



Ecole Nationale Polytechnique d'Alger  
Département d'Automatique



## End of studies' project

Submitted in partial fulfillment of the requirements for the State  
Engineer Degree in Automatics & Control Engineering

---

### Quantum PWM design and implmentation for DC motor Drive

---

#### Supervised by

- Pr Nadjjet ZIOUI
- Pr Mohamed TADJINE

#### Realized by

- Sohaib SAIDAT
- Rami BOUMEKHITA

Presented the 10 July 2023, in front of the jury :

Pr Omar STIHI	- President
Pr Nadjjet ZIOUI	- Promoter
Pr Mohamed TADJINE	- Promoter
Dr Hakim ACHOUR	- Examiner
Dr Reda DERMOUCHE	- Guest





Ecole Nationale Polytechnique d'Alger  
Département d'Automatique



## End of studies' project

Submitted in partial fulfillment of the requirements for the State  
Engineer Degree in Automatics & Control Engineering

---

### Quantum PWM design and implmentation for DC motor Drive

---

#### Supervised by

- Pr Nadjjet ZIOUI
- Pr Mohamed TADJINE

#### Realized by

- Sohaib SAIDAT
- Rami BOUMEKHITA

Presented the 10 July 2023, in front of the jury :

Pr Omar STIHI	- President
Pr Nadjjet ZIOUI	- Promoter
Pr Mohamed TADJINE	- Promoter
Dr Hakim ACHOUR	- Examiner
Dr Reda DERMOUCHE	- Guest



Ecole Nationale Polytechnique d'Alger  
Département d'Automatique



---

# Mémoire de projet de fin d'études

pour l'obtention du diplôme d'Ingénieur d'État en Automatique

---

## Conception d'un MLI quantique et implémentation pour un moteur à courant continu

---

Supervisé par

- Pr Nadjjet ZIOUI
- Pr Mohamed TADJINE

Réalisé par

- Sohaib SAIDAT
- Rami BOUMEKHITA

Présenté le 10 juillet 2023, devant le jury :

Pr Omar STIHI	- Président
Pr Nadjjet ZIOUI	- Promoteur
Pr Mohamed TADJINE	- Promoteur
Dr Hakim ACHOUR	- Examineur
Dr Reda DERMOUCHE	- Invité

# *Acknowledgements*

First of all, we thank **GOD** Almighty **Allah** for giving us the courage and patience for developing this modest work.

We would also like to extend our deepest appreciation to **Our Parents**, whose unconditional love, encouragement, and sacrifices have made our education and this project possible. Their unwavering belief in us has been a constant motivation.

Our deepest thanks go to our supervisors, Professor **Nadjet ZIOUI** and Professor **Mohamed TADJINE**. Their expertise, guidance, and mentorship have been invaluable throughout the entire project. Their commitment to excellence and their willingness to share their knowledge have inspired us and shaped our academic growth. We are truly fortunate to have had the opportunity to work under their guidance.

Finally, we would like to thank all the individuals who directly or indirectly contributed to the successful completion of our project. Their contributions, whether big or small, have played a significant role in shaping our understanding and achievements.

# ملخص

في السنوات الأخيرة، ظهرت الحوسبة الكمية كنموذج واعد يقدم قوة حسابية غير مسبوقه لحل المشاكل المعقدة. تركز هذه الدراسة على مجال الحوسبة الكمية وتكشف إمكاناتها في توليد إشارات تعديل عرض النبض (ت ع ن)، وهي تقنية حاسمة تستخدم على نطاق واسع في تطبيقات متنوعة مثل الإلكترونيات القوية وأنظمة الاتصالات.

يهدف هذا المشروع الختامي للدراسات إلى استكشاف الخوارزميات الكمية وملاءمتها لتوليد إشارات ت ع ن من خلال تحليل كافي لأطر الحوسبة الكمية الموجودة، بما في ذلك تلاعب القواعد الكمية وعمليات البوابات الكمية، نهدف إلى تطوير نهج جديد لتوليد إشارات ت ع ن باستخدام الخصائص الفريدة للأنظمة الكمية

تشمل منهجية المشروع تنفيذ الخوارزميات الكمية، بما في ذلك البوابات الكمية، لتحقيق الخصائص المطلوبة لإشارات ت ع ن.

جوهر ونواة هذا العمل تعتمد على تطوير تصميم جديد وتنفيذ لمقارنة الكم الكومبي التي يمكنها مقارنة رقمين حقيقيين، وهو عنصر حاسم لتحقيق حل تعديل عرض النبض الكومبي. المقارنات الكومبية الموجودة حاليًا في الأدبيات تتعامل فقط مع القيم الثنائية ولا يمكنها التعامل مع حالات الكم المتراكبة التي تمثل قيم احتمالية غير 0 أو 100%. من خلال تحليل مفصل، تمكنا من تقسيم الحل المقترح إلى جزئين، يتعامل كل منهما مع فترات احتمالية مكتملة. من خلال دمج هذين الدائرتين الكومبيتين، نحقق إدارة شاملة لفترات الأعداد الحقيقية الموجودة بين 0 و 100%. بالإضافة إلى ذلك، نقارن أداء النهج القائم على الحوسبة الكمية في توليد إشارات بوم مع الطرق الكلاسيكية التقليدية التي تُطبق في حلقة مغلقة للتحكم في سرعة محرك التيار المستمر من حيث جودة وأداء الإشارة والاستجابة للمرجع ومختلف المزايا الكمية المحتملة.

---

**كلمات مفتاحية:** الحساب الكومبي، تعديل عرض النبض، المقارن، الإحتمالية.

---

# Résumé

Ces dernières années, l'informatique quantique est apparue comme un paradigme prometteur offrant une puissance de calcul inégalée pour résoudre des problèmes complexes. Cette étude plonge dans le domaine de l'informatique quantique et explore son potentiel pour générer des signaux de modulation de largeur d'impulsion (MLI), une technique cruciale largement utilisée dans diverses applications telles que l'électronique de puissance et les systèmes de communication.

L'objectif de ce projet de fin d'études est d'explorer les algorithmes quantiques et leur adaptabilité à la génération de signaux MLI. À travers une analyse suffisante des cadres existants en informatique quantique, incluant la manipulation de qubits et les opérations de portes quantiques, nous visons à développer une approche novatrice pour générer des signaux MLI en exploitant les propriétés uniques des systèmes quantiques.

La méthodologie du projet implique la mise en œuvre d'algorithmes quantiques, y compris les portes quantiques, afin d'obtenir les caractéristiques souhaitées des signaux MLI.

L'essence et le cœur de ce travail reposent sur le développement d'une conception et d'une mise en œuvre novatrices d'un comparateur quantique capable de comparer deux nombres réels, un composant essentiel pour atteindre une solution de modulation d'impulsions en largeur (MLI) quantique. Les comparateurs quantiques existants décrits dans la littérature ne gèrent que des valeurs binaires et ne peuvent pas traiter les états quantiques superposés qui représentent des valeurs probabilistes autres que 0 et 100%. Grâce à une analyse détaillée, nous avons pu diviser notre solution proposée en deux parties, chacune gérant des intervalles probabilistes complémentaires. En fusionnant ces deux circuits quantiques, nous parvenons à une gestion complète des intervalles de nombres réels compris entre 0 et 100%. De plus, nous comparons les performances de l'approche de génération de signaux MLI basée sur l'informatique quantique avec les méthodes classiques conventionnelles appliquées dans une boucle fermée pour commander la vitesse d'un moteur à courant continu en termes de qualité et des performances du signal, aussi la comportement de la réponse à une certaine référence et d'autres avantages quantiques potentiels.

---

**Mots Clés:** Calcul quantique, MLI, Comparateur, Probabilité

---

# *Abstract*

Recently, quantum computing has emerged as a promising paradigm that offers unprecedented computational power for solving complex problems. This study delves into the domain of quantum computing and investigates its potential for generating Pulse Width Modulation (PWM) signals, a crucial technique widely used in various applications such as power electronics and communication systems.

The objective of this end-of-studies project is to explore quantum algorithms and their suitability for Pulse Width Modulation signal generation. Through an enough analysis of existing quantum computing frameworks, including qubit manipulation and quantum gate operations, we aim to develop a novel approach for generating PWM signals utilizing the unique properties of quantum systems.

The project methodology involves the implementation of quantum algorithms, including quantum gates, to achieve the desired PWM signal characteristics.

The essence and core of this work is based on the development of a novel design and implementation of a quantum comparator that can compare two real numbers, a crucial component for achieving a quantum Pulse Width Modulation (PWM) solution. Existing quantum comparators described in the literature only handle binary values and cannot handle superposed quantum states representing probabilistic values other than 0 et 100%. Through a detailed analysis, we were able to divide our proposed solution into two parts, each handling complementary probabilistic intervals. By merging these two quantum circuits, we achieve comprehensive management of real number intervals ranging from 0 to 100%.

Additionally, we compare the performance of the quantum-based PWM generation approach with conventional classical methods applied in a closed loop for controlling the speed of a DC motor in terms of signal quality and performance, response to the reference, and potential quantum advantages.

---

**Keywords:** Quantum computing, PWM, Comparator, Probability.

---



# Contents

List of Figures

List of Tables

<b>General Introduction</b>	<b>14</b>
<b>1 From Classical to Quantum Computing</b>	<b>16</b>
1.1 CLASSICAL COMPUTATION . . . . .	17
1.1.1 Introduction & History about classical logic . . . . .	17
1.1.2 Logic . . . . .	18
1.1.3 Bits . . . . .	18
1.1.4 Binary . . . . .	19
1.1.5 Logic Gates . . . . .	19
1.2 BASICS OF QUANTUM COMPUTING . . . . .	25
1.2.1 Introduction and overview . . . . .	25
1.2.2 History of quantum computation and quantum information . . . . .	25
1.2.3 Single Quantum bit system . . . . .	26
1.2.4 Multiple Quantum Bits . . . . .	32
1.3 Quantum Programming . . . . .	36
1.3.1 IBM Quantum . . . . .	37
1.3.2 Quirk toolkit . . . . .	43
<b>2 Speed Control of DC Motors using PWM</b>	<b>45</b>
2.1 INTRODUCTION TO SPEED CONTROL . . . . .	45
2.2 CLASSIFICATION OF DC MOTORS . . . . .	45
2.3 SPEED CONTROL METHODS . . . . .	46
2.3.1 FLUX CONTROL METHOD . . . . .	46
2.3.2 Armature or rheostat control method . . . . .	48

2.3.3	Rheostat control method . . . . .	48
2.3.4	VOLTAGE CONTROL METHOD . . . . .	49
2.3.5	PWM TECHNIQUE . . . . .	50
2.4	PWM TECHNIQUE . . . . .	50
2.5	DC Motor speed control using PWM method . . . . .	52
2.5.1	Methods for Generating PWM Signals . . . . .	53
2.6	Control Arrangements for D.C. Drives . . . . .	54
2.6.1	Current control . . . . .	56
2.6.2	Torque control . . . . .	58
2.6.3	Speed control . . . . .	59
2.7	Simulation of PWM Techniques for DC Motor Speed Control . . . . .	61
2.7.1	Controlled PWM Voltage . . . . .	62
2.7.2	H-bridge Drive for a DC motor . . . . .	63
2.7.3	Simulation results . . . . .	64
<b>3</b>	<b>Quantum PWM Algorithm</b>	<b>67</b>
3.1	The idea of a quantum comparator: . . . . .	68
3.1.1	Greater or equal $\geq$ comparator: . . . . .	69
3.1.2	The equivalent quantum comparator: . . . . .	69
3.1.3	Operating principle: . . . . .	70
3.1.4	Completely greater $>$ comparator: . . . . .	74
3.1.5	The equivalent quantum comparator: . . . . .	75
3.1.6	Operating principle: . . . . .	75
3.2	Quantum Circuit Programming in MATLAB Simulink: . . . . .	78
3.2.1	The $>$ comparator: . . . . .	79
3.2.2	The $\geq$ comparator: . . . . .	81
3.2.3	The Inputs: . . . . .	81
3.2.4	The Outputs: . . . . .	84
3.2.5	The relation between the rotation angle $\theta_{q_1}$ and the duty cycle $D$ : . . . . .	85
<b>4</b>	<b>Comparative Study between Quantum PWM and Classical PWM</b>	<b>94</b>
4.1	Introduction . . . . .	94
4.2	Comparing PWM Signals in Closed-Loop Control Systems: Quantum Com- parator PWM vs Classic Comparator PWM . . . . .	96
4.2.1	PWM Signals comparison results . . . . .	99

4.3	Comparing Speed Signals in Closed-Loop Control Systems: Quantum Com- parator PWM vs Classic Comparator PWM . . . . .	99
4.3.1	Speed comparison results . . . . .	102
4.4	Comparing error between reference and speed signals in closed-loop control systems: Quantum comparator PWM vs Classic comparator PWM . . . . .	102
4.4.1	Errors comparison results . . . . .	104
4.5	Staircase Reference Testing in Simulink: Evaluating Control System Per- formance . . . . .	104
4.6	Results and analysis . . . . .	106
	<b>General Conclusion</b>	<b>108</b>
	<b>Bibliography</b>	<b>110</b>

# List of Figures

- 1.1 Functional bloc of a classic single bit gate. . . . . 20
- 1.2 The Identity gate . . . . . 20
- 1.3 The NOT gate . . . . . 20
- 1.4 Functional bloc of a classic two-bit gate . . . . . 21
- 1.5 The AND gate . . . . . 21
- 1.6 The OR gate . . . . . 22
- 1.7 The XOR gate . . . . . 23
- 1.8 The NAND gate . . . . . 23
- 1.9 The NOR Gate . . . . . 24
- 1.10 State of a qubit on the Bloch sphere. . . . . 29
- 1.11 Quantum circuit that contains a CNOT gate . . . . . 36
- 1.12 Dashboard of IBM quantum, image generated from[16] . . . . . 37
- 1.13 IBM quantum processors . . . . . 38
- 1.14 Ibmq manila processor . . . . . 38
- 1.15 IBM quantum Composer . . . . . 39
- 1.16 GHZ state circuit . . . . . 39
- 1.17 GHZ state circuit using composer . . . . . 40
- 1.18 Teleportation circuit . . . . . 41
- 1.19 Teleportation circuit . . . . . 42
- 1.20 Teleportation algorithm. . . . . 44
  
- 2.1 The three types of DC motors . . . . . 46
- 2.2 FLUX CONTROL METHOD . . . . . 47
- 2.3 Rheostat control method . . . . . 48
- 2.4 System structural arrangement of WARD LEONARD . . . . . 49
- 2.5 5V Pulses with 0% through 100% duty cycle . . . . . 51
- 2.6 Simple speed controller . . . . . 52

2.7	Schematic diagram of analogue controlled-speed drive with current and speed feedback control loops . . . . .	55
2.8	Detail showing characteristic of speed error amplifier . . . . .	57
2.9	Simulink diagram of DC motor controlled by PWM . . . . .	62
2.10	Variation of DC Motor Speed across Varying Duty Cycles . . . . .	65
3.1	Up counter . . . . .	67
3.2	Down counter . . . . .	68
3.3	Up-down counter . . . . .	68
3.4	The classic logic circuit of $\geq$ comparator . . . . .	69
3.5	The quantum logic circuit of $\geq$ comparator . . . . .	70
3.6	The four case of comparison of the $\geq$ comparator . . . . .	70
3.7	The quantum circuit to compare between two qubits in superpostion . . . . .	71
3.8	The quantum logic circuit of $\geq$ comparator . . . . .	73
3.9	The classic logic circuit of $>$ comparator . . . . .	75
3.10	The quantum logic circuit of $>$ comparator . . . . .	75
3.11	The four case of comparison of the $>$ comparator . . . . .	76
3.12	The quantum logic circuit of $>$ comparator . . . . .	77
3.13	The quantum logic circuit of the $>$ comparator . . . . .	79
3.14	The inputs of the quantum comparator $>$ . . . . .	81
3.15	The inputs of the quantum comparator $\geq$ . . . . .	82
3.16	The probability $P_{s-s-q_3}( 1\rangle)$ before and after measurement . . . . .	86
3.17	The probability $P_{s-q_3}( 1\rangle)$ before and after measurement . . . . .	88
3.18	The QPWM algorithme . . . . .	89
3.19	Ramp bloc . . . . .	90
3.20	<i>Duty cycle</i> $D = 0\%$ . . . . .	91
3.21	<i>Duty cycle</i> $D = 25\%$ . . . . .	91
3.22	<i>Duty cycle</i> $D = 50\%$ . . . . .	92
3.23	<i>Duty cycle</i> $D = 75\%$ . . . . .	92
3.24	<i>Duty cycle</i> $D = 100\%$ . . . . .	93
4.1	Speed control circuits . . . . .	95
4.2	PWM Signals generated by quantum comparator technique in closed loop for a reference of 5 rad/s . . . . .	96

4.3	PWM Signals generated by classical comparator technique in closed loop for a reference of 5 rad/s . . . . .	97
4.4	PWM Signals generated by quantum comparator technique in closed loop for a reference of 10 rad/s . . . . .	97
4.5	PWM Signals generated by classical comparator technique in closed loop for a reference of 10 rad/s . . . . .	98
4.6	PWM Signals generated by quantum comparator technique in closed loop for a reference of 20 rad/s . . . . .	98
4.7	PWM Signals generated by classical comparator technique in closed loop for a reference of 20 rad/s . . . . .	99
4.8	Comparison of speed signals for a reference of 5 rad/s . . . . .	100
4.9	Comparison of speed signals for a reference of 10 rad/s . . . . .	101
4.10	Comparison of speed signals for a reference of 20 rad/s . . . . .	101
4.11	Comparison of error between reference and speed signals for a reference of 5 rad/s . . . . .	102
4.12	Comparison of error between reference and speed signals for a reference of 10 rad/s . . . . .	103
4.13	Comparison of error between reference and speed signals for a reference of 20 rad/s . . . . .	103
4.14	Comparison of speed signals for a staircase reference . . . . .	105

# List of Tables

- 1.1 Truth table of the identity gate . . . . . 20
- 1.2 The truth table of the NOT gate . . . . . 20
- 1.3 The truth table of two-bit gate . . . . . 21
- 1.4 The truth table of the AND gate . . . . . 22
- 1.5 The truth table of the OR gate . . . . . 22
- 1.6 The truth table of the XOR gate . . . . . 23
- 1.7 The truth table of the NAND gate . . . . . 23
- 1.8 The truth table of the NOR gate . . . . . 24
  
- 2.1 Variation of DC motor speed across carrying cuty cycles . . . . . 64
  
- 3.1 Truth table of the the  $\geq$  comparator . . . . . 69
- 3.2 The table of probabilities of the example in figure 3.7 . . . . . 72
- 3.3 Table of probabilities of the  $\geq$  comparator . . . . . 73
- 3.4 Table of duty cycle for the  $\geq$  comparator . . . . . 74
- 3.5 Truth table of the the  $>$  comparator . . . . . 74
- 3.6 Table of probabilities of the  $\geq$  comparator . . . . . 77
- 3.7 Table of duty cycle for the  $>$  comparator . . . . . 77

# General Introduction

In recent years, there has been a remarkable surge of interest and research in the field of quantum computing, as scientists and engineers explore the potential of harnessing the laws of quantum mechanics to revolutionize information processing. This exciting area of study has paved the way for novel applications and computational paradigms that were once thought to be beyond the world of classical computing. One such area where quantum computing shows promise is in the field of power electronics, specifically in the design and implementation of Pulse Width Modulation (PWM) techniques for controlling Direct Current (DC) motors.

The purpose of this report is to present our end-of-studies project, titled "Quantum PWM Design and Implementation for DC Motor Drive." This project aims to bridge the gap between classical and quantum computing by exploring the application of quantum computing principles in PWM-based speed control of DC motors. Our research delves into the development of a new PWM technique, known as Quantum PWM (QPWM), which leverages the power of quantum computing to enable more efficient and precise control of DC motor speeds. Inspired primarily by the classical PWM generator technique used to control power transfer between electrical components by rapidly switching between full power transfer and no power transfer that based on comparing two signals [19]: the first being a reference signal that represents the duty cycle, and the second being a variable signal that represents the carrier counter value, we propose a novel algorithm called QPWM or Quantum PWM by harnessing the principles of quantum computing like superposition and entanglement.

The report is organized into four chapters, each focusing on key aspects of our project. The first chapter, "From Classical to Quantum Computing," provides a comprehensive overview of the fundamental principles and key concepts underlying both classical and quantum computing. We explore the key differences between classical bits and quantum bits (qubits) and highlight the unique properties of quantum superposition and entanglement that form the basis of quantum computing.



---

The second chapter, "Speed Control of DC Motors Using PWM Technique," examines the conventional PWM technique widely employed in power electronics for DC motor control. We discuss the principles of PWM and its application in regulating the speed of DC motors, highlighting its advantages and limitations.

The third chapter, "Quantum PWM Algorithm," represents a crucial phase of our project where we introduce our novel QPWM technique. By leveraging the power of quantum computing, we have successfully developed two quantum comparators with their compound we will be able to compare between two qubits in superposition states, essentially enabling the comparison of two real numbers, where the first represent the duty cycle and the second represent the equivalent of the variable signal of the classic PWM (carrier counter) then we get the result in the third qubit that represent the result of comparison on which we make the measurement notion to achieve the necessary switching between the two states ( $|0\rangle$  and  $|1\rangle$ ), and this breakthrough allows us to generate the QPWM signal.

Finally, the fourth chapter, "Comparative Study between Quantum PWM and Classical PWM," presents the results of our experimental analysis. We have applied both the QPWM and classical PWM techniques in a closed-loop control system to regulate the speed of a DC motor. Through a rigorous comparative study, we have obtained results that demonstrate the similarity between the two techniques while highlighting the potential advantages offered by QPWM.

Our project represents a significant step towards the integration of quantum computing principles in power electronics, specifically in the domain of PWM-based DC motor speed control.

# Chapter 1

## From Classical to Quantum Computing

Over the past few years, there has been a revolutionary shift in the world of computing, as traditional classical computers find themselves facing new frontiers and challenges. The emergence of quantum computing has opened up possibilities that were once deemed unimaginable. This chapter explores the transition from classical to quantum computing, exploring the fundamental principles, key concepts, and profound implications that arise from harnessing the power of quantum mechanics for computational purposes.

Classical computing, which forms the backbone of our digital age, has significantly shaped the way we live, work, and interact with technology. It relies on binary digits or bits, represented by 0s and 1s, to process and store information. Classical computers manipulate these bits through logical operations, enabling us to solve complex problems, perform calculations, and execute algorithms efficiently. However, as the demands for computational power increase exponentially, classical computers face inherent limitations in solving certain problems efficiently.

Introducing quantum computing is a paradigm that revolutionizes information processing by leveraging the principles of quantum mechanics. Quantum computers harness quantum bits, or qubits, which can exist in superposition states, representing a blend of 0 and 1 simultaneously. Moreover, qubits can be entangled, resulting in a phenomenon where the state of one qubit becomes intrinsically linked to the state of another, even when physically independent.

This unique behavior of qubits unlocks a wide range of unprecedented computing capabilities. Quantum computers offer the potential to solve complex problems in cryptography, optimization, simulation, and drug discovery with exponential speedups compared to

classical computers. The promise of quantum computing has garnered tremendous attention from researchers, scientists, and industry leaders worldwide, sparking a race towards building practical and scalable quantum computers. However, the transition from classical to quantum computing is not without its challenges. Quantum systems are inherently delicate, highly susceptible to noise and decoherence, which can introduce errors and hinder the reliability of computations. Overcoming these hurdles requires groundbreaking advancements in quantum hardware, error correction techniques, and algorithm design. Throughout this chapter, we will explore the foundational principles of classical and quantum computing, unraveling the differences between the two paradigms, and examining the transformative potential of quantum computing. By comprehending the underlying concepts and implications, we can better appreciate the remarkable progress made in the field and the exciting prospects that lie ahead.

First, we will begin by delving into some fundamental concepts of classical computing..

## 1.1 CLASSICAL COMPUTATION

### 1.1.1 Introduction & History about classical logic

In this section[6], our focus is on classical computation, where we present the concepts in a chronological order. Our starting point is boolean functions and logic, which was first introduced by George Boole in the late nineteenth century. In the 1930s, Claude Shannon examined boolean algebra and discovered that boolean functions could be described using electrical switches. These switches, which are components of electrical circuits that correspond to boolean functions, are called logic gates. Our study of boolean functions will first be in terms of logic and then we will proceed to demonstrate how everything can be translated into circuits and gates. The material covered up to this point is now considered standard and can be found in any introductory computer science text. However, we will also discuss some ideas that are not typically included in the standard introduction.

During the early 1980s, the Nobel Prize-winning physicist Richard Feynman developed an interest in computing, and he gave a course on computation at the California Institute of Technology in the 1970s. These lectures were later documented as the Feynman Lectures on Computation. Feynman's interest in computation was partly due to his association with Edward Fredkin, who had unconventional opinions on physics and computation. Fredkin's belief that the universe functions like a computer and that reversible computation and gates should be studied due to the reversible nature of physics, is not commonly

accepted in the physics community, but his original ideas are admired. One of these ideas is the billiard ball computer. In Feynman's book, he explores the concept of reversible gates and explains how any computation can be achieved by bouncing balls off each other. We adopt Feynman's methodology, as it transpires that reversible gates are precisely what is required for quantum computing. Feynman's exploration of the billiard ball computer sparked his interest in quantum computing, which he further explored by contemplating particle interactions instead of ball interactions. We have included the billiard ball computer concept in this discussion not only due to its sheer simplicity and inventiveness but also because it has been an inspiration for Feynman's work on quantum computing.

### 1.1.2 Logic

In the late nineteenth century, George Boole recognized that specific aspects of logic could be viewed as algebraic, and he established that there were logical laws that could be articulated using algebra. We employ the widely accepted approach of introducing Boolean logic by utilizing truth tables for the fundamental operations of not, and and or.

### 1.1.3 Bits

The term bits has become fairly commonplace[33], where many people will state that bits are zeros and ones. The point is, have we taken a moment to consider what bits really are? Are they just numbers? Why are they important for computers? Let's take the coin as an example, assuming that coins do not balance on their edges, a single coin lying on a flat surface either has heads facing up or tails facing up. Let us call these two possible conditions, or states, heads ( $H$ ) and tails ( $T$ ). If we have two coins, there are four possible states: Both coins can be heads ( $HH$ ), the first can be heads and the second can be tails ( $HT$ ), the first can be tails and the second can be heads ( $TH$ ), or both can be tails ( $TT$ ). That is, the possible states are:

$$HH, HT, TH, TT$$

Since the first coin has two possible states (heads or tails), and the second coin has the same two possible states, there are  $2 \times 2 = 2^2 = 4$  possible states for the two coins. Adding a third coin, there are now eight possible states. They could all be heads, some mixture of heads and tails, or all tails. Listing all the permutations, the possible states are now

$$HHH, HHT, HTH, HTT, THH, THT, TTH, TTT$$

Since each of the three coins has two possible states, there are  $2 \times 2 \times 2 = 2^3 = 8$  possible states for three coins. Generalizing this, if we have  $n$  coins, the possible states range from all heads, through a mixture of heads and tails, to all tails:

$$H...HH, H...HT, \dots, T...TT.$$

With  $n$  coins, there are  $2^n$  possible states.

### 1.1.4 Binary

Previously, when we talked about the coin, when we have 2 coins then we have the four possibilities:

$$HH, HT, TH, TT$$

Now, replacing heads and tails with the bits 0 and 1, now we can write our possibilities as:

$$00, 01, 10, 11$$

In the beginning, we used symbols in order to understand the logic. As for our use of numbers at this stage, it is to standardize the concept of "Binary numbers"

### 1.1.5 Logic Gates

Previously, we treated the notion of bits and learned how we can encode an information from reality into binary numbers, now in this section we will manipulate these bits and know how we can use them to realise more complicated computations using what we call "Logic Gates". Logic gates are used to manipulate bits by taking one or more bits as inputs and producing one or more bits as outputs depending on the inputs. We will start by examining the most basic examples of logic gates, which operate on a single bit. Subsequently, we will delve into logic gates that function on two bits. Lastly, we will cover the physical aspects of logic gates and illustrate how they can execute all computations.

### 1.1.5.1 Single-Bit Gates

Single bit gates act on just one bit, as the circuit diagram depicted in Figure 1.1:

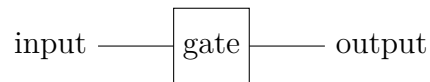


Figure 1.1: Functional bloc of a classic single bit gate.

For this type of gates we have the following fundamental gates:

1. This gate does not make any changing to the bit: 0 remains 0, and 1 remains 1, it is sometimes called the buffer gate. Figure 1.2 illustrates the symbol of the Identity gate whereas Table 1.1 summarizes the corresponding truth table.

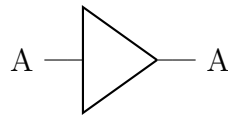


Figure 1.2: The Identity gate

Input (A)	Output (A)
0	0
1	1

Table 1.1: Truth table of the identity gate

2. The NOT gate flips a bit from 0 to 1, or 1 to 0.

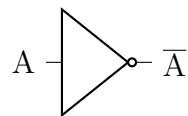


Figure 1.3: The NOT gate

Input (A)	Output ( $\bar{A}$ )
0	1
1	0

Table 1.2: The truth table of the NOT gate

### 1.1.5.2 Two-Bit Gates

A two-bit logic gate accepts two input bits, denoted as A and B, and produces a single output as the Figure 1.4 shows

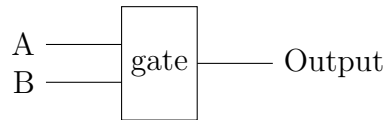


Figure 1.4: Functional bloc of a classic two-bit gate

A	B	Output
0	0	?
0	1	?
1	0	?
1	1	?

Table 1.3: The truth table of two-bit gate

The truth table 1.3 consists of four lines, corresponding to the four possible combinations of inputs: when both A and B are 0, when A is 0 and B is 1, when A is 1 and B is 0, and when both A and B are 1. It is worth noting that these inputs are listed in numerical order, representing the decimal numbers 0, 1, 2, and 3 (00, 01, 10, and 11, respectively). This ordering follows the conventional practice. Since the outputs vary depending on the gate used, the truth table currently displays question marks for the outputs. Each of the four outputs can take on a value of either 0 or 1, resulting in a total of 4 possible two-bit gates ( $2^2 = 4$ ). In the subsequent discussion, we will focus on five of the most significant gates.

