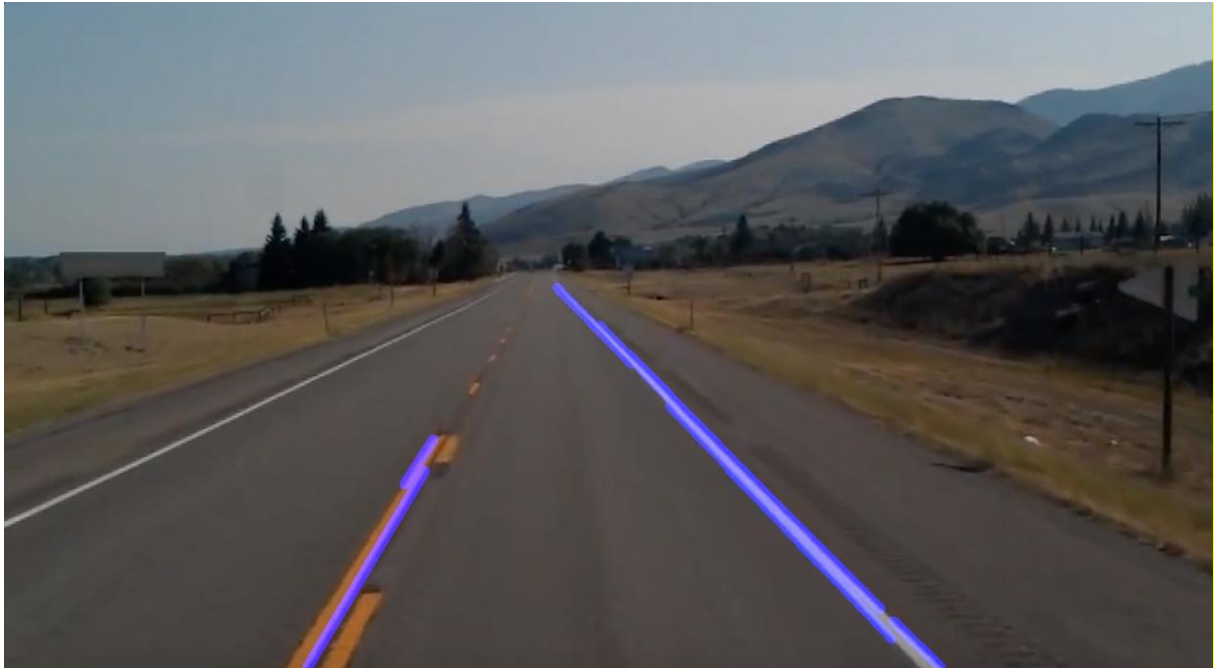


**Figure 3.61.** The blended image of the two images.

According to the figure below, the lines are displayed on top of the original image lanes. However, the left lane of the image has not been detected quite well. To avoid this error, it is important to optimize the detected lines that is discussed in Chapter 4.9.

### 3.7.2. The Vehicle Detection and Tracking System

The goal of this section is to create a pipeline to detect and track the vehicles in the image. It can be achieved by following the below tasks:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labelled training set of images and train a classifier Linear SVM.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Run pipeline on a video stream and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

These tasks are applied to detect and track the vehicles in the following sections. The algorithm is constructed in Python due to the easiness and rich libraries available for all kinds of tasks. Many libraries have been utilized to perform various tasks. Numpy is used for

importing and processing data. In addition, matplotlib is used for the visualization of the extracted colour features. Furthermore, Scikit-Learn is used to split data into train and test parts, and classifier functions are used to create the models and train them with the available data in the datasets. In the above-mentioned order, first, SVM, which is a supervised machine learning algorithm, performs the classification and regression tasks of the data set consisting of the vehicles and non-vehicles images. The working principle of the SVM algorithm is specified in the following section.

### 3.7.2.1. Support Vector Machines (SVM)

The SVM is a supervised machine learning model that uses classification algorithms for two-group classification problems. They are able to classify new text after providing the SVM model with a training data set that has been tagged for each category. They have two main advantages: higher speed and better performance with a limited number of samples (in the thousands). This makes the algorithm very suitable for text classification problems, where it's common to have access to a dataset of at most a couple of thousand tagged samples.

A simple example is the easiest way to understand the fundamentals of SVM and how it works. Consider the following scenario: we have two tags, one each of red and blue, and our data contains two features, one each of x and y. Labelled training data is placed as seen in the figure below.

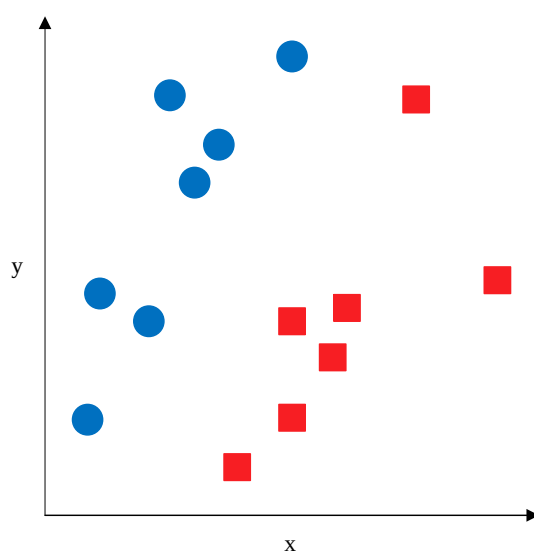


**Figure 3.62.** The labelled data.

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions is simply a line) that best separates the tags. This line is the decision boundary; anything that falls on one side of it we will classify as blue, and anything that falls on the other as red. As seen in the figure below, the best hyperplane is drawn for the SVM.

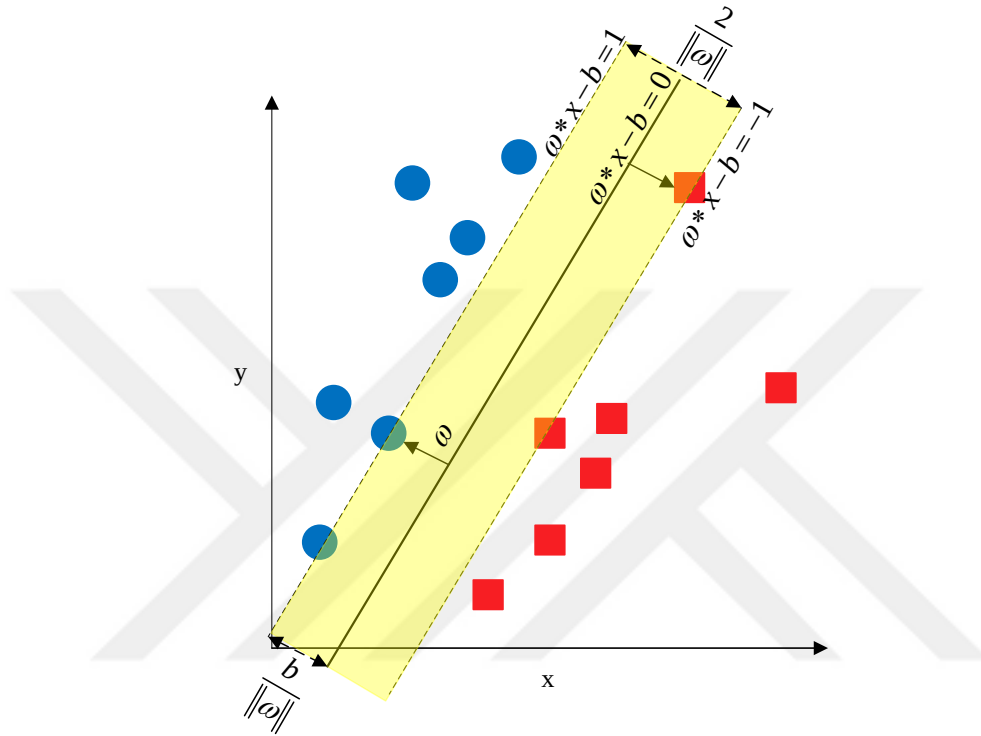


**Figure 3.63.** The simply a line for the best hyperplane.

However, more specifically, what is the best hyperplane? It is the one that optimizes the margins from both tags in the case of SVM. The hyperplane (remember it's a line in this instance) with the greatest distance to the closest element of each tag is known as the maximum distance hyperplane. If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the "margin" and the maximum-margin hyperplane is the hyperplane that lies halfway between them. With a normalized or standardized dataset, these hyperplanes can be described by the following equations:

$$\begin{aligned}\omega^T x - b &= 1 \text{ (label 1)} \\ \omega^T x - b &= -1 \text{ (label -1)}\end{aligned}\tag{3.101}$$

Geometrically, the distance between these two hyperplanes is  $\frac{2}{\|\omega\|}$ , so to maximize the distance between the planes, we want to minimize  $\|\omega\|$ . The distance is computed using the distance from a point to a plane equation. We also have to prevent data points from falling into the margin. We add the following constraint: for each  $i$  either

$$\begin{aligned}\omega^T x - b &= 1, \text{ if } y_i = 1 \\ \omega^T x - b &= -1, \text{ if } y_i = -1\end{aligned}\tag{3.102}$$

These constraints state that each data point must lie on the correct side of the margin. The equation (3.102) can be rewritten as

$$y_i (\omega^T x_i - b) \geq 1, \text{ for all } 1 \leq i \leq n\tag{3.103}$$

We can put this together to get the optimization problem that is shown in the equation below.

$$\text{Minimize } \|\omega\| \text{ subject to } y_i (\omega^T x_i - b) \geq 1, \text{ for } i=1, \dots, n\tag{3.104}$$

The  $\omega$  and  $b$  that solve this problem determine our classifier,  $x \rightarrow \text{sgn}(\omega^T x - b)$  where  $\text{sgn}(\cdot)$  is the sign function. An important consequence of this geometric description is that the max-margin hyperplane is completely determined by those  $\vec{x}_i$  that lie nearest it. These  $x_i$  are called support vectors.

### 3.7.2.2. Datasets of the Algorithm

The labelled dataset used for this algorithm has been downloaded from the Udacity and KITTI websites. These sample images are compiled from a variety of sources, including the GTI vehicle image collection, the KITTI vision benchmark suite, and examples taken directly from the project video. When the datasets downloaded from these sites are examined, there are 8782 vehicle images and 8975 non-vehicle images in total. Furthermore, the dimensions of the images are (64,64,3) and the data type is float32. The following figure shows that an example

from each of the classes is visualized. The examples from each of the classes are shown in Figure 3.64.

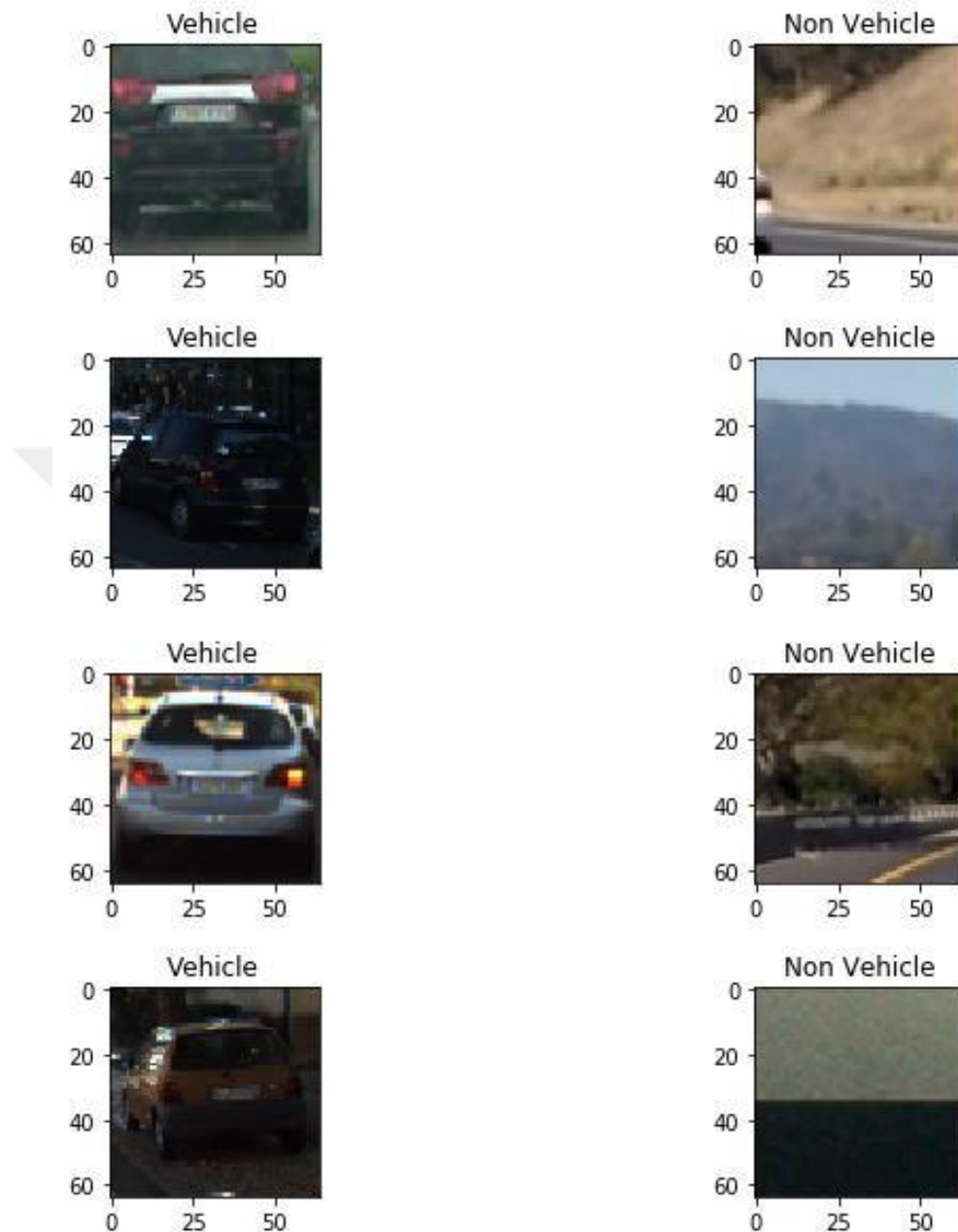


**Figure 3.64.** Vehicle and non-vehicle images.

Based on the information above, we may infer that the dataset is reasonably balanced since it comprises about equal quantities of vehicle and non-vehicle images. The dataset includes sequences of photographs in which the target item (in this example, vehicles) appears essentially the same over a full series of images. This is the most significant observation obtained. In such a case, performing a randomized train-test split will be subject to overfitting because images in the training set may be nearly identical to images in the test set. In order to

deal with the time-series problem, 90% of the first half of each class dataset (vehicle and non-vehicle) is set aside for training the classifier, with the remaining 10% being used as the test set.

### 3.7.2.3. Feature Extraction

In order to be able to successfully classify vehicles in an image, a set of useful features is needed over which a classifier could be trained. For this algorithm, the following features are used.

#### a. Spatially Binned Raw Colour Values

In the field of classification, it is well recognized that raw pixel intensity data might be beneficial in some situations. However, including the three colour channels of a full-resolution image might be time-consuming. As a result, we apply spatial binning to an image, which enables us to preserve enough valuable information to aid in the identification of cars while simultaneously lowering the number of pixels in the image. For our purposes, OpenCV's `cv2.resize()` command is used to scale down the resolution of an image. An example of this can be shown in the following figure.



**Figure 3.65.** Scaling down the resolution of the image by using OpenCV.

#### b. Histogram of the Colour Values

Despite the fact that the raw colour values offer us important information, the distribution of colour values is added in order to be able to remain resistant to changes in object appearance. This reduces our reliance on the object's structure and enables the matching of changes in aspects and orientations to be accomplished. A histogram is an exact portrayal of the

dissemination of numerical information. A histogram is simply a graphical representation of the number of pixels in the image that fall within a certain range, either luminance or colour. For example, for a normal luminance histogram, the graph shows the number of pixels for each luminance or brightness level from black to white. The higher the peak on the graph, the more pixels are at that luminance level. With a colour histogram, the principle is the same, but instead of seeing the levels of black graphed, the number of pixels for each of the three main colours is shown. The colour features and histogram for vehicles and non-vehicles are presented in Figure 3.66 and Figure 3.67.

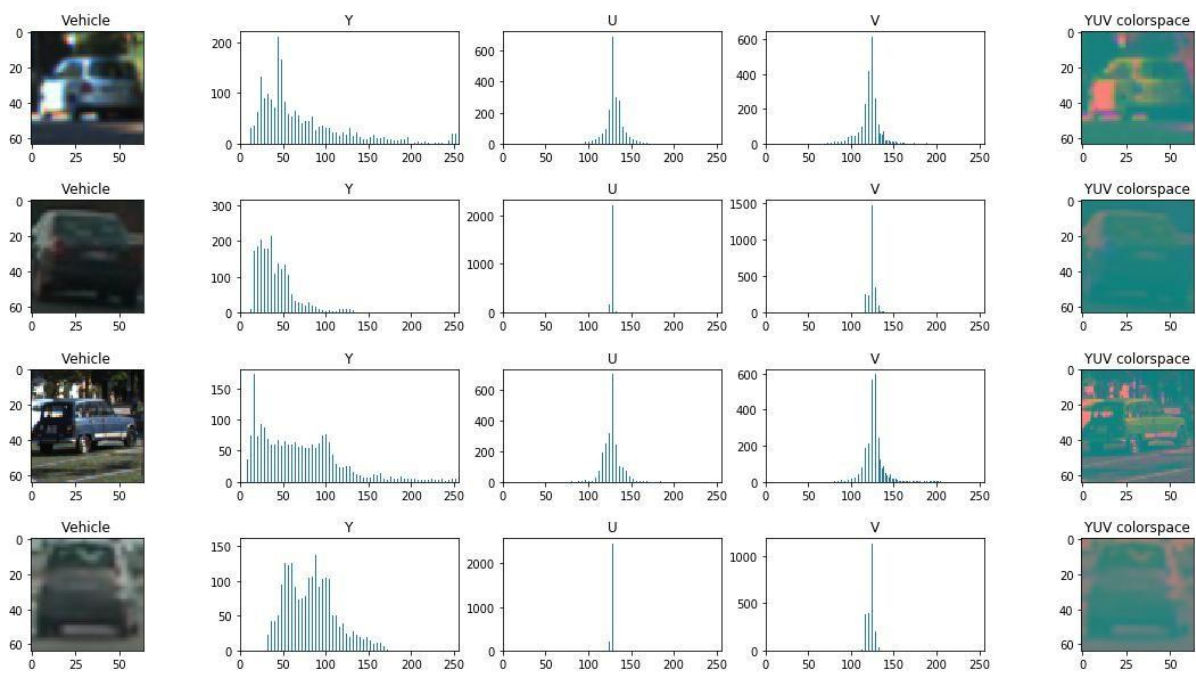


**Figure 3.66.** The histogram and colour feature of the vehicles.

The extraction of the colour feature is constructed in YUV colour space. It is a colour encoding system typically used as part of a colour image pipeline. The YUV model defines a colour space in terms of one luma (it represents the brightness in the image) component (Y) and two chrominance (it is the signal used in the video systems to convey the colour information of the image) components, called U (blue projection) and V (red projection).

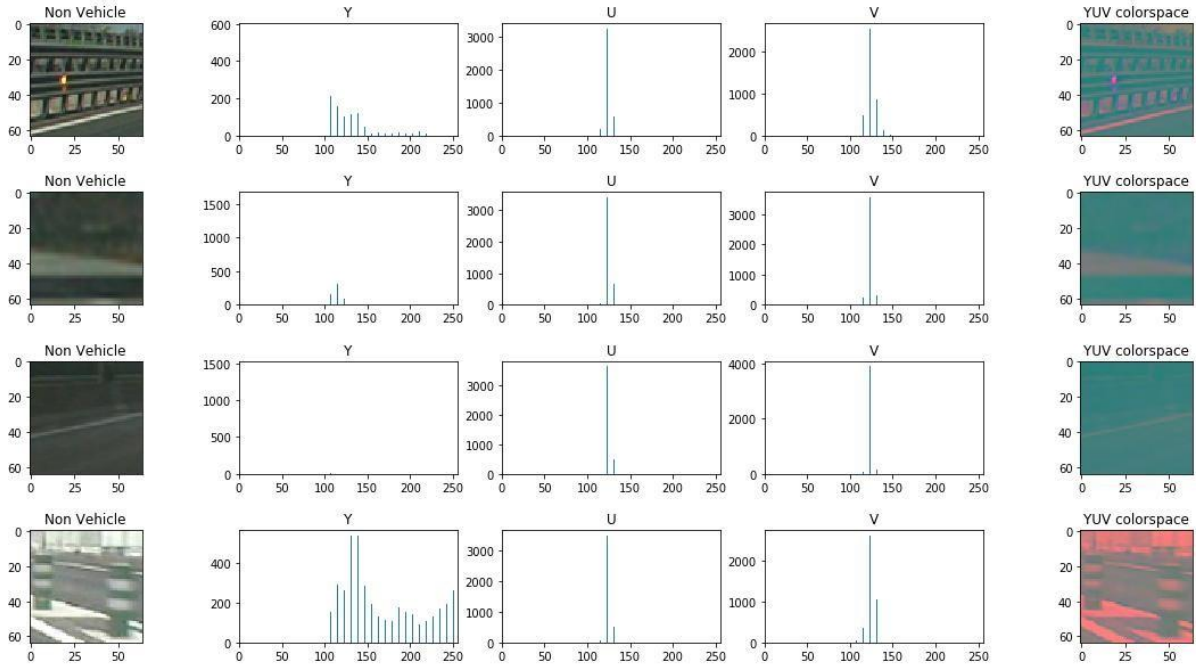


**Figure 3.67.** The histogram and colour feature of the non-vehicles.

In addition to the colour information, knowledge of the shape of the object is also considered useful in performing classification. As a result, the HOG features are appended to the overall feature vector. The HOG is chosen due to its robustness to noise over simply including per pixel gradient information.

### c. Histogram of Oriented Gradients (HOG)

The HOG is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) is used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes), and it is known that edges and corners pack in a lot more information about object shape than flat regions. In addition, the working principle of the HOG is the same as the Canny Edge Detector that has been used to detect the lines in the previous section. However, the HOG descriptor focuses on the structure or shape of an object. It is better than any edge descriptor as it uses magnitude as well as the

angle of the gradient to compute the features. For the regions of the image, it generates the histograms using the magnitude and orientation of the gradient. The result of the HOG extraction of the vehicles and non-vehicles is shown in Figure 3.68.

