République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Minisère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
Ecole Nationale Polytechnique

Département d'Electronique
Laboratoire de Communication et de Conversion Photovoltaïque

End of Studies Project Thesis
submitted for the fulfillment of:
State Engineer Degree in Electronics

Study, Modelling, Design and Realisation of a Zero-Emission Vehicle: Transformation of the Mechanical Power Train into Electrical of the KIA Pride 2000 Model

Authors:
Aymen MEHDI - Mohamed Alla Eddine BAHI
Under the supervision of Dr. Cherif LARBES Prof. ENP, Algiers

Presented and defended on June 20th, 2023 before the members of jury:

President	Mourad	HADDADI	Prof.	ENP, Algiers
Supervisor	Cherif	LARBES	Prof.	ENP, Algiers
Examiner	Mohammed O.	TAGHI	MAA	ENP, Algiers
Guest	Arezki	SMAILI	Prof.	ENP, Algiers

ENP 2023

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Minisère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
Ecole Nationale Polytechnique

Département d'Electronique
Laboratoire de Communication et de Conversion Photovoltaïque

End of Studies Project Thesis
submitted for the fulfillment of:
State Engineer Degree in Electronics

Study, Modelling, Design and Realisation of a Zero-Emission Vehicle: Transformation of the Mechanical Power Train into Electrical of the KIA Pride 2000 Model

Authors:
Aymen MEHDI - Mohamed Alla Eddine BAHI
Under the supervision of Dr. Cherif LARBES Prof. ENP, Algiers

Presented and defended on June 20th, 2023 before the members of jury:

President	Mourad	HADDADI	Prof.	ENP, Algiers
Supervisor	Cherif	LARBES	Prof.	ENP, Algiers
Examiner	Mohammed O.	TAGHI	MAA	ENP, Algiers
Guest	Arezki	SMAILI	Prof.	ENP, Algiers

ENP 2023

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Minisère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
Ecole Nationale Polytechnique

Département d'Electronique
Laboratoire de Communication et de Conversion Photovoltaïque

Mémoire du Projet de Fin d'Études
Pour l'obtention du diplôme:
Ingénieur d'Etat en Electronique

Etude, Modélisation, Conception, et Réalisation d'un Véhicule à Zéro Emission : Transformation de la Motorisation Mécanique en Electrique du Modèle KIA Pride 2000

Réaliser par:
Aymen MEHDI - Mohamed Alla Eddine BAHI
Sous la supervision de Dr. Cherif LARBES Prof. ENP, Alger

Présenté et défendu le 20 juin 2023 devant les membres du jury :

Président	Mourad	HADDADI	Prof.	ENP, Alger
Encadreur	Cherif	LARBES	Prof.	ENP, Alger
Examinateur	Mohammed O.	TAGHI	MAA	ENP, Alger
Invité	Arezki	SMAILI	Prof.	ENP, Alger

ENP 2023

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

ملخص

بحلول عام 2030، حولت الحكومات تركيزها إلى المركبات الكهربائية (EVs). وبالتالي، يركز هذا المشروع على تحويل نموذج Pride KIA 2000 إلى EV من خلال تحسين وحدة التحكم الإلكترونية. الهدف هو تطوير نظام دفع فعال للمركبة الكهربائية. لتحقيق التحكم الدقيق في سرعة المحرك الالمتزامن، يتم استخدام خوارزمية تغييرعرض الدفع المصحوبة بتقنية حذف المركبات التوافقية SHE PWM .ومع ذلك، نظرًا لطبيعة التقنيات التي تستغرق وقتًا طويلاً لحساب زوايا التبديل، فإن خوارزمية SHE PWM غير عملية للتطبيقات في الوقت الفعلي. للتغلب على هذا التحدي، تتم مقارنة نهجين، أحدهما باستخدام الشبكات العصبية الاصطناعية ANN والآخر حول استيفاء متعدد الحدود PI، وكلاهما بالاشتراك مع خوارزمية SHE PWM .تصف هذه الأطروحة، وتظهر برمجة كلا الخوارزميتين في متحكم دقيق لتقييم دقة وسرعة كلتا الطريقتين. تظهر النتائج تفوق نهج ANN. للتحقق من صحة الخوارزمية في تطبيق في الوقت الفعلي، يتم تقديم ومناقشة برمجة الطريقة المقترحة على بطاقة FPGA. يتم اختبار التطبيق بمنصة تجارب محرك لا متزامن متغير السرعة. تشير النتائج التي تم الحصول عليها إلى أن خوارزمية PWM ANNSHEتتحكم بكفاءة في الجهد الأساسي، مما يلغي التوافقيات المطلوبة في الوقت الفعلي عبر النطاق الكامل لتغيرات السرعة.

كلمات مفاتيح : SHE PWM , الشبكات العصبية ,استيفاء متعدد الحدود , بطاقة , FPGA ,السيارة الكهربائية, المحرك الالمتزامن.

Résumé

En 2030, plusieurs gouvernements se sont tournés vers les véhicules électriques (EV). Par conséquent, Ce projet vise à transformer un modèle KIA Pride 2000 en EV en optimisant l'unité de commande électronique. L'objectif est de développer un système de propulsion efficace et fiable pour l'EV. Pour obtenir un contrôle précis de la vitesse du moteur asynchrone, un algorithme connu sous le nom de modulation de largeur d'impulsion avec élimination harmonique sélective (SHE PWM) est utilisé. Cependant, en raison de la lenteur des techniques numériques requises pour calculer les angles de commutation, l'algorithme SHE PWM est peu pratique pour les applications en temps réel. Pour surmonter ce défi, deux approches sont comparées, l'une utilisant les réseaux neuronaux artificiels (ANN) et l'autre sur l'interpolation polynomiale (PI), les deux en combinaison avec l'algorithme SHE PWM. Cette thèse décrit et implémente les deux algorithmes dans un microcontrôleur pour évaluer la précision et la vitesse des deux méthodes. Les résultats démontrent la supériorité de l'approche ANN. Pour valider l'algorithme dans une application en temps réel, une implémentation FPGA est présentée et discutée. L'application est testée sur un banc d'essai à moteur à induction à vitesse variable. Les résultats obtenus indiquent que l'algorithme ANNSHE PWM contrôle efficacement la tension fondamentale, éliminant les harmoniques souhaitées en temps réel sur toute la gamme des variations de vitesse.

Mots clés: SHE PWM, Réseaux Neuronaux Artificiels, Interpolation Polynomiale, FPGA, Véhicule Electrique, Moteur asynchrone.

Abstract

By 2030, many governments have shifted their focus to electrical vehicles (EVs). Consequently, this project focuses on transforming a KIA Pride 2000 model to an EV by optimizing the electronic control unit. The objective is to develop an efficient and reliable propulsion system for the EV. To achieve precise speed control of the induction motor, an algorithm known as Pulse Width Modulation with Selective Harmonic Elimination (SHE PWM) is used. However, due to the time-consuming nature of the numerical techniques required for calculating switching angles, the SHE PWM algorithm is impractical for real-time applications. To overcome this challenge, two approaches are compared, one using Artificial Neural Networks (ANN) and the other on Polynomial Interpolation (PI), both in combination with the SHE PWM algorithm. This thesis describes, and implements both algorithms into a microcontroller to evaluate the accuracy and speed of both methods. The results demonstrate the superiority of the ANN approach. To validate the algorithm in a real-time application, an FPGA implementation is presented and discussed. The application is tested on a variable speed induction motor test bench. The obtained results indicate that the ANNSHE PWM algorithm efficiently controls the fundamental voltage, eliminating the desired harmonics in real-time across the entire range of speed variations.

Key words: SHE PWM, Artificial Neural Networks (ANN), Polynomial Interpolation (PI), FPGA, Electric Vehicle, Asynchronous Motor.

Dedication

“

To our dear family,

To our special friends,

To all those who are dear to us,

To all of you,

Thank you.

”

Aymen MEHDI - Mohamed Alla Eddine BAHI

Acknowledgment

First and foremost, we thank Allah, the Almighty, for granting us the determination to accomplish this work.

Our heartfelt appreciation goes out to our parents, whose unwavering support has been the cornerstone of our journey. Their guidance have left an indelible mark on our lives, and for that, we will forever remain thankful.

Furthermore, We would like to take this opportunity to express our deepest gratitude and appreciation towards our project supervisor, Pr. Cherif LARBES, a distinguished Professor at the National Polytechnic School. His consistent guidance and unwavering support throughout our work in the Laboratory of Photovoltaic Communication and Conversion Devices have been invaluable. We are truly grateful for his kindness, encouragement, and constant availability whenever we needed assistance. Moreover, the knowledge he shared with us abundantly and the passion for engineering he gave us inspired us with great ease and inspiration.

We would also like to express our gratitude to the members of the jury, particularly to the President of the jury, Pr. Mourad HADDADI, a respected Professor at the National Polytechnic School, and the Examiner, Mr. Mohamed Oussaid Taghi, a dedicated Teacher Researcher at the National Polytechnic School. Their willingness to serve on the reading committee and their constructive analysis of our work deserve our special appreciation. We are indebted to them for generously devoting their time, imparting their knowledge, and providing unwavering support throughout our three-year-long academic journey.

Last but not least, we would like to acknowledge the invaluable contributions of all our teachers, professors and fellow graduate electronic students. It is with deep gratitude that we recognize each and every one of you, as you have played a significant role in shaping our path and propelling us to where we stand today.

Contents

List of Tables

List of Figures

List of abbreviations

EV	*Electric vehicle.*
EPS	*Electric propulsion system.*
DC	*Direct current.*
AC	*Alternating current.*
PEV	*Pure electric vehicle.*
HEV	*Hybrid electric vehicle.*
FCEV	*Fuel cell electric vehicle.*
ICEV	*Internal combustion engine vehicle.*
EMU	*Energy management unit.*
ASM	*Asynchronous motor.*
IM	*Induction Motor.*
SHE	*Selective Harmonic Elemination.*
PWM	*Pulse Width Modulation.*
EMF	*Electromotive force.*
ANN	*Artificial Neural Network.*
MLP	*Multi Layer Perceptron.*
PI	*Polynomial Interpretation.*
FPGA	*Field Programmable Gate Array.*

VHDL	*LVHSIC Hardware Description Language.*
ASIC	*Application Specific Integrated Circuit.*
DSP	*Digital Signal Processor.*
FIL	*FPGA in the loop.*

General Introduction

The automotive industry is at a crucial crossroads, driven by the pressing need to address the alarming issues of excessive energy consumption, fossil resource depletion, pollution, and global warming. Some of these issues are directly attributed to combustion-engine vehicles.Many Governments have set targets to phase out the production of Internal Combustion Engine Vehicles (ICEVs) by 2030 [1, 2, 3]. In response, both researchers and automakers have intensified their efforts towards electric vehicles (EVs). These vehicles offer a promising alternative to build a safer, cleaner, more sustainable and intelligent mode of transportation.

The main objective of this research is the development of an EV that encompasses these desirable characteristics. By advancing existing technologies, aiming to create an innovative solution that addresses the challenges posed by traditional vehicles. This end of studies project thesis focuses on the study, modelling, design and transformation of an ICEV, specifically a KIA Pride 2000 model, into an electric vehicle with the intention of contributing to the advancement of EVs within our school. The project represents a collaborative effort, with mechanical engineering students undertaking the mechanical modifications, while our team, the electronic engineering team, takes responsibility for the electrical propulsion system. As part of this pursuit, our focus lies specifically in the field of power electronics and motor drive, which plays a pivotal role in enhancing the overall performance of EVs, including their range [4, 5].

The heart of the EV propulsion system lies in the engine drive, comprising an electric motor, a power converter, and an electronic control unit [6]. It is this electronic control unit that serves as the central focus of this thesis, as we aim to develop and optimise its functionality. By doing so, we seek to improve the efficiency, precision, and reliability of the EV's propulsion system.

When considering the choice of an electric motor for the transformed vehicle, we evaluated various options and ultimately decided to use an asynchronous motor over a DC motor. This decision was driven by the asynchronous motor's notable advantages, including its robustness, lower maintenance, higher power to weight and lower cost to power that made it a more suitable choice for our application [7, 8]. However, for the asynchronous motor to effectively compete with the DC motor, it is necessary to use an efficient and low-cost voltage inverter. Furthermore, to control the speed of this asynchronous motor, a three-phase inverter with a variable sine output in voltage and frequency is required.

Numerous control strategies are available but when they are implemented in practical applications, harmonics can arise in the variable output of the three-phase inverter [9, 10, 11]. Harmonics are unwanted frequencies that deviate from the fundamental frequency of the system. These harmonics have various effects on the asynchronous motor. They lead to increased heating, potentially reducing motor lifespan. Furthermore, they cause torque ripple, leading to vibrations and noise, while also increasing losses in the motor and reducing its efficiency.

To ensure efficient control and optimal performance of the asynchronous motor, we have employed the Selective Harmonic Elimination Pulse Width Modulation (SHE PWM) algorithm, which was originally developed by Patel and Hoft [12, 13]. This algorithm utilises switching moments to adjust the duration of pulses in the output waveform of the inverter, aiming to closely resemble a desired sinusoidal waveform with an amplitude A and a frequency f. By carefully selecting the pulse widths, specific harmonics can be targeted for removal, ensuring smooth and efficient motor operation, particularly when employed in an EV.

However, the equations involved in determining the appropriate switching instants are non-linear, This necessitates the utilisation of a numerical calculation technique like the Newton-Raphson method; but in cases where the initial values are not chosen accurately, this can result in multiple iteration cycles, and in some instances, the solution may not converge at all, making real-time implementation for the SHEPWM command computationally demanding, impractical and time consuming. As a result, the SHE PWM control strategy is primarily limited to offline use.

In our application, the speed of EVs experiences significant variations, requiring frequent adjustments in frequency and voltage. Consequently, it is essential to develop a command system that is both fast and reliable, also capable of performing on-line calculations of the switching angles. This system should operate in real-time and offer precise control over the motor's operation and speed.

To achieve this objective, as mentioned in [14] numerous researchers have explored alternative solutions. Following the same approach, in this work, we explored two distinct approaches. The first approach is an ANNSHE PWM algorithm, which is based on the theory of Artificial Neural Networks (ANN) developed by [14]. The second approach is a novel algorithm that we have developed utilising Polynomial Interpolation theory. Both of these approaches propose the utilisation of SHE PWM control to calculate the switching angles and generate on-line, real-time PWM control signals. To compare the performance of these algorithms, we have implemented them on a microcontroller.

By conducting a comprehensive comparison between the two methods, we determined that the Artificial Neural Network approach offered superior high-speed performance with good precision, making it as the most suitable for our work, ensuring the desired level of accuracy and responsiveness. To validate the effectiveness of the ANNSHE PWM algorithm, we have implemented it on an FPGA circuit.

In order to achieve our objectives, this work is divided into five chapters:

The first chapter provides a general history of EVs and their definition, along with an overview of the electric propulsion system. It discusses the fundamental concepts and components of EVs, including their general configuration.

In the second chapter, the focus is on asynchronous machines and voltage inverters. It explores the principles and workings of asynchronous machines, as well as the function and operation of voltage inverters. Then, we detail the principle of the SHE PWM command and the database generation.

The third chapter introduces both algorithms based on Artificial Neural Network (ANN) and Polynomial Interpolation (PI). It provides the foundations of both techniques and describes the development steps for each algorithm. Furthermore, a comparison is made between the two algorithms to determine the preferred choice for implementation.

Chapter four focuses on optimising the implementation of the ANNSHE PWM algorithm on an FPGA circuit. It details each part of the VHDL code and explains how it contributes to improving the performance of the system.

In the fifth chapter, the functionality of the system is verified, ensuring that it meets the specified requirements for the EV propulsion system. The performance of the system is then validated in real-world conditions by linking the command unit, the three-phase inverter, and the asynchronous motor. This chapter provides a practical assessment of the system's performance and its effectiveness in a real-world setting.

Chapter 1

Electric Vehicles

1.1 Introduction

Environmental protection and energy conservation are the main concerns of the 21st century which has now accelerated the pace to plan and develop electric vehicles technology. The EVs offer a zero emission, new automobile industry establishment, and economic development, an efficient and smart transportation system compared to conventional vehicles.

The continual development of EV technologies is the crucial factor to improve EVs performance and ensure its competitiveness, and with the push toward EVs, many predict that by 2030 electronics could account for up to 50 percent of a vehicle's value. EV have increasingly become popular in the automotive market to reduce the dependence of transport on oil, as in less than a decade, the market for electric vehicles has grown by almost a factor of twenty [2, 3].

Although electric vehicles offer numerous benefits, there are still some challenges to overcome. Even with the increased autonomy of EVs, this problem is still important and the limited driving range of electric vehicles needs to be increased to match the range of conventional vehicles. The cost of electric vehicles needs to be reduced to make them more accessible to consumers. There is also a need for more charging infrastructure to support the growing number of electric vehicles on the road [15].

In fact, the major concerns facing the electric vehicle industry are range, top speed, and cost. Ultimately the challenging technology is the battery or energy storage in general, which requires more attention to increase the range of EVs [16]. On the other hand, power electronics and controlling motor drive in particular, which are the subject of the application part of this final study project represent the fundamental technology of the EPS, which should be taken into account in order to improve the entire performance of EVs including autonomy and efficiency [4, 5].

In this chapter, we will provide an overview of the present status of EVs, starting with a brief history of electric vehicles. Subsequently, we will outline some of the advantages and limitations of EVs. We will then talk about its different characteristics such as its constitution and its operating principle.

1.2 History of EVs

While electric vehicles are often considered a modern technology, they have actually been around for quite a long time [2]. The history of electric vehicles began in Europe at the beginning of the 19th century. After the invention of the primary battery by Volta in 1800 and the demonstration of the principle of the electric motor in 1821 by Faraday, The first practical electric vehicle was developed by Scottish inventor Robert Anderson in 1830. The small vehicle operated on a non-rechargeable battery and managed to travel a short rail journey [6].

In 1859, the Frenchman Gaston Planté invented the secondary lead/acid rechargeable battery and in 1869 Gramme built the first electric DC motor with direct current having a power of more than one horse. Twelve years later, in 1881, Gustave Trouvé built the first electric vehicle powered by a secondary battery 1.1. The vehicle was powered by a 0.1 horsepower DC electric motor and weighed 160 kg with its driver. According to [6] it was preceded by Sir David Salomons who built an EV with a rechargeable battery in 1874. In 1881, Camille Faure improved the model of Gaston Planté. In 1884, newly unearthed photos show Thomas Parker sitting in what appears to be an electric vehicle which could be the first in the world.

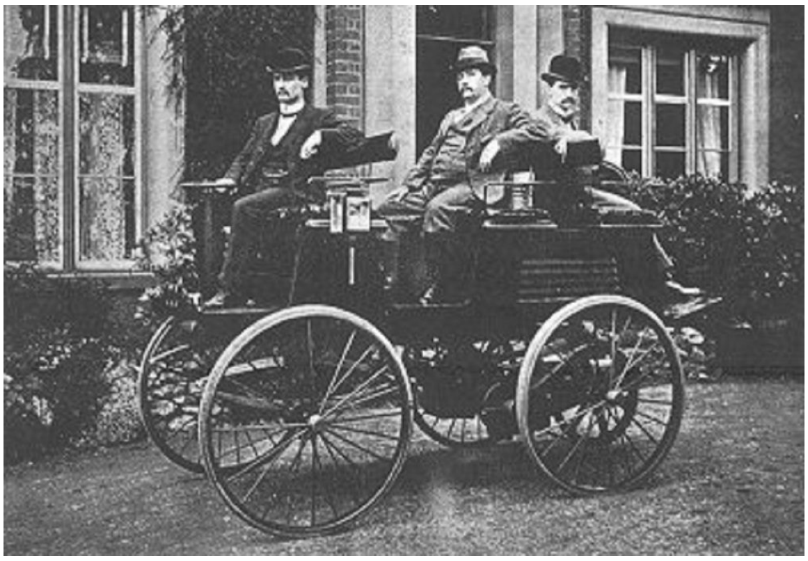


Figure 1.1: First electric vehicle built by Gustave Trouvéc

Later, in 1885, the Germans Daimler and Benz invented the first gasoline car. The early achievements did not attract public attention due to their still immature technology that could not compete with horse cars. But things changed quickly and in the years that followed the race was launched, electric vehicles competed with thermal vehicles but also with steam vehicles. The people, caught between the irresistible pull of gasoline-powered vehicles and the allure of environmentally friendly EVs.

Undoubtedly, the beginning of the 20th century was the golden age of EV. Indeed, it is an EV that for the first time crossed the limit of 100 km/h, on April 29, 1899, with the Belgian Camille Jenatzy in his car called «La Jamais Contente» ("The Never Satisfied") in the shape of shells 1.2. This EV had two engines driving the rear wheels directly, with a total maximum power of 50 kW (67 horsepower), powered by the 80 elements of the Fulmen battery weighing almost half the total weight of the 1.5-tonne vehicle [2].

Two years later, on October 12, 1901, the French engineer Louis Krieger made, without charging, the Paris-Châtellerault trip, 307 km at an average speed of 17.5km/h. This performance earned him a status among the most important electric vehicle manufacturers of the early century.

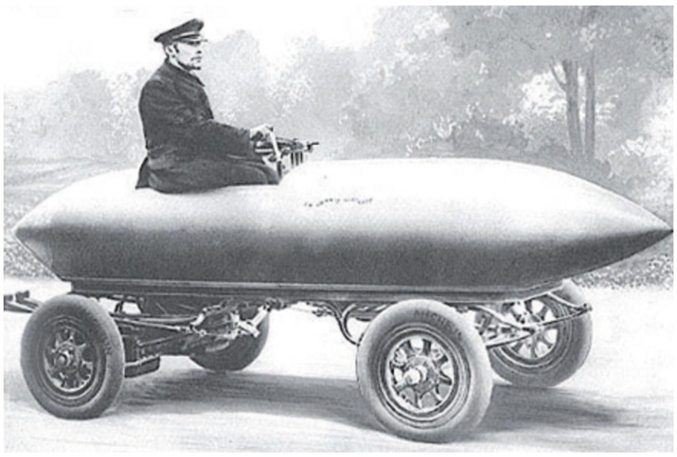


Figure 1.2: La Jamais Contente EV

The electric vehicle was therefore present in the world of the automobile. However, the continuous improvement of internal combustion engine vehicle performance, the emergence of cheap gasoline, and the persistence of battery capacity limitations did gradually remove EVs from the market. Thus, in the early 1930s, the production of electric vehicles was almost completely interrupted, the period 1921-60 being dominated by vehicles with a thermal engine. Therefore, EVs have had only specific uses.

New possibilities appeared in electric traction after 1945, when the Bell labs invented transistors and later, when thyristors were able to switch currents and high voltages. These discoveries led to the development of power electronics that allow the replacement of rheostats and the control of variable frequency AC motors [2].

At the same time, in the 1960s and 1990s, because of air pollution and especially the oil shocks of the 1970s and 1980s, many countries began to take an interest in EVs. Searches are repeated and the EV begins to reappear in small numbers. The modern era of EVs culminates between the 80's and 90's with a few vehicles made like the EV1 produced by General Motors 1.3. The next year, Toyota introduced the world's first commercial hybrid electric vehicle (HEV), Prius in Japan and 18,000 units were sold in the first production year [6].

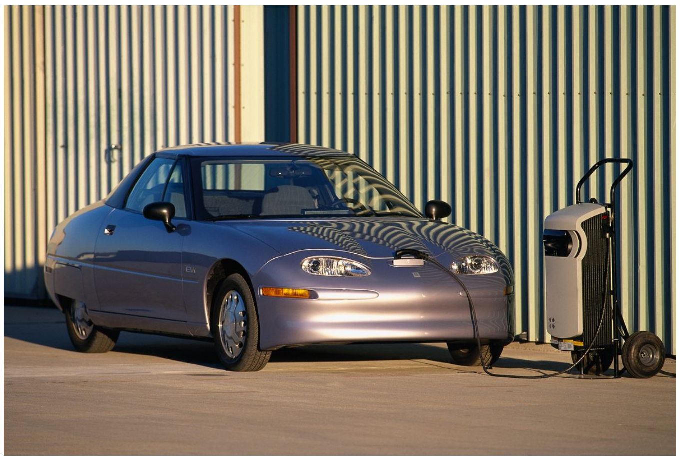


Figure 1.3: Electric vehicule EV1 produced by General Motors

Despite progress, in the 1990s it became clear that electric vehicles could not compete with conventional vehicles because of their insufficient range and performance. As at the beginning of the 20th century, the brake on their development is, and remains, the source of energy storage i.e the battery.

As the oil price kept increasing, more automakers were committed to vehicle electrification. From 2010 onwards, pure electric vehicles (PEVs) and hybrid electric vehicles (HEVs), such as Nissan Leaf, Chevrolet Volt and Tesla Model S have started to enter into the automotive industry. In 2020, there were 27 different models of PEVs available from 11 different manufacturers [5].

The progression towards more efficient and feasible electric vehicles has been also heavily reliant on the advancement of sophisticated motor drives and their control techniques. Motor drives are responsible for managing the speed and torque of the motor, rendering them an indispensable component of EVs. During the initial stages of EVs development, DC motors were a common choice. Nevertheless, due to their elevated efficiency and dependability, AC motors have pervaded in contemporary times.

Contemporary technological advancements and governmental support have sparked a resurgence in electric vehicle interest as a greener and more sustainable transportation mode. With ongoing advancements in battery technology and motor drives, the prospects of electric vehicles look extremely promising, and they are guaranteed to assume a progressively indispensable role in the transportation sector in the years ahead.

1.3 What is an EV

1.3.1 Definition of EVs

An EV is a road vehicle based on modern electric propulsion which is it's main organ, powered by one or more electric motors using electrical energy, it has its own distinct characteristics with an intelligent system that can readily be integrated with modern transportation networks [6].

1.3.2 EV Propulsion System Overview

The EPS of EVs has a general architecture which simply consists of an electric actuator, a transmission device and wheels. The drive, which is the assembly of the electric motor and static converters associated with an electronic drive, is the core of the propulsion system in EVs.

The figure 1.4 shows the block diagram of an EV system with its main components including the controller, power converter, electric motor, and energy source as long as the mechanical transmission.