1. The AND gate outputs 1 only when both input bits are 1. Its circuit diagram and truth table provided in Figure 1.5 and Table 1.4, respectively:

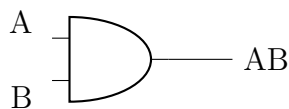


Figure 1.5: The AND gate

In circuit diagram of Figure 1.5, the bits A and B are inputted into an AND gate,

<b>A</b>	<b>B</b>	<b>AB</b>
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.4: The truth table of the AND gate

resulting in the output  $AB$ . It is important to note that standard multiplication principles apply here:  $0 \cdot 0 = 0$ ,  $0 \cdot 1 = 0$ ,  $1 \cdot 0 = 0$ , and  $1 \cdot 1 = 1$ . In various texts, the notation  $A \wedge B$  is also used to represent  $AB$ . This gate is referred to as the AND gate because it produces an output of 1 only when both A and B are 1. In the language of logic, considering 0 as false and 1 as true,  $AB$  is evaluated as true when both A and B are true.

2. The OR gate, which outputs 1 if either input (or both) is 1. Its circuit diagram and truth table provided in Figure 1.6 and Table 1.5, respectively:

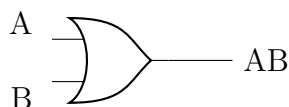


Figure 1.6: The OR gate

<b>A</b>	<b>B</b>	<b>A+B</b>
0	0	0
0	1	1
1	0	1
1	1	1

Table 1.5: The truth table of the OR gate

In accordance with numerous texts, we represent the logical OR operation of A and B as  $A+B$ , although it is important to note that this notation does not correspond to traditional addition because  $1+1$  equals 1 in this context. In certain texts, an alternative notation  $A \vee B$  is also used to indicate the OR operation.

3. The Exclusive OR (XOR) gate, which outputs 1 when only one input is one, but not both, note that this gate illustrate the concept of true addition that results 0



with a carry of 1. Its circuit diagram and truth table provided in Figure 1.7 and Table 1.6, respectively:

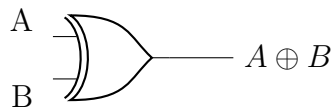


Figure 1.7: The XOR gate

<b>A</b>	<b>B</b>	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Table 1.6: The truth table of the XOR gate

4. The NAND gate, which stands for NOT of AND, and which outputs the NOT of the AND of the bits. Its circuit diagram and truth table provided in Figure 1.8 and Table 1.7, respectively:

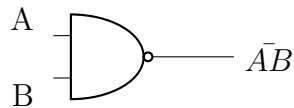


Figure 1.8: The NAND gate

<b>A</b>	<b>B</b>	$\bar{A}B$
0	0	1
0	1	1
1	0	1
1	1	0

Table 1.7: The truth table of the NAND gate

5. The NOR gate, which stands for NOT of OR, and which outputs the NOT of the OR of the bits. Its circuit diagram and truth table provided in Figure 1.9 and Table 1.8, respectively:

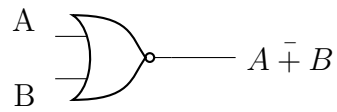


Figure 1.9: The NOR Gate

<b>A</b>	<b>B</b>	$A + \bar{B}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 1.8: The truth table of the NOR gate

This section allowed us to review the various fundamental concepts of classical combinatorial logic and serves as an introduction to the subsequent part that will introduce the foundations of quantum computing, emphasizing the equivalences and subtleties with classical logic.

## 1.2 BASICS OF QUANTUM COMPUTING

### 1.2.1 Introduction and overview

The objective of this research is to explore the fundamental concepts of quantum computation and quantum information[26], their development, and their practical applications. The introductory chapter aims to provide a broad overview of the field by answering several key questions. Firstly, it will examine the central concepts of quantum computation and information and how they have evolved over time. Secondly, it will investigate the potential uses of quantum computing and information. Lastly, the section will outline how these concepts will be presented throughout the research to provide readers with a basic understanding of the field and guide them on how to approach the rest of the material. Overall, this section seeks to offer a comprehensive picture of the field of quantum computation and quantum information, including its core concepts, historical evolution, and practical implications.

### 1.2.2 History of quantum computation and quantum information

The story of quantum computing begins at the turn of the twentieth century when an unheralded revolution was underway in science. A series of crises had arisen in physics. The problem was that the theories of physics at that time (now dubbed classical physics) were predicting absurdities such as the existence of an ‘ultraviolet catastrophe’ involving infinite energies, or electrons spiraling inexorably into the atomic nucleus. Initially, these problems were addressed by introducing ad hoc assumptions into classical physics. However, as our comprehension of atoms and radiation improved, these attempted explanations grew increasingly intricate and convoluted. The crisis came to a head in the early 1920s after a quarter century of turmoil, and resulted in the creation of the modern theory of quantum mechanics. Quantum mechanics has been a fundamental part of science ever since, and has been applied with enormous success to everything, including the structure of the atom, nuclear fusion in stars, superconductors, the structure of DNA, and the elementary particles of Nature.

Quantum mechanics is a mathematical framework or set of rules for the construction of physical theories. For example, there is a physical theory known as quantum electrodynamics which describes with remarkable accuracy the interaction of atoms and light.

The relationship of quantum mechanics to specific physical theories like quantum electrodynamics is rather like the relationship of a computer's operating system to specific applications software – the operating system sets certain basic parameters and modes of operation, but leaves open how specific tasks are accomplished by the applications.

### 1.2.3 Single Quantum bit system

#### 1.2.3.1 Superposition

A qubit is a unit of quantum information, analogous to the classical bit, which can represent a state of superposition between two classical states. In other words, a qubit can exist in multiple states simultaneously and can perform quantum operations such as entanglement and interference[27].

A common example of a qubit is an electron spin, which can be "up" or "down" (corresponding to the classical states 0 and 1), but in quantum mechanics can also exist in a superposition of both states at once. Another example is a photon, which can be polarized in different directions, again allowing for a superposition of states. The qubit is a fundamental building block of quantum computing and quantum information processing. Using bra-ket notation or Dirac notation[11] from quantum physics, we write 0 and 1 enclosed between a vertical bar and an angle bracket called a ket:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (1.1)$$

Where  $|0\rangle$ ,  $|1\rangle$  are vectors represented in the Hilbert space[34] that is mathematical space that describes the state of a quantum mechanical system. And we can write the qubit as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1.2)$$

where  $\alpha$  and  $\beta$  are complex coefficients that determine the probability amplitudes of measuring the qubit in the basis states  $|0\rangle$  and  $|1\rangle$ , which are related to the Born rule[8]:

$$|\alpha|^2 + |\beta|^2 = 1 \quad (1.3)$$

#### 1.2.3.2 Inner & Outer product

In quantum computing, the inner product is a mathematical operation used to calculate the probability of obtaining a certain measurement outcome. It takes two quantum states,

represented as vectors in a complex vector space, and returns a complex number. The inner product is used to calculate the probability of measuring a state after a quantum operation, and it is defined as the sum of the products of the complex conjugate of the first state's coefficients and the second state's coefficients.

$$\langle \psi | \phi \rangle = \sum_i \psi_i^* \phi_i \quad (1.4)$$

For example, if we have two qubits represented by the states  $|0\rangle$  and  $|1\rangle$ , the inner product of these states is equal to the complex number 0:

$$\langle 0 | 1 \rangle = 0 \quad (1.5)$$

This has the meaning that the two states are independent and cannot be equal.

The inner product between the qubit state  $|\psi\rangle$  and the basis state  $|0\rangle$  give us the complex coefficient (the amplitude) related to this latter:

$$\langle 0 | \psi \rangle = \alpha \langle 0 | 0 \rangle + \beta \langle 0 | 1 \rangle = \alpha \quad (1.6)$$

On the other hand, the outer product is a mathematical operation that takes two quantum states and returns a matrix.

The outer product is employed in the design and implementation of quantum algorithms. Many quantum algorithms, such as the Quantum Fourier Transform (QFT)[5] and the Quantum Phase Estimation (QPE), rely on the tensor product to operate on multiple qubits simultaneously and perform complex calculations.

Given two quantum states represented as column vectors  $|\phi\rangle$  and  $|\psi\rangle$ , the outer product is defined as:

$$|\psi\rangle\langle\phi| = \begin{pmatrix} \psi_1\phi_1^* & \psi_1\phi_2^* & \cdots & \psi_1\phi_n^* \\ \psi_2\phi_1^* & \psi_2\phi_2^* & \cdots & \psi_2\phi_n^* \\ \vdots & \vdots & \ddots & \vdots \\ \psi_n\phi_1^* & \psi_n\phi_2^* & \cdots & \psi_n\phi_n^* \end{pmatrix} \quad (1.7)$$

For example, if we have two qubits represented by the states  $|0\rangle$  and  $|1\rangle$ , the outer product of these states is:

$$|0\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad (1.8)$$

### 1.2.3.3 Measurement

The act of measurement in quantum computing involves converting quantum information stored in a quantum system into classical information[2]. This is usually done by reading

out a classical bit, which determines whether the quantum system is in a state of 0 or 1. It allows us to collapse the qubit into a given state depending on the probabilities each of its coefficients carries.

However, in quantum mechanics, measurement outcomes are probabilistic, meaning that there is a certain probability of obtaining a specific measurement outcome. For the single qubit state  $|\psi\rangle$ , the probabilities of obtaining the measurement outcomes  $|0\rangle$  and  $|1\rangle$  are given by the square of the absolute value of the inner product between  $|0\rangle$ ,  $|1\rangle$  (respectively) and the state  $|\phi\rangle$ .

$$|\langle 0|\psi\rangle|^2 = |\alpha|^2 \quad |\langle 1|\psi\rangle|^2 = |\beta|^2 \quad (1.9)$$

This relationship can be generalized to the probability of obtaining any bit string  $|x_1 \dots x_n\rangle$  after measuring an  $n$ -qubit state,  $|\phi\rangle$ . In this case, the probability is given by the square of the absolute value of the inner product between the bit string and the state  $|\phi\rangle$ .

#### 1.2.3.4 Bloch Sphere

The Bloch Sphere is used to represent graphically the state of a single Qubit. We can move the end of a state vector along the surface of the Bloch Sphere by applying unitary operators (which we will introduce in the next section) to the state vector[13]. The idea behind this representation, as we will see, is to get rid of the complex nature of the states' amplitudes, and only deal with real numbers by representing the relative phase between the amplitudes within a state vector with a rotation in the Bloch sphere, and neglecting the global phase since it has no effect on the output.

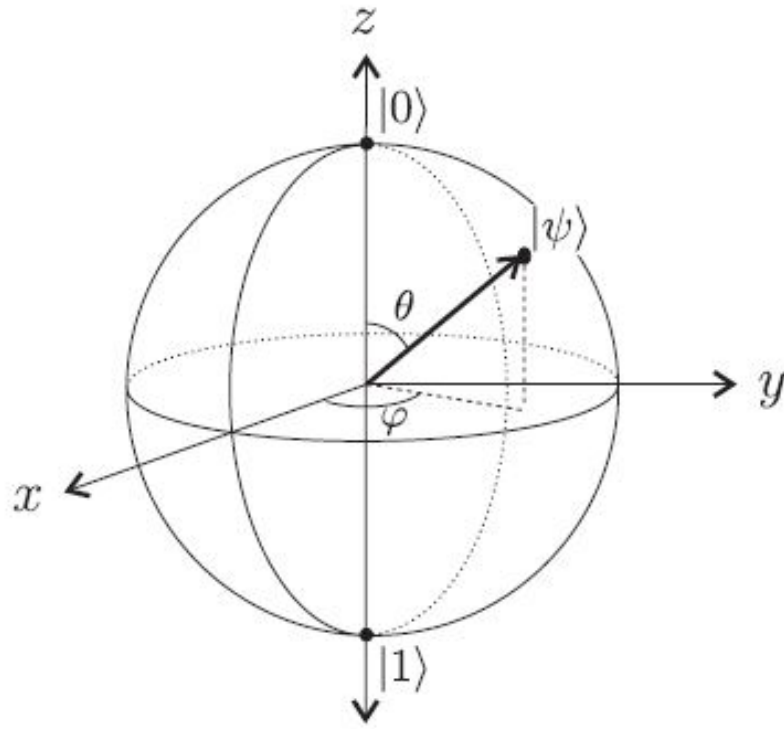


Figure 1.10: State of a qubit on the Bloch sphere.

As we saw in the previous section, a general state description of the Qubit is given by:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \alpha, \beta \in \mathbb{C} \quad (1.10)$$

Using the Bloch Sphere we can develop the equivalent form of the qubit state equation as following:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \quad (1.11)$$

Where:

$$\theta \in [0 \ \pi], \quad \varphi \in [0 \ 2\pi)$$

### 1.2.3.5 Processing & Time-Evolution of the quantum information

Since physical systems change in time, the state  $|\psi\rangle$  of a system will evolve to the state  $|\psi'\rangle$  between the two moments  $t_1$  and  $t_2$ , respectively, by the unitary operator  $U$ ,

$$|\psi'\rangle = U|\psi\rangle \quad (1.12)$$

Mathematically speaking, the evolution term leads us to a very interesting concept of the "Unitary Operator" which describes how quantum states of systems can be changed. A square matrix  $U$  having complex numbers entries is "Unitary" if it satisfies the equality:

$$U^\dagger U = U U^\dagger = I \quad (1.13)$$

where  $U^\dagger$  denotes the adjoint (conjugate transpose) of  $U$  and  $I$  represents the identity operator.

Note that quantum gates are unitary matrices, and unitary matrices are quantum gates. When discussing unitary notion, it is essential to introduce a fundamental and significant concept that is reversibility which is actually a consequence of unitarity. It refers to the ability to undo computational steps and recover the initial quantum state.

Another way of understanding irreversibility is to think of it in terms of information erasure[26]. If a logic gate is irreversible, then some of the information input to the gate is lost irretrievably when the gate operates – that is, some of the information has been erased by the gate. Conversely, in a reversible computation, no information is ever erased, because the input can always be recovered from the output. Thus, saying that a computation is reversible is equivalent to saying that no information is erased during the computation.

A matrix  $M$  is considered reversible or invertible if there exists a corresponding matrix denoted as  $M^{-1}$  such that:

$$MM^{-1} = M^{-1}M = I \quad (1.14)$$

Now, since a quantum gate  $U$  must be unitary, according to the equation 1.13, the inverse of  $U$  is simply  $U^\dagger$ , i.e.,  $U^{-1} = U^\dagger$ . So, a quantum gate is always reversible, and its inverse is its conjugate transpose[33].

### 1.2.3.6 Single Qubit Operators

They are quantum gates that act on a single qubit, i.e., a quantum bit. These gates are represented by unitary matrices, which are complex square matrices that satisfy certain properties, such as being invertible and preserving the inner product of quantum states. Some common examples of unitary gates in quantum computing include:

1. Identity gate: This gate leaves the state of a qubit unchanged. It is represented by the following unitary matrix:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (1.15)$$

2. Pauli-X gate (or NOT gate): This gate flips the state of a qubit from  $|0\rangle$  to  $|1\rangle$  or vice versa. It is represented by the following unitary matrix:

$$X = |0\rangle\langle 1| + |1\rangle\langle 0| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (1.16)$$



3. Pauli-Y gate: This gate rotates the state of a qubit around the Y-axis of the Bloch sphere by  $\pi$  radians. It is represented by the following unitary matrix:

$$Y = -i|0\rangle\langle 1| + i|1\rangle\langle 0| = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (1.17)$$

4. Pauli-Z gate: This gate rotates the state of a qubit around the Z-axis of the Bloch sphere by  $\pi$  radians. It is represented by the following unitary matrix:

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (1.18)$$

5. Hadamard gate: This gate creates a superposition of the  $|0\rangle$  and  $|1\rangle$  states by rotating the state of a qubit around an axis between the X-axis and the Z-axis of the Bloch sphere. It is represented by the following unitary matrix:

$$H = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (1.19)$$

6. Shift gate (or Phase gate): This gate adds a relative phase to the  $|1\rangle$  state of a qubit. It is represented by the following unitary matrix:

$$S_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \quad (1.20)$$

7. Rx Rotation Gate:

The Rx rotation matrix represents a rotation around the x-axis in a quantum system. It is used to introduce a phase shift and change the amplitudes of the quantum state. The general form of the Rx rotation matrix is:

$$Rx(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i \sin\left(\frac{\theta}{2}\right) \\ -i \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (1.21)$$

8. Ry Rotation Gate:

The Ry rotation matrix represents a rotation around the y-axis in a quantum system. It is used to introduce a phase shift and change the amplitudes of the quantum state. The general form of the Ry rotation matrix is:

$$Ry(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (1.22)$$

### 9. Rz Rotation Gate:

The Rz rotation matrix represents a rotation around the z-axis in a quantum system. It is used to introduce a phase shift and change the amplitudes of the quantum state. The general form of the Rz rotation matrix is:

$$Rz(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \quad (1.23)$$

Note that  $X$ ,  $Y$  and  $Z$  gates along with the identity gate are known as the Pauli gates, and any two qubits operator can be expressed using them, for example:

$$Rx(\theta) = \cos\left(\frac{\theta}{2}\right) \cdot I - i \sin\left(\frac{\theta}{2}\right) \cdot X \quad (1.24)$$

$$Ry(\theta) = \cos\left(\frac{\theta}{2}\right) \cdot I - i \sin\left(\frac{\theta}{2}\right) \cdot Y \quad (1.25)$$

$$Rz(\theta) = \cos\left(\frac{\theta}{2}\right) \cdot I - i \sin\left(\frac{\theta}{2}\right) \cdot Z \quad (1.26)$$

These gates, along with other unitary and multi-qubit gates, form the basis of quantum circuits that can perform various computations and transformations on quantum states.

## 1.2.4 Multiple Quantum Bits

Multiple quantum bits, or multi-qubits, are systems composed of two or more qubits. The state of a multi-qubit system can be represented by a vector in a high-dimensional Hilbert space, where the dimension is determined by the number of qubits.

### 1.2.4.1 Tensor Product:

When we have multiple qubits, we write their states as a tensor product [33]  $\otimes$ . For example, two qubits, both in the  $|0\rangle$  state, are written as:

$$|0\rangle \otimes |0\rangle \quad (1.27)$$

This is pronounced "zero tensor zero". Often, we compress the notation and leave out the tensor product in both writing and speech:

$$|0\rangle|0\rangle$$

We frequently compress the notation further until:

$$|00\rangle \tag{1.28}$$

With two qubits, the basis states are:  $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ . A general state is a superposition of these basis:

$$c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle \tag{1.29}$$

If we measure these two qubits states, we get:

$|00\rangle$  with probability  $|c_0|^2$

$|01\rangle$  with probability  $|c_1|^2$

$|10\rangle$  with probability  $|c_2|^2$

$|11\rangle$  with probability  $|c_3|^2$

Thus, the total probability is  $|c_0|^2 + |c_1|^2 + |c_2|^2 + |c_3|^2$ , and it should equal 1.

With three qubits, there are eight basis states or bit-strings:

$$|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, \text{ and } |111\rangle$$

#### 1.2.4.2 Entanglement:

Entanglement in quantum computing refers to a phenomenon where two or more quantum particles, such as electrons or photons, become correlated in such a way that their individual states cannot be described independently. When two particles are entangled, their quantum states are linked together, even if they are physically separated by large distances. This allows for the creation of highly interconnected quantum systems that can perform certain types of calculations much more efficiently than classical computers, also entanglement is behind the important concept of teleportation.

In general, states of a system of which cannot be expressed as a tensor product of states of its individual subsystems are called entangled states[2]. For example, it is possible for three qubits to be in a state that cannot be written as the tensor product of three single qubit states as following:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \tag{1.30}$$

For a system of  $n$  qubits, this means that an entangled state cannot be written a tensor product of  $n$  single qubit states. Entanglement makes it possible to create a complete  $2^n$  dimensional complex vector space to do our computations in, using just  $n$  physical qubits.

### 1.2.4.3 Multiple Qubits operators

They are mathematical operations that act on two input qubits or more. Here are some of the commonly used multiple qubits operators:

1. Controlled NOT (CNOT) Gate: This is a two-qubit gate where one qubit (called the control qubit) controls the operation on the other qubit (called the target qubit). The CNOT gate flips the state of the target qubit if and only if the control qubit is in the state  $|1\rangle$ . Its matrix representation is:

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (1.31)$$

2. Swap Gate: This is a two-qubit gate that swaps the states of two qubits. Its matrix representation is:

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1.32)$$

3. Controlled Phase Shift Gate: This is a two-qubit gate that applies a phase shift to the target qubit depending on the state of the control qubit. Its matrix representation is:

$$CP(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix} \quad (1.33)$$

In this matrix, the first two rows and columns correspond to the control qubit, and the last two rows and columns correspond to the target qubit. The diagonal elements are all equal to 1, and the bottom right element is a phase shift of  $e^{i\phi}$ .

4. Toffoli Gate (CCNOT): This is a three-qubit gate that applies a NOT gate to the

target qubit if both control qubits are in the state  $|1\rangle$ . Its matrix representation is:

$$CCNOT = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (1.34)$$

#### 1.2.4.4 Quantum Circuit

A quantum circuit is a graphical representation of the sequence of quantum operations that are applied to a set of quantum bits, or qubits. It is analogous to classical digital circuits, which are composed of logic gates, but in the quantum domain, it represents the flow of quantum information.

- **Reading Quantum Circuits:**

Quantum circuits are typically represented using a notation known as the quantum circuit diagram. In this diagram, qubits are represented by horizontal lines, and quantum gates are represented by labeled boxes acting on one or more qubits. The direction of time is depicted from left to right, indicating the order in which operations are applied.

- **Qubit Lines:**

Each horizontal line in the circuit diagram represents a qubit. The qubits are often labeled with their corresponding indices, such as  $|q\rangle$ ,  $|q\rangle$ , and so on. The leftmost side of the circuit represents the initial state of the qubits, while the rightmost side represents the final state after applying all the quantum operations.

- **Quantum Gates:**

Quantum gates are represented by labeled boxes, which act on one or more qubits. These gates manipulate the quantum state of the qubits to perform specific computations. Some common quantum gates include the Pauli gates (X, Y, Z), Hadamard gate (H), and controlled-NOT gate (CNOT).

#### Example Circuit with CNOT Gate:

Figure 1.11 an example of a quantum circuit that contains a CNOT gate:

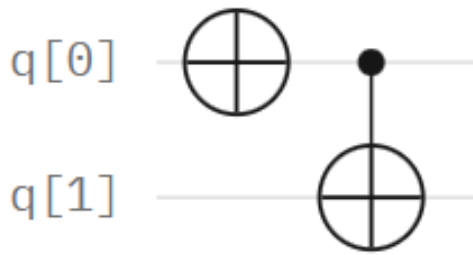


Figure 1.11: Quantum circuit that contains a CNOT gate

In this circuit, there are two qubits, labeled as  $|q_0\rangle$  and  $|q_1\rangle$ . The CNOT gate is represented by the box with an X inside it. The control qubit is  $|q_0\rangle$ , and the target qubit is  $|q_1\rangle$ . The lines connecting the control and target qubits indicate that the CNOT gate acts on these qubits.

The CNOT gate flips the target qubit ( $|q_1\rangle$ ) if and only if the control qubit ( $|q_0\rangle$ ) is in the state  $|1\rangle$ . This gate is widely used in quantum computing for entangling qubits and performing various computations.

By combining multiple quantum gates in a quantum circuit, it is possible to perform complex quantum computations and algorithms. Quantum circuits serve as a powerful tool for understanding and designing quantum algorithms, and they play a crucial role in the field of quantum computing.

### 1.3 Quantum Programming

Quantum computing is currently emerging from the research lab onto the marketplace. Many companies are building prototype quantum processors, and although these devices are not yet good enough for fault-tolerant quantum computation, they may still have uses. These rudimentary quantum processors are called noisy intermediate-scale quantum (NISQ) devices, where noisy means they suffer from too much decoherence to be fault-tolerant, and intermediate-scale means they have a moderate number of qubits, say roughly fifty to a few hundred. NISQ devices were used to demonstrate quantum computational supremacy. Many companies have made their rudimentary quantum processors available for people to experiment with. In this section, we will learn how to deal with IBM's quantum computers over the internet, Whereas, IBM has made several of their quantum processors freely available to the public, making them a prudent choice for a textbook. Furthermore, after learning one quantum programming toolkit, it will be easier to learn others, as there are many similarities across them, on the occasion of this, we'll

also take a look at Quirk toolkit which we will use later in our simulation.

## 1.3.1 IBM Quantum

### 1.3.1.1 Resources

IBM was the first to make their quantum processors available over the internet (over the “cloud”), and their online platform is called IBM Quantum Experience. It can be accessed at <https://quantum-computing.ibm.com>. Their smaller quantum processors are available to the public, and access to their larger, newer processors is available commercially. When we log in, we first see the Dashboard:

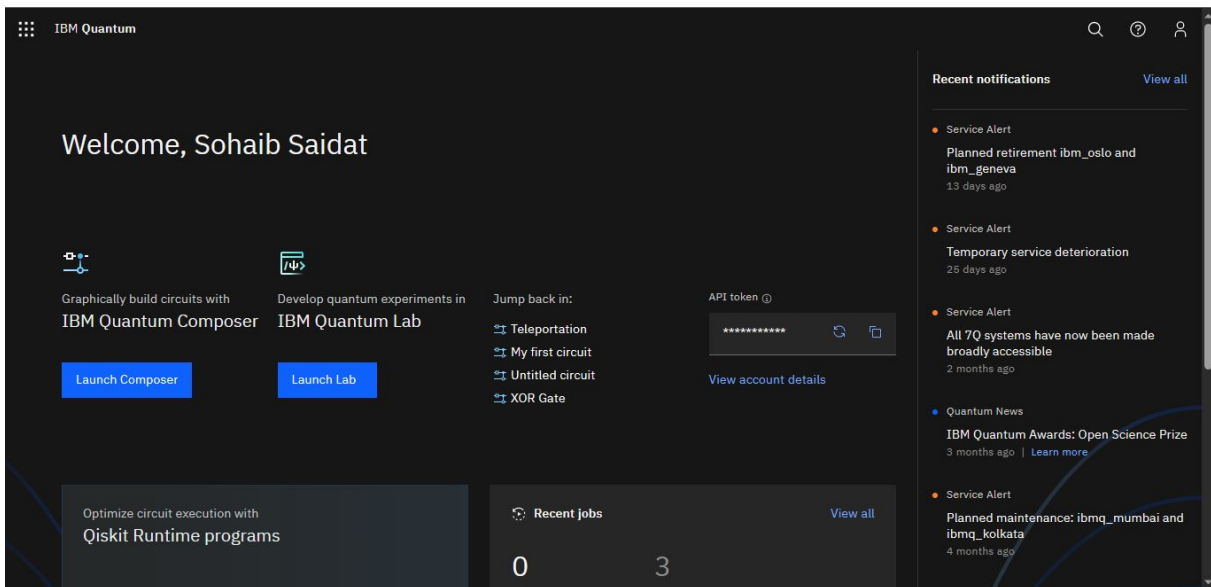
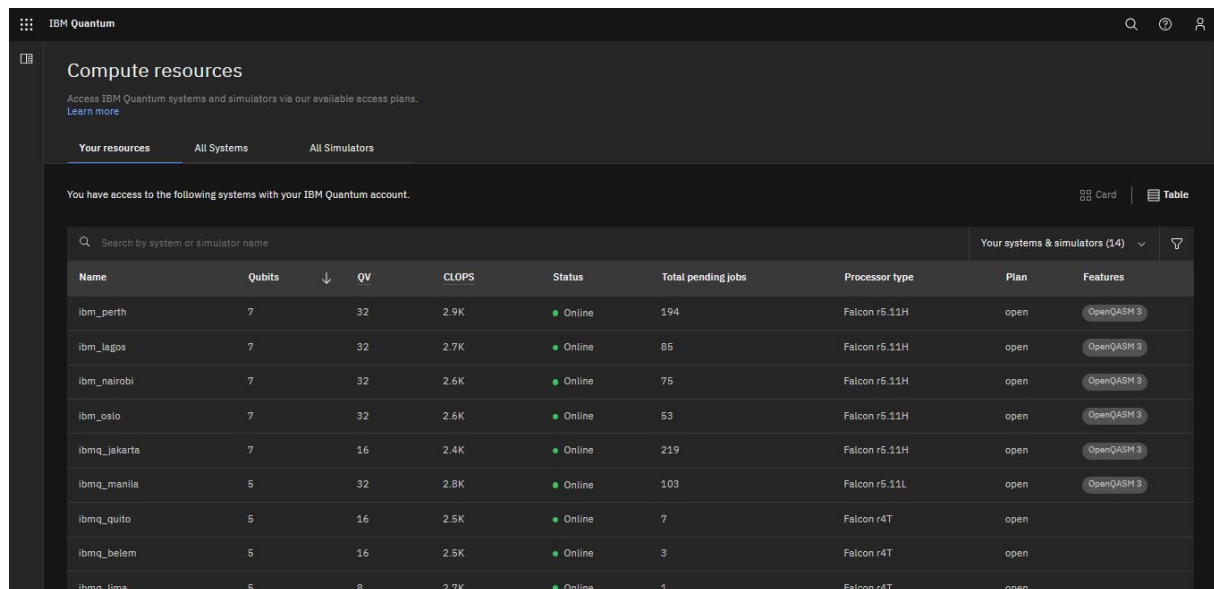


Figure 1.12: Dashboard of IBM quantum, image generated from[16]

We can go to the "Compute resources" page to view a list of quantum processors available to us.



The screenshot shows the 'Compute resources' page in the IBM Quantum interface. It lists 14 systems and simulators. The table below summarizes the data shown in the screenshot.

