Quantum Simulation of Quantum Tunneling

Vikram Sharma

A dissertation submitted for the partial fulfilment
of BS-MS dual degree in Science

Indian Institute of Science Education and Research Mohali
December 2015

Certificate of Examination

This is to certify that the dissertation titled **Quantum Simulation of Quantum Tunneling** submitted by **Vikram Sharma** (Reg. No. MS10055) for the partial fulfillment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Prof. Arvind	Dr. Kavita Dorai	Dr. Ramandeep S. Johal
(Supervisor)	(Supervisor)	

Dated: December 7, 2015

Declaration

The work presented in this dissertation has been carried out by me under the guidance Prof. Arvind and Dr. Kavita Dorai at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Vikram Sharma
(Candidate)

Dated: December 7, 2015

In my capacity as the supervisor of the candidates project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Prof. Arvind
(Supervisor)

Dr. Kavita Dorai
(Supervisor)

Acknowledgment

First and foremost I wish to thank my advisors, Prof. Arvind and Dr. Kavita Dorai for their excellent guidance. I have been amazingly fortunate to have advisors like them who gave me the freedom to explore the subject on my own and at the same time gave the valuable guidance and timely advices to keep me going. I must thank them for their patience and encouragement when I was getting derailed from the path during the summer. I could not have imagined having better advisors and mentors for my project.

I am also thankful to my committee member Dr. Ramandeep Singh Johal for his insightful comments and questions during my presentations. I specially thank Prof. K.S. Viswanathan, who has been constantly addressing to my questions related to academic as well as social life and suggesting solutions to them. I am also very thankful to Dr. Rajeev Kapri, who has been mentor for the last three years. His words gave me the much needed motivation when I lost a semester due to a medical problem. Finally, I need to thank Prof. N. Sathyamurthy for providing nice opportunities and facilities at IISER Mohali.

I specially thank Harpreet Singh for helping me in the NMR experiments. Also, I sincerely thank Amandeep Singh for the helpful discussions. I would also like to thank the lab members: Debmalya Das, Shruti Dogra, Navdeep Gogna, Satnam Singh, Rakesh Sharma, Jyotsana Ojha, Tara George, Akshay Gaikwad, Jaskaran Singh, Atul Arora, Rajendra Bhati, Kishor Bharti, Aakash Sherawat, Varinder Singh and Dr. Arun.

Friends are prized possessions. My journey at IISER Mohali would not have been the same without my friends. First of all, I would like to thank Vivek who has been an incredible friend to me over the years. I must mention the name of Anju, who supported and helped me in times of trouble. I need to further thank my friends: Samridhi, Jagdeep and Deepanshu, whom I could always count on. They made my time at IISER Mohali unforgettable.

I fall short of words to thank my sisters for being very supportive and encouraging throughout my whole life. Last but not the least; I would like to thank my parents for all the support and unconditional love they have provided me over the years. Without their supports and sacrifices, I would have never gotten to where I am today.

List of Figures

1.1 Circuit representation of CNOT gate 5
1.2 Circuit swapping two qubits. 6

2.1 Quantum Circuit for Quantum Fourier transform. Swap gates are not shown in the circuit. 13
2.2 Quantum Circuit for Phase Estimation 16
2.3 Classical Gradient Estimation using $d+1$ points : $\frac{\partial f}{\partial x_i} \simeq \left(\frac{f(\vec{x}+l\vec{e_i})-f(\vec{x})}{l}\right)$ where $\vec{e_i}$ is the ith normalized vector 18
2.4 Quantum Circuit for solving the linear System$A\vec{x} = \vec{b}$ 26

3.1 Quantum Circuit for one time step of the simulation. 37
3.2 Probability distribution of the particle in a double well potential as a function of time for the first ten steps of two qubit simulation. 39
3.3 Probability distribution of the particle as a function of time for the first ten steps of three qubit simulation. 42
3.4 Probability distribution of the particle as a function of time for a free particle (Two qubits). 44
3.5 Probability distribution of the particle as a function of time for a free particle (Three qubits). 46
3.6 Probability distribution of the particle as a function of time (2 Qubits). There is a potential barrier at site 3. 49
3.7 Probability distribution of the particle as a function of time with a barrier at location 5(3 Qubits). 51
3.8 Probability distribution of the particle as a function of time (3 Qubits) with barriers at location 3 and location 5. 54
3.9 Probability distribution of the particle (superposition initial state) as a function of time (3 Qubits) with a barriers at location 3 and 5. 56

