การศึกษาและเปรียบเทียบแพลตฟอร์มการจำลองประมวลผลควอนตัม

เธียร สืบวงษ์

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
ปีการศึกษา 2564

A STUDY AND COMPARISON OF QUANTUM COMPUTING SIMULATION PLATFORMS

TIAN SUEPWONG

A Thematic Paper Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering

Department of Computer Engineering

College of Innovative Technology and Engineering,

Dhurakij Pundit University

Academic Year 2021

ใบรับรองสารนิพนธ์

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

หัวข้อสารนิพนธ์	การศึกษาและเปรียบเทียบแพลตฟอร์มการจำลองประมวลผลควอนตัม
เสนอโดย	นายเธียร สืบวงษ์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาสารนิพนธ์	อาจารย์ ดร.ชัยพร เขมะลาภะพันธ์

ได้พิจารณาเห็นชอบโดยคณะกรรมการสอบสารนิพนธ์แล้ว

……ประธานกรรมการ

(รองศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ)

…… กรรมการและอาจารย์ที่ปรึกษาสารนิพนธ์

(อาจารย์ ดร.ชัยพร เขมะลาภะพันธ์)

……กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.มัชฌิกา อ่องแตง)

วิทยาลัย นวัตกรรมด้านเทคโนโลยีและวิศวกรรมคอมพิวเตอร์รับรองแล้ว

……คณบดีวิทยาลัย นวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

(อาจารย์ ดร.ชัยพร เขมะลาภะพันธ์)

วันที่...30....**เดือน**...กรกฎาคม...พ.ศ.2565

หัวข้อสารนิพนธ์ การศึกษาและเปรียบเทียบแพลตฟอร์มการจำลองประมวลผลควอนตัม
ชื่อผู้เขียน เธียร สืบวงษ์
อาจารย์ที่ปรึกษา ดร.ชัยพร เขมะภาตะพันธ์
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2564

บทคัดย่อ

ปัจจุบันมีความก้าวหน้าทางเทคโนโลยีมากขึ้นหนึ่งในนั้นคือเทคโนโลยีควอนตัมคอมพิวเตอร์ซึ่งอาจเป็นหนึ่งในเทคโนโลยีสำคัญในอนาคต ทำให้มีการพัฒนาแพลตฟอร์มสำหรับการประมวลผลควอนตัมหลายแพลตฟอร์ม ทำให้การเลือกใช้งานและศึกษาในแต่ละซอฟต์แวร์ควอนตัมแพลตฟอร์มค่อนข้างยุ่งยาก ทำให้ผู้ใช้งานจำเป็นต้องใช้เวลากับการศึกษาและเลือกใช้แพลตฟอร์มที่เหมาะสมและถูกต้อง

การศึกษานี้ได้คัดเลือกซอฟต์แวร์ควอนตัมแพลตฟอร์มที่น่าสนใจมาทั้งหมด 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) โดยจะทำการรวบรวมข้อมูลต่าง ๆ ภาพรวมระบบ การติดตั้ง, การใช้งาน ภาษา และรูปแบบการเขียนโปรแกรม การทดสอบฟังก์ชั่นและอัลกอริทึมว่าสามารถใช้งานอะไรได้บ้าง และนำมาเปรียบเทียบภาพรวมการ เพื่อคัดเลือก 2 แพลตฟอร์มที่เหมาะสมกับการใช้งาน โดยขั้นตอนสุดท้ายจะนำ 2 แพลตฟอร์มที่เลือกเอาไว้มาเปรียบเทียบประสิทธิภาพการทำงานแบบจำลองโดยใช้ Shor's algorithm

การทดสอบทั้งหมด 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) ทำให้เห็นภาพรวมของทั้ง 4 แพลตฟอร์มที่เป็นประโยชน์ต่อผู้ที่ต้องการศึกษาและพัฒนาโปรแกรมโดยเลือกใช้ 4 แพลตฟอร์มที่กล่าวมา และการคัดเลือก 2 แพลตฟอร์มที่มีฟังก์ชันและคุณสมบัติการทำงานครบถ้วนมากที่สุดในการจำลองการทำงานเพื่อวัดประสิทธิภาพประมวลผลโดยใช้ขึ้นตอนวิธี Shor's algorithm ซึ่งใช้สำหรับแยกตัวประกอบทางคณิตศาสตร์ โดยพบว่าแพลตฟอร์ม QisKit มีฟังก์ชันและคุณสมบัติโดยรวมค่อนข้างครบถ้วนมากกว่า รวมทั้งสามารถจำลองได้อย่างมีประสิทธิภาพเร็วกว่าและมีความถูกต้องมากกว่าเมื่อเทียบกับแพลตฟอร์มอื่น ๆ

Thematic Paper Title	STUDY AND COMPARISON OF SIMULATION PLATFORMS FOR QUANTUM
Author	Tian Suepwong
Thematic Paper Advisor	Dr.Chaiyaporn Khemapatapan
Department	Computer Engineering
Academic Year	2021

ABSTRACT

Nowadays, with more technological advances, quantum computing technology is one of the key technologies in the future. There are several simulation platforms for quantum computing. This makes the selection and study of each quantum platform quite difficult. Thus, users who need to use the platform to study and research have to take times and select the platform which is appropriate and accurate.

Four interesting quantum software platforms were selected: Forest (pyQuil), Qiskit, ProjectQ, and Quantum Developer Kit (Q#), which collect information, system overview, installation, usage. Programming language and style. Testing what functions and algorithms can be used and comparing the overall picture to select 2 platforms that are suitable for use The final step was to compare the performance of the two selected platforms using Shor's algorithm.

The four testing platforms Forest (pyQuil), Qiskit, ProjectQ and Quantum Developer Kit (Q#) provide an overview of the four platforms that are useful for those who wish to study and develop programs using the four platforms mentioned above. And the selection of two of the best platforms to measure model performance using Shor's algorithm identifies the most suitable platform to study and use.

กิตติกรรมประกาศ

สารนิพนธ์นี้ได้ดำเนินการสำเร็จลุล่วงและประสบความสำเร็จไปได้ด้วยดี ขอขอบพระคุณ อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์ อาจารย์ที่ปรึกษา ที่คอยให้คำปรึกษา กรุณาให้ความช่วยเหลือไม่ว่าจะเป็นข้อเสนอแนะการแก้ปัญหา เกี่ยวกับสารนิพนธ์ที่เป็นประโยชน์ในงานวิจัย รวมถึงการสละเวลา ให้คำปรึกษาตลอดเวลา อีกทั้งยังเอาใจใส่นักศึกษาสม่ำเสมอตลอดเวลา

ขอขอบพระคุณ คณาจารย์ในสาขาวิชาทุกท่านที่ได้สั่งสอนความรู้ ให้คำปรึกษาเสียสละเวลา และคณะกรรมการสอบทุกท่าน เพื่อนร่วมชั้นเรียนที่คอยให้กำลังใจ เสนอแนะข้อมูลในการทำสารนิพนธ์เล่มนี้ ท้ายที่สุดขอกราบขอบพระคุณบิดา มารดาของข้าพเจ้าที่ให้กำลังใจและสนับสนุนทางด้านการศึกษา

ขอขอบพระคุณทุกท่านอย่างสูงที่ให้ การสนับสนุน เอื้อเฟื้อและให้ความอนุเคราะห์ช่วยเหลือ จนกระทั่งสารนิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี

เธียร สืบวงษ์

สารบัญ

หน้า

บทคัดย่อภาษาไทย…… ฆ

บทคัดย่อภาษาอังกฤษ…….... ง

กิตติกรรมประกาศ……..... จ

สารบัญตาราง…… ซ

สารบัญภาพ……... ฌ

บทที่

1. บทนำ…….. 1

1.1 ความสำคัญและความเป็นมาของปัญหา…….. 2

1.2 วัตถุประสงค์การวิจัย……... 3

1.3 ขอบเขตการวิจัย……... 4

1.4 ประโยชน์ที่คาดว่าจะได้รับ…… 4

2. แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง…….. 5

2.1 กลศาสตร์ดั้งเดิม (Classical Mechanics)…….. 6

2.2 อะตอม (Atom)…… 9

2.3 กลศาสตร์ควอนตัม (Quantum Mechanics)…….. 17

2.4 การซ้อนทับของควอนตัม (Quantum Superposition)…….. 19

2.5 คิวบิท (Qubit)…….. 19

2.6 สมการชเรอดิงเงอร์ (Schrödinger equation)……... 20

2.7 เทคโนโลยีควอนตัมคอมพิวเตอร์ (Quantum Computer)……. 23

2.8 โอเพ่นซอร์สซอฟต์แวร์ (Open Source Software)……... 23

2.9 Qiskit……... 25

2.10 Forest (pyQuil)…….. 25

2.11 Project Q……... 26

2.12 Quantum Developer Kit (Q#)……... 27

สารบัญ (ต่อ)

บทที่		หน้า
	2.13 ควอนตัมอัลกอริทึม (Quantum Algorithms)……	28
	2.14 อัลกอริทึมของชอร์ (Shor's algorithm)……	29
	2.15 ก้าวหน้าในการคานวณควอนตัม….....	32
	2.16 งานวิจัยที่เกี่ยวข้อง……...	33
3.	วิธีการดำเนินงาน……	41
	3.1 ภาพรวมของระบบและหน้าที่ในการทดลอง……...	41
	3.2 แผนการดำเนินงาน……..	43
	3.3 เครื่องมือที่ใช้ในงานวิจัย……	44
	3.4 ปัจจัยที่ส่งผลต่อการทางานของระบบ Quantum Software Platform……...	46
	3.5 วิธีการทดลอง……..	46
	3.6 บันทึกผลจากการทดลอง……..	73
4.	ผลการทดลอง……..	75
	4.1 ผลการทดสอบการเชื่อมต่อ……..	75
	4.2 ผลการทดสอบการติดตั้ง ไวยากรณ์ และภาษาที่รองรับ……..	76
	4.3 ผลการทดสอบฟังก์ชั่นและควอนตัมอัลกอริทึมที่ใช้งานได้……..	78
	4.4 ผลการทดสอบเปรียบเทียบเพื่อคัดเลือก 2 แพลตฟอร์ม……..	79
	4.5 ผลการทดสอบการเปรียบเทียบ 2 แพลตฟอร์มแบบวัดประสิทธิภาพการทำงานแบบจำลองโดยใช้ Shor's algorithm……..	79
5.	บทสรุปและคำแนะนำ……	87
	5.1 สรุปผลการวิจัย……	87
	5.2 ข้อจำกัดของงานวิจัย……	88
	5.3 ข้อเสนอแนะ……	88
บรรณานุกรม……..		89
ประวัติผู้เขียน……		93

สารบัญตาราง

ตารางที่ **หน้า**

3.1 แผนการดำเนินงาน..... 44

3.2 โครงสร้างพื้นฐานของเครื่องมือสำหรับใช้ทดสอบ..... 45

3.3 ตารางรายละเอียดภาพรวม Forest (pyQuil)..... 53

3.4 ตารางรายละเอียดภาพรวม Qiskit..... 55

3.5 ตารางรายละเอียดภาพรวม Project Q..... 57

3.6 ตารางรายละเอียดภาพรวม Quantum Developer Kit (Q#)..... 59

3.7 ตารางทดสอบ Function และ Quantum Algorithm ของ pyQuil..... 59

3.8 ตารางทดสอบ Function และ Quantum Algorithm ของ pyQuil.....…… 60

3.9 ตารางทดสอบ Function และ Quantum Algorithm ของ Project Q.....… 61

3.10 ตารางทดสอบ Function และ Quantum Algorithm ของ Q#..... 62

3.11 ตางรางความสามารถและความสะดวกในการใช้งานทั้ง 4 แพลตฟอร์ม..... 63

3.12 ตารางบันทึกผลการทดลอง..... 73

4.1 ตารางรวมผลทดลองรายละเอียดภาพรวม 4 แพลตฟอร์ม..... 76

4.2 ตางรางรวมผลทดสอบฟังก์ชั่นและควอนตัมอัลกอริทึทั้ง 4 แพลตฟอร์ม..... 78

4.3 ผลการทดสอบเปรียบเทียบเพื่อคัดเลือก 2 แพลตฟอร์ม..... 79

4.4 เปรียบเทียบด้วย Shor's algorithm ที่ N = 9 และ A = 7..... 79

4.5 เปรียบเทียบด้วย Shor's algorithm ที่ N = 15 และ A = 7..... 80

4.6 เปรียบเทียบด้วย Shor's algorithm ที่ N = 21 และ A = 7..... 81

4.7 เปรียบเทียบด้วย Shor's algorithm ที่ N = 25 และ A = 7..... 81

4.8 เปรียบเทียบด้วย Shor's algorithm ที่ N = 33 และ A = 7..... 82

4.9 เปรียบเทียบด้วย Shor's algorithm ที่ N = 35 และ A = 7..... 82

4.10 เปรียบเทียบด้วย Shor's algorithm ที่ N = 39 และ A = 7..... 83

4.11 เปรียบเทียบด้วย Shor's algorithm ที่ N = 55 และ A = 7..... 84

4.12 เปรียบเทียบด้วย Shor's algorithm ที่ N = 65 และ A = 7..... 84

4.13 เปรียบเทียบด้วย Shor's algorithm ที่ N = 77 และ A = 7..... 85

สารบัญภาพ

ภาพที่ **หน้า**

2.1 รูปอะตอมที่แสดงถึง อิเล็กตรอน นิวตรอน โปรตอน..... 10
2.2 Quantum Tunneling..... 17
2.3 เปรียบเทียบ bit กับ Qubit..... 20
2.4 ตัวอย่างกระบวนการพัฒนา Open-Source Software..... 24
2.5 Qiskit..... 25
2.6 Rigetti and Forest (pyQuil)..... 26
2.7 Project Q..... 27
2.8 Microsoft Quantum Developer Kit..... 28
2.9 The Quantum software lifecycle..... 38
3.1 ภาพของระบบและหน้าที่ในการทดลอง..... 42
3.2 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ Forest (pyQuil)..... 47
3.3 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ Qiskit..... 48
3.4 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ Project Q..... 49
3.5 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ Quantum Developer Kit (Q#) 50
4.1 ภาพรวมทดสอบระบบการต่อควอนตัมคอมพิวเตอร์ของ 4 แพลตฟอร์ม..... 76
4.2 รูปภาพค่าเฉลี่ยการทดสอบเปรียบเทียบประสิทธิภาพการทำงานแบบจำลอง
Qiskit และ Project Q..... 86

บทที่ 1
บทนำ

ในยุคสมัยที่มีการพัฒนาเทคโนโลยีอย่างรวดเร็วในปัจจุบันและพัฒนาซอฟต์แวร์แข่งขันอย่างแพร่หลายเป็นจำนวนมาก การพัฒนาซอฟต์แวร์ไม่ว่าจะเป็นเพื่อใช้งานในภายในองค์กรแต่ละองค์กรที่มีธุรกิจที่แตกต่างกันนั้นทำให้มีการแข่งขันการพัฒนาที่ค่อนข้างสูงเพื่อให้ได้ซอฟต์แวร์ที่มีประสิทธิภาพและทำประโยชน์สูงสุดให้แก่องค์กรในแต่ละธุรกิจตัวนักพัฒนาซอฟต์แวร์เองก็ต้องพัฒนาความสามารถในการเขียนโปรแกรมแต่ละภาษาเพื่อเท่าทันเทคโนโลยีใหม่ๆที่มีการปรับเปลี่ยนอยู่ตลอดเวลา

เมื่อแต่ละองค์กรต้องการประสิทธิภาพและความเร็วในการทำงานของซอฟต์แวร์เพื่อแก้ไขปัญหาเฉพาะเจาะจงและมีความซับซ้อนของแต่ละองค์กร ตัวนักพัฒนาซอฟแวร์เองก็ต้องมองหาเทคโนโลยีที่ทันสมัยและมีประสิทธิภาพ คู่ควรกับเวลาที่ทำการศึกษา และนำความรู้มาประยุกต์ใช้ได้อย่างรวดเร็ว เทคโนโลยีควอนตัมจึงเป็นอีกทางเลือกขององค์กรที่ต้องการพัฒนาธุรกิจและเป็นหลักสำคัญที่นักพัฒนาซอฟต์แวร์ควรให้ความสำคัญแก่เทคโนโลยีสมัยใหม่ ซึ่งนักพัฒนาซอฟต์แวร์ซอฟต์แวร์จะได้พัฒนาซอฟต์แวร์โดยใช้เทคโนโลยีนี้อย่างแน่นอนในอนาคต เทคโนโลยีควอนตัมจะมีบทบาทความสามารถในการแก้ไขปัญหาได้รวดเร็วกว่าคอมพิวเตอร์ทั่วไป ซึ่งถือเป็นวิวัฒนาการคอมพิวเตอร์แบบก้าวกระโดด และในอนาคตมี Quantum Software Platform ให้เลือกใช้งานเป็นจำนวนมากอีกทั้งแต่ละ Quantum Software Platform ยังมี Framework หลายตัวในหนึ่งค่ายอีกด้วย

ดังนั้นในการเลือกใช้งาน Quantum Software Platform เป็นสิ่งสำคัญมากในการเลือกใช้พัฒนาซอฟต์แวร์ว่ามีค่าใช้จ่ายอะไร การติดตั้ง Framework มี Algorithm อะไรที่รองรับแต่ละ Framework บ้าง ภาษาที่ใช้ในการพัฒนาซอฟแวร์สามารถใช้ความรู้เดิมในการพัฒนาได้หรือไม่เพื่อจะลดระยะเวลาในการเลือกใช้งาน รวมถึงเวลาในการศึกษาของตัวนักพัฒนาซอฟต์แวร์ด้วย

1.1 ความสำคัญและความเป็นมาของปัญหา

ปัจจุบันมีเทคโนโลยีมีการพัฒนาอย่างรวดเร็ว เทคโนโลยีหนึ่งในนั้นที่มีศักยภาพและเป็นเทคโนโลยีขั้นสูงที่มีโอกาสใช้ในงานธุรกิจเพื่อเพิ่มโอกาสและมูลค่าให้แก่บริษัท และเป็นเทคโนโลยีที่พลิกผัน (Disruptive Technology) นั่นคือ เทคโนโลยีควอนตัม (Quantum Technology) โดยเทคโนโลยีควอนตัมที่สามารถเสริมสร้างโอกาสทางธุรกิจโดยหลักๆจะประกอบด้วยสามองค์ประกอบคือ ควอนตัมคอมพิวเตอร์ (Quantum Computer) ควอนตัมเซนเซอร์ (Quantum Sensor) และการสื่อสารเชิงควอนตัม (Quantum Communication) โดยธุรกิจหลักๆจะเป็นในกลุ่มซอฟต์แวร์ เป็นธุรกิจหนึ่งที่สามารถใช้ควอนตัมคอมพิวเตอร์ในการหาแนวทางและนำเสนอแนวทางแก้ไขปัญหาเฉพาะหรือที่จำเป็น (Solution Provider) โดยใช้การคำนวณ ออกแบบอัลกอริทึม (Algorithm Design) จำลองระบบ รวมเข้ากับระบบการเรียนรู้จักรกล (Machine Learning) และระบบปัญญาประดิษฐ์ (AI: Artificial Intelligence) เพื่อนำมาแก้ไขปัญหาที่ยากเกินกว่าคอมพิวเตอร์คลาสสิกจะแก้ปัญหาได้ นอกเหนือจากธุรกิจซอฟต์แวร์ ยังมาธุรกิจอื่นๆที่สามารถนำเทคโนโลยีควอนตัมคอมพิวเตอร์ไปใช้ในการพัฒนาหรือปรับปรุงประสิทธิภายในองค์กรเช่น ธุรกิจโลจิสติกส์ ธุรกิจค้าปลีก และธุรกิจการผลิต โดยจะทำการวิเคราะห์และคำนวณเครือข่ายการจัดการสินค้าและบริการ ตลอดจนกำหนดเส้นทาง ลำดับการขนส่ง จัดการกลุ่มรถคมนาคม พยากรณ์การเดินทางเที่ยวบิน วิเคราะห์ความต้องการของลูกค้า ส่วนในธุรกิจสุขภาพและธุรกิจเภสัชกรรม โดยการใช้ระบบเครือข่ายคำนวณเพื่อการวินิจฉัยเชิงลึกและค้นคว้าหาตัวยาใหม่ที่ใช้ในการรักษาโรค ส่วนในด้านธุรกิจพลังงานและธุรกิจปิโตเคมี สามารถใช้ในการประเมินความคุ่มค่าแหล่งขุดน้ำมันใหม่ๆที่เจอ ในส่วนธุรกิจธนาคาร การเงิน ตลาดหลักทรัพย์ สามารถใช้ในการดำเนินการซื้อขายอัตโนมัติ กำหนดราคาตราอนุพันธ์ ประเมินความเสี่ยง แนะนำทางการเงินตลอดจนคาดเดาตลาดการลงทุนเพื่อใช้สำหรับด้านการวางแผนลงทุน

จะเห็นได้ว่าเทคโนโลยีควอนตัมมีพัฒนามาอย่างต่อเนื่องตั้งแต่ ยุค Quantum 1.0 ที่มีการใช้เทคโนโลยีเกี่ยวกับ Quantum ไม่ว่าจะเป็น Transistors Laser Nuclear และ Magnetic Resonance Imaging (MRI) หรือ การแสกนร่างกายด้วยการตอบสนองทาง Quantum ในยุค Quantum 1.0 ยังไม่สามารถใช้คุณสมบัติ Superposition และ Entanglement ได้อย่างเต็มที่ท่าไหร่หนัก จนมีนักวิทยาศาสตร์ 3 คน ได้แก่ คือ Steven Chu, Claude Cohen-Tannoudji และ William D. Phillip นำอนุภาค Quantum มาทดลองโดยควบคุม Quantum แบบอะตอมเดี่ยว ทำให้สามารถนำไปใช้ประโยชน์จากคุณสมบัติ Superposition และ Entanglement ได้มากขึ้น เมื่อมีการปลดคุณสมบัติความสามารถของ Superposition และ Entanglement ที่สามารถควบคุมและใช้งานได้มากขึ้น ก็สามารถสร้าง Disruption ในเทคโนโลยีได้มากเพิ่มขึ้น ซึ่งเป็นสิ่งคำคัญหลักในความก้าวหน้า

ในยุค Quantum 2.0 ที่สามารถแบ่งเป็นหัวเรื่องหลักตามความก้าวหน้าในแต่ละด้านอันได้แก่ Quantum Metrology, Quantum Communication และ Quantum Simulation & Computing ซึ่ง Quantum Simulation & Computing เป็นตัวหลักในการ Disruption ที่สามารถนำไปพัฒนาและต่อยอดใน Quantum 2.0 ให้มีความสมบูรณ์ เป็นการนำเอาคุณสมบัติด้าน Superposition (Jonathan et al., 2000) มาใช้ประโยชน์ ปกติแบบในระบบ Digital จะเป็นการใช้ค่า 0 และ 1 เพื่อแทนคำสั่งว่า Yes/No (มี/ไม่มี) ซึ่งการประมวลผลและมีหน่วยข้อมูลเป็น Bit แต่ในระบบ Quantum ที่อนุภาคปรากฏในหลายที่พร้อมกัน และ มีสถานะได้หลายสถานะ Quantum จะสามารถมีสถานะที่เป็นได้ทั้ง 0 และ 1 (Yes/No) ได้ในเวลาเดียวกัน ทำให้เราสามารถเปลี่ยนแปลงความสามารถในการประมวลผลคอมพิวเตอร์ หรือระบบการเรียนรู้อื่นได้กว้างมาก ซึ่งการจัดเก็บข้อมูลของ Quantum มีหน่วยข้อมูลในรูปแบบ Qubit และในปัจจุบันได้มีการพัฒนา Quantum Computer ที่สามารถทำงานบน Cloud และยังเกิด Quantum Software Platform ที่ใช้งาน Quantum Simulation และ Computing ผ่าน Quantum Software Platform อีกมากมายหลายตัวทำให้นำไปสู่อีกขั้นในการพัฒนาตัวโปรแกรม ซึ่งเป็นความท้าทายใหม่ของนักพัฒนาซอฟต์แวร์ในปัจจุบัน ที่ต้องมีการเริ่มต้นศึกษาความรู้ตั้งแต่เริ่มและความหลากหลายของ Quantum Software Platform ทำให้เกิดการตัดสินใจเลือกใช้งานที่ยากมากขึ้นว่าจะตอบโจทย์แก่นักพัฒนาหรือต่อยอดความรู้เดิมได้มากน้อยเพียงใด ซึ่งเกิดจากการเติบโตอย่างรวดเร็วของ Quantum Software Platform แต่เข้าสู่ยุคใหม่ในไม่ช้า

ดังนั้นจึงได้มองเห็นปัญหาที่เกิดขึ้นกับนักพัฒนาซอฟต์แวร์ (Developer) ว่าจะเลือกใช้ Quantum Software Platform ให้เหมาะสมกับการใช้งานหรือประยุกต์ใช้ความรู้เดิมของนักพัฒนาซอฟต์แวร์ โดยจะเปรียบเทียบภาพรวม Software ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) ซึ่งช่วยให้นักพัฒนาสามารถภาพรวม วิเคราะห์ข้อกำหนดและการติดตั้ง Framework (requirements and installation), ภาษาและรูปแบบการเขียนโปรแกรม (language syntax), สิ่งที่สนันสนุน (Library Support), การจำลองควอนตัม (Simulated Quantum), ภาษาแอสเซมบลีควอนตัม (Quantum assembly Language), ควอนตัมคอมไพเลอร์ (Quantum Compil) และวัดประสิทธิภาพการทำงานแบบจำลอง (Simulator Performance) โดยการใช้ Shor's Algorithm

1.2 วัตุประสงค์การวิจัย

เพื่อเปรียบเทียบการใช้งานและภาพรวมซอฟต์แวร์ควอนตัมแพลตฟอร์ม ทั้ง 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) เพื่อหาแพลตฟอร์มที่เหมาะสำหรับการศึกษาและพัฒนา

1.3 ขอบเขตการวิจัย

1.3.1 ทดสอบการใช้งานและภาพรวมของซอฟต์แวร์ควอนตัมแพลตฟอร์มทั้ง 4 แพลตฟอร์ม ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#)

1.3.2 ทดสอบประสิทธิภาพการทำงานแบบจำลองโดยใช้ Shor's Algorithm โดยเลือก 2 แพลตฟอร์มที่ดีที่สุดจาก 4 แพลตฟอร์มที่กล่าวมา

1.4 ประโยชน์ที่คาดหวังจะได้รับ

1.4.1 ได้รู้ถึงภาพรวมการใช้งานของซอฟต์แวร์ควอนตัมแพลตฟอร์มทั้ง 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#)

1.4.2 ได้แนวทางการเลือกใช้งานซอฟต์แวร์ควอนตัมแพลตฟอร์มในการศึกษาหรือพัฒนาโปรแกรมในอนาคต

บทที่ 2
แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

บทนำ

งานวิจัยนี้เป็นการศึกษารวมภาพรวมและเปรียบเทียบ Quantum Software Platform โดยจะนำเอา 2 Platform ที่ดีที่สุดมาวัดประสิทธิภาพแบบจำลอง (Simulator Performance) โดยใช้ Shor's Algorithm ในการวัดผลการทำงาน เพื่อเป็นแนวทางให้นักพัฒนาซอฟต์แวร์เลือกให้ Platform ในการพัฒนาซอฟต์แวร์ในอนาคต รู้ถึงประสิทธิภาพ ความยากง่ายในการใช้งาน และสามารถใช้ความรู้เดิมในการพัฒนาซอฟต์แวร์แบบ Classical Computer มาประยุกต์ใช้กับ Quantum computer เพื่อเป็นการเตรียมตัวสู่เทคโนโลยีใหม่ที่มี Algorithm แตกต่างไปจาก Classical Computer อย่างมาก แต่มีประสิทธิภาพที่สูงกว่ามากด้วยเช่นกัน

ข้อมูลทฤษฎีและงานวิจัยที่เกี่ยวข้องดังต่อไปนี้

2.1 กลศาสตร์ดั้งเดิม (classical mechanics)

2.2 อะตอม (Atom)

2.3 กลศาสตร์ควอนตัม (quantum mechanics)

2.4 การซ้อนทับของควอนตัม (Quantum Superposition)

2.5 คิวบิท (Qubit)

2.6 สมการชเรอดิงเงอร์ (Schrödinger equation)

2.7 เทคโนโลยีควอนตัมคอมพิวเตอร์ (Quantum Computer)

2.8 โอเพ่นซอร์สซอฟต์แวร์ (Open Source Software)

2.9 Qiskit

2.10 Forest (pyQuil)

2.11 ProjectQ

2.12 Quantum Developer Kit (Q#)

2.13 ควอนตัมอัลกอริทึม (Quantum Algorithms)

2.14 อัลกอริทึมของชอร์ (Shor's algorithm)

2.15 ก้าวหน้าในการคำนวณควอนตัม

2.16 งานวิจัยที่เกี่ยวข้อง

2.1 กลศาสตร์ดั้งเดิม (classical mechanics)

กลศาสตร์ดั้งเดิม หรือ กลศาสตร์นิวตัน (classical mechanics) เป็นหนึ่งในสองวิชาที่สำคัญที่สุดของกลศาสตร์ (โดยอีกวิชาหนึ่ง คือ กลศาสตร์ควอนตัม) ซึ่งอธิบายถึงการเคลื่อนที่ของวัตถุต่าง ๆ ภายใต้อิทธิพลจากระบบของแรง โดยวิชานี้ถือเป็นวิชาที่ครอบคลุมในด้านวิทยาศาสตร์ วิศวกรรม และเทคโนโลยีมากที่สุดวิชาหนึ่ง อีกทั้งยังเป็นวิชาที่เก่าแก่ ซึ่งมีการศึกษาในการเคลื่อนที่ของวัตถุตั้งแต่สมัยโบราณ โดยกลศาสตร์ดั้งเดิมรู้จักในวงกว้างว่า กลศาสตร์นิวตัน

ในทางฟิสิกส์ กลศาสตร์ดั้งเดิมอธิบายการเคลื่อนที่ของวัตถุขนาดใหญ่โดยแปลงการเคลื่อนที่ต่าง ๆ ให้กลายเป็นส่วนของเครื่องจักรกล เหมือนกันกับวัตถุทางดาราศาสตร์ อาทิ ยานอวกาศ ดาวเคราะห์ ดาวฤกษ์ และ ดาราจักร รวมถึงครอบคลุมไปยังทุกสถานะของสสารทั้งของแข็ง ของเหลว และแก๊ส โดยจะให้ผลลัพธ์ที่มีความแม่นยำสูง แต่เมื่อวัตถุมีขนาดเล็ก หรือมีความเร็วที่สูงใกล้เคียงกับความเร็วแสง กลศาสตร์ดั้งเดิมจะมีความถูกต้องที่ต่ำลง ต้องใช้กลศาสตร์ควอนตัมในการศึกษาแทนกลศาสตร์ดั้งเดิมเพื่อให้มีความถูกต้องในการคำนวณสูงขึ้น โดยกลศาสตร์ควอนตัมจะเหมาะสมที่จะศึกษาการเคลื่อนที่ของวัตถุที่มีขนาดเล็กมาก ซึ่งได้ถูกปรับแต่งให้เข้ากับลักษณะของอะตอมในส่วนของความเป็นคลื่น-อนุภาคในอะตอมและโมเลกุล แต่เมื่อกลศาสตร์ทั้งสองไม่สามารถใช้ได้ จากกรณีที่วัตถุขนาดเล็กเคลื่อนที่ด้วยความเร็วสูง ทฤษฎีสนามควอนตัมจึงเป็นตัวเลือกที่นำมาใช้ในการคำนวณแทนกลศาสตร์ทั้งสอง