Name	Qubits	QV	CLOPS	Status	Total pending jobs	Processor type	Plan	Features
ibmq_perth	7	32	2.9K	Online	194	Falcon r5.11H	open	OpenQASM 3
ibmq_legos	7	32	2.7K	Online	85	Falcon r5.11H	open	OpenQASM 3
ibmq_nairobi	7	32	2.6K	Online	75	Falcon r5.11H	open	OpenQASM 3
ibmq_oslo	7	32	2.6K	Online	53	Falcon r5.11H	open	OpenQASM 3
ibmq_jakarta	7	16	2.4K	Online	219	Falcon r5.11H	open	OpenQASM 3
ibmq_manila	5	32	2.8K	Online	103	Falcon r5.11L	open	OpenQASM 3
ibmq_quito	5	16	2.5K	Online	7	Falcon r4T	open	
ibmq_belem	5	16	2.5K	Online	3	Falcon r4T	open	
ibmq_lima	5	8	2.7K	Online	1	Falcon r4T	open	

Figure 1.13: IBM quantum processors

Figure 1.13 illustrates the list of quantum processors that are available to us. If we click on a processor, such as `ibmq manila`, we can see more information about it:

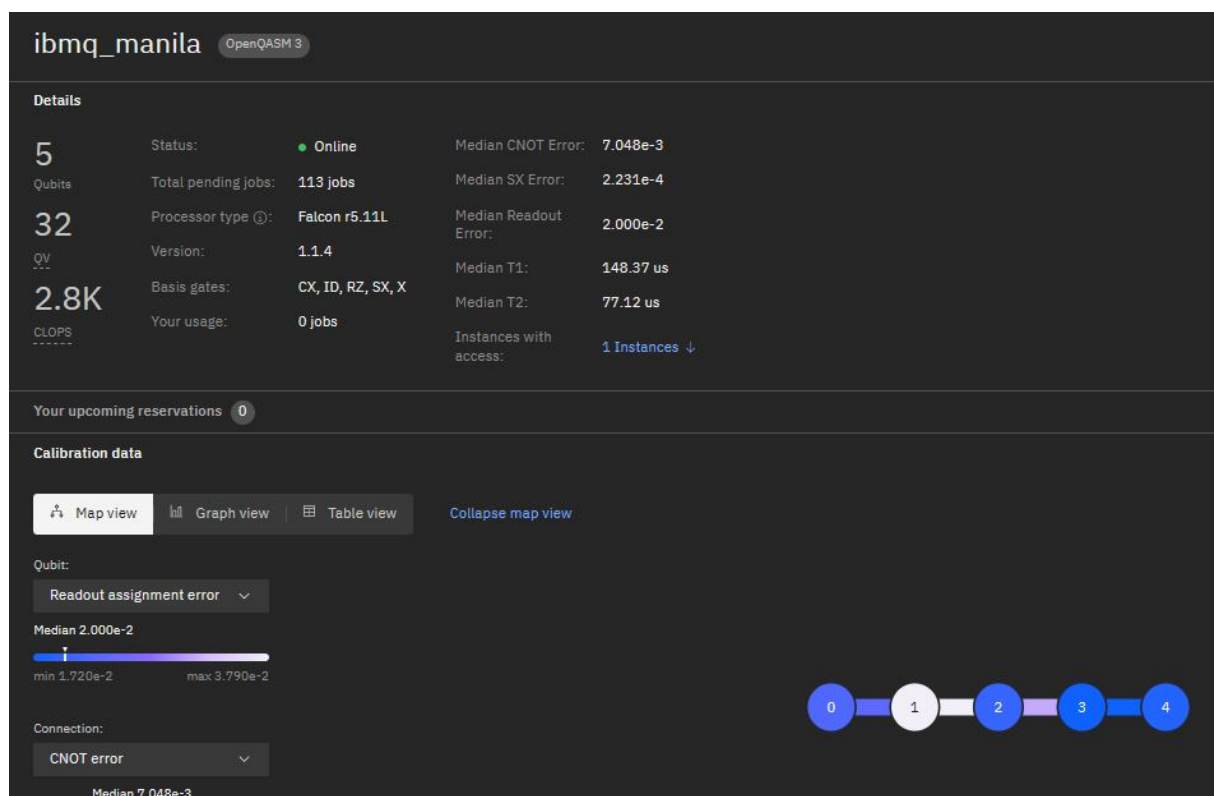


Figure 1.14: Ibmq manila processor

Figure 1.14 shows that Manila quantum processor has five qubits arranged in a line. This arrangement, or topology, can affect which quantum gates can be naturally applied. For example, we can naturally apply CNOT between qubits 0 and 1. If we want to apply



CNOT between qubits 0 and 2, however, we would need to, for example, SWAP qubits 2 and 1, apply CNOT between 0 and 1, then SWAP 1 back with 2.

### 1.3.1.2 Circuit Composer

The Circuit Composer provides a drag-and-drop interface for programming quantum circuits. To get to the Circuit Composer, we can click on the menu icon in the top-left corner and then click “Quantum Composer:”

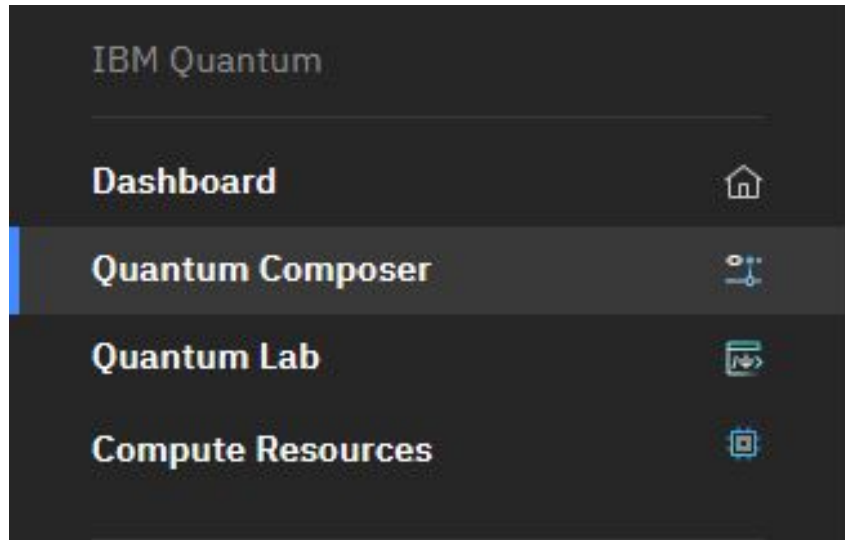


Figure 1.15: IBM quantum Composer

Figure 1.16 depicts an illustrative example of Greenberger–Horne Zeilinger state:

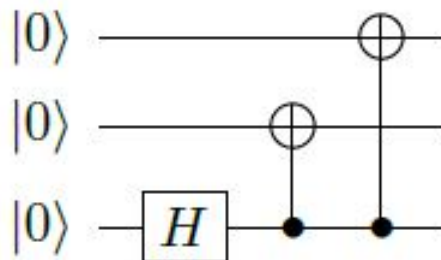


Figure 1.16: GHZ state circuit

This circuit produces the following state:

$$\begin{aligned}
 |000\rangle &\xrightarrow{H\otimes I\otimes I} \frac{1}{\sqrt{2}} (|000\rangle + |100\rangle) \\
 &\xrightarrow{\text{CNOT}_{21}} \frac{1}{\sqrt{2}} (|000\rangle + |110\rangle) \\
 &\xrightarrow{\text{CNOT}_{20}} \frac{1}{\sqrt{2}} (|000\rangle + |111\rangle).
 \end{aligned}$$

This state is known as the Greenberger–Horne Zeilinger state (GHZ state). It is an entangled state that we will revisit it in the next chapter. If we measure it, we find that all the qubits are 0 with probability 1/2 or all 1 with probability 1/2.

Using the Circuit Composer, we can create this circuit using a Hadamard gate and two CNOT gates:

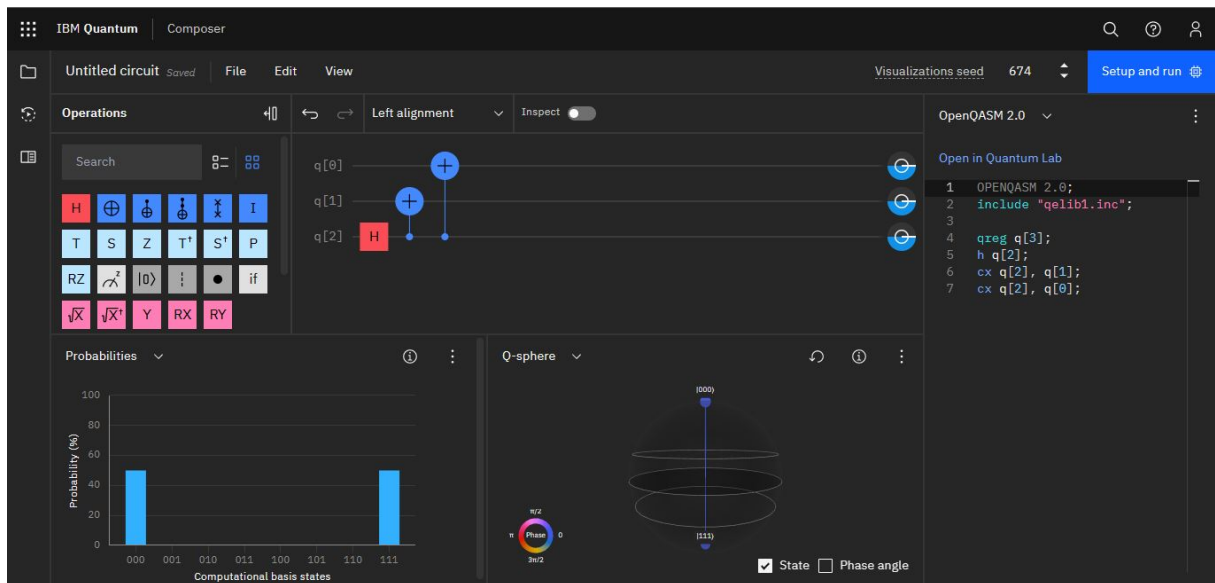
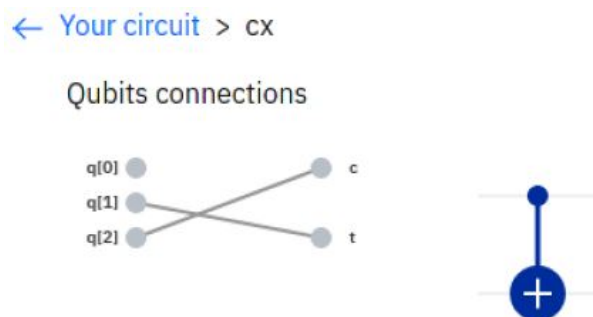


Figure 1.17: GHZ state circuit using composer

To change the control and target of CNOTs, we double-clicked on them and modified which qubit was the control and which was the target. For example, for  $CNOT_{21}$ , the control and target were set as shown below:



In the Circuit Composer, we also deleted some qubits so that there are only three. At the bottom of the webpage, the Circuit Composer automatically simulated the circuit, showing histograms indicating that the circuit yields  $|000\rangle$  with probability 50% or  $|111\rangle$  with probability 50%, as expected.

### Example: Teleportation

We used this example for two purposes. The first is to better familiarize with the programming interface "Quantum Composer", and the second is to employ the notion of entanglement in the circuit and know how it works in reality. So quantum teleportation[7] has been in the news as a result of developments in the field of quantum computing in general, let's understand what this notion means in a very simple way, so it is the transfer of quantum states from one qubit to another and this doesn't mean physically transporting a qubit from one place to another but in fact this is a transfer of quantum information from one qubit to another, It involves the transfer of quantum states, such as the spin or polarization of a photon, from one location to another by using entanglement, now, the question arises: why are we engaging in these endeavors? the answer is : now in classical computer copying something and transforming it isn't a problem, in fact we do it every day, however in quantum computers the act of trying to transfer by copying is not allowed because the moment we copy we are actually implicitly doing a measurement which destroys the quantum state that we're trying to transfer from point A to point B. In order to get around this problem we're going to take advantage of entanglement as a resource and build a circuit which is the quantum teleportation circuit, so firstly we create 3 classical bits and 3 quantum bits and we make the circuit to the end using various quantum gates and also the appropriate measurements, finally we obtain the following circuit:

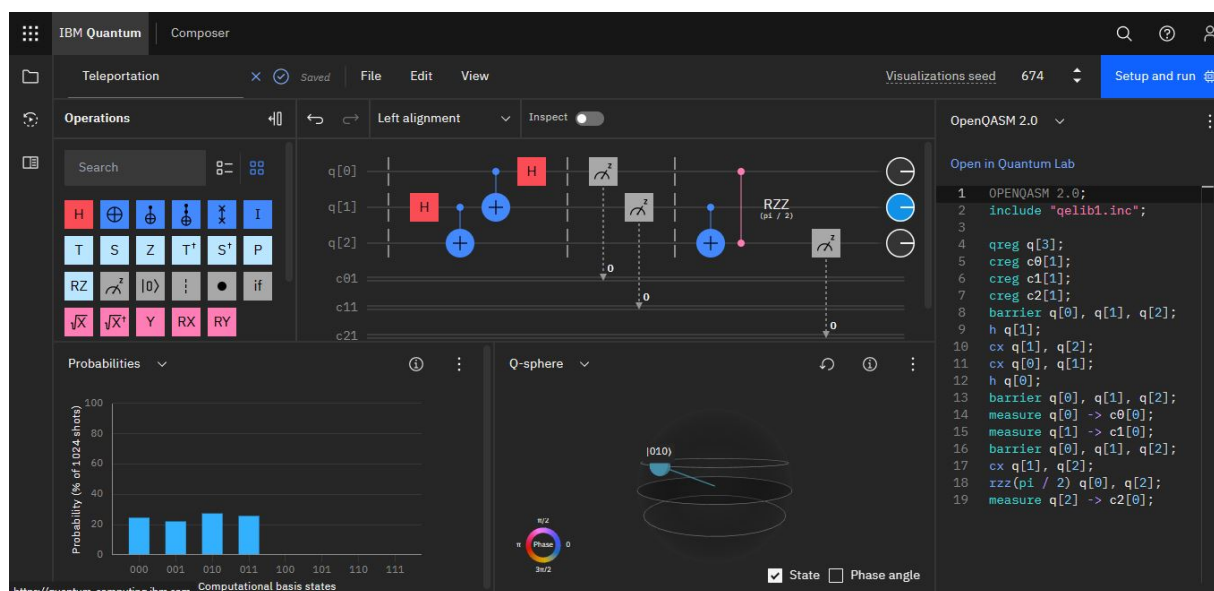


Figure 1.18: Teleportation circuit

The protocol requires a sender, a receiver, and a shared entangled state between them. The sender performs measurements on the qubit they wish to transfer and their half of

the entangled state, and communicates the results to the receiver. Based on the measurement results, the receiver performs a specific quantum gate operation on their half of the entangled state, which allows them to reconstruct the original qubit in their location. To summarize, the circuit for teleportation typically involves three qubits: the sender's qubit (which is to be teleported), the entangled qubit (which is shared between the sender and the receiver), and the receiver's qubit (which will receive the teleported state).

The entangled qubit is used to establish a special correlation between the sender's qubit and the receiver's qubit. This correlation allows the sender to "teleport" the state of their qubit to the receiver, without actually physically sending the qubit itself.

We can also highlight that entanglement forms the fundamental basis of quantum encryption, a technique that goes beyond the traditional binary representation of 0s and 1s. In quantum encryption, entanglement allows for the creation of an infinite number of transferable keys, represented by the Bloch sphere. This property of entanglement enables the establishment of highly secure data communication, offering unprecedented levels of data security. By utilizing entanglement, quantum encryption provides a robust mechanism to protect sensitive information from eavesdropping or unauthorized access. Returning to the example at hand, in the beginning when we had  $|000\rangle$  we see clearly that the first qubit (sender as an input) and the third qubit (receiver as an output) are in the same state  $|0\rangle$  and all the possibilities for the third qubit to be one are zero in the probabilities histogram, now let's see what happens when we flip the first qubit to one  $|1\rangle$ :

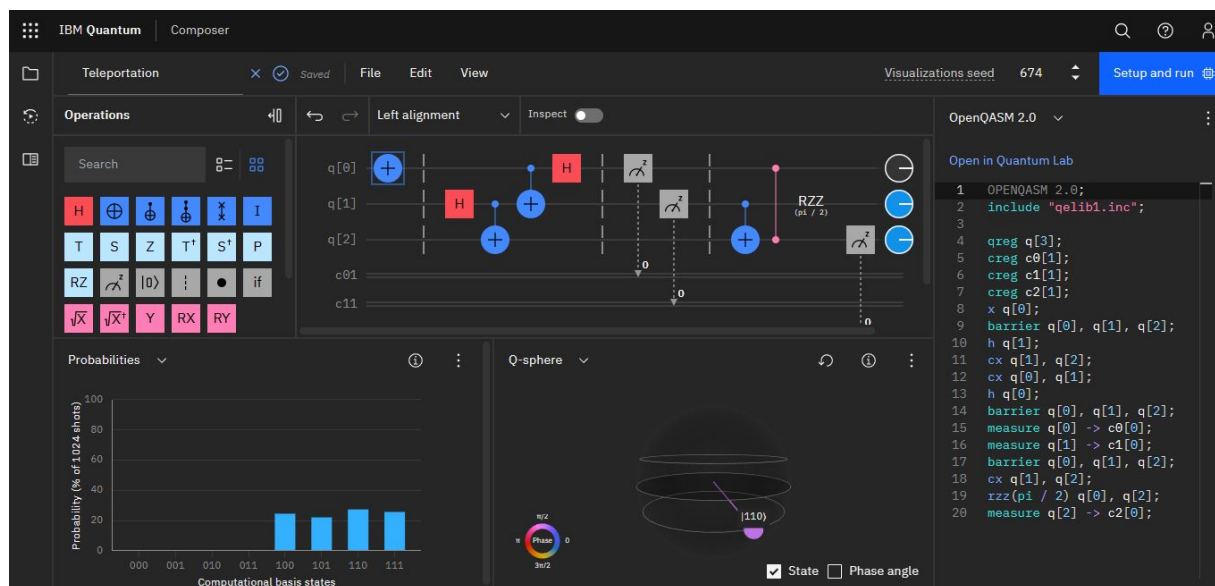


Figure 1.19: Teleportation circuit

When the input of the first qubit (the sender) is  $|1\rangle$  then output of the third qubit (the

receiver) is also one, and all the possibilities for the third qubit to be zero are zero in the probabilities histogram.

### 1.3.2 Quirk toolkit

Quirk is a quantum circuit simulator and visualization toolkit that allows users to design, simulate, and analyze quantum circuits in an interactive and intuitive manner. It is designed to provide a user-friendly interface for exploring and experimenting with quantum computation concepts.

The main features and capabilities of the Quirk toolkit include:

1. **Quantum Circuit Design:** Quirk provides a drag-and-drop interface that allows users to easily design quantum circuits. Users can add and connect various quantum gates, including single-qubit gates, multi-qubit gates, and measurement operations, to construct their desired circuit configurations.
2. **Interactive Simulation:** Quirk enables users to simulate the behavior of quantum circuits in real-time. It visualizes the state of each qubit throughout the computation, allowing users to observe how quantum gates transform the qubit states and how measurements collapse the quantum states.
3. **Visualization and Analysis:** Quirk offers a range of visualization options to help users understand and analyze quantum circuits. It provides visual representations of quantum states, including the Bloch sphere representation for single-qubit states, as well as histograms and probability distributions for measurement outcomes.
4. **Error and Noise Modeling:** Quirk includes features for modeling errors and noise in quantum circuits. Users can introduce various types of noise models, such as depolarizing noise or amplitude damping, to observe the effects of noise on quantum computations.
5. **Educational Tool:** Quirk is widely used as an educational tool to teach and learn quantum computing concepts. It provides an intuitive interface that allows beginners to explore quantum circuits and gain insights into quantum phenomena, while also offering advanced features for in-depth experimentation.

Overall, Quirk serves as a tool for experimenting with quantum circuits, visualizing quantum states, and gaining a deeper understanding of quantum computing principles.

**Example:**

**Quantum teleportation circuit:**, which we explained earlier, with Bloch sphere displays (showing that the qubit at the top has ended up at the bottom) :

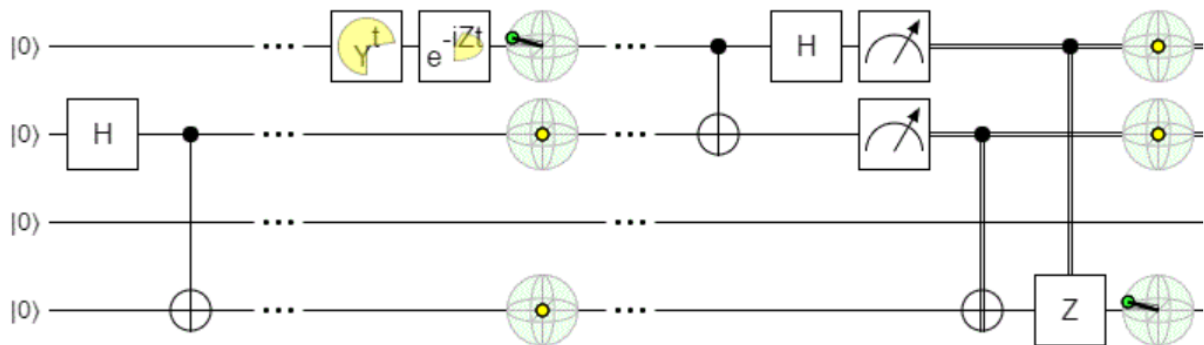


Figure 1.20: Teleportation algorithm.

# Chapter 2

## Speed Control of DC Motors using PWM

### 2.1 INTRODUCTION TO SPEED CONTROL

Speed control refers to the deliberate adjustment of the driving speed to meet the specific requirements of a particular work process. It is important to note that speed control does not encompass the natural speed variations resulting from changes in the load on the shaft. Industrial equipment can have its speed modified or regulated through mechanical means, such as stepped pulleys, sets of change gears, variable speed friction clutch mechanisms, and other mechanical devices. Historically, this has been the initial step towards transitioning from fixed-speed to adjustable-speed drives. However, electrical speed control offers numerous economic and engineering advantages compared to mechanical speed control. The specific speed control needs of an industrial drive depend on its type. Some drives necessitate continuous speed variation across the entire range from zero to maximum speed, or within a certain portion of this range. In contrast, others may require two or more predetermined fixed speeds.

### 2.2 CLASSIFICATION OF DC MOTORS

DC motors can be categorized into three types based on the method used to excite their field windings[32]. The connections for the field windings of these three types are illustrated in Figure 2.1.

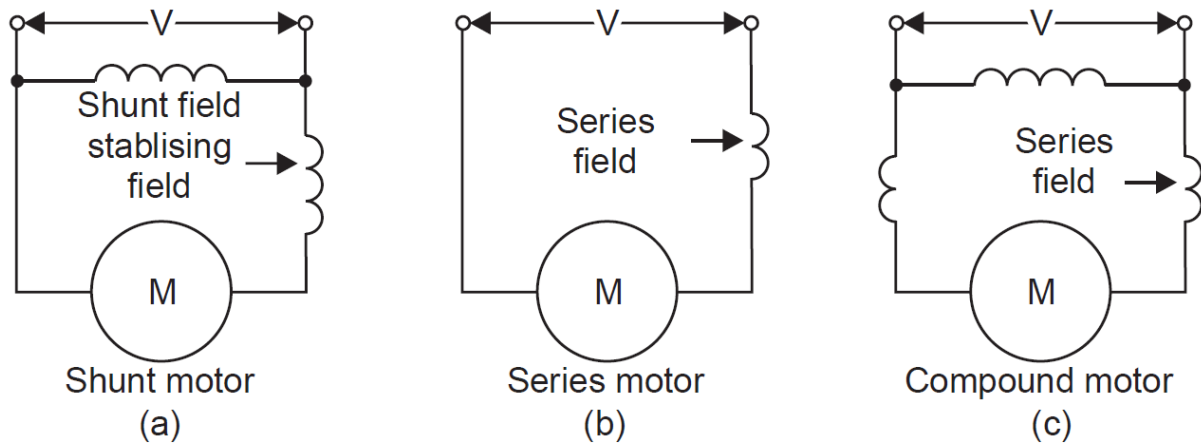


Figure 2.1: The three types of DC motors

## 2.3 SPEED CONTROL METHODS

The speed control of DC motors plays a crucial role in various industrial applications, robotics, and automation systems[14]. It enables precise regulation of motor speed, facilitating optimal performance, energy efficiency, and accurate positioning. Over the years, researchers and engineers have developed several speed control methods for DC motors to meet diverse application requirements[14].Where the most common are:

- Flux control method.
- Armature or Rheostatic control method.
- Voltage Control Method
- PWM TECHNIQUE.

### 2.3.1 FLUX CONTROL METHOD

It is well established that the speed of a DC motor, denoted by  $N(\text{rad/s})$ , is inversely proportional to the flux  $\phi$  ( $N \propto 1/\phi$ ). By reducing the flux, the speed can be increased, and vice versa. This method is commonly referred to as flux or field control. The flux of a DC motor can be adjusted by varying the current through the shunt field winding using a shunt field rheostat. Since the shunt field current ( $I_{sh}(A)$ ) is relatively small, the shunt field rheostat can be designed to handle only a small amount of current, resulting in a compact size for the rheostat[29].

This method is particularly efficient in non-interpolar machines. Using this approach, the speed can be increased by a ratio of 2:1. However, further weakening of the flux has a



detrimental effect on the motor's performance, particularly in terms of its commutation. Consequently, there is a limit to the maximum achievable speed using this method in machines equipped with interpoles. Typically, a common ratio of maximum to minimum speeds in such machines is 6:1.

The connection diagram for this type of speed control is illustrated in Figure 2.2.

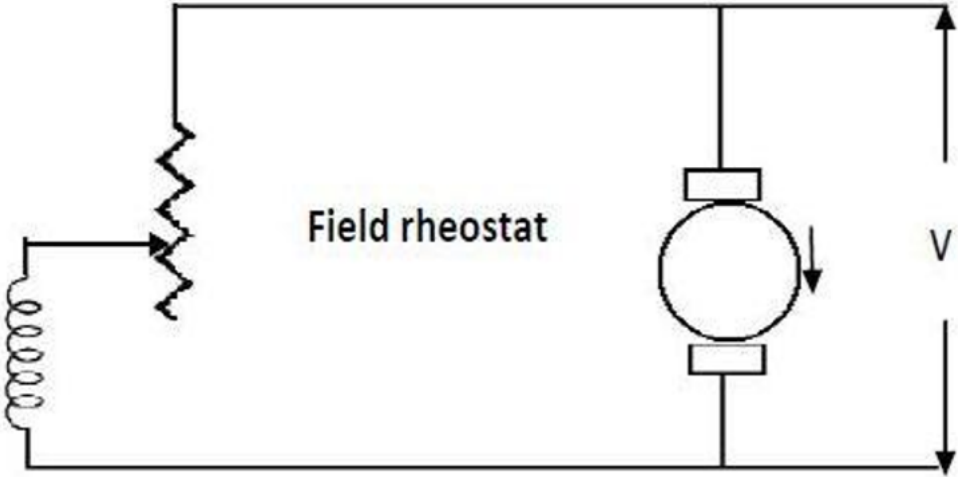


Figure 2.2: FLUX CONTROL METHOD

### 2.3.2 Armature or rheostat control method

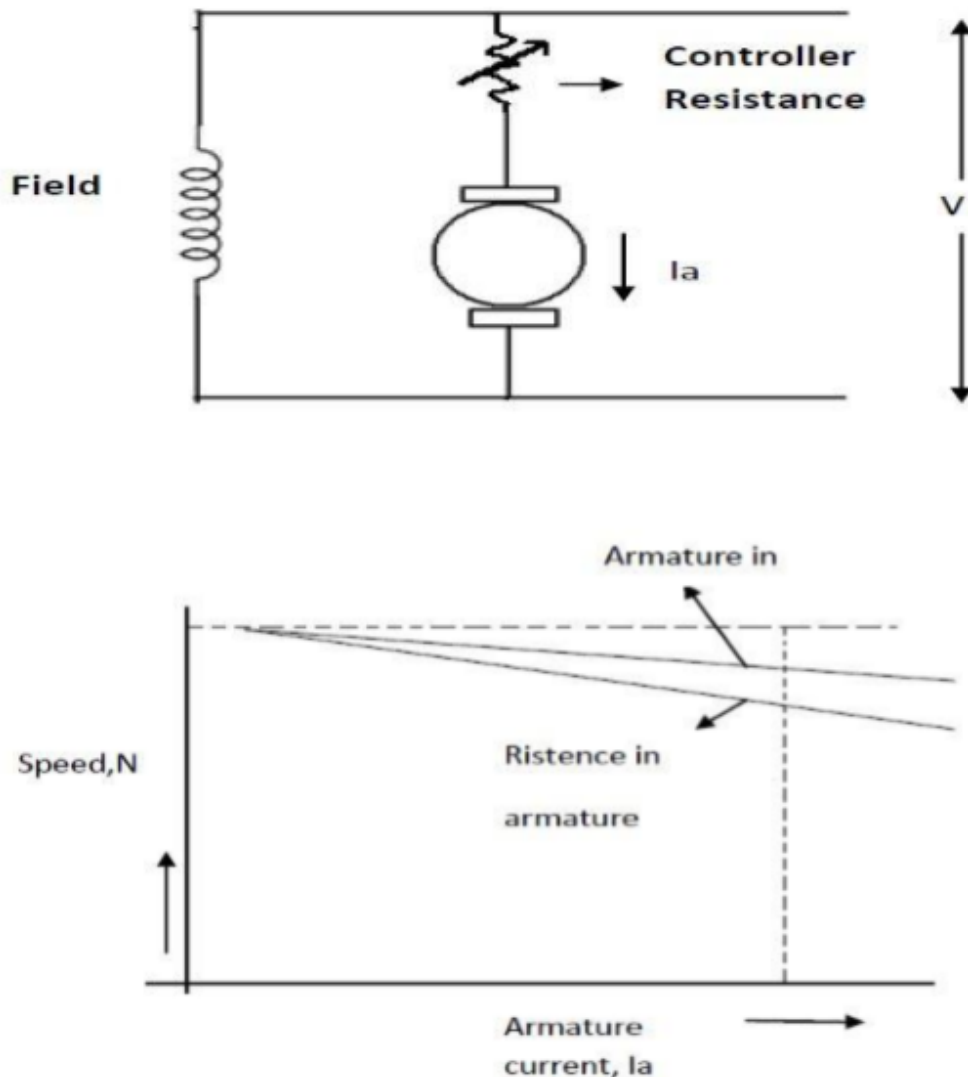


Figure 2.3: Rheostat control method

### 2.3.3 Rheostat control method

This method is employed when there is a need to achieve speeds lower than the no-load speed of a motor. Since the supply voltage remains constant, the variation in speed is achieved by adjusting the voltage across the motor's armature. This is accomplished by introducing a variable rheostat or controller resistance in series with the armature circuit, as depicted in Figure 2.3.

By increasing the controller resistance, the potential difference across the armature is reduced, leading to a decrease in the armature speed. In cases where the torque load remains constant, the speed is approximately proportional to the potential difference.

Examining the armature current characteristics illustrated in the figure 2.3, it is obvious that a higher resistance in the armature circuit results in a more substantial decrement in speed.

## 2.3.4 VOLTAGE CONTROL METHOD

### 2.3.4.1 MULTIPLE CONTROL VOLTAGE

The multiple control voltage method involves a fixed exciting voltage connected to the shunt field of the motor while the armature is connected to different voltages through suitable switchgear. This arrangement enables the armature to operate with varying voltages, and the motor speed is roughly proportional to these applied voltages. To achieve intermediate speeds, the shunt field can be adjusted accordingly.