3.10 Probability distribution of the particle as a function of time (3 Qubits) with three barriers at locations 3, 4 and 5. 58
3.11 Probability distribution of the particle (superposition initial state) as a function of time (3 Qubits) with a barriers at locations 3, 4 and 5. . . 60
3.12 Probability distribution of the particle as a function of time for Dirac Comb Potential. 62

4.1 Spectra for Hydrogen after the four rotations: II, IX, IY, XX 76
4.2 Spectra of Hydrogen after adding a phase of 90^0 for the four rotations: II, IX, IY, XX 77
4.3 Spectra for Carbon after the four rotations: II, IX, IY, XX 78
4.4 Spectra of Carbon after adding a phase of 90^0 for the four rotations: II, IX, IY, XX 79
4.5 Quantum State Tomography: Real part of the elements of Density matrix 80
4.6 Quantum State Tomography: Imaginary part of the elements of Density matrix 80
4.7 Quantum Circuit for two qubit simulation of quantum tunneling. . . . 81
4.8 Pulse sequence for Quantum Fourier Transform gate. 82
4.9 Pulse sequence for Kinetic energy gate. 82
4.10 Pulse sequence for Potential energy gate. 83

List of Tables

3.1 Probabilities at four positions for different time steps (Double Well Potential- 2 Qubits) . . . 40
3.2 Probabilities at eight positions for different time steps (Double Well Potential- 3 Qubits) . . . 42
3.3 Probabilities at four positions for different time steps (Free Particle - 2 Qubits) . . . 45
3.4 Probabilities at four positions for different time steps (Free Particle - 3 Qubits) . . . 47
3.5 Probabilities at four positions for different time steps (Potential Barrier at Position 3) . . . 49
3.6 Probabilities at eight positions for different time steps (Potential barrier at site 5.) . . . 52
3.7 Probabilities at eight positions for different time steps (Potential barrier at site 3 and 5.) . . . 54
3.8 Probabilities at eight positions for different time steps(Potential barrier at site 3 and 5.) . . . 56
3.9 Probabilities at eight positions for different time steps (Potential barriers at site 3, 4 and 5.) . . . 59
3.10 Probabilities at eight positions for different time steps (Potential barriers at site 3, 4 and 5.) . . . 61
3.11 Probabilities at eight positions for different time steps (Dirac Comb Potential- 3 Qubits) . . . 63

Notation

$\mathbb{I}$	Identity Matrix(in appropriate dimensions)
σ_x	Pauli-X Matrix
σ_y	Pauli-Y Matrix
σ_z	Pauli-Z Matrix
R_x	Rotation along x-axis
R_y	Rotation along y-axis
R_z	Rotation along z-axis
H	Hadamrd Gate
U	Unitary Matrix
$Tr[A]$	Transpose of matrix A
$\|0\rangle$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$
$\|1\rangle$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Abstract

Quantum computer has the potential to solve certain problems which are hard for a classical computer. It takes advantage of quantum mechanical phenomena such as superposition and entanglement to achieve computations at significantly higher speeds. Simulation of physical systems is one of the most important practical applications of computation. It plays a crucial role in advancing the scientific knowledge and developing technologies. But as far as simulation of a quantum system is concerned, the exponential increase of the Hilbert space with the system size forbids its efficient simulation on a classical computer. The exponentially large basis set is needed to describe the system and it becomes too complicated to solve the Schrodinger equation exactly. Quantum computer can make use of this exponential complexity of quantum systems to simulate the dynamics of other quantum system. An exponential speed up is achieved in simulation of a quantum system by a quantum computer. Here we present the digital quantum simulation of quantum tunneling in certain one dimensional potentials such as double well potential, Dirac comb potential, single potential barrier in path, two potential barriers in path, and three potential barriers in path. The algorithms are discussed for two qubit system as well as three qubit systems. For potential barriers in path, results clearly demonstrate the tunneling of wave function from one side of barrier to another. A similar behavior is observed for double well potential where it tunnels from one well to another in addition to the oscillations within the well. For the sake of comparison, we also simulated the evolution of free particle with zero potential using the same schemes.