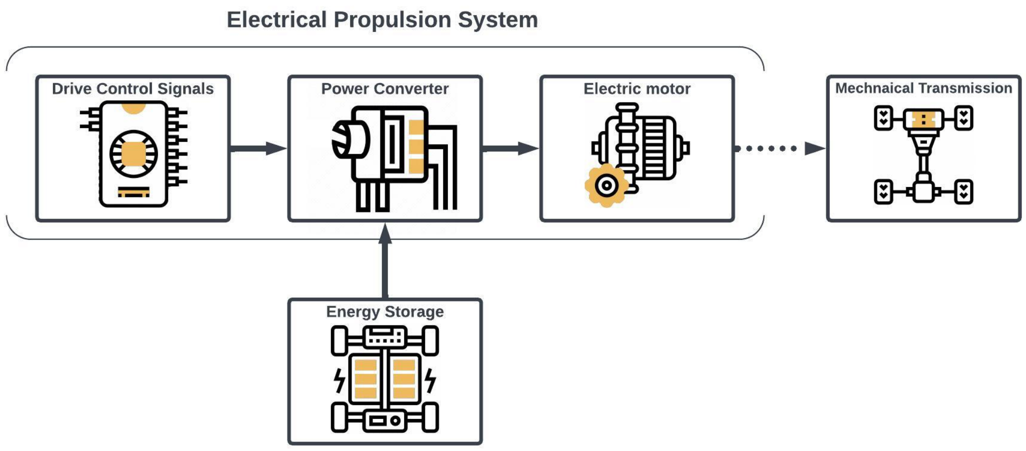


Figure 1.4: Block diagram of general architecture of an EV system

The transmission device sometimes is optional. In fact, the motor drive, comprising of the electric motor, power converter and electronic controller, is the core of the EV propulsion system.

The development of electric propulsion systems has advanced alongside various technologies, particularly in the realms of electric motors, power electronics, microelectronics, and control strategies [17]. Figure 1.5 provides an overview of the EV propulsion system, illustrating the different motor types, power converter devices, and topologies, as well as the hardware, software, and strategies for control. Currently, induction motors and DC motors are the preferred choices for motor technology. Similarly, PWM GBT inverters[1] have gained significant popularity for power converter technology. As for control technology, DSP-based vector controls or VVVF systems are very common.

Table 1.1: Comparison of different EV propulsion system technologies

Software		Devices		Type	
VVVF	Variable voltage variable frequency	GTO	Gate turn-off thyristor	DC	Direct current motor
FOC	Field oriented control	BJT	Bipolar-junction transistor	IM	Induction motor
MARC	Model reference adaptive control	MOSFET	Metal oxide field effect	SRM	Switched reluctance motor
STC	Self-tuning control	IGBT	Insulated-gate bipolar transistor	PMSM	Permanent magnet synchronous motor
VSC	Variable structure control	MCT	MOS controlled thyristor	PMBM	Permanent magnet brushless motor
NNC	Neural network control			PMHM	Permanent magnet hybrid motor
Fuzzy	Fuzzy control				

[1]Note that we didn't mention Silicon carbide (SiC) and gallium nitride (GaN) which are two wide-bandgap semiconductors that have recently gained significant attention in the field of power electronics for EVs. These materials are more promising options for high-performance and efficient EV power systems.

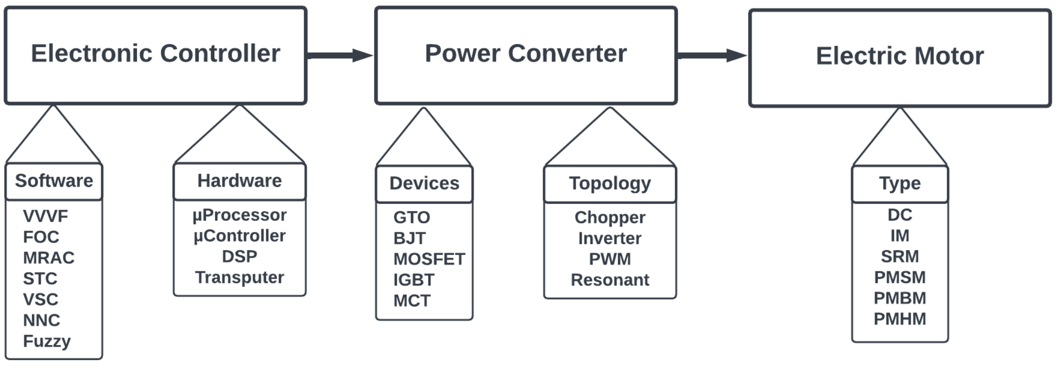


Figure 1.5: EV propulsion system technologies [17]

1.3.3 Classification of EVs

In general, EVs are classified as the PEV, HEV, and FCEV types on the basis of their energy sources and the propulsion devices [15].

The table 1.2 summarises the differences between those types :

Table 1.2: Comparison of different EV types

Types	*PEV*	*HEV*	*FCEV*
Drive section	Electric machine	Electrical machine, ICE	Electrical machine
Energy sources	Battery, Ultracapacitor	Battery, ICE unit, Ultracapacitor	Fuel cell
Energy supplements	Electricity and power system	Electricity and power system, Gasoline statione	Hydroge-nide

With independence of the vehicular architecture which all compete with ICE vehicles, the development of next generation green vehicles based on advanced electric drives requires focus on power converters, electric machines and control drives [18]. Throughout this thesis, we focus specifically on the Control Drive block of the Electric Motor in EVs.

1.3.4 Advantages and limitations of EVs

1.3.4.1 Advantages

EV has several advantages that makes the difference between it and thermal vehicles :

- **Zero Emissions :** Electric vehicles bring an environmentally friendly mode of transportation as they run on electric motors that don't release any emissions while in operation.
- **Cost-effective :** Generally they are cheaper to operate than conventional vehicles as they have fewer moving parts which means they have a lower risk of mechanical failure, require less maintenance and have a longer overall lifespan. Also electricity costs are generally more stable than gasoline costs [19].
- **Energy Efficiency :** They are more energy-efficient than conventional vehicles because they convert more of their stored energy into usable power. In contrast, internal combustion engines lose a significant amount of the energy they produce as heat, which decreases their overall efficiency.
- **Ease of Driving :** As they have instant torque, which means that they can accelerate quickly and smoothly without the need for a traditional transmission as the engine never stalls (no clutch) [16].

1.3.4.2 Limitations

The weight of these vehicles, the safety issue of it's silent engine since it is not always well heard by road users, the long recharging time as refuelling the car with gasoline requires only minutes, limited driving range which is estimated to be less than 480 km and the high prices of those EV's reduced their ability to gain a long-term market presence.

It should also be noted that the development of these vehicles poses other types of environmental challenges like emissions of pollutants related to the extraction of raw materials (e.g. lithium) without forgetting the difficulty of recycling, especially Li-ion batteries [20].

1.4 Overall EV system configuration

The conventional ICEVs employs a combustion engine for propulsion. Its energy source is liquid petrol or diesel. In contrast, the EV employs an electric motor and the corresponding energy sources are batteries. The key difference between the ICEV and EV is the device for propulsion [6].

There are many alternatives for configuring EVs. Compared with the ICEVs, the configuration of the EV is particularly flexible, it generally consists of three major subsystems: electric propulsion, energy source and auxiliary as shown in the figure 5.1.

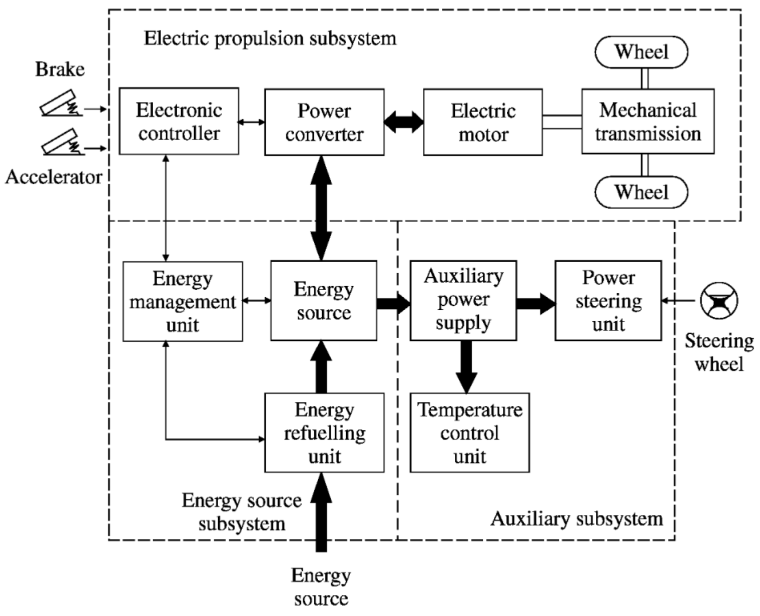


Figure 1.6: EV system configuration [6]

1.4.1 The electric propulsion system

Responsible for converting electrical energy into mechanical energy to propel the vehicle forward. This system consists of several crucial components, each playing a unique role in the overall functioning of the vehicle. Comprises the electronic controller, power converter, electric motor, mechanical transmission and driving wheels.

1.4.1.1 Electronic controller

Serves as the central command unit that provides proper control signals to switch on or off the power devices of the power converter block depending on the control inputs of the systems which represent the brake and accelerator pedals of the EV.

1.4.1.2 Power converter

Functions to modulate the power flow between the electric motor and energy source. In EVs, the power electronic converters are of two types, the DC-DC converters used in the case of DC motors and the DC-AC converters for AC motors.

- **Chopper :**
  DC-DC converter, it allows for the control and regulation of DC voltage levels, to obtain controlled voltages and currents, adjustable and adapted to the needs necessary for the supply of the various receivers. It is also known as a DC-DC buck-boost converter. In EVs systems, choppers are used because :
  - They are indispensable in supplying propulsion engines when they are DC motors.

  - They enable the conversion of high-voltage direct current (DC) from the battery to a lower voltage level that can be efficiently used by various vehicle subsystems.

- **Inverter :**
  DC–AC converter, an electronic device that converts DC power from the battery into AC power. It provides alternating voltages or currents with variable frequency and amplitude. In EVs, these inverters are used to power alternating current motors to be rotated with variable speed, allowing for precise control of the electric motor's speed and enabling smooth acceleration and deceleration.

1.4.1.3 Electric motor

The current stored in the batteries of the energy source is directed to the electric motor which rotates through its rotor under the action of a magnetic field generated in the stator, it converts the electrical energy into mechanical energy, enabling the wheels to turn and propel the vehicle forward. Unlike traditional internal combustion engines, electric motors are known for their high efficiency and instant torque. Several types of motors are used depending on the application, but both DC motors and asynchronous motors are the most common for EVs.

- **DC motor :**
  Short for direct current motor, operates using a direct current power source where the interaction between a magnetic field and an electric current flowing through the motor's windings generates rotational movement.

- **Asynchronous motor :**
  Also called induction motor, is an alternating electric machine that operates through the induction principle, where the rotating magnetic field is created by electromagnetic induction between the stator and rotor windings. It is used today in many applications, especially in transport (metro, trains, EVs) and in industry (machine tools).

1.4.1.4 Mechanical transmission

There are several mechanical configurations of the vehicle that transmit the power from the electric motor to the wheels. EVs typically employ a simple transmission system.

1.4.2 Energy source subsystem

Involves the energy source, energy management unit and energy refuelling unit.

1.4.2.1 Energy source

Responsible for storing and supplying the required electrical energy to both the electric motor for propulsion needed for the vehicle's operation and also for the auxiliary components. This is typically a high-capacity rechargeable battery pack, such as lithium-ion batteries.

1.4.2.2 Energy management unit

Works with the energy refuelling unit to control refuelling and to monitor usability of the energy source. It controls and optimises the flow of electrical energy between the energy source (battery pack) and the various components of the vehicle, such as the electric motor and auxiliary systems. The EMU ensures efficient utilisation of the available energy, manages power distribution, and monitors the battery's state of charge to maximise performance, range, and overall efficiency of the vehicle.

1.4.2.3 Refuelling unit

Works on replenishing the energy source (battery) when it becomes depleted. This typically involves connecting the EV to an external power source, such as a charging station or an electrical outlet, to recharge the battery pack.

1.4.3 The auxiliary subsystem

Consists of the power steering unit, temperature control unit and auxiliary power supply. The auxiliary power supply provides the necessary power with different voltage levels for all EV auxiliaries, especially the temperature control and power steering units. Besides the brake and accelerator, the steering wheel is another key control input of the EV. Based on its angular position, the power steering unit can determine how sharply the vehicle should turn.

1.5 Conclusion

In this chapter, we provided a comprehensive account of the historical evolution of EVs. Subsequently, we presented a general overview on EVs and the architecture of their propulsion system, we then included the EVs classification based on energy sources and motorization. We also discussed the advantages and disadvantages associated with electric cars, highlighting their unique characteristics. Finally, we presented a detailed examination of the overall system configuration found in modern and advanced EVs, starting from the input pedals and steering wheel and extending to the mechanical transmission that transfers the electrical power to the vehicle's wheels.

In the next chapter, we will start with an overview of asynchronous machines and mention some of their different characteristics. Then we will explain the type of the control command employed in this project for the electric propulsion system. Subsequently, we will introduce the principle of the SHE PWM command and the methodology used for generating the necessary database and conclude with some examples of illustration and interpretations.

Chapter 2

SHE PWM Command Applied for Speed Control of ASM in EVs

2.1 Introduction

Selecting the appropriate electric propulsion systems for electric vehicles primarily relies on three key factors: driver expectations, vehicle limitations, and energy sources. Driver expectations are determined by various elements such as acceleration, top speed, climbing ability, braking performance, and range [17].

For a modern EV, the preferred choice for propulsion is a three-phase induction motor. Accompanying this motor is a three-phase PWM inverter, serving as the suitable power converter [6]. By aligning the propulsion system with the driver's anticipated requirements, we can say that an ASM with appropriate control will response largely to the driver's expectation [7, 8]. In practice, control methods produce undesirable harmonics that have many unacceptable effects on the operation of the ASM, such as heating and other factors that lowers the motors lifespan and efficiency.

To ensure smooth and precise control of the ASM, an advanced modulation technique called Selective Harmonic Elimination Pulse Width Modulation (SHE-PWM) command has emerged as a promising solution [12, 13].

The SHE-PWM command technique offers numerous advantages, including enhanced motor performance, reduced harmonic distortions, and improved power quality. By strategically selecting and eliminating specific harmonics, this technique optimizes the motor drive signal, leading to efficient motor operation and reduced energy losses. Consequently, it contributes to the overall energy efficiency of EVs and extends their driving range.

In this chapter, we will start with an overview of asynchronous machines. we will talk about the various characteristics of the latter, such as its constitution and its operating principle. Next, we will explain the type of control and the inverter we will use in this project. Then, we will present the fundamental concept of the SHE PWM technique and the approach employed for generating PWM signals. Additionally, Lastly, we will provide a collection of illustrative examples, interpretations and a conclusion.

2.2 Asynchronous Motor

ASM is an AC machine powered by sinusoidal voltages and currents, widely used in industrial applications, particularly in two primary configurations: single-phase[1] and three-phase [2] motors. ASM is highly favored in EPS of EVs due to its numerous advantages including its simplicity, low cost, reliability, robustness, low torque ripple, low maintenance requirements and the ability to function effectively in harsh operating conditions.

[1]The number of phases refers to the number of AC signals needed for the ASM to work and those signals are phased between each others.

[2]It is worth noting that while there is a theoretical existence of a two-phase motor, it offers no advantages compared to the other configurations.

As shown in figure 2.1 The ASM consist of two parts:

- **The stator** is the stationary part of the motor and typically comprises a set of laminated iron cores with evenly spaced windings called ”Poles”, in typical motor we will find 2p poles which are responsible for generating the rotating magnetic field necessary for the motor’s operation.
- **The rotor** is the movable part of the motor. It is composed of a set of laminated iron cores with conductive bars or conductors arranged in a cylindrical shape. The rotor’s design allows it to interact with the rotating magnetic field generated by the stator, enabling the motor to convert electrical energy into mechanical motion.

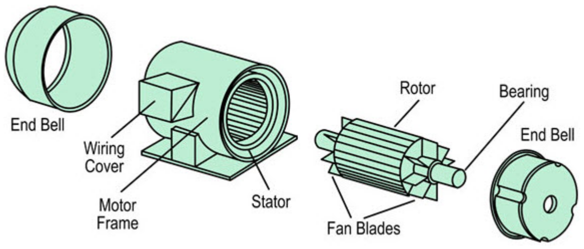


Figure 2.1: A three phase asynchronous motor

When an AC voltage is applied to the stator, it produces a rotating magnetic field that interacts with the conductive bars of the rotor. This interaction induces a current in the rotor, creating a magnetic field. As a result, the rotor initiates rotation at a speed slightly below the synchronous speed of the stator represented by N. The rotor also has an angular speed denoted by Ω. The direction of rotation is determined by the slip, which is the difference between the synchronous speed of the stator and the actual speed of the rotor.

$$N = \frac{120 \times f}{p} = \frac{w}{2\pi p}, \quad \Omega = \frac{w}{p} = \frac{2 \times \pi \times N}{60} \tag{2.1}$$

where :

- N is the synchronous speed RPM [revolutions/minute]
- Ω [3] is the angular speed [rad/s].
- f is the powering AC signal’s frequency [Hz].
- $w = 2\pi f$ is the angular frequency[rad].
- p is the number of poles in the motor.

[3]The rate at which the rotor of an asynchronous motor rotates.

The coupling between magnetic flux and motor torque[4] can be explained by Faraday's law. Whenever there is a variation in the magnetic flux traversing a coil, an electromotive force (EMF) is induced.

The EMF initiates the flow of electric current, resulting in the generation of torque which directly proportional to the magnetic flux. By regulating the magnetic flux the motor's torque and speed can be finely adjusted.

However, the nonlinear coupling between magnetic flux and motor torque and the dependency on other parameters makes its control difficult. Despite this limitation, today's widespread use of ASM can be attributed to significant technological advancements in automotive electronics leading to semiconductor-based transistorized power inverters (such as MOSFETs and IGBTs), also overtime the programmable electronic circuits made their way into drive controls (including FPGAs and DSPs). These advancements have enabled more efficient and sophisticated control of Asynchronous Motors, overcoming the previous challenges and unlocking more potential.

According to the comparative study about the different types of motors used in an EV done by [7, 8] and shown in the table 2.1, IM can achieve a large speed range with appropriate control. The well-known technology and existing manufacturing infrastructure makes IM today's leading motor technology in EV application.

Table 2.1: Electric motor drives comparison

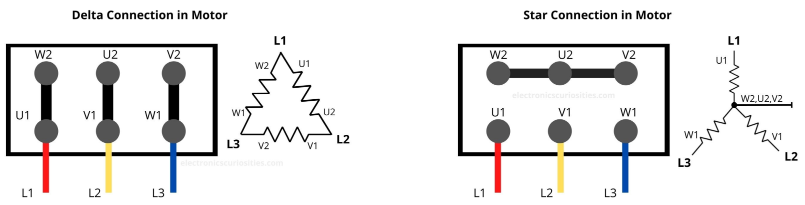
Motor drives	Brushed DC	IM	PMSM[5]	SRM[6]
	Adapted from [7]			
Efficiency	2.0	4.0	5.0	4.5
Weight	2.0	4.0	4.5	5.0
Cost motor	5.0	4.0	3.0	4.0
Total	**9.0**	**12.0**	**12.5**	**13.5**
	Adapted from [8]			
Power density	2.5	3.5	5.0	3.5
Efficiency	2.5	3.5	5.0	3.5
Controllability	5.0	5.0	4.0	3.0
Reliability	3.0	5.0	4.0	5.0
Technol. maturity	5.0	5.0	4.0	4.0
Cost	4.0	5.0	3.0	4.0
Total	**22.0**	**27.0**	**25.0**	**23.0**

As discussed earlier we have two options: single phase and three phase motor.
Single phase motor is used for light applications and features a simple design and is relatively inexpensive compared to the three phase motor, but they are less powerful and less efficient. In the other hand, three phase motors handles heavy loads, efficient, reliable, and capable of delivering higher torque compared to one-phase motors. Thus for our application a three phase motor is the most appropriate for EV design.

[4]Torque is a measure of the force that can rotate an object about an axis.

2.2.1 ASM configuration

The coils of the motor has two different configurations known as delta configuration Δ and star configuration Y, each present a specific wiring pattern of the motor's coils.



(a) (b)

Figure 2.2: The two motor's connections

- **The Delta Configuration** figure 2.2.(a) involves connecting the coils in a triangular configuration, forming a closed loop. This setup enables enhanced initial torque and is commonly employed in applications that demand a high level of torque at the start. Moreover, the delta configuration shows a good performance when we have an unbalanced load conditions.
- **The Star Configuration** figure 2.2.(b) connects the coils in a star or Y pattern. This pattern offers a lower current rating compared to the delta arrangement. It is frequently utilized in situations where a balanced load is expected during regular operation.

From the above we can say that the star configuration can contribute to improved efficiency and reduced voltage stress. But, the decision to opt for either the Δ or Y configuration depends on the specific requirements of the motor application. Various factors must be taken into account, such as starting torque, current levels, voltage conditions, and load characteristics.

2.3 Control of Asynchronous Motors

Different speed and torque control techniques are being applied and developed. We can find two types of command algorithms, **Scalar Command** algorithms which controls amplitude quantities and **Vectorial Command** algorithms that controls amplitude and phase quantities. In our case we will go for Scalar Command mainly because of the simplicity of its implementation and also its low cost comparing to the vectorial command that needs expensive resources. This make the scalar command relatively a great choice [21, 9, 10].

Another essential thing is the ability to go reverse, in three phase motor it can be achieved by reversing the direction of rotation of the stator flux. To do this, we simply swap the inputs of two phases [9].

2.3.1 Scalar Command

As mentioned in [17] high torque at low speeds for starting and climbing, as well as high power at high speed and very wide speed range including constant-torque and constant-power regions are among the major requirements of EV's motor drive.

The scalar command "$\frac{V}{f}$ **constant**" align with these requirements as it maintains the torque constant at its maximum value for all speed spectrum.

2.3.1.1 Constant $\frac{V}{f}$ Command

The constant torque is given with the equation :

$$Te = \frac{3pV^2}{4\pi f(R_s + \sqrt{R_s^2 + [2\pi f(L_s + L_{re})]}} \tag{2.2}$$

Where, R_s and L_s respectively are the resistance and the inductance of the stator, L_{re} is the inductance equivelent of the rotor to stator, p number of poles, V is the rms input voltage to one motor's phase and f is the frequency of the alimentation source.

From the equation 2.2 we notice that the resistances, inductances and the number of poles are characteristics of the motor and are constants, the only variables that is variable and we can actually control are the voltage V and the frequency f.

The synchronous speed mentioned in equation 2.1 is controlled by the frequency just like the torque. As result, if we wish to hold the torque constant at its maximum value T_{max}, we need to maintain the ratio $\frac{V}{f} = C^{te}$ which is equivalent to saying that the frequency varies lineally with the voltage ($f = C^{st}.V$).

In the event where the voltage V remains constant and only the frequency f that varies, we will face the situation of excessive flux induction. This occurs when the flux exceeds the rated capacity which leads to increased eddy current and hysteresis losses. This increased losses cause the heating of the core of the motor and as a result of this the insulation of core will get damaged.

A quite interesting thing is that the outcome of this command is a special square signal called PWM, that we will fairly speak about in section 2.5. Historically, in analog electronics, it is an outcome of a modulation technique [22] which is based on comparing a triangular carrier signal with amplitude (V_c) and frequency (f_c), and a sinusoidal reference signal with amplitude (V_r) and frequency (f_r).

As a technique of modulation we shall define the indexes of modulation as follows

- Amplitude index of modulation $im_a = \frac{V_r}{V_c}$.
- Frequency index of modulation $im_f = \frac{f_c}{f_r}$

In our case im_f can be written as

$$f = im.f_{nom} \tag{2.3}$$

Where f is the motor's input signal frequency (i,e the PWM's frequency) and f_{nom} is the rated frequency or the nominal frequency[7] at which the speed is at its peak value, and because $\frac{V}{f}$ constant than $im_a = im_f = im$.

This result is interesting because earlier we were speaking about speed control with frequency drive. Here the modulation index is the variable that describes the speed of the ASM. For instance, if $im = 1$ than $f = f_{nom}$, here we are at full speed (100% of maximum speed) and if $im = 0.5$ than we are at 50% of the maximum speed.

In situations where the frequency of the motor's input signal is low, resulting in low speeds, the motor encounter statoric losses that may prevent it from turning. To address this issue, a potential solution is to fix the frequency at a certain value and increase the voltage supplied to the motor. The speed at which the motor turns will correspond to that particular frequency and this approach allows for controlling the motor's speed in the low-frequencies range.

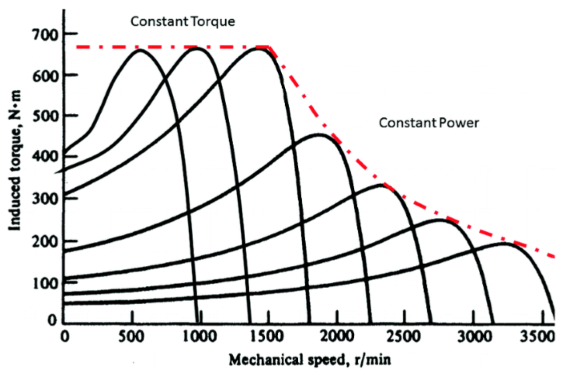


Figure 2.3: Motor's torque as a function of frequency with constant $\frac{V}{f}$ ratio then with constant V.

As we spoke about the command of the motor, we shall also note that we are talking about three phase motor which means it takes as input three phased signals with an exact phase of 120°. If the motor was meant to roll with a constant speed we could just use three oscillators with fixed frequency but for our work on EVs, we need to have instantaneous variation of the speed and that's why we need to think about another solution, this solution is the use of a 3 phase inverter.

2.4 Three Phase Inverter

An inverter, in simple words is a circuit that turns the continuous signal (DC) into alternating signal (AC).

In our case, and as we discussed in section 2.2 the EV's motor needs an AC to produce the rotating electromagnetic field while the energy source would likely be some installed batteries. The need to convert the batteries DC output into a proper AC is typically done by the use of a 3-phase inverter.
In a three phase inverter shown in figure 2.4 we will find 6 switches, typically those switches are IGBTs, Thyristors or in our case MOSFETs.

[7]The f_{nom} is indicated by the constructor

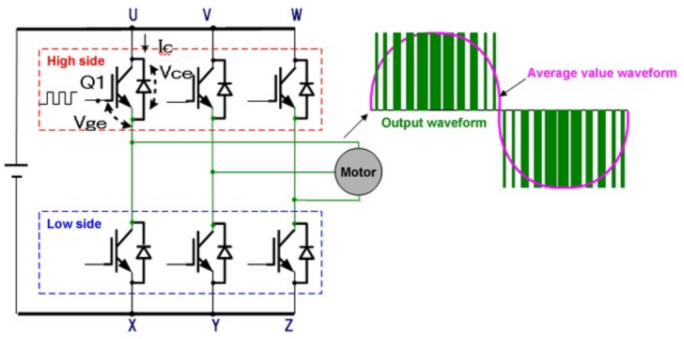


Figure 2.4: Three phase inverter schema

The control of those switches gives us the power to create a three phased signals that modulate the sine wave signal with a great flexibility specially in matter of the frequency and the phase.