### 2.3.4.2 WARD LEONARD SYSTEM

This system finds application in scenarios where a wide range of speed control, up to a ratio of 10:1, is required, with high sensitivity. It is commonly used in colliery winders, electric excavators, main drives in steel mills, blooming in paper mills, and similar applications. The system consists of two motors, M1 and M2, along with a generator, G. The field of motor M1 is connected permanently across the DC supply lines, allowing for speed control. Motor M2 is directly connected to the generator G.

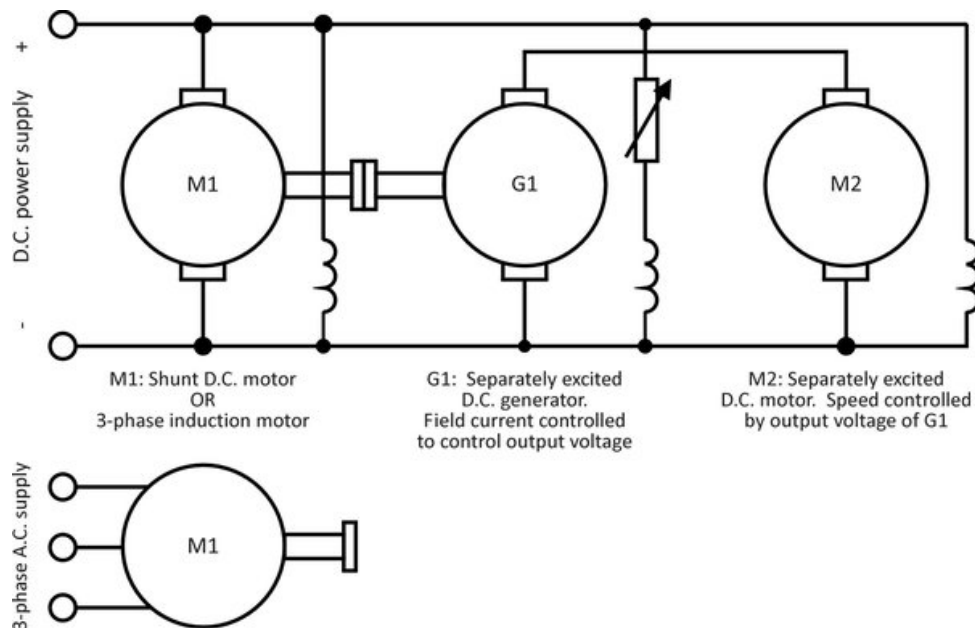


Figure 2.4: System structural arrangement of WARD LEONARD

The output voltage of the generator, which can be adjusted from zero to its maximum value using a field, is directly fed to the main motor M1. By reversing the field current of the generator through a reversing switch (RS), the generated voltage can be reversed, thus changing the direction of rotation of motor M1. It's important to note that the motor set always runs in the same direction.

To minimize fluctuations in power demand from the supply circuit, a flywheel is added to the system. The flywheel serves to reduce variations in power requirements.

The chief advantage of this system is its overall efficiency, particularly under optimal loads. It offers a wide range of speed control, ranging from maximum speed in one direction, through zero, to maximum speed in the opposite direction. Additionally, the system provides smooth acceleration for the motor.

### **2.3.5 PWM TECHNIQUE**

Pulse width modulation (PWM) control is a method that involves rapidly switching the power supplied to a motor on and off. It works by converting the DC voltage into a square wave signal, which alternates between fully on and off (zero), providing the motor with a series of power "kicks"[14].

The primary objective of using PWM is to achieve speed control and overcome the issue of poor starting performance in motors. In the context of motor speed control, PWM operates in a similar manner. Instead of providing a continuously varying voltage, the motor is supplied with a fixed voltage value. Afterward, the voltage is removed, and the motor continues to "coast". By repeating this cycle of voltage on/off with a varying duty cycle (the proportion of time the voltage is on), the speed of the motor can be controlled.

## **2.4 PWM TECHNIQUE**

Pulse-width modulation (PWM) or duty-cycle variation methods are commonly employed for controlling the speed of DC motors[14]. The duty cycle refers to the percentage of time the digital signal is in a "high" state compared to the total PWM period, which includes both the "high" and "low" states.

In Figure 2.5, pulses with a voltage of 5V are depicted, showcasing various duty cycles ranging from 0% to 100%. The average DC voltage value is dependent on the duty cycle. For instance, at 0% duty cycle, the average voltage is zero. At 20% duty cycle, the average

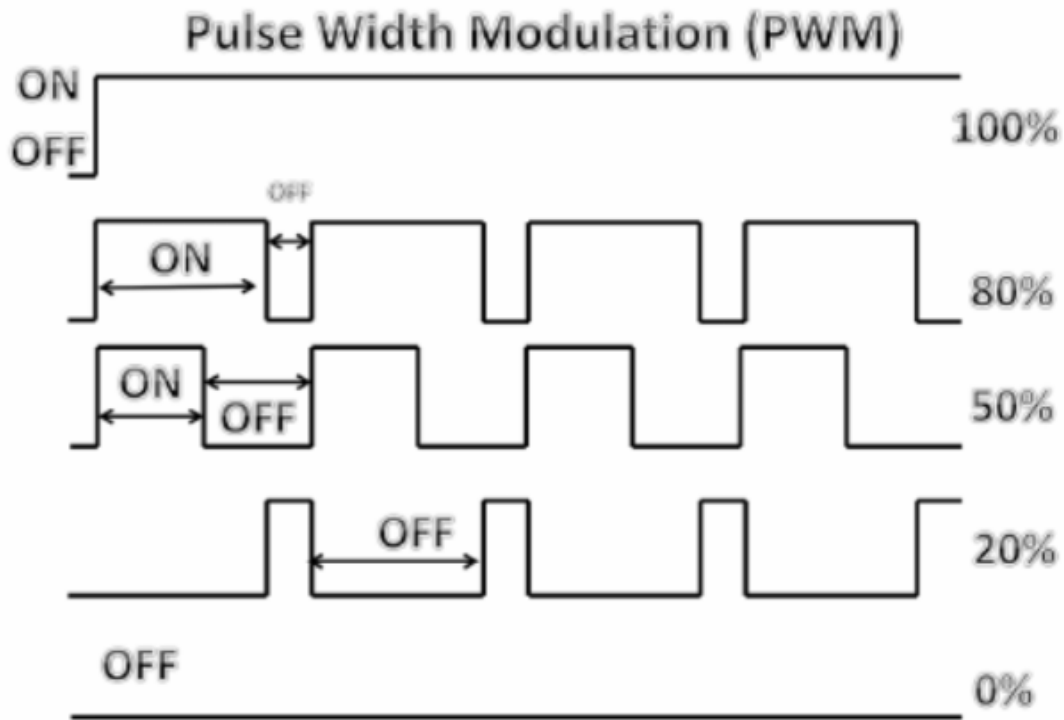


Figure 2.5: 5V Pulses with 0% through 100% duty cycle

voltage is 1.2V (20% of 5V). Similarly, at 50% duty cycle, the average voltage is 2.5V, and at 80% duty cycle, the average voltage is 4V. The maximum duty cycle is 100%, which results in a continuous DC waveform. By adjusting the pulse width, the average voltage across a DC motor can be varied, thus altering its speed??.

The average voltage in this context can be calculated using the following equation:

$$y' = D.Y_{max} + (1 - D).Y_{min} \quad (2.1)$$

However, in many cases, the minimum value is assumed to be zero, resulting in a simplified average voltage equation:

$$y' = D.Y_{max} \quad (2.2)$$

To illustrate a practical implementation of a speed controller for a miniature DC motor commonly found in devices like tape recorders and toys, refer to Figure 2.6, which depicts the circuit configuration.

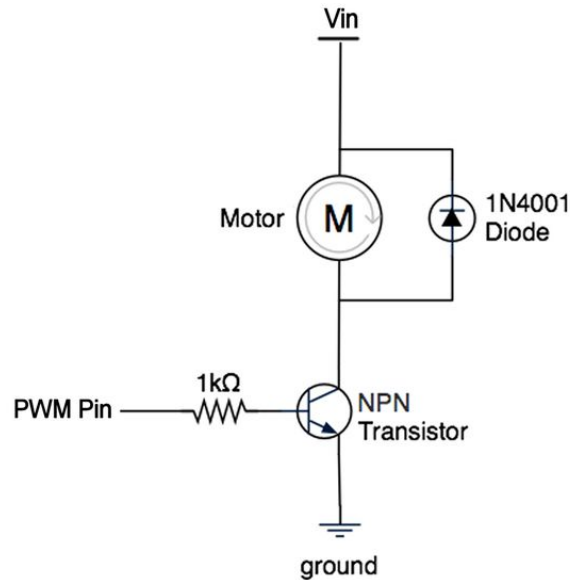


Figure 2.6: Simple speed controller

## 2.5 DC Motor speed control using PWM method

Pulse width modulation (PWM) is commonly used in DC motor control to address the issue of excessive heat dissipation in linear power amplifiers. Linear power amplifiers often generate significant heat, leading to the requirement of large heat sinks and sometimes forced cooling. By utilizing PWM amplifiers, this heat dissipation problem is greatly reduced due to their higher power conversion efficiency[14]. Another advantage of PWM control is that the input signal to the PWM driver can be directly derived from any digital system without the need for digital-to-analog (D/A) converters[9]. This simplifies the control system and eliminates the need for additional components. However, PWM power amplifiers have certain drawbacks. The desired signal is not directly translated into a voltage amplitude but rather into the time duration, or duty cycle, of a pulse. As a result, PWM is inherently a nonlinear operation. Nonetheless, in motor control applications, it is often reasonable to approximate PWM as a linear operation, effectively treating it as a pure gain. This approximation is based on the assumption that the average voltage is equal to the integral of the voltage waveform, which holds true in many motor control scenarios[14]. Thus:

$$V_S * T_{on} = V_{eq} * T \quad (2.3)$$

Where:

$V_S$ : the supply voltage (+12 volts)

$T_{on}$ : Pulse duration

$V_{eq}$ : the average or equivalent voltage seen by the motor

$T$ : Switching period (1/f)

In DC motor control using pulse width modulation (PWM), it is recommended to use a switching frequency of 300Hz. The switching frequency, denoted as 1/T, is determined based on the characteristics of the motor and amplifier.

The control variable in PWM is the duty cycle, which is defined as the on-time ( $T_{on}$ ) divided by the total period (T). It represents the percentage of time the PWM signal is in the "on" state during each period[14].

At each sampling time, the duty cycle needs to be recalculated based on the desired control input. This ensures the motor operates at the desired speed or torque.

### 2.5.1 Methods for Generating PWM Signals

PWM signals can be generated using various methods, including:

1. Analog Method: The analog method involves using analog circuitry to generate the PWM signals. This can be achieved using operational amplifiers, comparators, and other analog components such as Inductors, Transistors and Diodes. The input voltage is compared to a reference voltage, and the resulting error signal is used to control the duty cycle of the PWM signal[21].
2. Digital Method: The digital method utilizes digital circuits or microcontrollers to generate PWM signals. A digital system, such as a microcontroller, processes an input signal and converts it into a PWM signal by controlling the on/off times of the signal. This method offers flexibility and precise control over the PWM parameters[21].
3. Discrete IC: Dedicated integrated circuits such as PWM Controllers, PWM Generators Motor Driver ICs and designed specifically for generating PWM signals are available in the market. These ICs provide built-in features and functionalities for generating PWM signals with ease. They often offer additional features like adjustable frequency, duty cycle control, and protection mechanisms[21].

Each method has its own advantages and considerations. The choice of method depends on factors such as the complexity of the application, required flexibility, available resources, and cost constraints[21].

## 2.6 Control Arrangements for D.C. Drives

The commonly employed configuration, found in a wide range of drives ranging from small 0.5 kW drives to large industrial drives reaching several megawatts, is known as the two-loop control system. This setup consists of an inner feedback loop responsible for current control, regulating torque, and an outer loop that governs speed control. In cases where position control is required, an additional outer position loop is introduced. While we will initially focus on a two-loop scheme for a thyristor-based d.c. drive, it's important to note that the fundamental principles remain consistent in a chopper-fed drive [1]. Later on, we will explore simpler arrangements commonly utilized in low-cost small drives.

The following discussion focuses on analog control systems and aims to cover the aspects that are essential for users to comprehend. Once a drive has been commissioned, the user typically has access to only a few potentiometer adjustments or presets (in the case of digital control)[12]. While many of these adjustments are self-explanatory, such as maximum speed, minimum speed, acceleration and deceleration rates, there are some that may be less obvious, such as "current stability," "speed stability," and "IR comp." In the upcoming sections, we will provide explanations for these parameters to ensure a clear understanding.

To understand how a two-loop control scheme operates, let's consider how we would manually control the motor[30]. If we observed that the speed, as indicated by the tachogenerator, was below the desired target, our instinct would be to increase the current (and thus the torque) to achieve acceleration. This could be done by raising the armature voltage. However, we would need to proceed cautiously to avoid creating excessive current, as there is a delicate balance between the back electromotive force (E) and the applied voltage (V). Throughout this process, we would keep a close eye on the ammeter to prevent any damage to the thyristor stack. As the speed approached the target, we would gradually reduce the current (by lowering the applied voltage) to prevent overshooting the set speed. These manual actions are performed automatically by the drive system, which we will now delve into[1].

Figure 2.7 shows a standard d.c. drive system with speed and current control. The main objective of the control system is to achieve speed control. The speed reference signal is provided as the input on the left side of the diagram, and the output is the actual speed of the motor, which is measured by the tachogenerator (TG) on the right side.



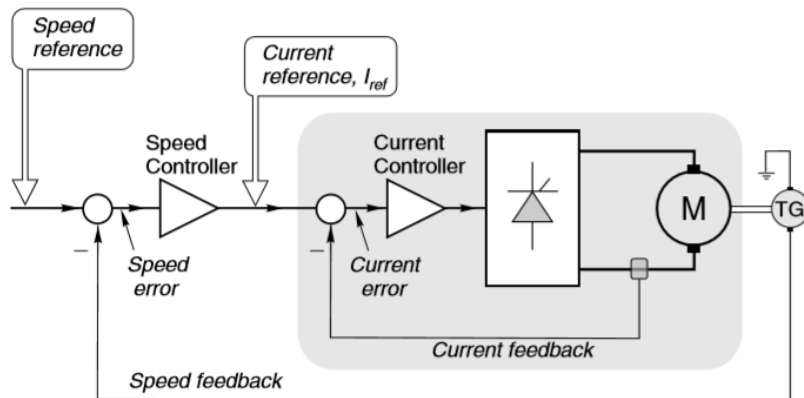


Figure 2.7: Schematic diagram of analogue controlled-speed drive with current and speed feedback control loops

In a closed-loop system like this, the overall performance greatly relies on the quality of the feedback signal, which in this case is the voltage proportional to the motor speed provided by the tachogenerator[23]. It is crucial to ensure that the tachogenerator is of high quality, meaning that its output voltage remains stable regardless of ambient temperature and is free from any ripple or fluctuations. Consequently, the cost of the tachogenerator often constitutes a significant portion of the total system cost.

We will take an overview of how the scheme operates first, and then examine the function of the two loops in more detail.

Let's examine the scenario where the motor is running at a set speed and the speed reference signal is suddenly increased. In this case, the set speed becomes higher than the actual speed, resulting in a speed error signal. This speed error is represented by the output of the left-hand summing junction in Figure 2.7. The presence of a speed error indicates the need for acceleration, which requires additional torque, i.e., more current. The speed controller, or speed-error amplifier, amplifies the speed error signal, which then becomes the reference or input signal for the inner control system. This inner control system is responsible for current control. As the current reference increases, the motor armature current also increases, providing extra torque and initiating acceleration[31]. As the speed gradually increases, the speed error diminishes, causing the current and torque to decrease accordingly, resulting in a smooth approach to the target speed[1].

We will now look in more detail at the inner (current -control) loop, as its correct operation is vital to ensure that the thyristors are protected against excessive overcurrents.

### 2.6.1 Current control

The closed-loop current controller, depicted by the shaded region in Figure 2.7, plays a central role in the drive system. Its purpose is to ensure that the actual motor current closely tracks the current reference signal ( $I_{ref}$ ) shown in Figure 2.7. To achieve this, the current loop compares a feedback signal of the actual motor current with the current reference signal. It then amplifies the difference between the two, known as the current error. The amplified current error signal, represented as an analog voltage, is used to control the firing angle ( $\alpha$ ) and consequently the output voltage of the converter. The current feedback signal can be obtained from either a DC current transformer, which provides an isolated analog voltage output, or from AC current transformer/rectifiers connected to the mains supply lines[1].

The task of comparing the reference (demand) current signal with the actual current signal and amplifying the resulting error signal is performed by the current error amplifier. By employing a high gain in the current error amplifier, the actual motor current will closely match the current reference signal, resulting in a small current error regardless of the motor's speed. In other words, the actual motor current is expected to continuously track the current reference signal, with the controller automatically adjusting the armature voltage to ensure that the current maintains the desired value irrespective of the motor's speed. It's important to note that no control system can achieve absolute perfection. However, it is customary for the current error amplifier to be of the proportional plus integral (PI) type, which means that under steady-state conditions, the actual and demanded currents will be precisely equal[4].

Preventing excessive converter currents is a critical aspect, as discussed earlier, and the current control loop plays a vital role in achieving this. The current control loop ensures that the motor current never surpasses the reference value. To achieve this, the magnitude of the current reference signal is limited through the use of a clamping circuit. By constraining the current reference signal, the motor current is effectively capped at the specified value.

In Figure 2.8, a small portion of Figure 2.7 is depicted, focusing on the characteristics of the speed controller within the shaded panel. It is evident from the diagram that for minor speed errors, the current reference increases proportionally to the speed error. This characteristic facilitates a linear system response, allowing for a smooth approach towards the desired speed. However, when the speed error exceeds a certain threshold, the output of the speed-error amplifier saturates[24]. As a result, the current reference no longer

increases beyond this point. By setting the maximum current reference to correspond to the full rated current of the system, it ensures that the motor and converter currents never exceed their designated limits, regardless of the magnitude of the speed error[4].

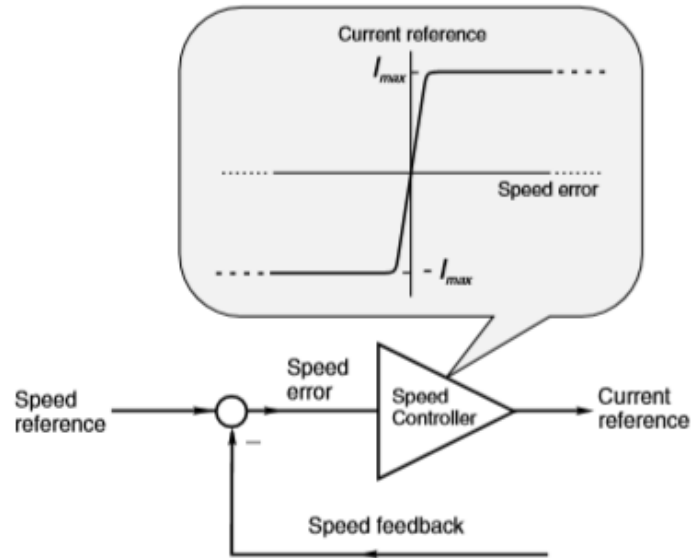


Figure 2.8: Detail showing characteristic of speed error amplifier

The implementation of 'electronic current limiting' is a crucial safety feature in any drive system. Its primary function is to protect the motor and other components by preventing excessive currents. In scenarios where the motor stalls abruptly, such as when the load seizes, and the back electromotive force (e.m.f.) drops significantly, the armature voltage is automatically reduced to a very low value. This reduction in voltage effectively restricts the current to a level that is within the maximum allowable limit. As a result, the electronic current limiting feature prevents damage to the motor and ensures the system operates within safe parameters. When setting up a drive system, one of the primary objectives is to establish a robust current control loop[15]. In this context, a "good" current loop refers to achieving two key characteristics. Firstly, the steady-state motor current should precisely match the current reference signal. Secondly, the response of the system to changes in the current reference should be rapid and well-damped.

To meet these requirements, the current-error amplifier incorporates an integral term, which ensures that the steady-state motor current aligns precisely with the current reference. Additionally, the choice of proportional gain and time constant in the amplifier plays a crucial role in achieving a fast and well-damped transient response to step changes in the current reference[20]. For the user's convenience and control, the "current stabil-

ity" adjustment is provided. This adjustment allows the user to fine-tune the transient response of the current loop, optimizing it according to specific requirements and preferences. By appropriately adjusting the current stability parameter, the user can achieve the desired balance between response speed and stability in the current control loop[15]. In the realm of terminology, it is worth noting that the current error amplifier is commonly referred to as either the "current controller" (as depicted in Figure 2.7) or the "current amplifier." While the former term is straightforward and accurately conveys its function, the latter term can be somewhat misleading. It is important to clarify that the term "current amplifier" does not imply that the motor current itself is being amplified. Instead, the term "current amplifier" is used to describe the function of the device, which amplifies the current error signal to generate the control signal that adjusts the converter's output voltage and, consequently, the motor current. The amplifier does not directly amplify the motor current, but rather acts as a controller to regulate and shape the current in response to the current reference signal[1].

## 2.6.2 Torque control

In applications where a specific torque is required regardless of the motor speed, such as in line tensioning systems, it is possible to eliminate the outer speed control loop. Instead, a direct current reference signal is fed into the current controller, typically through the "torque ref" terminal on the control board. Since torque is directly proportional to current, the current controller effectively functions as a torque controller. In some cases, an additional transient "inertia compensating" signal may be necessary to account for accelerating torque, which can be adjusted through a potentiometer or digital preset[1].

In the current control mode, the motor maintains a constant current at the set value, while the steady-state speed is determined by the load. For instance, if the torque reference signal is set at 50% and the motor is initially at rest, it will accelerate with a constant current equal to half of the rated value until the load torque matches the motor torque[29]. In the absence of a load, the motor will accelerate rapidly, with the applied voltage increasing to ensure it remains higher than the back electromotive force (e.m.f.) by the necessary amount to drive the specified current into the armature[3]. Eventually, the motor will reach a speed slightly above the normal "full" speed, at which point the converter's output voltage has reached its upper limit. Consequently, it becomes impossible to maintain the set current, and the motor speed will remain constant thereafter[29].

### 2.6.3 Speed control

In the drive system shown in Figure 2.7, the outer loop is responsible for speed control[9]. The speed feedback is obtained from a DC tachogenerator, and both the actual speed and the desired speed are inputted into the speed-error amplifier, which is often referred to as the speed amplifier or speed controller[14]. The speed amplifier compares the actual speed with the desired speed and amplifies any difference between the two. The output of the speed amplifier is then fed into the current loop as the input signal. Therefore, if the actual motor speed is lower than the desired speed, the speed amplifier will generate a current demand proportional to the speed error. This current demand will drive the motor to accelerate, aiming to minimize the speed error and bring the actual speed closer to the desired speed[14].

When the load on the motor increases, an immediate deceleration occurs, causing the speed-error signal to rise[9]. This increase in the speed error prompts the inner loop to demand more current. The additional torque generated by the increased current leads to acceleration, gradually reducing the speed error until a state of equilibrium is reached. At this equilibrium point, the current reference signal ( $I_{ref}$ ) produces a motor current that balances out the load torque.

In Figure 2.8, the speed controller is depicted as a simple proportional amplifier (P control). It is important to note that with a P controller, there will always be a finite speed error in order to maintain a steady-state value of  $I_{ref}$ . In other words, a P controller alone cannot achieve the exact target speed without any steady-state error. Although increasing the amplifier gain could reduce the speed error, it might also introduce instability to the system[28]. To eliminate the steady-state speed error, we can incorporate an integral (I) term into the speed controller in addition to the proportional (P) term. By using a PI controller, the output of the controller can have a non-zero value even when the input (speed error) is zero. This characteristic allows us to achieve zero steady-state error when implementing PI control.

The speed of the motor will be maintained at the level specified by the speed reference signal for all loads until the point where the maximum armature current is reached. If the load torque continues to increase beyond this point, the speed will start to decrease because the current loop will prevent any additional armature current from flowing. On the other hand, if the load tries to drive the speed above the set value, the motor current will be automatically reversed, causing the motor to act as a brake and regenerate power back to the mains. This ensures that the motor remains within its operational

limits and prevents the speed from exceeding the desired value. To highlight the crucial protective function of the inner loop, let's examine the scenario where the motor is at rest (assuming no load for simplicity) and we suddenly increase the speed reference from zero to its maximum value, creating a step demand for full speed. The speed error will be 100%, causing the output ( $I_{ref}$ ) from the speed error amplifier to immediately saturate at its maximum value. This maximum value is deliberately set to correspond to a demand for the maximum (rated) current in the motor[28]. Consequently, the motor current will reach its rated value, and the motor will accelerate with full torque. The speed and back electromotive force (E) will increase at a constant rate, and the applied voltage (V) will steadily rise to ensure that the difference (V - E) is sufficient to drive the rated current (I) through the armature resistance. In some drives, the current reference can temporarily reach 150% or even 200% of the rated value for a few seconds, providing a short-term torque boost. This feature, known as 'two-stage current limit,' is particularly useful for starting loads with high static friction.

The output of the speed amplifier will stay saturated until the actual speed approaches the target speed closely. During this time, the motor current will remain at its maximum value. Only when the speed is within a few percent of the target speed will the speed-error amplifier come out of saturation. From that point onward, as the speed continues to increase and the speed error decreases, the output of the speed-error amplifier will fall below the clamped level. At this stage, speed control enters a linear regime where the corrective current (and consequently the torque) is proportional to the speed error. This allows for a smooth and gradual approach to the final speed, ensuring a precise and controlled operation.

An effective speed controller should achieve zero steady-state error and exhibit a well-damped response to sudden changes in the desired speed. The integral term in the PI control handles the elimination of steady-state error, while the transient response is influenced by the proportional gain and time constant settings. To optimize the transient speed response, the "speed stability" potentiometer is made available for user adjustment. This allows the user to fine-tune the system and achieve the desired balance between speed accuracy and response time.

It is important to highlight that achieving a satisfactory transient response is generally easier with a regenerative drive. A regenerative drive can provide negative current (braking torque) to counteract any overshooting of the desired speed by the motor. On the other hand, a non-regenerative drive lacks the ability to supply negative current unless

it is equipped with reversing contactors. Therefore, if the speed overshoots the target in a non-regenerative drive, the only option is to reduce the armature current to zero and allow the motor to naturally decelerate. This approach is not ideal, and it is crucial to carefully configure the controller settings to avoid overshooting the target speed as much as possible[28].

In the event of a loss of feedback signal during system operation, as is the case with any closed-loop system, issues can arise. If the tachogenerator feedback becomes disconnected, the speed amplifier would quickly saturate, resulting in the application of full torque. Consequently, the speed would increase until the converter output voltage reaches its maximum limit. To prevent such situations, many drives are equipped with tacho-loss detection circuitry. Additionally, some systems automatically switch to armature voltage feedback as a backup in the event of tacho failure, ensuring continuous operation and control[1].

Drives that utilize weld weakening to expand the speed range are designed to automatically control both the armature voltage and weld current when operating above the base speed. In these systems, the weld current is maintained at its maximum value until the armature voltage reaches approximately 95% of the rated value[28]. When a higher speed is required, the additional armature voltage is accompanied by a simultaneous decrease in the weld current. This reduction is carefully managed so that when the armature voltage reaches 100%, the weld current is at its minimum safe level. This technique is commonly referred to as "spillover weld weakening." By adjusting the armature voltage and weld current in this manner, the drive system is able to extend its speed range effectively[1].

## **2.7 Simulation of PWM Techniques for DC Motor Speed Control**

In this section, we present a simulation-based study of Pulse Width Modulation (PWM) techniques for controlling the speed of a DC motor. PWM has emerged as a widely used method in motor control applications due to its ability to regulate motor speed with high precision and efficiency. By varying the duty cycle of the PWM signal, the average voltage applied to the motor can be adjusted, resulting in speed control.

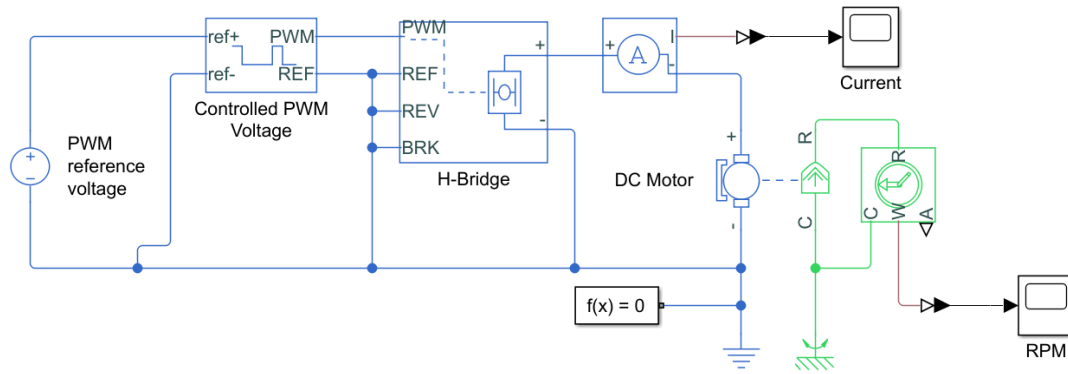


Figure 2.9: Simulink diagram of DC motor controlled by PWM

### 2.7.1 Controlled PWM Voltage

The "Controlled PWM Voltage" block is a fundamental component within our simulation model that plays a crucial role in generating pulse-width modulated (PWM) voltage signals. This block allows us to precisely control the average voltage applied to the DC motor, thereby enabling accurate speed regulation.

By utilizing PWM, we can effectively vary the duty cycle of the voltage waveform, which in turn alters the average voltage supplied to the motor. This modulation technique allows us to achieve fine-grained control over the motor's speed, enabling smooth and precise adjustments.

**The block has two modeling variants :**

**Electrical input ports:**The block calculates the duty cycle based on the reference voltage across its ref+ and ref- ports.

**PS input:**Specify the duty cycle value directly by using an input physical signal port

For the Electrical input ports variant of the block, the demanded duty cycle is:

$$100 \cdot \frac{V_{ref} - V_{min}}{V_{max} - V_{min}} \quad (2.4)$$

Where:

- $V_{ref}$  is the reference voltage across the ref+ and ref- ports.
- $V_{min}$  is the minimum reference voltage.
- $V_{max}$  is the maximum reference voltage.

The amplitude of the output voltage is determined by the value assigned to the "Output voltage amplitude" parameter. When the simulation starts, the pulse is initially set to a high level, unless the "Pulse delay time" parameter is greater than zero or the demanded



duty cycle is zero.

To introduce precise timing adjustments for turning on and off the pulse, the "Pulse delay time" and "Pulse width offset" parameters can be utilized. These parameters allow for the addition of a small delay during turn-on and a small advance during turn-off. This feature is particularly valuable when fine-tuning the switching times to minimize losses associated with the switching process.