Other than the quantum simulation, we have discussed about quantum algorithms and NMR quantum computing. Three recently developed quantum algorithms are discussed in detail: algorithm for estimating numerical gradients of a function, algorithm to solve linear system of equations and algorithm to solve non-linear differential equations whose non-linear terms are polynomials. In NMR computing section, we have discussed about pseudo state preparation, quantum state tomography, and simulation of tunneling on an NMR quantum computer.

Contents

List of Figures **8**

List of Figures **i**

Notation **ii**

Abstract **iii**

1 Quantum Computation **1**

1.1 Introduction 1
1.2 Quantum Bit/Qubit 2
1.3 Quantum Gates: 3
1.4 Measurement: 7

2 Quantum Algorithms **9**

2.1 Introduction 9
2.2 Quantum Fourier Transform 10
2.2.1 Discrete Fourier Transform 10
2.2.2 Fast Fourier Transform 10
2.2.3 Quantum Fourier Transform 11
2.3 Phase Estimation 14
2.3.1 Introduction 14
2.3.2 Algorithm 14
2.3.3 Circuit Diagram 16
2.4 Algorithm for Numerical gradient estimation 17
2.4.1 Introduction 17
2.4.2 Classical Algorithm 17
2.4.3 Quantum Algorithm 18
2.4.4 Computational Resources 21

2.5 Algorithm to solve Linear Equations . . . 22
2.5.1 Introduction . . . 22
2.5.2 Algorithm . . . 22
2.5.3 Circuit . . . 25
2.6 Algorithm to solve Non Linear Differential Equations . . . 26
2.6.1 Introduction . . . 26
2.6.2 Algorithm . . . 27
2.6.3 Computational Resources . . . 30
2.6.4 Extension to Cubic or Higher Systems . . . 30
2.7 Summary . . . 31

3 Quantum Simulation of Quantum Tunneling 33
3.1 Quantum Simulation . . . 33
3.2 Theoretical protocol for quantum simulation of quantum tunneling . . . 34
3.3 Results and Discussion . . . 38
3.3.1 Double Well Potential . . . 38
3.3.2 Free Particle . . . 42
3.3.3 Single potential barrier in the path . . . 47
3.3.4 Two potential barriers in the path . . . 52
3.3.5 Three potential barriers in path . . . 56
3.3.6 Dirac Comb Potential . . . 61

4 NMR Quantum Computing 65
4.1 Introduction . . . 65
4.2 Pseudo-Pure State . . . 66
4.3 Single Qubit Gates . . . 68
4.4 Two Qubit Gates . . . 70
4.5 Quantum State Tomography . . . 72
4.6 Simulation of quantum tunneling on an NMR information processor . . . 81

A Derivation of Fourier coefficient α which emerges after the Phase estimation subroutine in Harrow's algorithm 85

B Mathematica Codes 89

Bibliography 99

Chapter 1

Quantum Computation

"It's not a fantasy to explore this question about making computers that are much, much, more powerful than the kind that we have sitting around now – in which a grain of salt has all the computational powers of all the computers in the world."

-Seth Lloyd

1.1 Introduction

Computer technology has improved over a long period of time. Its history involved a long series of transformations from one type of physical realization to another: gears to relays to valves to transistors to integrated circuits and so on. Today's advanced technology can produce chips with features only a fraction of micron wide. Soon, the technology will yield even smaller parts and it is inevitable that a point will be reached where the logic gates will be made up of only a handful of atoms. At these length scales, the familiar classical laws of physics which determine the properties of conventional logic gates no longer hold. The world at these tiny length scales has entirely different behavior and is governed by laws of quantum mechanics. If computers are to become smaller in future, quantum technology must supplant the present conventional technology. The more interesting part is that this new quantum technology can support an entirely new kind of computation where certain difficult tasks that have been long thought intractable for a classical computer can be solved quickly and efficiently [Ste97].