คำว่า กลศาสตร์ดั้งเดิม (classical mechanics) ได้ถูกใช้เป็นครั้งแรกในช่วงต้นคริสต์ศตวรรษที่ 20 เพื่อกล่าวถึงระบบทางฟิสิกส์ของไอแซก นิวตันและนักปรัชญาธรรมชาติคนอื่นที่อยู่ร่วมสมัยในช่วงคริสต์ศตวรรษที่ 17 ประกอบกับทฤษฎีทางดาราศาสตร์ในช่วงแรกเริ่มของโยฮันเนส เคปเลอร์จากข้อมูลการสังเกตที่มีความแม่นยำสูงของทือโก ปราเออ และการศึกษาในการเคลื่อนที่ต่าง ๆ ที่อยู่บนโลกของกาลิเลโอ โดยมุมมองของฟิสิกส์ได้ถูกเปลี่ยนแปลงเรื่อยมาอย่างยาวนานก่อนที่จะมีทฤษฎีสัมพัทธภาพและกลศาสตร์ควอนตัม ซึ่งแต่เดิม ในบางแห่งทฤษฎีสัมพัทธภาพของไอน์สไตน์ไม่ถูกจัดอยู่ในกลศาสตร์ดั้งเดิม แต่อย่างไรก็ตามเมื่อเวลาผ่านไป หลายแห่งเริ่มจัดให้สัมพัทธภาพเป็นกลศาสตร์ดั้งเดิมในรูปแบบที่ถูกต้อง และถูกพัฒนามากที่สุด

แต่เดิมนั้น การพัฒนาในส่วนของกลศาสตร์ดั้งเดิมมักจะกล่าวถึงกลศาสตร์นิวตัน ซึ่งมีการใช้หลักการทางฟิสิกส์ประกอบกับวิธีการทางคณิตศาสตร์โดยนิวตัน ไลบ์นิซ และบุคคลอื่นที่เกี่ยวข้อง และวิธีการปกติหลายอย่างได้ถูกพัฒนา นำมาสู่การกำหนดกลศาสตร์ครั้งใหม่ ไม่ว่าจะเป็นกลศาสตร์แบบลากรางจ์ และกลศาสตร์แฮมิลตัน ซึ่งสิ่งเหล่านี้ได้ถูกพัฒนาขึ้นเป็นอย่างมากในช่วงคริสต์ศตวรรษที่ 18 และ 19 อีกทั้งได้ขยายความรู้เป็นอย่างมากพร้อมกับกลศาสตร์นิวตัน โดยเฉพาะอย่างยิ่งการนำกลศาสตร์เหล่านี้ไปใช้ในกลศาสตร์เชิงวิเคราะห์อีกด้วย

ในกลศาสตร์ดั้งเดิม วัตถุที่อยู่ในโลกของความเป็นจริงจะถูกจำลองให้อยู่ในรูปของอนุภาคจุด (วัตถุที่ไม่มีการอ้างอิงถึงขนาด) โดยเคลื่อนที่ของอนุภาคจุดจะมีการกำหนดลักษณะเฉพาะของวัตถุ ได้แก่ ตำแหน่งของวัตถุ มวล และแรงที่กระทำต่อวัตถุ ซึ่งจะกำหนดไว้เป็นตัวเลขที่อาจมีหน่วยกำหนดไว้ และกล่าวถึงมาเป็นลำดับ

เมื่อมองจากความเป็นจริง วัตถุต่าง ๆ ที่กลศาสตร์ดั้งเดิมกำหนดไว้ว่าวัตถุมีขนาดไม่เป็นศูนย์เสมอ (ซึ่งถ้าวัตถุที่มีขนาดเล็กมาก ๆ อย่างเช่น อิเล็กตรอน กลศาสตร์ควอนตัมจะอธิบายได้อย่างแม่นยำกว่ากลศาสตร์ดั้งเดิม) วัตถุที่มีขนาดไม่เป็นศูนย์จะมีความซับซ้อนในการศึกษามากกว่าอนุภาคจุดตามทฤษฎี เพราะวัตถุมีความอิสระของมันเอง (Degrees of freedom) อาทิ ลูกตะกร้อสามารถหมุนได้ขณะเคลื่อนที่หลังจากที่ถูกเตะขึ้นไปบนอากาศ อย่างไรก็ตาม ผลลัพธ์ของอนุภาคจุดสามารถใช้ในการศึกษาจำพวกวัตถุทั่วไปได้โดยสมมุติว่าเป็นวัตถุนั้น หรือสร้างอนุภาคจุดสมมุติหลาย ๆ จุดขึ้นมา ดังเช่นจุดศูนย์กลางมวลของวัตถุที่แสดงเป็นอนุภาคจุด

กลศาสตร์ดั้งเดิมใช้สามัญสำนึกเป็นแนวว่าสสารและแรงเกิดขึ้นและมีปฏิสัมพันธ์กันอย่างไร โดยตั้งสมมุติฐานว่าสสารและพลังงานมีความแน่นอน และมีคุณสมบัติที่รู้อยู่แล้ว ได้แก่ ตำแหน่งของวัตถุในปริภูมิ (Space) และความเร็วของวัตถุ อีกทั้งยังสามารถสมมุติว่ามีอิทธิพลโดยตรงกับสิ่งที่อยู่รอบวัตถุในขณะนั้นได้อีกด้วย (หรือเรียกอีกอย่างหนึ่งว่า Principle of locality) [1]

หลักการของกลศาสตร์ดั้งเดิม

เพื่อความง่ายในการวิเคราะห์ วัตถุที่อยู่ในโลกของความเป็นจริงจะถูกจำลองให้อยู่ในรูปของอนุภาคจุด (ไม่สนใจในขนาดของวัตถุ) โดยการเคลื่อนที่ของอนุภาคจุดจะมีการกำหนดเป็นพารามิเตอร์ที่มีค่าน้อย ได้แก่ ตำแหน่งของวัตถุ มวล และแรงที่กระทำต่อวัตถุ ซึ่งจะกำหนดไว้เป็นตัวเลขที่อาจมีหน่วยกำหนดไว้ และกล่าวถึงมาเป็นลำดับ

เมื่อมองจากความเป็นจริง วัตถุต่าง ๆ ที่กลศาสตร์ดั้งเดิมกำหนดไว้ว่าวัตถุมีขนาดไม่เป็นศูนย์เสมอ (ซึ่งถ้าวัตถุที่มีขนาดเล็ก*มาก ๆ* อย่างเช่น อิเล็กตรอน กลศาสตร์ควอนตัมจะอธิบายได้อย่างถูกต้องกว่ากลศาสตร์ดั้งเดิม) วัตถุที่มีขนาดไม่เป็นศูนย์จะมีความซับซ้อนในการศึกษามากกว่าอนุภาคจุดตามทฤษฎี เพราะวัตถุมีระดับความอิสระ (Degrees of freedom) ที่มาก อาทิ ลูกตะกร้อสามารถหมุนได้ขณะเคลื่อนที่หลังจากที่ถูกเตะขึ้นไปบนอากาศ อย่างไรก็ตาม ผลลัพธ์สำหรับอนุภาคจุดสามารถใช้ในการศึกษาจำพวกวัตถุทั่วไปได้โดยสมมุติว่าเป็นวัตถุนั้น หรือสร้างอนุภาคจุดสมมุติหลาย ๆ จุดขึ้นมา ดังเช่นจุดศูนย์กลางมวลของวัตถุที่แสดงเป็นอนุภาคจุด

กลศาสตร์ดั้งเดิมใช้สามัญสำนึกเป็นแนวว่าสสารและแรงเกิดขึ้นและมีปฏิสัมพันธ์กันอย่างไร โดยตั้งสมมุติฐานว่าสสารและพลังงานมีความแน่นอน และมีคุณสมบัติที่รู้อยู่แล้ว ได้แก่

ตำแหน่งของวัตถุในปริภูมิ (Space) และความเร็วของวัตถุ อีกทั้งยังสามารถสมมุติว่ามีอิทธิพลโดยตรงกับสิ่งที่อยู่รอบวัตถุในขณะนั้นได้อีกด้วย (หรือเรียกอีกอย่างหนึ่งว่า Principle of locality)

ตำแหน่งและอนุพันธ์ของตำแหน่ง

ตำแหน่ง ของอนุภาคจุดได้ถูกกำหนดตามจุดอ้างอิงที่กำหนดได้เองในปริภูมิ เรียกว่าจุดกำเนิด (Origin) ซึ่งในปริภูมิ จะให้ตำแหน่งอยู่ในระบบพิกัด โดยในระบบพิกัดอย่างง่ายมักกำหนดตำแหน่งวัตถุ และมีลูกศรที่มีทิศทางเป็นเวกเตอร์ในกลศาสตร์ดั้งเดิม โดยเริ่มจากจุดกำเนิดลากไปยังตำแหน่งของวัตถุ เช่น ตำแหน่ง r อยู่ในฟังก์ชันของ t (เวลา) ในสัมพัทธภาพช่วงก่อนไอน์สไตน์ (หรือเป็นที่รู้จักในชื่อ สัมพัทธภาพกาลิเลโอ) เวลาเป็นสิ่งสัมบูรณ์ คือ เวลาที่สังเกตมีระยะเท่ากันหมดในทุกผู้สังเกต ยิ่งไปกว่าเวลาสัมบูรณ์ กลศาสตร์ดั้งเดิมยังให้โครงสร้างของปริภูมิมีลักษณะโครงสร้างเป็นเรขาคณิตยูคลิดอีกด้วย

ความเร็วและอัตราเร็ว

ในกลศาสตร์ดั้งเดิม ความเร็วสามารถเพิ่มและลดได้โดยตรง ยกตัวอย่างเช่น ถ้ารถโดยสารประจำทางสายหนึ่งเดินทางด้วยความเร็ว 40 กม./ชม.ทิศตะวันตก แล้วมีรถจักรยานยนต์คันหนึ่งเดินทางด้วยความเร็ว 25 กม./ชม. ไปยังทิศตะวันออก เมื่อมองจากรถจักรยานยนต์ซึ่งมีอัตราเร็วต่ำกว่า รถโดยสารจะเดินทางด้วยความเร็ว 40-25 = 15 กม./ชม. ด้านทิศตะวันตก อีกด้านหนึ่ง ในด้านของรถโดยสารประจำทาง จะเห็นรถจักรยานเดินทางด้วยความเร็ว 15 กม./ชม. ด้านทิศตะวันออก ดังนั้นความเร็วสามารถเพิ่มหรือลดได้เป็นปริมาณเวกเตอร์ ซึ่งต้องจัดการโดยเวกเตอร์เชิงวิเคราะห์

$$\mathrm{v} = \frac{dr}{dt} \qquad (2\text{-}1)$$

ทำให้ได้สมการที่ (2-1) โดยกำหนดให้ v เป็นความเร็ว dr เป็นเวกเตอร์ระยะห่างของตำแหน่งเดิมและตำแหน่งใหม่ dt เป็นระยะเวลาที่ใช้เวลาเคลื่อนที่ไปยังตำแหน่งใหม่ ความเร็วหรือ อัตราการเปลี่ยนของตำแหน่งต่อเวลาได้นิยามไว้ด้วยอนุพันธ์เวลาของตำแหน่งตามสมการนี้

ในทางคณิตศาสตร์ ถ้าความเร็วของวัตถุแรกให้เป็น $\mathrm{u} = \mathrm{ud}$ และความเร็วของวัตถุที่สองให้เป็น $\mathrm{v} = \mathrm{ve}$ โดย v แ ละ u ป็นอัตราเร็วของวัตถุแรก และวัตถุที่สองตามลำดับ และ d กับ e เป็นเวกเตอร์หนึ่งหน่วยซึ่งแสดงถึงทิศทางการเคลื่อนที่ของวัตถุ ดังนั้นความเร็วของวัตถุแรกที่เห็นโดยวัตถุที่สอง คือ

$$u' = u - v$$

เช่นเดียวกับวัตถุที่หนึ่งที่มองกับวัตถุที่สอง

$$v' = v - u$$

เมื่อวัตถุเดินทางในทิศทางเดียวกัน สามารถทำสมการให้เป็นรูปอย่างง่าย ดังนี้

$$u' = (u - v)d$$

หรือถ้าไม่คำนึงถึงทิศทาง ความต่างนี้จะอยู่ในรูปของอัตราเร็วเท่านั้น ดังสมการนี้

$$u' = u - v$$

ความเร่ง

โดยความเร่งจะแสดงถึงความเร็วที่เปลี่ยนแปลงไปในช่วงเวลานั้น ๆ ไม่ว่าเป็นอัตราเร็ว ทิศทางของความเร็ว หรือทั้งสองอย่าง ซึ่งถ้าความเร็วลดลงไปเรื่อย ๆ เพียงอย่างเดียว

$$\mathrm{a} = \frac{dv}{dt} = \frac{d^2r}{dt^2} \qquad (2\text{-}2)$$

ซึ่งผลที่ได้จากการวิเคราะห์ตามสมการที่ (2-2) ก็สามารถเรียกได้ว่าความหน่วงเช่นกัน แต่ปกติแล้ว ทั้งความหน่วงและความเร่งมักถูกเรียกง่าย ๆ ว่าความเร่งเพียงอย่างเดียว

2.2 อะตอม **(Atom)**

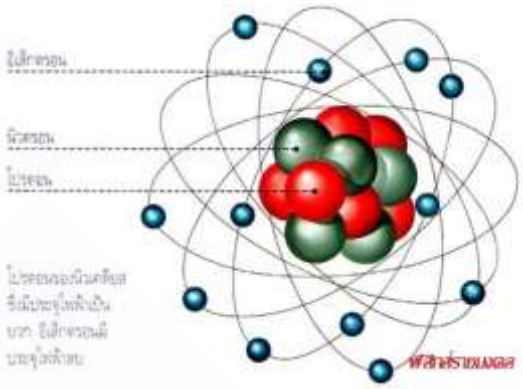
อะตอม (Atom) หมายถึง อนุภาคที่เล็กที่สุดของสสารที่ยังคงมีคุณสมบัติทางเคมีของธาตุนั้น ๆ อยู่ โครงสร้างของอะตอมประกอบด้วยนิวเคลียส ซึ่งอยู่ตรงกลางและมีอิเล็กตรอนโคจรอยู่รอบๆนิวเคลียส ภายในนิวเคลียสประกอบด้วยโปรตอนและนิวตรอนซึ่งในสภาวะปกติโครงสร้างของอะตอมจะมีจำนวน อิเล็กตรอนเท่ากับจำนวนโปรตอน

โปรตอน (Proton;p) หมายถึง อนุภาคที่มีประจุไฟฟ้าเป็นบวกหนึ่งหน่วย มีมวลประมาณ 1,837 เท่า ของอิเล็กตรอนซึ่งโปรตอนเป็นองค์ประกอบในนิวเคลียสของธาตุทุกชนิด

นิวตรอน (Neutron;n) หมายถึง อนุภาคที่ไม่มีประจุไฟฟ้า มีมวลมากกว่าโปรตอนเล็กน้อย

อิเล็กตรอน (Electron) หมายถึง อนุภาคที่มีประจุไฟฟ้าเป็นลบ โดยแต่ละอะตอมจะมีอิเล็กตรอน จำนวนหนึ่งอยู่ล้อมรอบนิวเคลียสของอะตอมนั้น

นิวตรอนและโปรตอนภายในนิวเคลียสถูกยึดให้อยู่ด้วยกันได้ด้วยแรงนิวเคลียร์ ซึ่งในสภาวะปกติภาย ในนิวเคลียสจะมีความเสถียร (Stable) จึงไม่มีการเปลี่ยนแปลงภายในนิวเคลียส หากภายในนิวเคลียส มีจำนวนโปรตอนหรือนิวตรอนที่มากหรือน้อยเกินไป เรียกสถานะเช่นนี้ว่าเกิดความไม่เสถียร (Unstable) นิวเคลียสจะปรับตัวเองเพื่อให้อยู่ในสภาวะเสถียรและมีการปล่อยพลังงานออกมาในรูปแบบต่างๆ ซึ่งเป็น ที่มาของคำว่าพลังงานปรมาณู [2]



ภาพที่ 2.1 รูปอะตอมที่แสดงถึง อิเล็กตรอน นิวตรอน โปรตอน

หมายเหตุ: จาก https://sites.google.com/site/atom11062541

ส่วนประกอบของอะตอม

อนุภาคที่เล็กกว่าอะตอม แม้คำว่า อะตอม จะมีกำเนิดจากรากศัพท์ที่มีความหมายถึงอนุภาคที่เล็กที่สุด ซึ่งไม่สามารถแบ่งได้อีกต่อไป แต่การใช้งานในทางวิทยาศาสตร์สมัยใหม่นั้น อะตอมยังประกอบด้วยอนุภาคที่เล็กกว่าอะตอมอีกมากมาย อนุภาคที่เป็นส่วนประกอบของอะตอม ได้แก่ อิเล็กตรอน โปรตอน และนิวตรอน อย่างไรก็ดี อะตอมของไฮโดรเจน-1 นั้นไม่มีนิวตรอน และประจุไฮโดรเจนบวกก็ไม่มีอิเล็กตรอน

เท่าที่รู้กันในปัจจุบัน อิเล็กตรอนเป็นอนุภาคที่มีมวลน้อยที่สุดในบรรดาอนุภาคทั้งหมด คือประมาณ 9.11×10−31 kg โดยมีประจุไฟฟ้าลบและมีขนาดที่เล็กเกินกว่าจะวัดได้ด้วยเทคนิคเท่าที่มีอยู่ โปรตอนมีประจุบวก และมีมวลราว 1,836 เท่าของอิเล็กตรอน คือประมาณ 1.6726×10−27 kg แม้ว่าอาจลดลงได้จากการเปลี่ยนแปลงพลังงานยึดเหนี่ยวของโปรตอนที่มีต่ออะตอม ส่วนนิวตรอนนั้นไม่มีประจุไฟฟ้า มีมวลราว 1,839 เท่าของมวลอิเล็กตรอนหรือ 1.6929×10−27 kg นิวตรอนกับโปรตอนมีขนาดพอ ๆ กันที่ประมาณ 2.5×10−15 ม. แม้ว่า 'พื้นผิว' ของอนุภาคเหล่านี้จะไม่สามารถระบุได้อย่างชัดเจนก็ตาม

ในแบบจำลองมาตรฐานทางฟิสิกส์ ทั้งโปรตอนและนิวตรอนต่างประกอบด้วยอนุภาคมูลฐานเรียกว่า ควาร์ก ควาร์กเป็นหนึ่งในสองชนิดของกลุ่มอนุภาคเฟอร์มิออนซึ่งเป็นองค์ประกอบพื้นฐานของสสาร องค์ประกอบอีกตัวหนึ่งคือเลปตอน ซึ่งมีอิเล็กตรอนเป็นส่วนประกอบ ควาร์กมีอยู่ 6 ประเภท แต่ละประเภทมีประจุไฟฟ้าเป็นเศษส่วนที่แตกต่างกันคือ +2/3 หรือ −1/3 โปรตอนประกอบด้วยอัพควาร์ก 2 ตัวและดาวน์ควาร์ก 1 ตัว ขณะที่นิวตรอนประกอบด้วยอัพควาร์ก 1 ตัว

และดาวน์ควาร์ก 2 ตัว ความแตกต่างนี้เป็นตัวบ่งบอกถึงความแตกต่างของมวลและประจุระหว่างอนุภาคสองชนิด ควาร์กยึดเหนี่ยวกันไว้ได้ด้วยแรงนิวเคลียร์อย่างเข้ม ซึ่งเป็นลักษณะของกลูออน กลูออนเป็นอนุภาคชนิดหนึ่งในตระกูลเกจโบซอน ซึ่งเป็นอนุภาคมูลฐานทางด้านแรงปฏิกิริยาทางฟิสิกส์ [3]

นิวเคลียส

โปรตอนกับนิวตรอนที่ยึดเหนี่ยวกันภายในอะตอมหนึ่ง ๆ จะรวมกันเป็นนิวเคลียสอะตอมขนาดเล็ก เรียกชื่อว่า นิวคลีออน รัศมีของนิวเคลียสมีค่าประมาณ $1.07\sqrt[3]{A}$ fm โดยที่ A คือจำนวนนิวคลีออนรวม นี่มีขนาดเล็กกว่ารัศมีของอะตอมมาก ประมาณ 105 fm นิวคลีออนยึดเหนี่ยวกันเอาไว้ด้วยแรงดึงดูดระยะใกล้ ๆ เรียกว่า แรงนิวเคลียร์ ที่ระยะห่างต่ำกว่า 2.5 fm แรงนี้จะมีพลังแรงมากกว่าแรงไฟฟ้าสถิตซึ่งทำให้ประจุโปรตอนที่เป็นบวกพยายามผลักตัวออกจากกัน

อะตอมของธาตุชนิดเดียวกันจะมีจำนวนโปรตอนเท่ากัน เรียกตัวเลขนี้ว่า หมายเลขอะตอม ในธาตุชนิดหนึ่ง ๆ อาจมีจำนวนนิวตรอนที่แตกต่างกัน ซึ่งเป็นตัวกำหนดค่าไอโซโทปของธาตุนั้น ๆ จำนวนโดยรวมของโปรตอนกับนิวตรอนเป็นตัวระบุนิวไคลด์ จำนวนนิวตรอนเทียบกับโปรตอนเป็นตัวกำหนดความเสถียรของนิวเคลียส และไอโซโทปที่ทำให้เกิดการสลายตัวของสารกัมมันตรังสี

นิวตรอนกับโปรตอนต่างเป็นเฟอร์มิออนเพียงแต่เป็นคนละชนิด ตามหลักการกีดกันของเพาลี คือผลจากแรงควอนตัมทำให้เฟอร์มิออน ที่เทียบเท่ากัน ไม่สามารถมีสถานะควอนตัมเดียวกันในเวลาเดียวกันได้ ดังนั้นโปรตอนทุกตัวในนิวเคลียสจึงต้องดำรงอยู่ในสถานะที่แตกต่างกันด้วยระดับพลังงานต่าง ๆ ของตัวเอง กฎเดียวกันนี้ยังใช้กับนิวตรอนทั้งหมดด้วย แต่ไม่ได้ห้ามโปรตอนกับนิวตรอนให้มีสถานะควอนตัมอันเดียวกัน

มีความเป็นไปได้ที่นิวเคลียสของอะตอมที่มีหมายเลขอะตอมต่ำซึ่งมีจำนวนโปรตอนกับนิวตรอนต่างกัน จะลดสถานะพลังงานต่ำลงจากการสลายตัวของสารกัมมันตรังสี อันทำให้จำนวนโปรตอนกับนิวตรอนเกือบจะเท่ากัน ผลที่เกิดขึ้นทำให้อะตอมที่มีจำนวนโปรตอนกับนิวตรอนเกือบเท่ากันนั้นมีภาวะเสถียรขึ้นและไม่สลายตัว อย่างไรก็ดี ยิ่งหมายเลขอะตอมสูงขึ้นแรงผลักระหว่างโปรตอนก็จะยิ่งทำให้สัดส่วนนิวตรอนที่ต้องมีเพื่อรักษานิวเคลียสให้เสถียรต้องเพิ่มจำนวนมากขึ้น ทำให้แนวโน้มเปลี่ยนแปลงไป ด้วยเหตุนี้จึงไม่มีนิวเคลียสที่เสถียรซึ่งมีจำนวนโปรตอนและนิวตรอนเท่า ๆ กันที่หมายเลขอะตอมมากกว่า Z = 20 (แคลเซียม) ยิ่ง Z มีจำนวนมากขึ้นไปสู่นิวเคลียสธาตุหนัก สัดส่วนนิวตรอนต่อโปรตอนที่ต้องมีเพื่อความเสถียรจะยิ่งเพิ่มขึ้นไปที่ประมาณ 1.5

จำนวนโปรตอนและนิวตรอนในนิวเคลียสอะตอมสามารถเปลี่ยนแปลงได้ แม้จะต้องใช้พลังงานสูงมากเพราะมีแรงยึดเหนี่ยวที่เข้มมาก ปฏิกิริยานิวเคลียร์ฟิวชั่นเกิดขึ้นเมื่ออนุภาคอะตอมหลายตัวรวมตัวกันทำให้เกิดเป็นนิวเคลียสใหม่ที่หนักกว่าเดิม เช่นจากการชนกันของนิวเคลียสสองตัว ยกตัวอย่างเช่นที่แกนกลางของดวงอาทิตย์ โปรตอนต้องการพลังงาน 3–10 keV เพื่อเอาชนะแรงยึดเหนี่ยวระหว่างกัน หรือ กำแพงคูลอมบ์ (coulomb barrier) แล้วหลอมรวมเข้าด้วยกันกลายเป็นนิวเคลียสเพียงอันเดียว ส่วนปฏิกิริยานิวเคลียร์ฟิชชันจะเกิดขึ้นในทางตรงกันข้าม คือการที่นิวเคลียสหนึ่งแตกตัวออกเป็นนิวเคลียสขนาดเล็กกว่า 2 ตัว โดยมากเกิดขึ้นจากการสลายตัวของสารกัมมันตรังสี นิวเคลียสยังอาจเปลี่ยนแปลงได้จากการยิงด้วยอนุภาคขนาดเล็กพลังงานสูง หรือโฟตอน ถ้าสามารถเปลี่ยนแปลงจำนวนโปรตอนในนิวเคลียสได้ อะตอมก็จะเปลี่ยนคุณลักษณะไปเป็นธาตุชนิดอื่น

ถ้ามวลของนิวเคลียสหลังจากเกิดปฏิกิริยาฟิวชั่นมีน้อยกว่าจำนวนมวลรวมของอนุภาคขณะที่ยังแยกกัน มวลที่แตกต่างกันระหว่างค่าทั้งสองอาจจะแพร่ออกไปในลักษณะของพลังงานบางอย่าง (เช่น รังสีแกมมา หรือพลังงานจลน์ของอนุภาคบีตา) ดังที่อัลเบิร์ต ไอน์สไตน์ อธิบายไว้ในสมการสมมูลระหว่างมวล-พลังงาน E = mc2 เมื่อ m คือมวลที่สูญหายไป และ c คือความเร็วแสง จำนวนที่หายไปนี้เป็นส่วนหนึ่งของพลังงานยึดเหนี่ยวของนิวเคลียสใหม่ และเป็นการสูญเสียพลังงานแบบไม่มีวิธีย้อนกลับ ซึ่งทำให้อนุภาคที่หลอมรวมกันยังคงอยู่ในสถานะที่จำเป็นต้องใช้พลังงานในระดับนั้นเพื่อแยกตัวออกจากกัน

การเกิดฟิวชั่นของนิวเคลียสสองตัวให้กลายเป็นนิวเคลียสเดียวที่ใหญ่ขึ้น โดยที่มีหมายเลขอะตอมต่ำกว่าเหล็กและนิกเกิล หรือจำนวนนิวคลีออนรวมประมาณ 60 เรียกว่ากระบวนการคายความร้อน ซึ่งจะปลดปล่อยพลังงานออกมามากกว่าพลังงานที่ต้องใช้ในการรวมตัวกันกระบวนการปลดปล่อยพลังงานเช่นนี้เองที่ทำให้เกิดปฏิกิริยานิวเคลียร์ฟิวชั่นในดาวฤกษ์ซึ่งสามารถเกิดขึ้นอย่างต่อเนื่อง สำหรับนิวเคลียสธาตุหนัก พลังงานยึดเหนี่ยวต่อนิวคลีออนในนิวเคลียสเริ่มต้นลดจำนวนลง นั่นคือกระบวนการฟิวชั่นที่สร้างนิวเคลียสที่มีหมายเลขอะตอมสูงกว่า 26 และมวลอะตอมมากกว่า 60 เรียกว่ากระบวนการดูดความร้อน นิวเคลียสมวลมากเหล่านี้ไม่สามารถสร้างปฏิกิริยาฟิวชั่นต่อเนื่องที่รักษาสภาวะสมดุลอุทกสถิตของดาวฤกษ์เอาไว้ได้

กลุ่มหมอกอิเล็กตรอน

อิเล็กตรอนในอะตอมถูกดึงดูดเอาไว้กับโปรตอนในนิวเคลียสด้วยแรงแม่เหล็กไฟฟ้า แรงชนิดนี้ยึดเหนี่ยวอิเล็กตรอนเอาไว้ภายในหลุมพลังงานไฟฟ้าสถิตที่อยู่รอบ ๆ นิวเคลียส นั่นแสดงว่าจำเป็นต้องได้รับพลังงานจากแหล่งภายนอกเพื่อช่วยให้อิเล็กตรอนหนี

ออกไปได้ ยิ่งอิเล็กตรอนอยู่ใกล้กับนิวเคลียสมากเท่าใด แรงดึงดูดนี้ก็ยิ่งมากขึ้น ดังนั้นอิเล็กตรอนที่อยู่ใกล้ศูนย์กลางของหลุมพลังงานจำเป็นต้องใช้พลังงานมากกว่าเพื่อจะหนีออกมาได้

เช่นเดียวกับอนุภาคอื่น อิเล็กตรอนมีคุณสมบัติแบบทวิภาค คือเป็นทั้งอนุภาคและเป็นทั้งคลื่น เมฆอิเล็กตรอนเป็นบริเวณภายในหลุมพลังงานที่อิเล็กตรอนแต่ละตัวจะสร้างคลื่นนิ่ง 3 มิติประเภทหนึ่งขึ้น อันเป็นรูปคลื่นที่ไม่เคลื่อนที่ตามนิวเคลียส พฤติกรรมนี้ถูกกำหนดจากออร์บิทัลของอะตอม ซึ่งเป็นฟังก์ชันคณิตศาสตร์ที่แสดงคุณสมบัติความเป็นไปได้ที่อิเล็กตรอนจะปรากฏตัวขึ้นที่จุดเฉพาะหนึ่ง ๆ ขณะที่ถูกวัดตำแหน่ง รอบ ๆ นิวเคลียสจะมีออร์บิทัลที่ไม่ต่อเนื่องกันล้อมรอบอยู่ในลักษณะของควอนตา ทั้งนี้เพราะรูปแบบคลื่นอื่นที่เป็นไปได้จะสลายตัวไปอย่างรวดเร็วเข้าสู่สถานะที่เสถียรมากกว่า ออร์บิทัลอาจมีลักษณะวงแหวนหนึ่งวง หลายวง หรือเป็นโครงสร้างโหนดก็ได้ ซึ่งมีความแตกต่างจากออร์บิทัลอื่น ๆ ทั้งด้านขนาด รูปร่าง และศูนย์กลาง

ออร์บิทัลอะตอมแต่ละแบบจะสอดคล้องกับระดับพลังงานเฉพาะของอิเล็กตรอนค่าหนึ่ง ๆ อิเล็กตรอนสามารถเปลี่ยนสถานะของมันไปยังระดับพลังงานที่สูงกว่าได้โดยการดูดซับโฟตอนที่มีพลังงานเพียงพอจะยกระดับตัวมันขึ้นไปสู่สถานะควอนตัมใหม่ ในทางกลับกันกระบวนการปลดปล่อยรังสีด้วยตัวเองทำให้อิเล็กตรอนที่ระดับพลังงานสูงสามารถลดระดับพลังงานลงไปยังสถานะที่ต่ำกว่าได้ขณะที่แผ่พลังงานส่วนเกินออกไปเป็นโฟตอน คุณลักษณะของค่าพลังงานที่กำหนดจากสถานะควอนตัมที่แตกต่างกันนี้เป็นสาเหตุของการเกิดเส้นสเปกตรัม

ปริมาณพลังงานที่จำเป็นต้องใช้ (ทั้งแบบเพิ่มเข้าไปหรือปลดปล่อยออกมา) ในการเปลี่ยนสถานะของอิเล็กตรอนนี้น้อยกว่าพลังงานยึดเหนี่ยวของนิวคลีออนมาก เช่น จำเป็นต้องใช้พลังงานเพียง 13.6 eV เพื่อให้อิเล็กตรอนจากอะตอมของไฮโดรเจนเปลี่ยนระดับลงไปยังสถานะพื้น เทียบกับพลังงาน 2.23 ล้าน eV ในการแยกนิวเคลียสของดิวเทอเรียม อะตอมมีประจุเป็นกลางถ้ามันมีจำนวนโปรตอนกับอิเล็กตรอนเท่ากัน อะตอมที่มีอิเล็กตรอนมากหรือน้อยกว่าปกติเรียกว่า ไอออน อิเล็กตรอนที่อยู่ไกลจากนิวเคลียสมากอาจถ่ายโอนไปยังอะตอมข้างเคียง หรืออยู่ร่วมระหว่างสองอะตอมก็ได้ ด้วยกลไกนี้ อะตอมจึงสามารถเกิดพันธะเคมีกลายเป็นโมเลกุลและสารประกอบเคมีอื่น ๆ เช่น การเกิดผลึกแบบไอโอนิกคริสตัลหรือโคเวเลนต์