The performance of the inverters depends on the structure of the inverters, the technique used for generating the control signals and the intended application. A remarkable effort has been made to introduce efficient orders, which has been reflected in the large number of works published in this context. There are many controlling algorithms that were widely mentioned in the literature explaining the different approaches and techniques developed by researchers to control the inverter [9, 10, 11].

2.5 Pulse Width Modulation

This technique consists of determining the switching moments of the inverter to meet certain criteria. This operating behavioral is generally stored and restored cyclically.

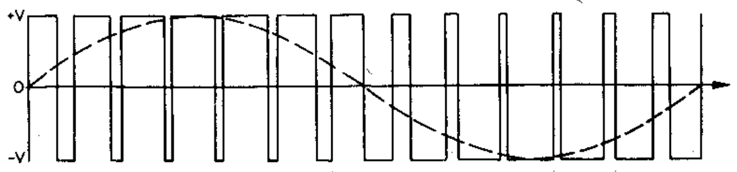


Figure 2.5: Pulse Width Modulation signal [23]

Generally the criteria is relating to the frequency spectrum of the wanted output signal, among those we can find the following criteria:

- The elimination of harmonics of specified ranks.
- The elimination of harmonics in a specified frequency band.
- The elimination of harmonics to a specific order.

Achieving particular requirement as mentioned in [17] and discussed in the beginning of this chapter defines the choice of particular PWM algorithm.

The PWM signal modulates a sinusoidal signal thus it will be convenient that it has the main characteristics of a sine wave which are periodicity and symetry, half wave symmetry and quarter wave symmetry (HWS and QWS respectively).

We will held a specific interest to the last mentioned criterion. The one based on eliminating a bench of harmonics. The technique that does that is called **Selective Harmonics Elimination Pulse Width Modulation**.

2.5.1 Selective Harmonics Elimination Pulse Width Modulation

The SHE PWM is based on the understanding of the frequency behavior of an ASM. In matter of fact, the motor behaves as low pass filter, thus it is sensitive to a specific range of frequencies depending on its speed.

The idea, is to eliminate a large amount of harmonics until we approach the limit of the sensitivity zone [12, 13].

Additionally, the presence of the harmonics specially the high order ones generate vibrations that reduces significantly the lifespan of the motor and attaching parts, beside that they contribute also to the statoric losses and thus power losses and heating.

Here SHE technique is a powerful tool to create a sustainable system. We can thoroughly discuss this technique in frequency domain as we said above that this special PWM signal noted $f(t)$ is periodic in time, it means that it can be viewed as fourier series defined by the chopping (switching) angles $\alpha_1, \alpha_2, \alpha_3 ... \alpha_{2M}$ as shown in the figure 2.6

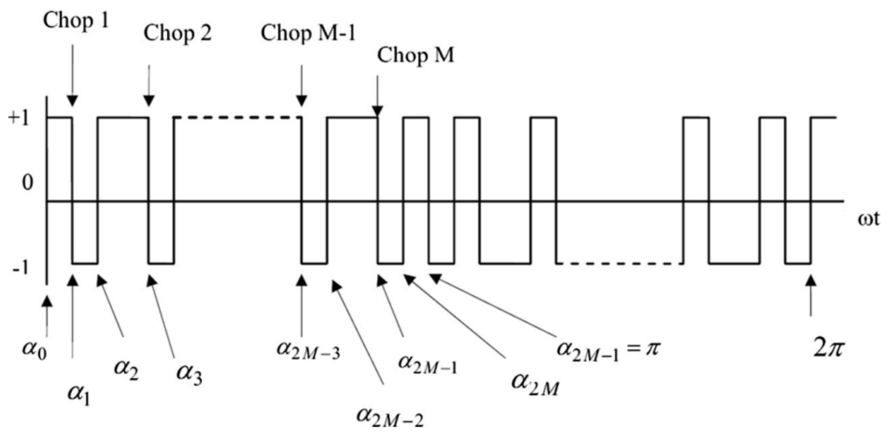


Figure 2.6: SHE PWM signal with 2M angles [11]

If we can write $f(t)$ as sum of only sines (or cosines) and applying the LPF[8] propriety of the ASM than we will end with a system described with a finite sum of sines where each one of them is multiplied by a factor (fourier coefficient).

[8]The low pass filter will eliminates the frequencies higher than the cutoff frequencies thus the only ones left are $f < f_{cutoff}$

Now, if we consider this finite sum and than cancel out all the sines except one, then $f(t)$ indeed is now a sine wave signal, we can do that by equating the fourier coefficients to zero without changing the PWM form as a square signal by introducing the chopping angles, the α's. This is the hard core of Hasmukh S. Patel and Richard G. Hoft's work [24].

2.5.1.1 The Fourier Series of SHE PWM

Let $f(t)$ be the signal SHE PWM, inverters output, a 2π periodic function. The Fourier series of $f(t)$ is given by the general form:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \sin(nwt) + b_n \cos(nwt)) \tag{2.4}$$

where a_0, a_n, b_n are the fourier coefficients calculated using the following formulas:

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\, dx$$

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(x)\, dx$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(x)\, dx$$

In the frequency domain the terms $a_0/2, a_n \cos(nx)$ and $b_n \sin(nx)$ are called the harmonics of the Fourier series and **n** is the order of the harmonics. $n = 1$ represents the first harmonic which is called the fundamental frequency.

As mentioned in [12], by substituting $f(t)$ in a_n, b_n where the amplitude of the SHE PWM was normalized. We should get:

$$a_n = \frac{2}{\pi} \sum_{k=0}^{2M} (-1)^k \int_{\alpha_k}^{\alpha_{k-1}} \sin(nwt) d(wt) \tag{2.5}$$

where $\alpha_0 = 0, \alpha_{2M+1} = \pi$ and $\alpha_1 < \alpha_2 < \alpha_3 ... < \alpha_{2M+1}$
After computing the integral and similarly computing b_n the coefficients of fourier would become:

$$a_n = \frac{2}{n\pi}[1 - (-1)^n + 2\sum_{k=1}^{2M} (-1)^k \cos n\alpha_k] \tag{2.6}$$

$$b_n = \frac{4}{n\pi}[-\sum_{k=1}^{2M} (-1)^k \sin n\alpha_k] \tag{2.7}$$

If we apply the symmetries mentioned in section 2.5 we will end with $a_0 = 0$, and the even coefficients will also be zero, so we will left with $n = 1, 3, 5....$

Using the half-wave symmetry property in 2.6 will reduce to

$$a_n = \frac{4}{n\pi}[1 + \sum_{k=1}^{2M} (-1)^k \cos n\alpha_k] \tag{2.8}$$

The next step is to use the QWS symmetry and than $b_n = 0$ and the sum of the 2M cosines in a_n[9] will also simplifies to its finale form:

$$a_n = \frac{4}{n\pi}[1 + 2\sum_{k=1}^{M}(-1)^k \cos n\alpha_k] \tag{2.9}$$

Now, we proved that the SHE PWM signal $f(t)$ is just a sum of cosines with coefficients that depends on the chopping angles as follows

$$f(t) = \sum_{n=1}^{\infty} a_n(\alpha_1, \alpha_2, \alpha_3, ..., \alpha_M) \cos nwt, \qquad n = 1, 3, 5... \tag{2.10}$$

Refering to 2.10, if we equate a_1 to an amplitude A, and then by eliminating all the other harmonics to the limit of the sensitivity zone, by this the motor will sense at each phase only the input $f(t) \simeq A\sin(wt) + \phi$ without the presence of other harmonics. The elimination process is done by equating $a_n = 0$ for $n > 1$ and then finding the proper angles.

2.5.1.2 Computing Chopping Angles

When we computed the fourier series of the SHE PWM signal we normalized the rated voltage $\frac{E}{2}$ (i,e $\frac{E}{2} = 1$) to simplify the calculation, when we did that the input voltage A now is defined by $0 < A < \frac{E}{2}$ which means two things. First we have $0 < A < 1$ and second $im = \frac{A}{\frac{E}{2}} = A$. Rigorously, the fundamental's amplitude a_1 is assigned to the modulation index but as mentioned in section 2.3 the modulation index **im** describes the ASM's speed as a portion of the maximum speed.

As a result, to cover the whole spepd range from 0% (stop) to 100% (full speed) we need to find the suitable angles for each **im** value until we cover all speed's range.

Now, computing the angles can be done by solving a non linear system where we will neglect the harmonics multiples of 3, because the three phase motor will eliminates them automatically while it's running.

To solve 2.12 we will use the Newton-Raphson non linear method defined by the general form

$$A_k = A_{k-1} - J[F]^{-1}F \tag{2.11}$$

[9]All detailed calculations were mentioned in [12]

Where $A = [\alpha_1 \alpha_2 ... \alpha_M]^T$, J is the jaccobian matrix and $F = [f_1 f_2 ... f_M]$ as following:

$$\begin{aligned}
f_1 &= \frac{4}{\pi}[\,1 + 2\sum_{k=1}^{M}(-1)^k \cos n\alpha_k]\; + im = 0 \\
f_2 &= \frac{4}{5\pi}[\,1 + 2\sum_{k=1}^{M}(-1)^k \cos n\alpha_k]\; = 0 \\
f_3 &= \frac{4}{7\pi}[\,1 + 2\sum_{k=1}^{M}(-1)^k \cos n\alpha_k]\; = 0 \\
&\vdots \\
f_M &= \frac{4}{n\pi}[\,1 + 2\sum_{k=1}^{M}(-1)^k \cos n\alpha_k]\; = 0
\end{aligned} \tag{2.12}$$

As mentioned in [25],So the Newton-Raphson iterative method converges, we need to:

- Assign a negative value to **im** and the sign (-) corresponds to a π phase shift of the fundamental that has no effect on the motor.
- Choose a good initial guess because Newton-Raphson method is highly sensitive to initial values.

Besides to the enormous number of equations needed to be solved, another problem is that the number of angles is not fixed. The sensitivity zone of the ASM changes according to the speed of the motor, the lower the speed the higher number of angles is needed.

For that, we have developed a MATLAB program that calculate the angles across all values of **im** and consequently it generates the angles data base.

2.6 Finding the Chopping Angles using MATLAB

Before discussing the way to find the good initial guess $A_0 = [\alpha_1^0 \alpha_2^0, ... \alpha_M^0]$, we shall determine the number of angles M.

When we refer back to the expression of the coefficients a_n as depicted in equation 2.9, we observe that the harmonic order **n** appears in the denominator. This implies that as the order increases, the amplitude of the harmonic decreases. Hence, practically, beyond a certain order l, the amplitudes $a_{n\geq l}$ becomes significantly small, rendering the harmonics negligible and treating it as noise.

If the amplitude of the fundamental harmonic is significantly larger than that of other harmonics, we can achieve a sinusoidal waveform (representing the fundamental) with minimal noise by eliminating a small number of harmonics. However, if the amplitude of a_1 is relatively low, we would need to eliminate a greater number of harmonics thus we need more chopping angles to distinguish the fundamental from the others, in order to maintain the same level of noise tolerance. Therefore, if the value of $a_1 = im$ is relatively large, the number of angles needed M, will be smaller. Conversely, if a_1 is smaller, the number of angles M needs to be larger.

To measure the noise tolerance, we shall refer to IEEE 519 norm that sets no more than 5% total harmonic voltage distortion[10] with the largest single harmonic being no more than 3% of the fundamental voltage.

The number of angles M was chosen based on experimental results [14] by considering the THD and the implementation on an FPGA device as shown in Table 2.2.

Table 2.2: Number of chopping angles as a function of **im**

Modulation index (**im**)	Switching angles number (**M**)
$0.001 < im \leq 0.159$	23
$0.16 < im \leq 0.319$	19
$0.32 < im \leq 0.559$	15
$0.56 < im \leq 0.759$	07
$0.76 < im \leq 0.919$	05
$0.92 < im \leq 1$	03

Now, we know that we have **M** equations to solve where we will eliminate **M-1** harmoics. The next thing is to find the proper initial values.

Taufik, Mellitt and Goodman's work [24] grants a proper way to find the initial value A_0 as shown in the figure 2.7

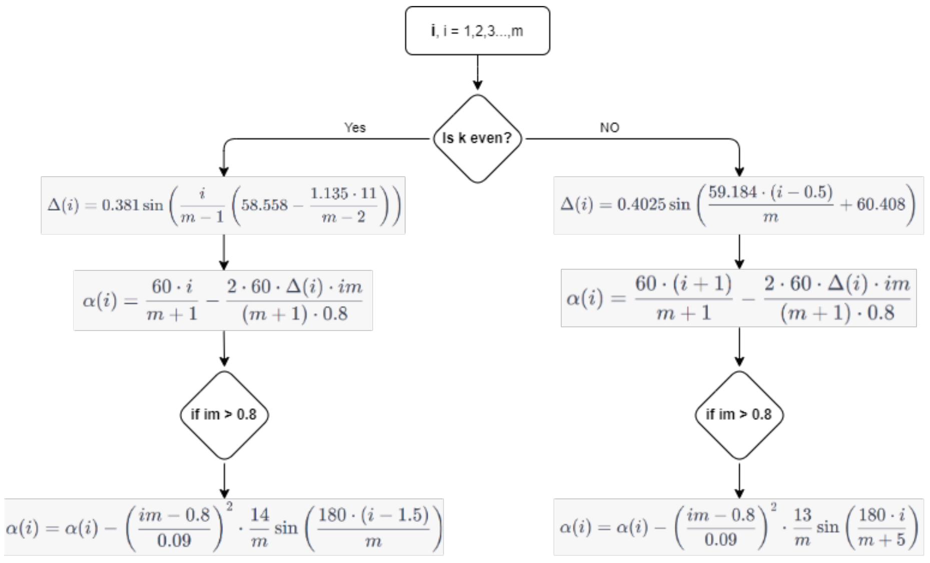

i, i = 1,2,3...,m
Is k even?
Yes
NO
$\Delta(i) = 0.381 \sin\left(\frac{i}{m-1}\left(58.558 - \frac{1.135 \cdot 11}{m-2}\right)\right)$
$\Delta(i) = 0.4025 \sin\left(\frac{59.184 \cdot (i-0.5)}{m} + 60.408\right)$
$\alpha(i) = \frac{60 \cdot i}{m+1} - \frac{2 \cdot 60 \cdot \Delta(i) \cdot im}{(m+1) \cdot 0.8}$
$\alpha(i) = \frac{60 \cdot (i+1)}{m+1} - \frac{2 \cdot 60 \cdot \Delta(i) \cdot im}{(m+1) \cdot 0.8}$
if im > 0.8
if im > 0.8
$\alpha(i) = \alpha(i) - \left(\frac{im - 0.8}{0.09}\right)^2 \cdot \frac{14}{m} \sin\left(\frac{180 \cdot (i-1.5)}{m}\right)$
$\alpha(i) = \alpha(i) - \left(\frac{im - 0.8}{0.09}\right)^2 \cdot \frac{13}{m} \sin\left(\frac{180 \cdot i}{m+5}\right)$

Figure 2.7: The process of Taufik, Mellitt and Goodman's algorithm to choose the initial guess of Newton-Raphson

Finally, to proper end the code we need a stopping criteria. In our case, we set the error tolerance to 10^{-12} and the maximum number of iterations to 200. During the code execution, it was consistently observed that the error criterion was satisfied before

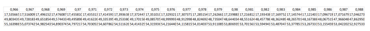
[10]total harmonic distortion or THD is the ratio of the sum of the powers of all harmonic components to the power of the fundamental frequency.

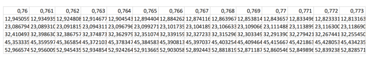
reaching the maximum number of iterations. Specifically, the highest number of iterations achieved was 6.

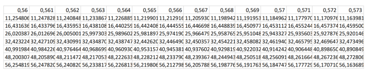
We ran the code and we solved the system of equations 2.12 for $im \in [0;1]$ with step of 0.001 each time.

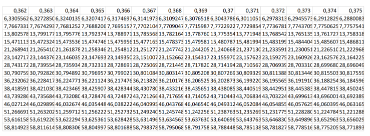
2.7 The Angles' Data Base

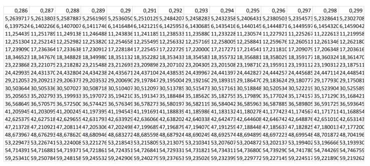
The set of solutions is now our data base as shown in Figure 2.8 saved in Excel sheets, it gives us an overview on how the data base looks like.


(a) $M = 3$


(b) $M = 5$


(c) $M = 7$


(d) $M = 15$


(e) $M = 19$

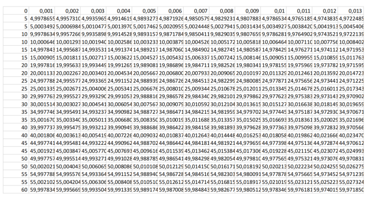

(f) $M = 23$

Figure 2.8: Samples from the data base depending on the number of angles **M**

Another figure that's quite important is the angles trajectory with respect to **im**, the Figure 2.9 shows that property

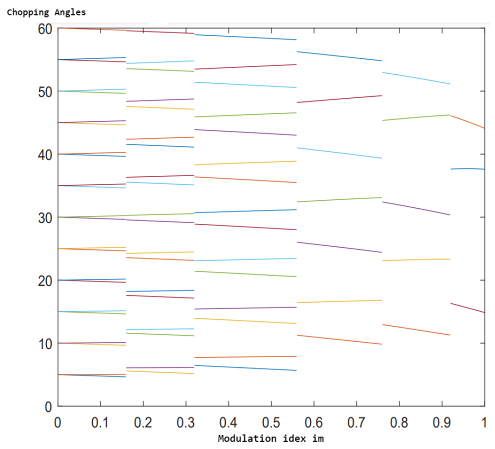


Figure 2.9: Angles(in deg) trajectory with respect to **im**

From Figure 2.8 we can say that the number of angles involved is exceedingly large. Consequently, implementing the Newton-Raphson algorithm on hardware becomes impractical as it requires a significant amount of time to approximate the solutions in an on-line application process. Another alternative is to store the angles in a memory device. However, this approach is also unrealistic due to the substantial memory allocation resources it demands, making it inefficient and incompatible with our objective of developing an industrial automobile.

Figure 2.9, on the other hand, confirms that dividing the angles into six sets was a wise decision. As depicted, within each range of **im** angles, a linear pattern can be observed. However, when considering all values of **im**, the behavior becomes nonlinear and shows discontinuity. This makes it challenging to describe the entire system using a continuous linear function that might have been simple tp deploy it on hardware with minimal cost.

Recognizing the significance of financial cost in projects like these, we are driven to explore alternative options which we will discuss further more in 2.10. For now, we will focus on generating the SHE PWM signal.

2.8 Generating SHE PWM Signal in Time Domain

The analysis performed in this chapter until now are not given in time basis. At beginning in section 2.5, we refered to PWM signal angles in degrees. In real world applications, PWM signals are time's based functions, thus we need a time adaptor bloc that generates the switching instants from the switching angles.

This bloc that transforms the angles to instants. It can be described as fellows

$$\begin{cases} 360^0 \to T \\ \alpha_i \to t_i \end{cases}$$

Where T is the period of the SHE PWM signal. The adaption can be described as follows

$$t_i = \frac{\alpha_i}{360f} \tag{2.13}$$

Now if we replace the SHE PWM frequency f by $im \times f_0$ as we explained in section 2.3, f_0 arbitrary is set to 50 Hz and **im** is in percent we will end with

$$t_i = \frac{\alpha_i}{180im} \tag{2.14}$$

Using MATLAB, we can load the angles from an Excel file, convert them into time instants, assuming a nominal frequency of $f = 50$ Hz. We can then generate SHE PWM signal as depicted in the Figure 2.10.

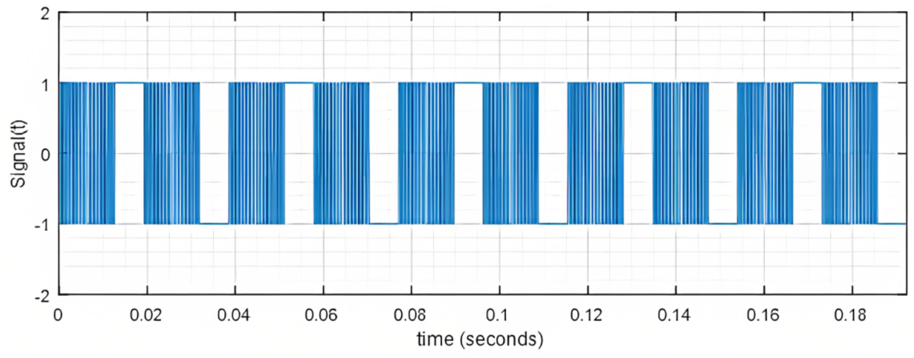


Figure 2.10: SHE PWM signal for **im** = 0.52

To verify the suppression of harmonics, we will perform a Fast Fourier Transform (FFT) and examine the frequency spectrum of the SHE PWM signal.

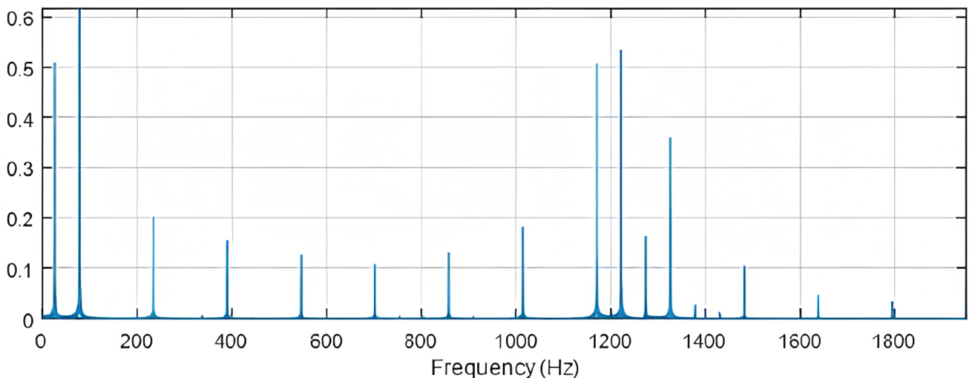


Figure 2.11: Frequency Spectrum of SHE PWM Presented in figure 2.10

2.9 Interpretation

As shown in Figure 2.11, the frequency spectrum demonstrates that only multiples of 3 are present, indicating the successful suppression of other frequencies. The first non-eliminated harmonic occurs at the 47^{th} harmonic. Additionally, the fundamental frequency is observed at $f = 26Hz$, which aligns with the theoretical expectation of $im \times f_0 = 0.52 \times 50 = 26Hz$. This consistency between the theoretical prediction and the observed fundamental frequency confirms the accuracy of the analysis.

If we apply the FFT across all intervals of **im**, we can approximate the minimum non-suppressed frequency within each interval. The table below illustrates the frequencies obtained

Table 2.3: Minimum first harmonics not eliminated with respect to **im**

Index of modulation (**im**)	Minimum first harmonic not eliminated (Hz)
$0.001 < im \leq 0.159$	355
$0.16 < im \leq 0.319$	472
$0.32 < im \leq 0.559$	752
$0.56 < im \leq 0.759$	644
$0.76 < im \leq 0.919$	646
$0.92 < im \leq 1$	506

2.10 Conclusion

In this chapter, we provided a comprehensive account of the ASM for EV propulsion system use, we then included the main approaches to control the ASM. We also discussed the advantages of choosing the scalar command over the vectorial command, highlighting the $\frac{V}{f} = constant$ scalar command as a proper variable frequency drive method to control an EV's AS motor. Finally, we thoroughly discuss the SHE PWM technique as our preferred method for motor control. We begin by examining the three-phase inverter as the interface between the ASM and the batteries, and subsequently explore the system of equations derived from the SHE PWM signal, which effectively governs the motor's speed regulation by adjusting the modulation index **im** and computing the appropriate switching (chopping) angles.

Nonetheless, determining the precise values of the switching angles when employing the SHE PWM technique involves solving a system of **M** nonlinear equations with **M** unknowns. To address this challenge, we have used the Newton-Raphson method in this chapter. However, it is important to note that implementing this method requires a significant amount of computational time and necessitates the estimation of initial values to ensure convergence. Therefore, we used the algorithm proposed by Taufik, Mellitt, and Goodman to effectively handle this requirement [24].

Even thought using Newton-Raphson method gives high precision of switching angles $\epsilon < 1.0^{-15}$. However, it is crucial to acknowledge that this comes at the expense of extensive computation time, which prevents real-time speed control in an ”on-line” scenario. Consequently, the SHE PWM technique is suitable for ”off-line” applications where the chopping angles are stored in a memory for a limited inputs range. Such process demands substantial memory capacity to store all the calculated angles and achieve optimal precision.

To address this limitation, our next chapter will introduce two distinct approaches that we have adopted to develop an efficient ”on-line” SHE PWM technique for real-time calculation of switching angles. These approaches aim to provide a more suitable algorithm for controlling the ASM of an EV propulsion system. The first approach is an algorithm based on Artificial Neural Networks (ANN) modeling, while the second approach introduces an innovative method utilizing polynomial interpolation (PI) techniques.

Chapter 3

On-Line SHE PWM Algorithm

3.1 Introduction

In the realm of EVs propulsion systems, achieving efficient control of the power electronics is essential to ensure optimal performance and driveability. One critical aspect that significantly influences the overall performance of an EV is the latency of acceleration and the rate of speed change. As discussed in chapter 2, solving the Patel and Hoft's equation system to eliminate specific harmonics using the iterative Newton-Raphson algorithm gives accurate results but requires extensive computational time, preventing real-time speed control.

Our primary focus centers around the implementation of an on-line SHE PWM algorithm with acceptable real-time response. All measurements and considerations must be aligned with this objective. However, it is essential to prioritize simplicity of the implementation and cost-effectiveness, as increased complexity would lead to inefficient resource utilization and higher expenses.

Using Newton-Raphson method, we have constructed a database that provided us with two potential approaches. The first approach is an algorithm based on Artificial Neural Networks (ANN), proposed by [14], which aims to calculate the switching moments of the PWM signal with high speed and with a precision that is very close to those calculated by the Patel and Hoft algorithm. Another algorithm worth exploring is Polynomial Interpolation, which utilizes similar principle of employing a database to generate various polynomials that calculate the switching angles in real-time with very high precision.