1.2 Quantum Bit/Qubit

The most fundamental unit of information in classical computation is bit. A bit is a physical system which can be prepared in one of the two different states representing two binary logical values: 0 or 1. Quantum computation is built upon an analogous concept and the quantum analogue of bit is referred to as Qubit(quantum bit). Qubit is a quantum system that encodes the '0' and the '1' into two distinguishable quantum states. Unlike the classical bit, a qubit is not confined to its two basis states, but can exist in a superposition state. The general state for a qubit can be written as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.1}$$

The numbers α and β are complex number which are constrained by the normalization condition $||\psi|| = |\alpha|^2 + |\beta|^2 = 1$. All possible states of a qubit form a two-dimensional complex vector space which can be visualized on a Bloch Sphere. The special states $|0\rangle$ and $|1\rangle$ are called computational basis states.

The general state of a n-qubit system can be written as:

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k|k\rangle \tag{1.2}$$

where:

$$\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1 \tag{1.3}$$

The major properties that make qubits different from bits are the existence of superposition states and the concept of entanglement.A quantum computer with n-qubits can be in a state which is superposition of all the possible 2^n states. This superposition gives quantum computers their inherent parallelism. An operation on this superposition state can be seen as on operation on 2^n states simultaneously. In contrast a classical computer can only be found in one of these 2^nstates at any one time. Thanks to superposition and entanglement, a quantum computer can thus process a very large number of calculations simultaneously.

1.3 Quantum Gates:

The basic component of classical computer circuits are "logic gates" which perform manipulations of the information, converting it from one form to another. Similarly in a quantum circuit, one must be able to manipulate the states of a qubit in order to process the information. These manipulation are performed my quantum gates. Using quantum gates one change the state of a qubit either unconditionally(eg. initialization of qubit), or conditionally, depending on the previous state of the qubit (e.g., NOT operation) or on the sate itself and state of another qubit (e.g., controlled rotation) and so on. A lot of complicated gates can be imagined where the state of one or more qubits depends on the state of arbitrary number of qubits. But all these complicated gates/operation can be decomposed to a finite set of "universal quantum gates". Only one- and two- qubit operations can be used to construct any arbitrary operation.

Although quantum gates follow Boolean Algebra, they differ from the classical logic gates in a variety of ways. In a quantum system, the bit values are represented using the quantum state of atomic system. Thus a logic gate can nether create or destroy the state. So the possibility of gates such as AND is ruled out. Two peculiar properties which contribute to the uniqueness of the Quantum logic gates are:

(1) Reversibility:
The initial state of the qubit is transformed to the final state using only those processes whose action can be inverted. This property is called reversibility.

(2) Unitarity: The transformation of the state through reversible operation is constrained by the condition that this operation must preserve the norm of vector which is mapped from one orthonormal basis to another. Thus the transformation is unitary, satisfying the following condition:

$$UU^{\dagger} = U^{\dagger}U = 1 \tag{1.4}$$

Some basic logic gates widely used in quantum circuits are:

NOT Gate:

Similar to the classical case, a quantum NOT gate takes the state $|0\rangle$ to state $|1\rangle$

and vice versa. As it can be seen, it is a single qubit operator.

$$\begin{aligned} X : |x\rangle &\longrightarrow |\bar{x}\rangle \\ X|0\rangle &\longrightarrow |1\rangle \\ X|1\rangle &\longrightarrow |0\rangle \end{aligned} \tag{1.5}$$

In a matrix form the NOT gate can be written as:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.6}$$

The action of X on a general qubit state $\alpha|0\rangle + \beta|1\rangle$ can be seen as:

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{1.7}$$

Hadamard Gate:

It is a unique single-qubit gate with no classical analogue. Its uniqueness lies in the fact that it transforms one computational basis state to a superposition of both basis states.

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \\ H|1\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \end{aligned} \tag{1.8}$$

The matrix form for the Hadamard gate can be written as:

$$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.9}$$

Hadamard gate is sometimes described as 'square root of NOT' gate as it turns $|0\rangle$ into halfway between $|0\rangle$ and $|1\rangle$ and has a similar effect on state $|1\rangle$.