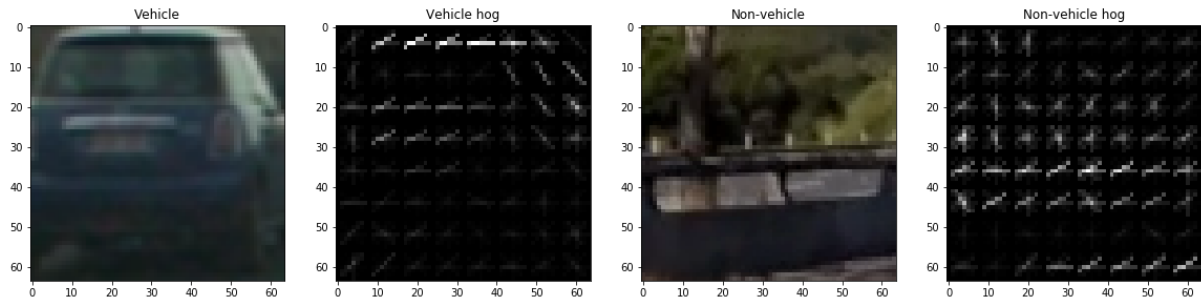


**Figure 3.68.** The HOG features extraction from the vehicle and non-vehicle.

The following stage is data preparation. In order to train any algorithm, we must prepare the data by cleaning it, scaling it, and so on. Due to the fact that this is an image dataset, there is no need for any kind of clean-up. Therefore, in this step, data splitting needs to be done. With the help of the `train_test_split()` function from the Scikit-Learn library, data is split into two parts: training and testing. They are divided into two categories: training and testing. In general, 90% is used for training and 10% is used for testing. Aside from that, the information must be standardized. To normalize the data, a standard scaling function, `StandardScaler()`, has been implemented in the SVM classifier procedure. In the above-mentioned, it has been shown that SVM works well with the HOG features. Therefore, an off-the-shelf Linear SVM from scikit-learn is used as a starting point. The parameters of the Linear SVM and some of their corresponding values that are being experimented with are included in the following table.

**Table 3.12.** The parameters of the Linear SVM.

|                                       |                      |
|---------------------------------------|----------------------|
| <b>Colour space</b>                   | HSV, HSL, YUV, YCrCb |
| <b>Spatial bin size</b>               | 32, 24, 16           |
| <b>Number of the histogram bins</b>   | 32, 24               |
| <b>Number of the orientation bins</b> | 12, 9, 6             |
| <b>Pixels per cell</b>                | 8                    |
| <b>Cells per block</b>                | 2                    |

The final parameters that are chosen are listed in the table below.

**Table 3.13.**The parameters of the Linear SVM.

| Parameters       | Values   |
|------------------|----------|
| Colour space     | YCrCb    |
| Spatial bin size | (16, 16) |
| Histogram bins   | 24       |
| Orientation bins | 9        |
| Pixels per cell  | 8        |
| Cells per block  | 2        |

After all of these steps have been completed, the SVM classifier has been trained and tested. The results acquired via the use of the algorithms are provided in Chapter 4.10.

#### **3.7.2.4. Sliding Window Technique**

After having successfully extracted the features and trained a linear SVM, a sliding window technique is implemented for searching for the vehicles in an image. Given that computing HOG features can be expensive, a more efficient method for doing the sliding window search has been implemented over its naive counterpart. In this approach, the HOG features are calculated just once for the whole image, rather than many times. Afterwards, the HOG feature vector is sub-sampled for each of the distinct multi-scaled windows. This implementation allows for a faster search for vehicles across an image or video frame, and the use of multi-scale windows makes detecting the target object from different aspects easier.

## 4. RESULTS AND DISCUSSION

In this chapter, the results of Chapter 3, which include the modelling of the battery, the modelling of BMS for EVs, the modelling of the EVs, the DR estimation of the vehicle, and intelligent control methods for the AVs, are discussed in detail.

### 4.1. The Simulation Results of the Battery in the Simulink

Three different types of inputs (step, pulse, and sinusoidal) have been used in the system in order to assess the battery model that has been created in Simulink. The outputs of the battery (SoC, current, and voltage) obtained as a result of these applied inputs have been analyzed.

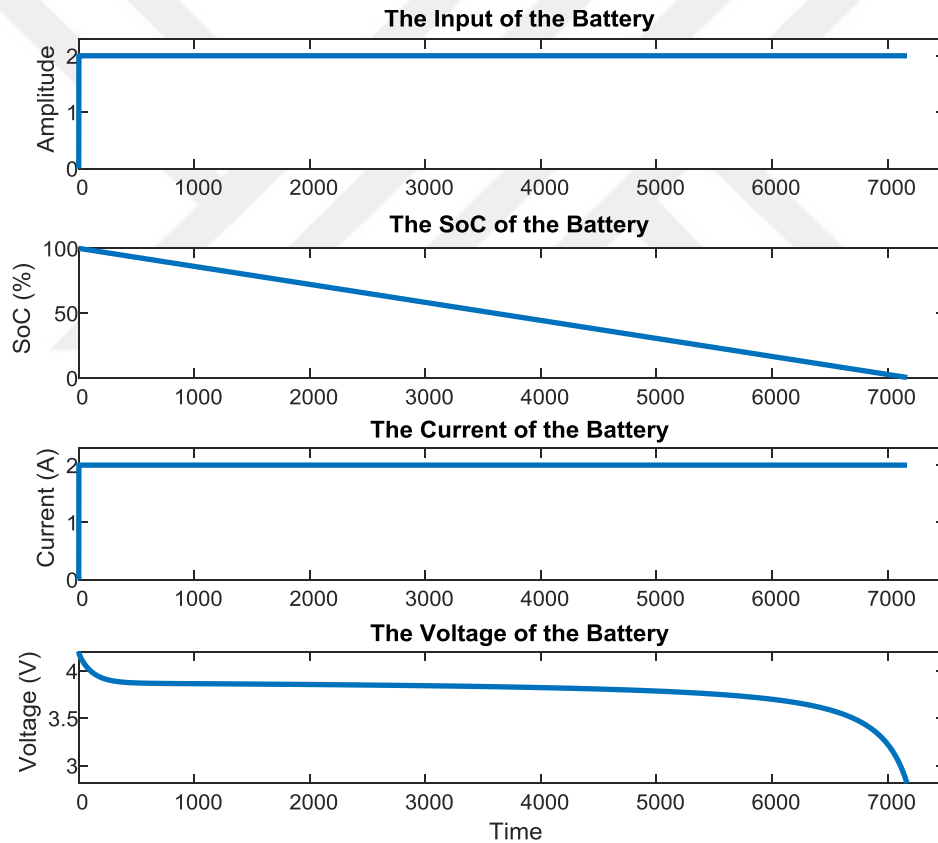


**Figure 4.1.** The input and outputs of the Battery with step input.

After a specific amount of time has elapsed, the SoC of the battery reduces to zero, as seen in the figure above when step is used as an input to the model. Also, it is seen that the input value is the same as the current drained from the battery. In contrast, when we look at the voltage value of the battery, we can observe that it declines between the highest and lowest voltage values, similar to the SoC.

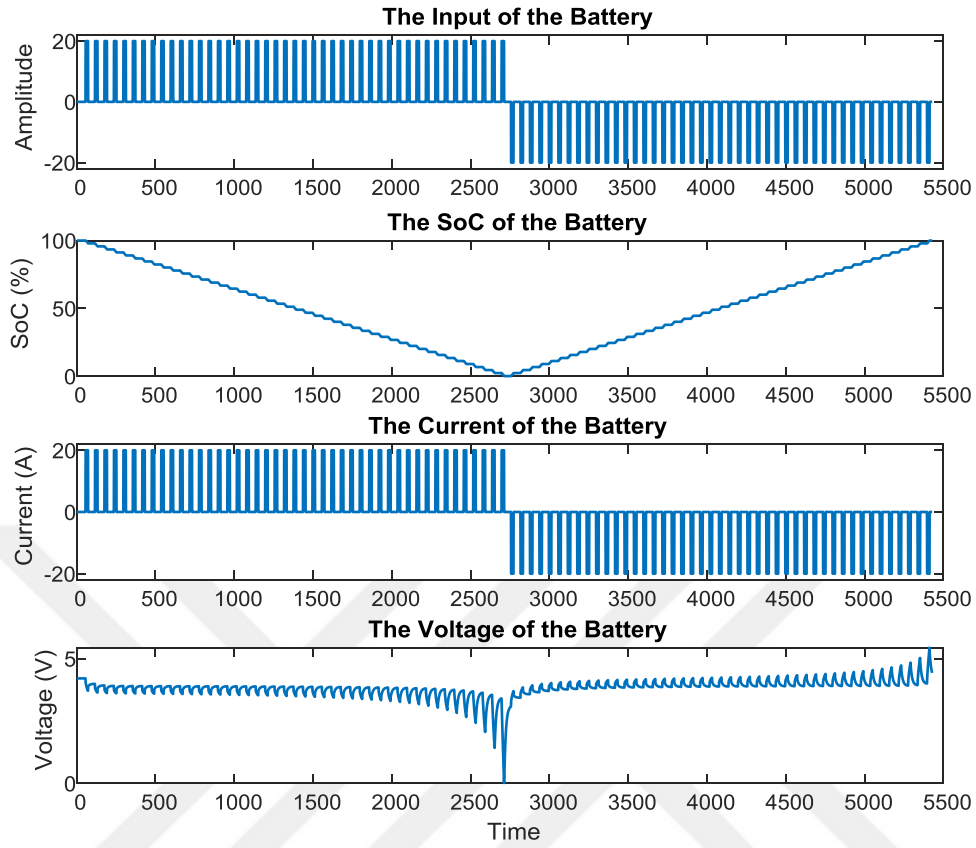


**Figure 4.2.** The input and outputs of the Battery with pulse inputs.

The SoC of the battery has been decreased and increased in the pulsed regions if we used an input consisting of numerous pulses (positive and negative) instead of the step input, as seen in the figure above. In the positive pulse zone, the SoC lowers to zero within a certain amount of time after the pulse has been received. The SoC is gradually increasing as one moves towards the negative pulse area. As with the step input, when pulse input is used, the current value of the battery and the input value are in the same form. Furthermore, the voltage output of the battery drops in the areas where the pulse is positive and rises in the regions where the pulse is negative.

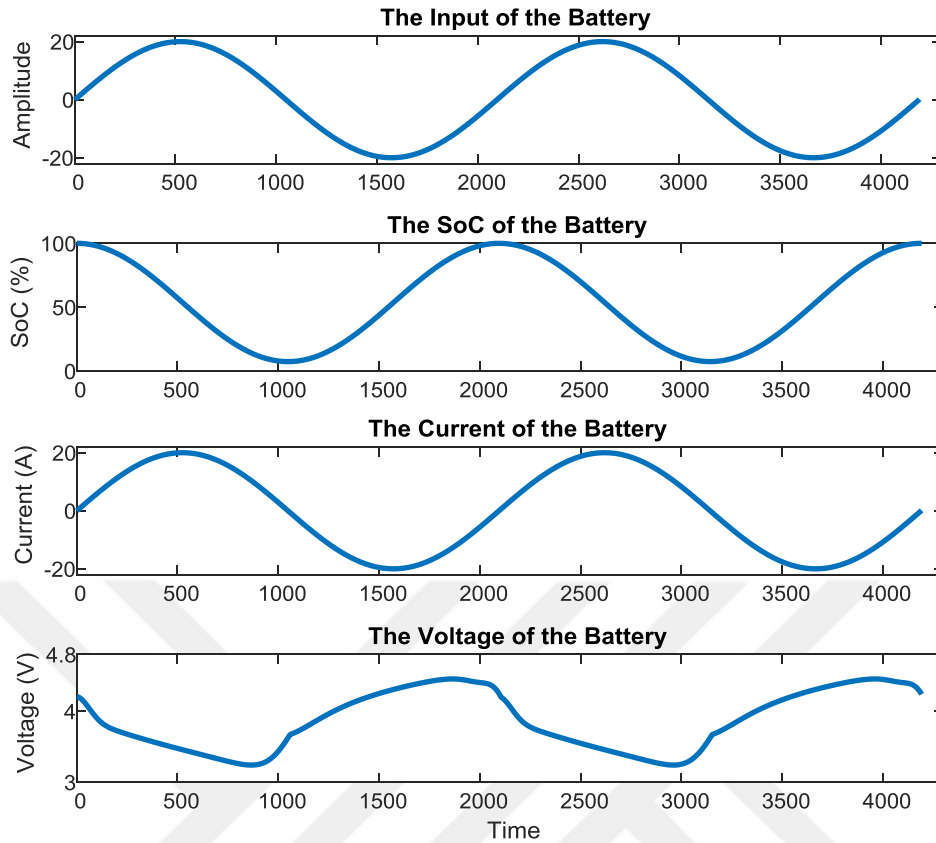


**Figure 4.3.** The input and outputs of the Battery with sine input.

Another input we use in this part, where we will analyze the outputs of the battery, is the sine wave. As shown in Figure 4.3, when the simulation's input is sinusoidal, the SoC reduces and grows in sine format, as seen in the graph. The voltage value of the battery, on the other hand, does not follow a sinusoidal pattern. The reason for this is that the characteristics of the voltage change of the battery are as in Figure 3.3.

#### 4.2. The Simulation Results of the Battery in the MapleSim

The battery, one of the most critical components of the vehicle, is modeled in this section using the MapleSim modeling software. As can be seen in Figure 4.4, the battery model is based on The Li-on battery type, which is common nowadays. In addition, the parameters of this model are the same as those of the model constructed using Simulink software in the previous section.

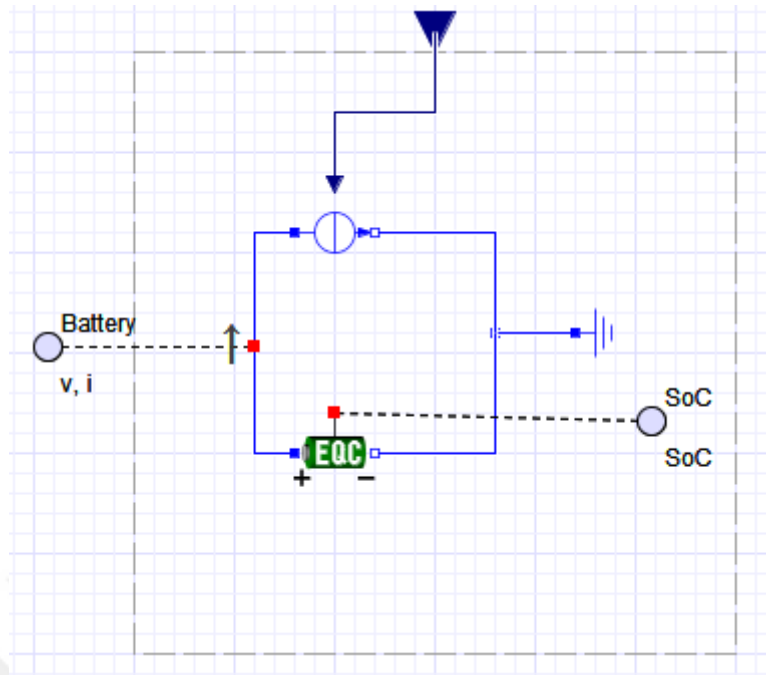


**Figure 4.4.** The modelling of the battery in the MapleSim.

Three different inputs (step, pulse, and sinusoidal) are used to analyze the battery modelled in the figure above. Because of the inputs that have been supplied to the system, the outputs of the battery that are SoC, current, and voltage have been acquired.

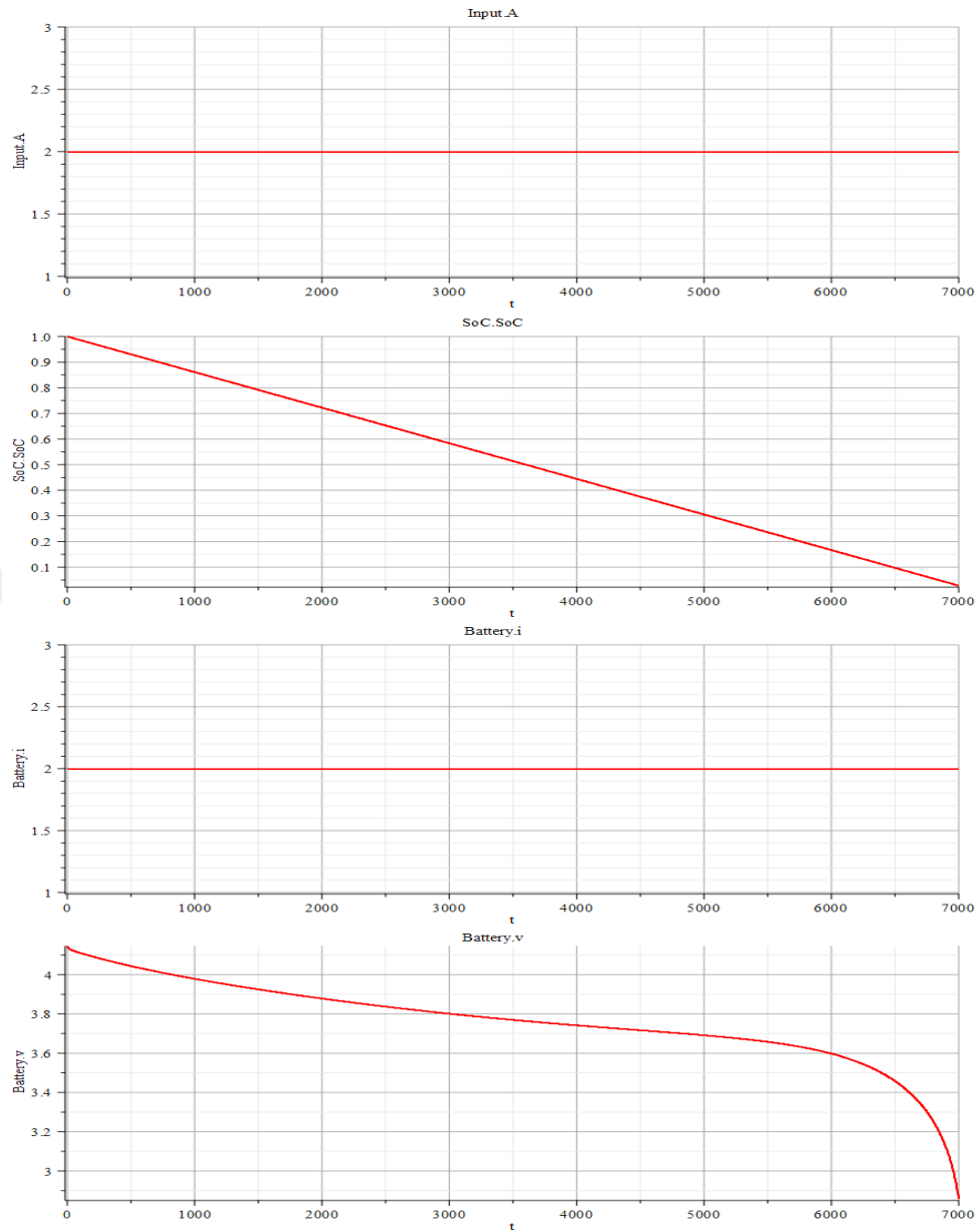


**Figure 4.5.** The input and outputs of the battery with step input.

Firstly, step input is applied to the battery model. As seen in Figure 4.5, after a certain period of time, the SoC of the battery depletes to zero. In addition, since the input value applied to the system is the current drawn from the battery, the battery and the input graph are the same. When we look at the voltage graph of the battery, it has been observed that the battery is the same as the discharge graph. In other words, the voltage of the battery drops from the highest value to the lowest value.

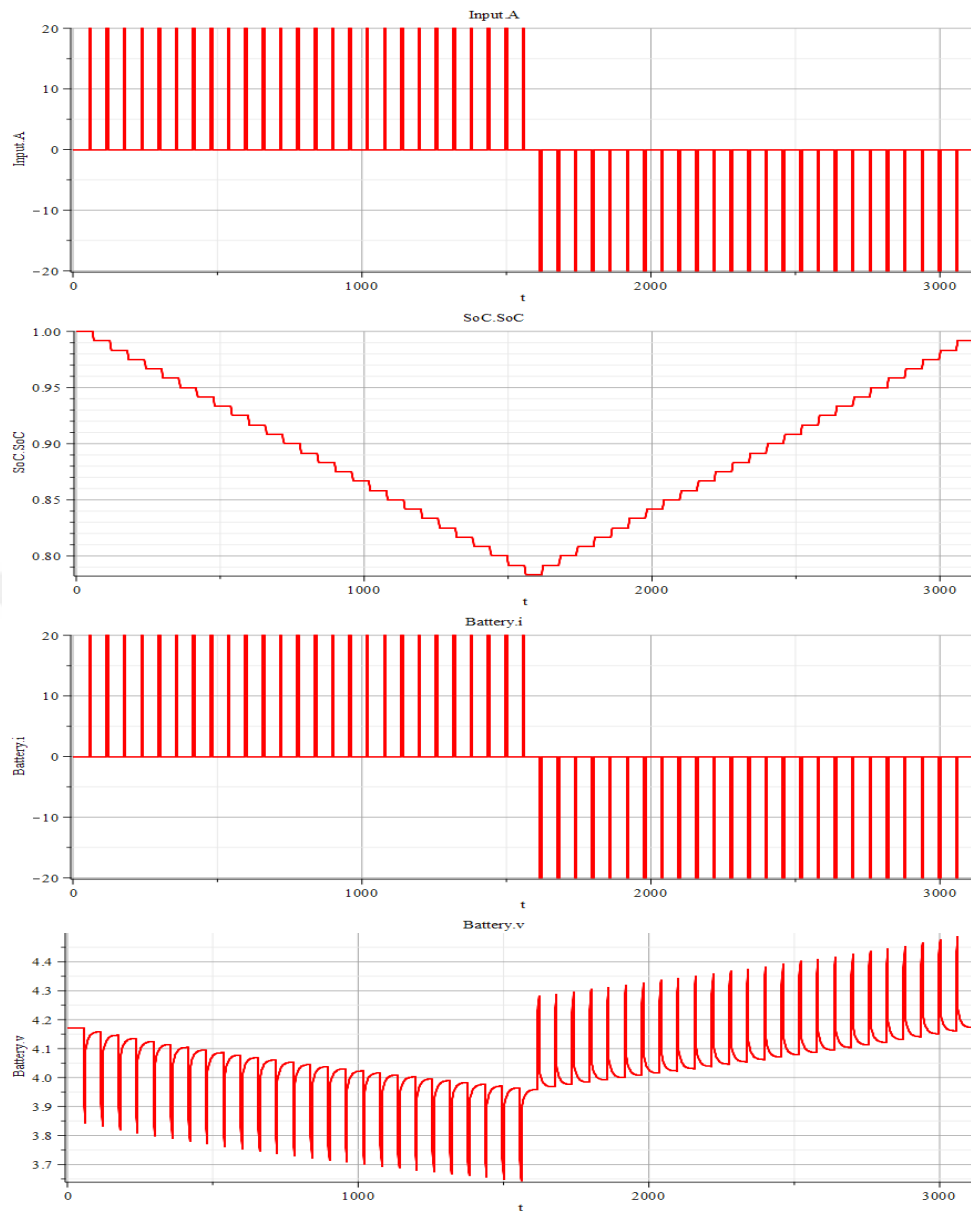


**Figure 4.6.** The input and outputs of the battery with pulse input.

Secondly, an input consisting of many pulses is applied to the model of the battery. As seen in Figure 4.6, the SoC, which is the first output of the system, is decreased by zero when the pulses are positive. Furthermore, the SoC is completely charged in the part where the pulses are negative. As with the other input, the current drawn from the battery is the same as the input value applied to the system. In addition, the voltage of the battery is completely different from the discharge graph, as opposed to the result in the other input. The voltage value decreases in the regions where the pulses applied to the battery are positive and increases in the regions where the pulses are negative.