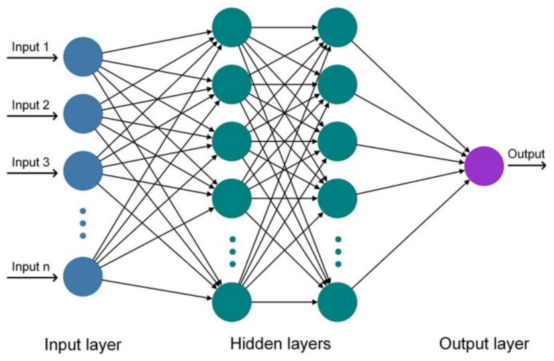
This chapter focuses on exploring two On-Line SHE PWM algorithms. To begin with, we will provide an overview of Artificial Neural Networks (ANNs). Subsequently, we will delve into the learning process employed and discuss the design of its topology. This will lead us to unveil the architecture for the operational ANNSHE PWM. Additionally, we will demonstrate the implementation of this architecture using MATLAB and offer insightful interpretations. Following this, we will adopt a similar approach to examine Polynomial Interpolation. Lastly, we will conclude by conducting a comparative analysis between the real-time implementation of both PISHE PWM and ANNSHE PWM algorithms on a micro-controller. By evaluating their respective advantages and limitations, we will determine the most suitable algorithm for our specific application.

3.2 Artificial Neural Network ANN

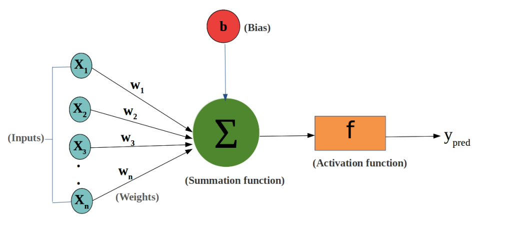
A neural network is a computational model that draws inspiration from the structure and functioning of the human brain. It serves as a powerful tool within the field of artificial intelligence and machine learning, enabling the processing of intricate data and facilitating predictions or decision-making.

3.2.1 Overview of ANN

At its core, a neural network comprises interconnected artificial neurons, also referred to as nodes or units, which are organized into layers. The initial data is received by the input layer, after which it traverses through hidden layers before ultimately reaching the output layer. Neurons within a layer establish connections with neurons in adjacent layers through weighted and biased connections, which enable the network to learn and adapt over time.



(a) Diagram of an ANN



(b) Diagram of one neuron

Figure 3.1: ANN's Schematic

During the learning process, a neural network adjusts the weights and the biases of its connections based on a given dataset, which contains input examples and their corresponding desired outputs. This adjustment, known as training, is achieved through a process called backpropagation[1], the network progressively refines its predictions by comparing them to the desired outputs and iteratively updating the connection weights and biases accordingly. The predicted output $Y_{predicted}$ of a single neuron, as illustrated in Figure 3.1.b, can be mathematically described as the result of a linear combination of the input vector X passed as an argument to an activation function.

$$Y_{predicted} = f(\sum WX_{input} + B)$$

Where Y is the output vector, f the activation function, W the weights vector and B the biases vector.

The comparison between the predicted value and the exact value is done using an error function. Many error function were presented in literature, the use of any of them depends

[1]backpropagation is an algorithm for training feedforward ANNs or other parameterized networks with differentiable nodes based on gradient descent.

on the study case. The simplest one and the most used one is the **Mean Squared Error**, which is defined by 3.1.

$$E_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.1}$$

Where n is the number of the desired outputs, y_i represents the actual value of the ith output, and $\hat{y}_i$ represents the predicted value of this output.

The equations to update the weights and biases can be expressed as shown in 3.2, 3.3.

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} - \eta \cdot \frac{dE_{MSE}}{dw_{ij}^{\text{old}}} \tag{3.2}$$

$$b_i^{\text{new}} = b_i^{\text{old}} - \eta \cdot \frac{dE_{MSE}}{db_i^{\text{old}}} \tag{3.3}$$

Where w_{ij}^{new} and b_i^{new} are the updated weight and bias connecting the ith input neuron to the jth output neuron, w_{ij}^{old} and b_i^{old} are the previous weight and bias values, η is the learning rate,[2] $\frac{\partial E_{MSE}}{\partial w_{ij}^{\text{old}}}$ is the error gradient of the jth output neuron.

The learning rate η determines the impact of each weight and bias update on the overall learning process. A higher learning rate may lead to faster convergence, but it can also cause instability or overshooting. On the other hand, a lower learning rate may result in slower convergence but increased stability.

Neural networks have demonstrated remarkable success across various applications, including tasks such as image and speech recognition, autonomous systems, and predictive analytics. They possess the ability to unveil intricate patterns and relationships within vast amounts of data, making them valuable tools for solving complex problems and generating accurate predictions with high speed ratio.

3.2.2 Activation function

The neurons process data through weighting the input's sum and further adding bias to it, then we apply a non-linear transformation called the activation function as shown in figure 3.1.

The backpropagation is possible due to the non-linearity aspect of the activation function since the gradients are supplied along with the error to update the weights and biases. In other words, without it a neuron would not be activated, we would loose the capacity to learn and the whole model would be simplified to another linear regression model.

Through literature and academic papers, multiple activation functions have been proposed [26], but for the time being, we will present a figure that shows a selection of wide used activation functions, including sigmoid the most widely used activation function.

[2]η is a hyperparameter that determines the step size of weight and biases updates

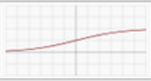
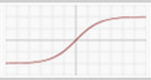
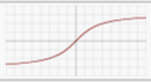
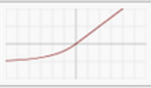
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 3.2: List of usual activation functions

3.2.3 Training the ANN

The ability to exhibit the same behavior on elements that are quite different in comparison to those in the data base is what makes ANNs so popular this days, at this point we can say that the model learned about that particular subject or in other words the model is well trained.

In typical training algorithms, the initial step involves finding the right weights and biases of the neural network, then the network's performance is evaluated to determine its validity. This iterative process continues until the validation error falls within an acceptable range. There exist various methods for training neural networks [27]. But mainly there is three types of learning, one approach involves explicitly setting the weights using prior knowledge or predetermined values, another approach involves training the neural network by providing it with teaching patterns and allowing it to modify its weights based on a specific learning rule and the third one is about receiving feedback in the form of rewards, the goal is to find the optimal strategy that maximizes the cumulative reward over time.

The figure below refers to the different methods of learning and the application of every method in real world application.

The types of learning depicted in figure 3.3, namely decision-making and skills acquisition, are not our primary focus of interest. Thus, we will explore the other two methods that mainly deals with data processing and modeling.

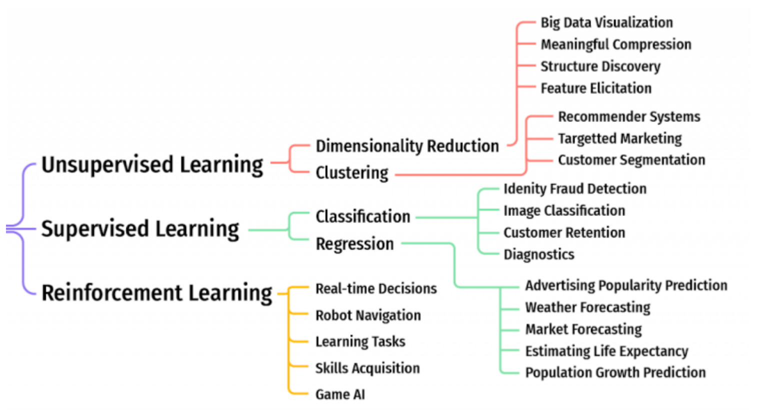


Figure 3.3: Main types of training (learning)

3.2.3.1 Supervised training

Supervised training is a method where the neural network is trained by supplying it with a set of inputs along with their corresponding expected outputs while learning the ANN undergoes multiple iterations, or epochs, until its output aligns with the anticipated output, with a relatively low error rate. Each epoch represents one complete cycle through the training samples.

3.2.3.2 Unsupervised training

Unsupervised training does not involve providing anticipated outputs. This type of training is commonly employed when the neural network is tasked with classifying inputs into multiple groups. Similar to supervised training, unsupervised training entails numerous epochs. As the training proceeds, the neural network ”discovers” the classification groups through its learning process.

3.3 Multi-Layer Perceptron ANN algorithm for SHE PWM Application

The Multi-layer Perceptron (MLP) is a supervised learning algorithm designed to learn a function F through training on a dataset. Such that if the data exhibit a pattern modulated by the function F the MLP will learn the pattern and map it to the Network's parameters represented essentially by its weights and biases and other parameters [25] including learning rate, the number of inputs, outputs, neurons, hidden layers and so on.

3.3.1 Data Base

In order to train the MLP network, we start by the data. Our data sets represent the angles data base mentioned in section 2.7, which is primarily a function of three parameters:

- The modulation index (**im**).
- The chopping angles α for each **im**.
- The number of angles (**M**), referring table 2.2.

Considering that our problem is that we want to construct a function noted F that gives us the angles for each value of **im**. Naturally, the input of this function is **im** and the output is the M angles as shown below.

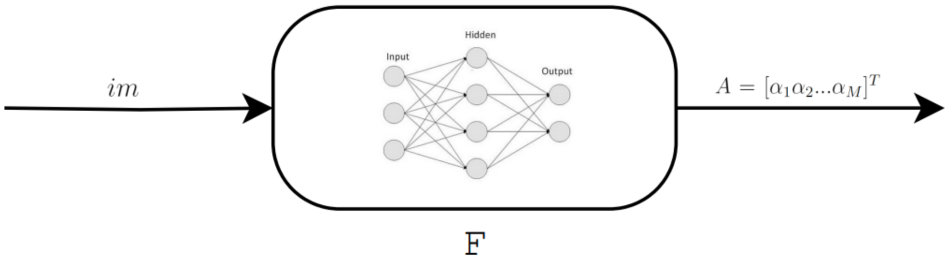


Figure 3.4: Schematic of ANN SHE PWM

Referring to Table 2.2, we have six data sets of angles, each one with different number of angles that correspond to the different intervals of **im**. So, it would be suitable to divide the database into six smaller datasets, assigned to train a particular MLP. We need six Fs just like the F shown in the figure 3.4. Those functions together construct the **ANN SHE PWM** model or ANNSHE PWM algorithm. For each ANN we will distribute the dataset into three sets

- Training set 95%, These elements are presented to the network during training.
- Validation set 5%, These elements are used to measure the generalization of the network, and finding the right parameters.
- Test set 5%, These elements provide an independent measure of network performance.

3.3.2 Training ANNSHE PWM model

The dataset consists of a pair (x, y). Here, x represents the **im** value, while y corresponds to the respective angles $A_i = [\alpha_1, \alpha_2, \alpha_3..., \alpha_M], i = 1...6$ and $M = 3, 5, 7, 15, 19, 23$.
Initially as discussed in section 3.1, the weights and biases of ANNSHE PWM are assigned with random values. Through the process of backpropagation, these values are continuously adjusted until the network's output y^* approximates the desired value y, within an acceptable tolerance.

3.3.3 ANNSHE PWM architecture

The last thing to do is designing the network's topology, this process often considered a complex and intricate process, for some complex problems it can be consuming to finally achieve a functional and effective network topology. The design consist of finding the best parameters for optimum approximation to the original values. There is no particular algorithm or specific method to choose this parameters, all what we can do is trying until we get the best results. In our case, numerous studies have already addressed this challenge [11, 25, 14, 28], and therefore we will not delve into the process of finding the hyper parameters again. Instead, we will utilize the established parameters that have proven to be suitable for our purposes.

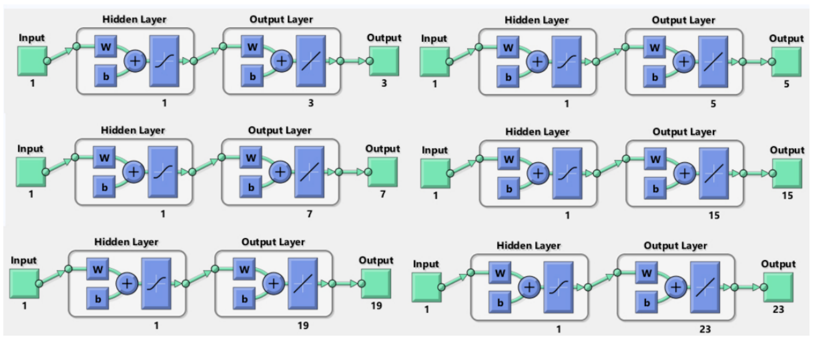


Figure 3.5: Overall ANNSHE PWM algorithm's architecture [25]

The parameters were found are as fellows

- For all six ANNs, the input layer has a single input (**im**).
- For all six ANNs, The output layer has a different number of neurons (3,5,7,15,19 and 23).
- For all six ANNs, The output layer's activation function is a linear function.
- For all six ANNs, a single hidden layer with a single neuron.
- For all six ANNs, the activation function of the hidden layer is the Hyperbolic Tangent tanh.
- Number of iterations = 100000.
- Learning error tolerance 10^{-5}

In relative work [26], a three-layer neural network with sigmoid activation function in the hidden layer and a linear activation function in the output layer, has the capability to approximate our non linear system described by the equation **??**. This approximation can be achieved with a certain level of accuracy. However, when aiming to incorporate various types of non-linearity within the network using only three layers, a substantial number of neurons becomes necessary. This requirement can lead to the creation of an excessively large neural network architecture.

In order to optimize the architecture of a neural network in terms of hardware resource consumption, it is essential to minimize the number of layers, neurons, and non-linear functions while still achieving the desired precision.

The parameters above ensure an optimal network design that utilizes resources effectively without compromising accuracy.

3.4 ANNSHE PWM using MATLAB

The database was stored in Excel format, consisting of six distinct sheets categorized based on the number of angles. MATLAB loads each sheet individually, and subsequently, each model is trained using the examples provided within the respective sheet.

In the learning phase, in order to improve the performance and efficiency of the network and at the same as mentioned in 3.1, to keep the capacity of the network to learn we must use a non linear activation function which generally demandes normalized inputs, Consequently, we normalized all the data in a standard range, it means that we added two blocs one for normalization(equation 3.3) and the other to the de-normalization.

3.4.1 Normalsation

The normalisation equation used is the following

$$im_{norm} = \frac{im_{norm,max} - im_{norm,min}}{im_{max} - im_{min}}(im - im_{min}) + im_{norm,min} \tag{3.4}$$

Her, because the normalization range is [1 ; 0] naturally we will have $im_{norm,max} = 1$ and $im_{norm,min} = 0$. by replacing in 3.4 we find

$$im_{norm} = \frac{im - im_{min}}{im_{max} - im_{min}} \tag{3.5}$$

3.4.2 De-Normalisation

The de-normalisation equation would as fellow

$$\alpha = \frac{\alpha_{\mathrm{max}} - \alpha_{\mathrm{min}}}{\alpha_{\mathrm{norm,max}} - \alpha_{\mathrm{norm,min}}}(\alpha_{norm} - \alpha_{\mathrm{norm,min}}) + \alpha_{\mathrm{min}} \tag{3.6}$$

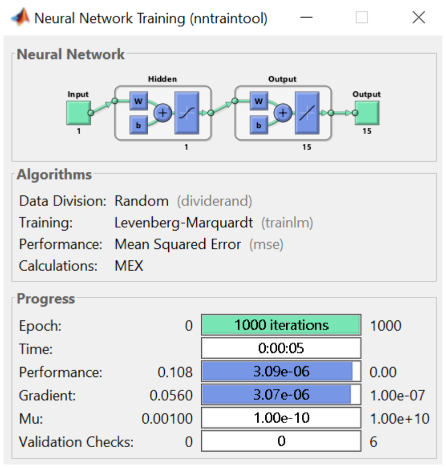
where α_{max} and α_{min} are the max and min values respectively of each angle in the dataset. In the learning stage $\alpha_{\mathrm{norm,max}} = 1$ and $\alpha_{\mathrm{norm,min}} = 0$ were used, with replacing in 3.9 we find

$$\alpha = (\alpha_{\mathrm{max}} - \alpha_{\mathrm{min}}) \times \alpha_{norm} + \alpha_{\mathrm{min}} \tag{3.7}$$

3.4.3 Time Adaptation

Now we proceed with time adaptation, following a similar approach as in section 2.8. Additionally, we compare the obtained results with the theoretical ones mentioned in 1.5.

We can take advantage of MATLAB's Neural Network Toolbox to simplify our work. This toolbox provides various functions and tools for designing, training, and implementing neural networks. It offers a user-friendly interface and a wide range of functionalities that can assist us in tasks such as network architecture design, training data preparation, training algorithm selection, and performance evaluation. By utilizing the Neural Network Toolbox, we can streamline our workflow and leverage the capabilities of neural networks effectively.



Neural Network Training (nntraintool)

Algorithms

Data Division:	Random (dividerand)
Training:	Levenberg-Marquardt (trainlm)
Performance:	Mean Squared Error (mse)
Calculations:	MEX

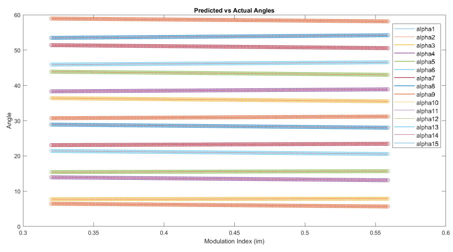
Progress

Epoch:	0	1000 iterations	1000
Time:		0:00:05	
Performance:	0.108	3.09e-06	0.00
Gradient:	0.0560	3.07e-06	1.00e-07
Mu:	0.00100	1.00e-10	1.00e+10
Validation Checks:	0	0	6

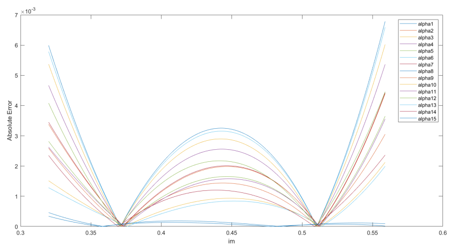
Figure 3.6: MATLAB's Toolbox interface while training

3.4.4 ANNSHE PWM Interpretation

The figure 3.7 displays the output angles of the ANN models for the ANNSHE PWM algorithm along with their error compared to the theoretical results discussed in section 2.7.



(a) ANNSHE PWM output for **im** = 52%



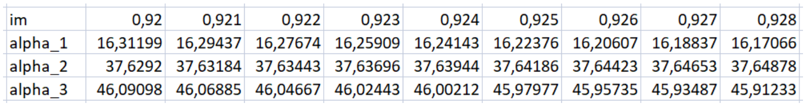
(b) Error Between Predicted vs Actual α

Figure 3.7: ANNSHE PWM signal and the error committed for **im** = 52%

In order to assess the accuracy of the proposed ANNSHE PWM, the obtained results demonstrate that the error between the theoretical angles and the predicted angles using the Neural Network algorithm is on the order of 10^{-3}. This indicates a high level of accuracy and confirms the effectiveness of the ANNSHE PWM approach in approximating the desired angles.

3.5 Polynomial interpolation

As discussed in section 2.7, the six datasets exhibit an interesting behavior. For each dataset we can create **M** row vectors where each row presents the angle α_M's values for all range of **im** in the dataset.



im	0,92	0,921	0,922	0,923	0,924	0,925	0,926	0,927	0,928
alpha_1	16,31199	16,29437	16,27674	16,25909	16,24143	16,22376	16,20607	16,18837	16,17066
alpha_2	37,6292	37,63184	37,63443	37,63696	37,63944	37,64186	37,64423	37,64653	37,64878
alpha_3	46,09098	46,06885	46,04667	46,02443	46,00212	45,97977	45,95735	45,93487	45,91233

Figure 3.8: An example on the **M** row vectors, hier $M = 3$

But, As we mentioned in section 2.8, we can not store all values for all angles. But what if we reduce the size of this vectors while preserving the information content, we can compress the values into a smaller set of parameters. This can be achieved through polynomial interpolation.

By utilizing polynomial interpolation, we can significantly reduce the dimensionality of the dataset while preserving the essential information.

3.5.1 Overview on the Polynomial interpolation

Polynomial interpolation is a mathematical technique used to approximate a function or a set of data points using a polynomial equation. The objective of polynomial interpolation is to find a polynomial function that passes through a given set of data points, enabling us to estimate values at intermediate points within the range.

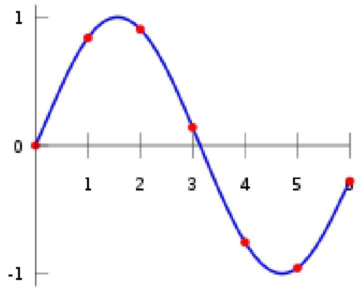


Figure 3.9: Interpolation example

There are several commonly used methods for polynomial interpolation, each with its own advantages and applications. While polynomial interpolation allows us to find a polynomial that passes through a given set of data points, it is important to note that in general, the resulting polynomial is not unique. The freedom to choose the coefficients of the polynomial can lead to different interpolating polynomials that fit the data equally well.

However, it is crucial to highlight that the uniqueness of the interpolation polynomial is guaranteed when the degree of the polynomial is equal to the number of data points minus one. In such cases, there exists a unique polynomial that passes through all the given data points. This property, known as the **Uniqueness Theorem for Interpolation Polynomials** [3], provides confidence in the accuracy and reliability of polynomial interpolation when applied correctly.

Polynomial interpolation finds applications in various fields such as mathematics, physics, engineering, and computer science. It allows us to estimate unknown values, fill in missing data points, and approximate functions. However, it is important to note that polynomial interpolation may introduce errors or inaccuracies, especially when extrapolating beyond the range of the given data or when dealing with noisy or sparse data.

3.6 Our original approach : Polynomial Interpolation Image Compression for asynchronous motor speed control

In reference to section 2.7, it has been noted that the database has been segregated into six distinct datasets, each of which displays a linear pattern. This observation has spurred numerous researchers to seek alternative methods of approximation. Some researchers have attempted to approximate the cosine functions within the non-linear system equations using a linear sinusoidal approximation, followed by re-solving the system [24]. While this approach significantly reduces the response time required to generate the angles, it is still impractical for online applications due to its considerable duration.

Another approximation involves approximating the trajectories of angles in two steps. The first step entails approximating the linear portion of each angle trajectory within one of the six data sets, followed by approximating the non-linear part for each angle [22]. However, this approach necessitates the use of loops, select cases[4], and divisions, which heavily consume hardware resources.

To address these challenges and overcome these limitations, we have developed our own approach based on polynomial interpolation and image compression that we named **PISHE PWM**.

3.6.1 PISHE PWM using MATLAB

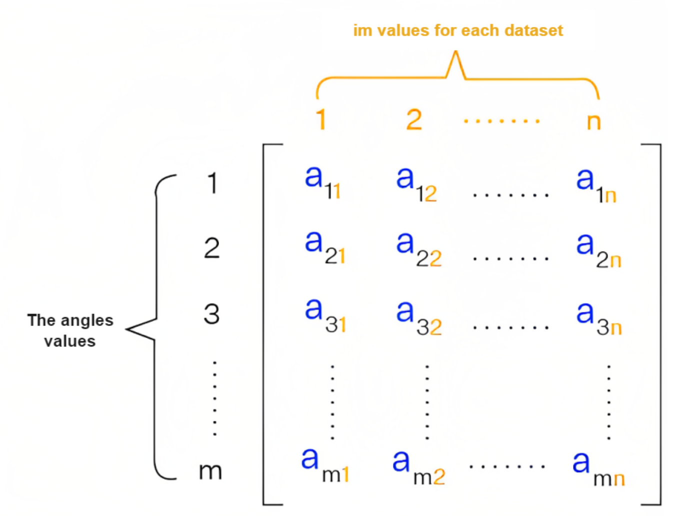
In our approach, we can consider each dataset as a digital image with dimensions of $m \times n$, where **m** is the number of the row vectors and **n** is the number of **im**'s values in the data set. This allows us to apply image processing techniques to the dataset. Specifically, we will use **Polynomial Interpolation Image Compression**.

The concept behind this technique is to transform discrete data into continuous data, enabling compression. This means that instead of needing to know the entire database to generate the SHE PWM signal, we only require knowledge of the coefficients of the

[3]If we have two interpolation polynomials, P(x) and Q(x), of degree at most n, and both polynomials pass through the same set of n+1 data points, then P(x) and Q(x) must be identical.

[4]it depends on the index of the angles

polynomials. With these coefficients, we can regenerate every value in the database. By employing this technique, we can effectively approximate the angles within the dataset.



$$\begin{array}{c|cccc} & 1 & 2 & \cdots & n \\ \hline 1 & a_{11} & a_{12} & \cdots & a_{1n} \\ 2 & a_{21} & a_{22} & \cdots & a_{2n} \\ 3 & a_{31} & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ m & a_{m1} & a_{m2} & \cdots & a_{mn} \end{array}$$

Figure 3.10: The dataset as a $m \times n$ image

Now, mathematically we can write each row vector as follows

$$\alpha_i = P_i(im), \quad i = 1...M \tag{3.8}$$

To perform the interpolation, we will utilize the **Cubic Spline** interpolation method. It is a commonly employed technique that involves fitting a piecewise cubic polynomial to the given data. Unlike polynomial interpolation, where a single polynomial is used to approximate the entire dataset, Cubic Spline interpolation breaks the dataset into smaller intervals and fits a cubic polynomial to each interval.

$$P_i = \sum_{k=0}^{3} a_{ik}(im)^k = a_{i3}(im)^3 + a_{i2}(im)^2 + a_{i1}(im) + a_{i0} \tag{3.9}$$

It offers several advantages like flexible and visually appealing approximation that avoids excessive oscillations often encountered in interpolations using high-degree polynomials. The piecewise[5] nature of cubic splines makes it suitable for our case.

In MATLAB, interpolation becomes easy and straightforward with the built-in function `polyfit`. This function facilitates polynomial interpolation by fitting a polynomial to a given set of data points 3.1. It is important to note that, due to the satisfaction of the conditions outlined in the Uniqueness Theorem for Interpolation Polynomials, regardless of the interpolation method employed, we will always obtain the same polynomial.

[5]A piecewise function is a function defined by multiple sub-functions, where each one applies to a different interval in the definition domain

As example the table 3.1 represents the polynomials for 3 angles dataset.

Table 3.1: Calculated polynomials angles for $92\% < im \leq 100\%$ using PISHE PWM

$P_{\alpha_1} = -21.3165 \times im^3 + 52.7283 \times im^2 - 60.5146 \times im + 43.9551$
$P_{\alpha_2} = -168.1964 \times im^3 + 441.4259 \times im^2 - 382.5502 \times im + 146.9253$
$P_{\alpha_3} = -160.3138 \times im^3 + 418.5186 \times im^2 - 385.1841 \times im + 171.0610$

At this point, we can conclude that for the original 3×80 values stored in the database representing the 3 angle SHE PWM signal, we have successfully reduced it to only 3×4 parameters (polynomial coefficients) that represent the compressed version of the original image.

This significant reduction in the number of parameters demonstrates the effectiveness of the compression techniques applied.