By manipulating these parameters, the timing characteristics of the pulse can be precisely controlled, leading to improved accuracy and efficiency in generating the PWM signal. This level of control is especially beneficial for optimizing the performance of the PWM technique, aiming to minimize switching losses and achieve optimal control over the DC motor[17].

### **2.7.2 H-bridge Drive for a DC motor**

The control of direction and speed in a DC motor can be achieved through the implementation of an H-bridge drive. This configuration employs a combination of switches, typically transistors or MOSFETs, to govern the current flow within the motor windings, thereby enabling bidirectional control.

The H-bridge arrangement involves connecting the motor between the switches in a configuration resembling the letter "H." These switches are organized into two pairs of transistors or MOSFETs, with each pair responsible for controlling the current flow in one direction through the motor.

To regulate the motor, specific switching patterns are employed by toggling the switches on and off. By selectively activating the appropriate switches, the polarity of the voltage applied to the motor terminals can be altered, facilitating rotation in either the forward or reverse direction.

The following are the fundamental switch configurations for various motor states:

1. Forward Rotation: The switches connected to one side of the motor are turned on (conducting), while the switches connected to the opposite side are turned off (non-conducting). This setup permits the current to flow through the motor in one direction, resulting in forward rotation.
2. Reverse Rotation: The switches connected to the opposite side of the motor are turned on, while the other switches are turned off. This reversal of the current flow causes the motor to rotate in the opposite direction.
3. Coast/Stop: All switches are turned off, discontinuing the current flow through the

motor and bringing it to a halt. In this state, the motor gradually decelerates and stops due to its inherent inertia.

4. Brake: In certain H-bridge configurations, a braking mechanism can be implemented. By simultaneously activating both switches on one side (top and bottom), a short circuit is created across the motor terminals, generating a braking effect.

It is important to recognize that directly driving an H-bridge from a microcontroller or digital logic signals may prove inadequate due to the high voltage and current demands of the switches. Consequently, supplementary components such as gate drivers are frequently utilized to establish an interface between the control signals and the H-bridge switches. Gate drivers play a vital role in providing the required voltage and current levels to efficiently drive the switches. Additionally, gate drivers often incorporate protective features to prevent damage to the system[18].

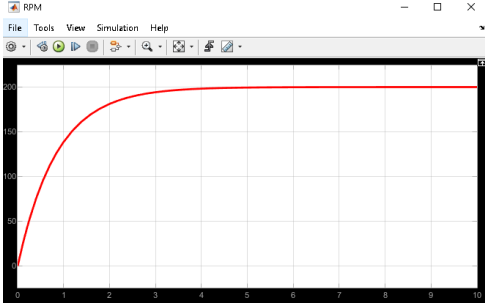
### 2.7.3 Simulation results

The outcomes of motor speed control through Pulse Width Modulation (PWM) are presented in the subsequent table2.1.

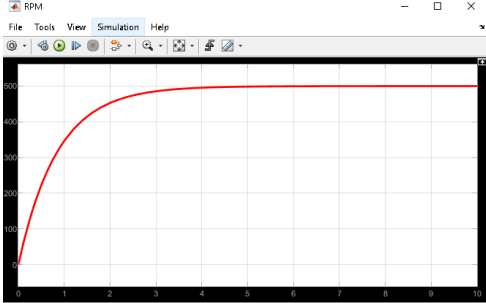
Duty cycle %	Output Voltage(V) $\in [0V5V]$	DC motor Speed (RPM)
10	0.5	200
25	1.25	500
50	2.5	1000
75	3.75	1500
100	5	2000

Table 2.1: Variation of DC motor speed across varying duty cycles

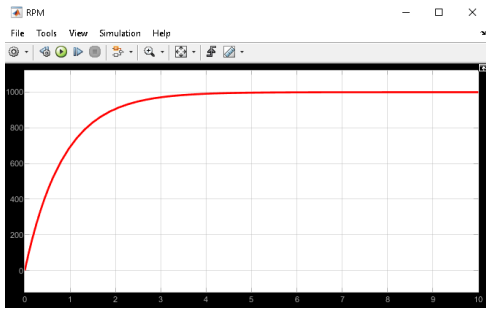
We provide here a comprehensive depiction of the graphical representations showcasing the speed characteristics of the DC motor corresponding to each duty cycle.



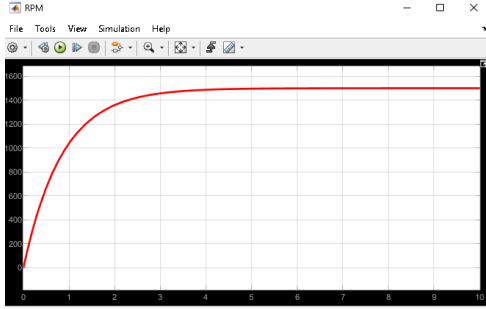
(a) DC motor speed for a 10% duty cycle



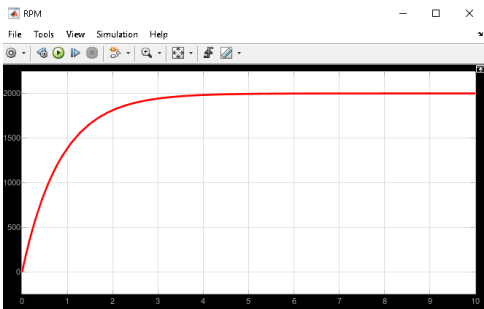
(b) DC motor speed for a 25% duty cycle



(c) DC motor speed for a 50% duty cycle



(d) DC motor speed for a 75% duty cycle



(e) DC motor speed for a 100% duty cycle

Figure 2.10: Variation of DC Motor Speed across Varying Duty Cycles

The utilization of Pulse Width Modulation (PWM) methodology presents a precise and discernible impact on the dynamic behavior of motor speed. The documented analysis clearly illustrates the consequential changes in motor speed resulting from the deliberate application of PWM. This empirical evidence serves to corroborate the assertion that PWM serves as an effective mechanism for achieving precise control over motor speed.

# Chapter 3

## Quantum PWM Algorithm

In this chapter we propose a new algorithm called **QPWM** or **Quantum PWM** that is inspired mainly by the classical PWM generator technique that controls power transfer from one electrical component to another by quickly switching between full power transfer and no power transfer, and which is based on comparing two signals, one is a reference signal, which represents the duty cycle, while the other is variable, which represents the carrier counter value.

The PWM generator block outputs either 1 when the duty cycle is greater than the carrier counter value, or 0 otherwise. We can set the period of each cycle by specifying the timer period  $T_{per}$ . We can change the initial output, or phase, of the PWM output by specifying one of three types of carrier counters:

- Up counter — The PWM output signal initializes at the start of the on cycle. Figure 3.1 shows the carrier counter signal and the corresponding PWM output[19].

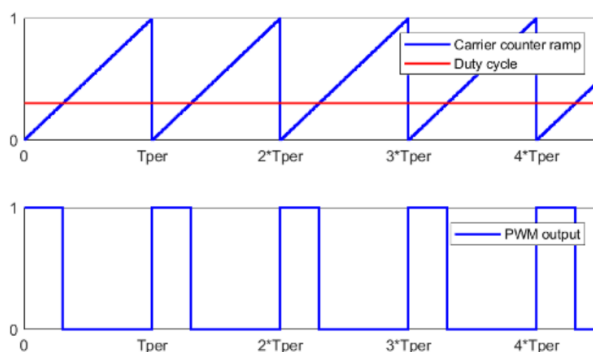


Figure 3.1: Up counter

- Down counter — The PWM output signal initializes at the start of the off cycle. Figure 3.2 shows the carrier counter signal and the corresponding PWM output[19].

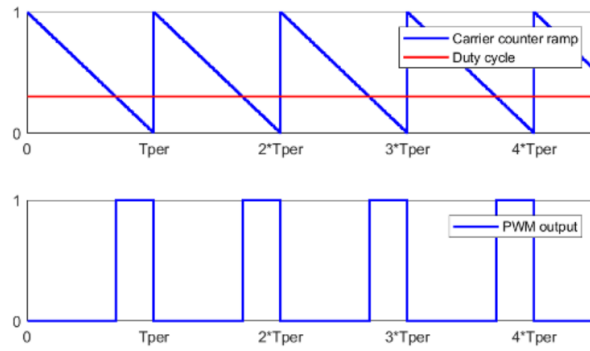


Figure 3.2: Down counter

- Up-down counter — The PWM output signal initializes halfway through the on cycle. Figure 3.3 shows the carrier counter signal and the corresponding PWM output[19].

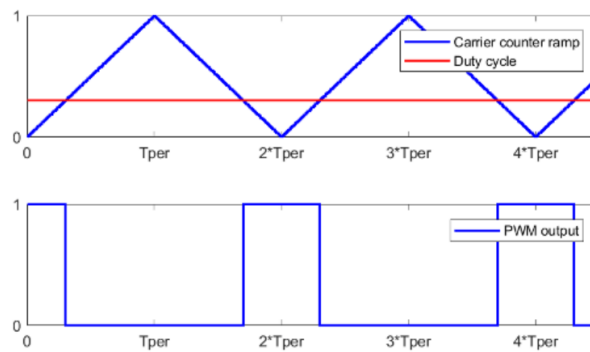


Figure 3.3: Up-down counter

Thus, for our project, we adopted a similar idea based on the comparison between two qubits, the first and the second, and the result of the comparison is at the circuit output in the third qubit, so initially our project is a circuit based on three qubits, but later we will make updates in it and explain it in detail. Since we talked about comparison, then we are going to realize two quantum comparators that we will be explained in this chapter.

### 3.1 The idea of a quantum comparator:

The idea of making a quantum comparator comes from the classical comparator that compares between two classical bits, but in the quantum version we are going to make

two comparators because we can't achieve our goal of getting a variable duty cycle in the range from 0% to one 100% using only one comparator, that's why we used a "greater or equal  $\geq$ " one and a "completely greater  $>$ " in order to get the whole domain.

### 3.1.1 Greater or equal $\geq$ comparator:

The truth table :

A	B	C
0	0	1
0	1	0
1	0	1
1	1	1

Table 3.1: Truth table of the the  $\geq$  comparator

The expression of the output:

$$C = A + \bar{B} \quad (3.1)$$

The classic logic circuit is illustrated in Figure 3.4:

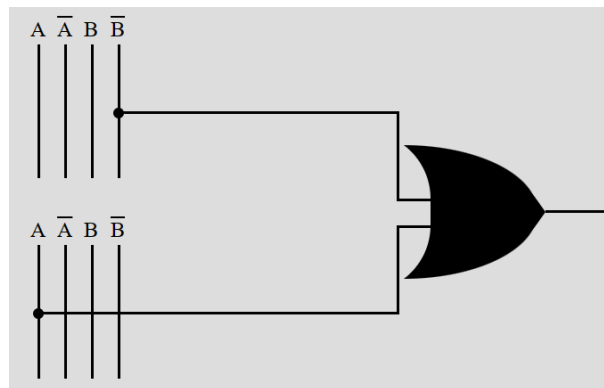
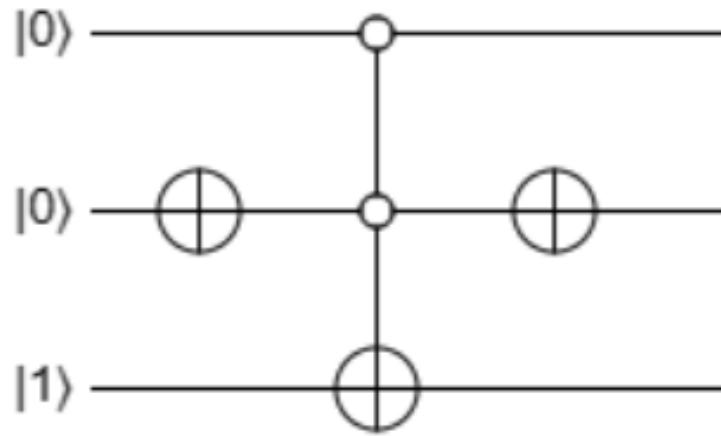


Figure 3.4: The classic logic circuit of  $\geq$  comparator

### 3.1.2 The equivalent quantum comparator:

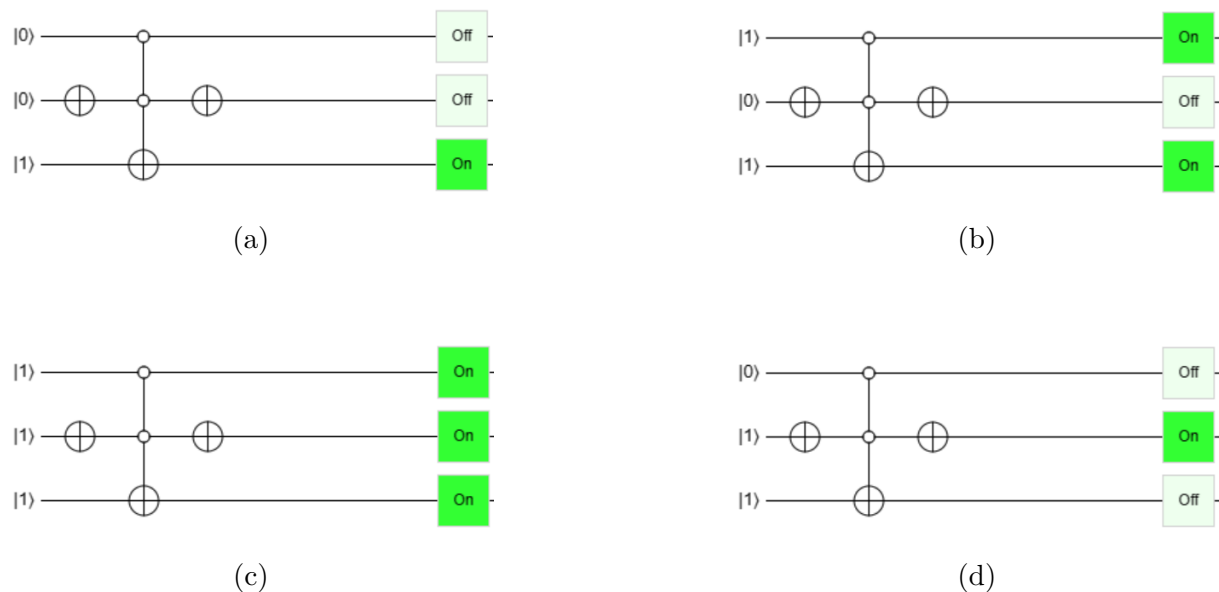
We know that the equivalent of the classical OR gate is the 'Anti-CCNOT'[33] such that the state of the 3<sup>rd</sup> qubit must be  $|1\rangle$  at the circuit input as illustrated in Figure 3.5.

Figure 3.5: The quantum logic circuit of  $\geq$  comparator

The NOT gate on the left will invert the initial state of the 2<sup>nd</sup> qubit, so we added another gate on the far right to restore the initial state of the 2<sup>nd</sup> qubit and allow us to observe it at the output. It should be noted that this rightmost gate has no effect on our circuit as it is placed at the end.

### 3.1.3 Operating principle:

For cases where each of the two input qubits is either in the state  $|0\rangle$  or  $|1\rangle$ , and not in superposition, the comparator will behave normally like the classical one and will give us the following four cases illustrated in Figure 3.6:

Figure 3.6: The four case of comparison of the  $\geq$  comparator



The comparator as defined operates as a classical binary comparator. However, when both qubits are in a superposition of states  $|0\rangle$  and  $|1\rangle$  ( $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ ), things become complicated, and the output becomes probabilistic. We take an example shown in Figure 3.7 to better explain the logic:

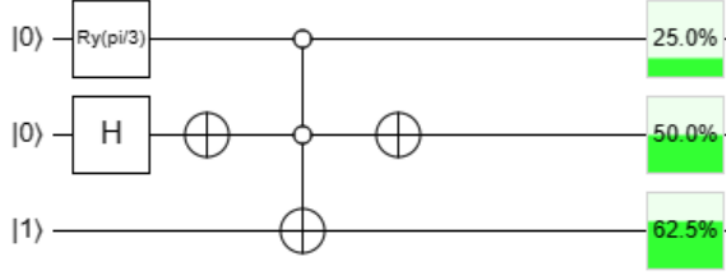


Figure 3.7: The quantum circuit to compare between two qubits in superposition

Upon initial examination of the output, we may suspect something is possibly wrong with the comparator circuit because we expect the output to be in the state  $|0\rangle$  since the probability of the first qubit  $|q_1\rangle$  is less than the second of  $|q_2\rangle$ . However, quantum logic works differently, and that's why we will demonstrate how things unfold.

For this example, we have two methods to calculate the probability of the output being  $|1\rangle$  and being  $|0\rangle$ :

- **The analytical method:**

Initially, we have the state  $|001\rangle$ . We perform a sequence of quantum operations as follows:

$$\begin{aligned}
 & |001\rangle \xrightarrow{Ry \otimes H \otimes I} \left( \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle \right) \left( \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) |1\rangle \\
 & \rightarrow \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |001\rangle + \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |011\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |101\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |111\rangle \\
 & \xrightarrow{I \otimes NOT \otimes I} \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |011\rangle + \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |001\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |111\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |101\rangle \\
 & \xrightarrow{Anti-CCNOT_{123}} \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |011\rangle + \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |000\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |111\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |101\rangle \\
 & \xrightarrow{I \otimes NOT \otimes I} \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |001\rangle + \frac{1}{\sqrt{2}} \cos\left(\frac{\theta}{2}\right) |010\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |101\rangle + \frac{1}{\sqrt{2}} \sin\left(\frac{\theta}{2}\right) |111\rangle \quad (3.2)
 \end{aligned}$$

Now, we calculate the probability for the output to be in the state  $|1\rangle$  and the state  $|0\rangle$  for a rotation angle of  $\theta = \frac{\pi}{3}$ .

$$P_{q_3}(|1\rangle) = \left( \frac{1}{\sqrt{2}} \cos\left(\frac{\pi}{6}\right) \right)^2 + 2 \left( \frac{1}{\sqrt{2}} \sin\left(\frac{\pi}{6}\right) \right)^2 = 0.625 \quad (3.3)$$

$$P_{q_3}(|0\rangle) = \left( \frac{1}{\sqrt{2}} \cos\left(\frac{\pi}{6}\right) \right)^2 = 0.375 \quad (3.4)$$

$$P_{q_3}(|1\rangle) + P_{q_3}(|0\rangle) = 1 \quad (3.5)$$

- **The deductive method:**

As  $(|q_1\rangle = \alpha|0\rangle + \beta|1\rangle)$  and  $(|q_2\rangle = \alpha|0\rangle + \beta|1\rangle)$ , all possibilities are opened, resulting in four possible scenarios. Among these scenarios, three are dedicated to the active output and one to the inactive output, as there is only one case where  $|q_3\rangle$  is inactive, namely when  $(|q_1\rangle = |0\rangle)$  and  $(|q_2\rangle = |1\rangle)$ . The different scenarios are summarized in the table 3.2:

$ q_1\rangle$	$ q_2\rangle$	$P( q_1\rangle)$	$P( q_2\rangle)$	$ q_3\rangle$	$P( q_3\rangle)$
$ 1\rangle$	$ 1\rangle$	0.25	0.5	$ 1\rangle$	$0.25 \times 0.5 = 0.125$
$ 1\rangle$	$ 0\rangle$	0.25	0.5	$ 1\rangle$	$0.25 \times 0.5 = 0.125$
$ 0\rangle$	$ 1\rangle$	0.75	0.5	$ 0\rangle$	$0.75 \times 0.5 = 0.375$
$ 0\rangle$	$ 0\rangle$	0.75	0.5	$ 1\rangle$	$0.75 \times 0.5 = 0.375$

Table 3.2: The table of probabilities of the example in figure 3.7

We sum up the probabilities  $|q_3\rangle$  and find:

$$P_{q_3}(|1\rangle) = 0.125 + 0.125 + 0.325 = 0.625 \quad (3.6)$$

$$P_{q_3}(|0\rangle) = 0.375 \quad (3.7)$$

Note that if we measure the output of  $q_3$ , it allows us to collapse the qubit to a specific state based on the probabilities associated with each coefficient. In our example, the  $q_3$  output is represented as  $|q_3\rangle = 0.375|0\rangle + 0.625|1\rangle$ , after the measurement, the output becomes  $|q_3\rangle = |1\rangle$ . Therefore, we will always have one of the two states, from this idea of switching between the two states, the QPWM technique was developed. To perform this switching, we add two rotation gates  $Ry(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$ , one gate has a constant angle that acts on  $|q_1\rangle$ , while the other gate has a variable rotation angle that acts on  $|q_2\rangle$ . As a result,  $|q_2\rangle$  alternates between the states  $|0\rangle$  and  $|1\rangle$  sinusoidally, passing through the intermediate states  $(|q_2\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)|1\rangle)$ , where  $\theta = 2\pi ft$  ( $f \in \mathbb{N}$ ) to cover all these states.

By manually varying the angle  $\theta$  of  $|q_1\rangle$ , we compare the different states, which are summarized in Table 3.3:

$ q_1\rangle$	$ q_2\rangle$	$P( q_1\rangle)$	$P( q_2\rangle)$	$ q_3\rangle$	$P( q_3\rangle)$
$ 1\rangle$	$ 1\rangle$	$\sin^2\left(\frac{\theta}{2}\right)$	$[0 \ 1]$	$ 1\rangle$	$\begin{bmatrix} 0 & \sin^2\left(\frac{\theta}{2}\right) \end{bmatrix}$
$ 1\rangle$	$ 0\rangle$	$\sin^2\left(\frac{\theta}{2}\right)$	$[1 \ 0]$	$ 1\rangle$	$\begin{bmatrix} \sin^2\left(\frac{\theta}{2}\right) & 0 \end{bmatrix}$
$ 0\rangle$	$ 1\rangle$	$\cos^2\left(\frac{\theta}{2}\right)$	$[0 \ 1]$	$ 0\rangle$	$\begin{bmatrix} 0 & \cos^2\left(\frac{\theta}{2}\right) \end{bmatrix}$
$ 0\rangle$	$ 0\rangle$	$\cos^2\left(\frac{\theta}{2}\right)$	$[1 \ 0]$	$ 1\rangle$	$\begin{bmatrix} \cos^2\left(\frac{\theta}{2}\right) & 0 \end{bmatrix}$

 Table 3.3: Table of probabilities of the  $\geq$  comparator

**Remark:**

$$|q_1\rangle = |0\rangle \xrightarrow{R_y} \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle \quad (3.8)$$

For that we have:

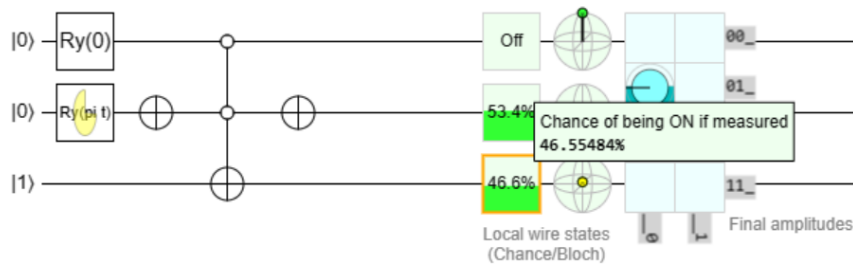
$$P(|q_1\rangle) = \cos^2\left(\frac{\theta}{2}\right) + \sin^2\left(\frac{\theta}{2}\right) \quad (3.9)$$

We obtain the following probabilities:

$$P_{q_3}(|1\rangle) \in \begin{bmatrix} 0 & \sin^2\left(\frac{\theta}{2}\right) \end{bmatrix} + \begin{bmatrix} \sin^2\left(\frac{\theta}{2}\right) & 0 \end{bmatrix} + \begin{bmatrix} \cos^2\left(\frac{\theta}{2}\right) & 0 \end{bmatrix} = \begin{bmatrix} 1 & \sin^2\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (3.10)$$

$$P_{q_3}(|0\rangle) \in \begin{bmatrix} 0 & \cos^2\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (3.11)$$

Table 3.3 summarizes what happens at the level of the comparator that can be found in the following link: [Greater or equal  \$\geq\$  comparator](#)


 Figure 3.8: The quantum logic circuit of  $\geq$  comparator

Now we replace  $\theta$  with ranges of values to see how the probabilities behave as summarized in Table 3.4:

$\theta$	$P( q_3\rangle)$ %	Duty cycle after measurement %
$[0 \quad \frac{\pi}{2}]$	[[100 0] [100 50]]	[50 100]
$[\frac{\pi}{2} \quad \pi]$	[[100 50] [100 100]]	100

 Table 3.4: Table of duty cycle for the  $\geq$  comparator

As we have seen in the table 3.4, for an angle  $\theta = 0$ , our probability interval is [100 0]. This means that after performing a measurement, we will have a switch between the states  $|0\rangle$  and  $|1\rangle$  with a duty cycle of 50%. If  $P(|1\rangle) \geq 50\%$ , we will have the state  $|1\rangle$  after the measurement, and vice versa. The duty cycle continues to increase until an angle of  $\frac{\pi}{2}$ , which gives us the probability interval of [100 50], this means that after the measurement, we will have only one state, which is  $|1\rangle$ , as  $P(|1\rangle)$  is always greater than 50%, implying a duty cycle of 100%. Beyond this value of  $\theta$ , the duty cycle remains at 100%. In conclusion, this comparator gives us a duty cycle interval of [50 100] for an angle interval of  $[0 \quad \frac{\pi}{2}]$ .

Since this comparator provides a duty cycle that varies between 50% and 100%, we will need another comparator that gives us the other interval range of 0 to 50%. We later discovered that the "strictly greater than  $>$ " comparator suits our needs.

### 3.1.4 Completely greater $>$ comparator:

The truth table :

A	B	C
0	0	0
0	1	0
1	0	1
1	1	0

 Table 3.5: Truth table of the the  $>$  comparator

The expression of the output:

$$C = A\bar{B} \tag{3.12}$$

The classic logic circuit is illustrated in Figure 3.9:

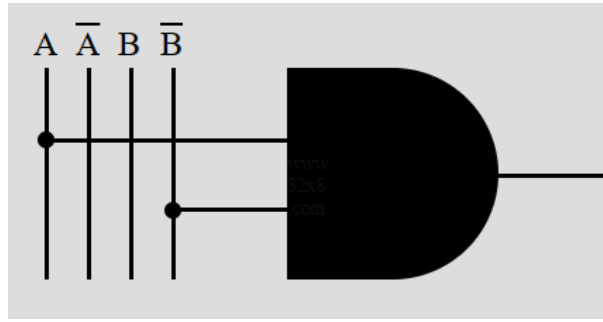


Figure 3.9: The classic logic circuit of  $>$  comparator

### 3.1.5 The equivalent quantum comparator:

We know that the equivalent of the classical AND gate is the "CCNOT" gate[33], where the state of the  $3^{rd}$  qubit must be in the state  $|0\rangle$  at the input of the circuit as illustrated in Figure 3.10.

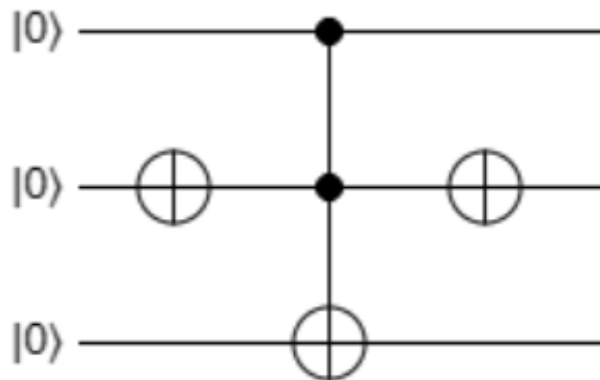


Figure 3.10: The quantum logic circuit of  $>$  comparator

We make the same observation as the NOT gate on the left, which will invert the initial state of the  $2^{nd}$  qubit. Therefore, we add another gate on the far right to restore the initial state of the  $2^{nd}$  qubit, allowing us to visualize it at the output. It's important to note that this rightmost gate has no effect on our circuit since it comes at the end of it.

### 3.1.6 Operating principle:

As long as  $|q_1\rangle$  and  $|q_2\rangle$  are just in the states  $|0\rangle$  or  $|1\rangle$ , the comparator will behave as a classical comparator, , and give us the following four cases illustrated in Figure 3.11:

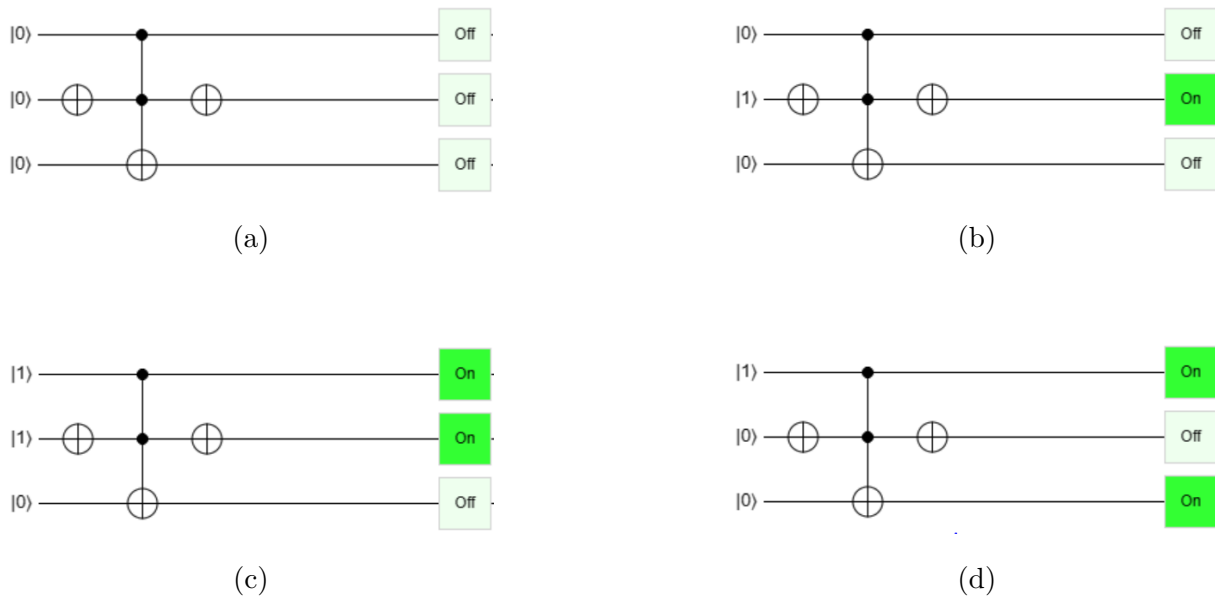


Figure 3.11: The four case of comparison of the  $>$  comparator

As the figure 3.11 shows, we have three possibilities to disable the output and only one to activate it.