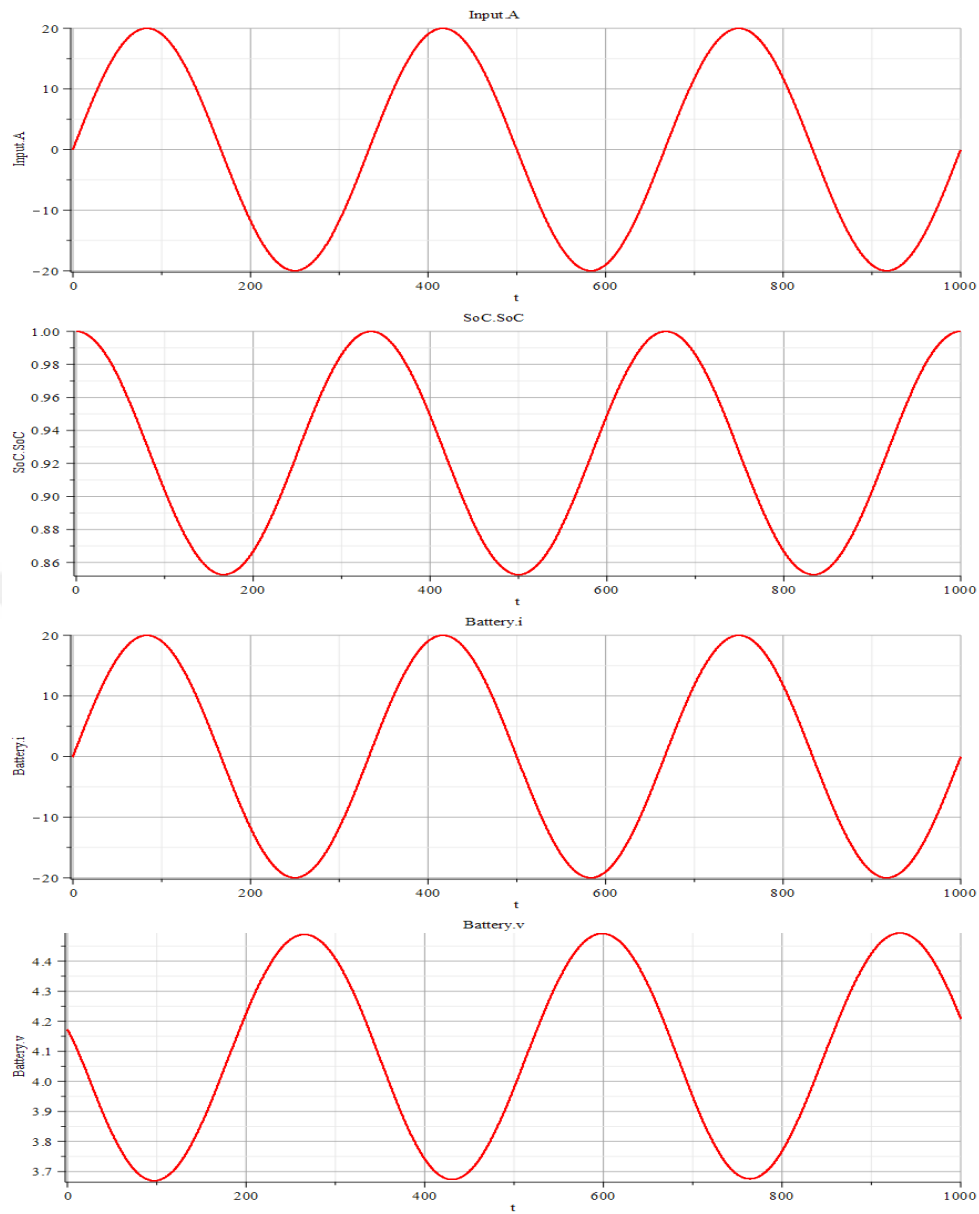


**Figure 4.7.** The input and outputs of the battery with sine input.

The third input assigned to the system is sinusoidal. As seen in Figure 4.7, if our input fluctuates sinusoidally, the capacity of the battery is also discharged in the SoC, representing the capacity. While the current value of the battery is the same as the input value, the voltage value of the battery is the inverse of the input value according to the x-axis. The reason for this is that the current drawn from the battery is inversely proportional to the voltage. The outputs acquired by using the Simulink software in the previous section are compared to those obtained by using the MapleSim software in this section, and it has been observed that the outputs generated by using three distinct inputs are identical.

### 4.3. The Simulation Results of the Battery in the Matlab

The first-order differential equation for the battery, which was found in the previous part, is discussed in detail in this section. It is possible to determine the SoC of the battery by applying step input to the battery model that has been constructed in the Matlab environment.  $\alpha$  value is the single most critical parameter that influences these SoCs. Whenever  $\alpha$  the alpha value is high, the charge value in the battery drops in the opposite direction to the alpha value. The equation (3.8) shows that alpha has an effect on the system's gain, and that the gain is proportional to how much charge is stored in the battery.

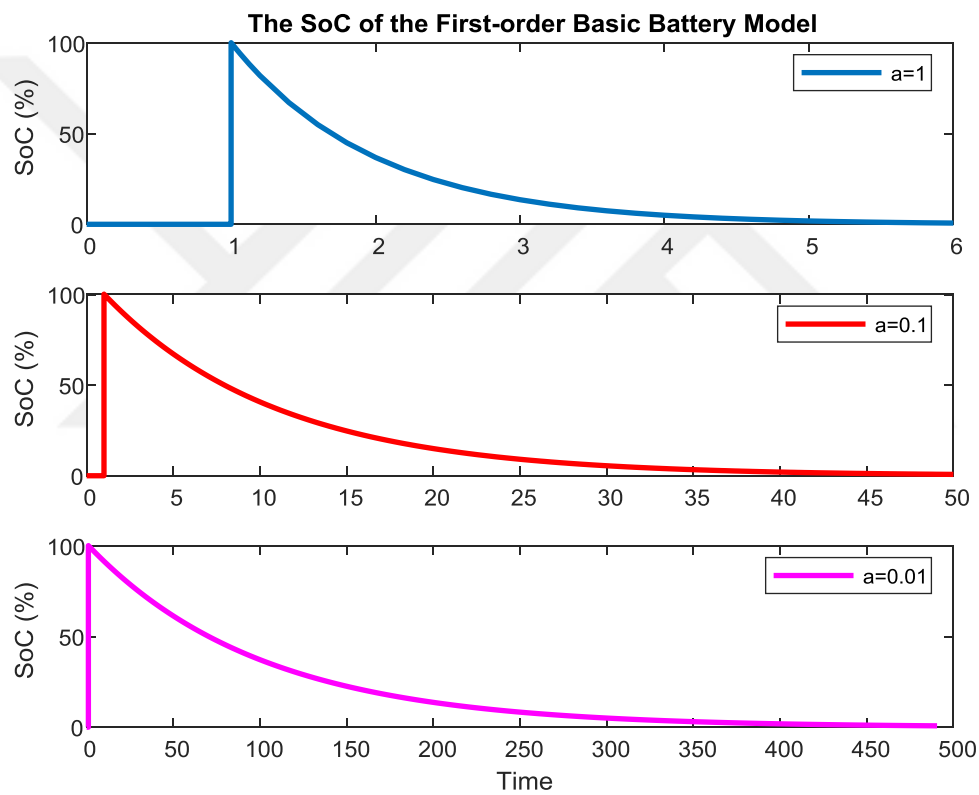


**Figure 4.8.** The comparison of the SoC with different  $\alpha$ .

In order to demonstrate that the alpha value has an effect on the battery, the equation (3.8) is examined in the Matlab environment. As shown in Figure 4.8, the alpha value was given three distinct values (1, 0.1, and 0.01), and the SoCs of the batteries were created by applying a step input to the model's input. Based on these findings, it can be concluded that as the alpha value lowers, the charge in the battery rises. As can be seen in the graph above, the battery's discharge time rises progressively as the charge value of the battery grows.

#### 4.4. The Simulation Results of the BMS in the Matlab/Simulink

When it comes to energy storage systems in airplanes, electric cars, portable gadgets, and other equipment that requires a dependable, high-density, low-weight power source, lithium-ion battery packs are the most popular choice. The BMS is in charge of ensuring that the battery operates safely, performs well, and has a long life under a variety of charge-discharge and environmental situations. When developing a BMS, engineers create feedback and supervisory control systems that do the following:

- Monitors the cell voltages
- Estimates the SoC
- Control the charging profile
- Balances the SoC of individual cells

In this subsection, it is aimed to obtain the above-mentioned items for the vehicle's battery with the help of developed algorithms. The BMS model of which the model has been created for the vehicle is shown in the figure below.

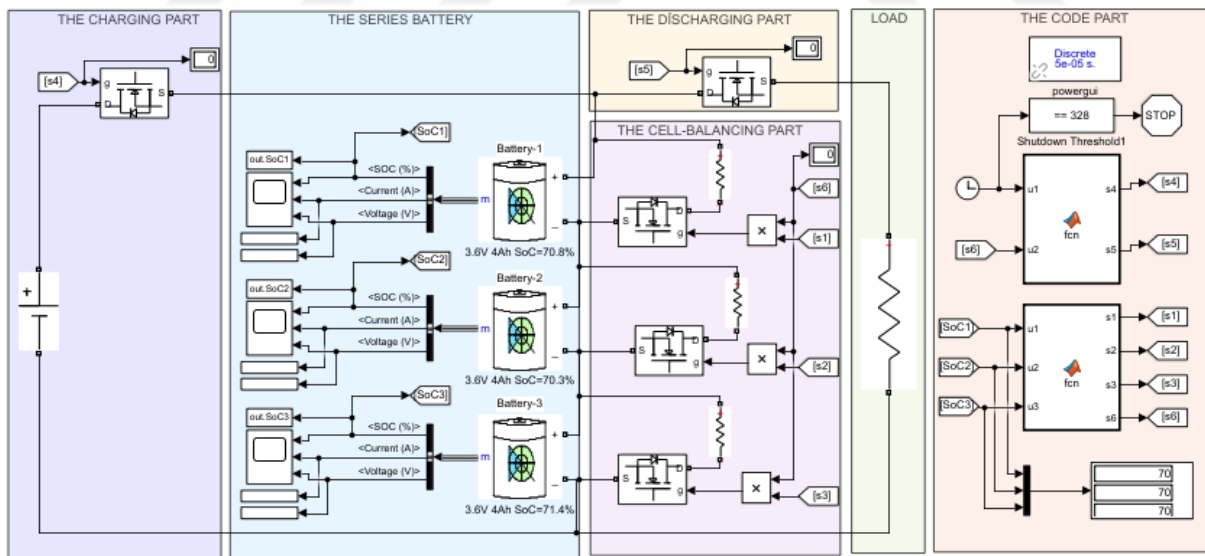


**Figure 4.9.** The Modelling of the BMS.

According to the model as shown in Figure 4.9, the BMS system is separated into 6 sections: the charging section, the battery section, the discharge section, the cell balancing section, the load section, and the code section. While constructing the model, three batteries are linked in series to provide power. These batteries have three different SoCs (70.8, 70.3, and 71.4),

which are distinct from one another. The first purpose of BMS is that if the SoCs of the batteries are different from each other, the SoC values of the batteries are equalized using the synchronization algorithm between the SoCs of the batteries. The synchronization algorithm can be specified as pseudo code in Table 4.1.

**Table 4.1.** The pseudo code of the synchronization algorithm.

|                                                                              |
|------------------------------------------------------------------------------|
| 1. <b>Input:</b> Battery SoCs: $SoC1$ , $SoC2$ , and $SoC3$ .                |
| 2. <b>Output:</b> Switches ( $s1$ , $s2$ , $s3$ , and $s6$ ) for the Mosfets |
| 3. <b>If</b> ( $SoC1 > SoC2$ or $SoC1 > SoC3$ ) <b>then</b>                  |
| 4. $s1$ is high                                                              |
| 5. <b>else</b>                                                               |
| 6. $s1$ is low                                                               |
| 7. <b>end if</b>                                                             |
| 8. <b>If</b> ( $SoC2 > SoC1$ or $SoC2 > SoC3$ ) <b>then</b>                  |
| 9. $s2$ is high                                                              |
| 10. <b>else</b>                                                              |
| 11. $s2$ is low                                                              |
| 12. <b>end if</b>                                                            |
| 13. <b>If</b> ( $SoC3 > SoC1$ or $SoC3 > SoC2$ ) <b>then</b>                 |
| 14. $s3$ is high                                                             |
| 15. <b>else</b>                                                              |
| 16. $s3$ is low                                                              |
| 17. <b>end if</b>                                                            |
| 18. <b>If</b> ( $SoC1 = SoC2$ and $SoC2 = SoC3$ ) <b>then</b>                |
| 19. $s6$ is low                                                              |
| 20. <b>else</b>                                                              |
| 21. $s6$ is high                                                             |
| 22. <b>end if</b>                                                            |

Following the equalization of the SoCs of the batteries, it is necessary to ascertain the state of charge of the vehicle's battery. If the vehicle's battery needs to be recharged, the algorithm in the code section creates a command to charge it. The algorithm also generates the command when the battery has been properly charged, and the switch ( $s4$ ) cuts off the current flowing from the charging section when the battery is sufficiently charged. The control algorithm of the charging profile can be specified as the pseudo code in Table 4.2.

**Table 4.2.**The pseudo code of the control algorithm of the charging profile.

|     |                                                                     |
|-----|---------------------------------------------------------------------|
| 1.  | <b>Input:</b> Battery SoC and Switch s6 (closed the cell balancing) |
| 2.  | <b>Output:</b> Switches s4 for the Mosfets in the charging section  |
| 3.  | <b>If</b> SoC is less than minimum SoC and s6 is low <b>then</b>    |
| 4.  | s4 is high                                                          |
| 5.  | <b>else</b>                                                         |
| 6.  | s4 is low                                                           |
| 7.  | <b>end if</b>                                                       |
| 8.  | <b>If</b> SoC is equal to desired SoC <b>then</b>                   |
| 9.  | s4 is low                                                           |
| 10. | <b>end if</b>                                                       |

During the BMS simulation study, the battery charged by the control algorithm should also be taken into consideration for discharge. In order for the battery to discharge in the BMS simulation, the switch (s5) must be closed, as illustrated in Figure 4.9. This switch causes the battery to enter a discharge condition if the signal it receives is high (that means 1). Thus, all possibilities that may occur in the battery have been examined. Figure 4.10 depicts the cell SoCs, currents, and voltages of the battery, which are the outputs to be received from the BMS simulation and which are the outputs of the BMS simulation.

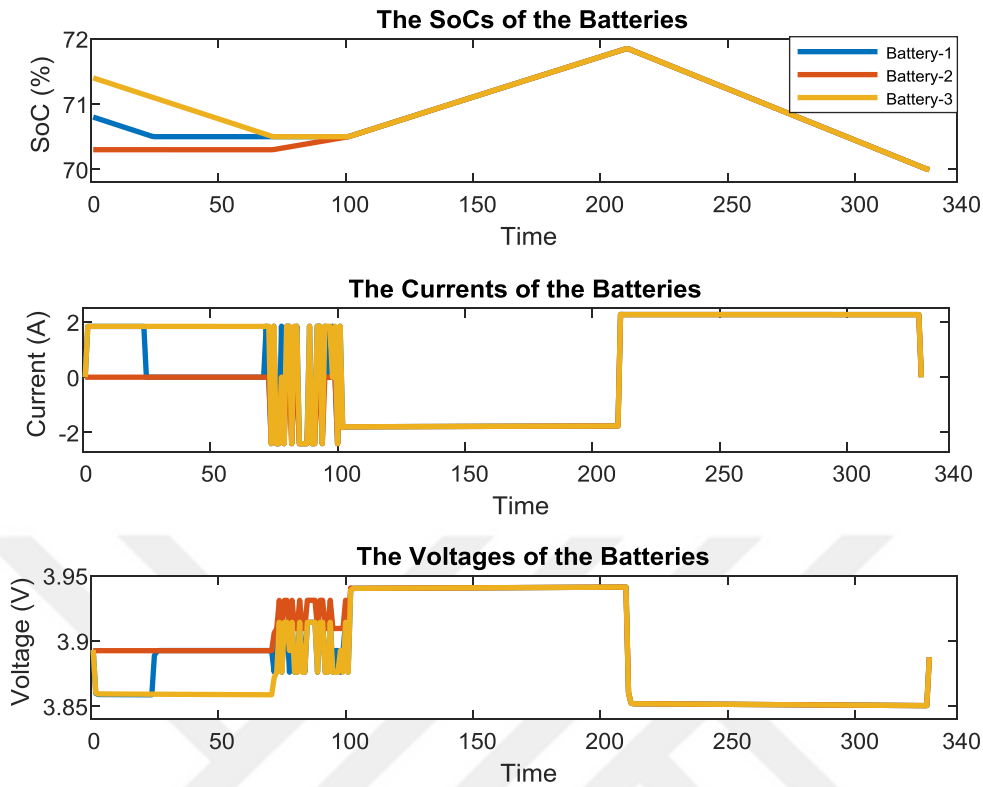


**Figure 4.10.** The outputs of the BMS simulation.

According to the simulation results shown in Figure 4.10, the first task of BMS is to balance the SoC differences between the cells, if any exist between them. It follows that the voltages of the cells will change if the SoCs of the cells are different. Therefore, the cells with high voltage will be overloaded, and the SoH of these cells will fall more than the SoH of other cells as a result of this overload. Because the discharge duration of the cell with falling SoH will be shorter, the overall performance of the battery will suffer as a result. In order to avoid this circumstance, the SoCs of the cells must be balanced prior to the cells being charged. In the simulation, the SoCs of the cells are 70.8, 70.3, and 71.4, respectively. These values of cells have been met at 70.5 after balancing. Aside from that, the battery is charged and drained up to the SoC value (72 and 70 in the simulation), which is calculated by the control algorithm when this procedure is completed.

#### 4.5. The Results of the EVs Modelling in the Matlab

The results of the linear and nonlinear modelling of the EVs and the modelled EVs with DC servo motor have been discussed in Section 3.3. In addition, the dynamics of the vehicle and control approaches with respect to the different environments have been implemented.

#### 4.6. The Simulation Results of the EVs in the Simulink

It is the goal of the simulation research, which is the result of modelling the EV using the Simulink program, to gather input information for the algorithm that will be utilized in the following sections of the dissertation. In this input information, the output information that has been acquired as a consequence of the simulation of the EV is shown, and this information has been gained via the use of four distinct driving cycles (FTP75, HWFET, ArtemisRR, and LA92). As previously discussed in Section 3.4, the driving cycle provides speed information to the intended vehicle as an input. This speed information is referred to as the reference speed of the vehicle, whilst the speed information derived from the simulation's output is referred to as the real speed of the vehicle. When a vehicle is moving at a certain pace, the reference speed that is used to calculate its movement is compared to its real speed, which is shown in Figure 4.11. This is the first of the results received from the simulation outputs, and it is the speed graph.

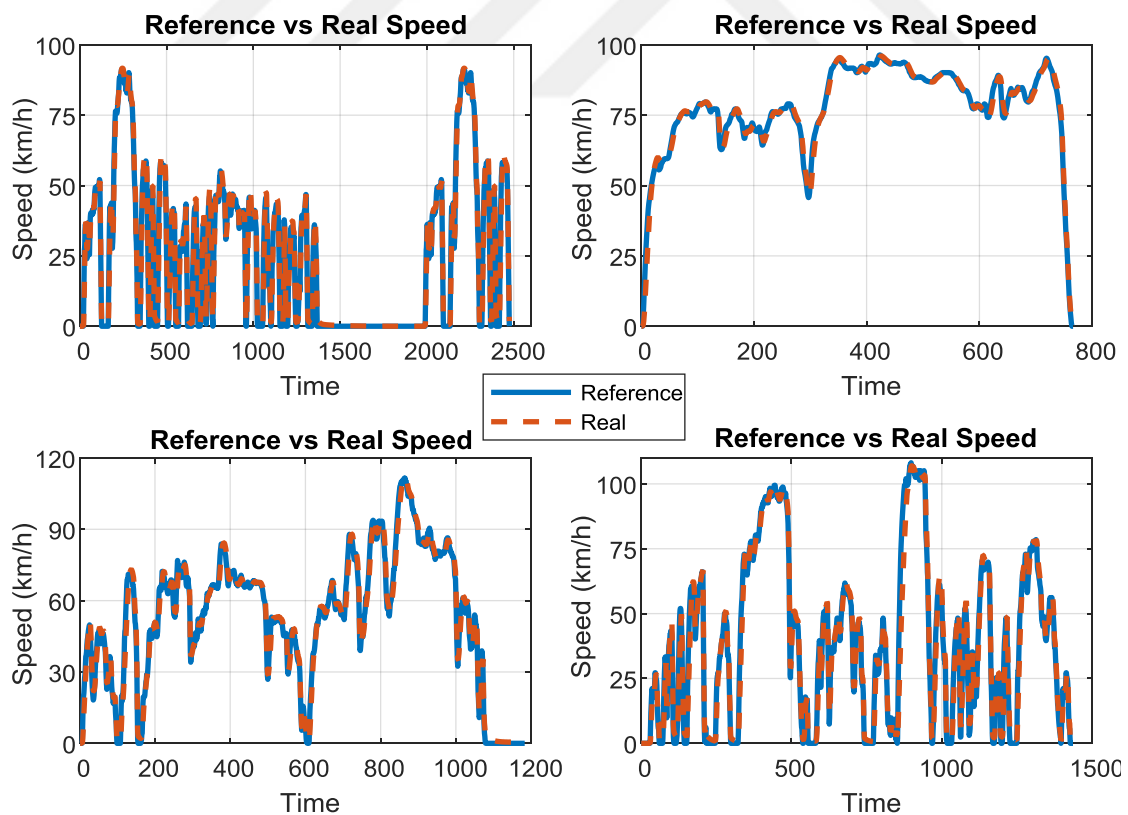


**Figure 4.11.** Reference versus real vehicle speed.

The position information of the vehicle is the second most essential output after the comparison of the reference speed established as the input of the vehicle and the actual speed of the vehicle. It is derived by integrating the actual vehicle speed with the reference speed, which is done with the help of the integral block. Furthermore, the vehicle's position is regarded as the most critical data set for the comparison and error-finding portion of the algorithm. Figure 4.12 shows the second output from the simulation of the car, which is four different positions, and it can be observed that the position information of the vehicle follows the reference position information.