โครงสร้างของเมฆอิเล็กตรอนอาจเปลี่ยนแปลงไปตามจำนวนอิเล็กตรอนที่มีในกลุ่มเมฆนั้น มีวิธีการนับจำนวนอิเล็กตรอนที่แตกต่างกันอยู่จำนวนหนึ่ง เช่น กฎออกเตต หรือ กฎ 18 อิเล็กตรอน ซึ่งโดยมากจะใช้เป็นเพียงกฎช่วยจำและไม่ได้ใช้แบบเดียวกันกับอะตอมทุกชนิด นักศึกษาใหม่ในวิชาเคมีมักถูกสอนให้จำโครงสร้างอะตอมแบบง่าย ๆ เป็น 2, 8, 8, 8, 8, 8, 8, [...] ทั้งนี้เพื่อให้ลำดับการสอนทำได้ง่ายขึ้น แต่จำนวนอิเล็กตรอนในแต่ละเชลล์สำหรับอะตอมขนาด

ใหญ่ที่จริงแล้วมีจำนวนที่ต่างไปจากนี้ เช่น 2, 8, 18, 32, 50, 72 แต่ต้องเป็นนักศึกษาชั้นสูงจึงค่อยทำความเข้าใจกับความซับซ้อนนี้

คุณสมบัติ

คุณสมบัติทางนิวเคลียร์ตามคำนิยามแล้ว อะตอมสองตัวที่มีจำนวนโปรตอนในนิวเคลียสเท่ากัน จะเป็นอะตอมของธาตุชนิดเดียวกัน อะตอมที่มีจำนวนโปรตอนเท่ากัน แต่มีจำนวน นิวตรอน แตกต่างกันจัดว่าเป็นไอโซโทปของธาตุเดียวกัน ตัวอย่างเช่น อะตอมของไฮโดรเจนทั้งหมดจะมีโปรตอน 1 ตัวเหมือนกัน แต่ไอโซโทปของไฮโดรเจนมีหลายชนิด ตั้งแต่แบบไม่มีนิวตรอน คือ ไฮโดรเจน-1, แบบนิวตรอน 1 ตัว (ดิวเทอเรียม), แบบนิวตรอน 2 ตัว (ทริเทียม) และที่มีนิวตรอนมากกว่า 2 ตัว ไฮโดรเจน-1 เป็นรูปแบบที่พบกันแพร่หลายมากที่สุด บางคราวก็เรียกว่า โปรเทียม ธาตุที่เรารู้จักแล้วมีกลุ่มหมายเลขอะตอมตั้งแต่ไฮโดรเจน ซึ่งมีโปรตอน 1 ตัว ไปจนถึง อูนอูนออกเทียม ซึ่งเป็นธาตุที่มีโปรตอน 118 ตัว ไอโซโทปของธาตุทั้งหมดที่เรารู้จักที่มีหมายเลขอะตอมมากกว่า 82 จัดเป็นสารกัมมันตรังสี

มีนิวไคลด์อยู่ 339 ชนิดที่เกิดขึ้นตามธรรมชาติบนโลก ในจำนวนนี้ 256 ชนิด (ประมาณ 76%) ไม่พบการสลายตัว ซึ่งจะเรียกว่าเป็นไอโซโทปเสถียร ในบรรดาธาตุ 80 ชนิดจะมีไอโซโทปเสถียรอย่างน้อย 1 ตัว สำหรับธาตุหมายเลข 43, 61, และทุกธาตุที่หมายเลข 83 หรือสูงกว่าไม่มีไอโซโทปที่เสถียร อาจกำหนดเป็นกฎได้ว่า สำหรับธาตุทุกชนิด มีไอโซโทปเสถียรอยู่เพียงจำนวนน้อยนิด เฉลี่ยมีไอโซโทปเสถียรประมาณ 3.1 ตัวต่อธาตุที่มีไอโซโทปเสถียร มีธาตุอยู่ 27 ชนิดที่มีไอโซโทปเสถียรเพียงตัวเดียว ขณะที่จำนวนไอโซโทปเสถียรมากที่สุดเท่าที่เคยพบคือ 10 โดยพบในดีบุก

ความเสถียรของไอโซโทปเกิดจากสัดส่วนระหว่างโปรตอนต่อนิวตรอน รวมไปถึง "จำนวนมหัศจรรย์" ของนิวตรอนหรือโปรตอนที่แสดงถึงระดับพลังงานควอนตัมทั้งแบบ closed และแบบ filled ระดับชั้นพลังงานควอนตัมเหล่านี้คือระดับพลังงานภายในแบบจำลองชั้นพลังงานของนิวเคลียส ดังเช่น filled shell ของโปรตอน 50 ตัวในดีบุก แสดงถึงความเสถียรของนิวไคลด์แบบไม่ปกติ จากจำนวนนิวไคลด์เสถียรทั้งหมดที่รู้จักกัน 256 ชนิด มีเพียง 4 ชนิดเท่านั้นที่มีจำนวนโปรตอนและนิวตรอนเป็นเลขคี่ ได้แก่ ไฮโดรเจน-2 (ดิวเทอเรียม), ลิเธียม-6, โบรอน-10 และ ไนโตรเจน-14 นอกจากนี้ สำหรับนิวไคลด์กัมมันต์แบบคี่-คี่ ที่มีครึ่งชีวิตมากกว่าพันล้านปีก็มีเพียง 4 ชนิดเท่านั้นคือ โปแตสเซียม-40, วานาเดียม-50, แลนทานัม-138 และ แทนทาลัม-180m นิวเคลียสที่มีจำนวนแบบคี่-คี่ ส่วนใหญ่จะไม่เสถียรอย่างมากโดยเกิดการสลายปลดปล่อยอนุภาคบีตา เพราะผลจากการสลายนั้นจะได้จำนวนมาเป็นแบบคู่-คู่ ซึ่งเป็นพันธะที่แข็งแรงกว่าตาม nuclear pairing effects

มวล

เนื่องจากมวลส่วนมากของอะตอมอยู่ในโปรตอนและนิวตรอน ดังนั้นจำนวนรวมของอนุภาคเหล่านี้ (เรียกรวมกันว่า "นิวคลีออน") ในอะตอมหนึ่ง ๆ จึงเรียกว่าเป็น เลขมวล โดยมากมักจะแสดงมวลนิ่งโดยใช้หน่วยมวลอะตอม (u) ซึ่งบางครั้งก็เรียกว่า ดาลตัน (Da) หน่วยนี้นิยามจาก 1 ส่วน 12 ของมวลของอะตอมอิสระที่เป็นกลางของคาร์บอน-12 ซึ่งมีค่าประมาณ 1.66×10^{-27} kg ไฮโดรเจน-1 ซึ่งเป็นไอโซโทปที่เบาที่สุดของไฮโดรเจนและเป็นอะตอมที่มีมวลน้อยที่สุด มีน้ำหนักอะตอมเท่ากับ 1.007825 u อะตอมหนึ่ง ๆ จะมีมวลโดยประมาณเท่ากับเลขมวลคูณด้วยหน่วยมวลอะตอม อะตอมเสถียรที่หนักที่สุด คือ ตะกั่ว-208 ซึ่งมีมวล 207.9766521 u

ถึงแม้จะเป็นอะตอมที่มีมวลมากที่สุด มันก็ยังเบาเกินกว่าที่เราจะไปทำอะไรด้วยโดยตรงได้ นักเคมีจึงนิยมใช้หน่วย โมล แทน โมลมีนิยามว่า หนึ่งโมลของธาตุใด ๆ จะมีจำนวนเท่ากับอะตอมเสมอ (ประมาณ 6.022×1023) ที่เลือกใช้จำนวนนี้ก็เพื่อว่า ถ้าธาตุใด ๆ มีเลขอะตอมเป็น 1 u แล้ว โมลอะตอมของธาตุนั้นจะมีมวลใกล้เคียงกับ 0.001 กก. หรือ 1 กรัม อาศัยคำนิยามของหน่วยมวลอะตอมนี้ คาร์บอน-12 จึงมีมวลอะตอมเท่ากับ 12 u พอดี และหนึ่งโมลของอะตอมคาร์บอนมีน้ำหนักเท่ากับ 0.012 กก.

รูปร่างและขนาด

เราไม่สามารถบอกขอบเขตที่แน่นอนของอะตอมได้ การบอกขนาดของอะตอมจึงมักอธิบายในลักษณะรัศมีอะตอม คือการวัดระยะห่างของเมฆอิเล็กตรอนที่แผ่ออกไปจากนิวเคลียส อย่างไรก็ดี การบอกระยะห่างเช่นนี้อยู่บนสมมุติฐานว่าอะตอมมีรูปร่างเป็นทรงกลม ซึ่งจะเป็นจริงก็ต่อเมื่ออะตอมนั้นอยู่ในสุญญากาศหรืออวกาศเสรีเท่านั้น รัศมีอะตอมหาได้จากระยะห่างระหว่างนิวเคลียสอะตอม 2 ตัวที่ดึงดูดกันอยู่ในพันธะเคมี มีค่าแปรเปลี่ยนไปตามแต่ตำแหน่งของอะตอมบนแผนผังอะตอม หรือตามชนิดของพันธะเคมี จำนวนอะตอมเพื่อนบ้าน (เลขโคออร์ดิเนชัน) และคุณสมบัติทางควอนตัมที่เรียกว่า สปิน ตามที่ปรากฏในตารางธาตุ ขนาดอะตอมมีแนวโน้มเพิ่มสูงขึ้น ถ้ายิ่งอยู่ในคอลัมน์ที่ต่ำลงไปข้างล่างแต่มีขนาดเล็กลง หากเปรียบเทียบจากด้านซ้ายไปขวาเทียบกันแล้ว อะตอมที่มีขนาดเล็กที่สุดคือ อะตอมของฮีเลียมซึ่งมีรัศมี 32 พิโคเมตร ส่วนอะตอมใหญ่ที่สุดคือ ซีเซียม มีขนาด 225 พิโคเมตร

ถ้าอะตอมอยู่ในสนามพลังงานอื่น ๆ เช่น สนามไฟฟ้า รูปร่างของอะตอมจะบิดเบี้ยวไปไม่เป็นทรงกลม การเปลี่ยนแปลงขึ้นอยู่กับขนาดความแรงของสนามพลังงานและชนิดวงโคจรของอิเล็กตรอนในเชลล์นอกสุด ซึ่งแสดงไว้ในทฤษฎีกรุป การเปลี่ยนแปลงรูปร่างอื่น ๆ เช่นรูปทรงคริสตัล เกิดขึ้นจากสนามพลังงานไฟฟ้า-คริสตัล ในการก่อตัวแบบสมมาตรต่ำ มีรูปทรงโดดเด่นอีกแบบหนึ่งคือทรงรี เกิดขึ้นกับไอออนซัลเฟอร์ในสารประกอบประเภท pyrite

มิติของอะตอมนั้นเล็กกว่าความยาวคลื่นของแสง (400-700 นาโนเมตร) นับหลายพันเท่า จึงไม่สามารถมองดูอะตอมด้วยกล้องจุลทรรศน์แบบใช้แสงได้ อย่างไรก็ดี เราสามารถสังเกตการณ์อะตอมเดี่ยว ๆ ได้โดยใช้กล้องจุลทรรศน์แบบส่องกราดในอุโมงค์ ซึ่งบางตัวอย่างอาจแสดงให้เห็นความเล็กจิ๋วของอะตอมได้ เส้นผมของมนุษย์มีขนาดความกว้างประมาณ 1 ล้านเท่าของอะตอมคาร์บอน หยดน้ำหนึ่งหยดมีอะตอมออกซิเจนอยู่ประมาณ 2 เซ็กซ์ทิลเลียน (2×1021) และอะตอมไฮโดรเจนอีก 2 เท่าของจำนวนนี้ เพชร 1 กะรัต มีมวล 2×10−4 กิโลกรัม ประกอบด้วยอะตอมคาร์บอนจำนวน 10 เซ็กซ์ทิลเลียน (1022) อะตอม ถ้าเราขยายขนาดผลแอปเปิ้ลให้ใหญ่เท่าขนาดของโลก หนึ่งอะตอมในแอปเปิ้ลจะมีขนาดประมาณผลแอปเปิ้ลปกติ

กฎธรรมชาติที่เหนือสามัญสำนึกของอะตอม

กฎที่อธิบายการเคลื่อนไหวของสิ่งต่างๆ ที่เราเห็นในชีวิตประจำวัน เราเรียกว่า ‘Classical Physics’ ซึ่งเป็นข้อจำกัดที่จะทำให้กฎของมัวร์ถึงจุดสิ้นสุด

และกฎที่นักวิทยาศาสตร์ค้นพบเมื่อร้อยกว่าปีที่ผ่านมา ว่าเมื่อลดขนาดเล็กลงไปถึงระดับ ‘อะตอม’ ในที่อุณหภูมิเย็นยะเยือก กฎเกณฑ์ทางฟิสิกส์หรือกฎของธรรมชาติจะเปลี่ยนไป เราสิ่งนี้เรียกว่า ‘Quantum Physics’ อันเป็นที่มาของความเร็วที่เหมือนไม่มีวันสิ้นสุดของคอมพิวเตอร์

กฎธรรมชาติในโลกของควอนตัมนั้นเป็นกฎที่อยู่ ‘เหนือสามัญสำนึก’ ทิวยกตัวอย่างปรากฏการณ์ที่เกิดขึ้นมา 3 ปรากฏการณ์

ปรากฏการณ์ 1: อะตอมตัวหนึ่งสามารถอยู่หลายตำแหน่งได้ในเวลาเดียวกัน (Quantum Superposition) เปรียบเสมือนอะตอมสามารถอยู่ ซ้าย-ขวา ล่าง-บน ในเวลาเดียวกัน

ปรากฏการณ์ 2: อะตอมสามารถทะลุผ่านกำแพงได้โดยไม่ต้องกระโดดข้าม (Quantum Tunneling)

ภาพที่ 2.2 Quantum Tunneling

หมายเหตุ: จาก https://becommon.co/world/quantum-computer/

ปรากฏการณ์ 3: อะตอมสองตัวที่อยู่ห่างกันสามารถสื่อสารกันได้ โดยไม่ต้องมีการส่งสัญญาณหากัน (Quantum Entanglement) [7]

2.3 กลศาสตร์ควอนตัม (Quantum Mechanics)

กลศาสตร์ควอนตัม (quantum mechanics) เป็นสาขาหนึ่งในทฤษฎีรากฐานของฟิสิกส์ ที่มีความสามารถในการอธิบายผลการทดลองต่างๆ และถูกใช้แทนที่กลศาสตร์นิวตัน (หรือกลศาสตร์ดั้งเดิม) และ กลศาสตร์ไฟฟ้าของแม็กซ์เวลล์ (หรือทฤษฎีแม่เหล็กไฟฟ้า) ซึ่งกลศาสตร์ดั้งเดิมเหล่านี้ไม่สามารถใช้อธิบายปรากฏการณ์ในวัตถุที่มีขนาดเล็กกว่าอะตอม แต่กลศาสตร์ควอนตัมนั้นสามารถคำนวณได้แม่นยำมากกว่า โดยเฉพาะอย่างยิ่งเมื่อขนาดของวัตถุที่สนใจนั้นเล็กถึงขนาดอะตอม จึงกล่าวได้ว่ากลศาสตร์ควอนตัมนั้นเป็นรากฐานเบื้องต้นของฟิสิกส์ที่มีความสำคัญมากกว่ากลศาสตร์นิวตันและกลศาสตร์ไฟฟ้าของแม็กซ์เวลล์ หรือใกล้เคียงกับความจริงมากกว่านั่นเอง

กลศาสตร์ควอนตัมเริ่มในปี พ.ศ. 2443 เมื่อ มักซ์ พลังค์ ตีพิมพ์ทฤษฎีที่อธิบายถึงการปล่อยสเปกตรัมออกจากวัตถุดำ ซึ่ง 18 ปีต่อมา เขาได้รับรางวัลโนเบลสาขาฟิสิกส์

ข้อแตกต่างของกลศาสตร์ดั้งเดิมและกลศาสตร์ควอนตัม กลายเป็นเรื่องประหลาดจนกระทั่งในปี พ.ศ. 2469 แวร์เนอร์ ไฮเซนแบร์ก และ แอร์วิน ชเรอดิงเงอร์ สามารถอธิบายทฤษฎีดังกล่าวทางคณิตศาสตร์ได้

สำหรับความเกี่ยวเนื่องกับทฤษฎีทางฟิสิกส์อื่นๆ นั้น หากรวมสัมพัทธภาพพิเศษลงในกลศาสตร์ควอนตัมจะเรียกว่า พลศาสตร์ไฟฟ้าควอนตัม หรือทฤษฎีสนามควอนตัม

ในปัจจุบัน ถือได้ว่า กลศาสตร์ควอนตัม และ สัมพัทธภาพทั่วไป เป็นเสาหลักของฟิสิกส์ยุคใหม่ ซึ่งยังไม่มีผู้ใดสามารถรวมสองทฤษฎีนี้เข้าด้วยกันได้ แต่ทฤษฎีสตริงอาจเป็นคำตอบสำหรับปัญหานี้

สูตรการแผ่รังสีของวัตถุดำ

เมื่อวัตถุถูกทำให้ร้อน มันจะปล่อยรังสีความร้อน ในรูปแบบของการแผ่รังสีแม่เหล็กย่านอินฟราเรด (ใต้แดง) เมื่อวัตถุกลายเป็นวัตถุแดงร้อน (red-hot) เราจะสามารถเห็นความยาวคลื่นสีแดงได้ แต่รังสีความร้อนส่วนใหญ่ที่แผ่ออกมายังคงเป็นอินฟราเรด จนกระทั่งวัตถุร้อนเท่ากับพื้นผิวของดวงอาทิตย์ (ประมาณ 6000 °C ที่ที่แสงส่วนใหญ่เป็นสีขาว)

สูตรการแผ่รังสีของวัตถุดำ เป็นผลงานแรก ๆ ของทฤษฎีควอนตัม ในกลางคืนวันอาทิตย์ที่ 7 ตุลาคม พ.ศ. 2443 โดยพลังก์ มันมาจากรายงานของรูเบนส์ (Rubens) จากการค้นพบ

ล่าสุดในการค้นหาอินฟราเรด คืนนั้นเองพลังก์เขียนสูตรลงบนโปสการ์ด รูเบนส์ ได้รับโปสการ์ดนั้นในเช้าวันถัดมา

$$E = nhf \tag{2-3}$$

ซึ่งผลที่ได้จากการวิเคราะห์ตามสมการที่ (2-3) จะพบว่าเมื่อ n= เลขควอนตัม E = พลังงาน f = ความถี่ single mode ที่แผ่รังสี h = ความร้อน

วันที่มีการค้นพบควอนตัม

จากการทดลอง พลังก์ค้นพบค่าของ h และ k ดังนั้นเขาสามารถรายงานในการประชุม the German Physical Society ในวันที่ 14 ธันวาคม พ.ศ. 2443 ที่ซึ่งการแจงหน่วย หรือ quantization (ของพลังงาน) ถูกเปิดเผยเป็นครั้งแรก ค่าของเลขอโวกาโดร (the Avogadro-Loschmidt number) , จำนวนของโมเลกุลในโมล (mole) และหน่วยของประจุไฟฟ้า มีความถูกต้องมากขึ้นหลังจากนั้นจนถึงปัจจุบัน

ควอนตัม เอนแทงเกิลเมนต์

ควอนตัม เอนแทงเกิลเมนต์ ครั้งหนึ่งเคยถูกมองเป็นเรื่องซับซ้อนและลึกลับเกินกว่าจะเป็นจริงได้ มาปัจจุบันกำลังกลายเป็นเรื่องที่ตื่นเต้นมาก และมีแนวโน้มจะเป็นหนึ่งในหลักการสำคัญของเทคโนโลยีแห่งศตวรรษที่ 21 อนุภาคที่พัวพันกัน กำลังจะถูกใช้ในการสร้างระบบการสื่อสารที่เป็นความลับ อาจเป็นพื้นฐานของคอมพิวเตอร์ควอนตัมความเร็วสูงพิเศษ และแม้แต่เครื่อง "Teleportation" ในสไตล์ของภาพยนตร์ชุดสตาร์เทรค นักทฤษฎีในปัจจุบันคิดว่า เอนแทงเกิลเมนต์อาจเป็นปรากฏการณ์ค่อนข้างทั่วไปในธรรมชาติ ความคิดที่นำมาสู่ความเป็นไปได้ว่า เรากำลังอาศัยอยู่ในใยคอสมิกจริงๆ ที่เชื่อมโยงถึงกันและกัน ข้ามมิติของตำแหน่งและเวลา [4]

2.4 การซ้อนทับของควอนตัม (Quantum Superposition)

ในขณะที่คอมพิวเตอร์แบบคลาสสิคจะมีหน่วยย่อยที่สุดของข้อมูลที่เรียกว่า บิต (Bit) ส่วนคอมพิวเตอร์เชิงควอนตัมจะมีหน่วยประมวลผลที่เรียกว่า คิวบิต (Qubit) ซึ่งย่อมาจากควอนตัมบิต (Quantum bit) นั่นเอง

โดยคอมพิวเตอร์แบบคลาสสิคจะแทนค่าข้อมูลด้วย Bit ที่ประกอบด้วยค่า 1 หรือ 0 ทีละตัว แต่คอมพิวเตอร์เชิงควอนตัมจะใช้คุณสมบัติของคิวบิตที่สามารถประมวลผลค่า 1 และ 0 ได้ในเวลาเดียวกัน หรือสิ่งที่นักฟิสิกส์เรียกกันว่า "การซ้อนทับของควอนตัม (Quantum Superposition) ด้วยค่า 1 และ 0" ซึ่งคิวบิตที่ว่านี้สามารถมีค่าได้ทั้งสองสถานะพร้อม ๆ กัน

นี่ไม่ได้เป็นการกล่าวตรงๆ ว่าคิวบิตนั้นมีค่าเป็นทั้ง 1 และ 0 ได้ในเวลาเดียวกัน – แต่ก็คงไม่ใช่เรื่องที่ผิด อย่างไรก็ตามมันจะปรากฏในรูปแบบที่ชัดเจนก็ต่อเมื่อถูกเราสังเกตแล้วเท่านั้น เมื่อสถานะซ้อนทับที่เรียกว่า Superposition ถูกรบกวนจากภายนอกก็จะเกิดการยุบตัวของสถานะ (Collapses) ซึ่งคุณก็จะได้รับความน่าจะเป็นในการค้นพบความจริงที่มีค่าเป็น 1 หรือ 0 อย่างใดอย่างหนึ่ง [5]

2.5 **คิวบิท (Qubit)**

qubit คือ คิวบิต หรืออาจเรียกว่า quantum bit คือ ซึ่งคือหน่วยของข้อมูลทางคอมพิวเตอร์โดยที่ถูกเก็บไว้ด้วยสถานะของอะตอมหรืออนุภาคหนึ่ง ดังนั้นจึงสามารถใช้เก็บข้อมูลหลายๆอย่างได้พร้อมกันในเวลาเดียวกันตามกฎของกลศาสตร์ควอนตัม ซึ่งแตกต่างจากปัจจุบันที่ข้อมูลเก็บด้วยสถานะของแรงดันไฟฟ้าที่เรียกว่า บิต (bit) โดยที่ bit จะเป็นได้ 0 หรือ 1 อย่างใดอย่างหนึ่งเท่านั้น [6]

ภาพที่ 2.3 เปรียบเทียบ bit กับ Qubit

หมายเหตุ: จาก https://www.sintef.no/en/projects/quantum-computing/

2.6 **สมการชเรอดิงเงอร์ (Schrödinger equation)**

Schrödinger สมเป็นเชิงเส้น สมการเชิงอนุพันธ์บางส่วนที่ควบคุมการทำงานของคลื่นของระบบควอนตัมกล 1–2 เป็นผลลัพธ์ที่สำคัญในกลศาสตร์ควอนตัมและการค้นพบนี้เป็นจุดสังเกตสำคัญในการพัฒนาตัวแบบ สมการนี้ตั้งชื่อตามErwin Schrödingerซึ่งตั้งสมมติฐานไว้ในปี

1925 และตีพิมพ์ในปี 1926 ซึ่งเป็นพื้นฐานสำหรับผลงานที่ทำให้เขาได้รับรางวัลโนเบลสาขาฟิสิกส์ในปี 1933

แนวคิด Schrödinger สมเป็นคู่ควอนตัมของกฎข้อที่สองของนิวตันในกลศาสตร์คลาสสิก เมื่อพิจารณาจากเงื่อนไขตั้งต้นที่ทราบกันดีอยู่แล้ว กฎข้อที่สองของนิวตันจะทำการทำนายทางคณิตศาสตร์ว่าระบบทางกายภาพนั้น ๆ จะใช้เส้นทางใดเมื่อเวลาผ่านไป สมการชโรดิงเงอร์ให้วิวัฒนาการเมื่อเวลาผ่านไปของฟังก์ชันคลื่นซึ่งเป็นลักษณะเฉพาะของควอนตัม-กลศาสตร์ของระบบทางกายภาพที่แยกได้ สมการจะได้รับจากความจริงที่ว่าผู้ประกอบการเวลาวิวัฒนาการจะต้องรวมกันและดังนั้นจึงต้องถูกสร้างขึ้นโดยชี้แจงของที่ผู้ประกอบการด้วยตนเอง adjoint ซึ่งเป็นควอนตัมมิล

สมการชโรดิงเงอร์ไม่ใช่วิธีเดียวที่จะศึกษาระบบกลไกควอนตัมและทำการคาดคะเนสูตรอื่น ๆ ของกลศาสตร์ควอนตั ได้แก่กลศาสตร์เมทริกซ์นำโดยเวอร์เนอร์ไฮเซนเบิร์กและเส้นทางสูตรหนึ่งที่พัฒนาส่วนใหญ่โดยริชาร์ดไฟน์แมน Paul Diracรวมกลศาสตร์เมทริกซ์และสมการชโรดิงเงอร์ไว้ในสูตรเดียว เมื่อเปรียบเทียบวิธีการเหล่านี้ บางครั้งการใช้สมการชโรดิงเงอร์เรียกว่า "กลศาสตร์คลื่น"

ฟังก์ชั่นคลื่น (Wave Function)

ฟังก์ชั่นคลื่นในฟิสิกส์ควอนตัมเป็นคำอธิบายทางคณิตศาสตร์ของสถานะควอนตัมของแยกระบบควอนตัม ฟังก์ชันคลื่นเป็นแอมพลิจูดความน่าจะเป็นที่มีมูลค่าเชิงซ้อน และความน่าจะเป็นของผลลัพธ์ที่เป็นไปได้ของการวัดที่เกิดขึ้นในระบบสามารถหาได้จากค่านี้ สัญลักษณ์ที่พบบ่อยที่สุดสำหรับฟังก์ชันคลื่นคือตัวอักษรกรีก ψ และ Ψ (ตัวพิมพ์เล็กและpsi ตัวใหญ่ตามลำดับ

ฟังก์ชั่นคลื่นเป็นฟังก์ชั่นขององศาอิสระที่สอดคล้องกับบางชุดสูงสุดของการเดินทาง observables เมื่อเลือกการแสดงดังกล่าวแล้วฟังก์ชันคลื่นจะได้มาจากสถานะควอนตัม

สำหรับระบบที่กำหนดการเลือกองศาอิสระในการใช้งานจะไม่ซ้ำกันและโดเมนของฟังก์ชัน wave ก็ไม่ซ้ำกันเช่นกัน ยกตัวอย่างเช่นมันอาจถูกนำไปเป็นหน้าที่ของทุกพิกัดตำแหน่งของอนุภาคมากกว่าพื้นที่ตำแหน่งหรือสักครู่ของอนุภาคมากกว่าทุกคนที่มีพื้นที่โมเมนตัม ทั้งสองมีความสัมพันธ์กันโดยฟูเรียร์ อนุภาคบางชนิดเช่นอิเล็กตรอนและโฟตอนมีสปินแบบไม่เป็นศูนย์และฟังก์ชันคลื่นสำหรับอนุภาคดังกล่าวรวมถึงการหมุนเป็นระดับอิสระที่อยู่ภายในและไม่ต่อเนื่องตัวแปรที่ไม่ต่อเนื่องอื่น ๆ ยังสามารถรวมเช่นisospin เมื่อระบบมีองศาอิสระภายในฟังก์ชันคลื่นที่แต่ละจุดในองศาอิสระต่อเนื่อง (เช่นจุดในอวกาศ) จะกำหนดจำนวนเชิงซ้อนสำหรับค่าองศาอิสระที่เป็นไปได้แต่ละค่า (เช่นองค์ประกอบ z ของ สปิน) - ค่าเหล่านี้มักจะแสดงในเมทริกซ์คอลัมน์ (เช่น2 × 1เวกเตอร์คอลัมน์อิเล็กตรอนที่ไม่สัมพันธ์กับการหมุน1 / 2) [9]

เบื้องต้น

หลักสูตรเบื้องต้นเกี่ยวกับฟิสิกส์หรือเคมีมักจะแนะนำสมการชโรดิงเงอร์ในลักษณะที่สามารถชื่นชมได้เมื่อรู้เพียงแนวคิดและสัญลักษณ์ของแคลคูลัสพื้นฐานโดยเฉพาะอย่างยิ่งอนุพันธ์ที่เกี่ยวกับอวกาศ และเวลากรณีพิเศษของสมการชโรดิงเงอร์ที่ยอมรับคำสั่งในเงื่อนไขเหล่านั้นคือสมการชโรดิงเงอร์ตำแหน่ง-สเปซสำหรับอนุภาคที่ไม่สัมพันธ์กันเพียงตัวเดียวในมิติเดียว

$$ih\frac{\partial}{\partial t}\psi(x,t)\rangle = \left[-\frac{\square^2}{2m}\frac{\partial^2}{\partial x^2} + V(x,t)\right]\psi(\mathrm{x},\mathrm{t}). \qquad (2\text{-}4)$$

ดังสมการที่ (2-4) ที่นี่ $\psi(x,t)$ คือ ฟังก์ชันคลื่น ฟังก์ชันที่กำหนดจำนวนเชิงซ้อนให้กับแต่ละจุด x ทุกครั้ง t พารามิเตอร์ m คือมวลของอนุภาค และ $V(x,t)$ คือศักยภาพที่แสดงถึงสภาพแวดล้อมที่มีอนุภาคอยู่ ค่าคงที่ i เป็นหน่วยจินตภาพและ h คือค่าคงที่พลังค์ที่ลดลง ซึ่งมีหน่วยของการกระทำ (พลังงานคูณด้วยเวลา)

สมการขึ้นอยู่กับเวลา

$$ih\frac{d}{dt}\big|\psi(t)\rangle = \hat{H}\big|\psi(t)\rangle \qquad (2\text{-}5)$$

ดังสมการที่ (2-5) รูปแบบของสมการชโรดิงเงอร์ขึ้นอยู่กับสถานการณ์ทางกายภาพ รูปแบบทั่วไปที่สุดคือสมการชโรดิงเงอร์ที่ขึ้นกับเวลา ซึ่งให้คำอธิบายของระบบที่วิวัฒนาการไปตามกาลเวลา