**The case where  $|q_1\rangle$  and  $|q_2\rangle$  in superposition, i.e  $|q_{1,2}\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle$ :**

As the " $\geq$ " comparator:

- We apply a rotation matrix  $Ry(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$  to the input of  $q_1$ , manually varying between 0 and  $\pi$ , which corresponds to setting  $|q_1\rangle = |0\rangle$  and  $|q_1\rangle = |1\rangle$  in order to control the duty cycle.
- We apply the same rotation matrix  $Ry(\theta)$  to the input of  $q_2$ . In this case, we set  $\theta = 2.\pi.f.t$  ( $f \in \mathbb{N}$ ) where  $t$  represents time and  $f$  is the frequency of variation in the probabilities between the states  $|0\rangle$  and  $|1\rangle$ .

As done previously, we summarize our comparison in the table 3.6 with all possible values of  $\theta$  in the range  $[0 \ \pi]$  for  $|q_1\rangle$ :

$ q_1\rangle$	$ q_2\rangle$	$P( q_1\rangle)$	$P( q_2\rangle)$	$ q_3\rangle$	$P( q_3\rangle)$
$ 1\rangle$	$ 1\rangle$	$\sin^2\left(\frac{\theta}{2}\right)$	$[0 \ 1]$	$ 0\rangle$	$\begin{bmatrix} 0 & \sin^2\left(\frac{\theta}{2}\right) \end{bmatrix}$
$ 1\rangle$	$ 0\rangle$	$\sin^2\left(\frac{\theta}{2}\right)$	$[1 \ 0]$	$ 1\rangle$	$\begin{bmatrix} \sin^2\left(\frac{\theta}{2}\right) & 0 \end{bmatrix}$
$ 0\rangle$	$ 1\rangle$	$\cos^2\left(\frac{\theta}{2}\right)$	$[0 \ 1]$	$ 0\rangle$	$\begin{bmatrix} 0 & \cos^2\left(\frac{\theta}{2}\right) \end{bmatrix}$
$ 0\rangle$	$ 0\rangle$	$\cos^2\left(\frac{\theta}{2}\right)$	$[1 \ 0]$	$ 0\rangle$	$\begin{bmatrix} \cos^2\left(\frac{\theta}{2}\right) & 0 \end{bmatrix}$

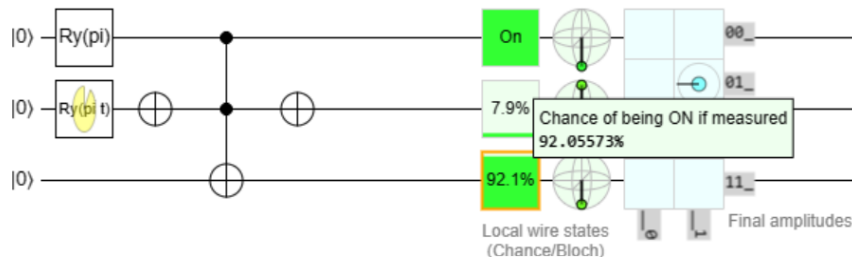
 Table 3.6: Table of probabilities of the  $\geq$  comparator

We obtain the following probabilities:

$$P_{q_3}(|1\rangle) \in \left[ \sin^2\left(\frac{\theta}{2}\right) \ 0 \right] \quad (3.13)$$

$$P_{q_3}(|0\rangle) \in \left[ 0 \ \sin^2\left(\frac{\theta}{2}\right) \right] + \left[ \cos^2\left(\frac{\theta}{2}\right) \ 0 \right] + \left[ 0 \ \cos^2\left(\frac{\theta}{2}\right) \right] = \left[ \cos^2\left(\frac{\theta}{2}\right) \ 1 \right] \quad (3.14)$$

The table 3.6 summarizes the behavior of the comparator that can be found in the following link: [Completely greater  \$>\$  comparator](#)


 Figure 3.12: The quantum logic circuit of  $>$  comparator

Now we replace  $\theta$  with ranges of values to observe the behavior of the probabilities as summarized in table 3.7:

$\theta$	$P( q_3\rangle) \%$	Duty cycle after measurement %
$\left[0 \ \frac{\pi}{2}\right]$	$\left[\begin{bmatrix} 0 & 0 \end{bmatrix} \ \begin{bmatrix} 50 & 0 \end{bmatrix}\right]$	0
$\left[\frac{\pi}{2} \ \pi\right]$	$\left[\begin{bmatrix} 50 & 0 \end{bmatrix} \ \begin{bmatrix} 100 & 0 \end{bmatrix}\right]$	$[0 \ 50]$

 Table 3.7: Table of duty cycle for the  $>$  comparator

As we observed in the table 3.7, this comparator provides us with a duty cycle range of  $[0, 50]$  for an angle interval of  $\theta \in \left[\frac{\pi}{2} \ \pi\right]$ .

Now we have two comparators that provide us with the entire range we need:

- The  $\geq$  comparator provides us with a duty cycle range of [50 100] for an angle interval of  $\theta \in [0 \ \frac{\pi}{2}]$ .
- The  $>$  comparator provides us with a duty cycle range of [0 50] for an angle interval of  $\theta \in [\frac{\pi}{2} \ \pi]$ .

To organize things better, we want to shift the angle intervals as follows:

- $[0 \ \frac{\pi}{2}] \Rightarrow [0 \ 50]$
- $[\frac{\pi}{2} \ \pi] \Rightarrow [50 \ 100]$

To achieve this, we make modifications to the rotation matrices:

- For the  $\geq$  comparator, we take Ry rotation:  $Ry(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2} - \frac{\pi i}{4}\right) & -\sin\left(\frac{\theta}{2} - \frac{\pi i}{4}\right) \\ \sin\left(\frac{\theta}{2} - \frac{\pi i}{4}\right) & \cos\left(\frac{\theta}{2} - \frac{\pi i}{4}\right) \end{pmatrix}$
- For the  $>$  comparator, we take Ry rotation:  $Ry(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2} + \frac{\pi i}{4}\right) & -\sin\left(\frac{\theta}{2} + \frac{\pi i}{4}\right) \\ \sin\left(\frac{\theta}{2} + \frac{\pi i}{4}\right) & \cos\left(\frac{\theta}{2} + \frac{\pi i}{4}\right) \end{pmatrix}$

Note that we only apply these modifications to the rotation matrices that act on  $|q_1\rangle$ , whether it is for the first comparator or the second.

## 3.2 Quantum Circuit Programming in MATLAB Simulink:

After completing the comparator circuit, we will use it to control the speed of a DC motor, and therefore we need MATLAB for our simulations. However, we encountered a problem, which is that MATLAB does not support quantum circuits except in the 2023 version, which we do not have yet. As a solution, we programmed all the gates, which are essentially matrices.



### 3.2.1 The $>$ comparator:

It consists of three stages as the figure 3.13 shows:

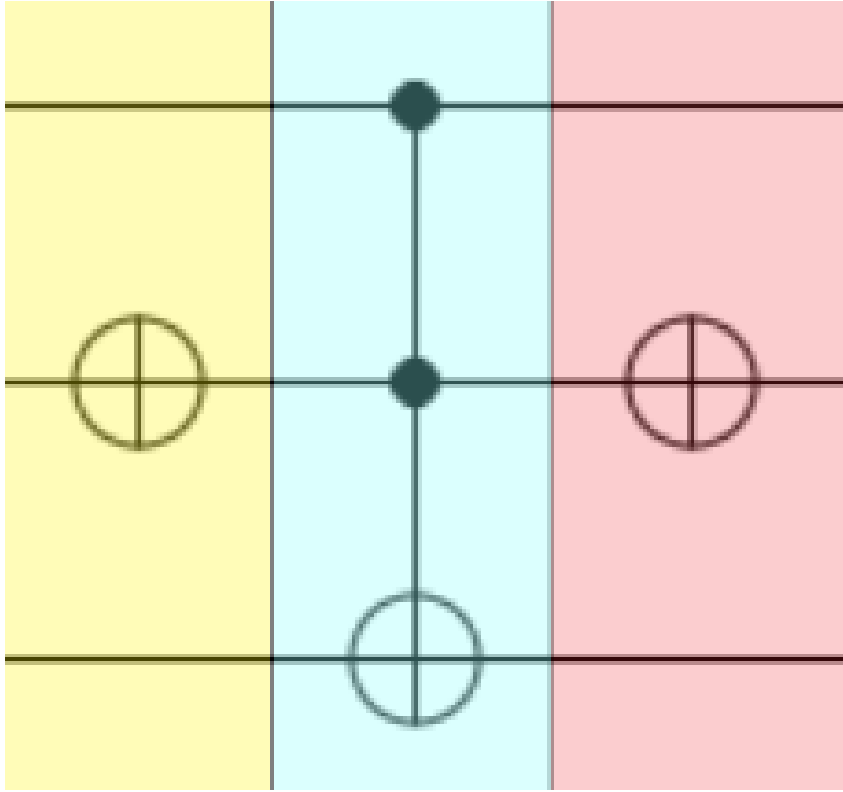


Figure 3.13: The quantum logic circuit of the  $>$  comparator

- Stage 1:

$$U_1 = I \otimes X \otimes I$$

$$U_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (3.15)$$

$$U_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (3.16)$$

- Stage 2:

We already know the matrix for the CCNOT gate, which is in the form:

$$U_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.17)$$

- Stage 3:

$$U_3 = U_1 \quad (3.18)$$

**The comparison matrix:**

$$COMP_{ss} = U_3 \cdot U_2 \cdot U_1$$

$$COMP_{ss} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.19)$$

### 3.2.2 The $\geq$ comparator:

Following the same previous steps, we find the following comparison matrix 3.20:

$$COMP_s = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.20)$$

### 3.2.3 The Inputs:

- The  $>$  comparator:

As mentioned earlier, our inputs are rotation matrices multiplied by the initial states of  $|q_1\rangle$ ,  $|q_2\rangle$ , and  $|q_3\rangle$  as the figure 3.14 illustrates:

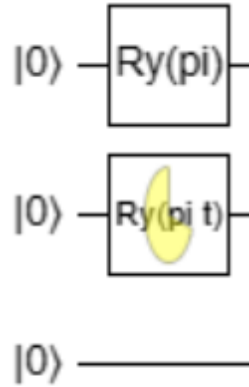


Figure 3.14: The inputs of the quantum comparator  $>$

$$U_{ss} = R_y \otimes R_y \otimes I = \tag{3.21}$$

$$\begin{pmatrix}
 \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 \\
 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\
 \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 \\
 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\
 \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 \\
 0 & \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\
 \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 \\
 0 & \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right)
 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \text{ we obtain the}$$

input 3.22:

$$E_{ss} = U_{ss} * |000\rangle = \begin{pmatrix} \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ 0 \end{pmatrix} \tag{3.22}$$

- **The  $\geq$  comparator:**

The input of this circuit is illustrated in the figure 3.15:

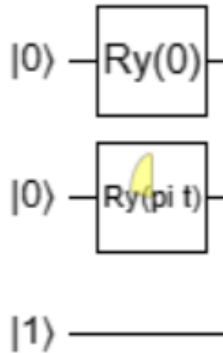


Figure 3.15: The inputs of the quantum comparator  $\geq$

When we calculate the stage which contain the rotation matrices we find the same matrix as before replacing  $+\frac{\pi}{2}$  by  $-\frac{\pi}{2}$ , we obtain:

$$U_s = \begin{pmatrix} \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 \\ 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 \\ 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 \\ \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 \\ 0 & \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & -\cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 \\ 0 & \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) & 0 & \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \end{pmatrix} \quad (3.23)$$

We multiply the matrix 3.23 by the vector of initial states  $|001\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ , we obtain the

input 3.24:

$$E_s = U_s * |001\rangle = \begin{pmatrix} 0 \\ \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \cos\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \sin\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \end{pmatrix} \quad (3.24)$$

### 3.2.4 The Outputs:

- **The  $>$  comparator:**

$$S_{ss} = COMP_{ss} * E_{ss} = \begin{pmatrix} \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ 0 \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \delta \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$S_{ss} = \alpha|000\rangle + \beta|010\rangle + \gamma|101\rangle + \delta|110\rangle \quad (3.25)$$

So, to measure the probability of  $|q_3\rangle = |1\rangle$ , which is our output, we take:

$$P_{ss-q_3}(|1\rangle) = \gamma^2 \times 100 \quad (3.26)$$

Where  $\gamma$  is the 6<sup>th</sup> coefficient of the output vector  $S_{ss}$ , we can note:

$$P_{ss-q_3}(|1\rangle) = S_{ss}(6)^2 \times 100 \quad (3.27)$$

- **The  $\geq$  comparator:**

$$S_s = COMP_s * E_s = \begin{pmatrix} 0 \\ \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right) \\ 0 \\ \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \sin\left(\frac{2\pi ft}{2}\right) \end{pmatrix} = \alpha' \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \beta' \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \gamma' \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \delta' \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$S_s = \alpha'|001\rangle + \beta'|010\rangle + \gamma'|101\rangle + \delta'|111'\rangle \quad (3.28)$$

So, to measure the probability of  $|q_3\rangle = |1\rangle$ , which is our output, we take:

$$P_{s-q_3}(|1\rangle) = (\alpha'^2 + \gamma'^2 + \delta'^2) \times 100 \quad (3.29)$$

Where  $\alpha'$ ,  $\gamma'$ , and  $\delta'$  are the 2<sup>nd</sup>, 6<sup>th</sup>, and 8<sup>th</sup> coefficients, respectively, of the output vector  $S_s$ , we can note:

$$P_{s-q_3}(|1\rangle) = (S_s(2)^2 + S_s(6)^2 + S_s(8)^2) \times 100 \quad (3.30)$$

### 3.2.5 The relation between the rotation angle $\theta_{q_1}$ and the duty cycle $D$ :

After completing the comparator, we ran into the problem of non-linearity i.e by increasing  $\theta_{q_1}$  the duty cycle increases with a non-linear relation, for example when we take  $\theta_{q_1} = \frac{\pi}{2}$  we obtain  $D = 50\%$  but when we take  $\theta_{q_1} = \frac{\pi}{4}$  we don't obtain what we want  $D = 25\%$ , so we can't control it as we want, for this we went to the solution to find the mathematical relationship that combine  $\theta_{q_1}$  and  $D$ , in other word we perform a change of the variable  $\theta_{q_1}$ :

- **The > comparator:**

We know that:

$$P_{ss-q_3}(|1\rangle) = \gamma^2 \times 100$$

Such as:

$$\gamma = \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \cdot \cos\left(\frac{2\pi ft}{2}\right)$$

To avoid manipulating in this complicated expression we use the other one 3.27

We use SIMULINK to visualize this probability for  $\theta = \frac{\pi}{4}$ , we obtain the graph shown in 3.16:

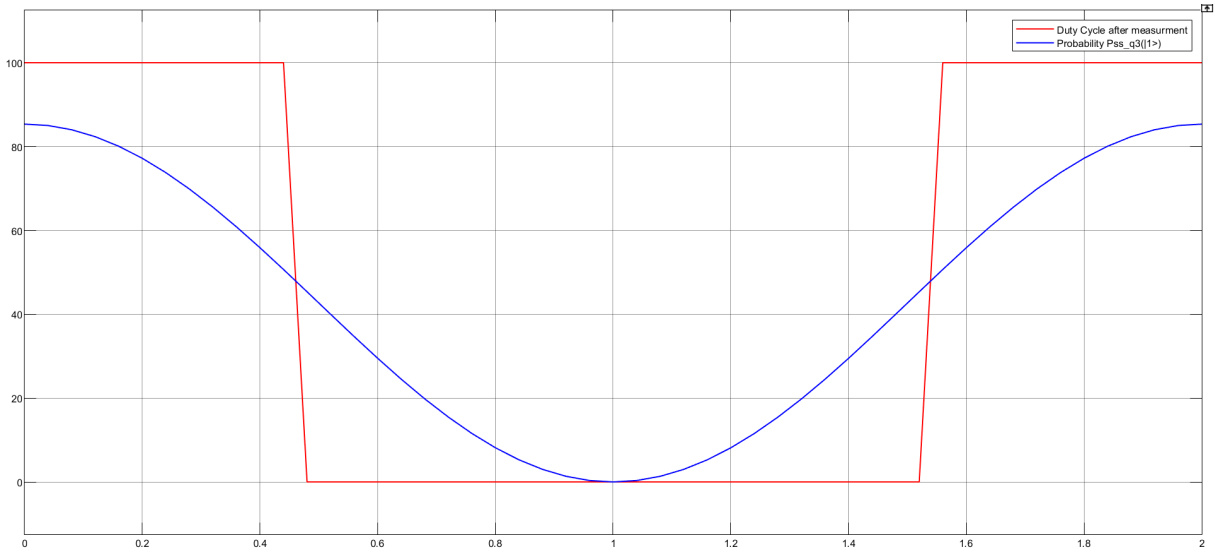


Figure 3.16: The probability  $P_{ss-q_3}(|1\rangle)$  before and after measurement

The equation of the probability curve can be written in the following form:

$$P_{q_3}(|1\rangle) = a \sin(2\pi ft + \phi) + b \quad (3.31)$$

In our case:  $\phi = \frac{\pi i}{2}$  and we have chosen  $f = \frac{1}{2}$  so that we can directly read the duty cycle from our graph on the time axis when the probability curve intersects the value 50% (according to the measurement where the switching between the state  $|1\rangle$  and  $|0\rangle$  takes places in the value of 50%), in other word for  $f = \frac{1}{2}$  and  $P_{ss-q_3} = 50$  we can write:  $D = t$  but in the general case, for any value of  $f$ , the expression for the duty cycle is given by:  $D = 2ft$ , in other the duty cycle is the coefficient multiplied by  $\pi$ .

We can extract the relation between  $\theta$  and  $D$  by finding the *max* of the previous expression of  $P$ , so according to the curve it's clear that the max is present at  $t = nT = \frac{n}{f}$  ( $n \in \mathbb{N}$ ) so:

$$\begin{aligned} \max(P_{ss-q_3}(|1\rangle)) &= a \sin(2n\pi + \frac{\pi}{2}) + b \\ &\Rightarrow \max = a + b \end{aligned}$$

Now we try to find a relation between  $a$  and  $b$ :

Note that the minimum of this curve is always fixed at the value of 0, as shown in the expression we demonstrated before for the  $>$  comparator 3.13, at the moment  $t = \frac{(2n+1)T}{2} = \frac{(2n+1)}{2f}$ , so remplacing that we find:

$$\begin{aligned} a \sin((2n+1)\pi + \frac{\pi}{2}) + b &= 0 \Rightarrow a = b \\ &\Rightarrow \max = 2a \end{aligned}$$



We extract  $a$  from the following expression:

$$a \sin(D\pi + \frac{\pi}{2}) + a = 50$$

We obtain:

$$a = \frac{50}{\sin(D\pi + \frac{\pi}{2}) + 1}$$

So we obtain the expression of the max value in terms of the duty cycle:

$$\max(P_{ss-q_3}(|1\rangle)) = \frac{100}{\sin(D\pi + \frac{\pi}{2}) + 1} \quad (3.32)$$

On the other hand, we have:

$$\max(P_{ss-q_3}(|1\rangle)) = 100 \sin^2\left(\frac{\theta}{2}\right)$$

according to 3.13

After shifting the angle:

$$\max(P_{ss-q_3}(|1\rangle)) = 100 \sin^2\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \quad (3.33)$$

From 3.32 et 3.33 we obtain:

$$\theta = \text{real} \left( 2 \cdot \arcsin \left( \frac{1}{\sqrt{\sin\left(\pi \cdot D + \frac{\pi}{2}\right) + 1}} \right) - \frac{\pi}{2} \right) \quad (3.34)$$

Where  $D \in [0 \quad \frac{1}{2}]$ .

To avoid having  $\theta$  as a complex number, we used the *real* function.

- **The  $\geq$  comparator:**

We follow the same steps as the previous one.

We use SIMULINK to visualize 3.30 for  $\theta = \frac{2\pi}{3}$ , we obtain the graph shown in 3.17:

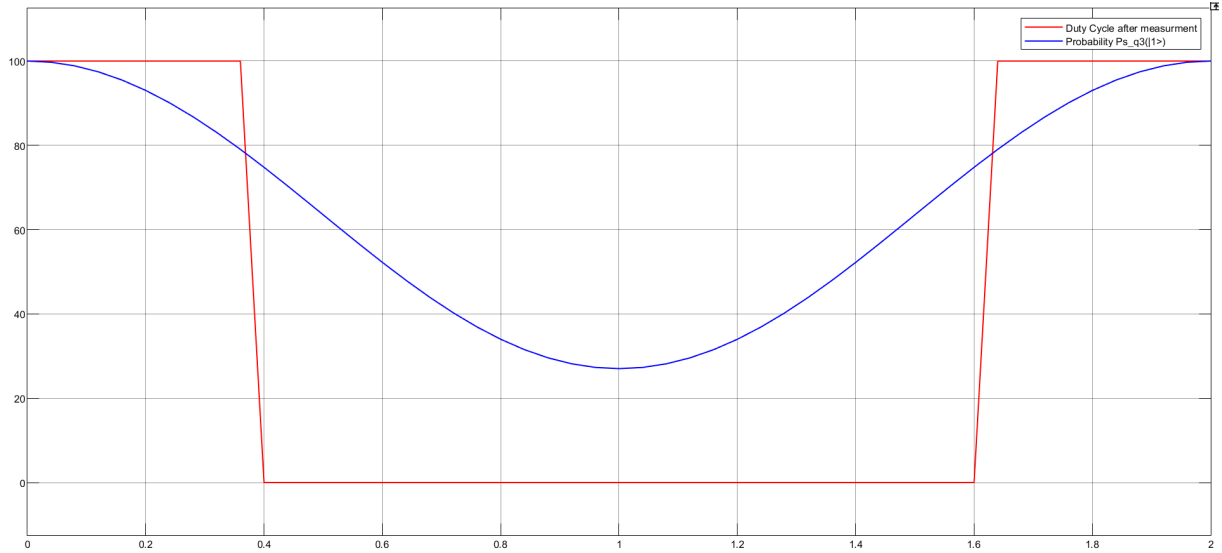


Figure 3.17: The probability  $P_{s-q_3}(|1\rangle)$  before and after measurement

This time, we look for the minimum of the function 3.31 because in this case the min is variable and the max is fixed at 100 according to 3.10.

Looking at the curve we find the min present at  $t = \frac{(2n+1)T}{2} = \frac{(2n+1)}{2f}$  so:

$$\begin{aligned} \min(P_{s-q_3}(|1\rangle)) &= a \sin\left((2n+1)\pi + \frac{\pi}{2}\right) + b \\ &\Rightarrow \min = -a + b \end{aligned}$$

Now it is clear that we will find the relation between  $a$  and  $b$  at  $t = nT = \frac{n}{f}$  where the max is fixed at the value 100:

$$\begin{aligned} a \sin\left(2n\pi + \frac{\pi}{2}\right) + b &= 100 \Rightarrow b = 100 - a \\ &\Rightarrow \min = -2a + 100 \end{aligned}$$

We extract  $a$  from the following expression:

$$a \sin\left(D\pi + \frac{\pi}{2}\right) - b + 100 = 50$$

We obtain:

$$a = \frac{-50}{\sin\left(D\pi + \frac{\pi}{2}\right) - 1}$$

So we obtain the expression of the min value in terms of the duty cycle:

$$\min(P_{s-q_3}(|1\rangle)) = \frac{100}{\sin(D\pi + \frac{\pi}{2}) - 1} + 100 \quad (3.35)$$

On the other hand, we have:

$$\min(P_{s-q_3}(|1\rangle)) = 100 \sin^2\left(\frac{\theta}{2}\right)$$

according to 3.10

After shifting the angle:

$$\min(P_{s-q_3}(|1\rangle)) = 100 \sin^2\left(\frac{\theta}{2} - \frac{\pi}{4}\right) \quad (3.36)$$

From 3.35 et 3.36 we obtain:

$$\theta = \text{real} \left( 2 \cdot \arcsin \left( \sqrt{\frac{1}{\sin(\pi \cdot D + \frac{\pi}{2}) - 1} + 1} \right) + \frac{\pi}{2} \right) \quad (3.37)$$

Where  $D \in [\frac{1}{2} \quad 1]$ .

Now we will program all these steps in MATLAB Simulink as shown in the figure 3.18:

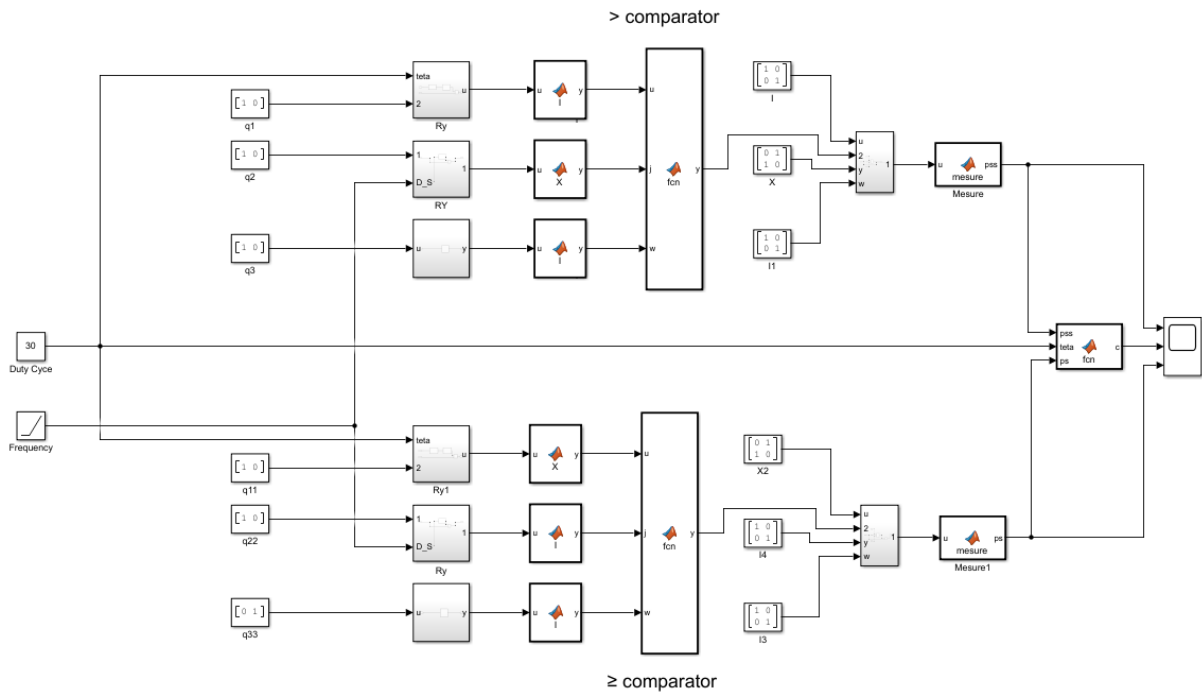


Figure 3.18: The QPWM algorithme

We notice that initially we implemented the two comparators separately, and then we connected them with an "if" condition as follows:

- If  $D \in [0 \ 50]$ , we work with the comparator " $>$ ".
- If  $D \in [50 \ 100]$ , we work with the comparator " $\geq$ ".