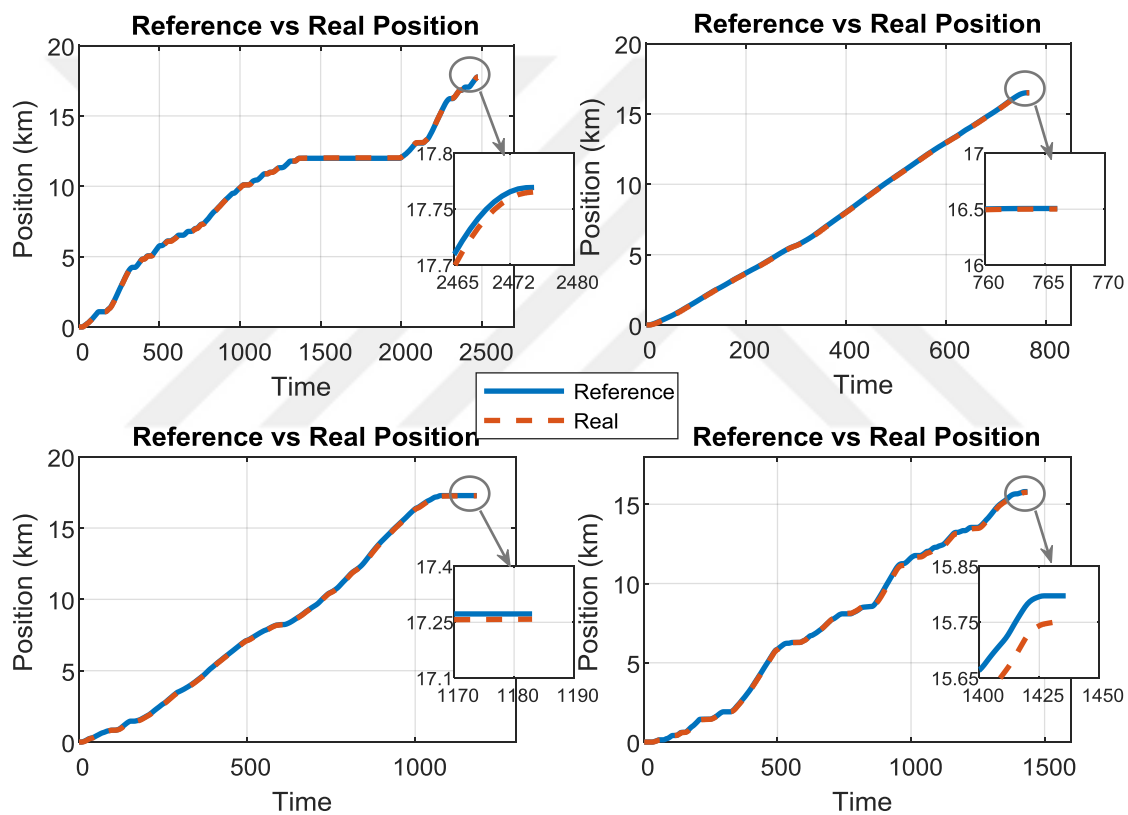


**Figure 4.12.** Reference versus real vehicle positions.

The driver model uses the PI controller to monitor the reference speed, which is the input signal, as well as the real speed of the vehicle, enabling the actual speed of the vehicle to match the reference speed as closely as possible. Additionally, the driver model sends the appropriate signals to the PWM block in the motor subsystem of the car, taking into account the driving cycle applied to the vehicle's input. A PWM block is used in this process to convert the driver model output signals into H-bridge-usable signals. This block offers a link between the input signal and the H-bridge. In the last step, the voltage values needed by the

engine are calculated by the H-bridge. The engine requires the energy to generate this voltage information and transfer it to the vehicle's dynamics via rotational motion. This is accomplished by the use of a battery to provide the energy needed by the motor. With the help of the existing energy in the battery, the motor is able to generate rotational movement via the formation of a current. This rotating motion is then transmitted to the vehicle's wheels via a process known as torque conversion.

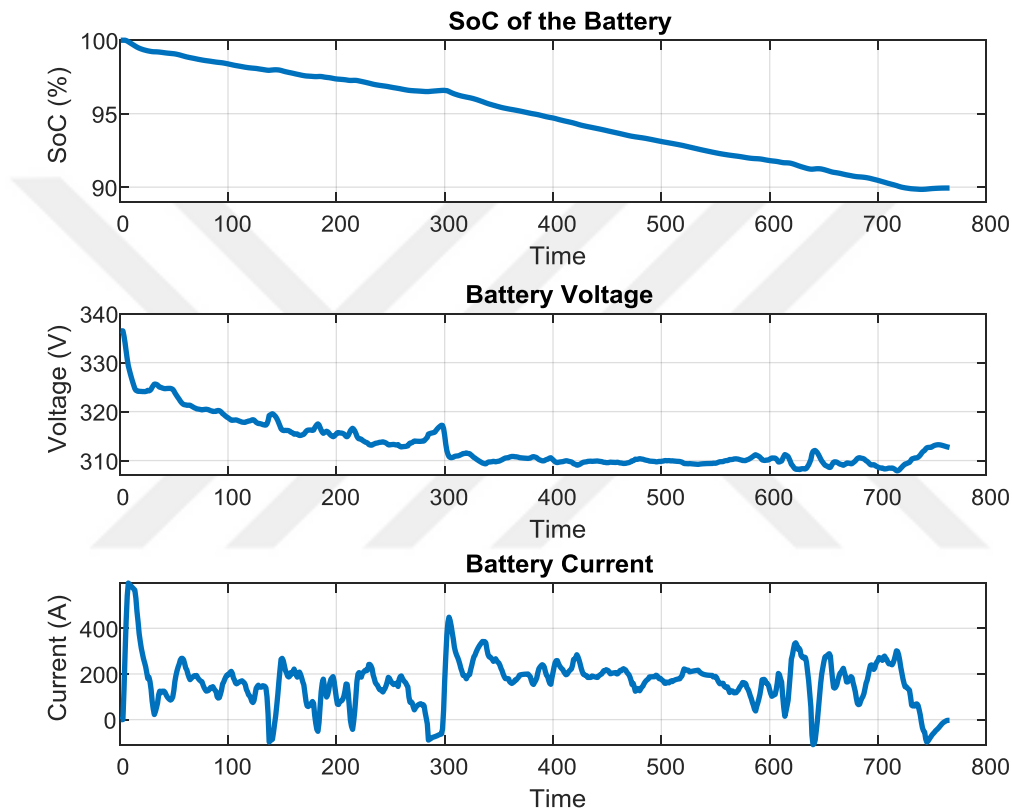


**Figure 4.13.** The SoC of the battery, the voltage, and the current drawn from the battery.

As shown in Figure 4.13, in order for the actual speed of the vehicle to follow the reference speed, changes occur in the current drawn from the battery in line with the signals given. This figure has been created for driving cycle (HWFET) data only. Furthermore, fluctuations in the voltage of the battery may be noticed as a result of these changes in the current flowing through it. Considering these changes, it is seen in the figure above that the percentage of energy remaining in the vehicle's battery decreases proportionally to the current drawn by the motor from the battery.

#### 4.7. The Simulation Results of the EVs in the MapleSim

The real speed and position information of the vehicle has been obtained using four different driving cycles (FTP75, HWFET, ArtemisRR, and LA92), as seen in Figure 4.11. Using four distinct driving cycles, the vehicle's real speed and position have been determined. The driving cycle sends speed information to the intended vehicle as an input. This speed information is referred to as the reference speed of the vehicle, whilst the speed information derived from the simulation's output is referred to as the real speed of the vehicle. Figure 4.14 shows the difference between the reference speed used to compare the movement of a vehicle and the real speed.

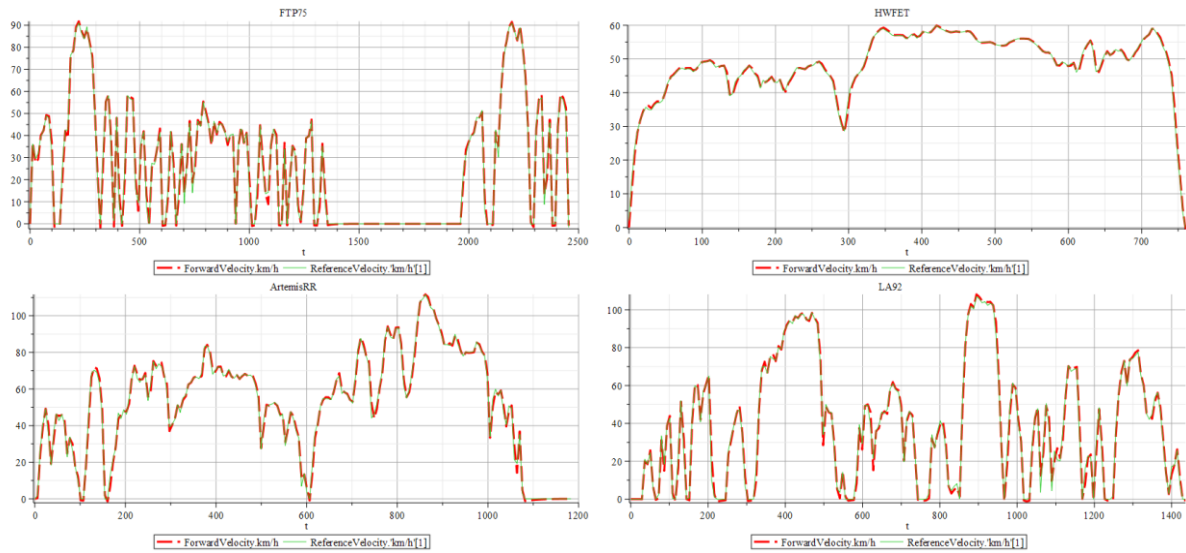


**Figure 4.14.** Reference versus real vehicle speeds.

After comparing the vehicle's reference speed to its real speed, the vehicle's position information is the second most important output. While the reference position information of the vehicle is found by integrating the driving cycle assigned as input, the real position information is obtained by integrating the real speed. For the algorithm's comparison and error-finding sections, the vehicle's position is considered the most significant data set. Figure 4.15 depicts the second output from the simulation of the vehicle, which consists of four separate positions. It can be seen that the position information of the vehicle follows the reference position information, as can be seen in the following figure.

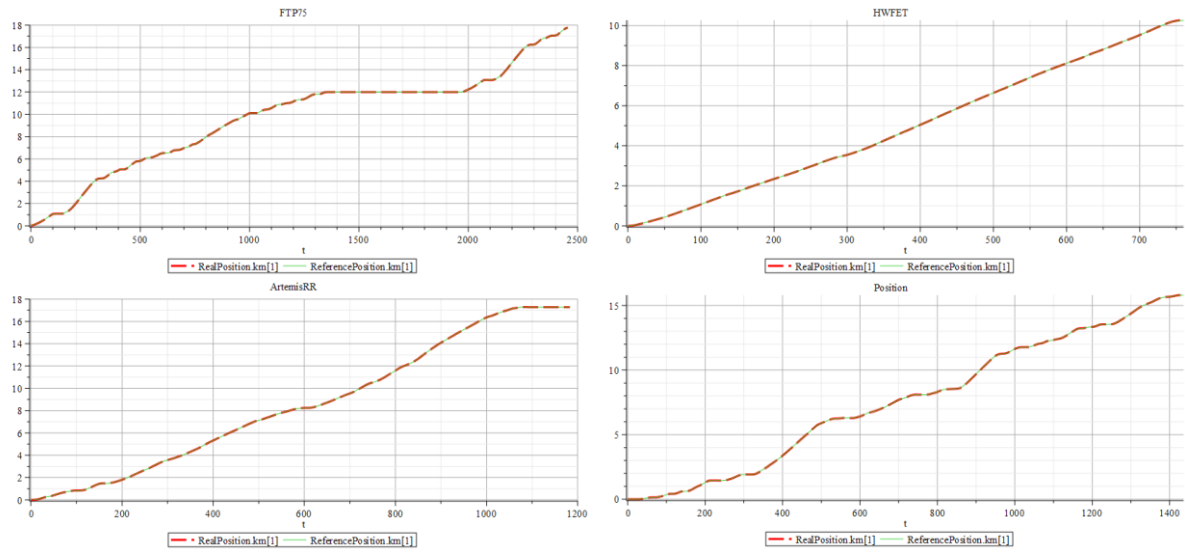


**Figure 4.15.** Reference versus real vehicle positions.

The driver model uses a PI controller to monitor the reference speed and ensure that the actual speed of the vehicle is close to the reference speed. Also, considering the driving cycle at the input, it sends appropriate signals to the PWM block in the vehicle's engine subsystem. The PWM block acts as a link between the input signal and the H-bridge. In the last step, the voltage values required for the motor and the H-bridge are calculated. The motor needs energy to produce this voltage information and transfer it to the dynamics of the vehicle with the rotational movement. This needed energy is also obtained by using a battery. The motor provides the rotational movement it needs by creating a current thanks to the energy in the battery. This rotational movement created is transferred to the wheels of the vehicle by the torque conversion process.

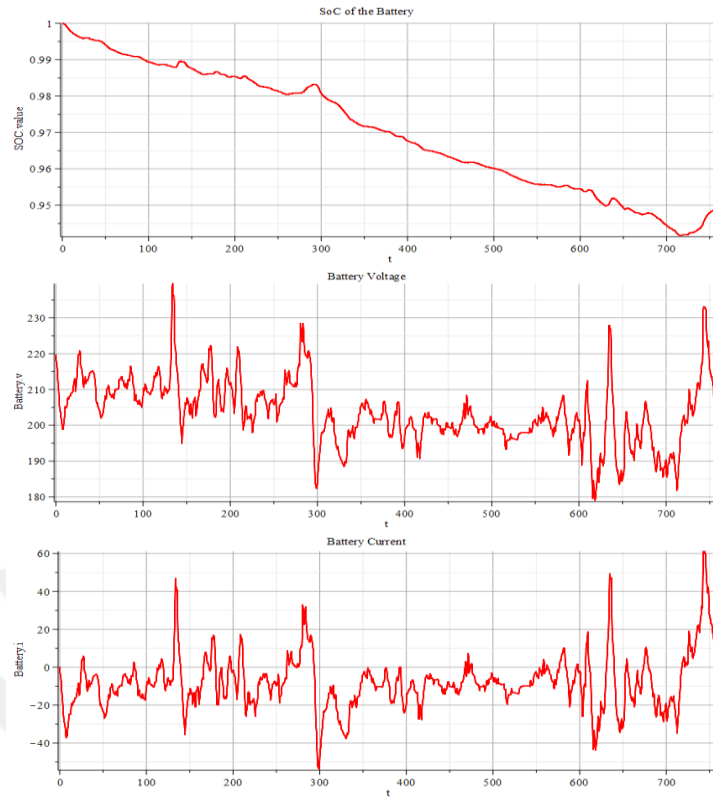


**Figure 4.16.** The SoC of the battery, the voltage, and current drawn from the battery.

As illustrated in Figure 4.16, variations in the current taken from the battery in accordance with the signals provided because of the actual speed of the vehicle to follow the reference speed. This figure has been created for driving cycle data only. In addition, fluctuations in the current in the battery are also reflected in the battery voltage. Considering these oscillations, it is observed in the figure that the remaining energy in the vehicle's battery decreases in parallel with the current drawn by the engine from the battery.

#### **4.8. Driving Range (DR) Estimation by Using MLP-BP**

This chapter firstly presents the construction of the algorithm that was mentioned in Section 3.6. In addition, in other subsections of this chapter, the values used for the input of the algorithm, which are vehicle speed, motor angular speed, motor voltage, and SoC, are changed in order, and the obtained DR results are compared for each input.

##### **4.8.1. Construction of the BP Algorithm**

The algorithm to be used for DR estimation has been defined and is aimed at forecasting the path that the vehicle can be driven with the current battery capacity based on the data obtained

from the simulation. While obtaining these training inputs, 10 different driving cycles were used in the simulation. Accordingly, four different input parameters to be used in the developed algorithm have been determined. The structure of the algorithm can be illustrated as shown in Figure 4.17. In addition, these inputs ( $x_2$ ) and their corresponding times ( $x_3$ ) have been sampled, and  $2 \times 41$  matrices have been generated for each parameter. Due to the structure of the algorithm used, a vector ( $x_1$ ) consisting of 1s of all elements known as bias is added to the input matrix obtained. In addition, the position parameter is represented with the  $y_j$ . The parameters, which are vehicle speed, motor angular speed, motor voltage, and SoC, are processed independently in the algorithm and the results are explained in the following subsections. Finally, DR estimations from each parameter have been compared in the following subsections.

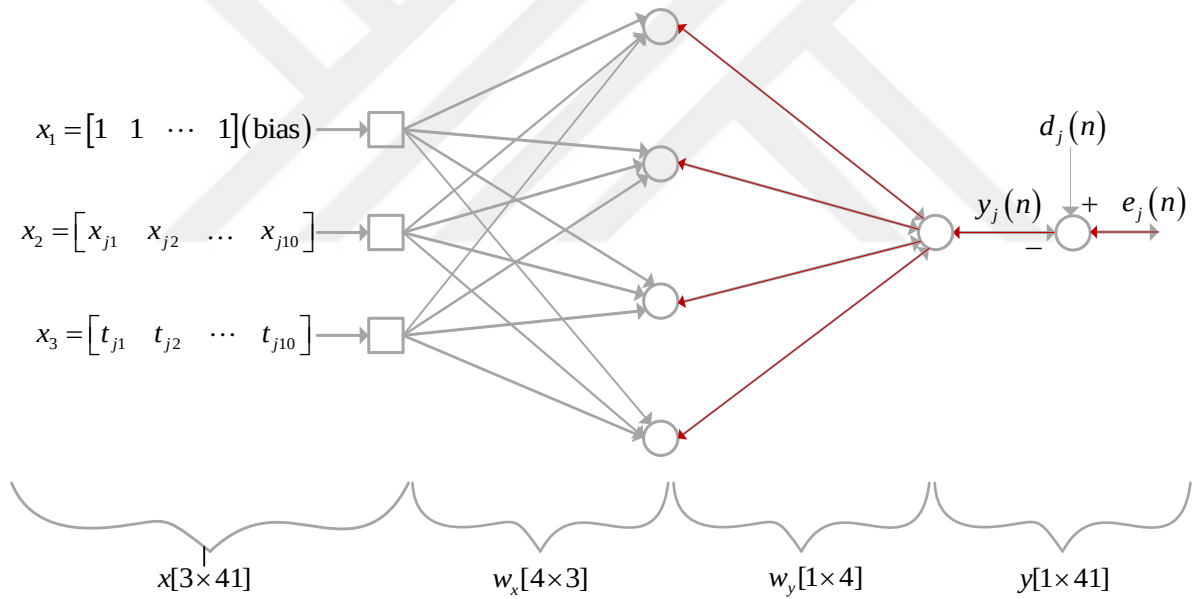


**Figure 4.17.** The structure of the BP algorithm.

#### 4.8.2. Vehicle Speed

One of the data obtained with the 10 different driving cycles used in the simulation specified in Section 3.4 is the actual speed of the vehicle. Since each driving cycle is determined as a different input, the actual vehicle speeds observed as a result of these driving cycles also differ.

#### 4.8.2.1. Training Part of the Algorithm

The vehicle speeds obtained from the developed simulation, which are shown in Figure 4.18, are assigned as input to the algorithm mentioned in Chapter 3. The structure of the algorithm includes 3 input nodes, 4 hidden layers, and 1 output node. The output obtained from this algorithm is the DR values to be estimated. As a result of the iteration, the mean error is monitored in the following figures as a result of the iteration.

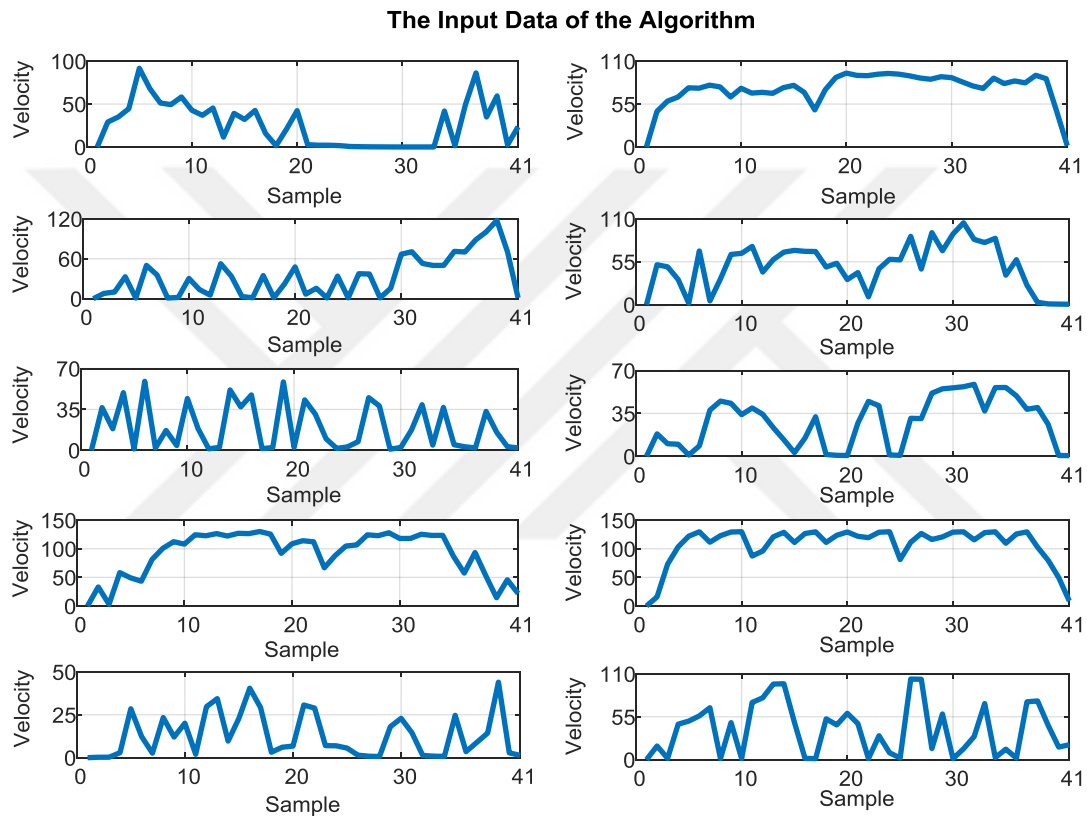


**Figure 4.18.** The input of the algorithm for the vehicle speeds.

This part aims to observe that convergence is ensured for the mentioned parameters. Convergence is often considered to indicate typical use in the area of neural networks. Convergence is defined as the situation in which the developed artificial neural network behaves in accordance with the training model without exceeding a certain error level within the scope of conventional artificial neural networks. In other words, where the negative exponentially of the detected parameter exists, convergence has occurred. Therefore, the developed neural network algorithm is suitable for the training. In Figure 4.19, it can be clearly seen that the convergence is achieved for the mean error, which means it is proof that learning exists in the algorithm.