3.6.2 PISHE PWM Interpretation

To evaluate the performance of PISHE PWM, we compared the angles obtained from polynomial substitution with the theoretically calculated angles using Newton Raphson. We then created an error diagram to visualize the discrepancies between the two sets of angles as seen in 3.11.

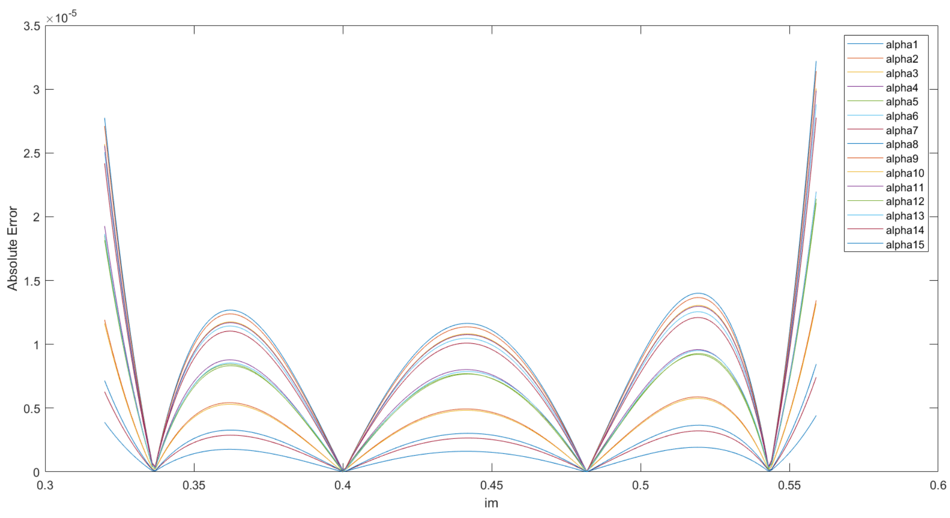


Figure 3.11: The absolute error of the predicted angles for **im** = 52%

By utilizing MATLAB and plotting error diagrams similar to the one mentioned in 3.11, we can construct the tables presented below:

Table 3.2: The absolute error of the predicted angles for all intervals of **im**

Modulation index (**im**)	Maximum absolute error in degrees
$0.001 < im \leq 0.159$	8×10^{-7}
$0.16 < im \leq 0.319$	1.8×10^{-6}
$0.32 < im \leq 0.559$	3.5×10^{-5}
$0.56 < im \leq 0.759$	1.5×10^{-4}
$0.76 < im \leq 0.919$	6×10^{-4}
$0.92 < im \leq 1$	4×10^{-4}

And to summarize the method performances we have

Table 3.3: PISHE PWM compression results

Data Base size : 12691 information (angle values)				
Number of angles	Original quantity of informaion	quantity of informaion after compression	Compression ratio	Absolute error of compression
23	3657	92	97.48%	8×10^{-7}
19	3021	76	97.48%	1.8×10^{-6}
15	3585	60	98.32%	3.5×10^{-5}
7	1393	28	97.98%	1.5×10^{-4}
5	795	20	97.48%	6×10^{-4}
3	240	12	95%	4×10^{-4}
New data base size : 288 information				

Instead of storing 12,691 values, we will now store only 288 values, which correspond to the coefficients of the polynomials. This significant compression of almost **97.73 %** allows for a much more efficient utilization of memory while still retaining the necessary information for accurate reconstruction of the original dataset.

3.7 PISHE PWM VS ANNSHE PWM

For the sake of comparison we consider that ANN approach[6] is also a method of compressing the data in the dataset, we can proceed with the following comparison.

Table 3.4: ANNSHE PWM vs. PISHE PWM quantity of information needed to reconstruct the database

Data Base size : 12691 information (angle values)			
Number of angles	Original quantity of informaion	quantity of informaion after PI compression	quantity of informaion after ANN compression
23	3657	92	48
19	3021	76	40
15	3585	60	32
7	1393	28	16
5	795	20	12
3	240	12	8
New data base size : $156 + 1 = 157$ information			

The addition of 1 in the new data size refers to the inclusion of information about the activation function used in all six ANNs.

Referring to the original work by Dr. Guellal [14], we can state that the average absolute error of ANNSHE PWM is, on average, in the order of 10^{-2}.

[6]Actually, the ANN approach is a method based on predicting the angles

When comparing the two algorithms, it is evident that interpolation is significantly more accurate than the Neural Network, with a minimum difference of 100 times in terms of accuracy.

However, as we discussed, our main interest lies in the implementation of the algorithm that functions in real time. Therefore, to make a fair comparison, we will evaluate both algorithms in a real-world application from the hardware perspective. This involves implementing 3-angles SHE PWM of both methods on the same Arduino UNO and then comparing their performances.

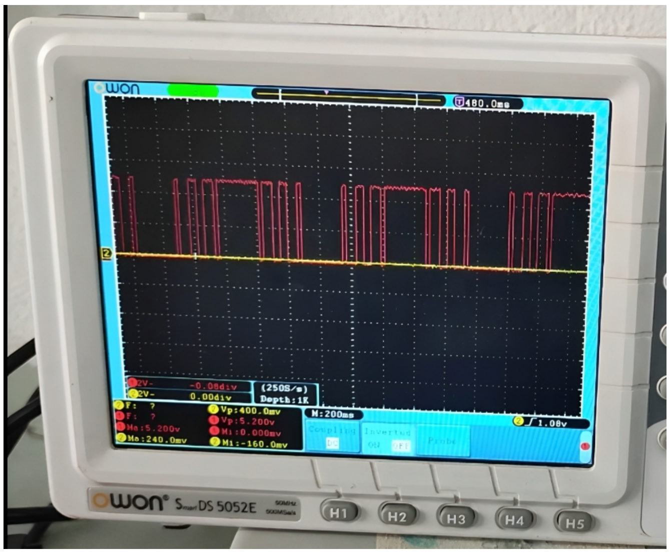
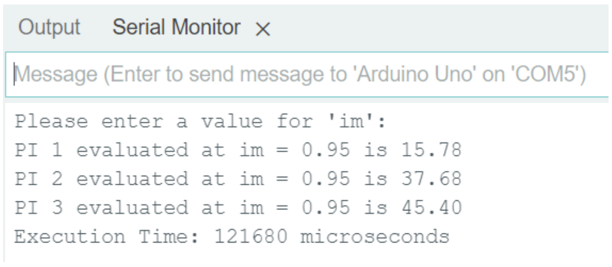


Figure 3.12: Output of arduino uno in both cases

At this stage, the error difference between the two methods for our application is negligible, and practically the output signals are the same. However, what we have observed is the difference in response time when changing the value of **im** within the 3-angle interval.

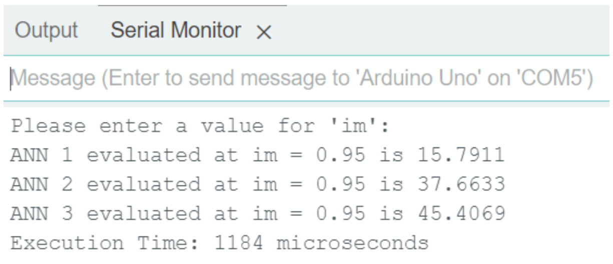
The execution time shown in Figure 3.13 clearly demonstrates that the response time and execution time of implementing ANNSHE PWM are significantly smaller compared to PISHE PWM, by approximately 100 times. This difference is more understandable when considering that for the same dataset, the neural network has fewer parameters and far fewer arithmetic operations needed compared to the polynomial interpolation approach as demonstrated in table 3.4.

However, when comparing the accuracy of the angles between the two algorithms, as shown in Figures 3.7 and 3.11, it is evident that the PISHE PWM method outperforms the ANNSHE PWM method. The error committed by the PISHE PWM algorithm for the same value of **im** is approximately 100 times smaller than that of the ANNSHE PWM algorithm. This indicates that the PISHE PWM method is more accurate and precise in approximating the desired angles.



```
Please enter a value for 'im':
PI 1 evaluated at im = 0.95 is 15.78
PI 2 evaluated at im = 0.95 is 37.68
PI 3 evaluated at im = 0.95 is 45.40
Execution Time: 121680 microseconds
```

(a) PISHE Time Response



```
Please enter a value for 'im':
ANN 1 evaluated at im = 0.95 is 15.7911
ANN 2 evaluated at im = 0.95 is 37.6633
ANN 3 evaluated at im = 0.95 is 45.4069
Execution Time: 1184 microseconds
```

(b) ANNSHE Time Response

Figure 3.13: Comparison of Time Responses using Arduino Uno for **im** = 95%

The indications shows that the ANN approach is more efficient and requires less computational overhead, making it a superior choice in terms of execution time, performance and most importantly hardware implementation.

In our specific application, achieving an accuracy of 10^{-4} (or more) is not necessary and can result in increased latency and resource consumption. From a driver's perspective, a precision of 10^{-2} offered by the ANNSHE PWM method is widely acceptable and requires relatively less hardware resources. Therefore, considering the trade-off between accuracy and resource consumption, the ANNSHE PWM algorithm will be the preferred choice for implementation on an FPGA.

3.8 Conclution

To address the challenge of real-time speed control of ASM for an EV, our focus in this chapter has been on the discussion of an on-line SHE PWM algorithm with real-time characteristics. We explored two potential approaches ANNSHE PWM and PISHE PWM.

The ANN-based algorithm, proposed by [14], demonstrated high-speed calculation of switching moments for the PWM signal with a speed that is 100 times faster then PI algorithm, and with an acceptable precision with a maximum error $\epsilon \leq 6\times10^{-3}$. Considering our application, which prioritizes real-time control, achieving precision to two decimal places is deemed sufficient.

On the other hand, our work based on image compression with polynomial interpolation excelled in achieving a high level of precision, reaching values close to the exact angles with a minimum factor of 100 times greater precision compared to the ANN algorithm.

In making our decision, we considered the primary objective of real-time control for ASM in an EV. While both ANNSHE PWM and Polynomial Interpolation offer commendable precision, we concluded that the ANNSHE PWM algorithm meets our requirements, considering its significantly faster calculation speed and the sufficient precision it provides even thought it demands high implementation complexity.

In the following chapter, This algorithm will be implemented on an FPGA circuit to generate the PWM signal in real time, we will exhibit the different steps that we followed to develop and test the ANNSHE PWM Algorithm.

Chapter 4

Implementation of the On-Line ANNSHE PWM algorithm

4.1 Introduction

Taking into account the accuracy and resource consumption, we embarked on implementing the ANNSHE PWM algorithm for EV propulsion system. Initially, we used a micro-controller for this purpose. However, despite the algorithm not demanding significant resources as we only implemented one of the six ANN models, the micro-controller proved inadequate in handling the task effectively. Recognizing the need for a better approach, we set out to explore alternative hardwares that could meet our requirements.

The common approaches to the implementation of ANNs are using Digital Signal Processors (DSPs), Application Specific Integrated Circuits (ASICs), Field Programmable Logic Arrays (FPGAs) or a combination of them. DSP-based implementation is sequential and hence does not preserve the parallel architecture of the neurons in a layer. ASIC implementations are used whenever the application requires performance beyond the abilities of current DSPs but they do not offer reconfigurability by the user. FPGA is the most suitable hardware for ANN implementation as it preserves the parallel architecture of the neurons and can be reconfigured by the developer [29].

The programmable logic (FPGA approach) provides a solution that combines the best of both DSP and ASIC technology, without their respective limitations. FPGAs are programmable and changeable (like DSPs); the designer can make changes quickly, without the additional cost and time of an ASIC design. On the other hand, FPGA implemented algorithms can process information faster than a general purpose DSP.

However, implementation of ANNs in an FPGA has some challenges, as ANN hardware implementation requires large resources because of the nonlinearity of activation function and enormous number of arithmetic operations.

In this chapter, we will present the architecture that minimises the complexity of the hardware's implementation and is appropriate for our application as mentioned in 1.5 low latency response is vital for modern EVs. This architecture comprises 6 crucial steps to effectively implement the ANNSHE PWN algorithm on fpga with its six different ANN models 2.10, using minimum resources, and ensuring a fast and efficient response.

4.2 Development method of the ANNSHE PWM Implementation

We consider it unnecessary to dedicate an entire section to introducing FPGA circuits or the VHDL hardware description language for the implementation of our algorithm. These topics are not the primary focus of our work and can be readily found and studied on the internet.

In our project, we aim to use FPGA hardware for the implementation of ANNSHE PWM to control the Induction Motor's speed for EVs. Specifically, we will employ the ANNSHE PWM algorithm proposed by [11] to achieve this goal. The algorithm does not require a learning algorithm to be implemented, as the necessary parameters, weights, and thresholds are computed offline. To develop the overall structure for implementing the ANNSHE PWM algorithm on the FPGA, we refer to the general architecture of ANNs depicted in Figure 3.5, we see that we have six distinct ANNs corresponding to different **im** intervals, as outlined in Table 2.2. By following the flow chart presented in Figure 4.1, we can obtain a more detailed description of how the entire algorithm will be executed on the FPGA, enabling precise control of the electric motor speed in the EV.

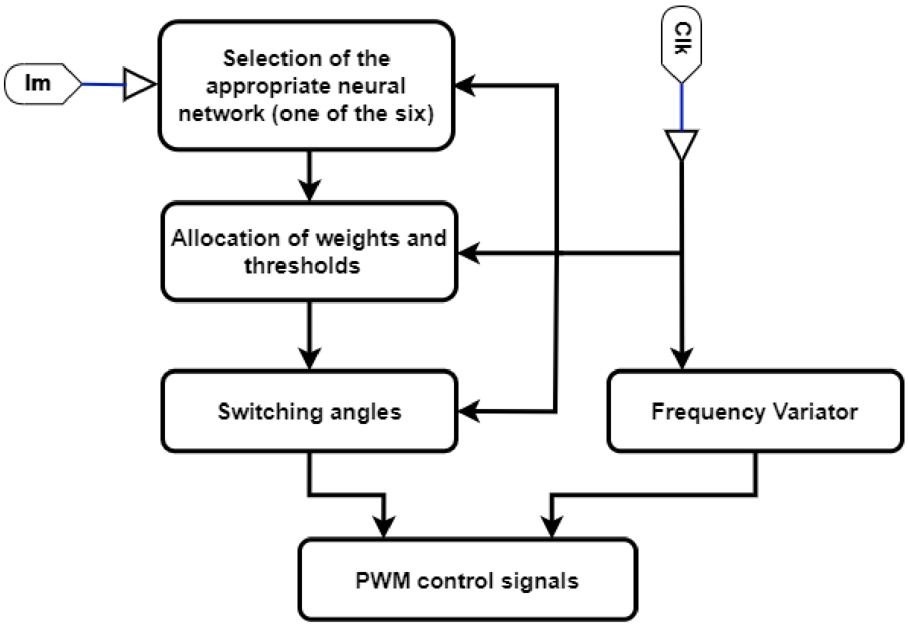

Im
Clk
Selection of the appropriate neural network (one of the six)
Allocation of weights and thresholds
Switching angles
Frequency Variator
PWM control signals

Figure 4.1: Flowchart of the proposed **ANNSHE PWM** algorithm [11]

The VHDL code serves the purpose of designing a circuit that can generate the six PWM signals from two inputs, the modulation index **im** and a clock. The detailed flow chart of the **ANNSHE PWM** algorithm used to deploy the VHDL code is shown in Figure 4.2.

Examining the structure of the **ANNSHE PWM** algorithm, as displayed in Figure 4.2, it is evident that there are six modules with a distinct role and crucial steps. In the subsequent subsections, we will see the role and specific details of each module, as well as outline the step-by-step process from the input parameters to the output signals within this structure.

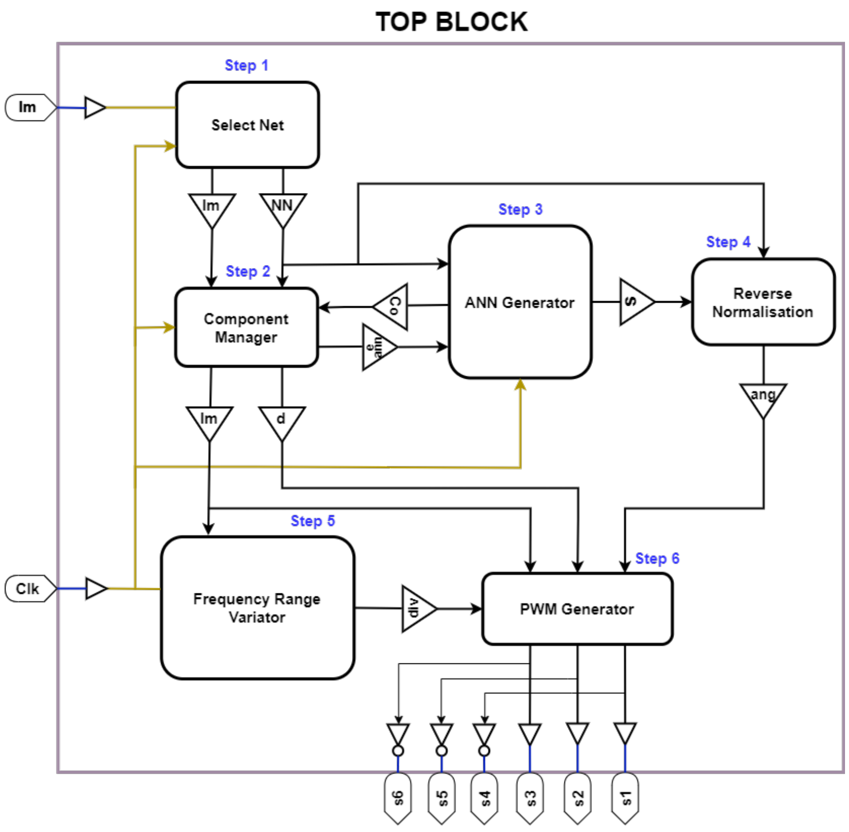


Figure 4.2: Detailed architecture of the proposed **ANNSHE PWM** algorithm

4.2.1 Top block

This block represents the global entity depicted to facilitate the interconnection of all internal blocks within the **ANNSHE PWM** algorithm. It serves as a central unit that enables communication and coordination between the various components, including the normalization block, ANN generator, frequency range variator, internal frequency regulator, and the PWM generator.

4.2.2 Select Net

As we have already stated earlier, the **im** interval has been divided into 6 sub-intervals. For each sub-interval, **ANN-i** is attributed in the structure.

Therefore, the purpose of the "Select Net" module, shown in the figure 4.3, is to select the appropriate **ANN-i** network which is suitable for the input **im**.

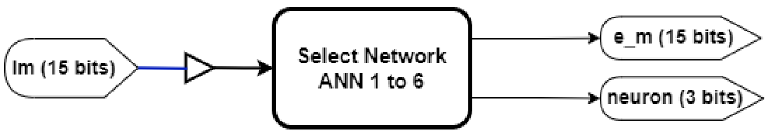


Figure 4.3: Synoptic Diagram of the Select Net Module

During the off-line learning phase, the modulation index (**im**) was expressed as a percentage value ranging from 0% to 100%. In our design, we allocated 14 bits for the input **im**, with seven bits dedicated to the integer part and seven bits allocated for the fractional part. As a result, the variation pitch of im is set at 0.0078%. It is worth noting that this variation pitch can be adjusted depending on the desired number of fractional bits for **im**. This entry has been adapted to the entry of our different ANNs by another module that we will introduce in the next step.

To simplify the selection of each interval based on **im**, we have defined the boundaries of each interval, as outlined in Table 4.1. This approach effectively minimizes the FPGA circuit's space consumption.

Table 4.1: The Intervals for the Variation of **im** of each **ANN-i**

ANN	Interval of variation of **im** in percentage $im_{\min} \le im < im_{\max}$	The lower limit in binary	(neuron)
RNA-6	$01\% \le im < 16\%$	00000000000000	101
RNA-5	$16\% \le im < 32\%$	00100000000000	100
RNA-4	$32\% \le im < 56\%$	01000000000000	011
RNA-3	$56\% \le im < 76\%$	01110000000000	010
RNA-2	$76\% \le im < 92\%$	10010000000000	001
RNA-1	$92\% \le im < 100\%$	10111000000000	000

Based on the information provided in Table 4.1, a VHDL code is developed for the Select Net module using combinatorial logic. This module's purpose is to determine and select the suitable ANN-i network based on the input value **im**.

4.2.3 Component Manager

The block given in figure 4.4 is the second element which is called in the Top block, it allows to manage the external environment of the chosen **ANN-i** according to the input **neuron** attributed from step one.

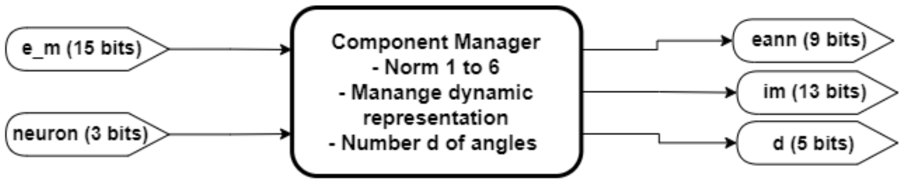


Figure 4.4: Synoptic diagram of the COmponent Manager module

Each internal **ANN-i** network possesses its own unique structure and data representation, including the size of the integer and fractional parts. The primary function of the Component Manager block is to associate the appropriate normalization block with each internal ANN.

Another crucial role of this block is to address the challenge of managing changes in the size of the input **im** for each block where it is utilized. Such changes can lead to compatibility issues in the connection of these blocks. There are two types of inputs: one consisting of 15 bits with a dynamic representation, which serves as the input for the normalization block to be used later by the **ANN** block, and the other representing inputs are injected into the blocks specific to the SHE PWM application. The block ensures that each input is adapted to its respective format based on the selected **ANN-i** network.

The advantage of employing a dynamic input for each ANN-i network becomes evident when considering an example with neuron=”101” corresponding to ANN-6. In this case, the input **im** ranges from 1% to 16%, necessitating only 4 bits to represent the variation range within this interval. Consequently, the remaining bits can be utilized to increase the precision of the input, enabling more accurate computations, particularly for low-speed variations.

Additionally, this block plays a crucial role in determining the appropriate number **d** of switching angles. The value of **M** varies for each **ANN-i** network based on the input **im**.

4.2.3.1 Normalisation Component Block

The normalization block serves the purpose of centralizing and normalizing the inputs of the **ANN** block according to the principal explained in 3.3 with the equation 3.8.

A significant challenge we encountered is the management of input sizes for each block related to the Normalization Component Block. Without proper management, this can lead to significant problems and incorrect results. It is crucial to carefully handle the sizes of inputs to ensure compatibility and accurate processing throughout the algorithm.

In the case of the **ANNSHE** algorithm, the input is represented by 15 bits in two’s complement format. When using a fixed size for both the integer and fractional parts, there can be a loss of precision. While, if the input is dynamic, the product used for normalization is also dynamic. This is where the importance of this block comes into play, as it allows for choosing the appropriate number of bits for each case. By adjusting the number of bits used for normalization, we can ensure that the precision of the algorithm is maintained and that the dynamic input values are properly handled. this can be noticed when we compare having an input **im** of 1% with 99%, if we used the same data representation for both inputs we gonna have a problem of precision. In the first case we would need only 4 bits for integer part, but for 99% the full 7 bits are used, the difference between the bits used is managed by the normalization component block to adapt the output with other portions of the code.

To ensure its functionality with minimum surface consumption [1], data normalisation block was designed using a strategy that makes the data dynamic, it assigns properly the size of the integer part and the fractional part for the next bloc.

[1]Surface consumption is a measure of how much of the available resources are being used by a particular design or configuration.

4.2.4 ANN Generator

The ANNs as dicussed in 3.1 have several aspects to take into account when designing ANNs' circuits, these include data representation, products and sum of products computations, and activation functions implementations.

The most important and challenging to implement are the inner-products and non-linear activation function, the complex that exhibit is the main reason behind the massive FPGA resources' consumption.

4.2.4.1 Data representation

A neural network operates with real numbers, it can be represented in many ways. However, Getting relatively good numeric precision is important for accuracy, convergence and resources conserving. Choosing the floating point representation is ideal, since it offers the greatest amount of precision (i.e. minimal quantization error), however using it in ANN implementation is not quite a good choice since it consumes huge amounts of resources which are very limited in this case.

Instead, fixed point with dynamic representation is used. Even though this means less precision, its benefits compromise its inconveniences. It is more area-efficient than floating point, much simpler in arithmetic operations and even From a driver's perspective, at this precision level, the encountered issues is totally tolerable.

In relative work [30], it was mentioned that several studies have shown that selecting 16 bits for weights and 8 bits for the input and output of the activation function yields excellent results. However, in the development of ANNSHE PWM [14], they opted to use 15 bits for weights and 9 bits for the activation function. This choice was made based on their specific requirements and considerations. After careful evaluation, it has been determined that this particular configuration yields favorable outcomes in their endeavors. Consequently, we have decided to adopt this approach.

4.2.4.2 Inner products

Multipliers has been identified as the most surface consumption arithmetic operators used.

To do a such arithmetic operation, we used direct full parallel bit multipliers called DSP slices. In FPGAs we have a limited number of these special multipliers, they must be used wisely. In the architecture of the ANN Generator, only 6 multipliers have been used in parallel. This architecture causes more latency in computation, but in the other hand it is more surface efficient and benefices the FPGA's parallelism. Hence, the other FPGA resources can be used for future needs like adding different control commands that modern EVs consists of, for example a Battery Management Unit or an Object Detection Control Unit.

In this architecture we will use multiplier per neuron at a time, it receives the neuron's associated inputs and multiply them by their appropriate weight in a sequential way.

4.2.4.3 Implementation of the Activation Function

Many efforts have been made to efficiently implement non-linear activation functions in hardwarilization of an ANN systems, aiming to maximize efficiency caused by the various arithmetic operations involved in the computation. Using polynomial expansion, researchers have been actively working on developing conventional arithmetic circuits that can deal with this problem more efficiently.

Within our architecture, we hold a specific interest in utilizing the Hyperbolic Tangent and Linear Activation functions. In a related study, the linear function has been successfully implemented, as documented in [26]. As for the tanh function, we have chosen to adopt an approach heavily influenced by the work of Promod Kumar Meher, who proposed an optimized addressing scheme for the LUT-based implementation of the hyperbolic tangent function in his article [31].