Also we have the ramp bloc named "Frequency" which aims to vary the frequency illustrated in Figure 3.19:

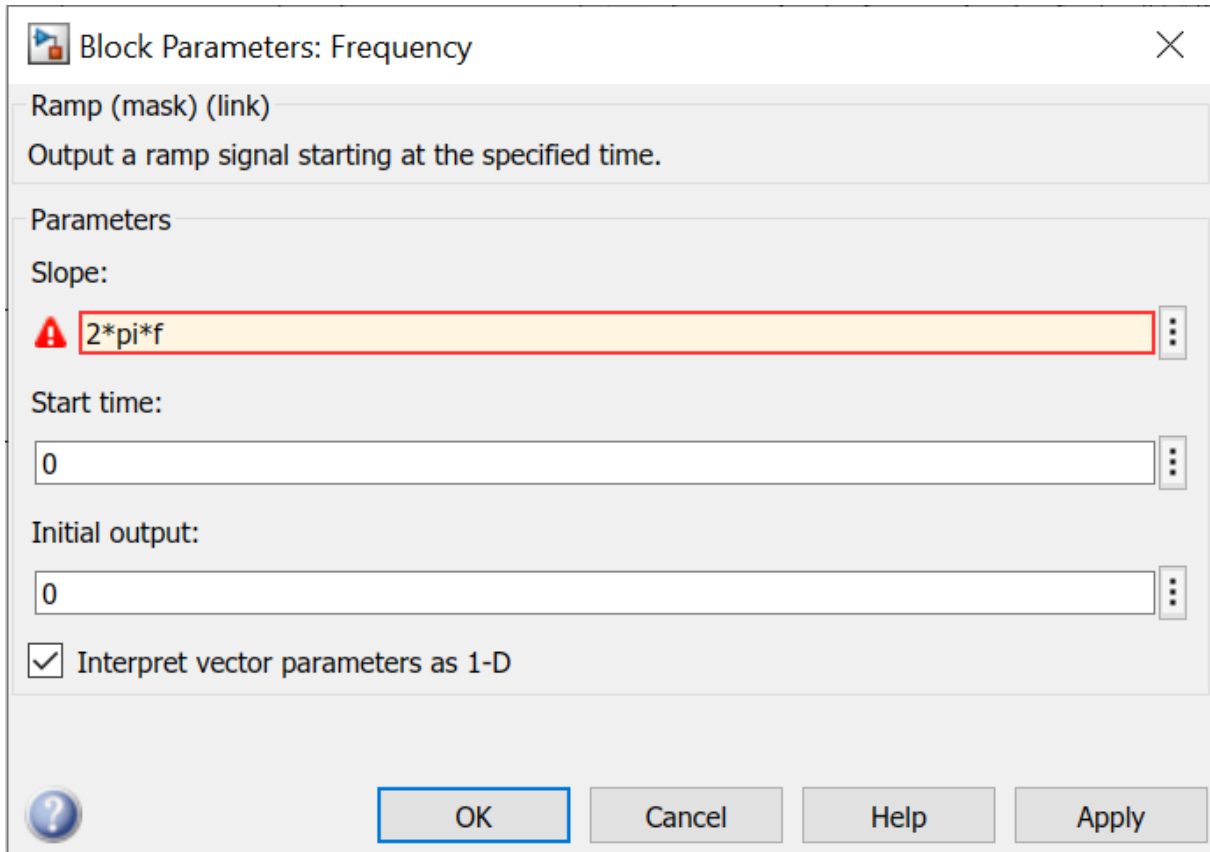


Figure 3.19: Ramp bloc

Here we have some simulations for some values for the duty cycle for  $f = \frac{1}{2}$ :

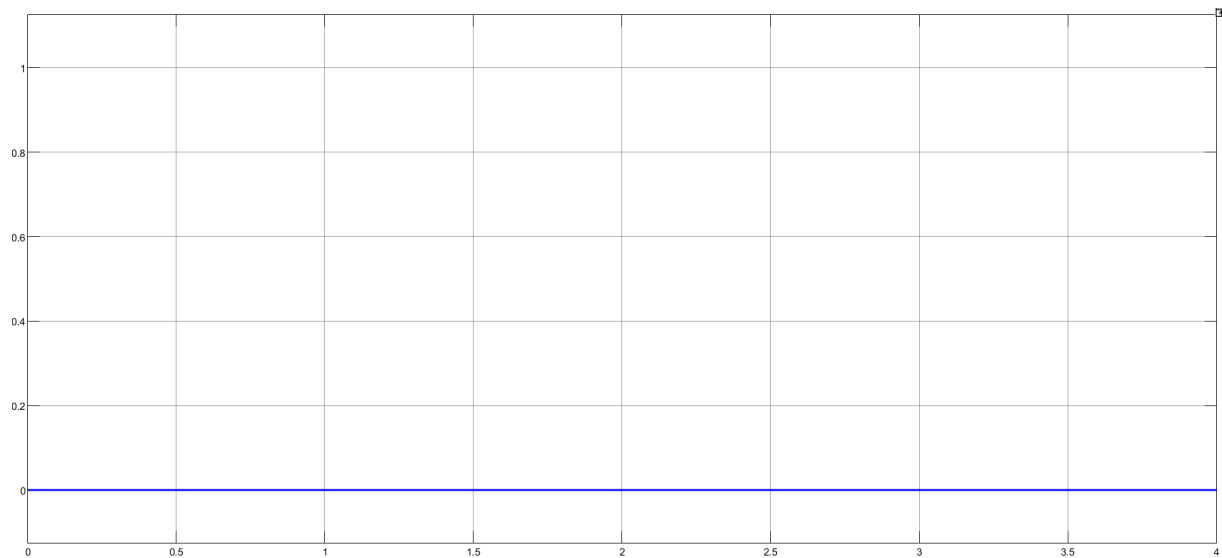


Figure 3.20: *Duty cycle*  $D = 0\%$

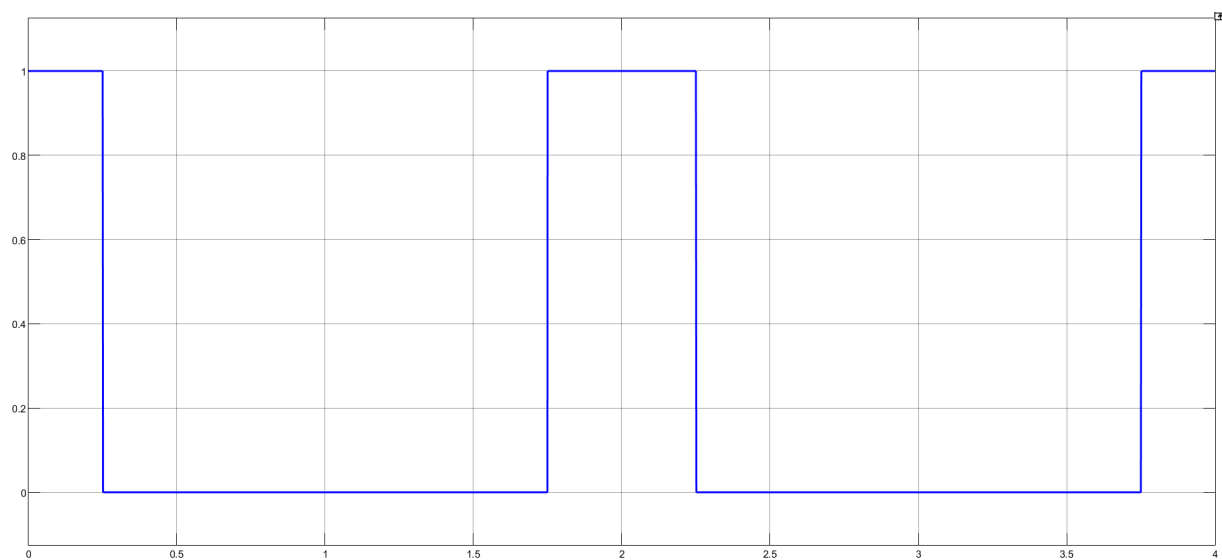


Figure 3.21: *Duty cycle*  $D = 25\%$

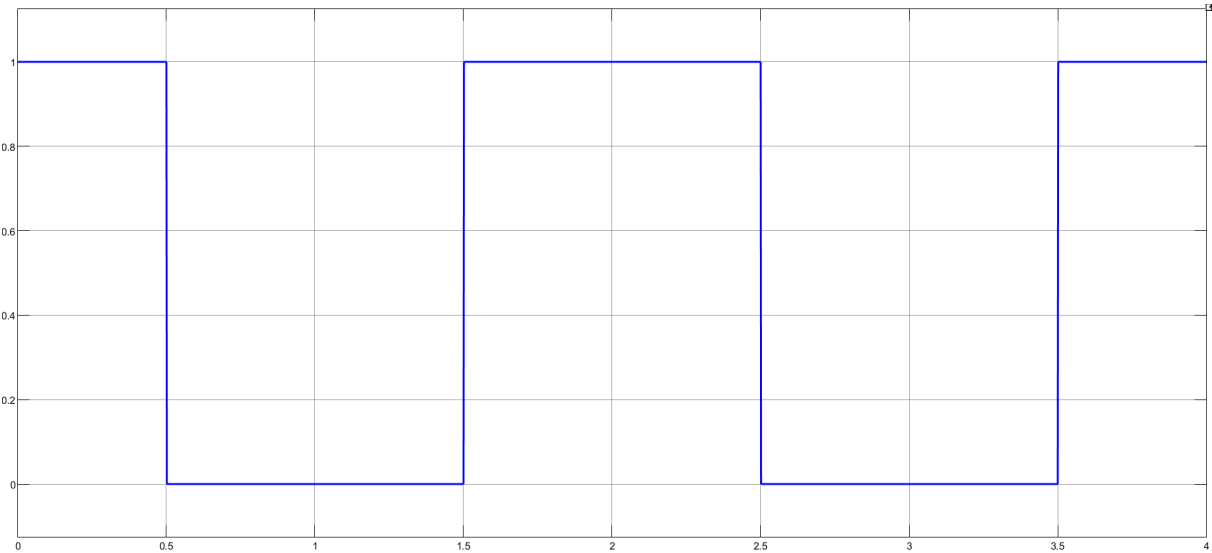


Figure 3.22: *Duty cycle  $D = 50\%$*

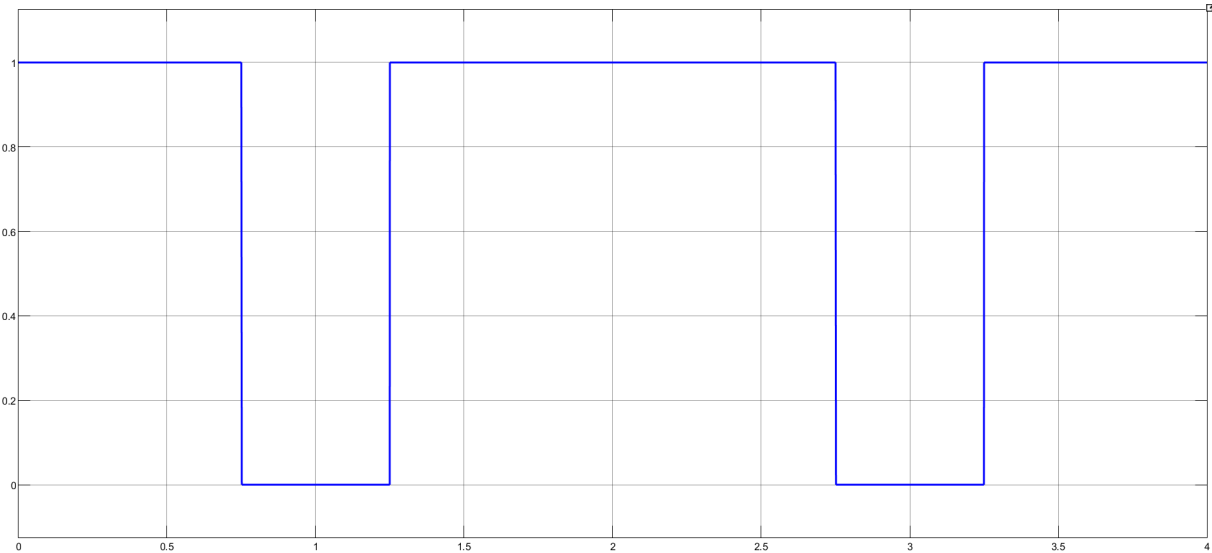


Figure 3.23: *Duty cycle  $D = 75\%$*

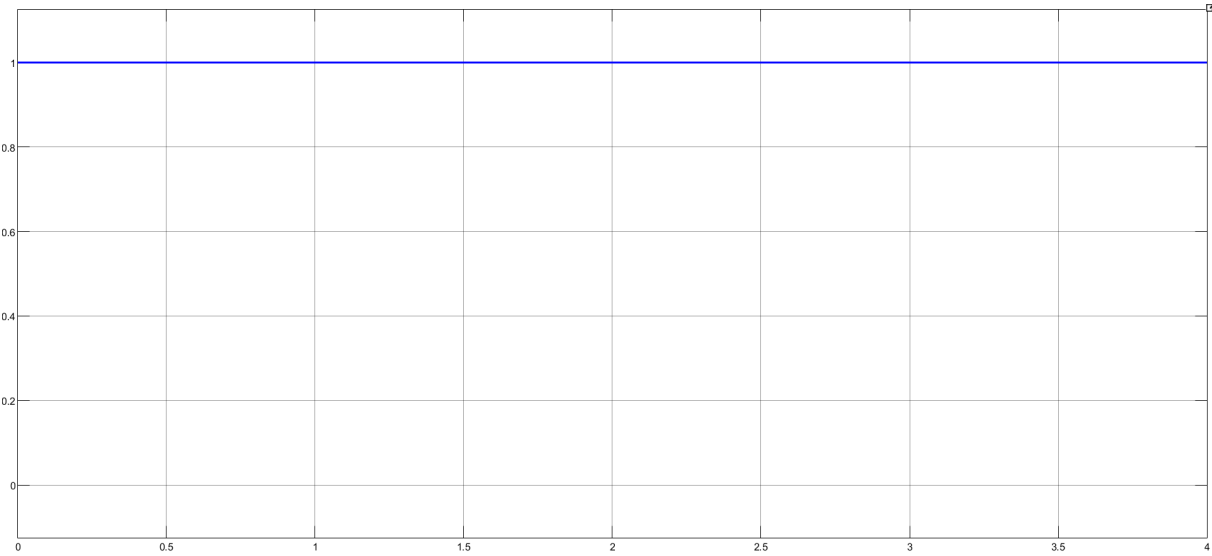


Figure 3.24: *Duty cycle*  $D = 100\%$

# Chapter 4

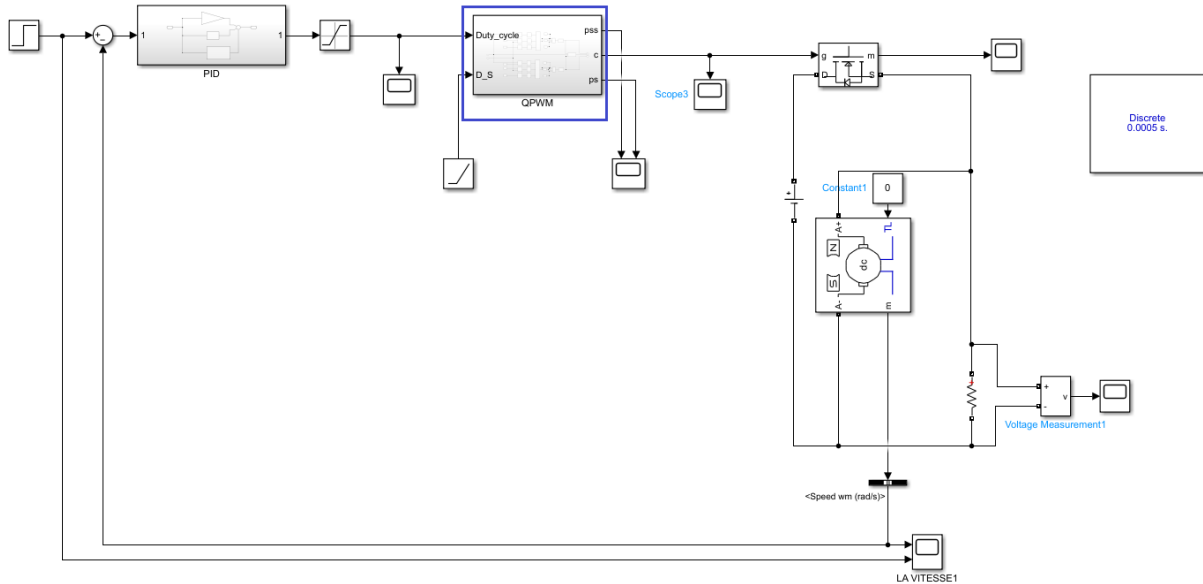
## Comparative Study between Quantum PWM and Classical PWM

### 4.1 Introduction

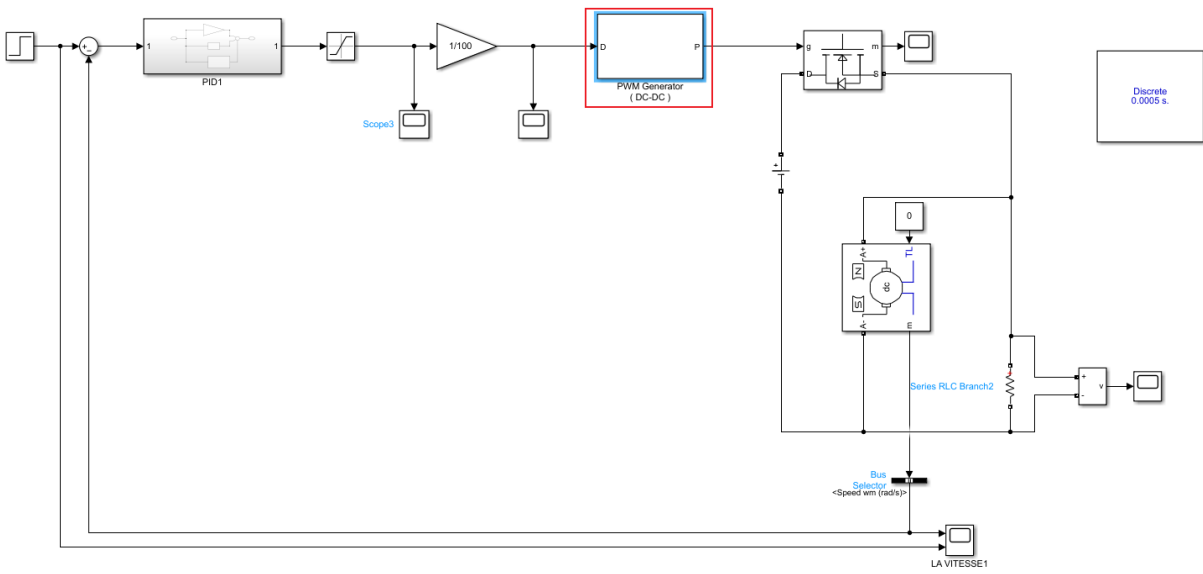
Closed-loop control systems are vital for achieving accurate and stable motor speed regulation[25]. In motor control applications, Pulse Width Modulation (PWM) techniques are widely employed. This study aims to investigate and compare the error between reference and speed signals when utilizing two distinct PWM techniques: Quantum Comparator PWM and Classic Comparator PWM. To accomplish this, two separate circuits were implemented as shown in Figure 4.1, each employing either Quantum Comparator PWM or Classic Comparator PWM, for the purpose of controlling the speed of a DC motor. The speed signals were continuously monitored and compared to their corresponding reference signals. To ensure stable speed control with minimal fluctuations and prevent speed overshoot, the control algorithms were appropriately configured.

In this chapter, our primary objective is to compare the error between the reference and speed signals in closed-loop control systems that employ Quantum Comparator PWM and Classic Comparator PWM techniques. The performance of these control systems will be evaluated based on their ability to achieve stable speed control, minimize fluctuations, avoid speed overshoot, and ensure precise motor operation. Experimental results will be presented to demonstrate the similarity in speed outcomes between the two PWM techniques, thereby validating their effectiveness in attaining desired speed setpoints without excessive overshoot.





(a) Speed control circuit utilizing PWM based on quantum comparator



(b) Speed control circuit utilizing classical PWM

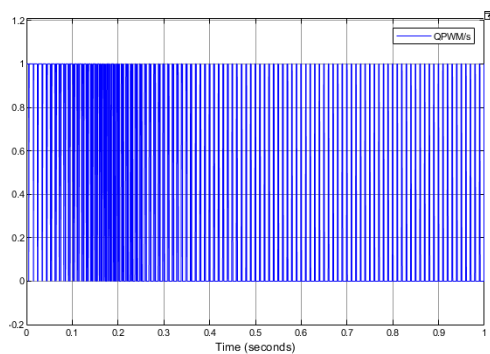
Figure 4.1: Speed control circuits

## 4.2 Comparing PWM Signals in Closed-Loop Control Systems: Quantum Comparator PWM vs Classic Comparator PWM

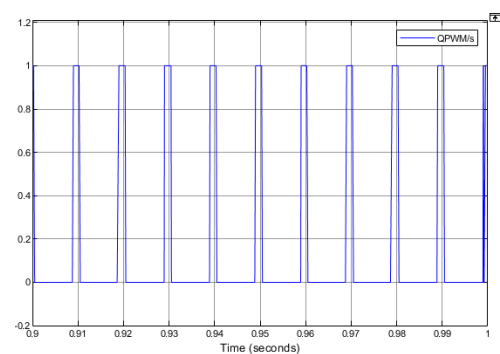
In this section, we present a comparative analysis of PWM signals in closed-loop control systems. Specifically, we examine the performance of two distinct PWM techniques (Quantum Comparator-based PWM and Classical PWM).

The Quantum Comparator-based PWM technique is implemented using the circuit depicted in Figure 4.1a, while the Classical PWM technique utilizes the circuit illustrated in Figure 4.1b. The obtained PWM signals are visualized in figure 4.2, figure 4.3, figure 4.4, figure 4.5, figure 4.6 and figure 4.7.

Throughout our analysis, we systematically vary the reference value and observe its impact on the PWM signals, aiming to discern the differences and characteristics between these two employed methods.

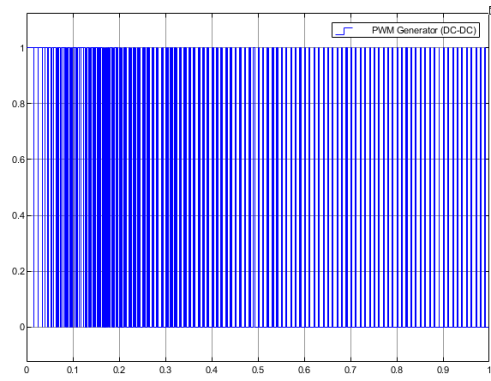


(a) PWM Signal for 1 Second

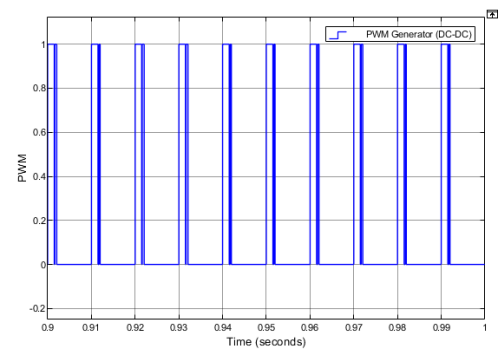


(b) PWM Signal for 0.1 Second

Figure 4.2: PWM Signals generated by quantum comparator technique in closed loop for a reference of 5 rad/s

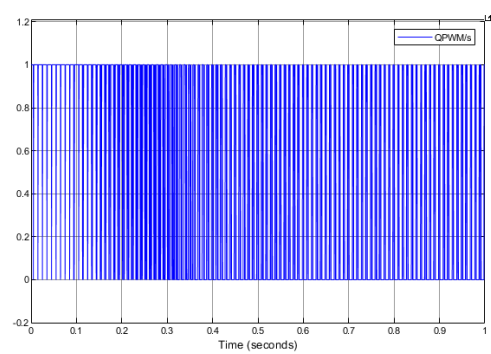


(a) PWM Signal for 1 Second

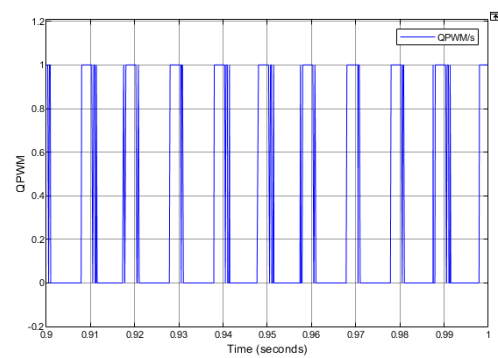


(b) PWM Signal for 0.1 Second

Figure 4.3: PWM Signals generated by classical comparator technique in closed loop for a reference of 5 rad/s

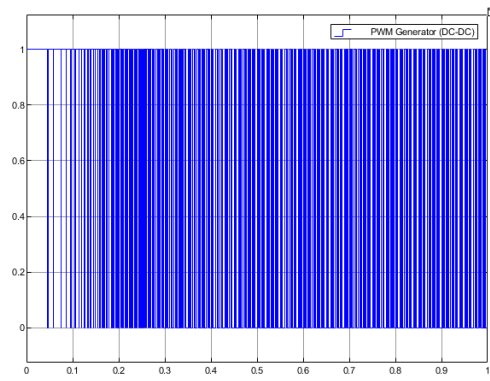


(a) PWM Signal for 1 Second

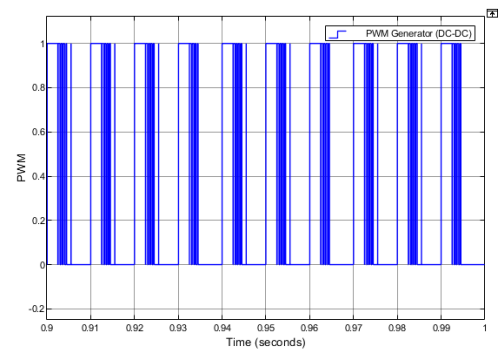


(b) PWM Signal for 0.1 Second

Figure 4.4: PWM Signals generated by quantum comparator technique in closed loop for a reference of 10 rad/s

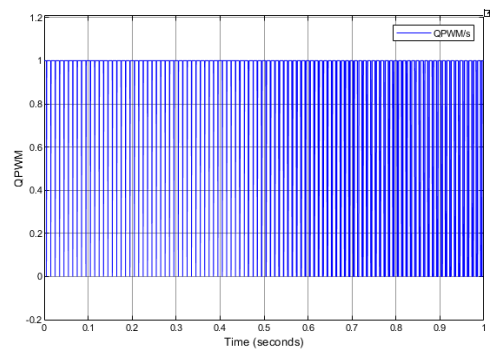


(a) PWM Signal for 1 Second

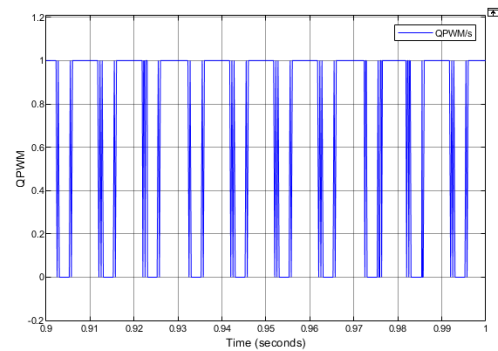


(b) PWM Signal for 0.1 Second

Figure 4.5: PWM Signals generated by classical comparator technique in closed loop for a reference of 10 rad/s

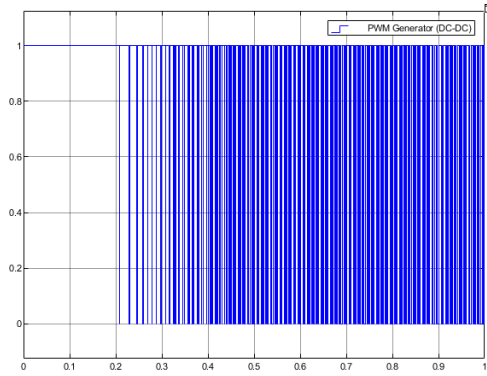


(a) PWM Signal for 1 Second

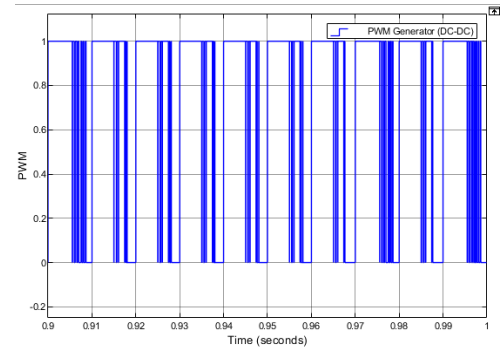


(b) PWM Signal for 0.1 Second

Figure 4.6: PWM Signals generated by quantum comparator technique in closed loop for a reference of 20 rad/s



(a) PWM Signal for 1 second



(b) PWM Signal for 0.1 second

Figure 4.7: PWM Signals generated by classical comparator technique in closed loop for a reference of 20 rad/s

### 4.2.1 PWM Signals comparison results

Both methods employ Pulse Width Modulation (PWM) signals to adaptively regulate the motor speed according to the desired setpoint. The comparators within the control systems continuously compare the instantaneous voltage of the waveform input with a reference voltage. When the waveform voltage exceeds the reference voltage, the comparator outputs a logic high ('1'), whereas a waveform voltage lower than the reference voltage results in a logic low ('0') output.

Through a comprehensive examination, it was observed that both Quantum Comparator PWM and Classic Comparator PWM techniques yielded comparable results in the context of PWM-based speed control. This indicates that the use of Quantum Comparator PWM, as a novel approach, was able to achieve similar outcomes to the conventional Classic Comparator PWM technique, regardless of changes in the reference voltage. Consequently, these findings highlight the effectiveness and success of our proposed Quantum Comparator PWM method in accurately controlling the motor speed.

## 4.3 Comparing Speed Signals in Closed-Loop Control Systems: Quantum Comparator PWM vs Classic Comparator PWM

In this section, we perform a comparative analysis of speed control signals in closed-loop control systems, specifically examining the discrepancies between the quantum comparator-based model and the classical model in motor speed control.

The primary objective is to evaluate the degree of differentiation between these two methods, assessing the efficacy and precision of the Quantum Comparator-based model. By examining the speed control signals generated by both techniques, we aim to gain insights into performance disparities and measure the success and accuracy of the quantum model.