ที่ไหน ψ (อักษรกรีกpsi) เป็นเวกเตอร์สถานะของระบบควอนตัม t คือเวลาและ $\hat{H}$ เป็นที่สังเกตได้ที่แฮมิลตัน คำว่า "สมการชโรดิงเงอร์" สามารถอ้างถึงทั้งสมการทั่วไปหรือสมการที่ไม่สัมพันธ์กันเฉพาะ สมการทั่วไปนั้นค่อนข้างทั่วไป ใช้ในกลศาสตร์ควอนตัมสำหรับทุกอย่างตั้งแต่สมการไดแรคไปจนถึงทฤษฎีสนามควอนตัมโดยการแทนค่านิพจน์ที่หลากหลายสำหรับแฮมิลตัน เวอร์ชันที่ไม่สัมพันธ์กันเป็นการประมาณที่ให้ผลลัพธ์ที่แม่นยำในหลายสถานการณ์ แต่ในระดับหนึ่งเท่านั้น (ดูกลศาสตร์ควอนตัมเชิงสัมพันธ์และทฤษฎีสนามควอนตัมสัมพัทธภาพ) ในการใช้สมการชโรดิงเงอร์ ให้จดแฮมิลตันเนียนสำหรับระบบ โดยคำนึงถึงพลังงานจลน์และพลังงานศักย์ของอนุภาคที่ประกอบเป็นระบบ แล้วใส่ลงในสมการชโรดิงเงอร์ สมการอนุพันธ์ย่อยที่เป็นผลลัพธ์จะได้รับการแก้ไขสำหรับฟังก์ชันคลื่น

$$\Pr(x,t) = |\psi(x,t)|^2 \quad (2\text{-}6)$$

ซึ่งประกอบด้วยข้อมูลเกี่ยวกับระบบ ในทางปฏิบัติของตารางค่าสัมบูรณ์ของฟังก์ชั่นคลื่นในแต่ละจุดจะนำไปกำหนดฟังก์ชั่นความหนาแน่นของความน่าจะเป็น ตัวอย่างเช่น กำหนดฟังก์ชันคลื่นในช่องว่างตำแหน่ง $\psi(\mathrm{x},\mathrm{t})$ ตามที่กล่าวไว้ข้างต้นดังสมการที่ (2-6)

สมการไม่ขึ้นกับเวลา

$$\hat{H}\left|\psi\right\rangle = E\left|\psi\right\rangle \quad (2\text{-}7)$$

สมการชโรดิงเงอร์ขึ้นกับเวลาที่อธิบายข้างต้นคาดการณ์ว่าฟังก์ชันคลื่นสามารถสร้างคลื่นยืนเรียกว่ารัฐนิ่ง สถานะเหล่านี้มีความสำคัญอย่างยิ่งเนื่องจากการศึกษาส่วนบุคคลในภายหลังทำให้งานแก้สมการชโรดิงเงอร์ที่ขึ้นกับเวลาสำหรับสถานะใด ๆ ง่ายขึ้น สถานะนิ่งสามารถอธิบายได้ด้วยสมการชโรดิงเงอร์รูปแบบที่ง่ายกว่า ซึ่งเป็นสมการชโรดิงเงอร์ที่ไม่ขึ้นกับเวลาดังสมการที่ (2-7)

ที่ไหน E คือพลังงานของระบบ ใช้เฉพาะเมื่อHamiltonianเองไม่ได้ขึ้นอยู่กับเวลาอย่างชัดเจน อย่างไรก็ตาม แม้ในกรณีนี้ ฟังก์ชันคลื่นทั้งหมดยังคงขึ้นอยู่กับเวลา ในภาษาของพีชคณิตเชิงเส้นสมการนี้เป็นสม eigenvalue ดังนั้น ฟังก์ชันคลื่นจึงเป็นฟังก์ชันลักษณะเฉพาะของโอเปอเรเตอร์แฮมิลตันที่มีค่าลักษณะเฉพาะที่สอดคล้องกัน E [8]

2.7 เทคโนโลยีควอนตัมคอมพิวเตอร์ (Quantum Computer)

เมื่อเทคโนโลยีการพัฒนา CPU เกือบจะเดินมาถึงทางตัน ทั้งขนาดทรานซิสเตอร์ที่ปัจจุบันอยู่ที่ 7 นาโนเมตร และการที่จะพัฒนาให้เล็กกว่านี้เป็นเรื่องที่แทบจะเป็นไปได้ยาก วิทยาศาสตร์แขนงใหม่จึงเป็นหนทางในการพัฒนาคอมพิวเตอร์ใน Generation ใหม่ ซึ่งหนึ่งในนั้นคือ Quantum Computer เทคโนโลยี Quantum เกี่ยวข้องกับ อนุภาคขนาดเล็กมาก ๆ ที่มีคุณสมบัติสองอย่างคือ

1. Superposition (สภาวะทับซ้อน) คืออะตอมจะมีสองสถานะในเวลาเดียวกัน หรืออะตอมตัวหนึ่งสามารถอยู่หลายตำแหน่งได้ในเวลาเดียวกัน

2. Entanglement (การพัวพันเชิงควอนตัม) อนุภาค Quantum มี Spin หรือคุณสมบัติ Spin Up และ Spin Down เมื่อมีการเชื่อมอนุภาค quantum เข้าด้วยกัน หรือ collapse จะทำให้อนุภาคทั้งสอง เกิดการ Entanglement กัน ยิงโฟตอนเข้าไปแยกอนุภาคออกจากกัน หรือยิงโฟตอน

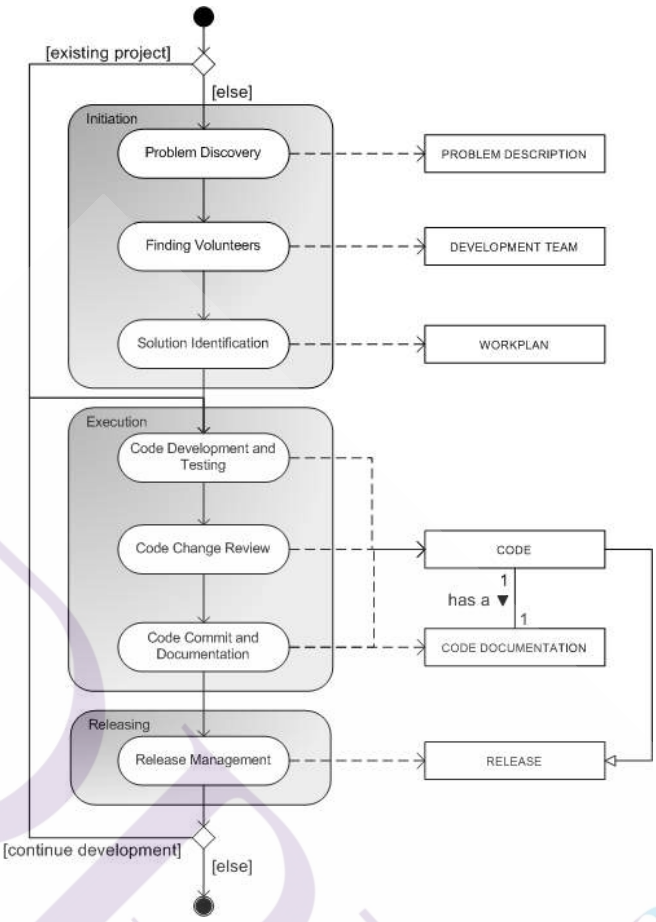
1 อนุภาคเข้าไปยังเครื่องแยกอนุภาค เมื่อส่วนนึงทำการสำรวจสภาวะแล้วเป็น + อีกส่วนจะเป็น – “ทันที” แม้ว่าทั้งสองอนุภาคจะอยู่ห่างไกลกันแค่ไหนก็ตาม ไอสไตน์เรียก ปรากฏการณ์นี้ว่า “Spooky, action at a distance” หรือปฏิสัมพันธ์ระยะไกลราวกับโดนผีหลอก (EPR paradox) [9]

2.8 โอเพ่นซอร์สซอฟต์แวร์ (Open Source Software)

ซอฟต์แวร์คอมพิวเตอร์ หรือ แอปพลิเคชัน ที่ทางนักพัฒนาซอฟต์แวร์รายแรกของซอฟต์แวร์นั้นๆ เปิดเผยที่มา หลักการของซอฟต์แวร์นั้นให้บุคคลภายนอก ผู้ใช้งาน นักพัฒนาซอฟต์แวร์รายอื่นได้ใช้งาน รวมถึงการนำ รหัสต้นฉบับ หรือ ซอร์สโค้ด (Source Code) ของซอฟต์แวร์ไปแจกจ่าย ทำซ้ำ ศึกษา ดัดแปลง พัฒนาต่อยอดได้อีก โดยไม่ผิดกฎหมาย หรือกฎการใช้แต่อย่างใด [10]

นิยามของ Open-Source Software

1. ไม่จำกัดการแจกจ่ายเฉพาะกลุ่มบุคคลกลุ่มใดกลุ่มหนึ่ง
2. ไม่จำกัดการแจกจ่าย Source Code ส่วนใดส่วนหนึ่ง จะต้องเปิดเผยทั้งหมด
3. ต้องแจกจ่ายทั้งโปรแกรมต้นฉบับและ Source Code มีช่องทางการแจกจ่ายให้เข้าถึงง่าย
4. มีเงื่อนไข Open-Source License กำกับ เช่น GPL, BSD



ภาพที่ 2.4 ตัวอย่างกระบวนการพัฒนา Open-Source Software

หมายเหตุ: จาก *https://commons.wikimedia.org/wiki/File:OSSD_process_data_diagram.png*

2.9 Qiskit

Qiskit เป็น Open Source สำหรับ Quantum Computer มีเครื่องมือสำหรับสร้างและจัดการโปรแกรมควอนตัมและเรียกใช้บนอุปกรณ์ควอนตัมต้นแบบบน IBM Quantum Experience หรือบนเครื่องจำลองบนคอมพิวเตอร์ภายใน มันเป็นไปตามรูปแบบวงจรสำหรับการคำนวณควอนตัมสากลและสามารถใช้กับฮาร์ดแวร์ควอน

ภาพที่ 2.5 Qiskit

หมายเหตุ: จาก https://blue-natchapol.medium.com/guide-for-the-ibm-quantum

Qiskit ก่อตั้งโดย IBM วิจัยที่จะช่วยให้การพัฒนาซอฟต์แวร์สำหรับพวกเขา Cloud Quantum Computing บริการ IBM ควอนตัมประสบการณ์ เงินสมทบยังมาจากผู้สนับสนุนภายนอก ซึ่งปกติแล้วจะมาจากสถาบันการศึกษา

Qiskit เวอร์ชันหลักใช้ภาษาการเขียนโปรแกรม Python เวอร์ชันสำหรับ Swift และ JavaScript ได้รับการสำรวจครั้งแรก แม้ว่าการพัฒนาสำหรับเวอร์ชันเหล่านี้จะหยุดลง แต่น้อยที่สุด อีกครั้งการดำเนินงานของคุณสมบัติพื้นฐานสามารถใช้ได้เป็น MicroQiskit ซึ่งจะทำให้ง่ายต่อการพอร์ตไปยังแพลตฟอร์มทางเลือก [11]

2.10 Forest (pyQuil)

Quil เป็นสถาปัตยกรรมชุดคำสั่งควอนตัม ที่เปิดตัวโมเดลหน่วยความจำควอนตัม/คลาสสิกที่ใช้ร่วมกันเป็นครั้งแรก มันถูกนำโดยโรเบิร์ตสมิธ , ไมเคิลเคอร์ติและวิลเลียมเซงในปฏิบัติควอนตัมสถาปัตยกรรมชุดคำสั่ง หลายขั้นตอนวิธีการควอนตัม (รวมteleportation ควอนตัม , แก้ไขข้อผิดพลาดควอนตัม , จำลอง, และการเพิ่มประสิทธิภาพขั้นตอนวิธีการ) ต้องใช้สถาปัตยกรรมหน่วยความจำที่ใช้ร่วมกัน Quil กำลังได้รับการพัฒนาสำหรับโปรเซสเซอร์ควอนตัมตัวนำยิ่งยวดที่พัฒนาโดย Rigetti Computing ผ่าน Forest ควอนตัมการเขียนโปรแกรม API หลาย Library ที่เรียกว่า pyQuil ได้รับการแนะนำในการพัฒนาโปรแกรม Quil กับโครงสร้างระดับที่สูงขึ้น Back End Quil ยังได้รับการสนับสนุนโดยสภาพแวดล้อมการเขียนโปรแกรมควอนตัมอื่นๆ

ภาพที่ 2.6 Rigetti and Forest (pyQuil)

หมายเหตุ: จาก https://github.com/rigetti/forest-software/blob/master/forest.png

Rigetti QVM Q

Rigetti Computing ได้พัฒนา Quantum Virtual Machine ใน Common Lisp ซึ่งจำลองเครื่อง Quantum Abstract Machine ที่กำหนดไว้บนคอมพิวเตอร์แบบคลาสสิก และสามารถแยกวิเคราะห์และเรียกใช้โปรแกรม Quil ซึ่งอาจดำเนินการจากระยะไกลผ่าน HTTP [12]

2.11 ProjectQ

ProjectQ เป็นเฟรมเวิร์กซอฟต์แวร์โอเพ่นซอร์สสำหรับการคำนวณควอนตัมที่ใช้ใน Python อนุญาตให้ผู้ใช้ใช้โปรแกรมควอนตัมใน Python โดยใช้ไวยากรณ์ที่ทรงพลังและใช้งานง่าย จากนั้น ProjectQ สามารถแปลโปรแกรมเหล่านี้เป็นแบ็คเอนด์ประเภทใดก็ได้ ไม่ว่าจะเป็นโปรแกรมจำลองที่ทำงานบนคอมพิวเตอร์แบบคลาสสิกหรือชิปควอนตัมจริง รวมถึงแพลตฟอร์ม IBM Quantum Experience แพลตฟอร์มฮาร์ดแวร์อื่น ๆ จะได้รับการสนับสนุนในอนาคต [13]

ภาพที่ 2.7 ProjectQ

หมายเหตุ: จาก https://ml2quantum.com/projectq

2.12 Quantum Developer Kit (Q#)

Microsoft ได้ออกมาประกาศเปิดตัว Quantum Development Kit รุ่น Preview สำหรับเปิดให้เหล่านักพัฒนาได้นำไปใช้ทดลองสร้างโปรแกรมสำหรับทำงานบน Quantum Computer โดยเฉพาะ ด้วยภาษา Q#

Quantum Development Kit นี้สามารถทำงานร่วมกับ Microsoft Visual Studio ได้เป็นอย่างดี อีกทั้งยังถูกออกแบบมาให้ทำงานร่วมกับ Quantum Simulator ที่ถูกรวมอยู่ในชุด Quantum Development Kit เพื่อทำการทดสอบได้ โดยรองรับการจำลองระบบ Quantum Computer ได้ถึงขนาด 30 Qubit บนเครื่อง Laptop ทั่วๆ ไป ทำให้เหล่านักพัฒนาสามารถเริ่มต้นเรียนรู้พัฒนาโปรแกรมบน Quantum Computer ได้ทันที และสามารถนำโค้ดเหล่านั้นไปใช้งานได้ทันทีเมื่อมีเครื่อง Quantum Computer ขนาด 30 Qubit จำหน่ายหรือเปิดให้เช่าใช้กันแล้ว

ส่วนผู้ที่สนใจทดลองพัฒนาโปรแกรมสำหรับระบบ Quantum Computer จำลองที่ขนาดใหญ่ขึ้นไป ทาง Microsoft ได้เตรียมตัว Simulator บน Microsoft Azure ที่สามารถจำลองตัวเองเป็น Quantum Computer ขนาด 40 Qubit ได้เอาไว้แล้ว [14]

ภาพที่ 2.8 Microsoft Quantum Developer Kit

หมายเหตุ: จาก https://www.neowin.net/news/microsoft039s-quantum-development-kit-now-supports-macos-and-linux/

2.13 ควอนตัมอัลกอริทึม (Quantum Algorithms)

คอมพิวเตอร์ธรรมดาในปัจจุบันก็สามารถคำนวณหาคำตอบตามที่เราต้องการได้หมดแล้ว แต่จะใช้เวลานานเท่าไหร่เค่นั้นเอง ดังนั้นขีดจำกัดของคอมพิวเตอร์ทั่วไปจึงอยู่ที่เวลาและหน่วยความจำที่ใช้ใน การประมวลผล ปัจจุบันยังมีปัญหาจำนวนมากที่ไม่สามารถหาคำตอบได้เพราะใช้เวลานานเกินไป บางปัญหาจึงใช้วิธีการจำลองแบบ (simulation) เพื่อหาคำตอบแทน แต่การจำลองแบบระบบที่ซับซ้อนมากๆ ก็ยังต้องใช้เวลานานแม้จะใช้ซูเปอร์คอมพิวเตอร์ที่เร็วที่สุดในปัจจุบัน โดยทั่วไปเวลาในการประมวลผลจะขึ้นกับความซับซ้อนของปัญหา ในคอมพิวเตอร์เราแบ่งความซับซ้อนของปัญหาออกเป็น 4 classes ครับ คือ P-problems, NP-Problems, NP-complete problems และ Exponential problems [15]

1. P-problems ก็คือปัญหาที่สามารถแก้ได้ในขอบเขตของ polynomial time/space – จำนวนครั้งในการประมวลผลอยู่ในขอบเขตของ nc เมื่อ n เป็นขนาดของปัญหาและ c เป็นค่าคงที่ที่หาได้แน่นอน ปัญหาระดับ P-problem ถือว่าเป็นแก้ได้ง่ายที่สุด และใช้ทรัพยากรน้อยที่สุด

2. NP-problems เป็น nondeterministic polynomial time อธิบายง่ายๆ หน่อยก็คือแต่ละ n ต้องเดาว่าคำตอบคืออะไรและการจะหาว่าคำตอบนั้นถูกหรือไม่สามารถทำได้ใน ขอบเขตของ polynomial time/space ไม่ว่าคำตอบนั้นจะถูกหรือผิด มีปัญหาหลายๆอันที่นักคณิตศาสตร์เคยคิดว่า

เป็น NP-problem แต่จริงๆ แล้วไม่ใช่ การที่มีคนพิสูจน์ได้ว่าปัญหาที่เป็น NP-problem หลายๆ ปัญหาลดลงมาเหลือ P-problem ได้ ทำให้มีความหวังว่าปัญหาทั้งหมดไม่ว่าจะซับซ้อนเพียงไรก็อาจจะหาคำตอบได้ภายในขอบเขตของ polynomial time/space

3. NP-complete problems แบ่งเป็นสองกลุ่มย่อยคือ NP-Hard problem: เป็นปัญหาที่สามารถเอา algorithm ในการแก้ปัญหามาแปลงเพื่อแก้ปัญหา NP-problem อื่นๆ ได้ หรือในทางกลับกันคือ NP-problem ทุกอันสามารถแปลงเป็น NP-Hard ได้…NP-complete: คือปัญหาที่เป็นทั้ง NP-problem และ NP-Hard problem ตัวอย่างเช่น Traveling Salesman Problem (TSP) สิ่งที่น่าสนใจของ NP-complete คือ algorithm สำหรับแก้ปัญหาอย่างนึงสามารถแปลงไปแก้ปัญหา NP-complete อื่นๆ ได้..ดังนั้นเมื่อไหร่ที่ NP-complete มี algorithm ที่ลดรูปเหลือ P-problem ได้ก็จะหมายความว่า NP-complete อื่นๆ จะถูกลดลงมาเหลือแค่ P-problem ซึ่งเป็น class ที่ใช้ space/time น้อยที่สุดมีนักคณิตศาสตร์จำนวนมากพยายามค้นหา algorithm ลักษณะนี้ จนถึงทุกวันนี้ยังไม่มีใครพบแม้แต่ algorithm เดียว แต่ก็ไม่สามารถพิสูจน์ว่ามันไม่มีอยู่จริง

4. Exponential problem คือปัญหาที่แก้ได้ในขอบเขตของ exponential time/space – จำนวนครั้งในการประมวลผลอยู่ในขอบเขตของ cn ปัญหาลักษณะนี้ถือว่ามีความซับซ้อนมากที่สุด ต้องใช้ space/time สูงสุดในบรรดาปัญหาที่แก้ได้ด้วย computer Exponential problem ที่รู้จักกันดีก็คือ Tower of Hanoi ซึ่งมีจำนวนครั้งของการย้ายจานเป็น exponential function ของจำนวนจาน

2.14 อัลกอริทึมของชอร์ (Shor's algorithm)

ตั้งชื่อตามนักคณิตศาสตร์ชื่อปีเตอร์ ชอร์ที่คิดขึ้นในปีค.ศ.1994 โดยขั้นตอนวิธีนี้เป็นขั้นตอนวิธีควอนตัม (ขั้นตอนวิธีที่ทำงานบนควอนตัมคอมพิวเตอร์) ที่ใช้ในการแยกตัวประกอบของจำนวนเต็ม ซึ่งโดยทั่วไปแล้วใช้ในการแก้ปัญหา:ให้จำนวนเต็ม N แล้วให้หาตัวประกอบเฉพาะของ N

ในควอนตัมคอมพิวเตอร์นั้น การแยกตัวประกอบด้วยขั้นตอนวิธีของชอร์จะใช้เวลาในการทำงานไม่เกินฟังก์ชันพหุนาม (Polynomial) ของขนาดข้อมูล โดยจะใช้เวลาเป็น $O((\log N)3)$ ซึ่งแสดงให้เห็นว่าเป็นการแก้ปัญหาการแยกตัวประกอบของจำนวนเต็มที่มีประสิทธิภาพในควอนตัมคอมพิวเตอร์ วิธีนี้จัดเป็นวิธีที่เร็วกว่าหลาย ๆ วิธีที่มีประสิทธิภาพที่รู้จักกันทั่ว ๆ ไปทีมักจะใช้เวลาเป็นฟังก์ชันเลขชี้กำลัง (Exponential) ขนาดของข้อมูล

ให้ควอนตัมคอมพิวเตอร์มีจำนวนคิวบิตที่เพียงพอ ขั้นตอนวิธีของชอร์จะสามารถถอดรหัสประเภทระบบเข้ารหัสแบบกุญแจอสมมาตรได้ ยกตัวอย่างเช่น รหัสลับ RSA โดย RSA นั้นตั้งอยู่บนสมมติฐานที่ว่าการคำนวณหาตัวประกอบของเลขที่มีจำนวนมาก ๆ นั้นเป็นไปไม่ได้

เท่าที่รู้กันสมมติฐานนี้ใช้ได้สำหรับคอมพิวเตอร์ทั่วไป และไม่มีขั้นตอนวิธีใดที่จะทำงานในเวลาไม่เกินฟังก์ชันพหุนาม อย่างไรก็ตามขั้นตอนวิธีของชอร์แสดงให้เห็นถึงวิธีการแยกตัวประกอบที่มีประสิทธิภาพในควอนตัมคอมพิวเตอร์ เพื่อให้ควอนตัมคอมพิวเตอร์มีขนาดที่ใหญ่พอที่จะถอดรหัส RSA ได้ จึงสร้างแรงจูงใจอย่างมากในการออกแบบและการสร้างควอนตัมคอมพิวเตอร์และสำหรับศึกษาแบบวิธีการบนควอนตัมคอมพิวเตอร์ใหม่ ๆ ในขณะที่ก็มีการให้การสนับสนุนการวิจัยระบบการเข้ารหัสแบบใหม่เพื่อสร้างความปลอดภัยจากควอนตัมคอมพิวเตอร์ เรียกว่าวิทยาการเข้ารหัสลับหลังควอนตัม (post-quantum cryptography)

กระบวนการของขั้นตอนวิธีของชอร์

ปัญหาที่เราต้องการจะหาคำตอบคือ: ให้ N เป็นจำนวนประกอบที่เป็นเลขคี่ ให้หาจำนวนเต็ม d ที่อยู่ระหว่าง 1และ N และ หาร N ลงตัว เราสนใจ N ที่มีค่าเป็นจำนวนคี่เนื่องจากถ้าเป็นจำนวนคู่จะมี “2” เป็นตัวประกอบเฉพาะอยู่แล้ว โดยเราสามารถใช้การทดสอบจำนวนเฉพาะเพื่อหาว่า N นั้นเป็นจำนวนประกอบหรือไม่

นอกจากนั้นแล้ว เรายังต้องการ N ที่ไม่ใช่เลขยกกำลังของจำนวนเฉพาะ เราสามารถทดสอบโดยใช้รากที่2, รากที่4, รากที่ k ของ N โดยที่ $K \leq \log_2 N$ และต้องตรวจสอบว่าไม่มีจำนวนใดในนั้นที่เป็นจำนวนเต็ม

เนื่องจาก N ไม่ใช่เลขยกกำลังของจำนวนเฉพาะ คำตอบต้องเป็นผลลัพธ์ของจำนวนเฉพาะสองตัวที่มีค่ามากกว่า 1 จากทฤษฎีบทเศษเหลือของจีน โดยจุดมุ่งหมายของอัลกอริทึ่มนี้คือการหาตัวประกอบของ N ที่มีค่าไม่เป็น 1 และ -1

ในการแยกตัวประกอบโดยใช้ขั้นตอนวิธีของชอร์นั้นประกอบด้วย2ส่วนคือ

1. ส่วนลดรูปปัญหา โดยส่วนนี้จะใช้ลดรูปปัญหาจากปัญหาการแยกตัวประกอบเป็นปัญหาในการหาลำดับ ซึ่งส่วนนี้จะสามารถทำได้ในคอมพิวเตอร์ทั่วไป
2. ส่วนแบบวิธีการควอนตัมที่ใช้ในการแก้ปัญหาในการหาลำดับ

ส่วนลดรูปปัญหา

1. หาคาบซ้ำๆ (r) ที่เกิดจากผลของฟังก์ชัน

$$f(x) = a^2 \, mod \, N \tag{2-8}$$

ดังสมการที่ (2-8) นั่นคือ r ลำดับ a ของ ใน $(Z_N)^X$ ที่เป็นจำนวนเต็มบวกที่น้อยที่สุดสำหรับ $f(x+r) = f(x)$

2. ถ้า $a^r - 1$ หารด้วย N ลงตัว และ r เป็นเลขคู่ แยกตัวประกอบของ $a^r - 1$ ได้เป็น $a^{r/2} - 1$ กับ $a^{r/2} + 1$
3. หรม.ของ $a^{\frac{r}{2}} \pm 1$ กับ N คือตัวประกอบที่เราสนใจของ N

ส่วนควอนตัมใช้หาคาบที่เกิดจากฟังก์ชัน

1. สร้างหน่วยความจำของควอนตัมคอมพิวเตอร์ขึ้น 2 ส่วน

1.1 ส่วนแรกใช้เก็บจำนวนเต็ม x ที่มีค่าตั้งแต่ 0 ถึง $q-1$ โดยที่ q เป็นเลขยกกำลังของ 2 ที่ $N^2 \leq q \leq 2N^2$

1.2 ใช้เก็บผลลัพธ์ฟังก์ชัน $f(x+r) = f(x)$ ซึ่งเป็นไปได้ตั้งแต่ 0 ถึง $N-1$ โดยจำนวนเต็มที่เก็บไว้ในหน่วยความจำแต่ละส่วนจะแทนด้วยฐาน (base) แต่ละตัวของหน่วยความจำ และสถานะของหน่วยความจำจะเป็นผลรวม (superposition) ของฐานต่าง ๆ ซึ่งมีแอมพลิจูด (amplitude) เท่ากันทุกฐาน

เช่น ถ้า N=15 q จะมีค่าเป็น 2^8 ซึ่งมากกว่า 15^2 แต่น้อยกว่า 2*15^2 จะได้ x มีค่าตั้งแต่ 0-255 ซึ่งต้องใช้หน่วยความจำส่วนแรกขนาด 8 คิวบิต เพื่อเก็บตัวเลข 256 ตัว คือ

ฐาน 00000000 แทนเลข 0

ฐาน 00000001 แทนเลข 1

ฐาน 00000010 แทนเลข 2

ฐาน 00000011 แทนเลข 3

ฐาน 11111111 แทนเลข 255

และหน่วยความจำส่วนที่สองต้องใช้ 4 คิวบิตเพื่อเก็บเลข 0 ถึง 14

2. ตอนเริ่มต้นนั้นหน่วยความจำของควอนตัมคอมพิวเตอร์จะอยู่ในสถานะ

$$q^{-1/2} \sum_{x=0}^{q-1} |x\rangle\,|0\rangle$$

เมื่อ $|x\rangle$ เป็นฐานที่แทนจำนวนเต็ม x

3. คำนวณค่าของฟังก์ชัน $f(x) = a^2 \bmod N$ จากค่า x ในหน่วยความจำส่วนแรกแล้วเก็บผลลัพธ์ไว้ในส่วนที่สอง ซึ่งจะให้สถานะของหน่วยความจำเป็น

$$q^{-1/2} \sum_{x=0}^{q-1} |x\rangle\,|f(x)\rangle$$

เนื่องจากคุณสมบัติที่ทำงานหลาย ๆ อย่างขนานกันไปได้ของควอนตัมคอมพิวเตอร์ จึงสามารถคำนวณค่า $f(x) = a^2 \bmod N$ ของจำนวน x ต่าง ๆ ได้พร้อม ๆ กันในคราวเดียว ทำให้สามารถทำงานได้รวดเร็วกว่าคอมพิวเตอร์ปกติซึ่งต้องคำนวณทีละตัว

4. ทำการวัดสถานะของหน่วยความจำส่วนที่สองซึ่งเก็บผลลัพธ์ไว้ การวัดนี้จะทำให้ฟังก์ชันคลื่นของหน่วยความจำเกิดการยุบรวมกัน เหลือเพียงฐานที่ทำให้เกิดผลลัพธ์ที่วัดได้ ถ้าวัดสถานะของผลลัพธ์ได้เป็น k จะเหลือเพียงฐาน

$$x = b, b + r, b + 2r, b + 3r, \dots, b + nr$$

เมื่อ : b คือจำนวนเต็มที่น้อยที่สุดที่ $f(b) = a^b \ mod \ N = k$

r คือคาบของ $f(x)$

n คือคาบของ$(q - b)/\ r$

หลังทำการวัดแล้วจะได้สถานะของหน่วยความจำเป็น

$$||A||^{-1/2} \sum_{x'=x\prime\in A} |x', k\rangle$$

เมื่อ A คือเซตของ X ซึ่งให้ $a^x \ mod \ N = k$ และ A คือจำนวนสมาชิกของเซตดังกล่าว

5. แปลงหน่วยความจำส่วนแรกด้วยการแปรงฟูริเยร์ไม่ต่อเนื่อง

$$|x\rangle \rightarrow q^{-1/2} \sum_{x=0}^{q-1} e^{2\pi ixc/q} \ |c\rangle$$

ซึ่งทำให้แอมพลิจูดของฐานซึ่งแทนจำนวนที่เป็นจำนวนเท่าของ q/r สูงขึ้นเป็นพีค

เมื่อทำการวัดสถานะของหน่วยความจำส่วนแรก จึงมีความน่าจะเป็นสูงที่สุดที่จะได้ค่าเป็นจำนวนดังกล่าว จากค่าที่วัดได้ดังกล่าวคำนวณค่า n/r จนได้คาบrตามลำดับ [16]