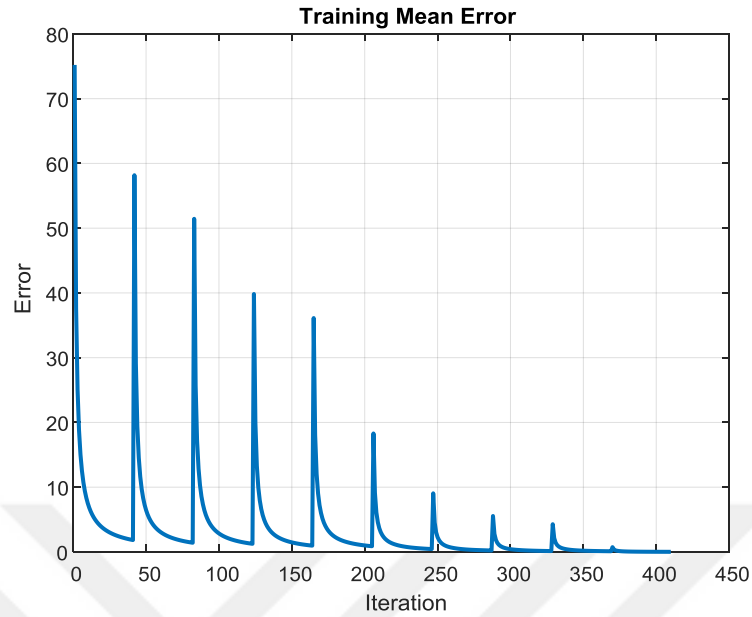


**Figure 4.19.** Mean error of the output for the vehicle speed.

Figure 4.20 shows the position data obtained by training the algorithm as a result of 10 different vehicle speeds used for DR estimation. The algorithm that allows the predicted position data to follow the actual vehicle data is discussed in detail in Section 3.6. The most important factor that provides this is the training of unknown/weight parameters determined randomly between nodes.

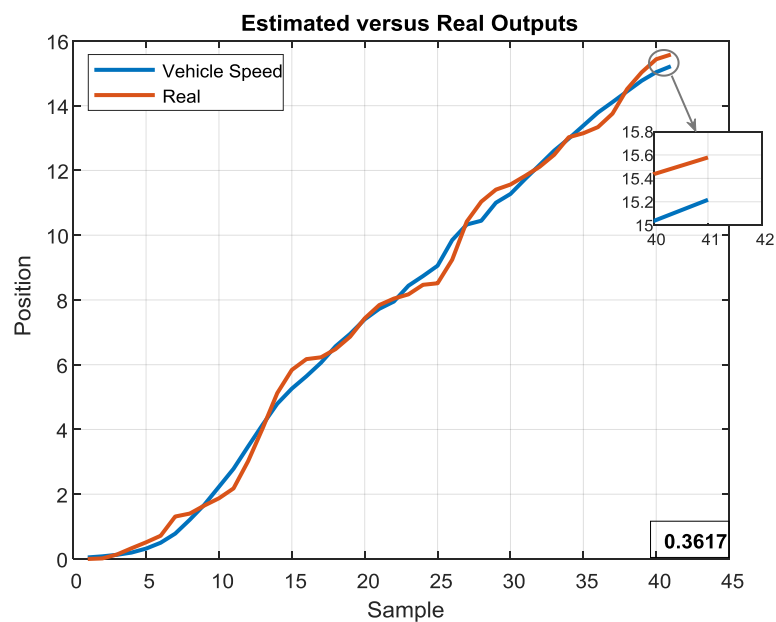


**Figure 4.20.** The estimated position of the vehicle with the trained data.

#### 4.8.2.2. Testing Part of the Algorithm

According to this study, the real position data obtained from the simulation and the position data detected from the developed algorithm have been compared. As clearly seen in Figure 4.21, the error range of the overall system is determined by considering the minimum and maximum values, which are  $[0.2427 - 0.66]$  respectively, among the calculated error values when comparing four different DRs. The error detected as a consequence of this comparison proves that the algorithm has high accuracy.

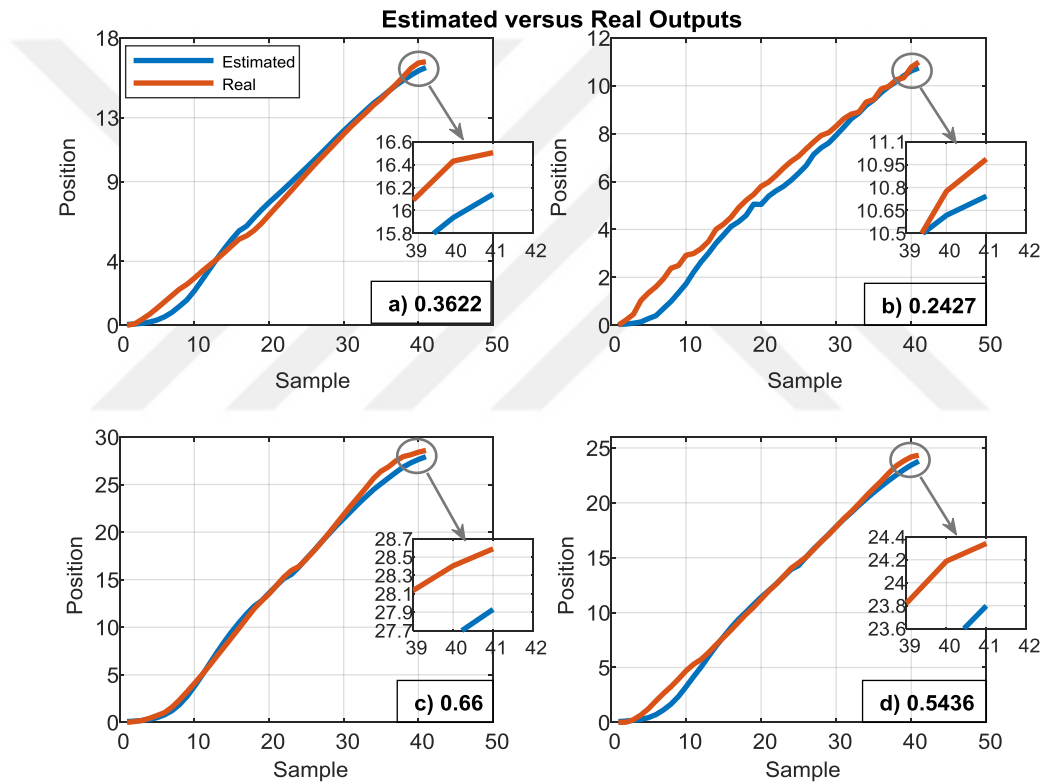


**Figure 4.21.** Tested vehicle position for four different driving cycles.

#### 4.8.3. Motor Angular Speed

In this section, instead of vehicle speed, motor angular speed is assigned as input to the algorithm developed to make a comparison with the data obtained from the simulation.

##### 4.8.3.1. Training Part of the Algorithm

The motor, which is one of the important parts of the electric vehicle simulation, provides the necessary angular motion for the vehicle. To obtain the motor angular speed value, the speed

information received from the driving cycle is transferred to the controller. The discrete signal produced in the controller is transferred to the PWM so that it can be translated into a form that the motor can process, which is voltage. As seen in Figure 4.22, the motor angular speeds obtained from the developed simulation are assigned as inputs to the algorithm.

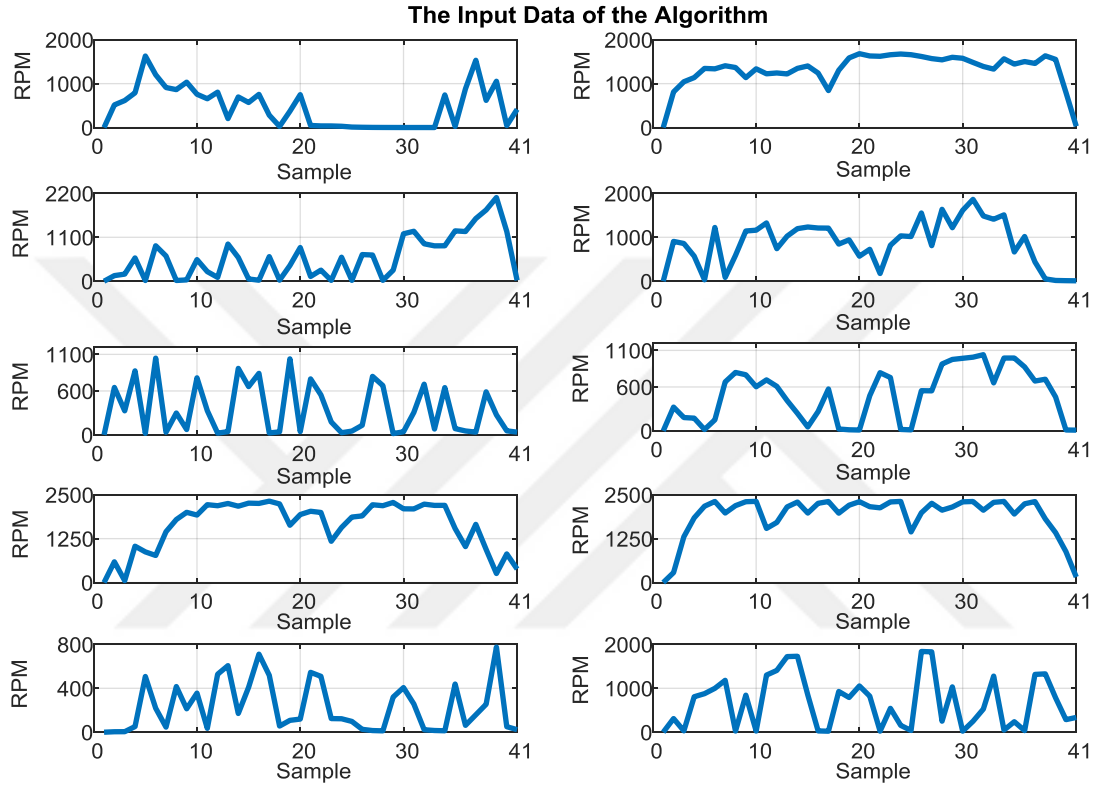


**Figure 4.22.** The input of the algorithm for the motor angular speeds.

As in other sections, our aim is to detect that the convergence of the mean error value converges to zero. As seen in Figure 4.23, the mean error has converged to zero, which indicates that the algorithm is working accurately.

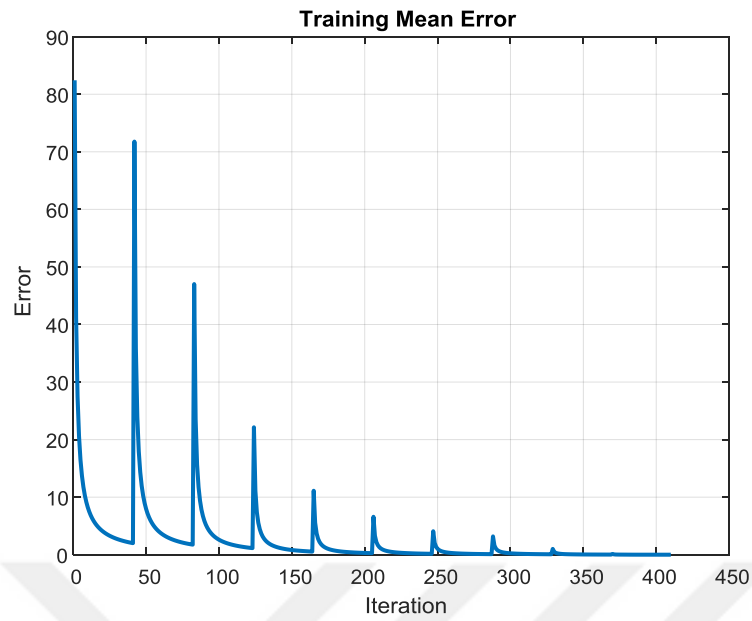


**Figure 4.23.** Mean error of the output for the motor angular speed.

Figure 4.24 shows the position data obtained by training the algorithm as a result of 10 different data samples of the motor angular speeds used for DR estimation. According to the following figure, the trained data for position is overlapped with the real-time data.

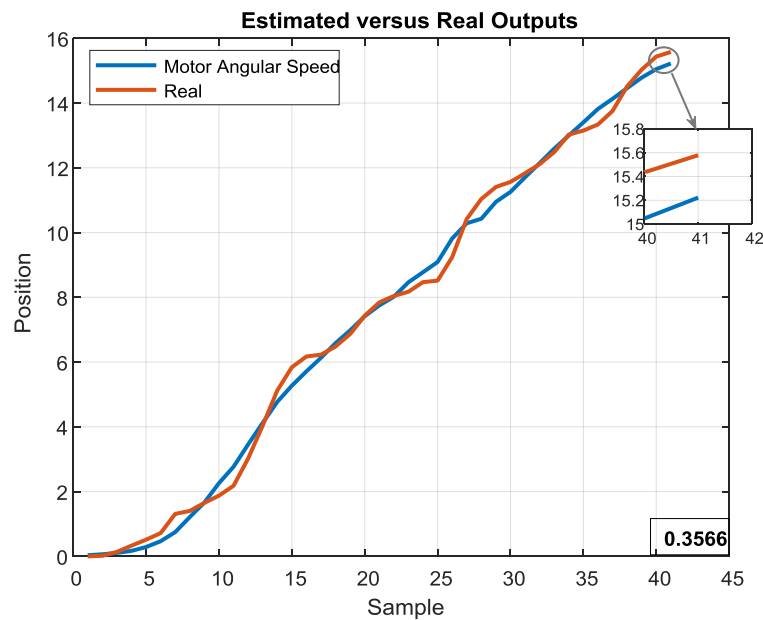


**Figure 4.24.** The estimated position of the vehicle with the trained data.

#### 4.8.3.2. Testing Part of the Algorithm

An error signalis  $[0.2378-0.5334]$  respectively is created by comparing the data observed from the simulation with the data obtained by assigning the motor angular speed to the algorithm as input. As seen in Figure 4.25, the final DR estimation converges to the actual position value of the vehicle at the desired level.

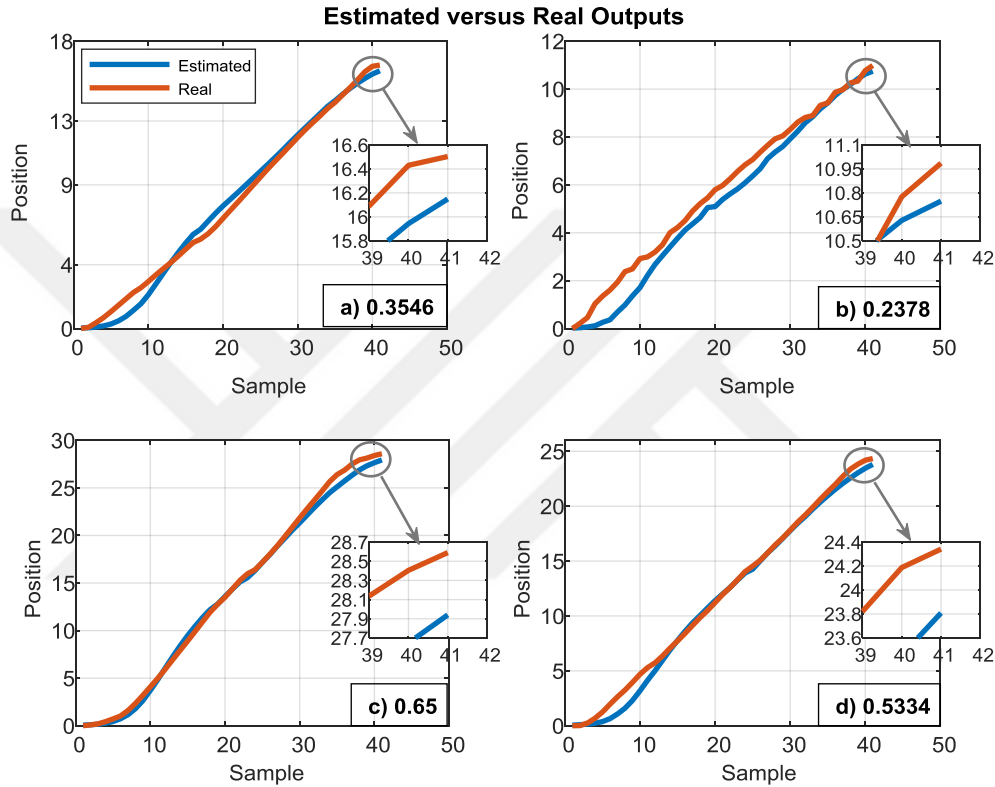


**Figure 4.25.** Tested vehicle position for four different driving cycles.

#### 4.8.3.3. Comparison of the Two Inputs

The aim is to numerically compare the position information obtained from the vehicle speed used as input in Section 4.8.2 and the position information observed with the motor angular speed selected as input in this section. As a consequence of the comparison, it has been observed that the position information forecasted by the motor angular speed input is closer to the actual data compared to the other. Finally, the error between the compared data is calculated and shown in Figure 4.26.

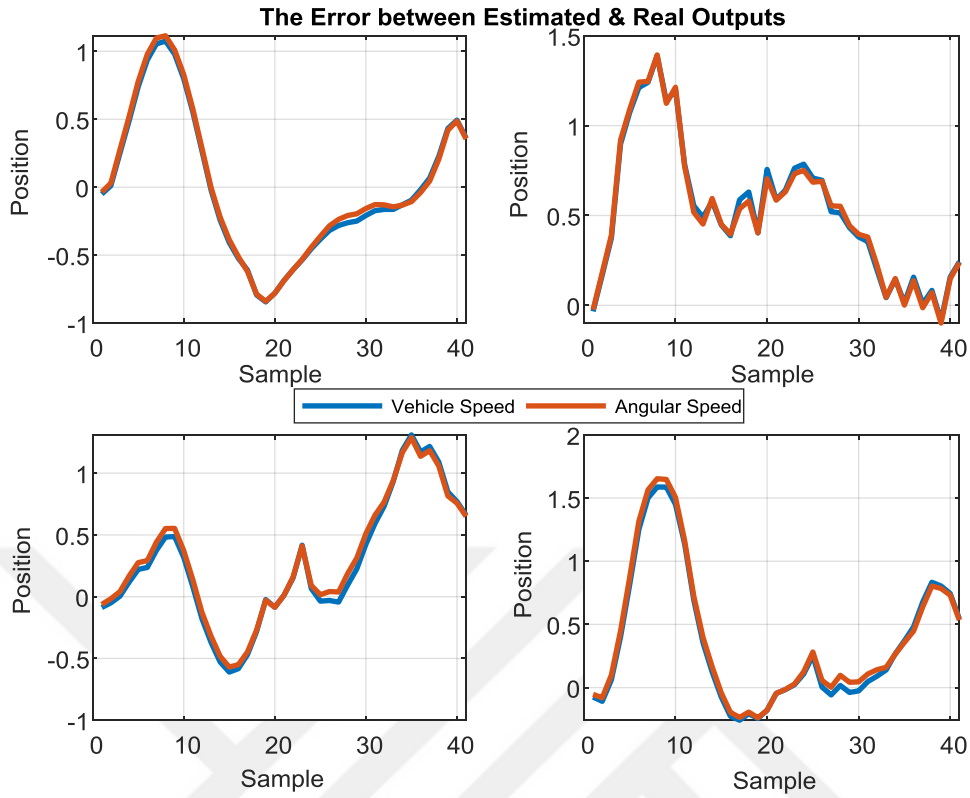


**Figure 4.26.** The error between estimated and real outputs.

#### 4.8.4. Motor Voltage

In this section, DR estimation has been performed for 10 different driving cycles by using the motor voltage, which is another input parameter of the algorithm.

##### 4.8.4.1. Training Part of the Algorithm

To compare the accuracy of this algorithm, four different input parameters have been determined and the obtained outputs have been compared in the following section. Before performing the comparison, the response of the algorithm to the motor voltage used as input has been examined. In Figure 4.27, the motor voltage, which is used as an input in the algorithm, is shown clearly.

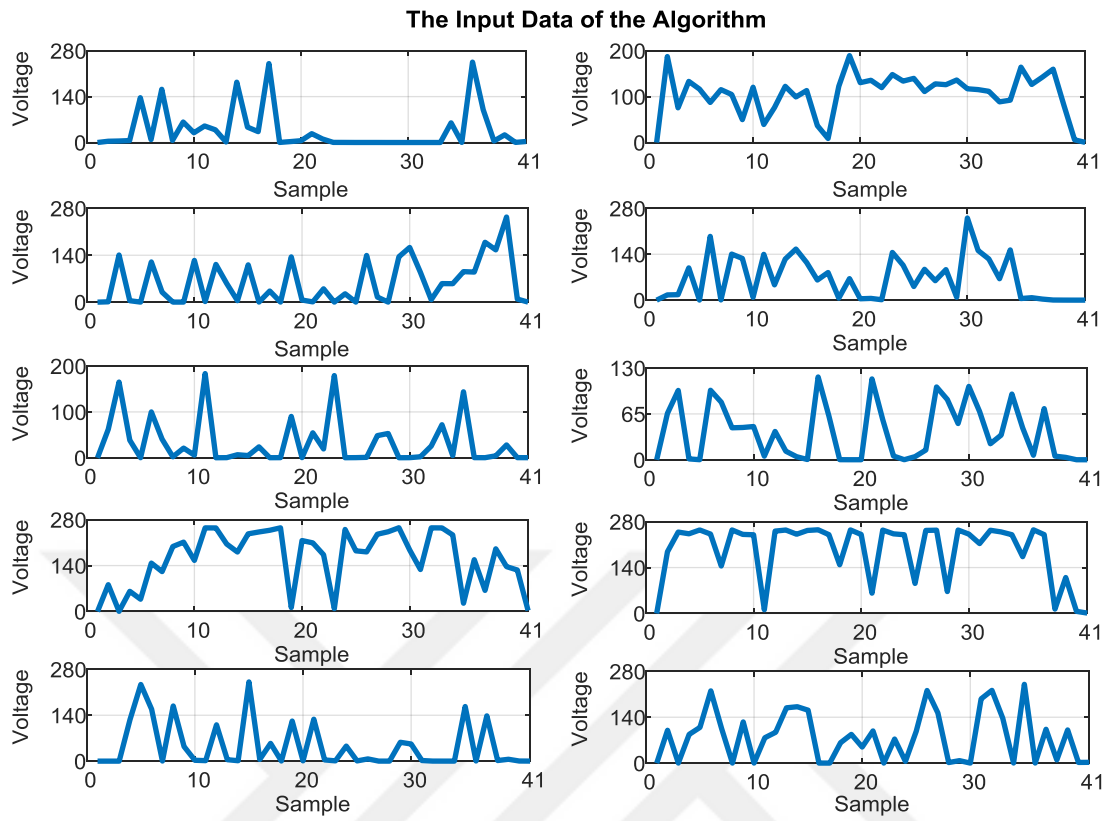


**Figure 4.27.** The input of the algorithm for the motor voltage.

To determine the accuracy of the algorithm, the mean error, which examines the estimated output obtained for the specified input parameter and the real output, converges to zero, as seen in Figure 4.28.

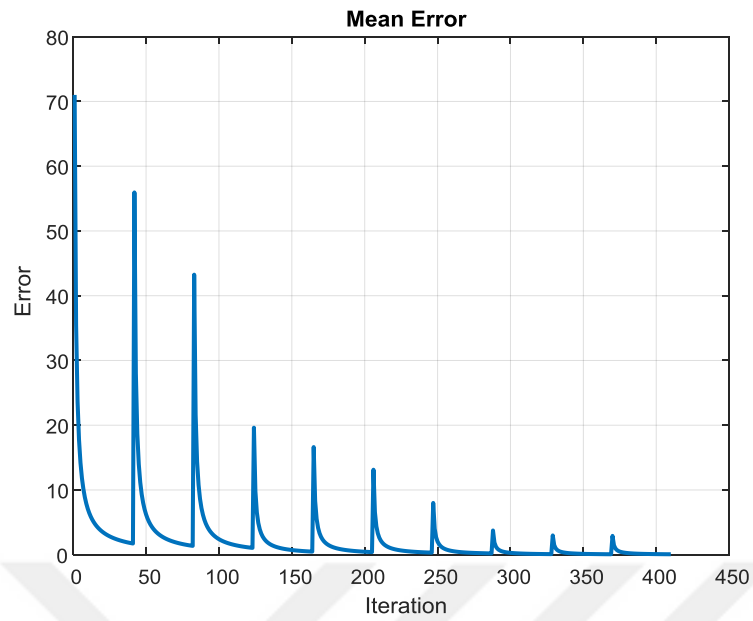


**Figure 4.28.** Mean error of the output for the motor voltage.

The position data produced by training the algorithm as a consequence of the 10 different motor voltages utilized for the DR estimate is shown in Figure 4.29. The training data for the position is overlapped with the real-time data in the diagram below.