CNOT gate:

The *controlled*-NOT or CNOT gate is a two-qubit quantum gate. Its two input qubits are known as control-qubit and target-qubit. The action of CNOT gate may be described as follow:

- If the control bit is in state $|0\rangle$, then the target bit is left alone.
- If the control bit is in state $|1\rangle$, then the target bit is flipped i.e., a NOT gate is applied to the target bit.

In equations it can be written as:

$$\begin{aligned} \text{CNOT} : |00\rangle &\longrightarrow |00\rangle \\ \text{CNOT} : |01\rangle &\longrightarrow |01\rangle \\ \text{CNOT} : |10\rangle &\longrightarrow |11\rangle \\ \text{CNOT} : |11\rangle &\longrightarrow |10\rangle \end{aligned} \tag{1.10}$$

or

$$\text{CNOT} : |x, y\rangle \longrightarrow |x, x \oplus y\rangle \tag{1.11}$$

where '$\oplus$' represents additional modulo two function.
The matrix form of CNOT-gate in the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ is written as:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.12}$$

The circuit representation of CNOT- gate is shown in the figure[1.1]. Here the top line represents the control qubit and the bottom line represents the target qubit.

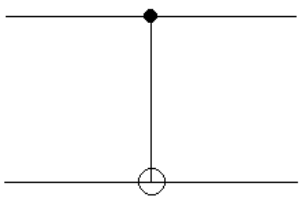


Figure 1.1: Circuit representation of CNOT gate

SWAP gate:

It swaps the states of two qubits.

$$\begin{aligned} \mathrm{S}|00\rangle &\longrightarrow |00\rangle \\ \mathrm{S}|01\rangle &\longrightarrow |10\rangle \\ \mathrm{S}|10\rangle &\longrightarrow |01\rangle \\ \mathrm{S}|11\rangle &\longrightarrow |10\rangle \end{aligned} \tag{1.13}$$

or

$$\mathrm{S} : |x, y\rangle \longrightarrow |y, x\rangle \tag{1.14}$$

In the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, it is represented by matrix:

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.15}$$

The swap gate can be achieved using three CNOT gates as shown in figure 1.2

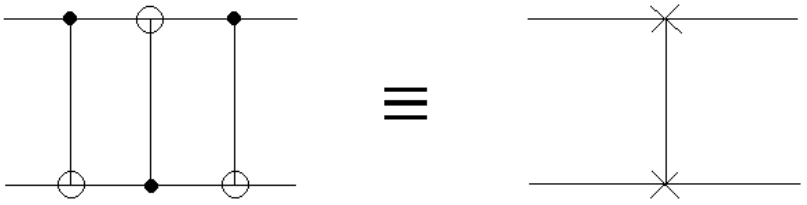


Figure 1.2: Circuit swapping two qubits.

The sequence of CNOT-gates has the following sequence of effects on a computation basis state $|x, y\rangle$:

$$\begin{aligned} |x, y\rangle &\longrightarrow |x, x \oplus y\rangle \\ &\longrightarrow |x \oplus (x \oplus y), x \oplus y\rangle = |y, x \oplus y\rangle \\ &\longrightarrow |y, (x \oplus y) \oplus y\rangle = |y, x\rangle \end{aligned} \tag{1.16}$$

The effect of the circuit is to interchange the state of two qubits i.e. a SWAP-gate.

1.4 Measurement:

The only irreversible operation which is possible in a quantum circuit is 'Measurement'. Information about a state can be gathered only by making measurements but the very act of measurement disturbs the system itself and its originality is hence lost. The process of measurement is difficult to grasp as it involves the phenomenon at the border quantum system and the environment. The issue of measurement serves as the base for the problem of the interpretation of quantum mechanics. Measurement is viewed in different ways among various interpretations of quantum mechanics. In spite of philosophical differences, all these interpretations agree on the practical question of what results form a routine measurement. The probability that a measurement will yield a given result is given by Born rule.