2.15 ก้าวหน้าในการคำนวณควอนตัม

IBM ได้เปิดตัวโปรเซสเซอร์ "ควอนตัม" ขั้นสูงซึ่งเป็นส่วนหนึ่งของความพยายามที่จะสร้างคอมพิวเตอร์ที่เร็วเป็นพิเศษ เครื่องจักรเหล่านี้สามารถปฏิวัติการคำนวณ โดยควบคุมโลกที่แปลกประหลาดของฟิสิกส์ควอนตัมเพื่อแก้ปัญหาที่เกินเอื้อมสำหรับ "คลาสสิก" ที่ล้ำหน้าที่สุด แต่อุปสรรคในการสร้างเวอร์ชันขนาดใหญ่ที่ใช้งานได้จริงทำให้คอมพิวเตอร์ควอนตัมถูกจำกัดให้อยู่ในห้องแล็บชิปใหม่มี 127 "qubits" ซึ่งมากเป็นสองเท่าของโปรเซสเซอร์ IBM รุ่นก่อน Qubits (ควอนตัมบิต) เป็นหน่วยข้อมูลพื้นฐานที่สุดในคอมพิวเตอร์ควอนตัม บริษัทเรียกโปรเซสเซอร์ Eagle ตัวใหม่ว่า "ก้าวสำคัญบนเส้นทางสู่การคำนวณควอนตัมที่ใช้งานได้จริง" แต่ผู้เชี่ยวชาญด้านคอมพิวเตอร์ควอนตัมคนหนึ่งกล่าวว่าจำเป็นต้องมีรายละเอียดเพิ่มเติมเพื่อประเมินว่านี่เป็นความก้าวหน้าที่สำคัญหรือไม่ ในช่วงไม่กี่ปีที่ผ่านมามีความสนใจในเครื่องจักรเพิ่มขึ้น เนื่องจากมีศักยภาพในการปรับปรุงพลังการประมวลผลอย่างมากมาย สามารถใช้เพื่อช่วยพัฒนาวัสดุและยา

ใหม่ ๆ หรือปรับปรุงด้านปัญญาประดิษฐ์ คอมพิวเตอร์ควอนตัมใช้ประโยชน์จากพฤติกรรมแปลก ๆ ในระดับที่เล็กมากในคอมพิวเตอร์คลาสสิก หน่วยของข้อมูลเรียกว่า "บิต" และสามารถมีค่าเป็นหนึ่งหรือศูนย์ก็ได้ แต่มันเทียบเท่าในระบบควอนตัม - คิวบิต – สามารถเป็นได้ทั้งหนึ่งและศูนย์ในเวลาเดียวกันนี่คือแนวคิดของการทับซ้อน ซึ่งบางสิ่งสามารถมีอยู่ได้ในหลายสถานะพร้อมกันเพื่อควบคุมพลังของมัน คิวบิตหลายตัวต้องเชื่อมโยงเข้าด้วยกัน กระบวนการที่เรียกว่าการพัวพันและด้วยการเพิ่มแต่ละ qubit เพิ่มเติม พลังในการคำนวณของโปรเซสเซอร์ก็เพิ่มขึ้นเป็นสองเท่าอย่างมีประสิทธิภาพ โปรเซสเซอร์ Eagle ตามหลัง Hummingbird 65-qubit ของบริษัท ซึ่งเปิดตัวในปี 2020 และ Falcon 27-qubit ในปี 2019 ดร.ดาริโอ กิล รองประธานอาวุโสและผู้อำนวยการฝ่ายวิจัยกล่าวว่า "การมาถึงของโปรเซสเซอร์ Eagle เป็นก้าวสำคัญสู่ยุคที่คอมพิวเตอร์ควอนตัมสามารถให้ประสิทธิภาพเหนือกว่าคอมพิวเตอร์คลาสสิกสำหรับการใช้งานที่เป็นประโยชน์" การคำนวณควอนตัมมีพลังในการเปลี่ยนแปลงเกือบทุกภาคส่วน และช่วยเราจัดการกับปัญหาที่ใหญ่ที่สุดในยุคของเรา"

อำนาจสูงสุดของควอนตัม

ขั้นตอนสำคัญคือการแสดงให้เห็นถึงสิ่งที่ได้รับการขนานนามว่า "อำนาจสูงสุดของควอนตัม" ในปี 2019 Google กล่าวว่าโปรเซสเซอร์ควอนตัม Sycamore 53-qubit มีประสิทธิภาพเหนือกว่าคอมพิวเตอร์ทั่วไป - ในงานเฉพาะ – เป็นครั้งแรก นักวิจัยของ Google เผยแพร่ผลงานดังกล่าวในวารสารวิชาการ Nature อันทรงเกียรติ ในขณะนั้น นักวิทยาศาสตร์จาก IBM ได้ตั้งคำถามเกี่ยวกับตัวเลขบางส่วนของ Google และคำจำกัดความของอำนาจสูงสุดของควอนตัม Prof. Scott Aaronson จาก University of Texas at Austin เกี่ยวกับชิป Eagle ตัวใหม่กล่าวว่า "ผมตั้งตารอที่จะได้เห็นรายละเอียดที่แท้จริง" ในบล็อกของเขา ผู้เชี่ยวชาญด้านคอมพิวเตอร์ควอนตัมเสริมว่าข้อมูลที่ IBM เผยแพร่จนถึงตอนนี้ยังขาดพารามิเตอร์สำคัญที่เขาใช้ในการประเมินความก้าวหน้าของคอมพิวเตอร์ควอนตัมในปี 2559 IBM เป็นบริษัทแรกที่นำการประมวลผลควอนตัมมาสู่คลาวด์ โดยเปิดโอกาสให้ผู้ใช้เข้าถึงเครื่องจักรได้มากขึ้น [17]

2.16 งานวิจัยที่เกี่ยวข้อง

Ryan LaRose กล่าววไว้ว่าคอมพิวเตอร์ควอนตัมพร้อมใช้งานบนระบบคลาวด์ แต่การขยายตัวของแพลตฟอร์มซอฟต์แวร์ควอนตัมเมื่อเร็ว ๆ นี้อาจทำให้ผู้ตัดสินใจว่าจะใช้อะไรอย่างล้นหลาม ในบทความนี้ เรานำเสนอภาพปัจจุบันของภูมิทัศน์คอมพิวเตอร์ควอนตัมที่พัฒนาอย่างรวดเร็วโดยเปรียบเทียบ ที่ทำให้นักวิจัยสามารถ ใช้อุปกรณ์ควอนตัมจริงและจำลอง การวิเคราะห์ของเราครอบคลุมข้อกำหนดและการติดตั้งไวยากรณ์ภาษาผ่านโปรแกรมตัวอย่าง การสนับสนุน

ไลบรารี และความสามารถจำลองควอนตัมสำหรับแต่ละแพลตฟอร์ม สำหรับแพลตฟอร์มที่รองรับคอมพิวเตอร์ควอนตัม เราจะเปรียบเทียบฮาร์ดแวร์ ภาษาแอสเซมบลีควอนตัม และคอมไพเลอร์ควอนตัม เราสรุปโดยครอบคลุมคุณสมบัติของแต่ละรายการและกล่าวถึงแพ็คเกจซอฟต์แวร์คอมพิวเตอร์ควอนตัมอื่น ๆ โดยสังเขป [18]

Damian S. Steiger, Thomas Häner และ Matthias Troyer กล่าววไว้ว่า ProjectQ ซึ่งเป็นความพยายามของซอฟต์แวร์โอเพ่นซอร์สสำหรับการคำนวณควอนตัม รุ่นแรกมีเฟรมเวิร์กของคอมไพเลอร์ที่สามารถกำหนดเป้าหมายฮาร์ดแวร์ประเภทต่างๆ ได้ เครื่องจำลองประสิทธิภาพสูงพร้อมความสามารถในการจำลอง และปลั๊กอินคอมไพเลอร์สำหรับการวาดวงจรและการประเมินทรัพยากร เราแนะนำภาษาเฉพาะโดเมนที่ฝังด้วย Python นำเสนอคุณสมบัติและจัดเตรียมตัวอย่างการใช้งานสำหรับอัลกอริธึมควอนตัม กรอบงานอนุญาตให้ทดสอบอัลกอริธึมควอนตัมผ่านการจำลองและเปิดใช้งานบนฮาร์ดแวร์ควอนตัมจริงโดยใช้แบ็คเอนด์ที่เชื่อมต่อกับบริการคลาวด์ IBM Quantum Experience ด้วยกลไกการขยาย ผู้ใช้สามารถจัดเตรียมแบ็คเอนด์ให้กับฮาร์ดแวร์ควอนตัมเพิ่มเติม และนักวิทยาศาสตร์ที่ทำงานเกี่ยวกับการคอมไพล์ควอนตัมสามารถจัดหาปลั๊กอินสำหรับการคอมไพล์ การเพิ่มประสิทธิภาพ การสังเคราะห์เกต และกลยุทธ์การจัดวาง [19]

Thomas Häner, Damian S Steiger, Krysta Svore and Matthias Troyer ได้กล่าวไว้ว่า ระเบียบวิธีซอฟต์แวร์สำหรับการรวบรวมโปรแกรมควอนตัมคอมพิวเตอร์ควอนตัมสัญญาว่าจะเปลี่ยนแนวคิดเรื่องการคำนวณของเราโดยเสนอกระบวนทัศน์ใหม่ทั้งหมด เพื่อให้ได้การคำนวณควอนตัมที่ปรับขนาดได้ การเพิ่มประสิทธิภาพคอมไพเลอร์และขั้นตอนการออกแบบซอฟต์แวร์ที่เกี่ยวข้องจะมีความจำเป็น เรานำเสนอสถาปัตยกรรมซอฟต์แวร์สำหรับการคอมไพล์โปรแกรมควอนตัมจากโปรแกรมภาษาระดับสูงไปจนถึงคำแนะนำเฉพาะฮาร์ดแวร์ เราอธิบายเลเยอร์ที่จำเป็นของนามธรรมและความแตกต่างและความคล้ายคลึงกันกับเลเยอร์คลาสสิกของโฟลว์การออกแบบโดยใช้คอมพิวเตอร์ช่วย สำหรับแต่ละเลเยอร์ของสแต็ก เราจะพูดถึงวิธีการพื้นฐานสำหรับการคอมไพล์และการปรับให้เหมาะสม ระเบียบวิธีซอฟต์แวร์ของเราอำนวยความสะดวกในการสร้างสรรค์นวัตกรรมที่รวดเร็วยิ่งขึ้นในหมู่นักออกแบบอัลกอริทึมควอนตัม วิศวกรฮาร์ดแวร์ควอนตัม และนักทดลอง ช่วยให้สามารถคอมไพล์อัลกอริธึมควอนตัมที่ซับซ้อนที่ปรับขนาดได้และสามารถกำหนดเป้าหมายไปยังการใช้งานฮาร์ดแวร์ควอนตัมเฉพาะใดๆ [20]

Danah Zohar กล่าวไว้ว่าจากยุคนิวโทเนียนถึงยุคควอนตัม ฟิสิกส์ "กลไก" แบบคลาสสิก ซึ่งเป็นผลงานของนักวิทยาศาสตร์ในศตวรรษที่สิบหกหลายคน ได้รับการสรุปและประมวลผลใน Principia Mathematica ของไอแซก นิวตันในช่วงต้นศตวรรษที่สิบเจ็ด ดังนั้นในรูปแบบที่นิยมจึงมักเรียกว่า "ฟิสิกส์ของนิวตัน" และฉันจะปฏิบัติตามการใช้งานนั้น กฎการ

เคลื่อนที่สามข้อของนิวตัน (บวกกฎความโน้มถ่วงของเขา) บรรยายถึงเอกภพที่เรียบง่าย ปฏิบัติตามกฎ คาดการณ์ได้ และควบคุมได้ คำอุปมาเรื่องจักรวาลของเขาคือ "เครื่องจักร" ที่หล่อเลี้ยงอย่างดี โมเดลนี้ให้กำเนิดจิตใจสมัยใหม่ ฟิสิกส์ของนิวตันมีอิทธิพลอย่างมากจนทำให้ฟิสิกส์ของนิวตันเป็นแบบอย่างสำหรับนักคิดที่ตามมา ในทุกๆ ด้านของความคิดในอีก 300 ปีข้างหน้า— Freud ในด้านจิตวิทยา Comte ในสังคมวิทยา Locke & Mill ในปรัชญาการเมือง Adam Smith ในด้านเศรษฐศาสตร์ และ Frederick เทย์เลอร์ในการคิดเชิงบริหาร (“Scientific Management” หรือ “Taylorism”) ดังนั้น สำหรับสาวกของนักคิดพื้นฐานเหล่านี้ และสำหรับบุคคลทั่วไปในวงกว้าง ไม่ว่าพวกเขาจะรู้หรือไม่ และไม่ว่าพวกเขาจะรู้อะไรเกี่ยวกับฟิสิกส์หรือไม่ก็ตาม การคิดแบบนิวโทเนียนได้กำหนดวิธีคิดของพวกเขา วิธีที่พวกเขาคิด รับรู้ วิธีที่พวกเขาเกี่ยวข้อง วิธีที่พวกเขาจัดระเบียบตัวเอง วิธีที่พวกเขากระทำ สิ่งที่พวกเขาเห็นคุณค่า และสิ่งที่พวกเขาคิดค้น ยุคที่สิบแปดถึงศตวรรษที่ยี่สิบเป็นยุคอุตสาหกรรม แต่ก็สามารถเรียกได้อย่างถูกต้องว่ายุคนิวโทเนียน สามศตวรรษนี้ทำให้เราก้าวหน้าอย่างที่ไม่เคยมีมาก่อน สร้างความมั่งคั่งมหาศาลและผลประโยชน์มากมาย แต่ด้วยความคิดแบบเดียวกันและค่านิยมที่เกี่ยวข้องกันนั้น ยังก่อให้เกิดปัญหาที่ร้ายแรง มีอยู่จริง ซึ่งเราเผชิญอยู่ในศตวรรษที่ 21 นี้ ได้แก่ การเปลี่ยนแปลงสภาพภูมิอากาศ การมีประชากรมากเกินไป การอพยพจำนวนมาก ความไม่เท่าเทียมกัน การขาดแคลนอาหารและน้ำ การคุกคามของการสูญพันธุ์ของนิวเคลียร์ และประเด็นด้านอัตลักษณ์ที่เป็นรากฐานของความวุ่นวายทางการเมืองแบบประชานิยมในปัจจุบันในยุโรปและอเมริกา (Brexit & Trump และอื่น ๆ) ในตอนต้นของศตวรรษที่ยี่สิบ ฟิสิกส์ใหม่ได้ถือกำเนิดขึ้น ทฤษฎีสัมพัทธภาพข้อแรกของไอน์สไตน์ ต่อด้วยฟิสิกส์ควอนตัม และต่อมาคือ ทฤษฎีความโกลาหลและวิทยาศาสตร์ที่ซับซ้อน ในบรรดาสิ่งเหล่านี้ ฟิสิกส์ควอนตัมเป็นพื้นฐานทางปรัชญามากที่สุด มันอธิบายจักรวาลว่าซับซ้อน ไม่แน่นอน คาดเดาไม่ได้ และจัดระเบียบตนเอง นอกจากนี้ยังเป็นฟิสิกส์ควอนตัมที่ก่อให้เกิดเทคโนโลยีใหม่ทั้งหมดที่หล่อหลอมและกำหนดชีวิตในศตวรรษที่ 21 ชิปซิลิกอนที่ช่วยให้คอมพิวเตอร์ สมาร์ทโฟน บ้านและเมือง อินเทอร์เน็ต และเทคโนโลยี 5G ที่ปฏิวัติวงการเร็วๆ นี้จะมาถึง ขึ้นอยู่กับกระบวนการควอนตัมในการทำงาน เป็นเรื่องที่ยุติธรรมที่จะบอกว่าด้วยการประดิษฐ์ชิปซิลิกอน เราทิ้งยุคอุตสาหกรรมและเข้าสู่ "ยุคควอนตัม" ในหนังสือเล่มก่อนๆ ของฉันทั้งหมด ฉันได้โต้แย้งว่าการเปลี่ยนแปลงพื้นฐานที่เกิดขึ้นจากการค้นพบฟิสิกส์ควอนตัมก่อให้เกิดกระบวนทัศน์ใหม่ ซึ่งมีพลังในทุกวิถีทางเช่นเดียวกับกระบวนทัศน์ของนิวตันที่กำลังเข้ามาแทนที่ เช่นเดียวกับการเปลี่ยนกระบวนทัศน์ที่นักวิทยาศาสตร์ได้ประสบด้วยตนเอง การเปลี่ยนกระบวนทัศน์ในการคิดของมนุษย์จะทำให้ทุกอย่างที่เราคุ้นเคยกลับหัวกลับหางและกลับด้าน มันจะสร้างทฤษฎีใหม่ โมเดลส่วนบุคคล สังคม และการเมืองใหม่ แนวทางแก้ไขปัญหาใหม่ (และปัญหาใหม่สองสามอย่างในตัวมัน

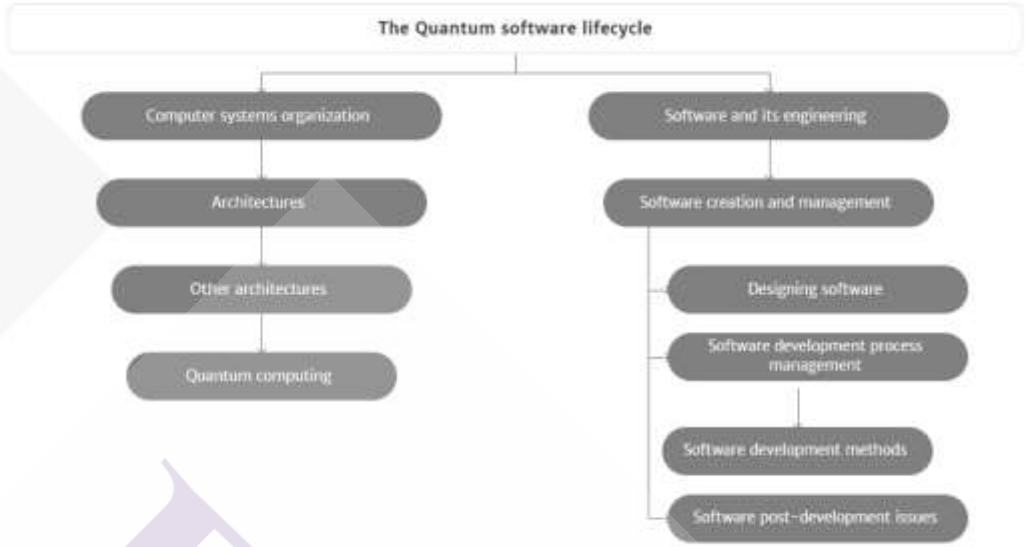
เอง) และแน่นอน เทคโนโลยีใหม่ที่ไม่มีที่สิ้นสุด เราเห็นผู้อุปถัมภ์ของจิตวิทยาควอนตัมใหม่ ทฤษฎีสังคมควอนตัม และชีววิทยาควอนตัมแล้ว หนังสือเล่มนี้รวมเอาทั้งหมดไว้ในทฤษฎีการจัดการควอนตัมใหม่ที่ครอบคลุม [21]

Katherine L. Brown ,William J. Munro และ Vivien M. Kendon ได้กล่าวว่าการจำลองเชิงตัวเลขของระบบควอนตัมมีความสำคัญอย่างยิ่งต่อการทำความเข้าใจปรากฏการณ์ทางธรรมชาติของเรา ระบบต่างๆ ที่น่าสนใจและมีความสำคัญ ในด้านต่างๆ เช่น วัสดุตัวนำยิ่งยวดและเคมีควอนตัม ได้รับการพิจารณาว่าสามารถอธิบายได้ด้วยแบบจำลองที่เราไม่สามารถแก้ได้ด้วยความแม่นยำที่เพียงพอ ทั้งในด้านการวิเคราะห์หรือเชิงตัวเลขด้วยคอมพิวเตอร์คลาสสิก การใช้คอมพิวเตอร์ควอนตัมเพื่อจำลองระบบควอนตัมดังกล่าวถือเป็นการประยุกต์ใช้หลักในการคำนวณควอนตัมตั้งแต่เริ่มต้นภาคสนามในช่วงทศวรรษ 1980 นอกจากนี้ คาดว่าผลลัพธ์ที่เป็นประโยชน์นอกเหนือจากการคำนวณแบบคลาสสิกจะสามารถเข้าถึงได้ด้วยน้อยกว่าร้อย qubits ทำให้การจำลองควอนตัมอาจเป็นหนึ่งในการประยุกต์ใช้คอมพิวเตอร์ควอนตัมที่เก่าแก่ที่สุด ในบทความนี้เราสำรวจการพัฒนาเชิงทฤษฎีและเชิงทดลองของการจำลองควอนตัมโดยใช้คอมพิวเตอร์ควอนตัมตั้งแต่แนวคิดแรกไปจนถึงความพยายามในการวิจัยอย่างเข้มข้นที่กำลังดำเนินการอยู่ [22]

Jonathan R. Friedman, Vijay Patel, W. Chen, S. K. Tolpygo และ J. E. Lukens ได้กล่าวว่าในปี ค.ศ. 1935 Schrödinger1 พยายามที่จะแสดงให้เห็นถึงข้อจำกัดของกลศาสตร์ควอนตัมโดยใช้การทดลองทางความคิดซึ่งแมวถูกวางลงในควอนตัมทับซ้อนของสภาวะที่มีชีวิตและตาย แนวคิดนี้ยังคงเป็นความอยากรู้อยากเห็นทางวิชาการจนถึงช่วงทศวรรษ 1980 เมื่อมันถูกเสนอ2,3,4 ว่าภายใต้สภาวะที่เหมาะสม วัตถุมหภาคที่มีระดับความเป็นอิสระด้วยกล้องจุลทรรศน์หลายระดับสามารถทำงานแบบกลไกควอนตัมได้ โดยมีเงื่อนไขว่าจะต้องแยกออกจากสิ่งแวดล้อมอย่างเพียงพอ แม้ว่าจะมีความคืบหน้าอย่างมากในการแสดงพฤติกรรมควอนตัมระดับมหภาคของระบบต่างๆ เช่น ตัวนำยิ่งยวด5,6,7,8,9, แม่เหล็กระดับนาโน10,11,12, ไอออนที่ดักจับที่ระบายความร้อนด้วยเลเซอร์13, โฟตอนในโพรงไมโครเวฟ14 และโมเลกุลC6015 ไม่มีการสาธิตการทดลองของการซ้อนทับควอนตัมของสถานะที่แตกต่างกันอย่างแท้จริงในระดับมหภาค ที่นี่ เรานำเสนอหลักฐานการทดลองว่าอุปกรณ์รบกวนควอนตัมที่มีตัวนำยิ่งยวด (SQUID) สามารถใส่ในการทับซ้อนของสถานะฟลักซ์แม่เหล็กสองสถานะ: สถานะหนึ่งสอดคล้องกับไมโครแอมแปร์สองสามตัวของกระแสตามเข็มนาฬิกา อีกส่วนหนึ่งสอดคล้องกับปริมาณของกระแสที่ไหลทวนเข็มนาฬิกาเท่ากัน [23]

Frank Leymann, Johanna Barzen และ Michael Falkenthal ได้คาดว่าคอมพิวเตอร์ควอนตัมจะแก้ปัญหาในโลกแห่งความเป็นจริงได้ กลายเป็นสินค้าทั่วไปภายในไม่กี่ปีถัดไป แต่ซอฟต์แวร์สำหรับควอนตัม คอมพิวเตอร์ต้องการทักษะที่แตกต่างกันมากเมื่อเทียบกับการสร้างซอฟต์แวร์สำหรับ คอมพิวเตอร์หรือเครือข่ายแบบเดิม ดังนั้นแนวทางที่ขับเคลื่อนโดยชุมชนเพื่อการสร้างซอฟต์แวร์สำหรับคอมพิวเตอร์ควอนตัมจะส่งเสริมการใช้สิ่งนี้อย่างแพร่หลายนวัตกรรมเทคโนโลยี นอกจากนี้ แพลตฟอร์มสำหรับซอฟต์แวร์ควอนตัมอาจให้โมเดลธุรกิจสำหรับผู้ใช้หลายกลุ่ม [24]

Benjamin Weder, Johanna Barzen, Frank Leymann, Marie Salm, Daniel Vietz, Authors Info และ Claims ได้กล่าวว่าคอมพิวเตอร์ควอนตัมเป็นกระบวนทัศน์ที่เกิดขึ้นใหม่ซึ่งช่วยให้สามารถแก้ปัญหาต่างๆ ได้อย่างมีประสิทธิภาพมากกว่าที่เป็นไปได้บนคอมพิวเตอร์แบบคลาสสิก เนื่องจากคอมพิวเตอร์ควอนตัมเครื่องแรกพร้อมใช้งาน อัลกอริทึมควอนตัมจึงสามารถนำไปใช้และดำเนินการกับฮาร์ดแวร์ควอนตัมจริงได้ อย่างไรก็ตาม ความสามารถของคอมพิวเตอร์ควอนตัมในปัจจุบันมีจำกัด และการคำนวณควอนตัมมักถูกรบกวนด้วยข้อผิดพลาดบางอย่าง ดังนั้นจำเป็นต้องมีการวิจัยเพิ่มเติมเพื่อพัฒนาหรือปรับปรุงอัลกอริทึมควอนตัม คอมพิวเตอร์ควอนตัมหรือการสนับสนุนเครื่องมือซอฟต์แวร์ที่จำเป็น เนื่องจากลักษณะสหวิทยาการของการคำนวณควอนตัม จำเป็นต้องมีความเข้าใจทั่วไปเกี่ยวกับวิธีการพัฒนาและดำเนินการแอปพลิเคชันซอฟต์แวร์ควอนตัม อย่างไรก็ตาม ในปัจจุบันยังไม่มีวิธีการหรือวงจรชีวิตที่ประกอบด้วยขั้นตอนที่เกี่ยวข้องทั้งหมดที่สามารถเกิดขึ้นได้ในระหว่างกระบวนการพัฒนาและดำเนินการ ดังนั้นในบทความนี้ เราจึงแนะนำวงจรชีวิตของซอฟต์แวร์ควอนตัมที่ประกอบด้วยสิบขั้นตอนที่แอปพลิเคชันซอฟต์แวร์ควอนตัมแบบใช้เกทควรผ่าน เราวิเคราะห์วัตถุประสงค์ของแต่ละขั้นตอน วิธีการและเครื่องมือที่สามารถใช้ได้ และปัญหาที่เปิดอยู่หรือคำถามการวิจัย ดังนั้น วงจรชีวิตสามารถใช้เป็นพื้นฐานสำหรับการอภิปรายและการวิจัยในอนาคต [25]



ภาพที่ 2.9 The Quantum software lifecycle

หมายเหตุ: จาก https://dl.acm.org/doi/abs/10.1145/3412451.3428497

Mark Fingerhuth ,Tomáš Babej and Peter Wittek ได้กล่าวว่าซอฟต์แวร์โอเพ่นซอร์สกำลังมีความสำคัญในการออกแบบและทดสอบอัลกอริธึมควอนตัม เครื่องมือหลายอย่างได้รับการสนับสนุนจากผู้ค้าเชิงพาณิชย์รายใหญ่โดยมีเป้าหมายเพื่อให้การพัฒนาซอฟต์แวร์ควอนตัมง่ายขึ้น: สิ่งนี้สะท้อนให้เห็นว่าเฟรมเวิร์กการเรียนรู้ของเครื่องเปิดที่ได้รับทุนสนับสนุนอย่างดีช่วยให้สามารถพัฒนาโมเดลที่ซับซ้อนและดำเนินการกับฮาร์ดแวร์ที่ซับซ้อนเท่ากันได้อย่างไร เราตรวจสอบซอฟต์แวร์โอเพนซอร์ซที่หลากหลายสำหรับการคำนวณควอนตัม ครอบคลุมทุกขั้นตอนของควอนตัมทูลเชนตั้งแต่อินเทอร์เฟซฮาร์ดแวร์ควอนตัมผ่านคอมไพเลอร์ควอนตัมไปจนถึงการใช้งานอัลกอริธึมควอนตัมตลอดจนกระบวนทัศน์การคำนวณควอนตัมทั้งหมด รวมถึงการหลอมด้วยควอนตัม และแบบไม่ต่อเนื่องและต่อเนื่อง การคำนวณควอนตัมรุ่นเกทแบบแปรผัน การประเมินแต่ละโครงการครอบคลุมคุณลักษณะต่างๆ เช่น เอกสารประกอบ ใบอนุญาต การเลือกภาษาโปรแกรม การปฏิบัติตามบรรทัดฐานของวิศวกรรมซอฟต์แวร์ และวัฒนธรรมของโครงการ เราพบว่าถึงแม้ความหลากหลายของโครงการจะน่าดึงดูดใจ แต่มีเพียงไม่กี่คนเท่านั้นที่ดึงดูดนักพัฒนาจากภายนอก และแม้แต่เฟรมเวิร์กที่ได้รับการสนับสนุนในเชิงพาณิชย์จำนวนมากก็ยังมีข้อบกพร่องในด้านวิศวกรรมซอฟต์แวร์ จากการสังเกตเหล่านี้ เราเน้นแนวปฏิบัติที่ดีที่สุด

ที่สามารถส่งเสริมชุมชนที่กระตือรือร้นมากขึ้นเกี่ยวกับซอฟต์แวร์การคำนวณควอนตัมที่ยินดีต้อนรับผู้มาใหม่สู่ภาคสนาม แต่ยังช่วยให้มั่นใจว่าโค้ดมีคุณภาพสูงและมีเอกสารครบถ้วน [26]

I.M. Georgescu, S. Ashhab, and Franco Nori ได้กล่าวไว้ว่าการจำลองกลศาสตร์ควอนตัมเป็นที่รู้จักกันว่าเป็นปัญหาการคำนวณที่ยาก โดยเฉพาะอย่างยิ่งเมื่อต้องรับมือกับระบบขนาดใหญ่ อย่างไรก็ตาม ปัญหานี้อาจเอาชนะได้โดยใช้ระบบควอนตัมที่ควบคุมได้บางระบบเพื่อศึกษาระบบควอนตัมอื่นที่ควบคุมได้น้อยกว่าหรือเข้าถึงได้ เช่น การจำลองควอนตัม การจำลองควอนตัมสัญญาว่าจะนำไปใช้ในการศึกษาปัญหาต่างๆ เช่น ฟิสิกส์ของสสารควบแน่น ฟิสิกส์พลังงานสูง ฟิสิกส์อะตอม เคมีควอนตัม และจักรวาลวิทยา การจำลองควอนตัมสามารถทำได้โดยใช้คอมพิวเตอร์ควอนตัม แต่ยังมีอุปกรณ์แอนะล็อกที่ง่ายกว่าซึ่งต้องการการควบคุมน้อยกว่า ดังนั้นจึงสร้างได้ง่ายกว่า ระบบควอนตัมจำนวนหนึ่ง เช่น อะตอมที่เป็นกลาง ไอออน โมเลกุลขั้ว อิเล็กตรอนในเซมิคอนดักเตอร์ วงจรตัวนำยิ่งยวด สปินนิวเคลียร์ และโฟตอน ได้รับการเสนอให้เป็นตัวจำลองควอนตัม การทบทวนนี้สรุปประเด็นหลักในเชิงทฤษฎีและการทดลองของการจำลองควอนตัม และเน้นย้ำถึงความท้าทายและคำสัญญาบางประการของสาขาที่เติบโตอย่างรวดเร็วนี้ [27]

Robert Wille Rod Van Meter และ Yehuda Naveh ได้กล่าวว่าคอมพิวเตอร์ควอนตัมสัญญาว่าจะเร่งความเร็วมากกว่าเครื่องจักรทั่วไปสำหรับการใช้งานจริงมากมาย แม้ว่าจะถือว่าเป็น "ความฝันแห่งอนาคต" มาเป็นเวลานาน แต่คอมพิวเตอร์ควอนตัมเครื่องแรกก็พร้อมให้ใช้งานแล้วในตอนนี้ ซึ่งใครๆ ก็สามารถใช้งานได้ แรงผลักดันสำคัญในการพัฒนานี้คือ IBM Research ซึ่งเปิดตัว IBM Q Experience ซึ่งเป็นความคิดริเริ่มทางอุตสาหกรรมครั้งแรกในการสร้างคอมพิวเตอร์ควอนตัมสากลและทำให้ผู้ชมทั่วไปเข้าถึงได้ผ่านการเข้าถึงระบบคลาวด์ โครงการริเริ่มนี้ได้เปิดตัวเครื่องมือ Qiskit ซึ่งช่วยให้นักวิจัย ครู นักพัฒนา และผู้สนใจทั่วไปสามารถเขียนโค้ดที่เกี่ยวข้องและเรียกใช้การทดลองบนเครื่องเหล่านั้นได้ ในขณะเดียวกัน ก็เป็นสนามเด็กเล่นในอุดมคติสำหรับชุมชนการออกแบบอัตโนมัติ ซึ่งผ่าน Qiskit ก็สามารถปรับใช้โซลูชันที่ปรับปรุงใหม่ได้ เช่น เกี่ยวกับการออกแบบและการใช้งานควอนตัม สรุปเซสชั่นพิเศษนี้มีจุดมุ่งหมายเพื่อให้คำแนะนำเบื้องต้นเกี่ยวกับ Qiskit และนำเสนอเรื่องราวความสำเร็จที่เลือกสรรเกี่ยวกับวิธีการทำงานและพัฒนาสำหรับสิ่งนี้ นอกจากนั้น ยังให้การอ้างอิงที่สอดคล้องกันสำหรับการอ่านเพิ่มเติมในแง่ของบทช่วยสอนและเอกสารทางวิทยาศาสตร์ ตลอดจนลิงก์ไปยังการใช้งานที่เผยแพร่ต่อสาธารณะสำหรับส่วนขยาย Qiskit [28]