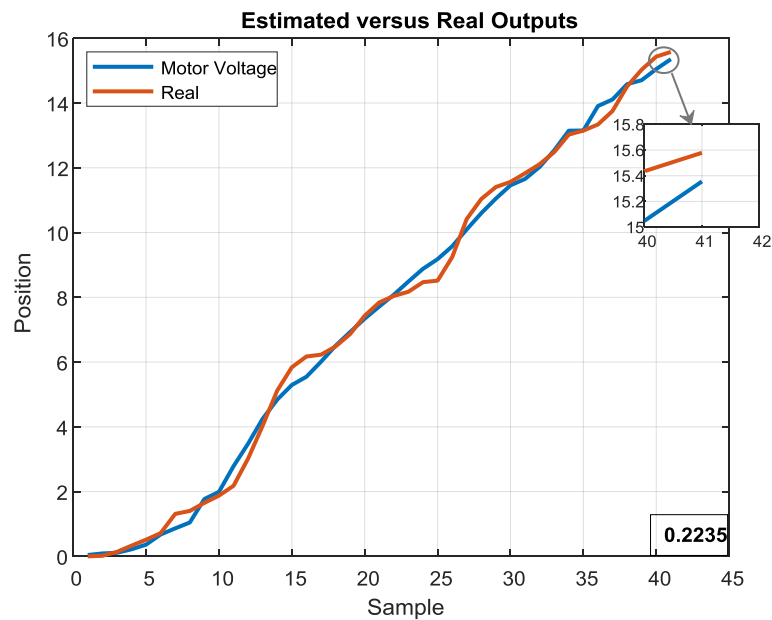


**Figure 4.29.** The estimated position of the vehicle with the trained data.

#### 4.8.4.2. Testing Part of the Algorithm

As seen in Figure 4.30, when we consider the error between the estimated output and real output obtained with the motor voltage assigned as input to the algorithm, the error rate for four different DRs is in between  $[0.2158-0.3241]$ . Therefore, these error values prove that the algorithm works with high accuracy.

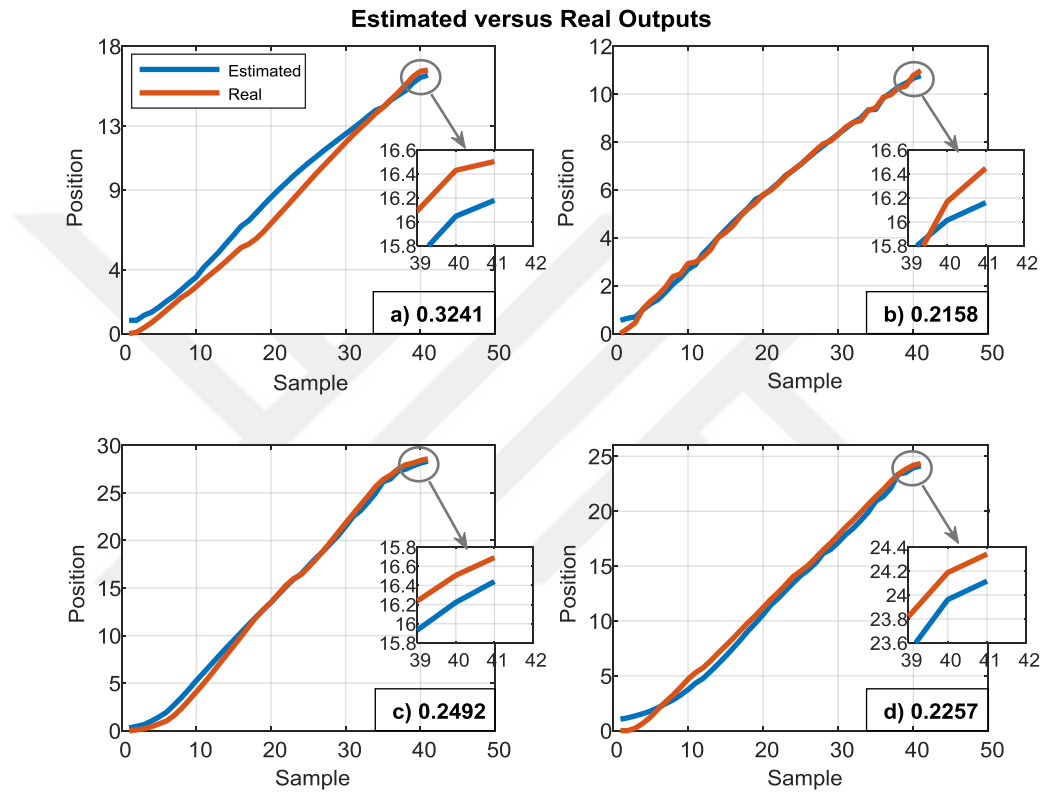


**Figure 4.30.** Tested vehicle position for 4 different driving cycles.

#### 4.8.4.3. Comparison of the Three Inputs

The goal is to quantitatively compare position data acquired from the vehicle speed used as input in Section 4.8.2, position data observed with the motor angular speed used as input in Section 4.8.3, and position knowledge derived from motor voltage. As a consequence of the comparison, it has been discovered that the final position information projected by the motor voltage input is closer to the actual data than the others, as shown in Figure 4.31.

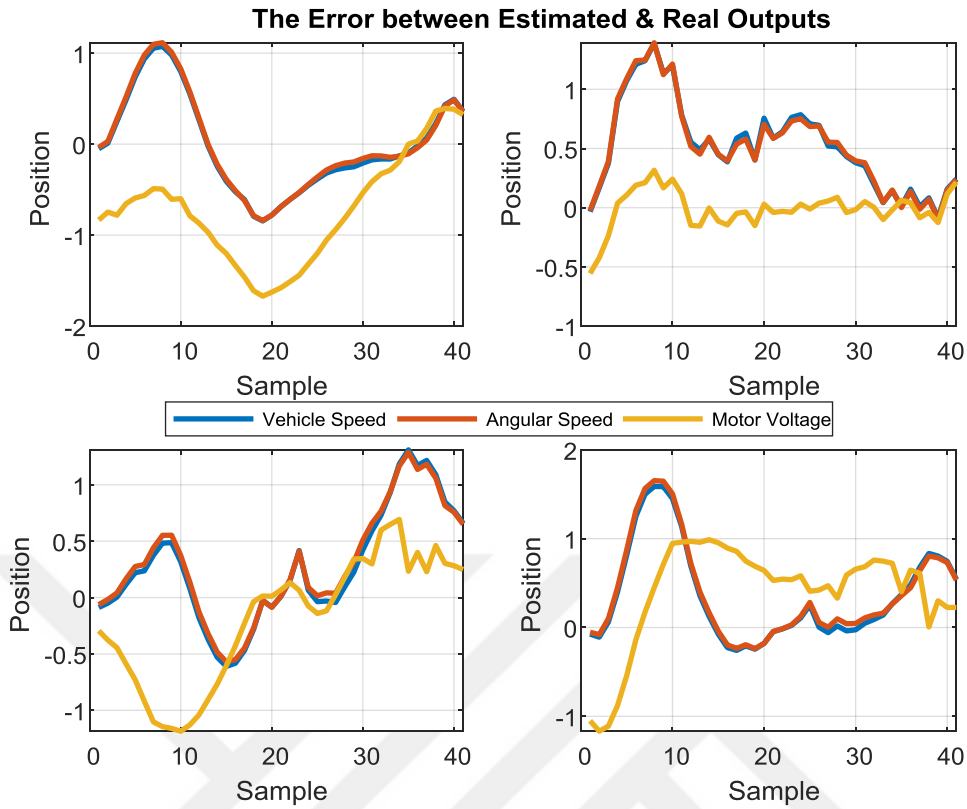


**Figure 4.31.** The error between estimated and real outputs.

#### 4.8.5. State of Charge (SoC)

In this section, DR estimation is presented with SoC, which is the last parameter used as the input of the algorithm.

##### 4.8.5.1. Training Part of the Algorithm

The ratio between the amount of charge extractable from the cell at a given point in time and the overall capacity is described as the SoC. In other words, the ratio of saved energy in the battery to the total energy that can be saved in the battery is known as the SoC. These variables of battery attained from the simulation are represented as the algorithm input that is observed in Figure 4.32.

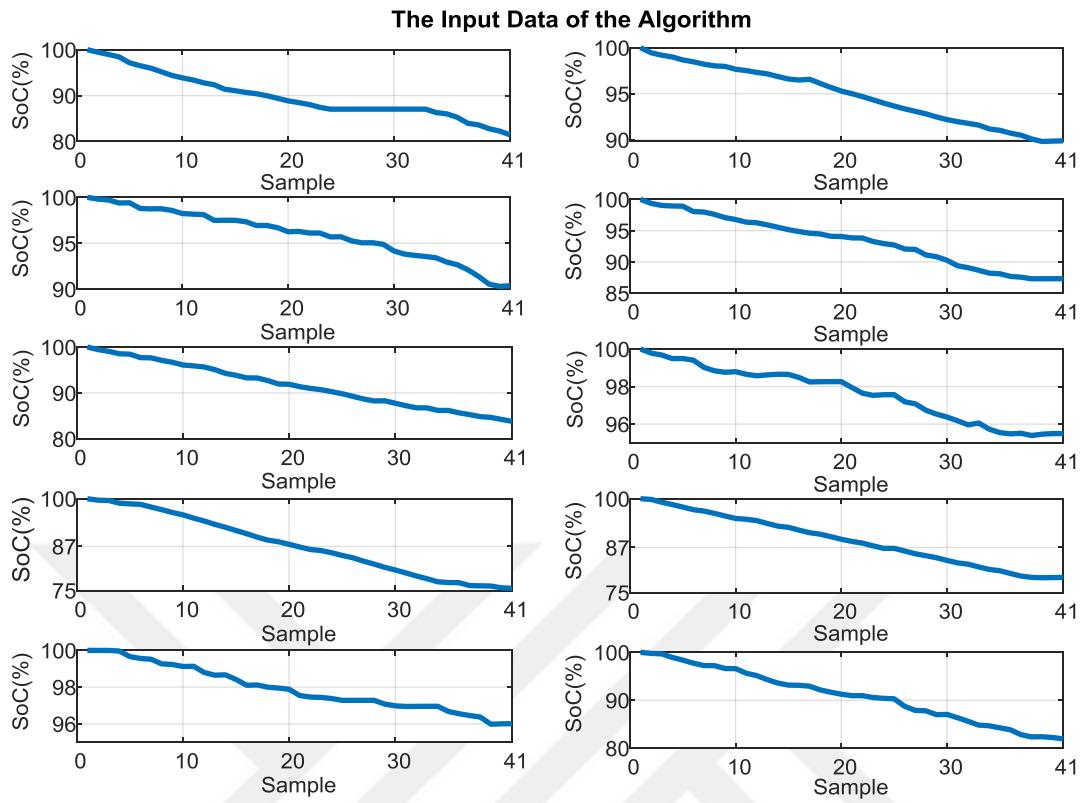


**Figure 4.32.** The input of the algorithm for the SoC.

As seen in Figure 4.33, an estimated output is acquired from the specified input parameter and the real output is examined. In the result of this examination, it is clearly observed that it converges to zero.

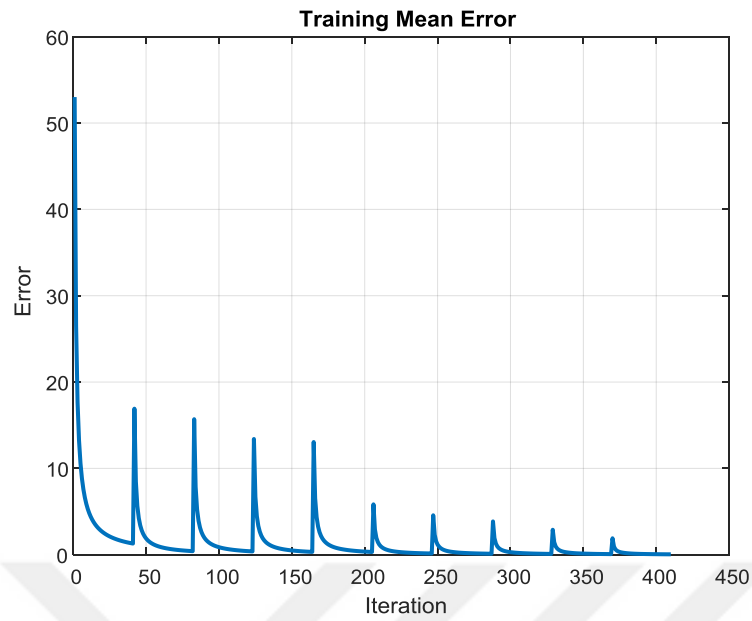


**Figure 4.33.** Mean error of the output for the SoC.

By training the algorithm with 10 different motor voltages, which are used for the DR estimate, the position data is generated. This produced value is compatible with the real-time data as shown in Figure 4.34. According to the mentioned forecasts in other sections, SoC variables have higher accuracy in coherence to real-time data.

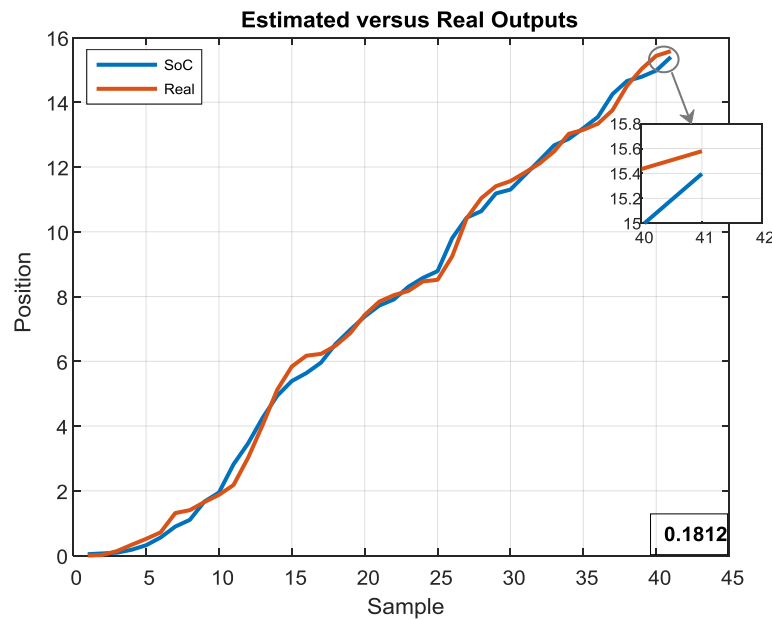


**Figure 4.34.** The estimated position of the vehicle with the trained data.

#### 4.8.5.2. Testing Part of the Algorithm

As a consequence of calculating the error between the real output and the estimated output achieved from the SoC variable assigned as input to the algorithm, it is observed that the error rate for four different DRs is in between  $[0.0884-0.2301]$ . These error values reveal that the developed algorithm gives high accuracy, as seen in Figure 4.35.

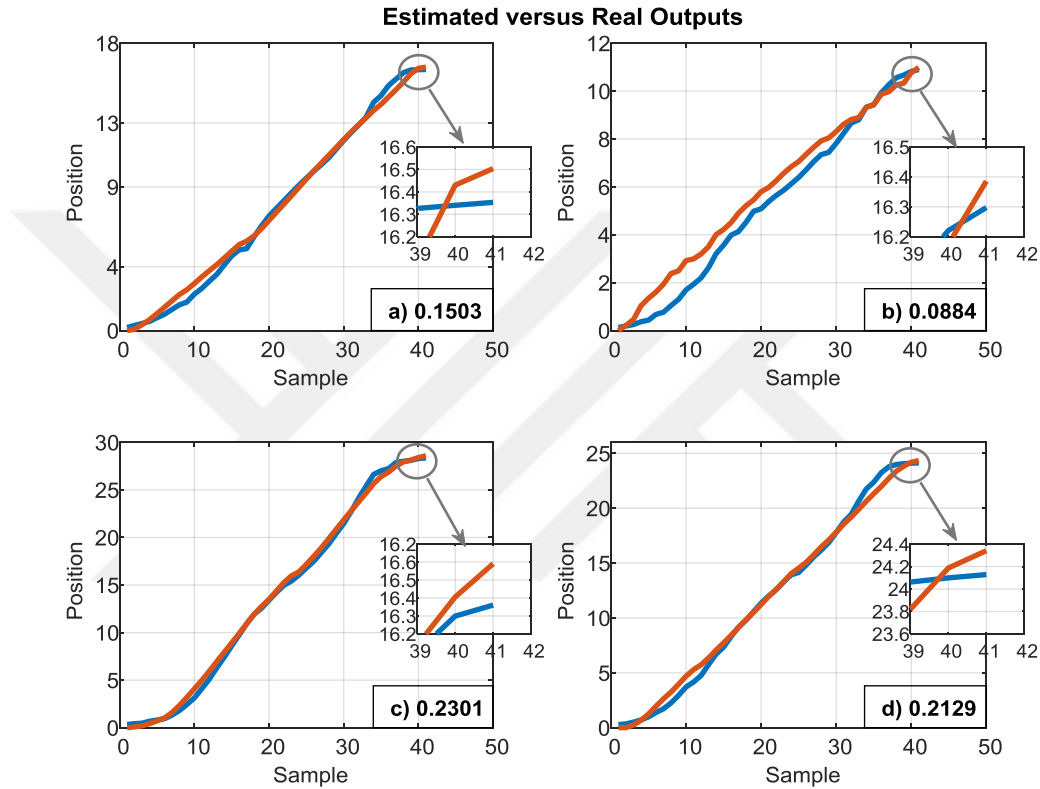


**Figure 4.35.** Tested vehicle position for four different driving cycles.

#### 4.8.5.3. Comparison of the Four Inputs

With the aim of comparing the position data resulted from the vehicle speed used as input in Section 4.8.2, motor angular speed used as input in Section 4.8.3, motor voltage utilized as input in Section 4.8.4, and finally, SoC variables of the battery utilized as input, the following figure is constructed. Depending on Figure 4.36, it is clearly observed that the best final position error between the compared data is in the SoC. Amongst the selected input parameters, the highest accuracy is provided by the SoC input, compared to the other ones.

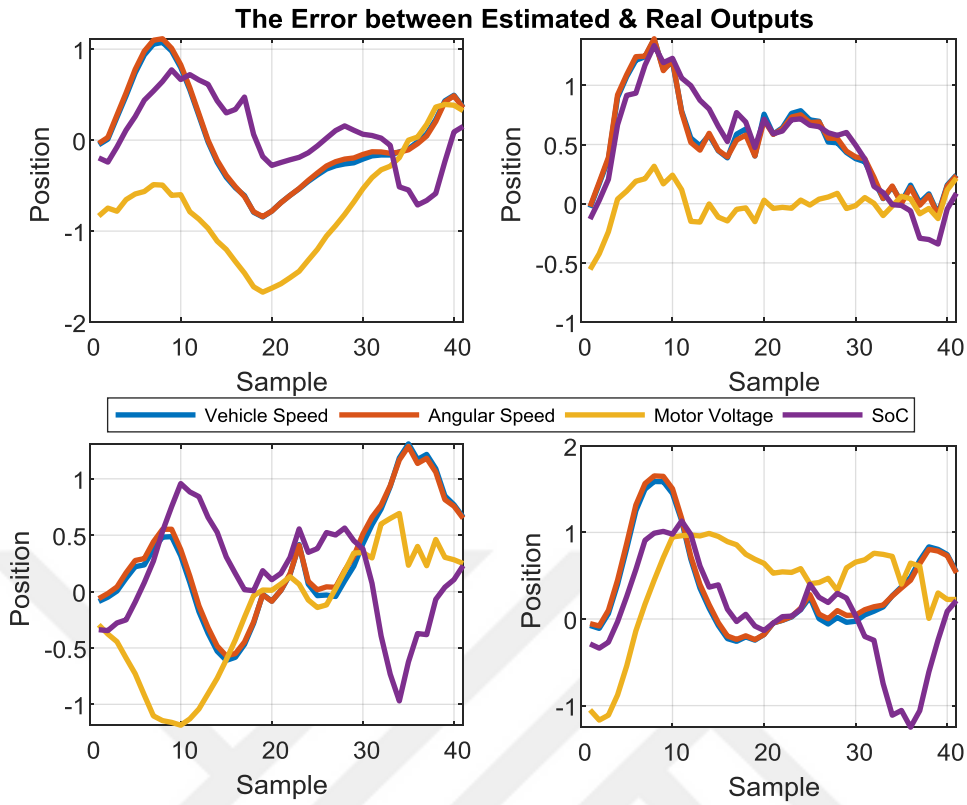


**Figure 4.36.** The error between estimated and real outputs.

#### 4.9. The Optimization of the Detected Lines

In this section, instead of having multiple lines, the slope and y-intercept of the detected lines are averaged out to construct a single line that traces out both lanes of the road. To find the average lines, two empty vectors are constructed for the coordinates of the left and right lines in the road. The coordinates of the lines contain four points that are the points of the x-axis and y-axis. By using these coordinates, the slope of the lines can be calculated by the change in the y-axis over the change in the x-axis. If the slope is negative, the empty vector of the left line is appended. Otherwise, the slope is appended to the empty vector of the right line. Afterward, these slope values for both sides are averaged out to find a single slope, separately. The values of the slope and y-intercept of the left and right lines are shown in the following equation.

$$\begin{aligned} [-1.611 \quad 1201] &\rightarrow \text{Left} \\ [1.024 \quad -298,88] &\rightarrow \text{Right} \end{aligned} \tag{4.1}$$

However, the lines that are found at slope and y-intercept cannot be added to the road image unless we specify their coordinates. In other words, we actually specify where the lines are placed on the x-axis and y-axis. The pseudo code of the optimization is shown in Table 4.3.

**Table 4.3.** The pseudo code of the optimization.

|     |                                                         |
|-----|---------------------------------------------------------|
| 1.  | <b>function</b> average_slope_interecept(image,lines)   |
| 2.  | left_fit = []                                           |
| 3.  | right_fit = []                                          |
| 4.  | <b>for</b> line in lines                                |
| 5.  | <b>shape</b> x1, y1, x2, y2 with respect to the line    |
| 6.  | apply the <b>polyfit</b> ((x1,x2), (y1,y2))             |
| 7.  | <b>slope</b> of the line                                |
| 8.  | <b>intercept</b> of the line                            |
| 9.  | <b>if</b> slope < 0                                     |
| 10. | <b>append</b> the slope and intecrpt in the left_fit    |
| 11. | <b>else</b>                                             |
| 12. | <b>append</b> the slope and intecrpt in the right_fit   |
| 13. | <b>endif</b>                                            |
| 14. | apply the <b>average</b> to the left_fit                |
| 15. | apply the <b>average</b> to the right_fit               |
| 16. | left_line = make_coordinates(image, left_fit_average)   |
| 17. | right_line = make_coordinates(image, right_fit_average) |
| 18. | <b>endfor</b>                                           |

As seen in the following figure, after collecting the coordinates of the lines, new lines are calculated by using the coordinates acquired on the image in order to guarantee that the lines in the road image are as smooth as possible. Once the road lines have been calculated, the coordinates of the road lines are merged, and the inside of the road lines is painted in a rectangular pattern.