The Born rule states if a measurement is made for an observable corresponding to Hermitian operator A on a quantum system with normalized wave function $|\psi\rangle$,then:

- the measurement will yield one of the eigenvalues 'λ' of matrix A.
- the probability of getting a particular eigenvalue 'λ_i' is equal to $|\langle\lambda_i|\psi\rangle|^2$ where $|\lambda_i\rangle$ is the i_{th} eigenvector. Due to normalization condition we have: $(\sum_i |\langle\lambda_i|\psi\rangle|^2) = 1$. Immediately after the measurement, the state of the system is:

$$\frac{\mathbf{P}_i|\psi\rangle}{||\mathbf{P}_i|\psi\rangle||} \tag{1.17}$$

  where $\mathbf{P}_i$ is the projection operator onto the eigenspace of A corresponding to λ_i [Sak94].

Thus the measurement gates project the state of the system to the computational basis i.e any measurement will yield either a zero or one. One cannot realize the coefficients or the respective amplitudes of any of these possible states by a single measurement.

Chapter 2

Quantum Algorithms

"There is something magical about quantum algorithms. This magic can make one, like me, who is not an expert in quantum mechanics feel lost. Being puzzled seems to be the first step toward showing that you understand quantum algorithms."

-Dick Lipton

2.1 Introduction

An algorithm is a finite sequence of instructions, or a step-by-step procedure for solving a problem. A classical algorithm is one where each of these steps or instruction can be executed on a classical computer. In the same way, a quantum algorithm is a step-by-step procedure, where all these steps can be performed on a quantum computer. Even though each and every classical algorithm can also be performed on a quantum computer, the phrase quantum algorithm is typically used for those algorithms which appear intrinsically quantum, or use some essential trait of quantum computation such as quantum superposition or quantum entanglement.

All the problems which can be worked out on a quantum computer can also be worked out on a classical computer. The undecidable problems on classical computer remain undecidable on a quantum computer. The interesting part about quantum algorithms is that in many cases these can be executed much faster than the classical counterparts. The most renowned quantum algorithms are : Shor's algorithm for factoring, and Grover's algorithm for searching an unstructured database. While the Shors algorithms is exponentially faster than its classical counterpart, Grovers algorithm runs quadratically faster than the best known classical algorithm for the

same problem.

In the past few years, several other quantum algorithms have been developed exploiting various algebraic structures, symmetries and geometries. In this chapter, we have explored some of the recently given quantum algorithms such as algorithm to find out the gradient of a function, algorithm to solve linear system of equations and algorithm to solve non-linear differential equations. Key ideas such as Fourier transform and phase estimation which are the building blocks to these algorithms are also discussed.

2.2 Quantum Fourier Transform

2.2.1 Discrete Fourier Transform

The classical discrete Fourier transform maps a complex input vector with components $x_0, x_1, ..x_{N-1}$ to an output vector with components $y_1, y_2, ..., y_{N-1}$ as per the following formula [NC00] [SS08] :

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi\iota}{N} jk} x_j \tag{2.1}$$

This transformation can be seen as $N \times N$ matrix multiplied to a N-dim vector involving N^2 multiplications. Let $\omega_N = e^{\frac{2\pi\iota}{N}}$ be the N-th root of identity. Then Fourier transform is a $N \times N$ matrix F_N with (j,k)th entry $= \frac{1}{\sqrt{N}}\omega_N^{jk}$. It can be easily seen that F_N is a unitary matrix. Since it is unitary and symmetric, the inverse $F_N^{-1} = F_N^*$ only differs from F_N by having a negative sign in the exponent of the entries.

2.2.2 Fast Fourier Transform

A more efficient way of computing Fourier transform involves $O(N\log(N))$ steps instead of $O(N^2)$ steps. The algorithm is called Fast Fourier Transform (FFT). It is based on the idea that by separating even and odd j in (2.1), one can reduce the number of operations to half. The equation (2.1) can be rewritten in a following way after separating even and odd j (Assuming N to be even):

$$y_k = \frac{1}{\sqrt{N}} \left(\sum_{l=0}^{\frac{N}{2}-1} e^{\frac{2\pi\iota}{N/2}kl} x_{2l} + e^{\frac{2\pi\iota}{N}k} \sum_{l=0}^{\frac{N}{2}-1} e^{\frac{2\pi\iota}{N/2}kl} x_{2l+1} \right) \tag{2.2}$$

The above equation can be seen as a discrete Fourier transform of two $N/2$-dimensional vectors leading to an operation count of $2 \times \left(\frac{N}{2}\right)^2 = \frac{1}{2}N^2$. Following that Fourier transform of each $N/2$-dimensional vector can further be divided into two $N/4$-dimensional vectors. This process can be continued further and further until no more divisions can be made. Continuation of this process for $N = 2^n$ yields the FFT algorithm. As mentioned earlier, it reduces the operation count from $O(N^2)$ to $O(N\log(N))$.