Daniel Koch, Laura Wessing และ Paul M. Alsing ได้กล่าวว่าในขณะที่วงการคอมพิวเตอร์ควอนตัมเติบโตขึ้นเรื่อย ๆ ประชาชนทั่วไปก็สนใจที่จะทดสอบคอมพิวเตอร์ควอนตัม

ที่เผยแพร่ต่อสาธารณะด้วยเช่นกัน อย่างไรก็ตาม หลายคนอาจพบว่าการเรียนรู้ข้อมูลเสริมทั้งหมดที่เข้าสู่อัลกอริธึมควอนตัมเป็นงานที่น่ากลัวและท้อแท้ บทช่วยสอนนี้เป็นชุดบทเรียนที่มุ่งสอนพื้นฐานของอัลกอริทึมควอนตัมให้กับผู้ที่อาจมีพื้นฐานเพียงเล็กน้อยหรือไม่มีเลยในฟิสิกส์ควอนตัมและ/หรือความรู้เพียงเล็กน้อยเกี่ยวกับการเขียนโปรแกรมในหลาม แต่ละบทเรียนครอบคลุมหัวข้อฟิสิกส์/การเข้ารหัสที่เลือกซึ่งจำเป็นสำหรับการเขียนอัลกอริธึมควอนตัม ในที่สุดก็สร้างชุดเครื่องมือเพื่อจัดการกับอัลกอริธึมควอนตัมที่ท้าทายมากขึ้นเรื่อยๆ ชุดบทช่วยสอนนี้ออกแบบมาเพื่อให้บริการสองบริการแก่ผู้อ่านจากภูมิหลังใดๆ: 1) ความเข้าใจอย่างรัดกุมและละเอียดถี่ถ้วนเกี่ยวกับอัลกอริธึมควอนตัมที่เป็นที่นิยม/มีความสำคัญทางวิชาการ 2) ความเข้าใจอย่างคล่องแคล่วเกี่ยวกับวิธีการเขียนโค้ดสำหรับอัลกอริธึมควอนตัม โดยใช้ Pyquil ที่เผยแพร่ต่อสาธารณะของ Rigetti [29]

Miguel-Angel Sicilia, Salvador Sánchez-Alonso, Marçal Mora-Cantallops และ Elena García-Barriocanal ได้กล่าวว่ามีการเสนอและใช้งานภาษาโปรแกรมควอนตัมระดับสูงจำนวนมากในปีที่ผ่านมา ข้อเท็จจริงนี้เปิดโอกาสในการศึกษาโครงสร้างของซอร์สโค้ดของซอฟต์แวร์ควอนตัม โดยใช้เมตริกเดียวกันกับที่ใช้ในซอฟต์แวร์คลาสสิกในขั้นต้น ที่นี่ เรารายงานการศึกษาเบื้องต้นเกี่ยวกับโครงสร้างโมดูลและการใช้ประตูควอนตัมในไลบรารีของแพลตฟอร์มการพัฒนาควอนตัมของ Microsoft QDK (Quantum Developer Kit) ที่ใช้ภาษาเฉพาะอย่าง Q# โครงสร้างการพึ่งพาและการใช้พื้นฐานได้รับการวิเคราะห์ในซอร์สโค้ดทั้งหมดที่มีอยู่ในที่เก็บ Github ที่เกี่ยวข้องกับแพลตฟอร์มจนถึงปัจจุบัน [30]

G. L. Long ได้กล่าวไว้ว่าในอัลกอริธึมมาตรฐานของ Grover สำหรับการค้นหาควอนตัม ความน่าจะเป็นในการค้นหารายการที่ทำเครื่องหมายไว้ไม่ใช่ 1 อย่างแน่นอน ในบทความนี้ เรานำเสนออัลกอริธึมของ Grover เวอร์ชันที่แก้ไขแล้วซึ่งค้นหาสถานะที่ทำเครื่องหมายด้วยอัตราความสำเร็จเต็มจำนวน การปรับเปลี่ยนทำได้โดยการแทนที่การกลับเฟสโดยการหมุนเฟสผ่านมุม φ. มุมการหมุนจะได้รับการวิเคราะห์เป็น $\varphi = 2\arcsin(\sin[\pi/(4J+6)]/\sin\beta)$, where $\sin\beta = 1/\sqrt{N}$, N คือจำนวนรายการในฐานข้อมูล และ J คือจำนวนเต็มใดๆ ที่เท่ากับหรือมากกว่าส่วนของจำนวนเต็มของ $[(\pi/2)-\beta]/(2\beta)$ เมื่อวัดที่ (J+1) ซ้ำแล้วซ้ำเล่า ได้สถานะที่ทำเครื่องหมายไว้อย่างแน่นอน [31]

บทที่ 3
วิธีการดำเนินงาน

ในบทนี้กล่าวถึงองค์ประกอบและฟังก์ชันของระบบที่ใช้ในการ ภาพรวมและเปรียบเทียบซอฟต์แวร์ควอนตัมแพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) โดยนำ 2 แพลตฟอร์มที่ดีที่สุดมาวัดประสิทธิภาพการทำงานแบบจำลองโดยใช้ Shor's algorithm รูปแบบและวิธีการทดลอง ผลที่บันทึกได้จากการทดลองการคำนวณผลที่บันทึกได้ และเกณฑ์ในการทดสอบการทำงาน

3.1 ภาพรวมของระบบและหน้าที่ในการทดลอง

3.2 แผนการดำเนินงาน

3.3 เครื่องมือที่ใช้ในงานวิจัย

3.4 ปัจจัยที่ส่งผลต่อการทำงานของระบบตรวจหาการบุกรุก

3.5 วิธีการทดลอง

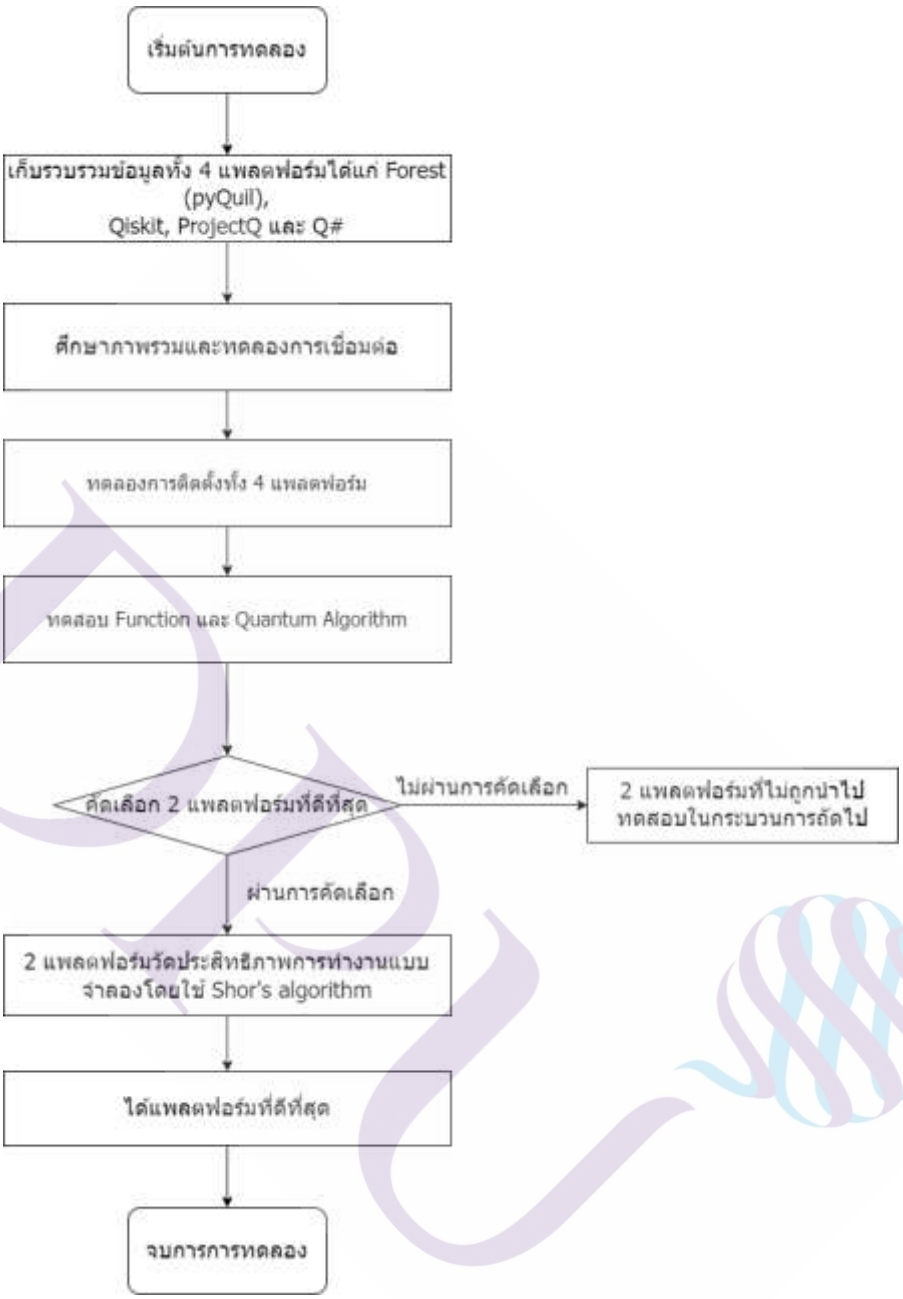
3.6 บันทึกผลจากการทดลอง

3.7 การคำนวณผลที่ได้จากการทดลอง

3.8 เกณฑ์ในการเปรียบเทียบประสิทธิภาพการตรวจจับการบุกรุก

3.1 ภาพรวมของระบบและหน้าที่ในการทดลอง

การออกแบบที่ใช้ในการทดลองครั้งนี้ปประกอบด้วย เก็บรวบรวมข้อมูล Quantum Software Platforms ทั้ง 4 ตัว, แสดงการติดตั้งโปรแกรมและสิ่งที่ Platforms ทั้ง 4 ตัวต้องการ, ทดสอบ Algorithm ที่รองรับแต่ละ Platforms, แสดงภาพรวม Quantum Software Platforms ทั้ง 4 ตัว, นำ 2 Platforms ที่ดีที่สุดมาวัดวัดประสิทธิภาพการทำงานแบบจำลอง โดยการใช้ Shor's algorithm



ภาพที่ 3.1 ภาพของระบบและหน้าที่ในการทดลอง

3.1.1 เก็บรวบรวมข้อมูล Quantum Software Platform ทั้ง 4 Platforms ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) ตั้งแต่ความเป็นมาของ Platforms, Qubit Local Simulator, Register for API Key, Qubit in Cloud, สิทธ์การใช้บน Cloud

3.1.2 การติดตั้งวิเคราะห์ข้อกำหนดและการติดตั้ง Software Platforms (requirements and installation), ภาษาและรูปแบบการเขียนโปรแกรม (language syntax), สิ่งที่สนันสนุน (Library

Support), การจำลองควอนตัม (Simulated Quantum), ภาษาแอสเซมบลีควอนตัม (Quantum assembly Language) และควอนตัมคอมไพเลอร์ (Quantum Compiler) ของ Quantum Software Platform ทั้ง 4 Platforms ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#)

3.1.3 เมื่อติดตั้งทั้ง 4 Platforms ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) นำแต่ละ framework ทดสอบ function และ Quantum Algorithm ต่าง ๆ ว่าสามารถใช้งานอะไรได้บ้าง

3.1.4 แสดงภาพรวมทั้งหมดของ Quantum Software Platform ทั้ง 4 Platforms ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) เพื่อที่จะได้เปรียบเทียบความสามารถและนำ Platform ที่ 2 Platforms ที่ดีที่สุดในการทดลองไปวัดประสิทธิภาพการทำงานแบบจำลองต่อไป

3.1.5 นำ 2 Platforms ที่ดีที่สุดในการเปรียบเทียบภาพรวมมาวัดประสิทธิภาพการทำงานแบบจำลอง (Simulator Performance) โดยการใช้ Shor's algorithm ในการวัดผล

3.2 แผนการดำเนินงาน

แผนการดำเนินดำเนินงาน เพื่อเป็นกรอบการดำเนินงานวิจัยเพื่อให้เสร็จลุล่วงได้ตามวัตถุประสงค์ ซึ่งใช้เป็นแนวทางปฏิบัติงาน

3.2.1 เตรียมการกำหนดหัวข้อสารนิพนธ์

3.2.2 ศึกษาเอกสาร และงานวิจัยที่เกี่ยวข้อง

3.2.3 ค้น คว้าและออกแบบเครื่องมือในการทดสอบ

3.2.4 กำหนดกรอบเวลาในการดาเนินงาน

3.2.5 ดำเนินงานพัฒนาและจัดสร้างระบบจำลองเพื่อการทดสอบ

3.2.6 ทดสอบและบันทึกผลการทดลอง

3.2.7 วิเคราะห์ผลการลอง

3.2.8 สรุปผลการทดลอง

3.2.9 จัดทำสารนิพนธ์ฉบับสมบูรณ์

ผู้วิจัยได้วางแผนการและระยะเวลาดำเนินงานโดยแบ่งขั้นตอนดำเนินงานเป็น 6 ขั้นตอน และใช้ระยะเวลาในการดำเนินงานทั้งสิ้นเป็นเวลา 12 เดือน โดยเริ่มต้นตั้งแต่เดือนมิถุนายน พ.ศ. 2564 จนถึงเดือนพฤษภาคม พ.ศ. 2565 โดยรายละเอียดของแผนการดำเนินงานดูได้จากตารางที่ 3.2.1

ตารางที่ 3.1 แผนการดำเนินงาน

	ปี พ.ศ. 2564 - 2565					
แผนการดำเนินการ	มิ.ย. – ก.ค. 64	ส.ค. – ก.ย. 64	ต.ค. – พ.ย. 64	ธ.ค. – ม.ค. 65	ก.พ. – มี.ค 65	เม.ย. – พ.ค 65
1.ศึกษาค้นคว้าแนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง	→	→				
2.ออกแบบภาพรวมของระบบตรวจจับการบุกรุก		→	→			
3.เก็บข้อมูล		→	→			
4.วิเคราะห์ข้อมูลและเปรียบเทียบประสิทธิภาพ			→	→		
5.รับปรุงแก้ไขระบบ				→	→	
6. สรุปผลการศึกษาและนำเสนอ					→	→

3.3 เครื่องมือที่ใช้ในงานวิจัย

ทำการประเมินจากณครงสร้างและความต้องการของ Quantum Software Platform โดยอิงจากโครงสร้างพื้นฐานและข้อมูลเพื่อให้ตรงกับแนวทางที่ได้ออกแบบเอาไว้ โดยเริ่มจากการจัดหาและจัดเตรียมความพร้อมในส่วนของฮาร์ดแวร์ เพื่อให้เพียงพอก็ความต้องการทดลอง มีพื้นที่มากพอสำหรับการทดสอบ แลพส่วนสุดท้ายคือส่วนของซอฟท์แวร์ในการทดสอบจะนำเสนอ 2 ส่วนใหญ่ ๆ คือ ส่วนแรกเป็นส่วนของฮาร์ดแวร์ ในที่นี้คือคอมพิวเตอร์พกพาที่ใช้ในการทดสอบ ส่วนที่สองซอฟท์แวร์โดยที่ระบุว่างานวิจัยนี้ใช้ซอฟท์แวร์อะไรบ้าง ซอฟท์แวร์ตัวไหน เวอร์ชั่นไหนบ้าง สามารถใช้ภาษาอะไรได้บางในการพัฒนาซอฟท์แวร์ ดังแสดงในตารางที่ 3.3.1

ตารางที่ 3.2 โครงสร้างพื้นฐานของเครื่องมือสำหรับใช้ทดสอบ

อุปกรณ์	ผลิตภัณฑ์	รายละเอียด	หน้าที่
ฮาร์ดแวร์	Dell Notebook	CPU: Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.59 GHz Ram: 12 GB SSD: 512 GB OS: Window 10 (64 Bit)	ใช้ในการติดตั้งซอฟท์แวร์
ซอฟท์แวร์	Qiskit	Version 0.36.1	สำหรับทดสอบการทำงาน
	Forest (pyQuil)	Version 3.1.0	สำหรับทดสอบการทำงาน
	ProjectQ	Version 0.7.3	สำหรับทดสอบการทำงาน
	Quantum Developer Kit (Q#)	Version 0.24.208024	สำหรับทดสอบการทำงาน
	Jupyter notebook	Version 6.3.0	ใช้ในการเขียน code และ run code

เป็นโครงสร้างพื้นฐานของเครื่องมือสำหรับใช้ทดสอบจะระบุรายละเอียดต่างๆ ของอุปกรณ์ฮาร์ดแวร์และซอฟท์แวร์ สำหรับการติดตั้งระบบปฏิบัติการต่างๆ และระบุหน้าที่การทำงานของแต่ละส่วนด้วย

3.4 ปัจจัยที่ส่งผลต่อการทำงานของระบบ Quantum Software Platform

การทดสอบ Quantum Software Platform ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) มีปัจจัยที่เป็นผลกระทบโดยหลักแล้ว Software บางตัวยังไม่สมบูรณ์แบบเท่าที่ควรมี error ที่ยังไม่ได้รับการแก้ไขหายจุดอีกทั้งยังไม่มีเอกสารที่แน่ชัดในการใช้งานใน Function ส่วนต่างๆ อีกทั้งยังไม่ได้เป็นที่สนใจมากนักในปัจจุบันทำให้การศึกษาแต่ละ Platform ค่อนข้างยาก และทั้งนี้บาง Platform มีเว็บไซต์หลักที่ให้ข้อมูลหลายที่แต่ไม่เหมือนกันและไม่ได้ update เอกสารและ code ให้เป็นปัจจุบัน

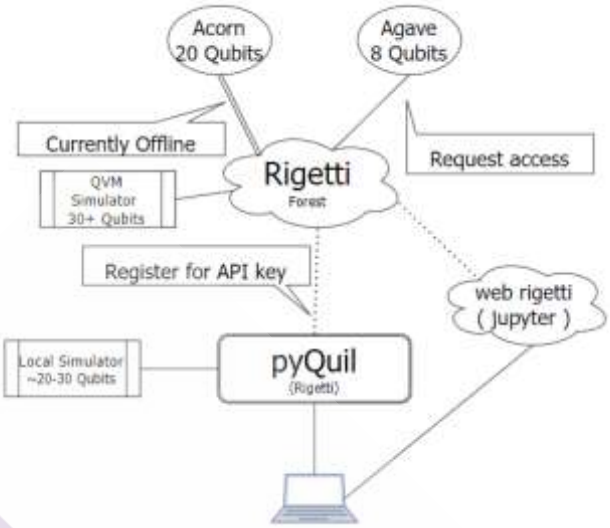
3.5 วิธีการทดลอง

เป็นการทดลองเปรียบเทียบภาพรวมการทำงานของ Quantum Software Platform ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) โดยวัดประสิทธิภาพการทำงานแบบจำลอง (Simulator Performance) โดยการใช้ Shor's algorithm ในการวัดผลโดยในการผลลองนี้จะแบ่งการทดลองออกเป็นดังนี้

3.5.1 ทำการเก็บรวบรวมข้อมูล Quantum Software Platform ทั้ง 4 Platforms ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) ตั้งแต่ความเป็นมาของ Platforms, Qubit Local Simulator, Register for API Key, Qubit in Cloud, สิทธ์การใช้บน Cloud

Forest (pyQuil)

Rigetti เป็นบริษัท start-up ที่ก่อตั้งขึ้นเมื่อปีพ.ศ. 2556 โดยนักฟิสิกส์อเมริกันด้านการคำนวณเชิงควอนตัมชื่อ ดร. Chad Rigetti จากการระดมทุนกว่า 119 ล้านดอลลาร์สหรัฐ (ประมาณ 3.6 พันล้านบาท) สำนักงานใหญ่ตั้งอยู่ที่เมือง Berkeley ในมลรัฐแคลิฟอร์เนีย มีพนักงานจำนวน 120 คน ทำภารกิจประการเดียวคือพัฒนาสร้างคอมพิวเตอร์ควอนตัม มีคอมพิวเตอร์ควอนตัมขนาด 19 คิวบิต ให้ได้ทดลองใช้และพัฒนาควอนตัมอัลกอริทึมผ่านทางออนไลน์ Madhav Thattai หัวหน้าฝ่ายกลยุทธของบริษัททำนายว่าในอีก 10-15 ปี องค์กรขนาดใหญ่ทั้งหลายจะใช้เทคโนโลยีการคำนวณเชิงควอนตัมกันทั้งนั้น

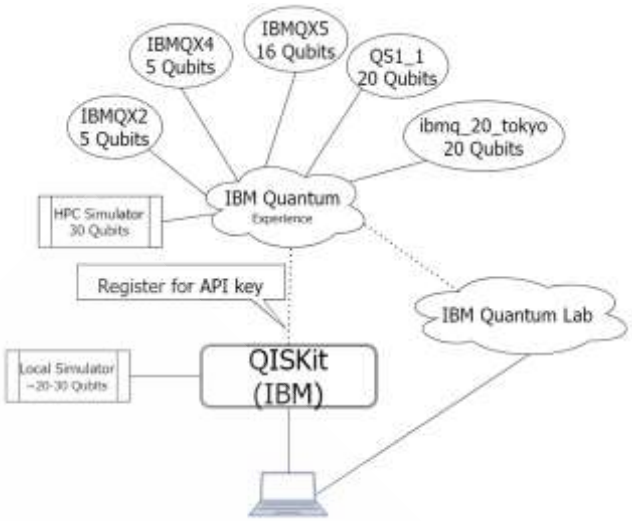


ภาพที่ 3.2 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ Forest (pyQuil)

Forest SDK ประกอบด้วย pyQuil, quilc และ QVM โดยสามารถใช้งาน QVM ที่ Rigetti ได้ 2 วิธีคือการการติดตั้ง SDK ที่ฮาร์ดแวร์ของตัวเองก่อนหรือใช้งานผ่าน Web Rigetti ที่มี Jupyter Notebook ออนไลน์อยู่ โดยทั้ง 2 แบบจะต่อกับ Rigetti Forest สามารถต่อได้เฉพาะแบบ QVM Simulator เท่านั้นไม่สามารถต่อ Quantum Computer จริงได้ต้องมีสิทธิ์เฉพาะในการใช้งาน

Qiskit

IBM เป็นบริษัทสนใจในเรื่องคอมพิวเตอร์ควอนตัมมากว่า 30 ปี โดยมีความก้าวหน้าเป็นลำดับ ปัจจุบันมีศูนย์ IBM Quantum Computation Center ดูแลเรื่องนี้เป็นการเฉพาะ ในปี พ.ศ. 2559 ได้มีการเปิดเว็บไซต์ชื่อ “IBM Q Experience” ที่มีเครื่องคอมพิวเตอร์ควอนตัมขนาด 5 คิวบิต (มีการขยายความเรื่อง qubit ไว้ในภาคผนวกท้ายบทความนี้) ให้บริการฟรีผ่านทางอินเตอร์เน็ตแก่บุคคลทั่วไปที่อยากเรียนรู้และพัฒนาการคำนวณเชิงควอนตัม ต่อมาได้เพิ่มเครื่องขนาด 5 คิวบิต กับ 16 คิวบิตอีกอย่างละเครื่องเข้าไปเสริม ปลายปีพ.ศ. 2560 IBM ได้ประกาศว่ามีเครื่องคอมพิวเตอร์ควอนตัมขนาด 20 คิวบิต จำนวน 2 เครื่องไว้ให้ลูกค้าที่เป็นสมาชิกของ IBM Q Network เช่าเวลาใช้งานผ่านทางระบบคลาวด์ (cloud-based quantum computing system) นอกจากนั้นยังประกาศว่าได้สร้างโปรเซสเซอร์ควอนตัมต้นแบบขนาด 50 คิวบิตสำเร็จแล้ว

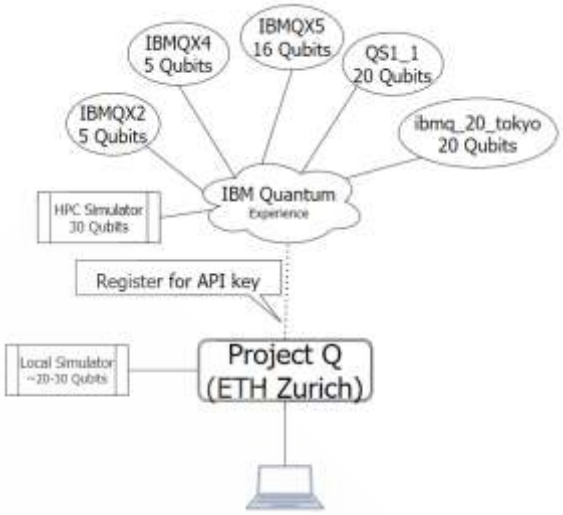


ภาพที่ 3.3 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ Qiskit

การใช้งาน Qiskit เพื่อเชื่อมต่อกับ Quantum Computer จริงนั้นสามารถใช้งานได้แต่ประเภท 5 Qubit เท่านั้นและจะต้องรอเวลาในการทำงานขึ้นอยู่กับจำนวนเข้าคิวการใช้งาน Quantum Computer ในระบบโดยจากติดตั้งที่ฮาร์ดแวร์ของตัวเองหรือเขียน code ที่ IBM Quantum Lab ซึ่งเป็น Jupyter Notebook ออนไลน์อยู่ โดยบนเว็บไซต์ของ IBM

ProjectQ

ETH มักถูกจัดอันดับอยู่ในกลุ่มของมหาวิทยาลัยชั้นนำในระดับนานาชาติอยู่เสมอ ซึ่งในการจัดอันดับแต่ละครั้งนั้น ETH มักจะได้รับอันดับเป็นมหาวิทยาลัยที่ดีที่สุดในประเทศสวิตเซอร์แลนด์ เป็นหนึ่งในมหาวิทยาลัยที่มีอันดับอยู่ในห้าอันดับแรกของยุโรป และเป็นหนึ่งในมหาวิทยาลัยที่มีอันดับอยู่ในสิบอันดับแรกของโลกตลอดระยะเวลาที่ผ่านมาในอดีต ETH ได้รับชื่อเสียงโด่งดังโดยเฉพาะในสาขาวิชาเคมี คณิตศาสตร์ และฟิสิกส์ โดยมีบุคลากรของ ETH ที่ได้รับรางวัลโนเบลจากสาขาต่างๆ ไม่ว่าจะเป็น ไอน์สไตน์เจ้าของทฤษฎีสัมพัทธภาพ เรินเก่นผู้ค้นพบรังสีเอ็กซ์ เพาลี และท่านอื่นจำนวนมากถึง 21 ท่าน โดยนับเฉพาะนักศึกษาที่จบการศึกษาจาก ETH และคณาจารย์ที่ได้รับการยกย่องจากผลงานของพวกเขาที่ได้คิดค้นขึ้นมาที่ ETH เท่านั้นหนึ่งในนั้นคือ ProjectQ

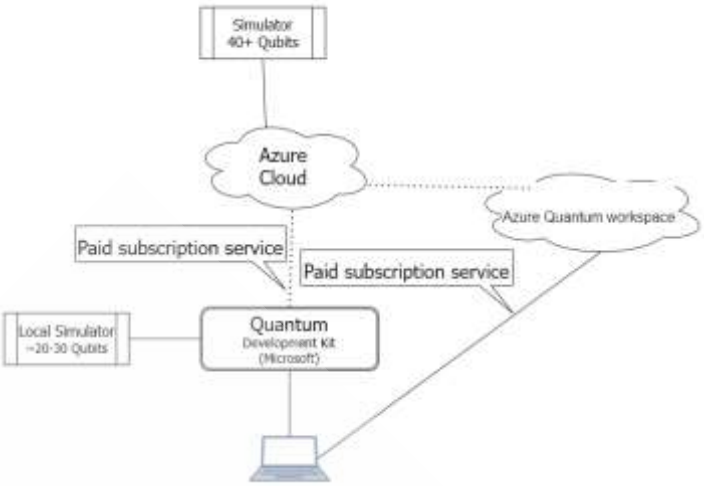


ภาพที่ 3.4 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ ProjectQ

การใช้งาน ProjectQ นั้นสามารถใช้งาน Quantum Computer ร่วมกับ IBM ได้ง่ายแต่ก็ด้วยเงื่อนไขด้วยกันกับ Qiskit ที่สามารถใช้งาน Quantum Computer จริงที่ตัว 5 Qubit ซึ่งเป็น Open-Source ที่ดีมากหนึ่งตัวที่พร้อมจะเติบโตตาม Quantum Computer ของ IBM ไปด้วยแต่มีที่แบบติดตั้งที่ฮาร์ดแวร์ของตัวเองเท่านั้น

Quantum Developer Kit (Q#)

Microsoft: บริษัทแนวหน้าด้านการพัฒนาและผลิตซอฟท์แวร์เจ้านี้ก็กระโจนลงสู่สนามคอมพิวเตอร์ควอนตัมเช่นกัน บริษัทได้จัดตั้งห้องแลป “Station Q” ขึ้นในสถาบันวิจัยชั้นนำหลายแห่ง เช่นที่ UCSB, Purdue University, University of Maryland, University of Sydney, TU Delft, Niels Bohr Institute, ETH Zurich เป็นต้น แต่มีแนวทางต่างจากชาวบ้านค่อนข้างมาก คือสนใจในเทคโนโลยี topological qubit เพราะเชื่อว่าจะมี “ความคลาดเคลื่อน (error)” น้อยกว่าและง่ายกว่าที่จะขยายสู่ระดับการใช้งานเชิงพาณิชย์ จากบทความใน Computer Weekly ฉบับเดือนพฤษภาคม พ.ศ. 2561 รองประธานบริษัท Microsoft ที่รับผิดชอบด้านการคำนวณเชิงควอนตัมให้ความเห็นว่าคอมพิวเตอร์ควอนตัมของบริษัทจะให้บริการเชิงพาณิชย์ผ่านทาง Azure Cloud ในด้านของซอฟท์แวร์ บริษัทได้เปิดให้ได้ชิมลาง quantum computing development kit ซึ่งสามารถดาวน์โหลดได้ฟรี



ภาพที่ 3.5 ภาพรวมระบบการต่อควอนตัมคอมพิวเตอร์ของ Quantum Developer Kit (Q#)

การใช้งาน Quantum Developer Kit (Q#) นั้นสามารถใช้งานได้ 2 วิธีคือการติดตั้งที่ฮาร์ดแวร์ของตัวเองหรือใช้งานผ่าน Azure Quantum workspace ก็ได้แต่ทางการใช้งานค่าย Microsoft นั้นต้องมีค่าใช้จ่ายในการใช้งานถึงแม้ว่าจะเป็นเพียง Simulator เท่านั้น