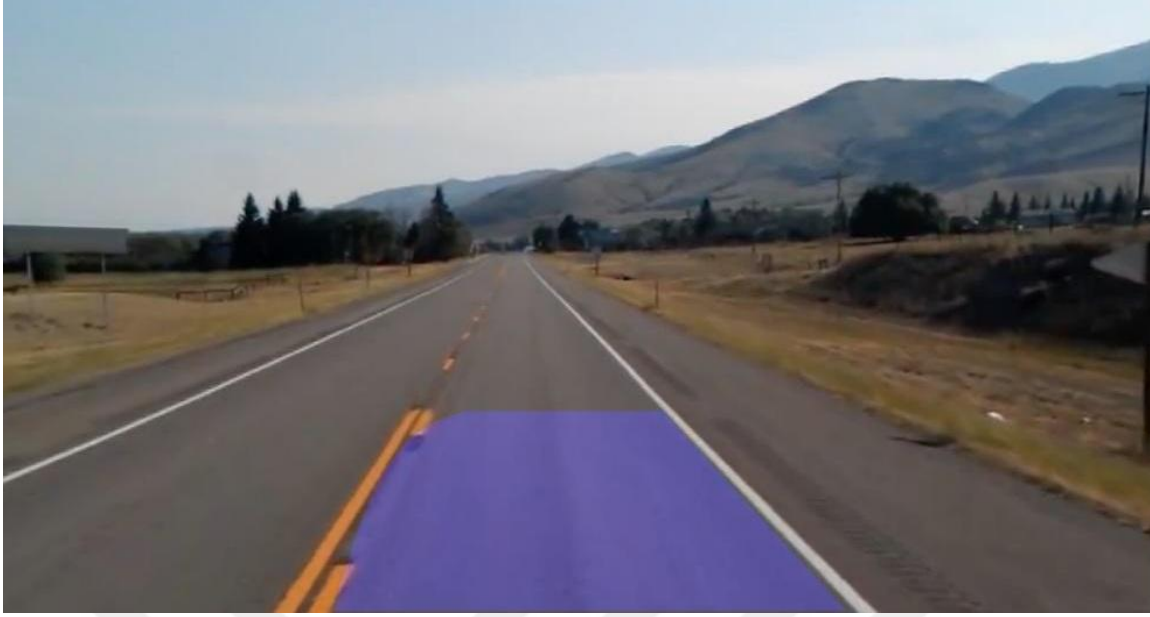


**Figure 4.37.** The optimization of the detected lines in the road image.

As can be seen in the figure above, the lane lines in the road image have been correctly detected. However, lane lines may be detected using several road images to determine the accuracy of the code prepared for line detection. In other words, the aim of this process is to identify whether the prepared code is suitable for various road types.



**Figure 4.38.** The lines detection in the various road images.

As seen in Figure 4.38, the constructed algorithm is applied to various road images to detect the lines. As in Figure 4.37, after the coordinates of the road lines are determined, these coordinates are combined and painted as a rectangular area. In order for the vehicle to move within this area, the necessary signals are sent to the wheels of the vehicle. It is the following step after detecting the lines on the road to detect and track any cars that are traveling down the particular stretch of road.

#### **4.10. The Results of the Vehicle Detection and Tracking**

There are several tasks to detect and track the vehicle, which are: feature extraction, implementing the SVM, training and testing the classifier, and applying the sliding-window. These tasks are mentioned in Section 3.7.2. In this section, the results of the vehicle detection

and tracking are discussed in depth. The testing results of the SVM classifier are shown in Table 4.4.

**Table 4.4.**The SVM classifier testing results.

| Parameters              | Values   |
|-------------------------|----------|
| Times                   | 8 second |
| Error Count             | 43       |
| Accuracy                | 0.9871   |
| Mean Squared Error      | 0.012936 |
| Root Mean Squared Error | 0.113736 |
| Mean Absolute Error     | 0.012936 |

After the model has been trained and tested, the sliding-window technique that has been introduced in Section 3.7.2 is the second task. The original sampling rate with a scale of 1.0 consists of a window with 8 cells, and each cell with 8 pixels. Different sliding window scales, as well as different search area sizes for each of these windows, are experimented with to see which works best. The parameters of the sliding window are shown in the following table.

**Table 4.5.**The parameters of the sliding window.

| Window sizes | Scales | Search region (y_start) | Search region (y_end) |
|--------------|--------|-------------------------|-----------------------|
| 64x64        | 1.0    | 400                     | 496                   |
| 96x96        | 1.5    | 400                     | 592                   |
| 128x128      | 2.0    | 400                     | 656                   |

After that, a heat-map is created by integrating detections from all of the multi-scaled windows into a single operation. This is then thresholded using criteria in order to reject false positives. The following figure depicts the steps involved in the multi-scale window HOG sub-sampling, heat map creation, and thresholding processes.

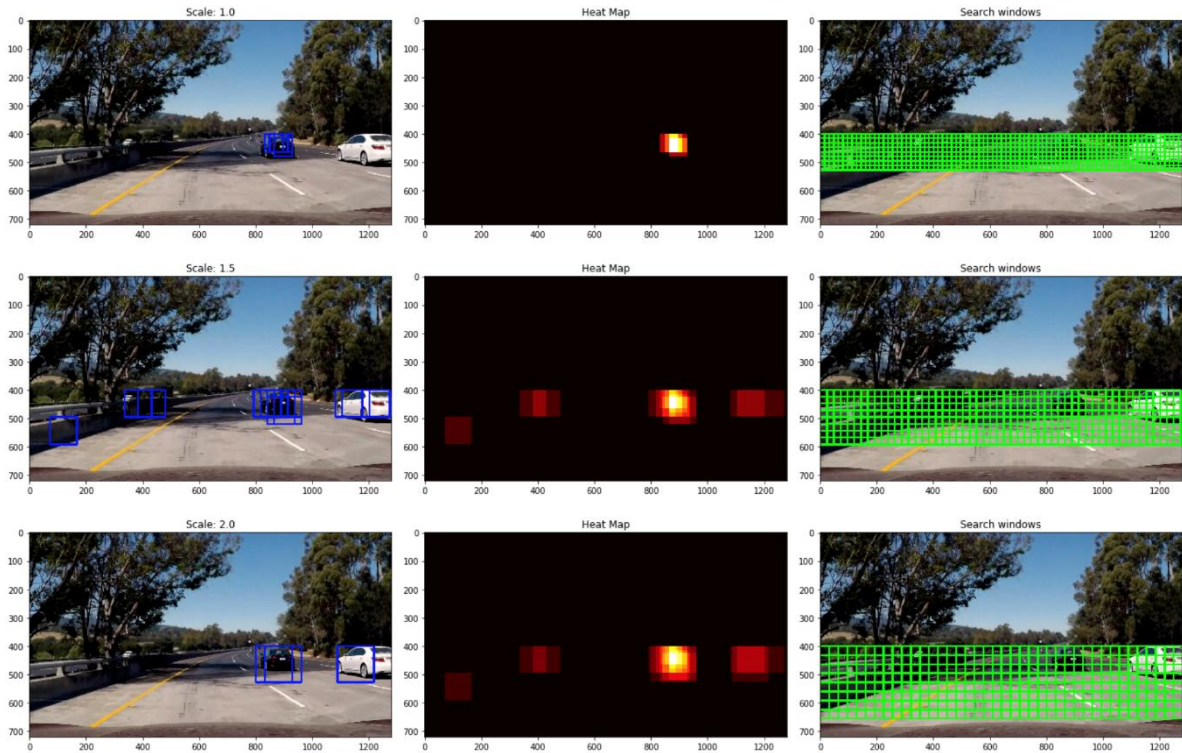


**Figure 4.39.** The process of the sliding window and heat map.

The process of drawing boxes around the cars is getting more precise. Then, after these, the heat map is applied in order to increase the pixel count by one inside each box. Then we need to apply a threshold value to the image to filter out low pixel cells, and find pixels for each car number, and draw the final bounding boxes. The testing heat map images are shown in the following figure.

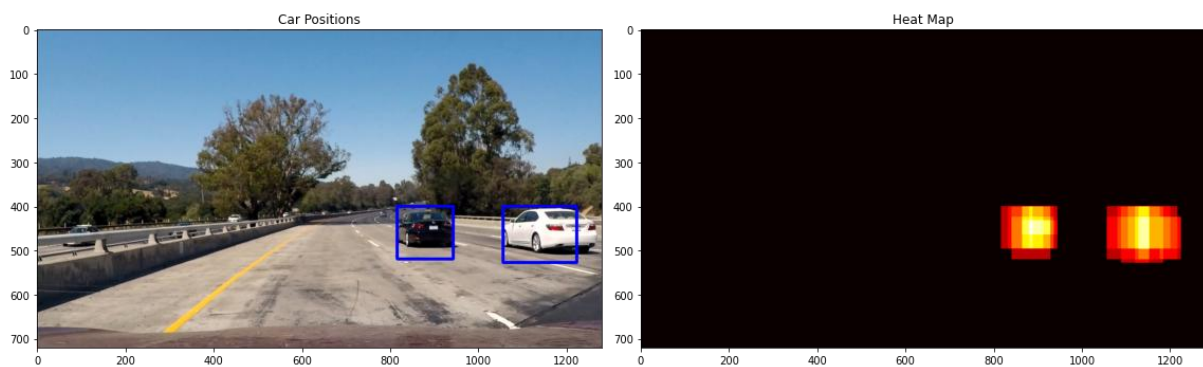


**Figure 4.40.** The result of the vehicle detection.

As seen in the figure above, after the processes are applied to the image, the vehicles in the image are determined and taken for the boxes. So far, the constructed algorithm detects the

cars in the image and draws boxes around them. This algorithm works well on images. However, the second aim of the algorithm is to track the vehicle in the video.

Until now, the models have been able to detect the vehicles and draw boxes around their locations. The primary work has been completed. It is true that the system is capable of detecting the vehicles, but it is not capable of tracking them in a video. The pipeline has been observed over the video and the following statements have been observed:

- Despite the implementation of the HOG sub-sampling window search, the pipeline is incredibly sluggish. It has a processing speed of around 2 frames per second.
- When estimating bounding boxes for the detection of the vehicles, there are a lot of jitters in the results.

The following adjustments are reconstructed in order to enhance the pipeline:

- The pipeline has conducted a full image HOG sub-sampling window search once every 12 frames, and a reduced image HOG sub-sampling window search every 6 frames. The reduced window search simply searches a small portion of the area around the previously detected vehicle positions. As a result, the processing speed has risen from 2 frames per second to 8 frames per second.
- Using a Python deque object with a maximum length of 12, the vehicle bounding boxes of the most recent 12 frames are saved in memory. The current bounding boxes are calculated from a thresholded heat-map of accumulated boundary boxes over the recent 12 frames. This not only helped to smooth up the design of the vehicle bounding boxes but also helped to filter out any false positives that could have occurred.

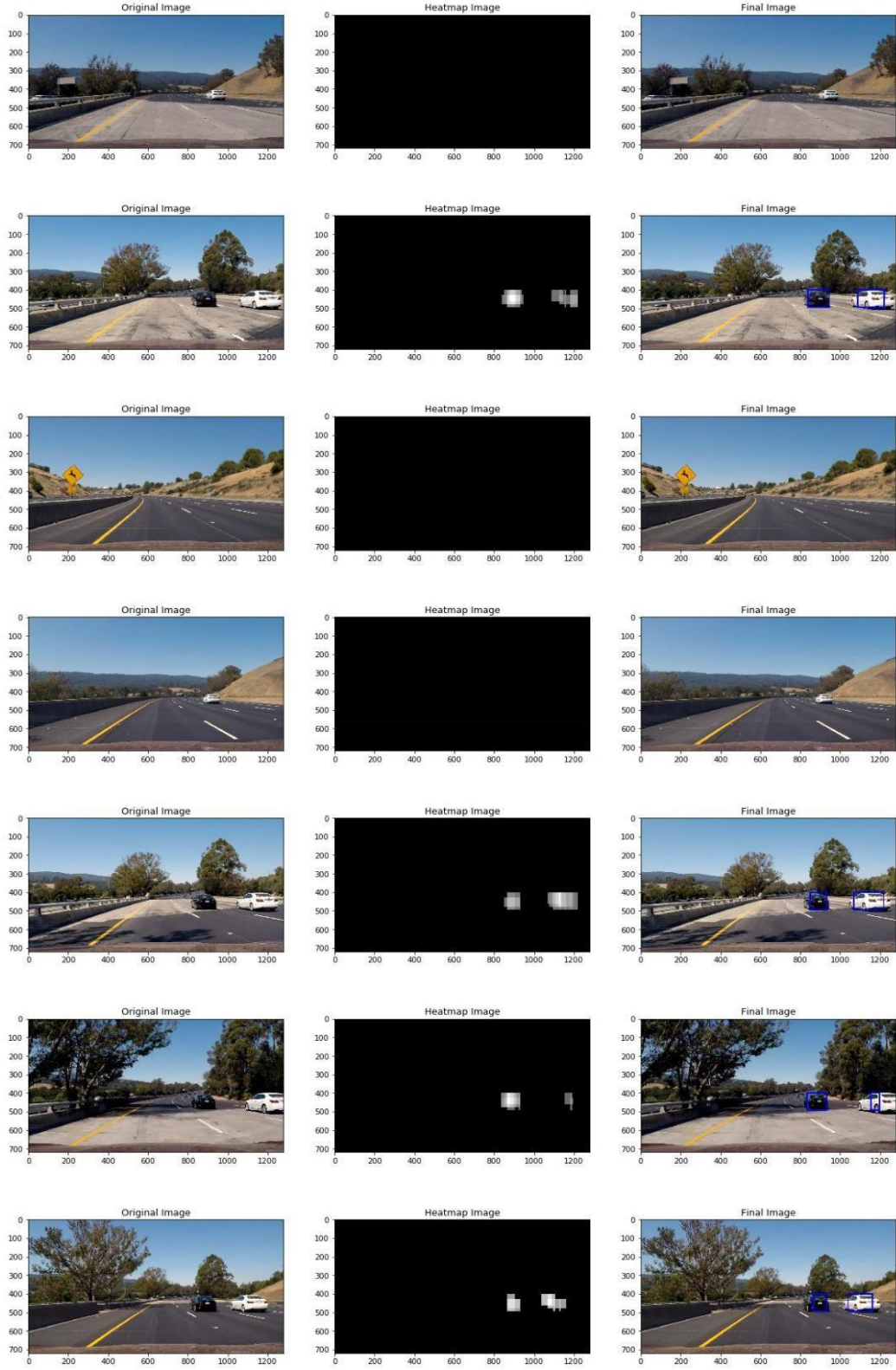


**Figure 4.41.** The pipeline sample on the test images.

The same concept is conducted on the videos that are tested in the dissertation. Since the video is a series of images, this concept can be applied to the video similarly. The algorithm

detects the vehicles passing in the video and tracks the vehicles by drawing a box around them. As a result of the classifier and functions that have been developed in feature extraction and the sliding window, a pipeline that could detect and track vehicles in videos has been constructed. A sample of the pipeline result on test images is shown in Figure 4.41.



## 5. CONCLUSIONS

In this dissertation, there exist main topics about the AVs with the aim of clarifying and providing a contribution to the technology of the AVs. One of the dissertation's primary objectives is to perform an examination on the components that constitute the AVs. These components include the vehicle's battery, BMS, mathematical model, and controller. The main goal of describing these components is to discuss them one by one in order to identify the fundamental simulation produced for the EVs. All of the components described in Chapter 3 have been merged into the simulation environment, and the EV model has been produced. Following this, physical characteristics that have been derived by the EV modelling, such as vehicle speed, SoC, engine angular speed, motor voltage, and position, have been employed as inputs and outputs in the DR estimate process. The DR estimate has been carried out using the MLP-BP technique, which is detailed in the relevant section of the dissertation.

The dissertation concludes with an examination of intelligent control of the AVs, which utilizes developed algorithms to conduct the autonomous features. These algorithms have been utilized to accomplish lane line/vehicle detection and tracking, which may be regarded as the foundation for the vehicle's acquiring autonomous capabilities. The technique, which combines the Canny Function and the Hough Transform, has been used to recognize lane lines in images or videos of a road. A gradient image of the road has been constructed using the Canny Function. The Hough Transform has been used to detect the lines in the vehicle's region of interest in the gradient image. Additionally, an algorithm for detecting and tracking vehicles on the road has been developed using feature extraction, SVM, and sliding-window approaches. Due to the fact that the concentrated information in the raw data contains a substantial quantity of superfluous information, the feature extraction approach has been used to gather and analyse just the most critical information about the item. The SVM classifier has been utilized to categorize the data extracted by feature extraction. The sliding window approach has been used in the last section to detect and track automobiles inside an image or video.

## 6. RECOMMENDATIONS

This dissertation includes modelling and intelligent control of the AV using MATLAB/Simulink, MapleSim, and Python software. The purpose of using different software is to examine the elements that compose the EV separately and compare the results. In addition, the DR estimation of the vehicle has been made by using the simulation results obtained by combining the equipment that forms the EV and the MLP-BP algorithm. The developed MLP-BP algorithm can be used for the DR estimation of the AVs. Furthermore, lane line/vehicle detection and tracking algorithms are constructed for the intelligent control of the AV. The traffic sign detection algorithm, which will be developed together with these intelligent control algorithms, will be applied to the modelled mini autonomous vehicle and the results will be observed. As a result of these observations, the accuracy of the developed algorithms can be determined.

## REFERENCES

- Amer, N. H., Zamzuri, H., Hudha, K., & Kadir, Z. A. (2017). Modelling and control strategies in path tracking control for autonomous ground vehicles: a review of state of the art and challenges. *Journal of intelligent & robotic systems*, 86(2), 225-254. <https://doi.org/10.1007/s10846-016-0442-0>
- Andre, D., Meiler, M., Steiner, K., Walz, H., Soczka-Guth, T., & Sauer, D. U. (2011). Characterization of high-power lithium-ion batteries by electrochemical impedance spectroscopy. II: Modelling. *Journal of Power Sources*, 196(12), 5349-5356. <https://doi.org/10.1016/j.jpowsour.2010.07.071>
- Antonelli, D., Nesi, L., Pepe, G., & Carcaterra, A. (2019, July). A novel control strategy for autonomous cars. In *2019 American Control Conference (ACC)* (pp. 711-716). IEEE. <https://doi.org/10.23919/ACC.2019.8814944>
- Arvind, C. S., & Senthilnath, J. (2019, May). Autonomous RL: Autonomous vehicle obstacle avoidance in a dynamic environment using MLP-SARSA reinforcement learning. In *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)* (pp. 120-124). IEEE. <https://doi.org/10.1109/ICMSR.2019.8835462>
- Arvind, C. S., & Senthilnath, J. (2020). Autonomous vehicle for obstacle detection and avoidance using reinforcement learning. In *Soft Computing for Problem Solving* (pp. 55-66). Springer, Singapore. [https://doi.org/10.1007/978-981-15-0035-0\\_5](https://doi.org/10.1007/978-981-15-0035-0_5)
- Awasthi, N. (2020). Designing of Electric Vehicle Using MATLAB and Simulink. Available at SSRN 3701771. <https://doi.org/10.2139/ssrn.3701771>
- Babak, S. J., Hussain, S. A., Karakas, B., & Cetin, S. (2017, July). Control of autonomous ground vehicles: a brief technical review. In *IOP Conference Series: Materials Science and Engineering* (Vol. 224, No. 1, p. 012029). IOP Publishing. <https://doi.org/10.1088/1757-899x/224/1/012029>

Baboselac, I., Hederić, Ž., & Benšić, T. (2017). MatLab simulation model for dynamic mode of the Lithium-Ion batteries to power the EV. *Tehnički glasnik*, 11(1-2), 7-13. <https://hrcak.srce.hr/183739>

Bacha, S., Saadi, R., Ayad, M. Y., Aboubou, A., & Bahri, M. (2017, March). A review on vehicle modeling and control technics used for autonomous vehicle path following. In *2017 International Conference on Green Energy Conversion Systems (GECS)* (pp. 1-6). IEEE. <https://doi.org/10.1109/GECS.2017.8066221>

Baras, N., Nantzios, G., Ziouzos, D., & Dasygenis, M. (2019, May). Autonomous obstacle avoidance vehicle using lidar and an embedded system. In *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAS)* (pp. 1-4). IEEE. <https://doi.org/10.1109/MOCAS.2019.8742065>

Barcellona, S., & Piegari, L. (2017). Lithium ion battery models and parameter identification techniques. *Energies*, 10(12), 2007. <https://doi.org/10.3390/en10122007>

Bejgam, R. (2021, March). Brief Study and Review on the Next Revolutionary Autonomous Vehicle Technology. In *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)* (pp. 34-37). IEEE. <https://doi.org/10.1109/ICACITE51222.2021.9404763>

Berrueta, A., Urtasun, A., Ursúa, A., & Sanchis, P. (2018). A comprehensive model for lithium-ion batteries: From the physical principles to an electrical model. *Energy*, 144, 286-300. <https://doi.org/10.1016/j.energy.2017.11.154>

Bester, J. E., El Hajjaji, A., & Mabwe, A. M. (2015, November). Modelling of Lithium-ion battery and SOC estimation using simple and extended discrete Kalman Filters for Aircraft energy management. In *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society* (pp. 002433-002438). IEEE. <https://doi.org/10.1109/IECON.2015.7392467>

Bhikkaji, B., & Söderström, T. (2001). Reduced order models for diffusion systems using singular perturbations. *Energy and buildings*, 33(8), 769-781. [https://doi.org/10.1016/S0378-7788\(01\)00071-8](https://doi.org/10.1016/S0378-7788(01)00071-8)

Bin Issa, R., Das, M., Rahman, M., Barua, M., Rhaman, M., Ripon, K. S. N., ... & Rabiul, G. (2021). Double Deep Q-Learning and Faster R-CNN-Based Autonomous Vehicle Navigation and Obstacle Avoidance in Dynamic Environment. *Sensors*, 21(4), 1468. <https://doi.org/10.3390/s21041468>

Bizon, N., Dascalescu, L., & Tabatabaei, N. M. (Eds.). (2014). *Autonomous vehicles: intelligent transport systems and smart technologies*. Nova Science Publishers.