1. **Implementation of the Hyperbolic Tangent**

The Hyperbolic Tangent is defined by the following equation:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.1}$$

It produces an S-shape curve as shwon in the figure 4.5

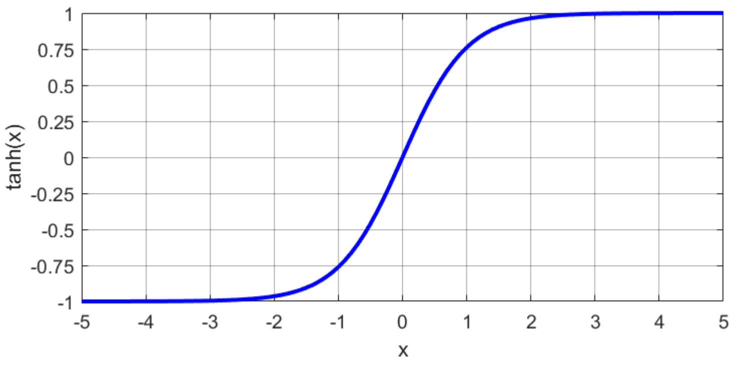


Figure 4.5: The Hyperbolic Tangent activation function

As presented in [31], the approach involves dividing the tanh curve into distinct intervals and constructing a Lookup Table (LUT) with the minimum number of elements while minimizing the associated error by using the various characteristics of the function.

1.a. Characteristics of tanh and strategy of implementation

- **Property 1**

$$tanh(-x) = -tanh(x) \tag{4.2}$$

Considering the mirror symmetry property about Y-axis, we need to store only the right half of the curve in 4.5 where $x > 0$. We can handle the negative side by reversing the stored positive values, this is done by using the 2's complement operation if the input is negative.

- **Property 2**

$$\lim_{x\to 0} tanh(x) = x \tag{4.3}$$

Equation 4.3 implies that when the input is small, the *tanh* is approximately linear. As a result, there is no need to store values in the LUT for this particular range, as the required values can be directly derived from the input values. This could be implemented by a simple Multiplexer.

- **Property 3**

$$\lim_{x\to\infty} \frac{dtanh(x)}{dx} = 0 \tag{4.4}$$

$$\lim_{x\to\infty} tanh(x) = 1 \tag{4.5}$$

According to the two equations 4.4 and 4.5, *tanh* is approximately constant for big input values, for $|x| \geq 3$ the variation is less than 0.00015 and we can store a value of +1 in the LUT for all values above 3.

By using these properties all together, we are left to find the other LUT's values.
Within the range of $\delta_{min} < x < \delta_{max}$, where δ_{min} and δ_{max} represent the boundaries of the non linear interval, where the tanh is approximated using LUT. This approximation maintains a permissible error threshold, ensuring accurate representation of the function within the specified interval. We can measure the error committed as the following

$$\begin{cases} |\tanh(\delta_{min}) - (\delta_{min})| \leq \epsilon \\ |\tanh(\delta_{max}) - (1)| \leq \epsilon \end{cases}$$

Where ϵ is the maximum allowable error. Naturally, for a specific ϵ we must choose δ_{min} and δ_{max} accordingly.

- $\boldsymbol{\delta_{\mathbf{max}}}$
It is the upper bound such that "$x \geq \delta_{max} \implies tan(x) = 1$", this particular x is as +1.
if we consider the case of $\delta_{max} = 3$ we will have $|\tanh(x) - \tanh(3)| < 0.00015$, a

very high accuracy than what is required for many applications including ANN's applications, from the Figure 4.6 we can also notice

$$\begin{cases} \text{for } x = 2.0, \tanh(x) > 0.96 \\ \text{for } x = 2.4, \tanh(x) > 0.98 \\ \text{for } x = 2.7, \tanh(x) > 0.99 \end{cases}$$

It means, tanh can be approximated by +1 for $x > \delta_{max}$ where δ_{max} is $2, 2.4, \text{or } 2.7$ if the maximum allowable errors are 0.04 , 0.02 , or 0.01 respectively.

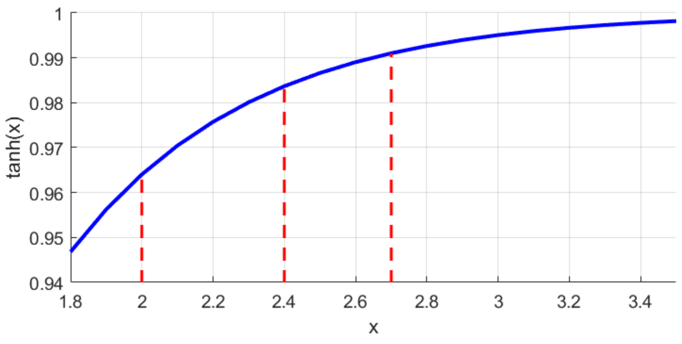


Figure 4.6: Saturation region of Hyperbolic Tangent Sigmoid

- $\delta_{\mathbf{min}}$

  Using the same approach, the determination of δ_{min} depends on the accuracy that we need for our application.

  Using the Taylor expansion, we get

$$\tan(x) = x - \frac{x^3}{3} + \frac{x^5}{15} - \frac{17x^7}{315} + ... \tag{4.6}$$

  For $x \to 0$ high order terms can be ignored. by assuming $\tanh(x) = x$ we can find δ_{min} using the equation 4.7

$$\left| \delta_{min} - \left(\delta_{min} - \frac{\delta_{min}^3}{3} + \frac{\delta_{min}^5}{15} \right) \right| < \epsilon \tag{4.7}$$

  By simplifying 4.7 we find

$$\frac{\delta_{min}^3}{3} - \frac{\delta_{min}^5}{15} \leq \epsilon \tag{4.8}$$

  Based on the above equation 4.8 we have that $\delta_{min} = 0.390625$ for an error $\epsilon = 0.02$.

As shwon in Figure 4.5, the rate of variation of $\tanh(x)$ when $\delta_{min} < x < \delta_{max}$ is not uniform, therefore not all the input values correspond to a different LUT value for a given accuracy. By knowing this, one single value can be stored for several input values forming sub-domains.

1.b Optimized Lookup table design for the tanh

Unlike conventional lookup tables where each input value (address) corresponds to one location of the LUT which represent the memory unit, Maher used in his work [31] what is so called range addressing which like an addressing scheme, where one address corresponds to a range of input values that have the same value of tanh stored in the LUT, reducing by that the number of stored words.

The methodology for the addressing scheme used in [31] is that for a given sub-domain the value stored is the mean of the boundary values of the function in that sub-domain. This is unlike other works where they stored the function value corresponding to the lower-boundary address, and here the difference between the maximum and the minimum values of the function could be the double of the allowable error.

Designing the LUT is then done by following the steps that [31] proposed, which are:

1. **Determination of the Lower and Upper Limits of the LUT Input**
   For $\epsilon = 0.02$, $\delta_{min} = 0.390625$ and $\delta_{max} = 2.4$.

2. **Selection of the Address Width (precision)**
   By simulations in Matlab [31], it was found that a width of 9 bits for the input values represented in 2's complement allow to have $\tanh(x_2) - \tanh(x_1) \leq \epsilon = 0.02$ where $x1$ and $x2$ are two consecutive inputs.

3. **Selection of Domain Boundaries**
   The range of $\tanh(x)$ for $0.3906250 < x < 2.4$ is divided into n sub-domains $R_i\left(x_{i1}, x_{i2}\right)$ such that $\left|\tanh\left(x_{i1}\right) - \tanh\left(x_{i2}\right)\right| \leq 2\epsilon$.
   With $1 \leq i \leq n$, given that n is the smallest integer where $x_{n2} \geq 2.4$ then all $\left(x_{i1}, x_{i2}\right)$ should be determined.
   In our case we have $n \geq \left\lceil \frac{\tanh(2.4)-\tanh(0.3906250)}{2\epsilon} \right\rceil$, which gives us $n = 15$.

4. **LUT Assignment**
   The stored word is the mean of the boundary values of the function which means $\tanh\left(x_i\right) = \left[\tanh\left(x_{i1}\right) + \tanh\left(x_{i2}\right)\right]/2$.

We have successfully implemented a series of steps to create a straightforward LUT words. The outcomes of the efforts can be observed in table 4.2. It is worth noting that our approach differs from Meher's work in one crucial aspect. Instead of simply storing the exact mean value, we opted to store the value closest to the mean of the boundary range. This decision was made due to the inherent limitations of binary representation, which can result in errors within sub-domains surpassing the error criterion of $\left|\tanh\left(x_{i1}\right) - \tanh\left(x_{i2}\right)\right| \leq 2\epsilon$. Upon examining the table, we can see that the maximum error achieved is $\epsilon = 0.033$.

It is also important to note that we only used 15 words for the LUT, while in other works it would require 1024 words for same error limit.

Table 4.2: LUT for the hyperbolic tangent activation Function [26].

LUT N°	x_{i1}	x_{i2}	Mean tanh(x)	Stored value	Stored value in binary	Maximal error
	...	0.390625	x	N/A	N/A	N/A
1	0.390625	0.453125	0.398182	0.390625	00011001	0.0338394
2	0.453125	0.515625	0.44939	0.453125	00011101	0.0286606
3	0.515625	0.578125	0.497809	0.5	00100000	0.0256837
4	0.578125	0.640625	0.543313	0.546875	00100011	0.0255737
5	0.640625	0.703125	0.585836	0.578125	00100101	0.0282226
6	0.703125	0.78125	0.629886	0.625	00101000	0.0284236
7	0.78125	0.859375	0.67468	0.671875	00101011	0.0240605
8	0.859375	0.9375	0.715003	0.71875	00101110	0.0228145
9	0.9375	1.048675	0.757681	0.75	00110000	0.0312907
10	1.048675	1.171875	0.803082	0.796875	00110011	0.0279973
11	1.171875	1.328125	0.846831	0.84375	00110110	0.0250403
12	1.328125	1.53125	0.889714	0.890625	00111001	0.0218347
13	1.53125	1.859375	0.93163	0.9375	00111100	0.0268617
14	1.859375	2.90625	0.973329	0.96875	00111110	0.0252879
15	2.90625	...	NA	1	01000000	NA

2. **LUT structure for evaluation of Hyperbolic Tangent**

The LUT structure used is shown in figure 4.7. It is evident that the LUT receives a 9-bit input word represented in two's complement format. The input word then traverses through various blocks within the structure before giving the final output.

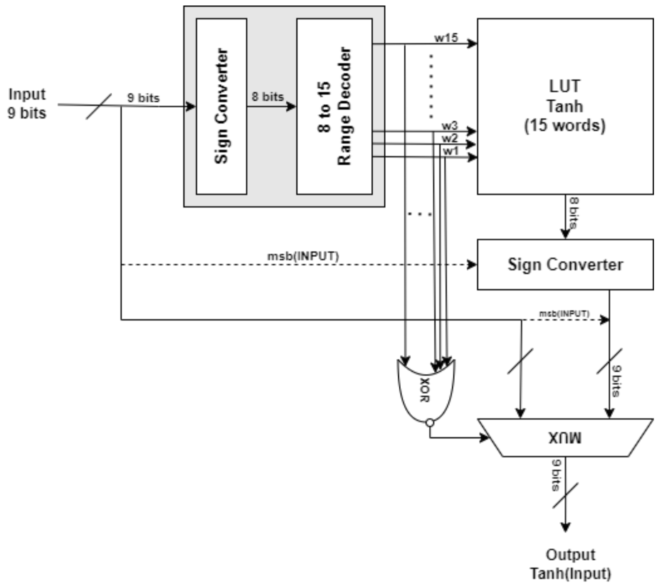


Figure 4.7: Hyperbolic Tangent LUT block structure

2.1 **Sign Converter**
It is implemented both before and after the LUT block simply to person the two's complement operation. Its primary function is twofold. Firstly, the first Sign Converter calculates the magnitude of the input to provide it as input to the LUT. Since the LUT only operates on positive numbers. Secondly, when the input is negative, the second Sign Converter negates the corresponding LUT word, allowing for the accurate evaluation of $\tanh(x)$ for negative inputs.

2.2 **Range Decoder**
It is incorporated to facilitate range-addressing operations. Given the input magnitude, the Range Decoder determines the appropriate range by utilizing its word-select outputs (w1, w2, ..., w15), it's implementation is well detailed in [31].

2.3 **Multiplexer**
It serves the purpose of selectively passing either the LUT words or the input directly. The decision to allow the input to pass through occurs when no word is selected by the Range Decoder. In other words, when reading the table 4.2, this corresponds to the condition where $|X| < 0.390625$. This functionality can be implemented by employing a NOR logic gate that operates on all the word-select signals generated by the Range Decoder outputs.

3. **Simulation of The Function**

Figure 4.8 illustrates the values of the word-select signals for the Range decoder block across the entire input range.

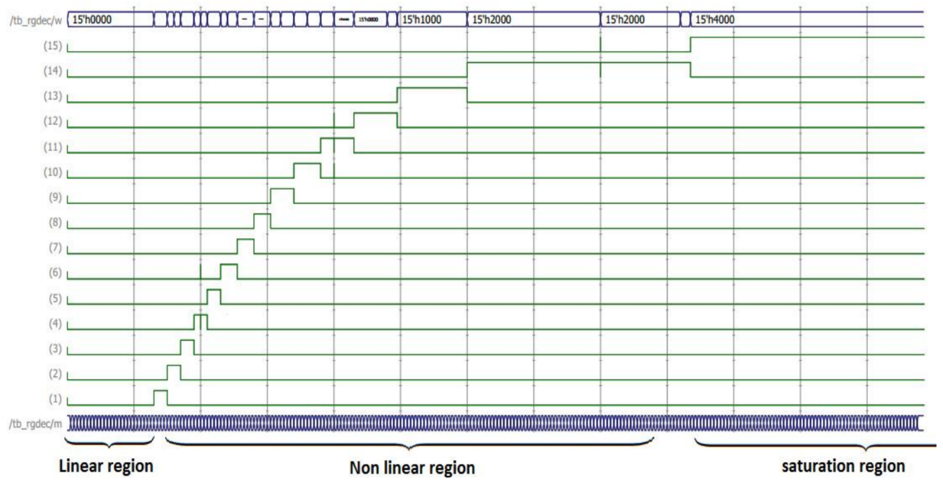


Figure 4.8: Range Decoder block signals simulation [26]

The figure 4.8 demonstrates that the inputs can be categorized into three distinct regions:

- **Linear Region:** This region corresponds to a linear relationship between the input and the output, where the input bypasses the LUT and directly propagates to the output. It is characterized by all word-select signals being set to a low level (0).

- **Nonlinear Region:** In this region, the shape of the tanh function becomes evident. Each word-select signal (w1, w2, ..., w15) assumes the value of 1 within its specific sub-range, while the remaining sub-ranges have these signals set to 0. This configuration allows for the formation of the desired Hyperbolic Tangent shape.

- **Saturation Region:** This region represents the saturation region of the system. Inputs falling within this region satisfy certain conditions.

From the analysis of 4.8, it can be concluded that the Range Decoder effectively performs its intended function. It decodes the input values into sub-domains, and these sub-domains can be identified when their corresponding word-select signals are set to 1.

Figure 4.9 provides evidence of the flawless performance of the Tangent function block. The figure demonstrates that the block operates perfectly under different scenarios. For instance, when the first input value falls within the linear range, the output equals the input value directly, showcasing the block's accurate translation of the linear relationship. On the other hand, the third input value resides within the nonlinear region, highlighting the block's ability to handle and process inputs within this specific range successfully.

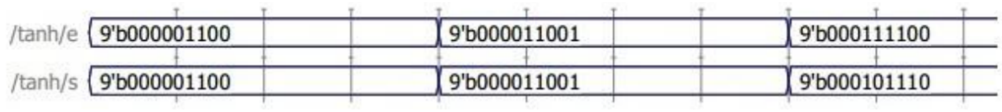


Figure 4.9: Hyperbolic Tangent block output simulation [26]

4.2.4.4 Neural Network Manager

This block allows to manage the internal environment of ANN-i, it is the first element that is used in the internal architecture of the ANN Generator, each internal ANN has its own anatomy and its own weights and biases, the role of this block is managing all these internal parameters for each ANN.

4.2.4.5 The architecture of the ANN Generator

The configuration of an Artificial Neural Network (ANN), including the number of inputs, outputs, layers, and neurons per layer, is unique to each application. However, implementing multilayer networks can require significant resources and may not be a practical solution for real-time applications, such as motor speed control for EVs. In order to address this challenge, we adopted an architectural approach inspired by the concept of layer multiplexing as introduced in the paper [29]. This innovative approach enables the implementation of a large number of ANNs while minimizing resource requirements.

By leveraging this principle, we were able to design a more efficient and resource-friendly solution for our specific application, accommodating the demands of real-time operations for motor speed control in EVs.

1. **Neural Network Implementation using Layer Multiplexing**

In a multi-layer feed-forward ANN, data flows sequentially from one layer to another, with computations occurring one layer at a time. This characteristic allows for a unique approach known as layer multiplexing, where it is unnecessary to implement all layers simultaneously. Instead, only the largest layer, typically the one with the maximum number of neurons, needs to be implemented. This layer is then called repeatedly, acting as different layers with the assistance of a control unit.

The control block plays a vital role in ensuring the complete computation of the ANN using layer multiplexing. It orchestrates the sequencing and placement of inputs, weights, biases, and the values of the activation function (obtained from the LUT) for each layer.

In contrast to conventional architectures, the work described in [29] deviates from implementing the complete ANN, as depicted in the Figure 4.10.a. Instead, it simplifies the architecture to focus only on the implementation of the layer-multiplexed ANN, as shown in the Figure 4.10.b.

Within our architecture, all the ANN models have a structure of 1-1-d (d is the number of outputs that changes depending on the input). Hence, we have implemented the largest layer as the output layer. This decision was made due to our ANN models having only one hidden neuron and d output neurons. Consequently, the largest layer effectively serves as the output layer.

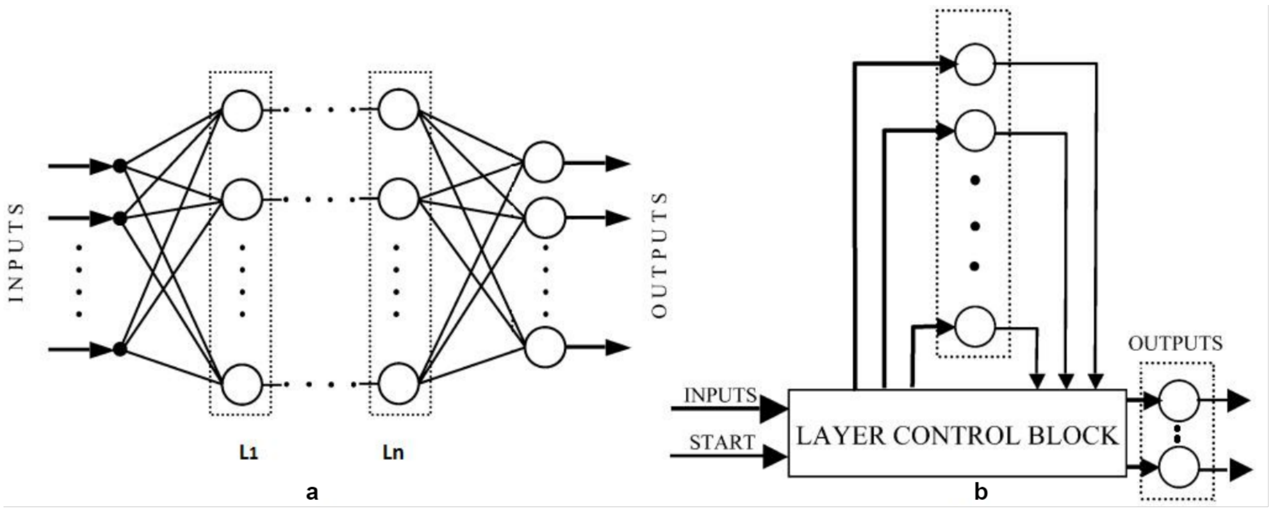


Figure 4.10: Layer multiplexed ANN [29]

2. **Single Neuron Architecture**

According to the mathematical model presented in Chapter 3, a neuron carries out two fundamental arithmetic operations. Firstly, it performs the computation of the sum of the products between inputs and their respective synaptic weights. Subsequently, the resulting sum is passed through an activation function to determine the neuron's output.

In our specific ANN models, the sum of products calculation is not required since we only have one hidden layer with one hidden neuron. However, the activation function is implemented through circuits illustrated in Figure 4.7. The overall architecture of a neuron can be visualized in Figure 4.11.

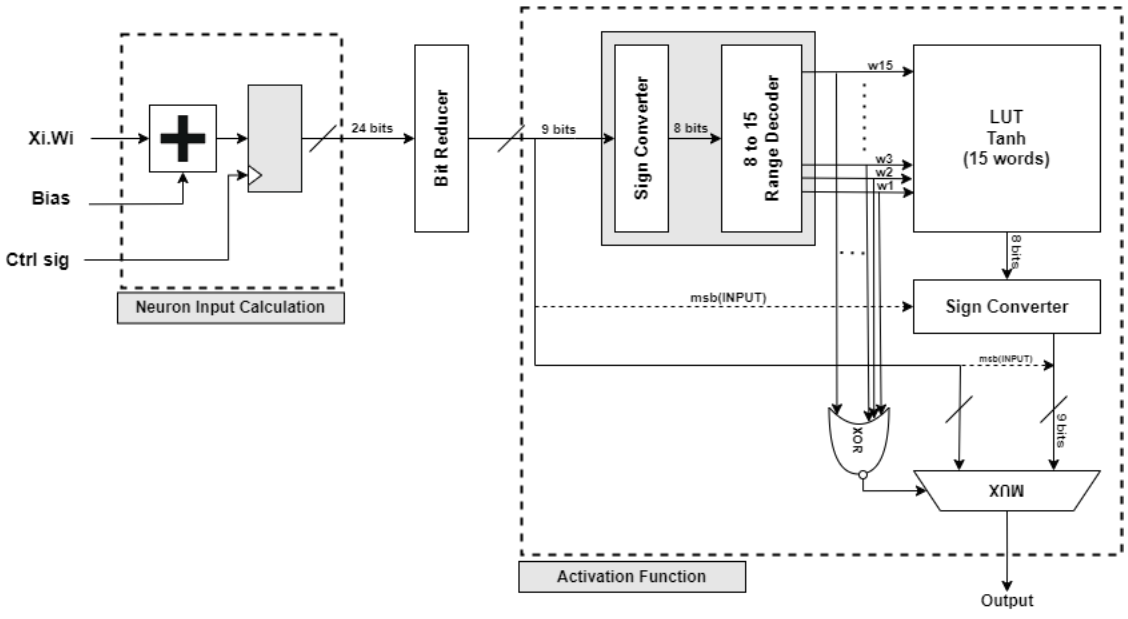


Figure 4.11: A Single Neuron Block Architecture

The weight width used in this work is 15 bits, consisting of 7 bits for the integer part and 8 bits for the fractional part. Consequently, the product of the weight (15 bits) with the input (9 bits) yields a 24-bit result. This result is then passed to an adder, which also produces a 24-bit output. However, a challenge arises when adapting the width of the data from the adder to match the input width of the activation function block without affecting the results. To address this issue, we introduced a block called the bitreducer, depicted in Figure 4.11, which performs this adaptation.

Due to the presence of a saturation region in the hyperbolic tangent function, the output of the LUT is consistently 1 for input values that verifies $|X| > 2.4$. Therefore, if the output of the adder surpasses 2.4, the bitreducer adjusts it to a value that can be represented in 9 bits, ensuring an output of 1 from the LUT. Specifically, since the LUT input has 3bits for the integer part, if the value is greater than 2.4, the bitreducer sets it to a value of 3, which can be represented using 9 bits with 3 bits for the integer part of the input to the LUT. Conversely, if the value is less than -2.4, the bitreducer sets it to

-4, as these values also yield an output of 1 from the LUT while fitting within the 9-bit representation.

This approach enables the appropriate adaptation of the data width from the adder to the input of the activation function block, ensuring that the results remain accurate while considering the saturation region properties of the hyperbolic tangent function and the limitations of the available bit representations.

3. The Global Architecture of The Implemented ANN

The figure 4.12 shows the block diagram of the global architecture of the implemented ANN in our application.

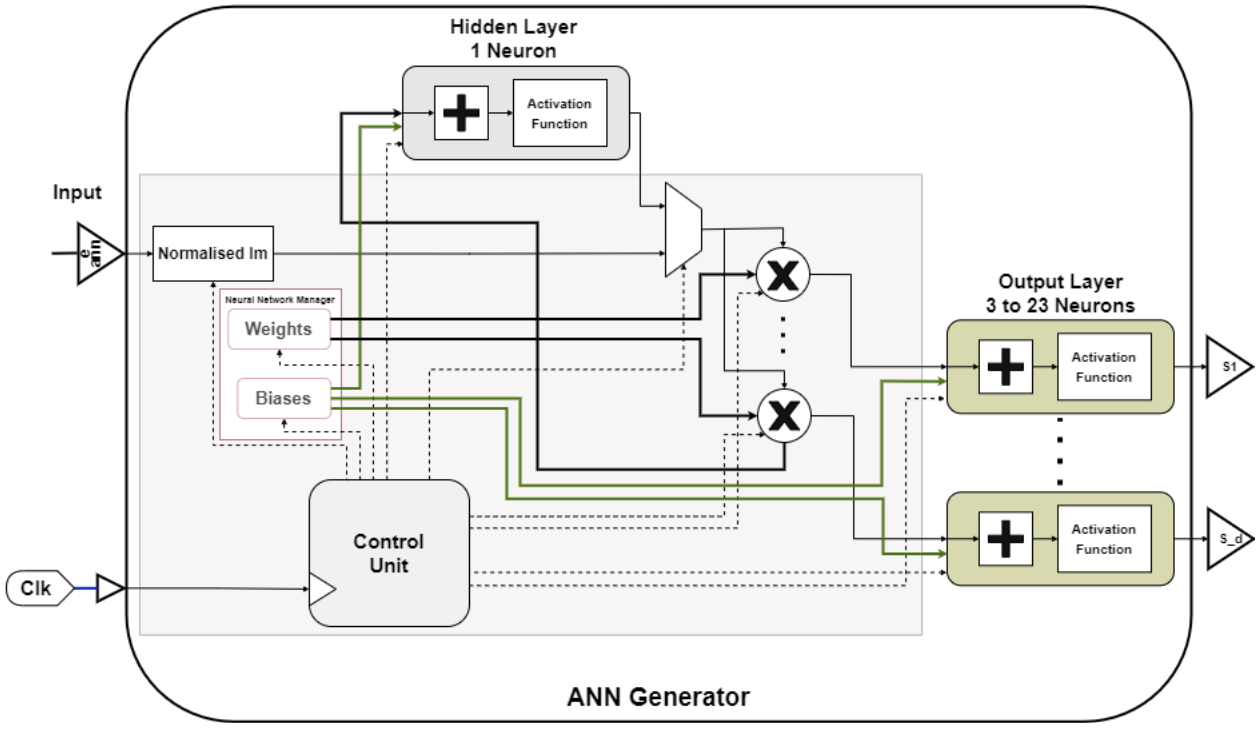


Figure 4.12: Global ANN model Architecture

The Figure 4.12 clearly illustrates the utilization of Layer multiplexing concept in implementing the output layers, along with the presence of a single hidden layer with one neuron as described earlier.