3.5.2 การติดตั้งวิเคราะห์ข้อกำหนดและการติดตั้ง Software Platforms (requirements and installation), ภาษาและรูปแบบการเขียนโปรแกรม (language syntax), สิ่งที่สนันสนุน (Library Support), การจำลองควอนตัม (Simulated Quantum), ภาษาแอสเซมบลีควอนตัม (Quantum Assembly Language) และควอนตัมคอมไพเลอร์ (Quantum Compiler) ของ Quantum Software Platform ทั้ง 4 Platforms ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#)

Forest (pyQuil)

Forest เป็นแพลตฟอร์มซอฟต์แวร์ควอนตัมที่พัฒนาโดย Rigetti ซึ่งรวมถึง pyQuil ซึ่งเป็นโอเพ่นซอร์ส ควอนตัม โดยที่ Classical Host Language คือ Python สำหรับเขียน code และ run program

ความต้องการและการติดตั้ง (Requirements and Installation)

pyQuil ต้องใช้ Python 3 โดยวิธีง่ายสุดในการติดตั้ง pyQuil คือการใช้ Python pip ในการจัดการติดตั้งสิ่งต่าง ๆ โดยใช้คำสั่งนี้บน Linux Ubuntu

1. pip install pyquil

   หรือทำการติดตั้ง Anaconda โดยพิมพ์คำสั่ง

2. conda install –c rigetti pyquil

หรือคำสั่งนี้เป็นอีกทางเลือกในการดาวน์โหลดและการติดตั้ง source code ที่เก็บไว้บน git

1) git clone http://gitub.com/rigetti/pyquil

2) cd pyquil

3) pip install –e

ในการติดตั้งก็จะได้ Rigetti Quantum Virtual Machine (qvm) ซึ่งช่วยให้สามารถจำลองโปรแกรม Quil ที่มีประสิทธิภาพสูงและได้ Rigetti Quil Compiler (quilc) ซึ่งอนุญาตให้รวบรวมและปรับแต่งโปรแกรม Quil ให้เป็นชุด Gate พื้นฐานการใช้งานก็ใช้คำสั่งดังนี้

```
### CONSOLE 1
$ qvm –S
### CONSOLE 2
$ quilc –S
```

ไวยากรณ์ (Syntax)

pyQuil การเขียนที่เข้าใจได้ง่ายเพราะเขียนบน Python

```
1. from pyquil import Program
2. from pyquil.gates import *
3. from pyquil import get_qc
```

จากบรรทัดที่ 1 การเริ่มเขียนโปรแกรมควอนตัมใน Forest ต้องเริ่มจากการประกาศ Odject ที่ชื่อ Program ซึ่งเป็นสิ่งคำสัญในการเริ่มเขียนโปรแกรม Quil

จากบรรทัดที่ 2 คำสั่งในบรรทัดที่ 1 Program ถูกสร้างขึ้นเพื่อให้เพิ่ม Gates Quantum เข้าไป ซึ่งกำหนดไว้ใน module เราสามารถนำเข้า Gates พื้นฐานทั้งหมดได้จากบรรทัดตัวอย่างเช่น

```
p = Program()
p += X(0)
```

มาเรียกใช้ Fuction Program() และเพิ่มการดำเนินการลงไป เราจะตั้งค่า X gate บน qubit 0

จากบรรทัดที่ 3 คำสั่งจากเชื่อมต่อกับ Object ต่างๆของ QuantumComputerโดยการเชื่อมต่อหรือตั้งค่าต่างๆก็จะใช้ตัวนี้ ตัวอย่างการเชื่อมต่อกับ QVM คือ

```
from pyquil import get_qc
...
qc = get_qc('1q-qvm')
```

```
executable = qc.compile(p)
result = qc.run(executable)
bitstrings = result.readout_data.get('ro')
print(bitstrings)
```

ภาษาควอนตัม (Quantum Language)

ภาษา Quil คล้ายคลึงกับภาษาแอสเซมบลีบนคอมพิวเตอร์คลาสสิก Quil คือ GATE โดยที่ GATE คือประตูควอนตัม เพื่อนำไปใช้กับ qubit ที่จัดทำ index (0, 1, 2,...) pyQuil มีคุณสมบัติสำหรับสร้างรหัส Quil จากโปรแกรมที่เราเขียน

ฮาร์ดแวร์ควอนตัม (Quantum Hardware)

Rigetti มีกระบวนการเพื่อให้ผู้ขอเข้าใช้งานต้องขอผ่านเว็บไซต์ Rigetti และ ระบุชื่อเต็ม ที่อยู่อีเมล ชื่อองค์กร และคำอธิบายเหตุผลในการเข้าถึง QPU เสร็จแล้วตัวแทนบริษัทจะติดต่อไป ผ่านทางอีเมลเพื่อกำหนดเวลาในการให้สิทธิ์แก่ผู้ใช้ QPU โดยจะให้ช่วงเวลามาข้อดีคือไม่ต้องแย่งกันใช้มีเวลาสำหรับเข้าใช้งาน ข้อเสียคือไม่สามารถใช้งานได้ตลอดเวลาต้องรอรอบของตัวเอง

การจำลองควอนตัม (Simulated Quantum)

Simulator เครื่องเสมือนควอนตัม (QVM) เป็นโปรแกรมที่เขียนขึ้นเพื่อรันบน CPU แบบคลาสสิก ที่ป้อนรหัส Quil และจำลองวิวัฒนาการของคอมพิวเตอร์ควอนตัมจริง หากต้องการเชื่อมต่อกับ QVM หนึ่งต้องลงทะเบียนสำหรับคีย์ API ฟรีบน https://www.rigetti.com/forest โดยระบุชื่อและที่อยู่อีเมลจะถูกส่งด้วย key API และ ID ผู้ใช้ซึ่งต้องตั้งค่าโดย

```
pyquil ―config –setup
```

โดยใช้คำสั่งนี้จะหลังติดตั้ง pyQuil จะปรากฏขึ้นเพื่อให้ key ที่ส่งมายังอีเมล key API จะสามารถเข้าถึง QVM ได้สูงสุดที่ 30 qubits แต่ต้องทำเรื่องขอเพิ่มเติม โดย key ของผู้พัฒนา code ให้สามารถเข้าใช้งานได้ 26 qubits หากไม่ได้ทำเรื่องขอเพิ่ม

ในการอ้างอิง QVM มาใช้งานง่ายนิดเดียวคือการ import api ดัง

```
import pyquil.api from api
```

ตารางที่ 3.3 ตารางรายละเอียดภาพรวม Forest (pyQuil)

Institution	Rigetti
Version	Version 3.1.0
Open Source	True
OS	Mac, Windows, Linux
Requirements	Python 3, Anaconda
Classical Host Language	Python
Quantum Program Language	pyQuil
Quantum Language	Quil
Quantum Hardware	-
Simulator	20 qubits locally, 26 qubits with most API keys to QVM, 30+ w/ private access
Features	Generate Quil code, example algorithms in Grove, topology-specific compiler, noise capabilities in simulator, community Slack channel

Forest (pyQuil) ถูกสร้างจากหน่วยงาน Rigetti มี Version ล่าสุดคือ Version 3.1.0 และตัว Quantum Software Platform เป็นแบบ Open Source สามารถติดตั้งได้ทั้ง Mac, Windows และ Linux โดยมีความต้องการก่อนการติดตั้งคือ Python 3 และ Anaconda ในส่วนของ Classical Host Language จะใช้ Python ในการเขียน code อีกทั้งมี pyQuil เป็น Quantum Program Language โดยมี Quil เป็น Quantum Language ส่วน Quantum Hardware ที่เห็นได้ชัดจะมีหน่วยความจำที่ 8 qubits ที่สามารถขอเข้าใช่งานได้กรณีพิเศษเท่านั้น ในส่วนของแบบจำลองนั้น สามารถจำลอง local ที่ 20 qubits, API ที่ขอเข้าใช้งาน 26 qubits โดยสามารถขอเพิ่มเป็น 30 qubit ได้โดยต้องทำเรื่องขอ

Qiskit

Qiskit เป็นชุดซอฟต์แวร์ข้อมูลควอนตัมแบบ Open Source ด้วยภาษาควอนตัม OpenQASM การคำนวณอยู่ใน IBM Q Experience Qiskit มีอยู่ใน Python, JavaScript และ Swift

ความต้องการและการติดตั้ง (Requirements and Installation)

สามารถติดตั้งได้ใน OS ไม่ว่าจะเป็น macOS, Windows และ Linux จำเป็นต้องใช้ Python 3.7 ขึ้นไป และควรติดตั้ง Jupyter notebooks เพื่อใช้เขียน code และ run code ทดสอบสิ่งต่าง ๆ ส่วน Anaconda 3 เอาไว้แยก environment ต่างๆ วิธีที่ง่ายที่สุดในการติดตั้ง Qiskit คือการใช้ pip ตัวจัดการแพ็คเกจ Python โดยการพิมพ์คำสั่ง

pip install qiskit

ไวยากรณ์ (Syntax)

Qiskit เป็นแพ็คเกจใน Python สำหรับทำการคำนวณควอนตัมโดยจะมีการ import หลักๆเป็น Quantum cicuits และ Quantum registes ซึ่งเป็นหัวใจสำคัญ

ตัวอย่างใช้งาน Quantum cicuits

1. from qiskit import QuantumCircuit

2. qc = QuantumCircuit()

เป็นวงจรว่างที่ไม่มี qubits และ outputs

ตัวอย่างใช้งาน Quantum registes

1. from qiskit import QuantumRegister

2. qr = QuantumRegister(2,'a')

เป็นการสร้างตัวแปร qubit ที่มีคือว่า a 2ตัวขึ้นมาได้แก่ a0 a1

1. qc.add_register(qr)

2. qc.qregs

เป็นการลงทะเบียน qubit เข้าไปในวงจร qc

ภาษาควอนตัม (Quantum Language)

OpenQASM เป็นภาษาควอนตัมที่ใช้กับอุปกรณ์ตอมพิวเตอร์ควอนตัมจริงโดยจะคล้ายกับ Quil และ Forest โดยไวยากรณ์ทั่วไปของ OpenQASM จะเป็นในรูปแบบ Gate และ qubit โดยที่ Qiskit มีคุณสมบัติสำหรับสร้าง OpenQASM

ตัวอย่าง QASM

1. OPENQASM 2.0;

2. Include “qelib.inc”;

3. qreg q0 [1];

4. creg c0 [1];

5. h q0 [0];

6. measure q0[0] -> c0 [0];

ฮาร์ดแวร์ควอนตัม (Quantum Hardware)

มีอุปกรณ์ได้แก่ IBMQX2 (5 Qubits), IBMQX4 (5 Qubits), ibmq_20_tokyo (20 Qubits), QS1_1 (20 Qubits), IBMQX5 (16 Qubits)

การจำลองควอนตัม (Simulated Quantum)

IBM มีตัวจำลองวงจรควอนตัมที่ทำงานบนเครื่องและประมวลแบบ cloud computing

ตารางที่ 3.4 ตารางรายละเอียดภาพรวม Qiskit

Institution	IBM
Version	Version 0.36.1
Open Source	True
OS	Mac, Windows, Linux
Requirements	Python 3.7+, Jupyter Notebooks, Anaconda 3
Classical Host Language	Python, JavaScript, Swift
Quantum Program Language	Qiskit
Quantum Language	OpenQASM
Quantum Hardware	IBMQX2 (5 qubits), IBMQX4 (5 qubits), IBMQX5 (16 qubits), QS1 1 (20 qubits)
Simulator	～25 qubits locally, 30 through cloud
Features	Generate QASM code, topology specific compiler, community Slack channel, circuit drawer, Aqua library

ProjectQ

ProjectQ เป็นแพลตฟอร์มซอฟต์แวร์ควอนตัมโอเพ่นซอร์ส สำหรับการคำนวณควอนตัมที่มีคุณสมบัติการเชื่อมต่อกับแบ็กเอนด์ควอนตัมของ IBM ซึ่งเป็นควอนตัมที่มีประสิทธิภาพสูง โปรแกรมจำลองควอนตัมคอมพิวเตอร์และปลั๊กอินไลบรารีหลายตัว

ความต้องการและการติดตั้ง (Requirements and Installation)

การติดตั้ง Python 3.7+ ในการเริ่มใช้ ProjectQ เพียงเรียกใช้คำสั่ง

1. python -m pip install --user projectq

ไวยากรณ์ (Syntax)

เป้าหมายของ ProjectQ คือต้องมีไวยากรณ์ที่เข้าใจง่าย เพื่อให้สามารถเรียนรู้ได้ง่าย ดังนั้น ProjectQ จึงมีรูปแบบลีนที่ใกล้เคียงกับสัญกรณ์คณิตศาสตร์ที่ใช้ในฟิสิกส์

ตัวอย่าง applying an x-rotation by an angle *theta* to a qubit

1. Rx(theta) | qubit

ในขณะที่สัญกรณ์ที่สอดคล้องกันในฟิสิกส์จะเป็น $R_x(\theta)|\text{qubit}\rangle$

ภาษาควอนตัม (Quantum Language)

เนื่องจากไม่มีแบ็กเอนด์ควอนตัมเฉพาะ ProjectQ จึงไม่มีภาษาควอนตัมเฉพาะของตัวเอง หากมีการใช้ ProjectQ ร่วมกับแบ็กเอนด์ของ IBM รหัสในที่สุดก็ถูกแปลงเป็น OpenQASM ซึ่งเป็นภาษาแอสเซมบลีควอนตัมของ IBM

ฮาร์ดแวร์ควอนตัม (Quantum Hardware)

ไม่มีควอนตัมคอมพิวเตอร์เป็นของตัวเองแต่ใช้งานแบ็กเอนด์ร่วมกับ IBM

การจำลองควอนตัม (Simulated Quantum)

ClassicalSimulator สำหรับการจำลองโคลงอย่างมีประสิทธิภาพวงจรนั่นคือ วงจรที่ประกอบด้วยเกทจากนอร์มัลไลเซอร์ของกลุ่มเพาลี ซึ่งสามารถสร้างขึ้นจากHadmard, CNOT และประตูเฟส เครื่องจำลองนี้สามารถจัดการ qubits นับพันเพื่อตรวจสอบได้

ตารางที่ 3.5 ตารางรายละเอียดภาพรวม ProjectQ

Institution	ETH Zurich
Version	Version 0.7.3
Open Source	True
OS	Mac, Windows, Linux
Requirements	use pip version v6.1.0+ , Python 3.7+
Classical Host Language	Python
Quantum Program Language	ProjectQ
Quantum Language	-
Quantum Hardware	can connect to IBM backends
Simulator	28 qubits locally
Features	Draw circuits, connect to IBM backends, multiple library plug-ins

Quantum Developer Kit (Q#)

แพลตฟอร์มซอฟต์แวร์ที่นำเสนอในบทความนี้ อย่างไรก็ตาม QDK นำเสนอภาษา "ที่เน้นควอนตัม" ใหม่ เรียกว่า Q# ที่มีการบูรณาการที่แข็งแกร่งกับ Visual Studio และ Visual Studio Code และสามารถจำลองควอนตัมได้วงจรสูงถึง 30 qubits

ความต้องการและการติดตั้ง (Requirements and Installation)

สามารถติดตั้ง QDK ใน Visual Studio Code โดยใช้คำสั่ง

dotnet new –i “Microsoft.Quantum.ProjectTemplates::0.2 -- *”

ไวยากรณ์ (Syntax)

ไวยากรณ์ของ Q# ค่อนข้างแตกต่างจาก 3 ภาษาก่อนหน้า ใกล้เคียงกับ C# และละเอียดกว่า Python โดยจะแสดงตัวอย่าง ramdom bit generator circuit in Q#

```
1. Operation ramdom() : Int
2. {
```

```
3.  body
4. {
5.  Mutable measured = 0;
6. Using (qubit = Qubit [1])
7. {
8. Set (Zero, qubits [0]);
9. H(qubit [0]);
10. let res = M (qubit [0]);
11. If (res == One)
12. {
13. Set measured = 1;
14. }
15. Set (Zero, qubits [0]);
16. }
17. return measured;
18. }
19. }
```

ภาษาควอนตัม (Quantum Language)

QDK ไม่มีความสามารถในปัจจุบันในการเชื่อมต่อกับของจริงคอมพิวเตอร์ควอนตัม

ฮาร์ดแวร์ควอนตัม (Quantum Hardware)

QDK ไม่มีความสามารถในปัจจุบันในการเชื่อมต่อกับของจริงคอมพิวเตอร์ควอนตัม ดังนั้นจึงไม่มีภาษาคำสั่ง

การจำลองควอนตัม (Simulated Quantum)

ในเครื่องคอมพิวเตอร์ของผู้ใช้ QDK รวมถึงเครื่องจำลองควอนตัมที่สามารถเรียกใช้วงจรของมากถึง 30 qubits QDK ตัวจำลองถูกเขียนขึ้นโดยนักพัฒนาของ ProjectQ ดังนั้น คาดว่าประสิทธิภาพจะใกล้เคียงกับ ProjectQ

ตารางที่ 3.6 ตารางรายละเอียดภาพรวม Quantum Developer Kit (Q#)

Institution	Microsoft
Version	Version 0.24.208024
Open Source	True
OS	Mac, Windows, Linux
Requirements	Visual Studio Code
Classical Host Language	C#
Quantum Program Language	Q#
Quantum Language	-
Quantum Hardware	-
Simulator	30 qubits locally, 40 through Azure cloud
Features	Built-in algorithms, example algorithms

3.5.3 เมื่อติดตั้งทั้ง 4 Platforms ได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) นำแต่ละ framework ทดสอบ function และ Quantum Algorithm ต่างๆว่าสามารถใช้งานอะไรได้บ้าง

Forest (pyQuil)

ตารางที่ 3.7 ตารางทดสอบ Function และ Quantum Algorithm ของ pyQuil

Algorithm	pyQuil
Random Bit Generator	✓
Teleportation	✓
Swap Test	✓
Deutsch-Jozsa	✓
Grover's Algorithm	✓
Quantum Fourier Transform	✓
Shor's Algorithm	
Bernstein Vazirani	✓

Phase Estimation	✓
Optimization/ QAOA	✓
Simon's Algorithm	✓
Variational Quantum Eigensolver	✓
Amplitude Amplification	✓
Quantum Walks	
Ising Solver	✓
Quantum Gradient Descent	✓
Five Qubit Code	
Repetition Code	
Steane Code	
Draper Adder	
Beauregard Adder	
Arithmetic	
Fermion Transforms	✓
Trotter Simulation	
Electronic Structure (FCI, MP2, HF, etc.)	
Process Tomography	✓
Vaidman Detection Test	

Qiskit

ตารางที่ 3.8 ตารางทดสอบ Function และ Quantum Algorithm ของ pyQuil

Algorithm	Qiskit
Random Bit Generator	✓
Teleportation	✓
Swap Test	
Deutsch-Jozsa	✓
Grover's Algorithm	✓
Quantum Fourier Transform	✓
Shor's Algorithm	✓
Bernstein Vazirani	✓

Phase Estimation	✓
Optimization/ QAOA	✓
Simon's Algorithm	✓
Variational Quantum Eigensolver	✓
Amplitude Amplification	
Quantum Walks	✓
Ising Solver	
Quantum Gradient Descent	
Five Qubit Code	
Repetition Code	✓
Steane Code	
Draper Adder	
Beauregard Adder	
Arithmetic	
Fermion Transforms	✓
Trotter Simulation	
Electronic Structure (FCI, MP2, HF, etc.)	
Process Tomography	✓
Vaidman Detection Test	✓

ProjectQ

ตารางที่ 3.9 ตารางทดสอบ Function และ Quantum Algorithm ของ ProjectQ

Algorithm	ProjectQ
Random Bit Generator	✓
Teleportation	✓
Swap Test	
Deutsch-Jozsa	
Grover's Algorithm	✓
Quantum Fourier Transform	
Shor's Algorithm	✓
Bernstein Vazirani	

Phase Estimation	
Optimization/ QAOA	
Simon's Algorithm	✓
Variational Quantum Eigensolver	
Amplitude Amplification	
Quantum Walks	
Ising Solver	
Quantum Gradient Descent	
Five Qubit Code	
Repetition Code	
Steane Code	
Draper Adder	✓
Beauregard Adder	✓
Arithmetic	
Fermion Transforms	
Trotter Simulation	
Electronic Structure (FCI, MP2, HF, etc.)	
Process Tomography	
Vaidman Detection Test	

Quantum Developer Kit (Q#)

ตารางที่ 3.10 ตารางทดสอบ Function และ Quantum Algorithm ของ Q#

Algorithm	Q#
Random Bit Generator	✓
Teleportation	✓
Swap Test	
Deutsch-Jozsa	✓
Grover's Algorithm	
Quantum Fourier Transform	
Shor's Algorithm	
Bernstein Vazirani	✓

Phase Estimation	
Optimization/ QAOA	
Simon's Algorithm	
Variational Quantum Eigensolver	
Amplitude Amplification	
Quantum Walks	
Ising Solver	✓
Quantum Gradient Descent	
Five Qubit Code	
Repetition Code	
Steane Code	
Draper Adder	
Beauregard Adder	
Arithmetic	
Fermion Transforms	
Trotter Simulation	
Electronic Structure (FCI, MP2, HF, etc.)	
Process Tomography	
Vaidman Detection Test	

3.5.4 แสดงภาพรวมทั้งหมดของ Quantum Software Platform ทั้ง 4 Platforms ได้แก่ Forest (pyQuil),Qiskit, ProjectQ และ Quantum Developer Kit (Q#) เพื่อที่จะได้เปรียบเทียบความสามารถและนำ Platform ที่ 2 Platforms ที่ดีที่สุดในการทดลองไปวัดประสิทธิภาพการทำงานแบบจำลองต่อไป

ตารางที่ 3.11 ตารางความสามารถและความสะดวกในการใช้งานทั้ง 4 แพลตฟอร์ม

องค์ประกอบ	pyQuil	Qiskit	ProjectQ	Q#
ซอฟต์แวร์ฟรี	✓	✓	✓	✓
มีควอนตัมคอมพิวเตอร์จริง	✓	✓		
สามารถเข้าใช้งานควอนตัมคอมพิวเตอร์จริง		✓	✓	
สามารถเข้าใช้งานบนคอมพิวติ้ง	✓	✓		✓
การใช้งานคลาวด์คอมพิวติ้งฟรี	✓	✓		

มีเอกสารที่ใช้ในการศึกษาที่ละเอียด		✓	✓	✓
ซอฟต์แวร์มีข้อผิดพลาดน้อย		✓	✓	
สามารถใช้อัลกอริทึมที่สำคัญได้ครบ		✓	✓	
ภาษาที่ใช้ในการพัฒนาเข้าใจง่าย	✓	✓	✓	
การแสดงผลลัพธ์ใช้งานง่าย	✓	✓	✓	

3.5.5 นำ 2 Platforms ได้แก่ ที่ดีที่สุดในการเปรียบเทียบภาพรวมมาวัดประสิทธิภาพการทำงานแบบจำลอง (Simulator Performance) โดยการใช้ Shor's algorithm ในการวัดผล

ตัวอย่าง code Qiskit

```
import matplotlib.pyplot as plt
import numpy as np
from qiskit import QuantumCircuit, Aer, transpile, assemble
from qiskit.visualization import plot_histogram
from math import gcd
from numpy.random import randint
import pandas as pd
from fractions import Fraction
import time
def c_amod15(a, power):
    """Controlled multiplication by a mod 15"""
if a not in [2,4,7,8,11,13]:
    raise ValueError("'a' must be 2,4,7,8,11 or 13")
U = QuantumCircuit(4)
for iteration in range(power):
    if a in [2,13]:
        U.swap(0,1)
        U.swap(1,2)
        U.swap(2,3)
if a in [7,8]:
        U.swap(2,3)
        U.swap(1,2)
```

```
            U.swap(0,1)
        if a in [4, 11]:
            U.swap(1,3)
            U.swap(0,2)
        if a in [7,11,13]:
            for q in range(4):
                U.x(q)
    U = U.to_gate()
    U.name = "%i^%i mod 15" % (a, power)
    c_U = U.control()
    return c_U
def qft_dagger(n):
    """n-qubit QFTdagger the first n qubits in circ"""
    qc = QuantumCircuit(n)
    # Don't forget the Swaps!
    for qubit in range(n//2):
        qc.swap(qubit, n-qubit-1)
    for j in range(n):
        for m in range(j):
            qc.cp(-np.pi/float(2**(j-m)), m, j)
        qc.h(j)
    qc.name = "QFT†"
    return qc


def qpe_amod15(a):
    n_count = 8
    qc = QuantumCircuit(4+n_count, n_count)
    for q in range(n_count):
        qc.h(q)     # Initialize counting qubits in state |+>
    qc.x(3+n_count) # And auxiliary register in state |1>
```

```
    for q in range(n_count): # Do controlled-U operations
        qc.append(c_amod15(a, 2**q),
                  [q] + [i+n_count for i in range(4)])
    qc.append(qft_dagger(n_count), range(n_count)) # Do inverse-QFT
    qc.measure(range(n_count), range(n_count))
    # Simulate Results
    aer_sim = Aer.get_backend('aer_simulator')
    # Setting memory=True below allows us to see a list of each sequential
reading
    t_qc = transpile(qc, aer_sim)
    qobj = assemble(t_qc, shots=1)
    result = aer_sim.run(qobj, memory=True).result()
    readings = result.get_memory()
    print("Register Reading: " + readings[0])
    phase = int(readings[0],2)/(2**n_count)
    print("Corresponding Phase: %f" % phase)
    return phase


a = 7
N = 35


for x in range(10):
    print("\nthe times %i:" %(x +1))
    factor_found = False
    t0 = time.time()
   # tic = time.perf_counter()
    attempt = 0
    while not factor_found:
        attempt += 1
        print("\nAttempt %i:" % attempt)
```

```
        phase = qpe_amod15(a) # Phase = s/r
        frac = Fraction(phase).limit_denominator(N) # Denominator should
(hopefully!) tell us r
        r = frac.denominator

        print("Result: phase = %i" % phase)
        print("Result: frac = %i" % frac)
        print("Result: r = %i" % r)
        if phase != 0:
          # Guesses for factors are gcd(x^{r/2} ±1 , 15)
          print("Result: gcd a1 = %i" % (a**(r//2)-1))
          print("Result: gcd a2 = %i" % (a**(r//2)+1))
          guesses = [gcd(a**(r//2)-1, N), gcd(a**(r//2)+1, N)]
          print("Guessed Factors: %i and %i" % (guesses[0], guesses[1]))
          t1 = time.time()
          total = t1-t0
          print(total)
          factor_found = True
          for guess in guesses:
            if guess not in [1,N] and (N % guess) == 0: # Check to see if guess is a
factor
              print("*** Non-trivial factor found: %i ***" % guess)
```

ตัวอย่าง code ProjectQ

```
import math
import random
import sys
from fractions import Fraction
import time
```

```
try:
    from math import gcd
except ImportError:
    from fractions import gcd

from builtins import input

import projectq.libs.math
import projectq.setups.decompositions
from projectq.backends import ResourceCounter, Simulator
from projectq.cengines import (
    AutoReplacer,
    DecompositionRuleSet,
    InstructionFilter,
    LocalOptimizer,
    MainEngine,
    TagRemover,
)
from projectq.libs.math import AddConstant, AddConstantModN,
MultiplyByConstantModN
from projectq.meta import Control
from projectq.ops import QFT, All, BasicMathGate, H, Measure, R, Swap, X,
get_inverse


def run_shor(eng, N, a, verbose=False):
    """
    Run the quantum subroutine of Shor's algorithm for factoring.
    Args:
        eng (MainEngine): Main compiler engine to use.
```

```
    N (int): Number to factor.
    a (int): Relative prime to use as a base for a^x mod N.
    verbose (bool): If True, display intermediate measurement results.
Returns:
    r (float): Potential period of a.
"""
n = int(math.ceil(math.log(N, 2)))

x = eng.allocate_qureg(n)

X | x[0]

measurements = [0] * (2 * n)  # will hold the 2n measurement results

ctrl_qubit = eng.allocate_qubit()

for k in range(2 * n):
    current_a = pow(a, 1 << (2 * n - 1 - k), N)
    # one iteration of 1-qubit QPE
    H | ctrl_qubit
    with Control(eng, ctrl_qubit):
        MultiplyByConstantModN(current_a, N) | x

    # perform inverse QFT --> Rotations conditioned on previous outcomes
    for i in range(k):
        if measurements[i]:
            R(-math.pi / (1 << (k - i))) | ctrl_qubit
    H | ctrl_qubit

    # and measure
```

```
            Measure | ctrl_qubit
            eng.flush()
            measurements[k] = int(ctrl_qubit)
            if measurements[k]:
                X | ctrl_qubit

            if verbose:
                print("\033[95m{}\033[0m".format(measurements[k]), end="")
                sys.stdout.flush()

        All(Measure) | x
        # turn the measured values into a number in [0,1)
        y = sum([(measurements[2 * n - 1 - i] * 1.0 / (1 << (i + 1))) for i in range(2 *
n)])

        # continued fraction expansion to get denominator (the period?)
        r = Fraction(y).limit_denominator(N - 1).denominator

        # return the (potential) period
        return r

# Filter function, which defines the gate set for the first optimization
# (don't decompose QFTs and iQFTs to make cancellation easier)
def high_level_gates(eng, cmd):
    """Filter high-level gates."""
    g = cmd.gate
    if g == QFT or get_inverse(g) == QFT or g == Swap:
        return True
    if isinstance(g, BasicMathGate):
        return False
```

```
        if isinstance(g, AddConstant):
            return True
        elif isinstance(g, AddConstantModN):
            return True
        return False
    return eng.next_engine.is_available(cmd)

if __name__ == "__main__":
    # build compilation engine list
    resource_counter = ResourceCounter()
    rule_set = DecompositionRuleSet( modules= [ projectq. libs. math,
projectq.setups.decompositions])
    compilerengines = [
        AutoReplacer(rule_set),
        InstructionFilter(high_level_gates),
        TagRemover(),
        LocalOptimizer(3),
        AutoReplacer(rule_set),
        TagRemover(),
        LocalOptimizer(3),
        resource_counter,
    ]

    # make the compiler and run the circuit on the simulator backend
    eng = MainEngine(Simulator(), compilerengines)

    # print welcome message and ask the user for the number to factor
    print(
        "\n\t\033[37mprojectq\033[0m\n\t--------\n\tImplementation of Shor" "\'s
algorithm.",
```

```
        end="",
    )
    N = int(input('\n\tNumber to factor: '))
    print("\n\tFactoring N = {}: \033[0m".format(N), end="")

    # choose a base at random:
#     a = int(random.random() * N)
    a = 7

    for x in range(10):
        print("\nthe times %i:" %(x +1))
        t0 = time.time()
#         if not gcd(a, N) == 1:
#             print("\n\n\t\033[92mOoops, we were lucky: Chose non relative prime"
" by accident :)")
#             print("\tFactor: {}\033[0m".format(gcd(a, N)))
#         else:
            # run the quantum subroutine
        r = run_shor(eng, N, a, True)

        print("\nthe r %i:" %(r))

        # try to determine the factors
        if r % 2 != 0:
            r *= 2
        apowrhalf = pow(a, r >> 1, N)
        f1 = gcd(apowrhalf + 1, N)
        f2 = gcd(apowrhalf - 1, N)

        print("\nthe f1 %i:" %(f1))
```

```
print("\nthe f2 %i:" %(f2))

if (not f1 * f2 == N) and f1 * f2 > 1 and int(1.0 * N / (f1 * f2)) * f1 * f2 ==
N:
    f1, f2 = f1 * f2, int(N / (f1 * f2))
if f1 * f2 == N and f1 > 1 and f2 > 1:
    print("\n\n\t\033[92mFactors found :-) : {} * {} = {}\033[0m".format(f1,
f2, N))
else:
    print("\n\n\t\033[91mBad luck: Found {} and {}\033[0m".format(f1, f2))
t1 = time.time()
total = t1-t0
print(total)
```