Blundell, M., & Harty, D. (2004). *Multibody systems approach to vehicle dynamics*. Elsevier. <https://doi.org/10.1016/b978-0-7506-5112-7.x5000-3>

Boukerche, A., & Robson, E. (2018). Vehicular cloud computing: Architectures, applications, and mobility. *Computer networks*, 135, 171-189. <https://doi.org/10.1016/j.comnet.2018.01.004>

Brown, M., Funke, J., Erlien, S., & Gerdes, J. C. (2017). Safe driving envelopes for path tracking in autonomous vehicles. *Control Engineering Practice*, 61, 307-316. <https://doi.org/10.1016/j.conengprac.2016.04.013>

Cai, L., & White, R. E. (2008). Reduction of model order based on proper orthogonal decomposition for lithium-ion battery simulations. *Journal of the Electrochemical Society*, 156(3), A154. <https://doi.org/10.1149/1.3049347>

Campbell, S., O'Mahony, N., Krpalcova, L., Riordan, D., Walsh, J., Murphy, A., & Ryan, C. (2018, June). Sensor technology in autonomous vehicles: A review. In *2018 29th Irish Signals and Systems Conference (ISSC)* (pp. 1-4). IEEE. <https://doi.org/10.1109/ISSC.2018.8585340>

Cao, J., Liu, H., Li, P., & Brown, D. (2008). An interval type-2 fuzzy logic controller for quarter-vehicle active suspensions. *Proceedings of the Institution of Mechanical Engineers*,

Part D: *Journal of Automobile Engineering*, 222(8), 1361-1373. <https://doi.org/10.1243/09544070JAUTO767>

Chai, Z., Nie, T., & Becker, J. (2021). *Autonomous Driving Changes the Future*. Springer. <https://doi.org/10.1007/978-981-15-6728-5>

Chen, W., Liang, J., Yang, Z., & Li, G. (2019). A review of lithium-ion battery for electric vehicle applications and beyond. *Energy Procedia*, 158, 4363-4368. <https://doi.org/10.1016/j.egypro.2019.01.783>

Conte, G., Scaradozzi, D., Mannocchi, D., Raspa, P., Panebianco, L., & Screpanti, L. (2018). Development and experimental tests of a ROS multi-agent structure for autonomous surface vehicles. *Journal of Intelligent & Robotic Systems*, 92(3), 705-718. <https://doi.org/10.1007/s10846-017-0700-9>

Dao, T. S., Vyasarayani, C. P., & McPhee, J. (2012). Simplification and order reduction of lithium-ion battery model based on porous-electrode theory. *Journal of Power Sources*, 198, 329-337. <https://doi.org/10.1016/j.jpowsour.2011.09.034>

Dieter, S., Hiller, M., & Baradini, R. (2018). *Vehicle Dynamics: Modeling and Simulation*. [https://doi.org/10.1007/978-3-662-54483-9\\_5](https://doi.org/10.1007/978-3-662-54483-9_5)

Dixit, S., Fallah, S., Montanaro, U., Dianati, M., Stevens, A., Mccullough, F., & Mouzakitis, A. (2018). Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects. *Annual Reviews in Control*, 45, 76-86. <https://doi.org/10.1016/j.arcontrol.2018.02.001>

Doyle, M., Newman, J., Gozdz, A. S., Schmutz, C. N., & Tarascon, J. M. (1996). Comparison of modeling predictions with experimental data from plastic lithium ion cells. *Journal of the Electrochemical Society*, 143(6), 1890. <https://doi.org/10.1149/1.1836921>

Ehsani, M., Wang, F. Y., & Brosch, G. L. (Eds.). (2013). *Transportation Technologies for Sustainability*. Springer. <https://doi.org/10.1007/978-1-4614-5844-9>

Feng, P., Zou, J., Li, H., & Gao, S. (2020, April). An Obstacle Avoidance Method for Autonomous Vehicle in Straight Road Based on Expanded Circle. In *2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)* (pp. 43-46). IEEE. <https://doi.org/10.1109/IPEC49694.2020.9115118>

Forman, J. C., Bashash, S., Stein, J. L., & Fathy, H. K. (2010). Reduction of an electrochemistry-based li-ion battery model via quasi-linearization and pade approximation. *Journal of the Electrochemical Society*, 158(2), A93. <https://doi.org/10.1115/dscc2010-4084>

Gandoman, F. H., Jaguemont, J., Goutam, S., Gopalakrishnan, R., Firouz, Y., Kalogiannis, T., ... & Van Mierlo, J. (2019). Concept of reliability and safety assessment of lithium-ion batteries in electric vehicles: Basics, progress, and challenges. *Applied Energy*, 251, 113343. <https://doi.org/10.1016/j.apenergy.2019.113343>

Goh, J. Y., & Gerdes, J. C. (2016, June). Simultaneous stabilization and tracking of basic automobile drifting trajectories. In *2016 IEEE Intelligent Vehicles Symposium (IV)* (pp. 597-602). IEEE. <https://doi.org/10.1109/IVS.2016.7535448>

Guruswamy, D., Divyashree, D. P., Karn, M., & Mishra, G. (2018, December). Algorithm for Autonomous Vehicle Control. In *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)* (pp. 1-6). IEEE. <https://doi.org/10.1109/ICNEWS.2018.8903966>

Gutiérrez, R., López-Guillén, E., Bergasa, L. M., Barea, R., Pérez, Ó., Gómez-Huélamo, C., ... & López-Fernández, J. (2020). A waypoint tracking controller for autonomous road vehicles using ROS framework. *Sensors*, 20(14), 4062. <https://doi.org/10.3390/s20144062>

Haizhou, Z. (2017). Modeling of Lithium-ion Battery for Charging/Discharging Characteristics Based on Circuit Model. *International Journal of Online Engineering*, 13(6). <https://doi.org/10.3991/ijoe.v13i06.6799>

Hannan, M. A., Hoque, M. M., Hussain, A., Yusof, Y., & Ker, P. J. (2018). State-of-the-art and energy management system of lithium-ion batteries in electric vehicle applications: Issues and recommendations. *Ieee Access*, 6, 19362-19378. <https://doi.org/10.1109/ACCESS.2018.2817655>

Hitesh, & Sunanda, C. (2020). Modeling and Performance Analysis of an Electric Vehicle with MATLAB/Simulink. <https://www.irjet.net/archives/V7/i6/IRJET-V7I6204.pdf>

Hoermann, S., Stumper, D., & Dietmayer, K. (2017, June). Probabilistic long-term prediction for autonomous vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)* (pp. 237-243). IEEE. <https://doi.org/10.1109/IVS.2017.7995726>

How, D. N., Hannan, M. A., Lipu, M. H., & Ker, P. J. (2019). State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review. *Ieee Access*, 7, 136116-136136. <https://doi.org/10.1109/ACCESS.2019.2942213>

Hu, J., Bhowmick, P., Arvin, F., Lanzon, A., & Lennox, B. (2020). Cooperative control of heterogeneous connected vehicle platoons: An adaptive leader-following approach. *IEEE Robotics and Automation Letters*, 5(2), 977-984. <https://doi.org/10.1109/LRA.2020.2966412>

Hu, X., Stanton, S., Cai, L., & White, R. E. (2012). Model order reduction for solid-phase diffusion in physics-based lithium ion cell models. *Journal of Power Sources*, 218, 212-220. <https://doi.org/10.1016/j.jpowsour.2012.07.007>

Huh, K., Kim, J., & Hong, J. (2000). Handling and driving characteristics for six-wheeled vehicles. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 214(2), 159-170. <https://doi.org/10.1177/095440700021400205>

Isele, D. (2019, October). Interactive decision making for autonomous vehicles in dense traffic. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (pp. 3981-3986). IEEE. <https://doi.org/10.1109/ITSC.2019.8916982>

Jazar, R. N. (2017). *Vehicle dynamics: theory and application*. Springer.[https://doi.org/10.1007/978-3-319-53441-1\\_2](https://doi.org/10.1007/978-3-319-53441-1_2)

Jian, Z., Zhang, S., Chen, S., Lv, X., & Zheng, N. (2019, June). High-definition map combined local motion planning and obstacle avoidance for autonomous driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)* (pp. 2180-2186). IEEE.<https://doi.org/10.1109/IVS.2019.8814229>

Jiang, J., & Zhang, C. (2015). *Fundamentals and applications of lithium-ion batteries in electric drive vehicles*. John Wiley & Sons.<https://doi.org/10.1002/9781118414798>

Jongerden, M. R., & Haverkort, B. R. (2009). Which battery model to use?. *IET software*, 3(6), 445-457. <https://doi.org/10.1049/iet-sen.2009.0001>

Julien, C., Mauger, A., Vijh, A., & Zaghib, K. (2016). Lithium batteries. In *Lithium batteries* (pp. 29-68). Springer, Cham.[https://doi.org/10.1007/978-3-319-19108-9\\_2](https://doi.org/10.1007/978-3-319-19108-9_2)

Jurgen, R.K. (2013). Autonomous Vehicles for Safer Driving.<https://doi.org/10.4271/PT-158>

Khayyat, M., Elgendy, I. A., Muthanna, A., Alshahrani, A. S., Alharbi, S., & Koucheryavy, A. (2020). Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access*, 8, 137052-137062.<https://doi.org/10.1109/ACCESS.2020.3011705>

Khodayari, A., Ghaffari, A., Ameli, S., & Flahatgar, J. (2010, September). A historical review on lateral and longitudinal control of autonomous vehicle motions. In *2010 International Conference on Mechanical and Electrical Technology* (pp. 421-429). IEEE. <https://doi.org/10.1109/ICMET.2010.5598396>

Kim, C., & Ro, P. I. (2002). An accurate full car ride model using model reducing techniques. *J. Mech. Des.*, 124(4), 697-705.<https://doi.org/10.1115/1.1503065>

- Kim, Y. H., & Ha, H. D. (1997). Design of interface circuits with electrical battery models. *IEEE transactions on Industrial electronics*, 44(1), 81-86. <https://doi.org/10.1109/41.557502>
- Kiyakli, A. O., & Solmaz, H. (2018). Modeling of an electric vehicle with MATLAB/Simulink. *International journal of automotive science and technology*, 2(4), 9-15. <https://doi.org/10.30939/ijastech..475477>
- Kortüm, W. (1993). Review of multibody computer codes for vehicle system dynamics. *Vehicle System Dynamics*, 22(S1), 3-31. <https://doi.org/10.1080/00423119308969463>
- Kuhne, F., Lages, W. F., & da Silva Jr, J. G. (2004, September). Model predictive control of a mobile robot using linearization. In *Proceedings of mechatronics and robotics* (pp. 525-530). [http://www.ece.ufrgs.br/~fetter/mechrob04\\_553.pdf](http://www.ece.ufrgs.br/~fetter/mechrob04_553.pdf)
- Laghmara, H., Boudali, M. T., Laurain, T., Ledy, J., Orjuela, R., Lauffenburger, J. P., & Basset, M. (2019, June). Obstacle avoidance, path planning and control for autonomous vehicles. In *2019 IEEE Intelligent Vehicles Symposium (IV)* (pp. 529-534). IEEE. <https://doi.org/10.1109/IVS.2019.8814173>
- Lee, J. L., Aldrich, L. L., Stetzel, K. D., & Plett, G. L. (2014). Extended operating range for reduced-order model of lithium-ion cells. *Journal of Power Sources*, 255, 85-100. <https://doi.org/10.1016/j.jpowsour.2013.12.134>
- Lee, J. L., Chemistruck, A., & Plett, G. L. (2012). Discrete-time realization of transcendental impedance models, with application to modeling spherical solid diffusion. *Journal of Power Sources*, 206, 367-377. <https://doi.org/10.1016/j.jpowsour.2012.01.134>
- Lee, J. L., Chemistruck, A., & Plett, G. L. (2012). One-dimensional physics-based reduced-order model of lithium-ion dynamics. *Journal of Power Sources*, 220, 430-448. <https://doi.org/10.1016/j.jpowsour.2012.07.075>

Leman, Z. A., Ariff, M. H. M., Zamzuri, H., Rahman, M. A. A., & Mazlan, S. A. (2019, June). Model predictive controller for path tracking and obstacle avoidance manoeuvre on autonomous vehicle. In *2019 12th Asian Control Conference (ASCC)* (pp. 1271-1276). IEEE. <https://ieeexplore.ieee.org/abstract/document/8765109>

Li, J., Wang, S., Fernandez, C., Wang, N., & Xie, H. (2017, December). The battery management system construction method study for the power lithium-ion battery pack. In *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)* (pp. 285-289). IEEE. <https://doi.org/10.1109/ICRAE.2017.8291396>

Li, L., Ota, K., & Dong, M. (2018). Humanlike driving: Empirical decision-making system for autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 67(8), 6814-6823. <https://doi.org/10.1109/TVT.2018.2822762>

Li, Y., Liang, C., & Wang, S. (2015, April). Study of Vehicle Handling Stability Based on ADAMS/Car. In *2015 International Conference on Automation, Mechanical Control and Computational Engineering*. Atlantis Press. <https://doi.org/10.2991/amcce-15.2015.326>

Liu, J., Jayakumar, P., Overholt, J. L., Stein, J. L., & Ersal, T. (2013, October). The role of model fidelity in model predictive control based hazard avoidance in unmanned ground vehicles using LIDAR sensors. In *Dynamic Systems and Control Conference* (Vol. 56147, p. V003T46A005). American Society of Mechanical Engineers. <https://doi.org/10.1115/DSCC2013-4021>

Liu, S., Li, L., Tang, J., Wu, S., & Gaudiot, J. L. (2017). Creating autonomous vehicle systems. *Synthesis Lectures on Computer Science*, 6(1), i-186. <https://doi.org/10.2200/S00787ED1V01Y201707CSL009>

Liu, S., Tang, J., Wang, C., Wang, Q., & Gaudiot, J. L. (2017). A unified cloud platform for autonomous driving. *Computer*, 50(12), 42-49. <https://doi.org/10.1109/MC.2017.4451224>

Liu, T., Huang, B., Deng, Z., Wang, H., Tang, X., Wang, X., & Cao, D. (2020). Heuristics-oriented overtaking decision making for autonomous vehicles using reinforcement

learning. *IET Electrical Systems in Transportation*, 10(4), 417-424.<https://doi.org/10.1049/iet-est.2020.0044>

Lu, L., Han, X., Li, J., Hua, J., & Ouyang, M. (2013). A review on the key issues for lithium-ion battery management in electric vehicles. *Journal of power sources*, 226, 272-288.<https://doi.org/10.1016/j.jpowsour.2012.10.060>

Ma, Z., Wang, Z., Xiong, R., & Jiang, J. (2018). A mechanism identification model based state-of-health diagnosis of lithium-ion batteries for energy storage applications. *Journal of Cleaner Production*, 193, 379-390.<https://doi.org/10.1016/j.jclepro.2018.05.074>

Macek, K., Thoma, K., Glatzel, R., & Siegwart, R. (2007). Dynamics modeling and parameter identification for autonomous vehicle navigation. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3321-3326). IEEE.<https://doi.org/10.1109/IROS.2007.4399640>

MapleSoft (2021). Case Study: High Fidelity, High Performance Battery Models Help Manufacturers Build Greener Cars. Retrieved from [https://www.maplesoft.com/company/casestudies/Stories/green\\_cars.aspx](https://www.maplesoft.com/company/casestudies/Stories/green_cars.aspx)

Martin, J., Casquero, O., Fortes, B., & Marcos, M. (2019). A generic multi-layer architecture based on ROS-JADE integration for autonomous transport vehicles. *Sensors*, 19(1), 69.<https://doi.org/10.3390/s19010069>

McHenry, R. R. (1984). An analysis of the dynamics of automobiles during simultaneous cornering and ride motions. *Computer-Aided Design*, 1(3), 19-32.[https://doi.org/10.1016/S0010-4485\(69\)80082-5](https://doi.org/10.1016/S0010-4485(69)80082-5)

Miao, Y., Hynan, P., Von Jouanne, A., & Yokochi, A. (2019). Current Li-ion battery technologies in electric vehicles and opportunities for advancements. *Energies*, 12(6), 1074.<https://doi.org/10.3390/en12061074>

Miura, K., Tokunaga, S., Horita, Y., Oda, Y., & Azumi, T. (2021). CoSAM: Co-Simulation Framework for ROS-based Self-driving Systems and MATLAB/Simulink. *Journal of Information Processing*, 29, 227-235. <https://doi.org/10.2197/ipsjip.29.227>

Odhams, A. M., Roebuck, R. L., Cebon, D., & Winkler, C. B. (2008). Dynamic safety of active trailer steering systems. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of multi-body dynamics*, 222(4), 367-380. <https://doi.org/10.1243/14644193JMBD174>

Özgüner, Ü., Acarman, T., & Redmill, K. A. (2011). *Autonomous ground vehicles*. Artech House.

Pappalardo, C. M., Lombardi, N., & Guida, D. (2020). A Model-Based System Engineering Approach for the Virtual Prototyping of an Electric Vehicle of Class L7. *Engineering Letters*, 28(1). [http://www.engineeringletters.com/issues\\_v28/issue\\_1/EL\\_28\\_1\\_26.pdf](http://www.engineeringletters.com/issues_v28/issue_1/EL_28_1_26.pdf)

Petrovic, S. (2021). Lithium Batteries. In *Battery Technology Crash Course* (pp. 93-112). Springer, Cham. [https://doi.org/10.1007/978-3-030-57269-3\\_6](https://doi.org/10.1007/978-3-030-57269-3_6)

Piao, J., & McDonald, M. (2008). Advanced driver assistance systems from autonomous to cooperative approach. *Transport reviews*, 28(5), 659-684. <https://doi.org/10.1080/01441640801987825>

Rafaila, R. C., & Livint, G. (2015, October). Nonlinear model predictive control of autonomous vehicle steering. In *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 466-471). IEEE. <https://doi.org/10.1109/ICSTCC.2015.7321337>

Rajamani, R. (2011). *Vehicle dynamics and control*. Springer Science & Business Media. <https://doi.org/10.1007/978-1-4614-1433-9>

Rakhmatov, D., Vrudhula, S., & Wallach, D. A. (2002, August). Battery lifetime prediction for energy-aware computing. In *Proceedings of the 2002 international symposium on Low power electronics and design* (pp. 154-159).<https://doi.org/10.1145/566408.566449>

Raviteja, T. (2020). An introduction of autonomous vehicles and a brief survey. *Journal of Critical Reviews*, 7(13), 196-202. <https://doi.org/10.31838/jcr.07.13.33>

SAE International. "Automated Driving Levels of Driving Automation are Defined in New SAE International Standard J3016." (2014).<https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic>

Salem, F. A. (2013). Modeling and control solutions for electric vehicles. *European Scientific Journal*, 9(15).<https://core.ac.uk/download/pdf/236412713.pdf>

Sanchez, I., D'Jorge, A., Raffo, G. V., González, A. H., & Ferramosca, A. (2021). Nonlinear model predictive path following controller with obstacle avoidance. *Journal of Intelligent & Robotic Systems*, 102(1), 1-18.<https://doi.org/10.1007/s10846-021-01373-7>

Schwarting, W., Alonso-Mora, J., & Rus, D. (2018). Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1, 187-210.<https://doi.org/10.1146/annurev-control-060117-105157>

Shabana, A. (2020). *Dynamics of multibody systems*. Cambridge university press.<https://doi.org/10.1017/CBO9781107337213>

Smith, K. A., Rahn, C. D., & Wang, C. Y. (2008). Model order reduction of 1D diffusion systems via residue grouping.<https://doi.org/10.1115/1.2807068>

Subramanian, V. R., Boovaragavan, V., & Diwakar, V. D. (2007). Toward real-time simulation of physics based lithium-ion battery models. *Electrochemical and Solid State Letters*, 10(11), A255.<https://doi.org/10.1149/1.2776128>

Sudworth, J. L. (2001). The sodium/nickel chloride (ZEBRA) battery. *Journal of power sources*, 100(1-2), 149-163.[https://doi.org/10.1016/S0378-7753\(01\)00891-6](https://doi.org/10.1016/S0378-7753(01)00891-6)

Sun, F., Cheng, N., Zhang, S., Zhou, H., Gui, L., & Shen, X. (2018, December). Reinforcement learning based computation migration for vehicular cloud computing. In *2018 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.<https://doi.org/10.1109/GLOCOM.2018.8647996>

Taeihagh, A., & Lim, H. S. M. (2019). Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks. *Transport reviews*, 39(1), 103-128.<https://doi.org/10.1080/01441647.2018.1494640>

Tesla (2021). URL: <https://www.tesla.com/gigafactory>

Tsai, J. S. H., Juang, J. C., Tu, C. H., Tsai, T. Y., Chung, P. C., Hsu, C. C., ... & Lin, C. F. (2019, November). Development of key technologies for autonomous driving vehicles. In *2019 International Automatic Control Conference (CACS)* (pp. 1-6). IEEE.<https://doi.org/10.1109/CACS47674.2019.9024730>

Wang, P., Gao, S., Li, L., Sun, B., & Cheng, S. (2019). Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. *Energies*, 12(12), 2342.<https://doi.org/10.3390/en12122342>

Wen, J., Yu, Y., & Chen, C. (2012). A review on lithium-ion batteries safety issues: existing problems and possible solutions. *Materials express*, 2(3), 197-212.<https://doi.org/10.1166/mex.2012.1075>

Xiong, R., Li, L., & Tian, J. (2018). Towards a smarter battery management system: A critical review on battery state of health monitoring methods. *Journal of Power Sources*, 405, 18-29.<https://doi.org/10.1016/j.jpowsour.2018.10.019>

Xiong, R., Li, L., Li, Z., Yu, Q., & Mu, H. (2018). An electrochemical model based degradation state identification method of Lithium-ion battery for all-climate electric vehicles application. *Applied energy*, 219, 264-275. <https://doi.org/10.1016/j.apenergy.2018.03.053>

Xiong, R., Sun, F., Chen, Z., & He, H. (2014). A data-driven multi-scale extended Kalman filtering based parameter and state estimation approach of lithium-ion polymer battery in electric vehicles. *Applied Energy*, 113, 463-476. <https://doi.org/10.1016/j.apenergy.2013.07.061>

Xiong, R. (2020). Battery Management Algorithm for Electric Vehicles. *Springer*. <https://doi.org/10.1007/978-981-15-0248-4>

Yang, S. M., & Lin, Y. A. (2021). Development of an Improved Rapidly Exploring Random Trees Algorithm for Static Obstacle Avoidance in Autonomous Vehicles. *Sensors*, 21(6), 2244. <https://doi.org/10.3390/s21062244>

Yeong, D. J., Velasco-Hernandez, G., Barry, J., & Walsh, J. (2021). Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6), 2140. <https://doi.org/10.3390/s21062140>

Zhang, J., & Letaief, K. B. (2019). Mobile edge intelligence and computing for the internet of vehicles. *Proceedings of the IEEE*, 108(2), 246-261. <https://doi.org/10.1109/JPROC.2019.2947490>

Zhao, J., Liang, B., & Chen, Q. (2018). The key technology toward the self-driving car. *International Journal of Intelligent Unmanned Systems*. <https://doi.org/10.1108/ijius-08-2017-0008>

Zofka, M. R., Töttel, L., Zipfl, M., Heinrich, M., Fleck, T., Schulz, P., & Zöllner, J. M. (2020, September). Pushing ROS towards the Dark Side: A ROS-based Co-Simulation Architecture for Mixed-Reality Test Systems for Autonomous Vehicles. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)* (pp. 204-211). IEEE. <https://doi.org/10.1109/MFI49285.2020.9235238>