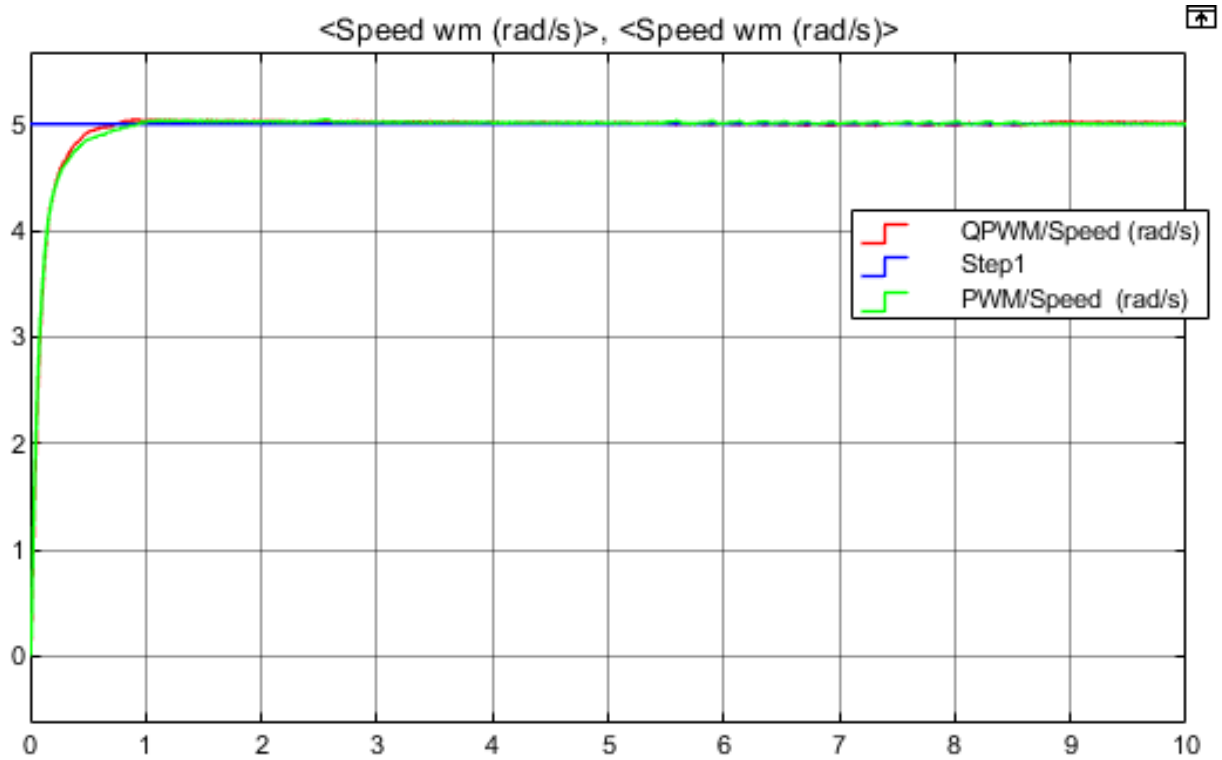


Figure 4.8: Comparison of speed signals for a reference of 5 rad/s

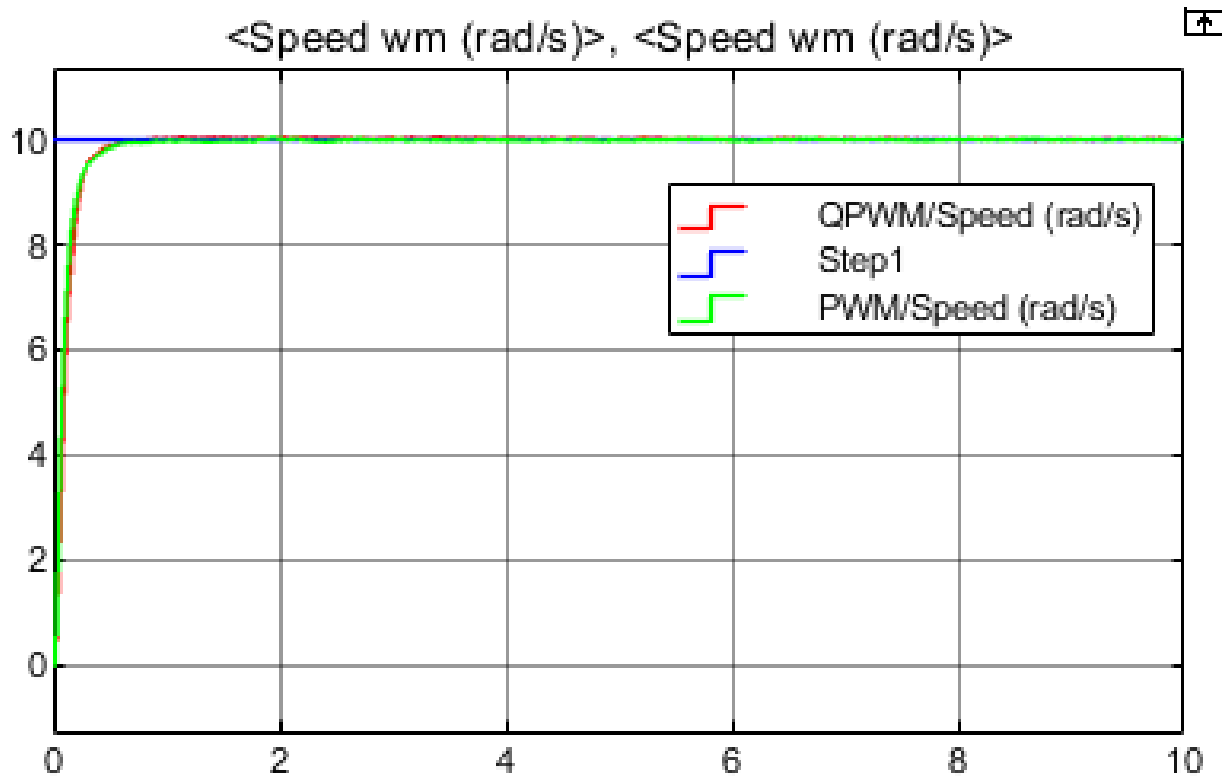


Figure 4.9: Comparison of speed signals for a reference of 10 rad/s

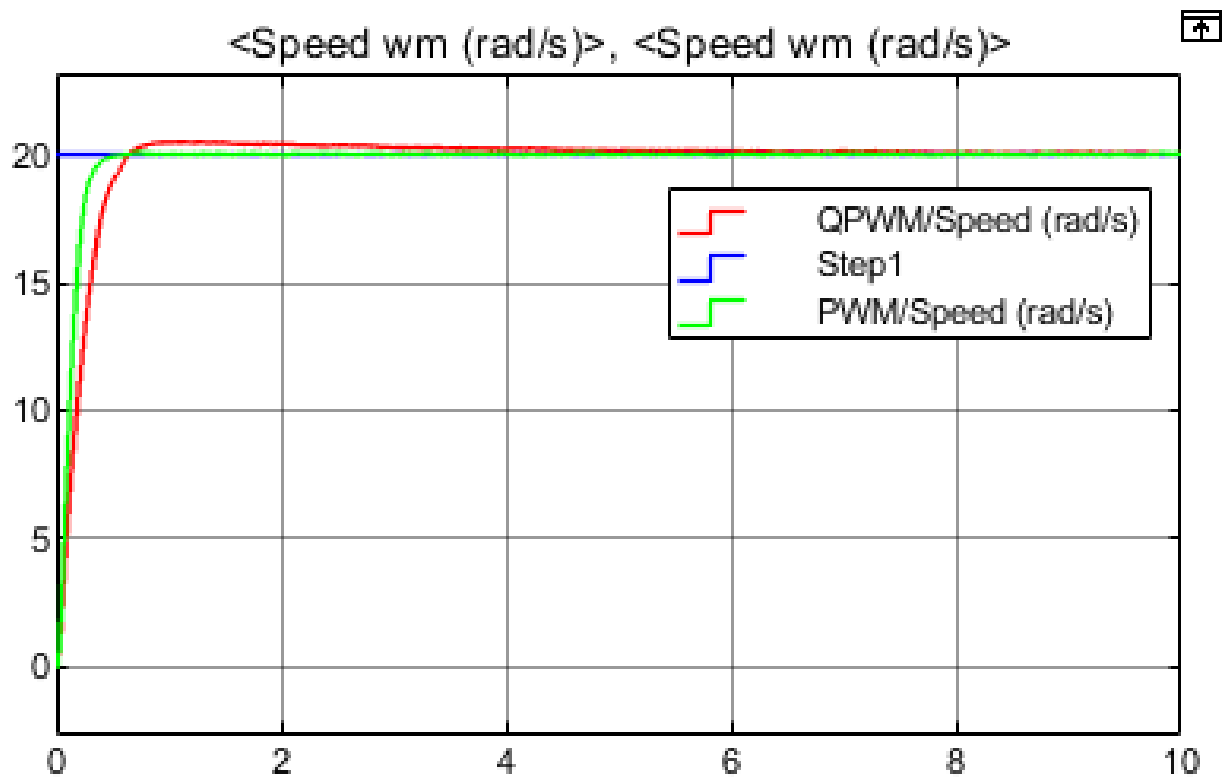


Figure 4.10: Comparison of speed signals for a reference of 20 rad/s

### 4.3.1 Speed comparison results

The analysis of the speed signals obtained from our simulated study using Simulink demonstrates that both our Quantum Comparator PWM and Classic Comparator PWM techniques produced similar speed outcomes. Our closed-loop control systems, which integrated our Quantum Comparator PWM, exhibited effective maintenance of stable speed control. The speed signal consistently tracked the desired speed setpoint, indicating the successful implementation of our control algorithms in accurately adjusting the control signal to achieve the desired speed. This precise control ensured the motor operated with accuracy, guaranteeing optimal speed performance.

## 4.4 Comparing error between reference and speed signals in closed-loop control systems: Quantum comparator PWM vs Classic comparator PWM

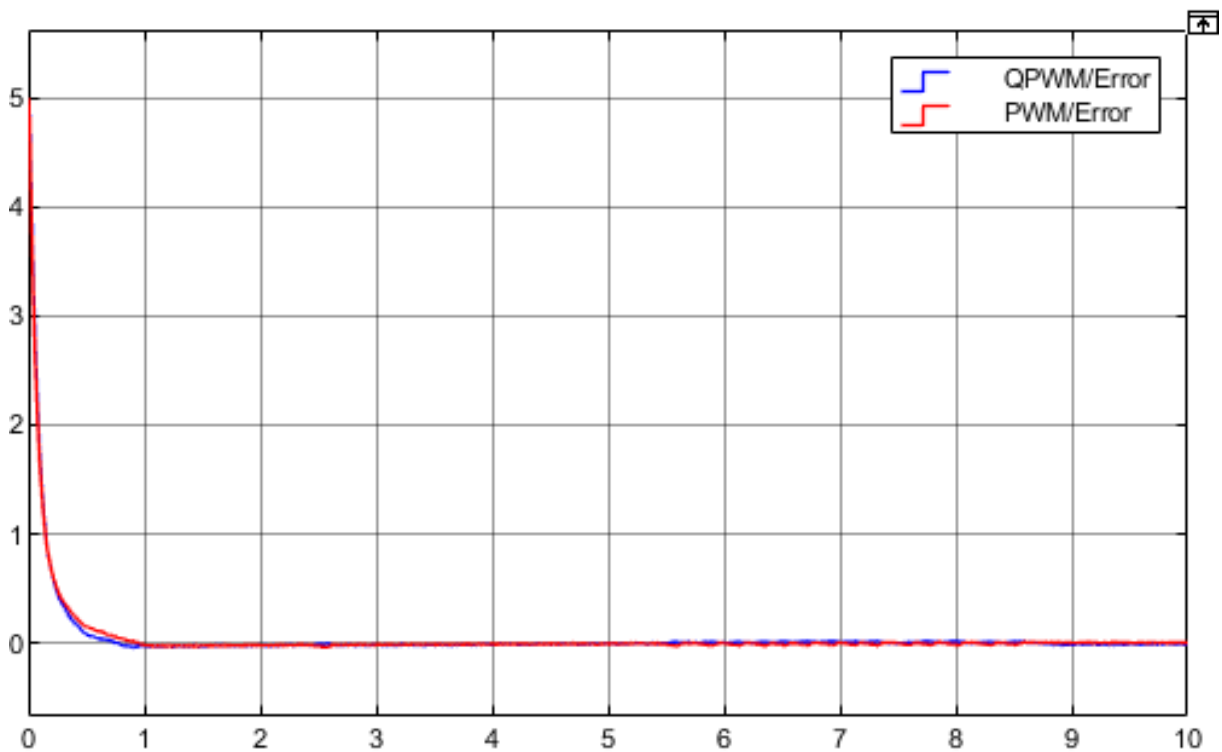


Figure 4.11: Comparison of error between reference and speed signals for a reference of 5 rad/s



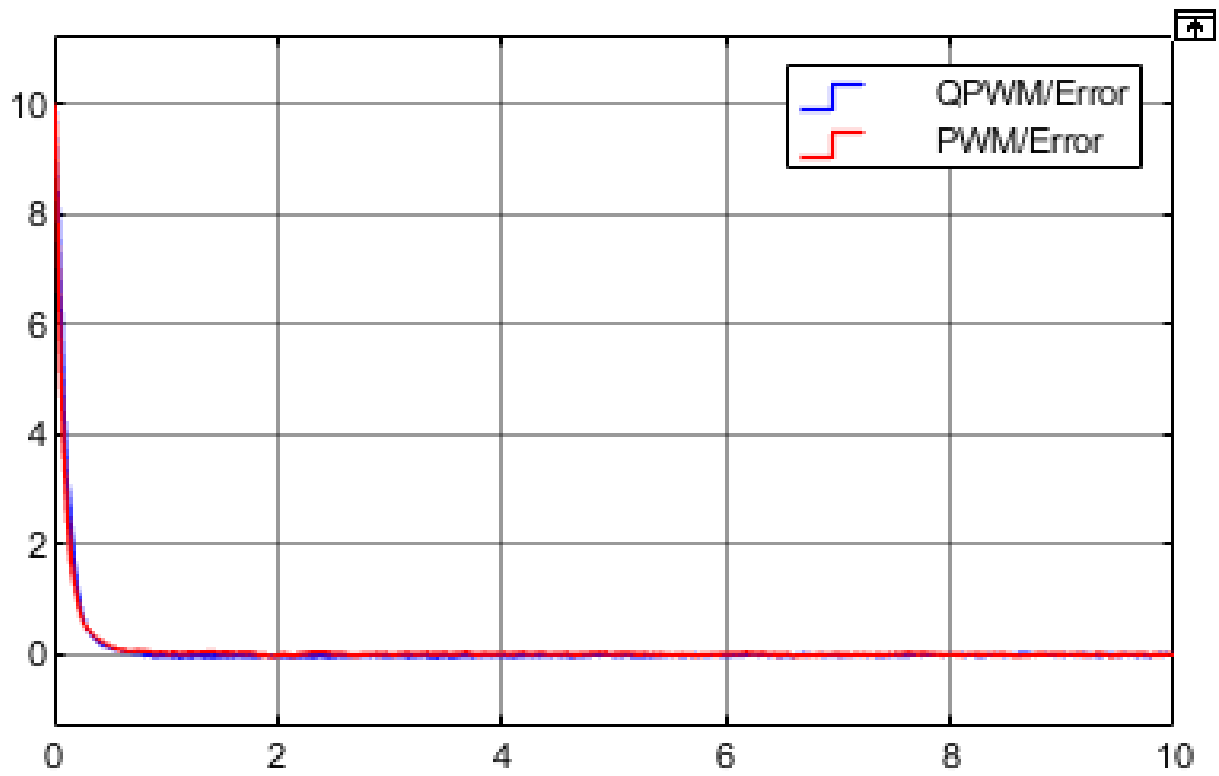


Figure 4.12: Comparison of error between reference and speed signals for a reference of 10 rad/s

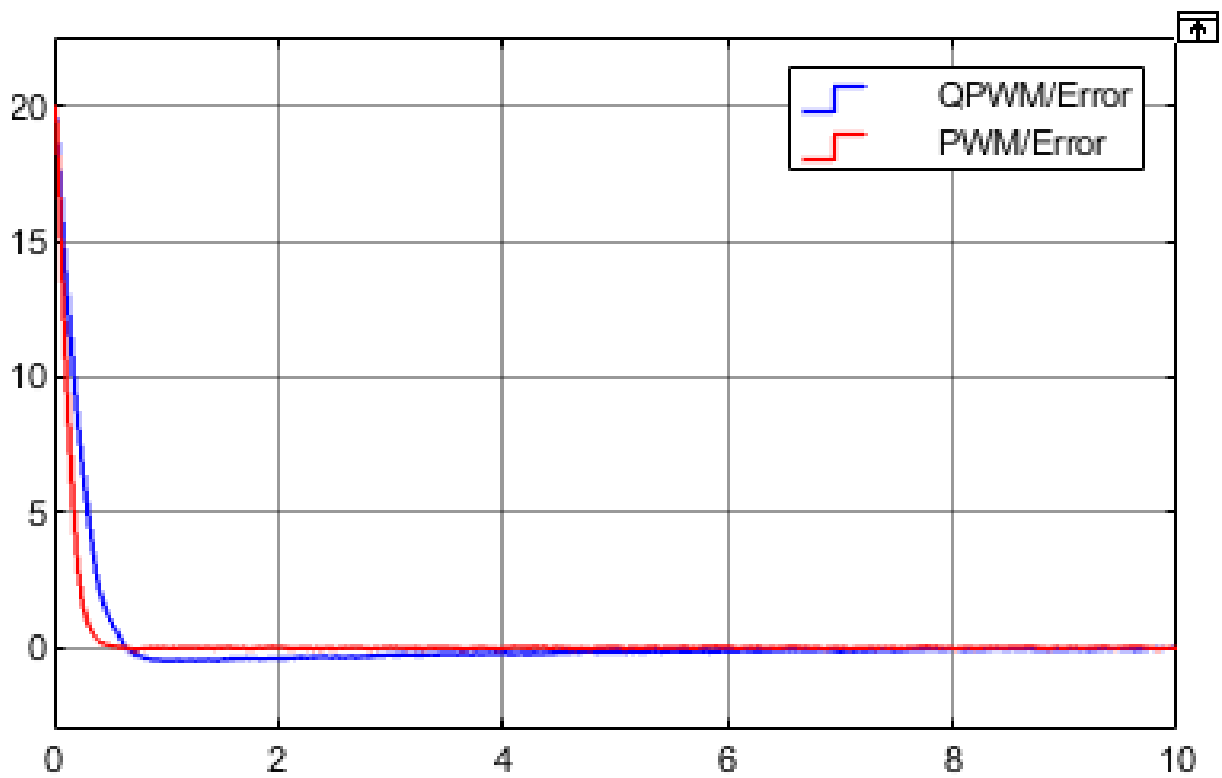


Figure 4.13: Comparison of error between reference and speed signals for a reference of 20 rad/s

#### 4.4.1 Errors comparison results

Our study reveals that both approaches yielded similar results with respect to the reduction of errors. The integration of closed-loop control systems with our Quantum Comparator PWM technique effectively maintained the error within acceptable limits across various speed references, ultimately converging the error signals towards zero. To ensure a comprehensive evaluation of the error reduction capabilities, this study employed performance metrics and statistical analyses. These quantitative measures allowed for an objective assessment of the effectiveness of both the Quantum Comparator PWM and Classic Comparator PWM techniques. Moreover, an in-depth examination of the design of the control algorithm shed light on its significant influence on error reduction. This emphasizes the crucial role of well-designed algorithms in achieving accurate speed control within closed-loop systems[10].

The findings of this study hold practical implications, as they underscore the applicability of both the Quantum Comparator PWM and Classic Comparator PWM techniques in achieving precise and reliable control within closed-loop systems. These insights provide valuable guidance for practitioners seeking to implement these techniques in real-world applications, thus contributing to the advancement of control systems technology.

### 4.5 Staircase Reference Testing in Simulink: Evaluating Control System Performance

In this section, a comprehensive analysis of Staircase Reference Testing will be presented to evaluate the accuracy of the Quantum PWM model in comparison to the Classical PWM model. The Staircase Reference Testing method will be employed to examine the performance of both PWM models under controlled conditions. The objective is to assess the fidelity of the Quantum PWM model by comparing its output with that of the Classical PWM model. The objective is to ascertain the fidelity of the Quantum PWM model by comparing its simulated output with that of the Classical PWM model. This simulation-based investigation aims to provide insights into the accuracy and reliability of the Quantum PWM model, further contributing to the understanding and improvement of PWM control systems in closed-loop simulations.

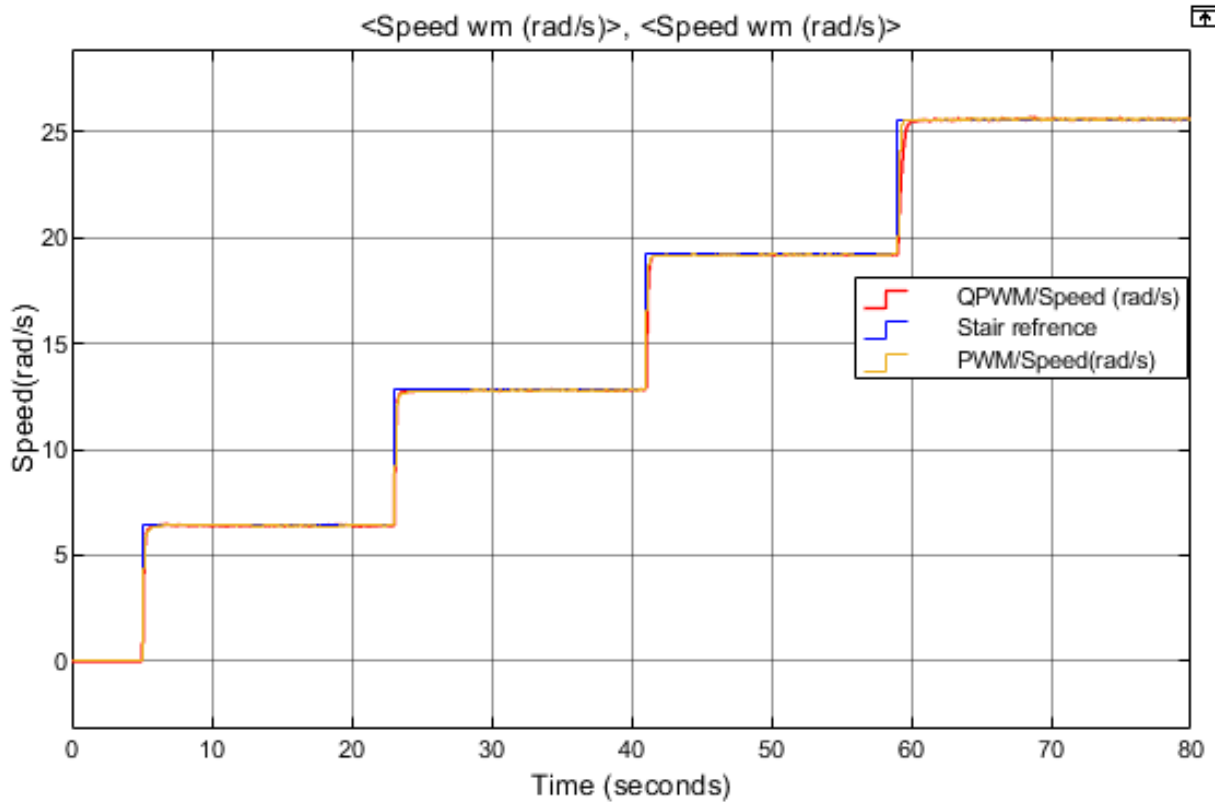


Figure 4.14: Comparison of speed signals for a staircase reference

The graph obtained from the simulation results provides valuable insights into the performance of the quantum PWM model compared to the classic PWM model in response to the staircase reference test.

Upon careful examination, it is evident that both models demonstrate a consistent and accurate response to the increase in the reference value. As the reference value climbs, the speed of the system exhibits a corresponding and proportional increase. This indicates that both models effectively regulate the motor speed in accordance with the desired reference, showcasing their ability to accurately control the system under varying operating conditions. Moreover, the close resemblance between the results of the quantum PWM model and the classic PWM model reinforces the validity and reliability of the former. The similarity in their responses suggests that the quantum PWM model can serve as a suitable alternative to the well-established classic PWM model, providing comparable performance in terms of speed control accuracy.

It is noteworthy that the observed behavior of the graph reinforces the robustness of the quantum PWM model in maintaining speed regulation across different reference values. This characteristic is of paramount importance in various applications where precise speed control is critical, such as robotics, industrial automation, and motion control systems. The graph provides compelling evidence that the quantitative PWM model performs

---

admirably in accurately adjusting the motor speed in response to variations in the reference value. These findings affirm the potential of the quantum PWM model as a viable alternative to the classic PWM model, encouraging further exploration and application of this approach in real-world systems.

## 4.6 Results and analysis

The analysis of the speed signals obtained from our simulated study using Simulink reveals that both our Quantum Comparator PWM and Classic Comparator PWM techniques yielded similar speed results. The closed-loop control systems, incorporating our Quantum Comparator PWM, effectively maintained stable speed control, with the speed signal closely tracking the desired speed setpoint. This finding demonstrates the successful implementation of our control algorithms in adjusting the control signal to match the desired speed, ensuring precise motor operation. Furthermore, the observed minimal fluctuations in the speed signals emphasize the precision and robustness of our control algorithms, utilized in both our Quantum Comparator PWM and Classic Comparator PWM techniques. These algorithms effectively adapt and adjust the control signal to mitigate disturbances and variations encountered during motor operation. By minimizing fluctuations, our control algorithms enhance the overall performance of the control systems, ensuring consistent and reliable motor speed regulation[22].

This accomplishment highlights the safety and reliability of our control systems, meeting the requirements of various industrial applications.

The successful control of motor speed achieved by our Quantum Comparator PWM technique, developed through our joint efforts, underscores its viability and effectiveness in closed-loop control systems. Our technique provides a practical and efficient means of regulating motor speed with accuracy and stability. By comparing and evaluating the performance of our Quantum Comparator PWM technique alongside the Classic Comparator PWM, we gain valuable insights into its suitability for specific motor control applications.

It is important to highlight that while our Quantum Comparator PWM technique has demonstrated effectiveness and promise in achieving accurate speed control in closed-loop systems, its true potential can only be fully realized through the implementation of its algorithm in a real quantum computer. The unique properties and computational capabilities of quantum computing platforms have the potential to unlock additional benefits and

optimizations that may further enhance the performance and efficiency of our Quantum Comparator PWM approach. Therefore, future research and practical implementation efforts should focus on harnessing the power of quantum computing to fully exploit the advantages of our proposed technique and pave the way for advancements in motor control and other related fields.

# General Conclusion

In conclusion, our work has explored the application of quantum computing principles in the world of power electronics, specifically in the design and implementation of Pulse Width Modulation (PWM) techniques for DC motor speed control. Through our research and experimentation, we have successfully developed a Quantum PWM (QPWM) technique and conducted a comparative study with classical PWM. By introducing the concept of QPWM and conducting a comparative study, we provide valuable insights into the potential benefits and challenges of implementing quantum computing techniques in practical applications.

During the development of our QPWM technique, we focused on a crucial aspect: the ability to compare two qubit numbers that are in superposition and vary around the Y-axis of the Bloch sphere. This unique approach allows for the comparison of real numbers, unlike existing quantum comparators described in the literature, which only handle binary values. By dividing our proposed solution into two complementary parts, each managing probabilistic intervals, we achieved comprehensive management of real number intervals ranging from 0 to 100%. By leveraging the concept of measurement on the output of the third qubit that which represents our result of comparison, we successfully achieved the necessary switching between the two states ( $|0\rangle$  and  $|1\rangle$ ) on which the PWM technique relies.

Our findings indicate that both QPWM and classical PWM techniques yield very close performance in controlling the speed of a DC motor. While the QPWM technique harnesses the power of quantum computing and showcases its potential, we observed that the benefits in terms of performance improvement were not significant compared to classical PWM. However, it is important to note that the full potential of quantum computing may not have been fully realized in our project due to limitations in available hardware and resources. To unlock the true power of quantum computing for DC motor control and achieve better performance, it would be necessary to apply quantum algorithms on a quantum computer. This would enable us to exploit the unique properties of quantum

systems, such as superposition and entanglement, to devise more advanced and efficient control strategies.

In summary, our project demonstrates that while QPWM and classical PWM techniques offer similar performance for DC motor speed control, the potential for significant performance improvement lies in the future application of quantum algorithms on quantum computers. This opens up avenues for further research and development in the field of quantum power electronics, as we strive to unlock the full capabilities of quantum computing in enhancing the efficiency, precision, and sophistication of DC motor control systems.

# Bibliography

- [1] H. ABU-RUB, A. IQBAL, AND J. GUZINSKI, *High Performance Control of AC Drives with Matlab / Simulink Models*, Wiley, 2012.
- [2] A. ADEDOYIN, J. AMBROSIANO, P. ANISIMOV, W. CASPER, G. CHENNUPATI, C. COFFRIN, H. DJIDJEV, D. GUNTER, S. KARRA, N. LEMONS, ET AL., *Quantum algorithm implementations for beginners*, arXiv preprint arXiv:1804.03719, (2018).
- [3] M. AHMAD, *Advances in Motor Torque Control*, IntechOpen, 2011.
- [4] A. ASKINAS, N. P. S. M. CA., AND N. P. S. (U.S.), *Pulsewidth Modulated Speed Control of Brushless DC Motors*, Naval Postgraduate School, 1984.
- [5] B. E. BAAQUIE AND L.-C. KWEK, *Phase estimation and quantum fourier transform (qft)*, in *Quantum Computers: Theory and Algorithms*, Springer, 2023, pp. 155–169.
- [6] C. BERNHARDT, *Quantum Computing for Everyone*, The MIT Press, MIT Press, 2019.
- [7] D. BOUWMEESTER, J.-W. PAN, K. MATTLE, M. EIBL, H. WEINFURTER, AND A. ZEILINGER, *Experimental quantum teleportation*, *Nature*, 390 (1997), pp. 575–579.
- [8] P. BRUMER AND J. GONG, *Born rule in quantum and classical mechanics*, *Physical Review A*, 73 (2006), p. 052109.
- [9] S. DAS BARMAN, A. HUSSAIN, AND T. AHMED, *Speed Control of Dc Motor Using Pwm Technique*, Lap Lambert Academic Publishing GmbH KG, 2012.
- [10] J. D’AZZO AND C. HOUPIS, *Linear Control System Analysis and Design: Conventional and Modern*, McGraw-Hill Computer Science Series, McGraw-Hill, 1988.
- [11] P. A. M. DIRAC, *A new notation for quantum mechanics*, *Mathematical Proceedings of the Cambridge Philosophical Society*, 35 (1939), p. 416–418.



- [12] B. DRURY, C. T. (FIRM), AND I. OF ELECTRICAL ENGINEERS, *Control Techniques Drives and Controls Handbook*, IEE Power Series, Institution of Engineering and Technology, 2001.
- [13] M. EL AMINE BOUDJOGHRA, S. A. SOFIANE DAIMELLAH, N. ZIOUI, Y. MAHMOUDI, AND M. TADJINE, *State-domain equations and their quantum computing solution based hhl algorithm.*, *Mathematical Modelling of Engineering Problems*, 9 (2022).
- [14] N. ERTUGRUL, *LabVIEW for Electric Circuits, Machines, Drives, and Laboratories*, National Instruments virtual instrumentation series, Prentice Hall PTR, 2002.
- [15] I. GOTTLIEB, *Electric Motors & Control Techniques*, TAB Books, 1994.
- [16] [HTTPS://QUANTUM COMPUTING.IBM.COM](https://quantum.computing.ibm.com).
- [17] [HTTPS://WWW.MATHWORKS.COM/HELP/SPS/REF/CONTROLLEDPWMVOLTAGE.HTML](https://www.mathworks.com/help/spc/ref/controlledpwmvoltage.html).
- [18] [HTTPS://WWW.MATHWORKS.COM/HELP/SPS/REF/HBRIDGE.HTML?s\\_tid = doc\\_ta](https://www.mathworks.com/help/spc/ref/hbridge.html?s_tid=doc_ta).
- [19] [HTTPS://WWW.MATHWORKS.COM/HELP/SPS/REF/PWMGENERATOR.HTML](https://www.mathworks.com/help/spc/ref/pwmgenerator.html).
- [20] R. KRISHNAN, *Permanent Magnet Synchronous and Brushless DC Motor Drives*, CRC Press, 2017.
- [21] S. LATFI, *Development of Control Shceme for DC Motor Speed Control Applications*, UMP, 2010.
- [22] R. LOZANO, *Adaptive Control*, Communications and Control Engineering, Springer London, 1997.
- [23] A. MERABET, *Advanced Control Systems for Electric Drives*, MDPI AG, 2020.
- [24] E. MONMASSON, *Power Electronic Converters: PWM Strategies and Current Control Techniques*, ISTE, Wiley, 2013.
- [25] A. MUTAMBARA, *Design and Analysis of Control Systems*, Taylor & Francis, 1999.
- [26] M. NIELSEN AND I. CHUANG, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 2010.
- [27] C. PHILIPPIDIS, C. DEWDNEY, AND B. J. HILEY, *Quantum interference and the quantum potential*, *Nuovo Cimento B*, 52 (1979), pp. 15–28.

- [28] J. PYRHONEN, V. HRABOVCOVA, AND R. SEMKEN, *Electrical Machine Drives Control: An Introduction*, Wiley, 2016.
- [29] S. RINDERKNECHT, P. JARDIN, AND A. ESSER, *Future Powertrain Technologies*, MDPI AG, 2020.
- [30] F. RODRIGUEZ, *Advanced Digital Control Techniques for Brush-less DC (BLDC) Motor Drives*, Illinois Institute of Technology, 2006.
- [31] J. RODRIGUEZ AND P. CORTES, *Predictive Control of Power Converters and Electrical Drives*, IEEE Press, Wiley, 2012.
- [32] J. WILEY AND SONS, *Design and Control of High-Performance DC Motor Drives*, John Wiley & Sons, 2015.
- [33] T. WONG, *Introduction to Classical and Quantum Computing*, Rooted Groove., 2022.
- [34] N. YOUNG, *An Introduction to Hilbert Space*, Cambridge mathematical textbooks, Cambridge University Press, 1988.