The Control Unit plays a crucial role in coordinating all the blocks and data, ensuring the proper functioning of the implemented layers, it simply contains a finite state machine that produces signals to synchronize all the ANN system.

In the diagram, the Normalised **im** value represents the output of the Component Manager block. Its purpose is to centralize and normalize the inputs of the neural network. This step is essential for the accurate operation of the neural network as we trained the ANN models with normalised inputs.

The Neural Network Manager block encompasses the biases and weights of all the neurons in the entire neural network. The Control Unit synchronizes the outputs of this block, ensuring that the biases and weights are provided at the correct time and with the appropriate values.

Overall, these components and the coordination by the Control Unit contribute to the effective functioning and management of the neural network system.

4.2.5 Reverse Normalisation

The reverse normalisation block allows the de-normalization of ANN outputs according to the principle explained in 3.3 in 3.7. Its implementation uses the same principle as that of the normalisation block.

The number of angles varies from [3 to 23] depending on both given inputs, the reverse normalisation block gives the final switching angles which will be used later on by the PWM generator module.

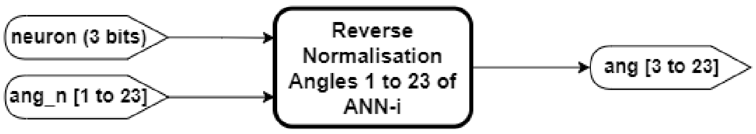


Figure 4.13: Synoptic diagram of the Reverse Normalisation module

4.2.6 Frequency Range Variator

The main feature of this block 4.14 is to change the entire operating frequency range based on the nominal frequency f_0 for the PWM signal generator block that controls the inverter; this change is managed by the two internal signal freq which is fixed in the code.

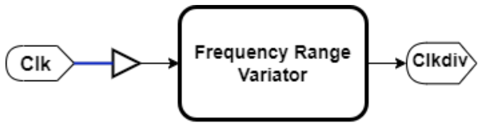


Figure 4.14: Synoptic diagram of the Regulation or Frequency Variator module

The clock used for the ANNs is 10 ns (fixed by the FPGA board), in order to control the range of variation of the frequency we, implemented a frequency divider with variable output frequency controlled by the internal signal ”freq”; The latter is used as a clock in the PWM generator to control the fundamental frequency of the PWMs signals.

This block allowed us to achieve a dependence between the frequency and the modulation index because we can always vary the frequency range as we want. By changing **im** we always have an instantaneous frequency variation due to the constant v/f law as shown in equation 2.3.

4.2.7 PWM Generator

The last step of the ANNSHE PWM algorithm is to generate the three phased PWM signals **s1**, **s2** and **s3** from the switching angles α_i generated by the reverse normalisation block.

The PWM Generator module converts these angles to times, thus we obtain the switching times, the block then uses them to generate the output signals.

The generation takes place in two Steps as shown in the figure 4.15.

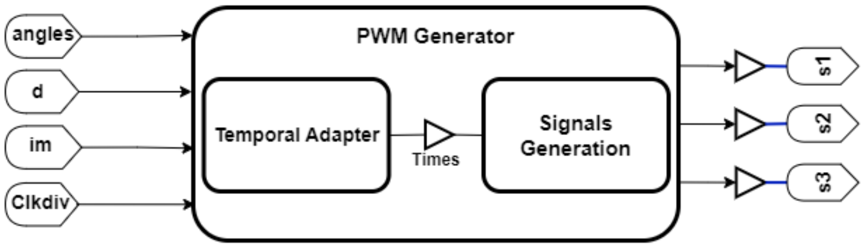


Figure 4.15: Synoptic diagram of the PWM Generator module

With the switching angles in degrees,(**s1**,**s2** and **s3**) the PWM signals, d the number of switching angles and the Clk_{div} which corresponds to the output of the frequency range variator block.

4.2.7.1 Step 1

In order to generate the PWM control signals it is necessary to transform the switching angles into switching times, so the switching angles must pass through a time adapter which allows to define the basis of the time signal by converting the switching angles α_i into switching moments t_i using the equation 2.14, and as we have used a 1MHz clock we get the following equation

$$t_i(\mu s) = \frac{10^5 \alpha_i}{18 im} \tag{4.9}$$

Hence :

$$t_i(\mu s) = \frac{\theta_i}{im} \tag{4.10}$$

With

$$\theta_i = \frac{10^5 \alpha_i}{18} \tag{4.11}$$

According to FPGA circuit. To avoid this division, the values of θ_i given by the equation are calculated in this step. Then, in the step of generating control signals, an internal signal **"counter"** in the form of a counter is created. The equation

$$im \times t_i(\mu s) = \theta_i \tag{4.12}$$

The **"counter"** represents the value of $im \times t_i(\mu s)$. It is initialized by 0 and it increments by **im** at each rising edge of the clock (1 µs) then we compare it, each time, with θ_i.

4.2.7.2 Step 2

To generate the PWM signals **s1**, **s2** and **s3** from the switching instants t_i calculated in the previous step, we use the PWM signals generator.

Initially, the signal **s1** starts at 1. As previously mentioned, a ”counter” is initialized at 0 and increments by **im**. This counter is then compared to θ_1. When the ”counter” surpasses θ_1, **s1** is toggled, and **“counter”** is compared then to θ_2 until it becomes greater then θ_2 then **s1** is toggled again and so on. This sequence repeats until the **“counter”** becomes greater than θ_d. Next, we start compareing**“counter”** to $\theta_\pi - \theta_d$ where θ_π is the value of θ that corresponds to the half-period $\alpha = \pi$. Once **“counter”** becomes greater than $\theta_\pi - \theta_d$, **s1** is toggled, and **“counter”** is compared then to $\theta_\pi - \theta_{d-1}$ untiluntil it surpasses it, then **s1** is toggled. This toggling process repeats until **“counter”** becomes greater than $\theta_\pi - \theta_1$. Then we compare it to θ_π. When **“counter”** becomes greater than θ_π, **s1** is toggled, the **“counter”** is reset to 0, and the same process is repeated again.

Based on the database used during the learning phase for all **ANN-i** networks, it was noted that $\theta_i < 60°$, we know that **s2** is phase-shifted by 120° with respect to **s1**, besides **s1**=0 after 120° and the next switching instant is at $\theta_\pi - \theta_d$. Consequently, at the beginning the signal **s2** starts at 0, its counter **”counter2”** starts at $\theta_{\frac{2\pi}{3}}$ and it’s compared to $\theta_\pi - \theta_d$ then the same process for generating **s1** is repeated. The same with **s3**, we find that it starts at 1 and its counter **“counter3”** starts at $\theta_{\frac{\pi}{3}}$ and it’s compared to $\theta_\pi - \theta_d$ then the same process for generating **s1** is repeated.

4.3 Simulation and Results

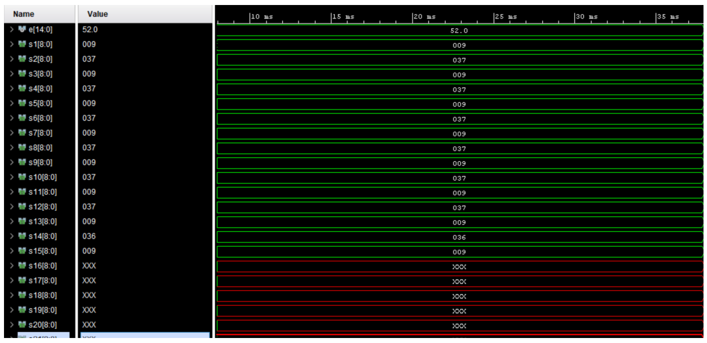
In this section, the software Vivado has been used to simulate the PWM signals generated by the algorithm ANNSHE PWM designed by the VHDL codes.

The ANNSHE PWM architecture has two inputs, which are the modulation index **im** and the clock **clk**, and three outputs representing the three PWM commands which are phase shifted by 120°

4.3.1 ANN Architecture Simulation

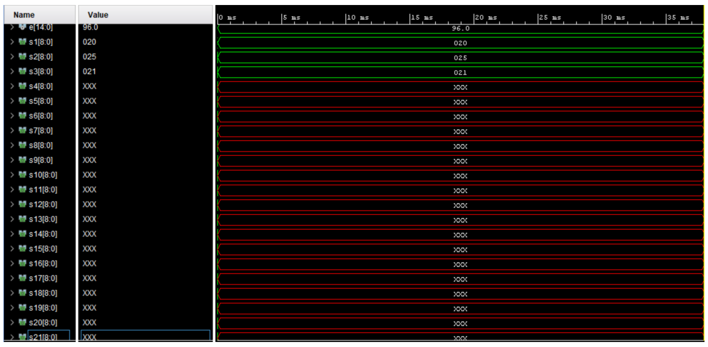
In order to evaluate the effectiveness of the implemented ANN structure, we have conducted two distinct simulations as illustrated in the accompanying Figures 4.16, 4.17. The initial simulation, characterized by an **im** value of 52%, corresponds to a **neuron** configuration of ”010”, thereby necessitating the utilization of **ANN-3**. In the second simulation, an **im** value of 96% and a **neuron** configuration of ”000”, warrants the application of **ANN-1**.

Notably in Figures 4.16, 4.17, a noticeable discrepancy arises in the allocation of output assignments between the two ANN models. Specifically, the output layer of ANN-3 for **im**=52% encompasses a total of 15 neurons, whereas ANN-1 with **im**=96% consists of only 3 neurons.



Name	Value
e[14:0]	52.0
s1[8:0]	009
s2[8:0]	037
s3[8:0]	009
s4[8:0]	037
s5[8:0]	009
s6[8:0]	037
s7[8:0]	009
s8[8:0]	037
s9[8:0]	009
s10[8:0]	037
s11[8:0]	009
s12[8:0]	037
s13[8:0]	009
s14[8:0]	036
s15[8:0]	009
s16[8:0]	XXX
s17[8:0]	XXX
s18[8:0]	XXX
s19[8:0]	XXX
s20[8:0]	XXX

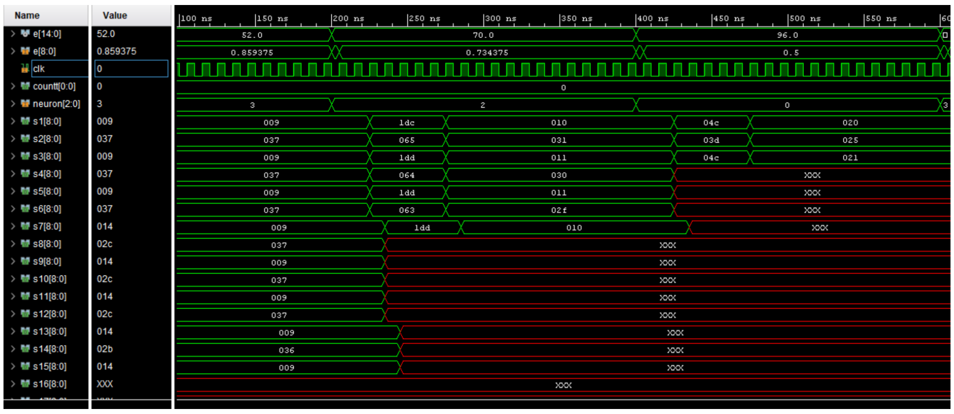
Figure 4.16: ANN for im = 52%



Name	Value
e[14:0]	96.0
s1[8:0]	020
s2[8:0]	025
s3[8:0]	021
s4[8:0]	XXX
s5[8:0]	XXX
s6[8:0]	XXX
s7[8:0]	XXX
s8[8:0]	XXX
s9[8:0]	XXX
s10[8:0]	XXX
s11[8:0]	XXX
s12[8:0]	XXX
s13[8:0]	XXX
s14[8:0]	XXX
s15[8:0]	XXX
s16[8:0]	XXX
s17[8:0]	XXX
s18[8:0]	XXX
s19[8:0]	XXX
s20[8:0]	XXX
s21[8:0]	XXX

Figure 4.17: ANN for im = 96%

To test the latency of the algorithm, we ran a simulation of the ANN architecture 4.18, generating output results for three different values of the variable **im**. Ideally, all the outputs should have their values assigned within the third cycle of the clock. However, the number of neurons in the output layers varies across different ANNs, and it significantly exceeds the number of products in the first case. (since we have employed six multipliers, with one multiplier assigned per neuron). Consequently, this discrepancy explains why it takes an additional two clock cycles to assign values to outputs that surpass six products. On the contrary, the third ANN, which possesses the largest output layer consisting of three neurons, manages to assign values to all its outputs within the third clock cycle.



Name	Value
e[14:0]	52.0
e[8:0]	0.859375
clk	0
countt[0:0]	0
neuron[2:0]	3
s1[8:0]	009
s2[8:0]	037
s3[8:0]	009
s4[8:0]	037
s5[8:0]	009
s6[8:0]	037
s7[8:0]	014
s8[8:0]	02c
s9[8:0]	014
s10[8:0]	02c
s11[8:0]	014
s12[8:0]	02c
s13[8:0]	014
s14[8:0]	02b
s15[8:0]	014
s16[8:0]	XXX

Figure 4.18: ANNs Outputs for Three Different **im** Values

4.3.2 PWM Simulation

Figures 4.19, 4.20 show under Vivado a simulation of the implementated ANNSHE PWM algorithm in the FPGA for different values of **im**.

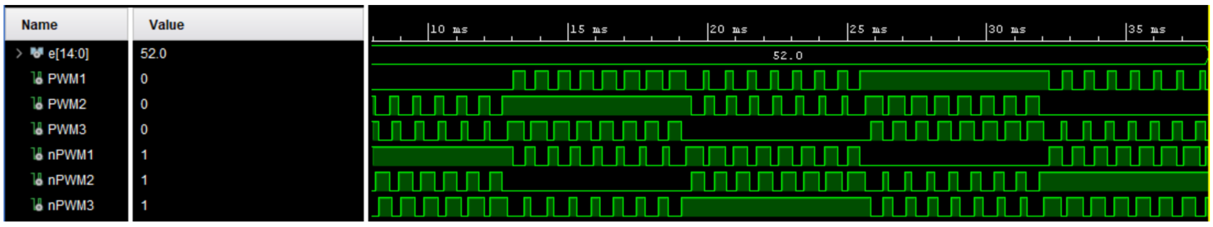


Figure 4.19: PWM for im = 52%

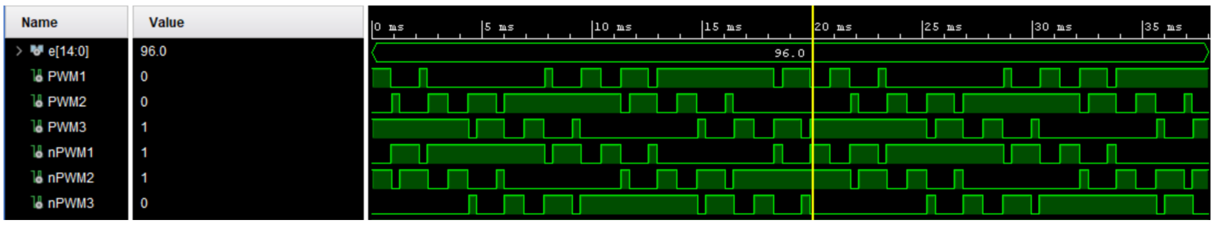


Figure 4.20: PWM for im = 96%

According to Figures 4.19, 4.20, the six signals are generated in parallel and are independent of each other with phase of 120 for the first signals, which confirms the utility of the FPGA circuit to generate a three-phase PWM signal.

The signal PWM1 from Figure 4.19 has 15 switching angles and it has a period $T = 38ms$, thus a frequency of $f = 26Hz$.

These results are confirmed by the fact that for **im** = 52%, and the selected ANN is the ANN-3 which has 15 switching angles. And from the equation 2.3 we find that the output frequency is $f = 26Hz$.

Similarly, the signal shown in Figure 4.20, also referred to as PWM1, displays 3 switching angles and possesses a period of $T = 20.8ms$, resulting in a frequency of $f = 48Hz$.

Again the results are confirmed. for **im** = 96% the selected ANN is the ANN-3 which has 3 switching angles. Using the previous equation we find that the output frequency is $f = 48Hz$.

4.4 Conclusion

In this chapter, we have seen in detail the complete architecture used to deploy the ANNSHE PWM algorithm in an FPGA board, we also have explained all the steps that are necessary for its implementation and we proved the efficiency and the intelligence of the latter. We presented the steps for implementing the different six blocks on the FPGA.

We insisted on optimising the source consumption because of its importance and its impact on the performance of our controlling system for the EPS of the EV.

Finally, in this chapter, we validated the efficiency and precision of the algorithm proposed to control the ASM for an electric propulsion system, and by simulation tests we confirm its effectiveness and accuracy in the control of the fundamental voltage.

In the next chapter, our attention will be directed towards the experimental results and practical real-world validation of the operation of the whole EPS system connected together.

Chapter 5

Experimental Results and Practical Validation

5.1 Introduction

The simulation of ANNSHE PWM motor control technique using a simulation environment serves as a critical step towards the practical validation of this control technique. While the theoretical analysis and mathematical modeling that we did in chapter 2 and chapter 3 provide valuable insights, the simulation in a virtual environment allows us to verify its effectiveness and performance before implementing it in practical applications. Furthermore, simulating ANNSHE PWM algorithm offers a safe and cost-effective means of extensive testing and validation. It eliminates the need for physical prototypes and minimizes the risks associated with testing on actual hardware.

In this chapter, our focus lies on the experimental confirmation of the proposed on-line ANNSHE PWM technique. we start with an emulation using Simulink's simulation and modeling hardware verification toolbox to do validation and progressing to experimental verification. Subsequently, we delve into the experimental results and tests conducted within the laboratory using a real world asynchronous motor with a three phase inverter of our work.

5.2 Hardware verification using FPGA In the Loop

Traditionally, digital designs are simulated using software simulation tools like **ModelSim** or **PSIM**, which model the behavior of the design based on the description written in a hardware description language (HDL) like Verilog or VHDL. While software simulation is valuable for early-stage verification, it may not capture the complete behavior of the design, especially when it interacts with external devices or real-world inputs.

FIL handles this limitation by incorporating an actual FPGA device into the simulation and verification process, allowing it to interact with actual signals and communicate with external devices, such as sensors or actuators. This enables more comprehensive testing and verification, as the FPGA can respond in real-time to inputs and produce outputs as it would in a physical implementation.

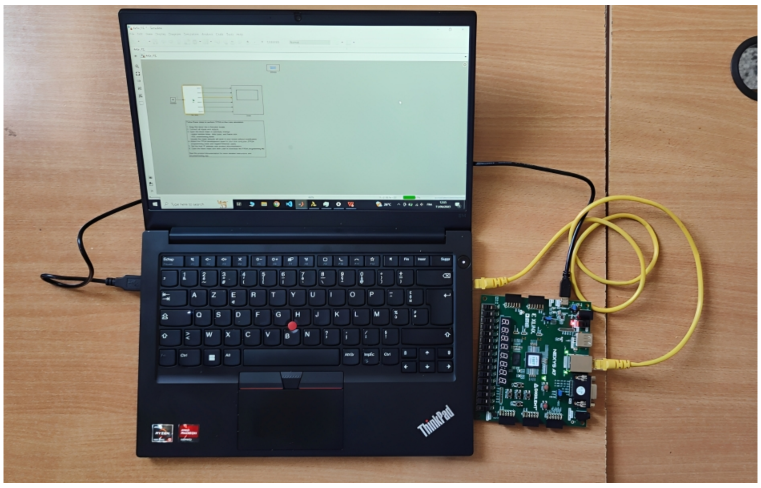


Figure 5.1: FPGA in the loop using NEXYS A7

The FPGA executes the VHDL code, and the resulting outputs are transmitted to MATLAB's Simulink environment in real-time. This communication is typically achieved through an Ethernet cable or a JTAG connector, ensuring immediate and synchronized response between the FPGA and Simulink.

5.2.1 Configuration of FIL

The steps for FIL configuration are as fellow

- **step 1**
  The process of FIL involves establishing a connection between **MATLAB** and **Vivado**. However, successful integration relies on compatibility between the versions of both software. In our case, we utilized **MATLAB 2017a** and **Vivado 2018.2**, ensuring compatibility between the two. It is essential to note that MATLAB should be installed prior to Vivado for proper functionality. Finally, we enter the following command into MATLAB's command line[1]:

  ```
  hdlsetuptoolpath('ToolName','Xilinx Vivado','ToolPath',...
  'C:\Xilinx\Vivado\2018.2\bin\vivado.bat');
  ```

- **step 2**
  After launching FIL using the command `filWizard`, the next step is to add the specifications of the FPGA board in the **Launch Board Manager** and we choose the method of connection either using Ethernet or JTAG connection.

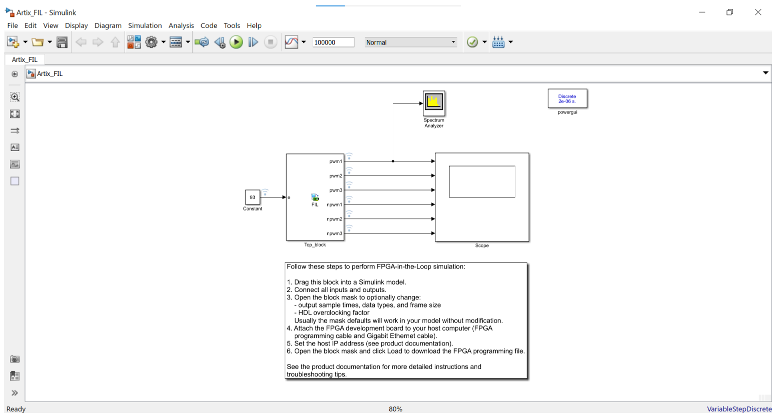
- **step 3**
  If we opt for an Ethernet connection, it is crucial to ensure that both the FPGA board and the PC are on the same network. This can be achieved by adjusting the PC's Ethernet settings(in change adapter setting, in internet protocol version 4 proprieties), specifically by configuring the subnet mask. In our case, the FPGA IP address is set to '192.168.0.2', the PC IP address is '192.168.0.3', and the subnet mask is '255.255.255.0'.

- **step 4**
  we load the VHDL code, specifying the top block of our design. Following that, we establish a connection test to ensure there are no issues and that we have correctly inputted the FPGA's specifications.

[1]The command in the command window is written in two separate line

After successfully completing the aforementioned steps, upon running the FIL, we had a FIL block in Simulink's model. At this stage, we can conveniently observe the output signals generated by the FPGA.

We shall note that in our work we used Nexys A7 FPGA and Ethernet connection for the FIL verification.



Follow these steps to perform FPGA-in-the-Loop simulation:

1. Drag this block into a Simulink model.
2. Connect all inputs and outputs.
3. Open the block mask to optionally change:
   - output sample times, data types, and frame size
   - HDL overclocking factor
   Usually the mask defaults will work in your model without modification.
4. Attach the FPGA development board to your host computer (FPGA programming cable and Gigabit Ethernet cable).
5. Set the host IP address (see product documentation).
6. Open the block mask and click Load to download the FPGA programming file.

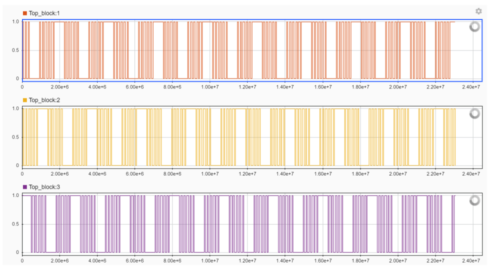
See the product documentation for more detailed instructions and troubleshooting tips.

Figure 5.2: FIL block for Nexys A7 FPGA

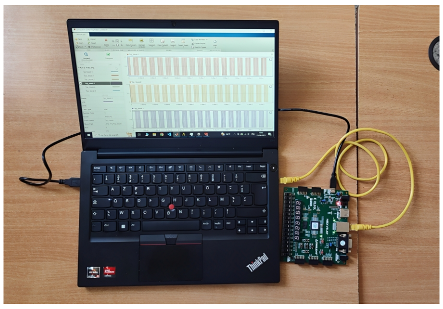
In figure 5.2, the FIL block consists of one input and six outputs. The input is referred to as the modulation index, denoted as **im**(in percent %), and the six outputs are the phased signals. Three of the outputs represent the actual signals (pwm1, pwm2 and pwm3), while the remaining three outputs are the negations (not) of those signals (not(pwm1), not(pwm2) and not(pwm3). the three signals control the upper transistors and the not signals control the lower transistors as shown in figure 5.8

5.2.2 Results of FIL's verification

After running the simulation we got the following results presented in 5.3



(a) Three phased ANNSHE PWM signals and the output of the FPGA



(b) ANNSHE PWM verification and testing using FIL

Figure 5.3: FIL verification for **im** = 76%

By comparing the output of the FIL block with the theoretical values of the instants, as well as the simulation results obtained in Vivado using logic analyzer, we can conclude that the FPGA output signals are accurate and verified.

Now, we can proceed with the experimental and practical realization of the system, as well as its validation.

5.3 Experimental validation

As we have extensively discussed throughout this work, the entire traction system consists of a command card, which is an FPGA, along with an inverter and a motor.

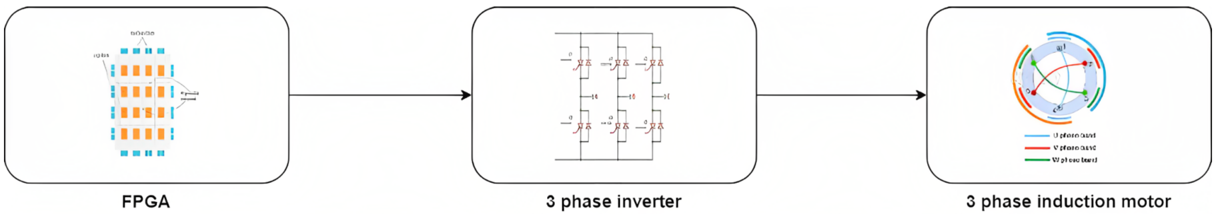


Figure 5.4: Our model for EV's traction system

Once we upload the code into the FPGA, the resulting output signals that we verified using FIL governs the switching of the inverter, thereby controlling the speed of the motor. However, in order to validate our work, it is crucial to speak about a vital component of an electric vehicle (EV), namely the three-phase inverter.

In contrast to the theoretical discussions presented in chapter 1.5, we will now construct the three phase inverter, serving as a model for a real-world EV. Subsequently, we will conduct comprehensive testing of the entire algorithm on this constructed inverter model.