3.6 บันทึกผลจากการทดลอง

ในระหว่างการทดสอบจะบันทึกข้อมูลการทดลองตามองค์ประกอบดังตารางนี้

ตารางที่ 3.12 ตารางบันทึกผลการทดลอง

องค์ประกอบ	ค่าที่ทำการบันทึก
เก็บรวบรวมข้อมูล Quantum Software Platform	ความเป็นมาของ Platforms
	Qubit Local Simulator
	Register for API Key
	Qubit in Cloud
	สิทธ์การใช้บน Cloud
การเริ่มต้นการใช้งาน Quantum Software Platform	Installation
	Language Syntax
	Library Support
	Simulated Quantum
	Quantum assembly Language
	Quantum Compiler

Function และ Quantum Algorithm ที่ใช้งานได้	Random Bit Generator
	Teleportation
	Swap Test
	Deutsch-Jozsa
	Grover's Algorithm
	Quantum Fourier Transform
	Shor's Algorithm
	Bernstein Vazirani
	Phase Estimation
	Optimization/ QAOA
	Simon's Algorithm
	Variational Quantum Eigensolver
	Amplitude Amplification
	Quantum Walks
	Ising Solver
	Quantum Gradient Descent
	Five Qubit Code
	Repetition Code
	Steane Code
	Draper Adder
	Beauregard Adder
	Arithmetic
	Fermion Transforms
	Trotter Simulation
	Electronic Structure (FCI, MP2, HF, etc.)
	Process Tomography
	Vaidman Detection Test
เปรียบเทียบภาพรวม Quantum Software Platform 2 Platform ที่ดีที่สุด	วัดประสิทธิภาพการทำงานแบบจำลอง (Simulator Performance) โดยการใช้ Grover's algorithm และบันทึกเวลา

บทที่ 4

ผลการทดลอง

ในบทนี้กล่าวถึงผลการทดสอบการเปรียบเทียบภาพภาพรวมและเปรียบเทียบซอฟต์แวร์ควอนตัมแพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) โดยนำ 2 แพลตฟอร์มที่ดีที่สุดมาวัดประสิทธิภาพการทำงานแบบจำลองโดยใช้ Shor's algorithm โดยมีผลการทดสอบดังนี้

4.1 ผลการทดสอบการเชื่อมต่อ

4.2 ผลการทดสอบการติดตั้งและภาษาที่รองรับ

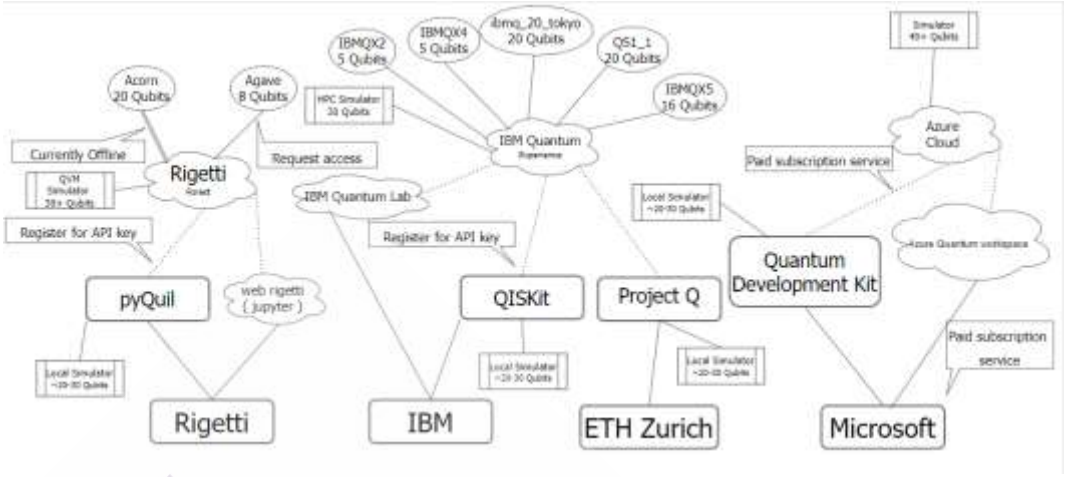
4.3 ผลการทดสอบฟังก์ชั่นและควอนตัมอัลกอริทึมที่ใช้งานได้

4.4 ผลการทดสอบเปรียบเทียบเพื่อคัดเลือก 2 แพลตฟอร์ม

4.5 ผลการทดสอบการเปรียบเทียบ 2 แพลตฟอร์มแบบวัดประสิทธิภาพการทำงานแบบจำลองโดยการใช้ Shor's algorithm

4.1 ผลการทดสอบการเชื่อมต่อ

ในการทดลองเป็นการทดลองเพื่อทดสอบว่าแพลตฟอร์มที่ 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) มีการเชื่อมต่อไปยังคอมพิวติ้งและควอนตัมคอมพิวเตอร์จริงแบบใด และตรวจสอบว่าแพลตฟอร์มใดไม่มีควอนตัมคอมพิวเตอร์หรือเสียค่าใช้จ่ายอะไรในการใช้งานสามารถสรุปภาพรวมได้ดังภาพนี้



ภาพที่ 4.1 ภาพรวมทดสอบระบบการต่อควอนตัมคอมพิวเตอร์ของ 4 แพลตฟอร์ม

4.2 ผลการทดสอบการติดตั้ง ไวยากรณ์ และภาษาที่รองรับ

ในการทดลองเป็นการทดลองเพื่อทดสอบว่าแพลตฟอร์มที่ 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) โดยจะเก็บข้อมูล Software Platforms (requirements and installation), ภาษาและรูปแบบการเขียนโปรแกรม (language syntax), สิ่งที่สนันสนุน (Library Support), การจำลองควอนตัม (Simulated Quantum), ภาษาแอสเซมบลีควอนตัม (Quantum Assembly Language) และควอนตัมคอมไพเลอร์ (Quantum Compiler) เพื่อเป็นข้อมูลเปรียบเทียบการเลือกใช้งาน สามารถสรุปภาพรวมได้ดังตารางนี้

ตารางที่ 4.1 ตารางรวมผลทดลองรายละเอียดภาพรวม 4 แพลตฟอร์ม

รายละเอียด	ซอฟต์แวร์ควอนตัมแพลตฟอร์ม			
Institution	Rigetti	IBM	ETH Zurich	Microsoft
Version	Version 3.1.0	Version 0.36.1	Version 0.7.3	Version 0.24.208024
Open Source	True	True	True	True
OS	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux
Requirements	Python 3,	Python 3.7+,	use pip	Visual Studio

	Anaconda	Jupyter Notebooks, Anaconda 3	version v6.1.0+ , Python 3.7+	Code
Classical Host Language	Python	Python, JavaScript, Swift	Python	C#
Quantum Program Language	pyQuil	Qiskit	ProjectQ	Q#
Quantum Language	Quil	OpenQASM	-	-
Quantum Hardware		IBMQX4 (5 qubits)	can connect to IBM backends	-
Simulator	20 qubits locally, 26 qubits with most API keys to QVM, 30+ w/ private access	25 qubits locally, 30 through cloud	28 qubits locally	30 qubits locally, 40 through Azure cloud
Features	Generate Quil code, example algorithms in Grove, topology-specific compiler, noise capabilities in simulator, community Slack channel	Generate QASM code, topology specific compiler, community Slack channel, circuit drawer, Aqua library	Draw circuits, connect to IBM backends, multiple library plug-ins	Built-in algorithms, example algorithms

4.3 ผลการทดสอบฟังก์ชั่นและควอนตัมอัลกอริทึมที่ใช้งานได้

ในการทดลองนี้เป็นการทดลองเพื่อทดสอบว่าแพลตฟอร์มที่ 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) ใช้งานฟังก์ชั่นและควอนตัมอัลกอริทึมอะไรได้บางโดยมีแต่ตัวฟังก์ชั่นหลักที่ไม่ได้ลงส่วนเสริมใดๆเข้าไป สามารถสรุปได้ดังตารางนี้

ตารางที่ 4.2 ตางรางรวมผลทดสอบฟังก์ชั่นและควอนตัมอัลกอริทึมทั้ง 4 แพลตฟอร์ม

Algorithm	pyQuil	Qiskit	ProjectQ	Q#
Random Bit Generator	✓	✓	✓	✓
Teleportation	✓	✓	✓	✓
Swap Test	✓			
Deutsch-Jozsa	✓	✓		✓
Grover's Algorithm	✓	✓	✓	
Quantum Fourier Transform	✓	✓		
Shor's Algorithm		✓	✓	
Bernstein Vazirani	✓	✓		✓
Phase Estimation	✓	✓		
Optimization/ QAOA	✓	✓		
Simon's Algorithm	✓	✓	✓	
Variational Quantum Eigensolver	✓	✓		
Amplitude Amplification	✓			
Quantum Walks		✓		
Ising Solver	✓			✓
Quantum Gradient Descent	✓			
Five Qubit Code				
Repetition Code		✓		
Steane Code				
Draper Adder			✓	
Beauregard Adder			✓	
Arithmetic				
Fermion Transforms	✓	✓		
Trotter Simulation				
Electronic Structure (FCI, MP2, HF, etc.)				
Process Tomography	✓	✓		
Vaidman Detection Test		✓		

4.4 ผลการทดสอบเปรียบเทียบเพื่อคัดเลือก 2 แพลตฟอร์ม

ในการทดลองนี้เป็นการเปรียบเทียบภาพรวมการใช้งานของแพลตฟอร์มทั้ง 4 แพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) ว่าสามารถตอบโจทย์กับการเลือกใช้งานของผู้ใช้งาน หรือผู้พัฒนาได้มากน้อยเพียงใดโดยจะเห็นได้ว่า 2 แพลตฟอร์มที่มีภาพรวมการใช้งานได้ดีได้ Qiskit และ ProjectQ สามารถสรุปได้ดังตารางต่อไปนี้

ตารางที่ 4.3 ผลการทดสอบเปรียบเทียบเพื่อคัดเลือก 2 แพลตฟอร์ม

องค์ประกอบ	pyQuil	Qiskit	ProjectQ	Q#
ซอฟต์แวร์ฟรี	✓	✓	✓	✓
มีควอนตัมคอมพิวเตอร์จริง	✓	✓		
สามารถเข้าใช้งานควอนตัมคอมพิวเตอร์จริง		✓	✓	
สามารถเข้าใช้งานบนคอมพิวติ้ง	✓	✓		✓
การใช้งานคลาวด์คอมพิวติ้งฟรี	✓	✓		
มีเอกสารที่ใช้ในการศึกษาที่ละเอียด		✓	✓	✓
ซอฟต์แวร์มีข้อผิดพลาดน้อย		✓	✓	
สามารถใช้อัลกอริทึมที่สำคัญได้ครบ		✓	✓	
ภาษาที่ใช้ในการพัฒนาเข้าใจง่าย	✓	✓	✓	
การแสดงผลลัพธ์ใช้งานง่าย	✓	✓	✓	

4.5 ผลการทดสอบการเปรียบเทียบ 2 แพลตฟอร์มแบบวัดประสิทธิภาพการทำงานแบบจำลองโดยใช้ Shor's algorithm

ตารางที่ 4.4 เปรียบเทียบด้วย Shor's algorithm ที่ N = 9 และ A = 7

รอบที่	Qiskit	ProjectQ
1	2.40954447 s	2.01663184 s
2	5.54915786 s	1.98466611 s
3	3.70509219 s	2.03954601 s
4	1.78720474 s	2.05552125 s
5	1.81219649 s	2.04152441 s

6	3.53849459 s	2.17916989 s
7	1.83213568 s	2.24403548 s
8	1.76524401 s	2.18113828 s
9	1.8849678 s	2.13827515 s
10	1.75230503 s	2.08143282 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 9 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.752305031 วินาทีและ ProjectQ คือ 2.081432819 วินาทีและได้คำตอบที่เหมือนกันคือ 3*3

ตารางที่ 4.5 เปรียบเทียบด้วย Shor's algorithm ที่ N = 15 และ A = 7

รอบที่	Qiskit	ProjectQ
1	3.48968816 s	1.86700034 s
2	6.33805609 s	1.83809328 s
3	1.58475614 s	1.82215524 s
4	1.63463664 s	1.82710242 s
5	1.56680894 s	1.87995052 s
6	1.56082726 s	1.89695406 s
7	1.92780399 s	1.78121448 s
8	1.63266611 s	1.88497162 s
9	1.71740556 s	1.88197255 s
10	1.52296567 s	1.93479419 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 15 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.52296567 วินาทีและ ProjectQ คือ 1.934794188 วินาทีและได้คำตอบที่เหมือนกันคือ 3*5

ตารางที่ 4.6 เปรียบเทียบด้วย Shor's algorithm ที่ N = 21 และ A = 7

รอบที่	Qiskit	ProjectQ
1	1.63365674 s	3.58640289 s
2	4.5603838 s	3.57647491 s
3	1.50495028 s	3.51461673 s
4	3.34109545 s	3.5863812 s
5	1.71940541 s	3.67417979 s
6	1.48303795 s	3.64924908 s
7	3.45604491 s	3.52755427 s
8	1.46907043 s	3.92749572 s
9	1.52691674 s	3.5953896 s
10	1.44914031 s	3.50773811 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 21 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.44914031 วินาทีและ ProjectQ คือ 3.507738113 วินาทีและได้คำตอบที่ไม่เหมือนกัน Qiskit ได้ 3*7 และ ProjectQ ได้ 1*21

ตารางที่ 4.7 เปรียบเทียบด้วย Shor's algorithm ที่ N = 25 และ A = 7

รอบที่	Qiskit	ProjectQ
1	1.74334645 s	3.94545245 s
2	1.58273458 s	3.57205176 s
3	3.2812562 s	3.54950285 s
4	2.80220062 s	3.57150936 s
5	1.63801026 s	3.55678606 s
6	3.2552948 s	3.58054113 s
7	1.67053437 s	3.72503757 s
8	1.75630116 s	4.14591742 s
9	3.36400342 s	3.73101711 s
10	1.81813812 s	3.83870769 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 25 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.818138123 วินาทีและ ProjectQ คือ 3.838707685 วินาทีและได้คำตอบที่เหมือนกันคือ 1*25

ตารางที่ 4.8 เปรียบเทียบด้วย Shor's algorithm ที่ N = 33 และ A = 7

รอบที่	Qiskit	ProjectQ
1	1.52555609 s	7.31152129 s
2	1.47408199 s	7.10331345 s
3	1.52788258 s	7.16430426 s
4	4.90496874 s	7.05113697 s
5	3.44486094 s	6.91592693 s
6	1.49899149 s	7.44110012 s
7	3.21739578 s	7.05923867 s
8	1.62964225 s	7.28030992 s
9	1.49799442 s	6.98232651 s
10	1.64859009 s	7.44700527 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 33 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.648590088 วินาทีและ ProjectQ คือ 7.447005272 วินาทีและได้คำตอบที่เหมือนกันคือ 3*11

ตารางที่ 4.9 เปรียบเทียบด้วย Shor's algorithm ที่ N = 35 และ A = 7

รอบที่	Qiskit	ProjectQ
1	3.50879526 s	7.74524069 s
2	1.80925226 s	8.18155551 s
3	1.90680695 s	7.90590477 s
4	1.87920046 s	7.47014952 s
5	7.29190016 s	7.39980388 s
6	1.87835717 s	7.4753828 s
7	1.71040463 s	7.67493176 s

8	1.75269651 s	7.79955482 s
9	3.49825406 s	7.93472242 s
10	1.67679453 s	8.01993132 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 35 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.676794529 วินาทีและ ProjectQ คือ 8.019931316 วินาทีและได้คำตอบที่ไม่เหมือนกัน Qiskit ได้ 5*7 และ ProjectQ ได้ 1*35

ตารางที่ 4.10 เปรียบเทียบด้วย Shor's algorithm ที่ N = 39 และ A = 7

รอบที่	Qiskit	ProjectQ
1	1.8260808 s	7.76069999 s
2	3.70181537 s	8.16374111 s
3	1.87399006 s	7.94822216 s
4	1.77628732 s	8.12110138 s
5	1.93478727 s	8.40496469 s
6	5.26448178 s	7.83407044 s
7	1.76703882 s	8.48078704 s
8	1.63510537 s	7.80121803 s
9	3.6841805 s	8.23250794 s
10	1.81841135 s	8.5355165 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 39 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.81841135 วินาทีและ ProjectQ คือ 8.5355165 วินาทีและได้คำตอบที่เหมือนกันคือ 3*13

ตารางที่ 4.11 เปรียบเทียบด้วย Shor's algorithm ที่ N = 55 และ A = 7

รอบที่	Qiskit	ProjectQ
1	3.37991738 s	7.69125152 s
2	1.71943283 s	8.30282068 s
3	1.76198125 s	7.61241388 s
4	3.4963541 s	7.62816405 s
5	3.64624953 s	7.63380146 s
6	1.68155718 s	7.76736641 s
7	1.82079053 s	7.82383132 s
8	1.73804426 s	8.05775285 s
9	5.37947154 s	8.38981724 s
10	2.05893493 s	8.08801746 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 55 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 2.058934927 และ ProjectQ คือ 8.088017464 และได้คำตอบที่เหมือนกันคือ 5*11

ตารางที่ 4.12 เปรียบเทียบด้วย Shor's algorithm ที่ N = 65 และ A = 7

รอบที่	Qiskit	ProjectQ
1	1.67911005 s	16.6661744 s
2	1.70733285 s	16.0571299 s
3	3.86532354 s	16.1774898 s
4	1.78388715 s	16.6335371 s
5	1.80377388 s	16.601738 s
6	1.78375816 s	16.166909 s
7	1.828125 s	15.3442585 s
8	1.83175635 s	16.0300951 s
9	1.90489316 s	15.4067061 s
10	1.6948514 s	15.4971001 s

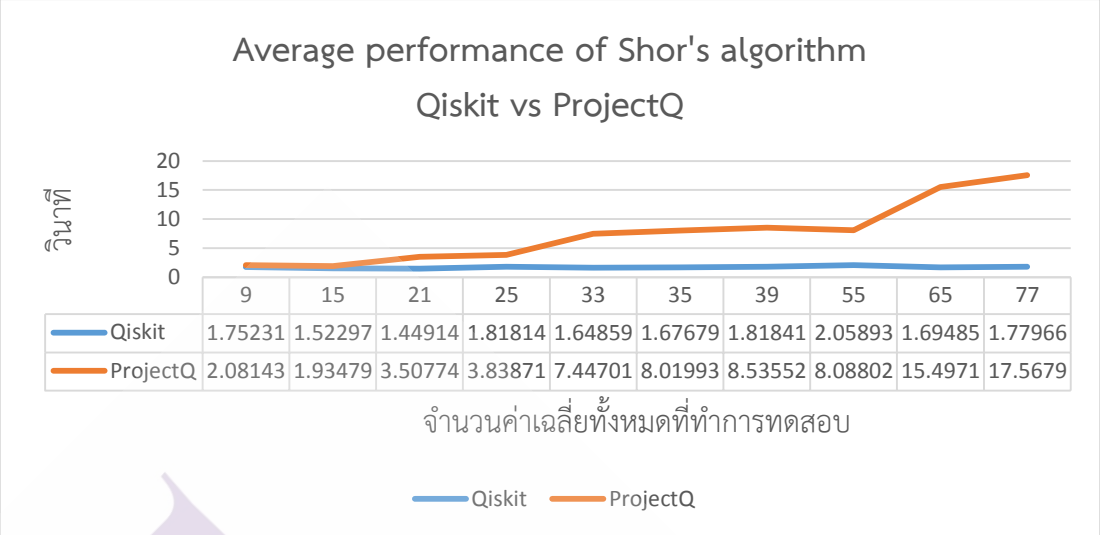
ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 65 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.694851398 วินาทีและ ProjectQ คือ 15.49710011 วินาทีและได้คำตอบที่ไม่เหมือนกัน Qiskit ได้ 5*13 และ ProjectQ ได้ 1*65

ตารางที่ 4.13 เปรียบเทียบด้วย Shor's algorithm ที่ N = 77 และ A = 7

รอบที่	Qiskit	ProjectQ
1	4.01446581 s	16.1296835 s
2	1.88286805 s	15.8890388 s
3	1.98422956 s	16.3092031 s
4	1.75864244 s	17.6504595 s
5	3.40062976 s	17.5179324 s
6	2.20947933 s	17.2953129 s
7	1.76713872 s	17.8826678 s
8	9.52694917 s	18.5774808 s
9	1.64702797 s	18.0826783 s
10	1.77965546 s	17.5678787 s

ทดลองโดยการวัดประสิทธิภาพการทำงานแบบจำลองจาก Shor's algorithm ที่ N = 77 และ A = 7 โดยที่ทำงานจำนวน 10 รอบทั้ง 2 Platforms โดยที่ค่าเฉลี่ยของ Qiskit คือ 1.779655457 วินาทีและ ProjectQ คือ 17.56787872 วินาทีและได้คำตอบที่ไม่เหมือนกัน Qiskit ได้ 7*11 และ ProjectQ ได้ 1*77

ในการทดลองนี้เป็นการเปรียบเทียบประสิทธิภาพการทำงานแบบจำลองทั้ง 2 แพลตฟอร์มได้แก่ Qiskit และ ProjectQ โดยใช้ Shor's algorithm โดยการไล่ระดับ N จาก 9, 15, 21, 25, 33, 35, 39, 55, 65 และ 77 ตามลำดับและ a ค่าอยู่ที่ 7 ตลอดเวลาจะได้ค่าเฉลี่ยของ Qiskit ค่า N ตามลำดับที่กล่าวมาโดยค่าเฉลี่ยจะได้อยู่ที่ 1.721978688 วินาทีและ ProjectQ จะได้อยู่ที่ 7.65181222 วินาที



ภาพที่ 4.2 รูปภาพค่าเฉลี่ยการทดสอบเปรียบเทียบประสิทธิภาพการทำงานแบบจำลอง Qiskit และ ProjectQ

บทที่ 5
บทสรุปและคำแนะนำ

เนื้อหาบทนี้ได้กล่าวสรุปถึงผลการดำเนินการทดสอบภาพรวมและเปรียบเทียบซอฟต์แวร์ควอนตัมแพลตฟอร์มได้แก่ Forest (pyQuil), Qiskit, ProjectQ และ Quantum Developer Kit (Q#) โดยนำ 2 แพลตฟอร์มที่ดีที่สุดมาวัดประสิทธิภาพการทำงานแบบจำลองโดยใช้ Shor's algorithm โดยอ้างอิงตามวัตถุประสงค์ที่ระบุไว้ โดยแบ่งการนำเสนอออกเป็น 4 ส่วนซึ่งประกอบด้วย การสรุปผลการวิจัย ข้อจำกัดของงานวิจัย และส่วนสุดท้ายกล่าวถึงข้อเสนอแนะ และแนวทางการพัฒนาในอนาคต โดยมีรายละเอียดดังต่อไปนี้

5.1 สรุปผลการวิจัย

ผลการทดลองการเชื่อมต่อซอฟต์แวร์ควอนตัมแพลตฟอร์มสามารถเชื่อมต่อได้ทุกแพลตฟอร์มแต่ Quantum Developer Kit (Q#) และ Forest (pyQuil) ไม่สามารถเชื่อมต่อควอนตัมคอมพิวเตอร์รจริงเพราะตัว Quantum Developer Kit (Q#) ไม่ได้มีควอนตัมคอมพิวเตอร์จริงมีแต่แบบจำลองให้ใช้งานเท่านั้น ส่วนในด้าน Forest (pyQuil) จะใช้ได้เฉพาะ QVM หากต้องการใช้คอมพิวเตอร์จริงต้องทำเนื่องข้อสิทธิ์ต่างๆที่ค่อนข้างยาก การทดสอบการติดตั้งของแต่ละแพลตฟอร์มไม่มีปัญหาสามารถติดตั้งได้ง่าย ยกเว้นตัวของ Quantum Developer Kit (Q#) ที่ต้องติดตั้ง 2 ขั้นตอนและใช้ภาษาในการพัฒนาที่แตกต่างจากแพลตฟอร์มอื่น การใช้งานฟังก์ชั่นและอัลกอริทึมของแต่แพลตฟอร์มใช้งานได้ดีแต่จะโดดเด่นที่ 2 แพลตฟอร์มคือ Qiskit และ ProjectQ ที่มีอัลกอริทึมเด่นๆและมีชื่อเสียงที่ดีกว่าแพลตฟอร์ม Forest (pyQuil) และ Quantum Developer Kit (Q#) โดยเปรียบเทียบจากการใช้งาน ทดสอบฟังก์ชั่นและอัลกอริทึมแล้วจะเห็นได้ว่า Qiskit และ ProjectQ จะมีความครบเครื่องใช้งานมาก จึงเป็น 2 แพลตฟอร์มที่ถูกคัดเลือกมาเปรียบเทียบประสิทธิภาพแบบจำลองโดยใช้ Shor's algorithm จะการทดสอบทำให้เห็นว่าQiski เป็นแพลตฟอร์มที่ดีที่สุดไม่ว่าจะเป็นในเรื่องการใช้งาน ฟังก์ชั่น อัลกอริทึม เอกสารที่ใช้สำหรับศึกษาสามารถตอบโจทย์ได้ดีกว่า 3 แพลตฟอร์ม

5.2 ข้อจำกัดของงานวิจัย

การวิจัยในครั้งมีข้อจำกัดในเรื่องของการใช้งานควอนตัมคอมพิวเตอร์จริงที่มีการขอสิทธิ์เข้าใช้งานที่ยุ่งยาก อีกทั้งแต่ละแพลตฟอร์มมีการจำลองคิวบิตที่จำกัดจำนวนและการใช้งานควอนตัมคอมพิวเตอร์จริงที่มีคิวบิตน้อย ทำให้การแก้ไขปัญหาไม่ว่าจะเป็นฟังก์ชั่นและอัลกอริทึมที่ใช้ทรัพยากรในการแก้ปัญหาที่สูงไม่สามารถใช้งานได้อย่างเต็มประสิทธิภาพ อีกทั้งยังแต่ละแพลตฟอร์มตัวโปรแกรมยังไม่มีความเสถียรมากเท่าที่ควรทำให้การทำงานบ้างครั้งสามารถเกิดข้อผิดพลาดระหว่างทำงานได้

5.3 ข้อเสนอแนะ

หากมีแพลตฟอร์มทางเลือกเพิ่มขึ้นในอนาคตควรเลือกแพลตฟอร์มที่มีเอกสารให้ศึกษาที่ชัดเจน เพื่อง่ายต่อการพัฒนาและศึกษาต่อและควรเลือกแพลตฟอร์มที่สามารถใช้ภาษาในปัจจุบันพัฒนาได้

บรรณานุกรม

บรรณานุกรม

[1] เอกสารออนไลน์ (13 เมษายน 2565). กลศาสตร์ดั้งเดิม สืบค้นจาก https://th.wikipedia.org/wiki/กลศาสตร์ดั้งเดิม.

[2] เอกสารออนไลน์ (13 เมษายน 2565). ความรู้เกี่ยวกับนิวเคลียร์ สืบค้นจาก https://www.oap.go.th/resources/105-articles/nuclear/129-nuclear-terms.

[3] เอกสารออนไลน์ (13 เมษายน 2565). อะตอม สืบค้นจาก https://sites.google.com/site/atom11062541/

[4] เอกสารออนไลน์ (13 เมษายน 2565). กลศาสตร์ควอนตัม สืบค้นจาก https://th.wikipedia.org/wiki/กลศาสตร์ควอนตัม.

[5] เอกสารออนไลน์ (13 เมษายน 2565). ทฤษฎีการทับซ้อน สืบค้นจาก http://rmutldoisaketnews.blogspot.com/2014/09/superposition.html.

[6] เอกสารออนไลน์ (14 เมษายน 2565). คิวบิท สืบค้นจาก https://dict.drkrok.com/qubit/.

[7] เอกสารออนไลน์ (14 เมษายน 2565). กฎธรรมชาติที่เหนือสามัญสำนึกของอะตอม สืบค้นจาก https://dict.drkrok.com/qubit/.

[8] เอกสารออนไลน์ (14 เมษายน 2565). สมการชเรอดิงเงอร์ สืบค้นจาก https://hmong.in.th/wiki/Schr%C3%B6dinger_equation.

[9] เอกสารออนไลน์ (15 เมษายน 2565). เทคโนโลยีควอนตัมคอมพิวเตอร์ สืบค้นจาก https://iconext.co.th/th/2021/12/07/เทคโนโลยีควอนตัมคอมพิว.

[10] เอกสารออนไลน์ (15 เมษายน 2565). โอเพ่นซอร์ส สืบค้นจาก https://tips.thaiware.com/1686.html.

[11] เอกสารออนไลน์ (15 เมษายน 2565). Qiskit สืบค้นจาก https://hmong.in.th/wiki/Qiskit.

[12] เอกสารออนไลน์ (15 เมษายน 2565). Quil สืบค้นจาก https://hmong.in.th/wiki/Quil_(instruction_set_architecture).

[13] เอกสารออนไลน์ (15 เมษายน 2565). ProjectQ สืบค้น https://ml2quantum.com/projectq.

[14] เอกสารออนไลน์ (15 เมษายน 2565). Quantum Development Kit สืบค้นจาก https://www.techtalkthai.com/microsoft-quantum-development-kit-is-released-with-q-sharp-language

บรรณานุกรม (ต่อ)

[15] เอกสารออนไลน์ (15 เมษายน 2565). Quantum Algorithms สืบค้นจาก https://ipass.wordpress.com/2009/06/20/computer-science-problems/

[16] เอกสารออนไลน์ (15 เมษายน 2565). อัลกอริทึมของชอร์ สืบค้นจาก https://th.wikipedia.org/wiki/ขั้นตอนวิธีของชอร์

[17] เอกสารออนไลน์ (15 เมษายน 2565). ก้าวหน้าในการคำนวณควอนตัม สืบค้นจาก https://www.bbc.com/news/science-environment-59320073

[18] Ryan LaRose. (2019) Overview and Comparison of Gate Level Quantum Software Platforms.

[19] Damian S. Steiger, Thomas Häner, and Matthias Troyer. (2016). ProjectQ: An Open Source Software Framework for Quantum Computing.

[20] Thomas Häner, Damian S Steiger, Krysta Svore and Matthias Troyer. (2018) A software methodology for compiling quantum programs.

[21] Danah Zohar. (2021). From the Newtonian Age to the Quantum Age.

[22] Katherine L. Brown ,William J. Munro and Vivien M. Kendon. (2010). Using Quantum Computers for Quantum Simulation.

[23] Jonathan R. Friedman, Vijay Patel, W. Chen, S. K. Tolpygo & J. E. Lukens. (2000). Quantum superposition of distinct macroscopic states

[24] Frank Leymann, Johanna Barzen and Michael Falkenthal. (2019). Towards a Platform for Sharing Quantum Software.

[25] Benjamin Weder, Johanna Barzen, Frank Leymann, Marie Salm, Daniel Vietz, Authors Info & Claims. (2020). The Quantum software lifecycle.

[26] Mark Fingerhuth ,Tomáš Babej and Peter Wittek. (2018). Open source software in quantum computing.

[27] I.M. Georgescu, S. Ashhab, and Franco Nori. (2014). Quantum simulation.

[28] Robert Wille Rod Van Meter and Yehuda Naveh. (2019). IBM's Qiskit Tool Chain: Working with and Developing for Real Quantum Computers.

บรรณานุกรม (ต่อ)

[29] Daniel Koch, Laura Wessing and Paul M. Alsing. (2019). Introduction to Coding Quantum Algorithms: A Tutorial Series Using Pyquil.

[30] Miguel-Angel Sicilia, Salvador Sánchez-Alonso, Marçal Mora-Cantallops and Elena García-Barriocanal. (2020). On the Source Code Structure of Quantum Code: Insights from Q# and QDK.

[31] G. L. Long. (2001). Grover algorithm with zero theoretical failure rate.

ประวัติผู้เขียน

ชื่อ – นามสกุล	นายเธียร สืบวงษ์
ประวัติการศึกษา	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยราชภัฏสวนสุนันทา พศ. 2561
ตำแหน่งและสถานที่ทำงานปัจจุบัน	Senior Sofeware Engineer บริษัท Ascend Commerce B2B