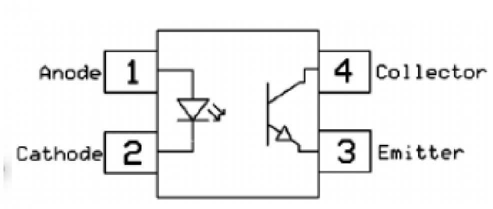
5.3.1 Three-Phase Power Inverter

The inverter is constructed using two main components: the inverter itself or the power level and its dedicated command card.

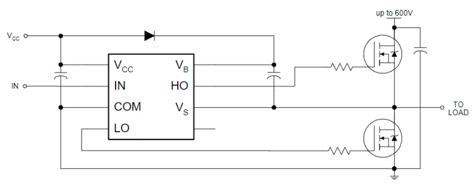
5.3.1.1 The inverter's command card

The command card serves as an interface between the FPGA output and the power level, while also acting as a protective barrier between the high-power electronics and other components. In the event of reverse current from the motor, a massive current sink, or any other issues, the card plays a crucial role in safeguarding the FPGA and other low power circuits. It consists of six components, namely three optocouplers and three half bridge drivers.

- **The optocoplers**
  The protection components on the card. it helps in isolating the grounds and duplicating the PWM signal to the driver IC.
- **The half bridge driver**
  It provides the necessary signals to control the switching of these power switches, each driver control one inverter arm (or a half bridge).



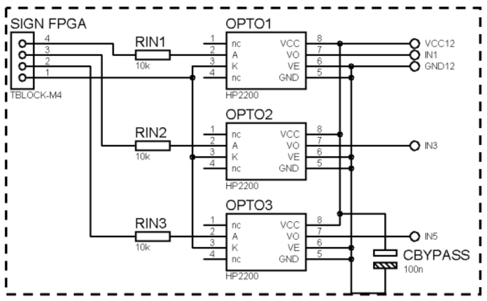
(a) Optocoupler diagram



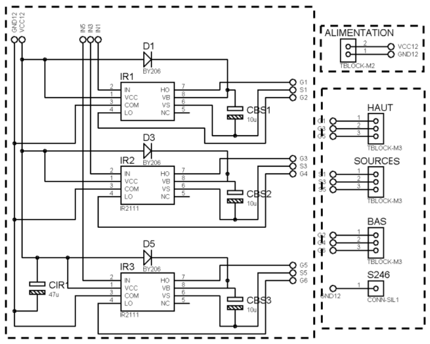
(b) IR2111 Half bridge driver

Figure 5.5: Inverter command card components

Based on the design established in [10], the card has the following design



(a) Optocoupler's isolation circuit



(b) Inverter's driver circuit

Figure 5.6: Command card circuit [10]

By connecting the two circuits mentioned in figure 5.14, we obtain the command card.

During our work in the laboratory, we encountered two old card that required maintenance. We conducted a thorough investigation to identify the problems, replaced the faulty components, and proceeded with our experimental validation.

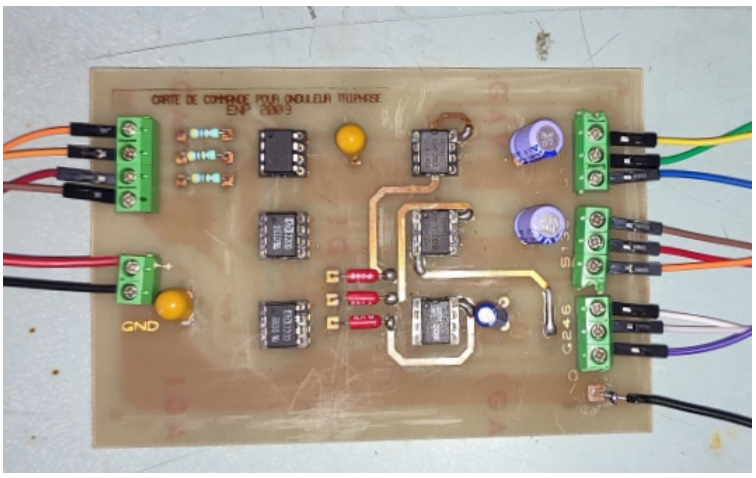
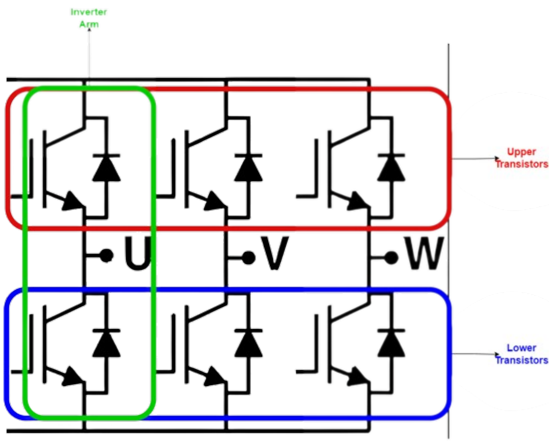


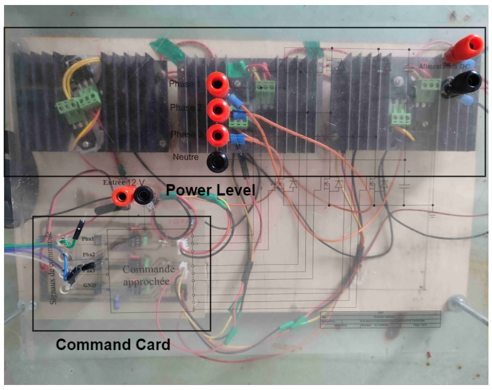
Figure 5.7: inverters command card

5.3.1.2 The power level of the inverter

Similar to the command card, we intended to use the laboratory's test inverter. However, it was also non-functional, so we had to diagnose and fix the problem. It took a significant amount of time to identify the issue, and considering our tight schedule, we decided to dismantle the inverter. We then proceeded to test the MOSFETs individually and replaced the damaged ones. Finally, we reconstructed the inverter from scratch depending on the diagram shown in figure 5.8 to ensure its proper functionality and ease of maintenance.



(a) Diagram of the three phase inverter



(b) The 3 phase inverter constructed in the laboratory

Figure 5.8: The 3 phase inverter

Referring to figure 5.8

- All upper D pins are connected to V_{cc}, the positive plate of the EV's battery.
- All lower s pins are connected to GND, the negative plate of the EV's battery.
- All upper S pins are connected to lower D pins, in the figure ..... are mentioned as U, V and W and each one of them is connected to one phase input of the ASM.
- For every inverter arm, the upper transistor's gate is connected to driver's **HO** pin.
- For every inverter arm, the lower transistor's gate is connected to driver's **LO** pin.
- For every inverter arm, the upper transistor's S is connected to driver's **VS** pin.

Once the outputs U, V, and W from the command card were connected to the motor inputs, we proceeded to modify the FPGA inputs. As anticipated, the motor responded in real time by changing its speed.

5.3.1.3 Test bench

To validate the concept of the entire system, we assembled all the components and performed speed control of an ASM. The test bench used for this purpose is depicted in figure 5.9 and includes the following components:

1. An asynchrounous motor (0.7 kW).
2. An FPGA development board to implement the ANNSHE PWM algorithm.
3. A three-phase voltage inverter to drive the ASM.
4. A DC power supply to power the voltage inverter which may represent a DC battery in an EV.
5. An oscilloscope to visualize signals.

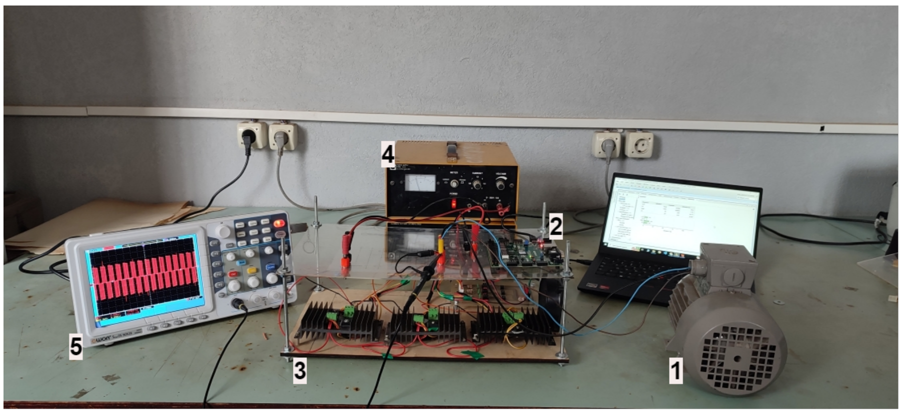
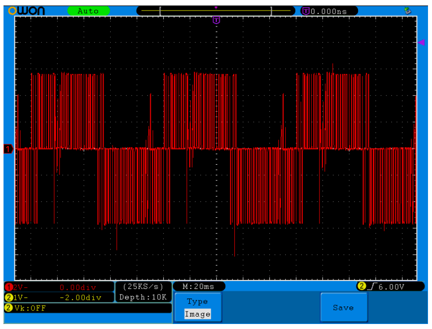


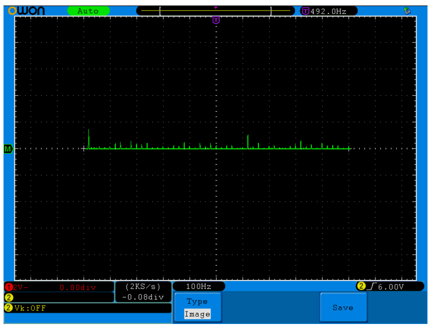
Figure 5.9: The test bench

5.3.1.4 Experimental results

After connecting all the components together and conducting tests on the online algorithm, we observed that the motor indeed changed its speed in accordance with the input value of **im**. Simultaneously, we conducted FFT, and the results revealed the elimination of harmonics, as demonstrated in the figures presented below.

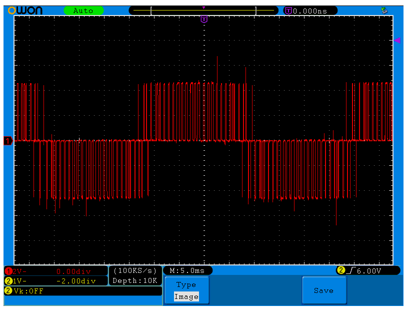


(a) Output signal

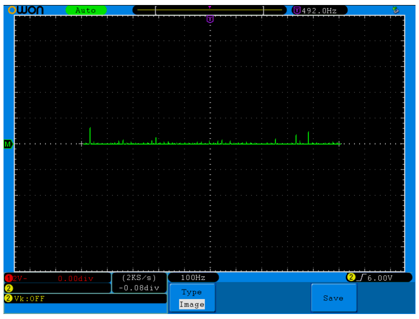


(b) FFT

Figure 5.10: ANNSHE PWM phase to phase output signal **im=20%**

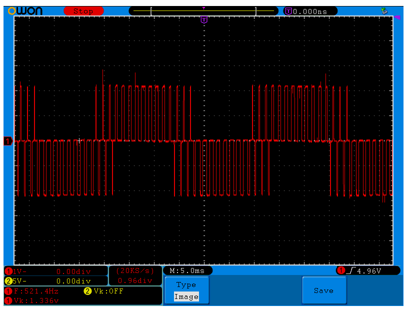


(a) Output signal

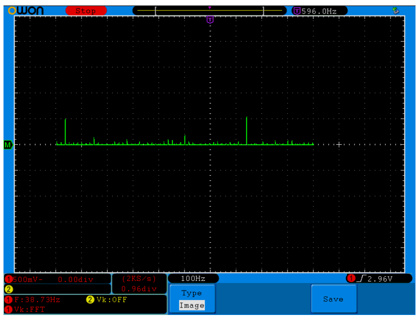


(b) FFT

Figure 5.11: ANNSHE PWM phase to phase output signal **im=48%**

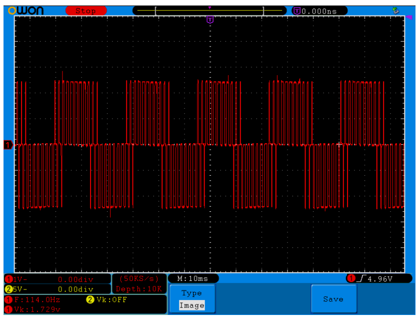


(a) Output signal

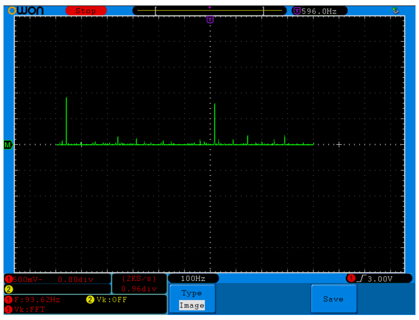


(b) FFT

Figure 5.12: ANNSHE PWM phase to phase output signal **im=64%**

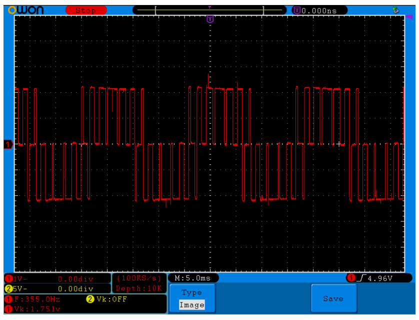


(a) Output signal

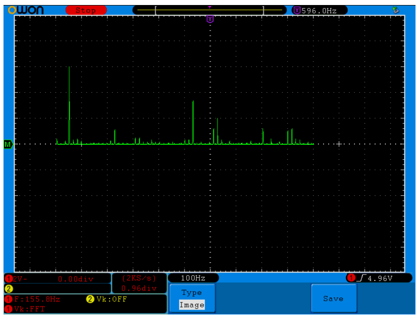


(b) FFT

Figure 5.13: ANNSHE PWM phase to phase output signal **im=80%**



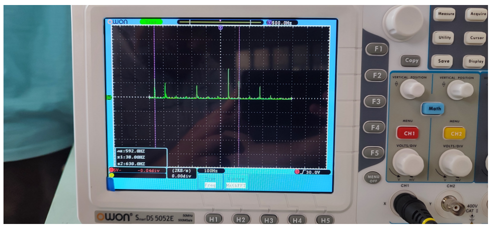
(a) Output signal



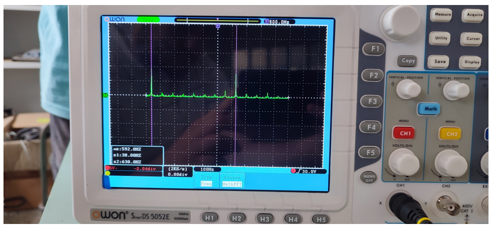
(b) FFT

Figure 5.14: ANNSHE PWM phase to phase output signal **im=96%**

To obtain final validation, we performed FFT on the signals with the motor connected and with the motor disconnected. The resultswere as follows:



(a) FFT of the inverter's output without ASM



(b) FFT of the inverter's output with ASM

Figure 5.15: FFT of ANNSHE PWM signal for **im = 76%**

By comparing the two sub figures in Figure 5.15, we can observe that, theoretically, for a value of **im** $= 76\%$, the fundamental frequency is calculated as $f = \text{im} \times f_0 = 0.76 \times 50 = 38$ Hz, which precisely matches the frequency indicated by the cursor in both sub figures.

Prior to connecting the motor, the output signal displayed the presence of harmonics multiples of three, with the first non-eliminated harmonic occurring at $f = 630$ Hz. This value is in close proximity to the theoretical value mentioned in the table 2.3 ($f_{\text{min}} = 633$ Hz), falling within the range of experimental error. After connecting the motor, we observed the elimination of all multiples of three harmonics, which validates our assumption and confirms the success of our work.

5.4 Conclusion

In this chapter, we discussed the configuration of the FIL as our chosen software tool for hardware verification. We then proceeded to verify the functionality, performance and precision of the ANNSHE PWM algorithm implemented on the NEXYS A7 FPGA board by utilizing the FIL.

Subsequently, we embarked on the experimental part of our study. We implemented the ANNSHE PWM ASM speed control algorithm on the FPGA in real time application, which served as our command circuit for the entire electric propulsion system model.

Additionally, we described the steps to reconstruction of the three-phase inverter and then we practically varied the ASM motor speed based on different **im** values. We verified the effectiveness of this approach by examining the FFT analysis results, which confirmed the successful elimination of harmonics.

Finally, we validated that all the theoretical work presented in previous chapters were translated into a successful practical implementation through our prototype model.

Conclusion and perspectives

The fabrication of an electric vehicle is a complex and resource-intensive process that requires considerable time and effort. Our focus was specifically on the core component of the propulsion system, namely the the electric motor. When designing speed control system for an EV's motor, it is crucial to take into account the expectations of drivers, which have now transformed into requirements for electric vehicles in today's market.

In our work, our primary interest revolved around the speed control of the electric motor. After thorough deliberation, we determined that utilizing a three-phase asynchronous motor and employing PWM signals to control it would be the most suitable approach.

During our work, we discovered that precise control of the switching instants of the three-phase inverter's transistors enables us to control the speed of the motor. However, this problem is mathematical described by a non linear system of equations. The vast number of possible solutions made it impractical to store all the values in memory, leading us to explore alternative approaches for real-time response and efficient implementation.

We explored two approaches in this work. The first approach, called ANNSHE PWM, involved using an Artificial Neural Network (ANN) to predict the switching angles instead of storing all the values in a database. This method proved to be effective, accurate, and had low latency and low hardware resource consumption. However, its implementation complexity was a drawback.

The second approach that we developed was called PISHE PWM, which combined image processing theory and electrical machine theory. It treated the solution set as an images to be compressed and so, we need only to store polynomial coefficients instead of the original values. This approach achieved a remarkable compression rate of 97.43%. While it was more precise and simpler to implement compared to ANNSHE PWM, it exhibited significant higher latency. Ultimately, for the application of an electric vehicle, we chose to proceed with ANNSHE PWM due to its acceptable precision for our application and low latency response.

The experimental validation of our algorithm was conducted in two stages. In the first stage, we performed FPGA-in-the-Loop (FIL) verification by implementing the proposed algorithm on a Nexys A7 FPGA. The results were then transmitted to Simulink via an Ethernet connection. Comparisons with off-line results showed significant similarity, allowing us to proceed with the test bench and conducting experimental validation.

In the second stage, we carried out the final experimental verification on a test bench. The test bench consisted of an ASM powered by a three-phase inverter that we reconstructed from an old one used in the laboratory. The control of the inverter was based on the proposed online PWM technique. We measured and evaluated the output signals of the inverter during the experiment, and a FFT was performed to confirm the elimination of harmonics. The obtained results closely matched those predicted by theory and the FIL verification.

Throughout the entire speed range, we successfully eliminated low-order harmonics, achieved desirable switching frequencies, and effectively controlled the motor's speed. For future work, there are several perspectives to consider within the framework of SHE PWM methods and electric vehicle (EV) engineering.

Regarding the PISHE PWM approach, further research and development are necessary to enhance its capabilities and performance in real-time applications. Exploring different image processing techniques and algorithms could lead to improvements in terms of precision, latency, and hardware resource consumption. Continuously exploring and refining this method has the potential to yield even better results.

In the context of EVs, an upscale study is needed to adapt the proposed algorithm and techniques for actual automobile fabrication. This involves integrating the inverter and power blocks into a comprehensive system that meets the requirements and standards of modern electric vehicle technology. This would entail considering various aspects such as safety, efficiency, reliability, and cost-effectiveness.

Additionally, future work could involve embedding other essential systems within the EV propulsion system. This may include battery management systems, regenerative braking mechanisms, advanced control algorithms for improved energy efficiency, and intelligent monitoring and diagnostic systems for enhanced reliability and maintenance.

Continuing research in SHE PWM methods and EV propulsion system engineering holds great promise for further advancements in electric vehicle technology. The aim is to develop more efficient and reliable systems that contribute to the widespread adoption and sustainability of electric transportation.

Bibliographie

2. DMSBS, Per Enge Ph; MSBS, Nick Enge; DMSBS, Stephen Zoepf Ph. *Electric Vehicle Engineering.* McGraw-Hill Education, 2021, pp. 10-60.
3. KIVILUOMA, Juha; MEIBOM, Peter. Methodology for modelling plug-in electric vehicles in the power system and cost estimates for a system with either smart or dumb electric vehicles. *Energy.* 2011, vol. 36, no. 3, pp. 1758–1767.
4. DU, Zhong; OZPINECI, Burak; TOLBERT, Leon M; CHIASSON, John N. DC–AC cascaded H-bridge multilevel boost inverter with no inductors for electric/hybrid electric vehicle applications. *IEEE Transactions on Industry Applications.* 2009, vol. 45, no. 3, pp. 963–970.
5. EMADI, Ali; WILLIAMSON, Sheldon S; KHALIGH, Alireza. Power electronics intensive solutions for advanced electric, hybrid electric, and fuel cell vehicular power systems. *IEEE Transactions on power electronics.* 2006, vol. 21, no. 3, pp. 567–577.
6. CHAN, CC; CHAU, KT, et al. *Modern electric vehicle technology.* Vol. 47. Oxford University Press on Demand, 2001, pp. 16-31.
7. XUE, XD; CHENG, Ka Wai Eric; CHEUNG, NC. Selection of electric motor drives for electric vehicles. In: *2008 Australasian Universities power engineering conference.* IEEE, 2008, pp. 1–6.
8. ZERAOULIA, Mounir; BENBOUZID, Mohamed El Hachemi; DIALLO, Demba. Electric motor drive selection issues for HEV propulsion systems: A comparative study. *IEEE Transactions on Vehicular technology.* 2006, vol. 55, no. 6, pp. 1756–1764.
9. KHIDER, Moussa. Commande de vitesse en temps réel d'un moteur asynchrone triphasé. In: *mémoire de Magister 2003, Ecole Nationale Polytechnique, Algiers, Algeria.* ENP, 2003.
10. BENDIB, Douadi. Etude et réalisation d'une commande MLI on-line sur circuit FPGA. In: *mémoire de Magister 2009, Ecole Nationale Polytechnique, Algiers, Algeria.* ENP, 2009.
11. GUELLAL, Amar. Implémentation sur FPGA d'une commande MLI on-line basée sur le principe des réseaux de neurones. In: *mémoire de Magister 2010, Ecole Nationale Polytechnique, Algiers, Algeria.* ENP, 2010.
12. PATEL, Hasmukh S; HOFT, Richard G. Generalized techniques of harmonic elimination and voltage control in thyristor inverters: Part I–Harmonic Elimination. *IEEE Transactions on Industry Applications.* 1973, no. 3, pp. 310–317.

13. PATEL, Hasmukh S; HOFT, Richard G. Generalized techniques of harmonic elimination and voltage control in thyristor inverters: part II—voltage control techniques. *IEEE Transactions on Industry Applications*. 1974, no. 5, pp. 666–673.

14. GUELLAL, Amar. Contribution à l'étude et à l'implémentation des commandes en temps réel pour MAS. In: *thèse de doctorat 2015, Ecole Nationale Polytechnique, Algiers, Algeria*. ENP, 2015.

15. CHAU, Kwok Tong. *Electric vehicle machines and drives: design, analysis and application*. John Wiley & Sons, 2015, pp. 6-16.

16. LEITMAN, Seth; BRANT, Bob. *Build your own electric vehicle*. McGraw-Hill, Inc., 2008, pp. 3-10.

17. CHAN, CC; WONG, YS. The state of the art of electric vehicles technology. In: *The 4th International Power Electronics and Motion Control Conference, 2004. IPEMC 2004*. IEEE, 2004, vol. 1, pp. 46–57.

18. LÓPEZ, I; IBARRA, E; MATALLANA, A; ANDREU, J; KORTABARRIA, I. Next generation electric drives for HEV/EV propulsion systems: Technology, trends and challenges. *Renewable and Sustainable Energy Reviews*. 2019, vol. 114, p. 109336.

19. LARMINIE, James; LOWRY, John. *Electric vehicle technology explained*. John Wiley & Sons, 2012, pp. 2-25.

20. DHAMEJA, Sandeep. *Electric vehicle battery systems*. Elsevier, 2001, pp. 1-21.

21. BAGHLI, L. Modélisation et commande de la machine asynchrone. In: IUFM de Lorraine - UHP, 2005.

22. BENDIB, Douadi. Contribution à l'étude et à l'implémentation sur circuits FPGA de la commande MLI à élimination sélective des harmoniques temps réel. In: *thèse de doctorat 2017, Ecole Nationale Polytechnique, Algiers, Algeria*. ENP, 2017.

23. HOULDSWORTH, John A; GRANT, Duncan A. The use of harmonic distortion to increase the output voltage of a three-phase PWM inverter. *IEEE Transactions on industry applications*. 1984, no. 5, pp. 1224–1228.

24. TAUFIQ, JA; MELLITT, B; GOODMAN, CJ. Novel algorithm for generating near optimal PWM waveforms for AC traction drives. In: *IEE Proceedings B (Electric Power Applications)*. IET, 1986, vol. 133, pp. 85–94. No. 2.

25. BOURENANE, Aomar. Générateur intelligent de multi-réseaux neuronaux artificiels Application : La commande SHE PWM pour le contrôle de vitesse des moteurs Asynchrones. In: *Projet Fin d'Etude 2018, Ecole Nationale Polytechnique, Algiers, Algeria*. ENP, 2018.

26. SAADI, Khalid; OUADRIA, Anes Abderrahim. *Implementation of artificial neural on an FPGA board application on induction motor speed control*. Ecole Nationale Polytechnique, 2017. PhD thesis.

27. KROSE, Ben; SMAGT, P. An introduction to neural networks, University of Ámsterdam. *Amesterdam, Netherland*. 1996, p. 29.

28. BENDIB, Douadi; LARBES, Cherif; GUELLAL, Ammar; KHIDER, Moussa; AKEL, Fethi. FPGA-based implementation of online selective harmonic elimination PWM for voltage source inverter. *International Journal of Electronics*. 2017, vol. 104, no. 10, pp. 1715–1731.

29. HIMAVATHI, S.; ANITHA, D.; MUTHURAMALINGAM, A. Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization. *IEEE Transactions on Neural Networks*. 2007, vol. 18, no. 3, pp. 880–888. Available from DOI: 10.1109/TNN.2007.891626.

30. OMONDI, Amos R; RAJAPAKSE, Jagath Chandana. *FPGA implementations of neural networks*. Vol. 365. Springer, 2006.

31. MEHER, Pramod Kumar. An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks. In: *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*. IEEE, 2010, pp. 91–95.

Webographie

1. CLEAN TRANSPORTATION (ICCT), International Council on. *Growing Momentum: Global Overview of Government Targets for Phasing Out Sales of New Internal Combustion Engine Vehicles* [https://theicct.org/growing-momentum-global-overview-of-government-targets-for-phasing-out-sales-of-new-internal-combustion-engine-vehicles/]. 2021, Accessed 10/06/2023.