2.2.3 Quantum Fourier Transform

The quantum Fourier Transform is an efficient quantum algorithm to perform Fourier transform of quantum mechanical amplitudes. It is exactly the same transformation, mapping an N-dimensional vector of amplitudes to another N-dimensional vector of amplitudes. QFT is exponentially faster than even the FFT. It gives us the entries of the Fourier transform only as the amplitudes of the resulting state.

The F_N matrix can be seen as a linear operator which maps the basis state of N-dimensional Hilbert space as:

$$|j\rangle \longrightarrow \sum_{j=0}^{N-1} e^{\frac{2\pi\iota}{N}jk} |k\rangle \tag{2.3}$$

Quantum Fourier transform (F_N) maps an arbitrary state to some other state as following:

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle \tag{2.4}$$

where the amplitudes y_k's of the resulting state are the discrete Fourier transform of the amplitudes x_j's of the initial state.

Product Representation

Suppose $N = 2^n$ and $|0\rangle, |1\rangle, ..., |2^n - 1\rangle$ is the computational basis of a n-qubit computer. Basis state can be represented by the integer j ($0 \leq j \leq 2^n - 1$) with binary

representation $j = j_1 j_2 j_3 ... j_n$. Quantum Fourier transform of $|j\rangle$ can be written as:

$$|j\rangle \longrightarrow 2^{-\frac{n}{2}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi\iota}{2^n} jk} |k\rangle \tag{2.5}$$

Inserting the binary expansion of k leads to:

$$|j\rangle \longrightarrow 2^{-\frac{n}{2}} \sum_{k_1=0}^{1} ... \sum_{k_n=0}^{1} e^{\frac{2\pi\iota}{2^n} j\left(\sum_{l=1}^{n} k_l 2^{n-l}\right)} |k_1 ... k_n\rangle \tag{2.6}$$

$$= 2^{-\frac{n}{2}} \sum_{k_1=0}^{1} ... \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi\iota j k_l 2^{-l}} |k_l\rangle \tag{2.7}$$

$$= 2^{-\frac{n}{2}} \bigotimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi\iota j k_l 2^{-l}} |k_l\rangle \right) \tag{2.8}$$

$$== 2^{-\frac{n}{2}} \bigotimes_{l=1}^{n} \left(|0\rangle_l + e^{2\pi\iota j 2^{-l}} |1\rangle_l \right) \tag{2.9}$$

Also we know:

$$j2^{-l} = \sum_{v=1}^{n} j_v 2^{n-v-l} = j_1 j_2 ... j_{n-l} . j_{n-l+1} ... j_n \tag{2.10}$$

The integer part can be discarded as $e^{\iota 2\pi k} = 1$. So quantum Fourier transform can be written in a product form as:

$$|j\rangle \longrightarrow 2^{-\frac{n}{2}} \left(|0\rangle_1 + e^{2\pi\iota 0.j_n} |1\rangle_1 \right) \left(|0\rangle_2 + e^{2\pi\iota 0.j_{n-1} j_n} |1\rangle_2 \right) ... \left(|0\rangle_n + e^{2\pi\iota 0.j_1 j_2 ... j_{n-1} j_n} |1\rangle_n \right) \tag{2.11}$$

Circuit

The quantum Fourier transform can be seen as qubit-wise phase shift . The $|1\rangle$ state of each qubit gets an extra phase factor. Suppose gate R_k represents the unitary transformation:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi\iota}{2^k}} \end{bmatrix} \tag{2.12}$$

Fig 2.1 shows the circuit for QFT [NC00]:

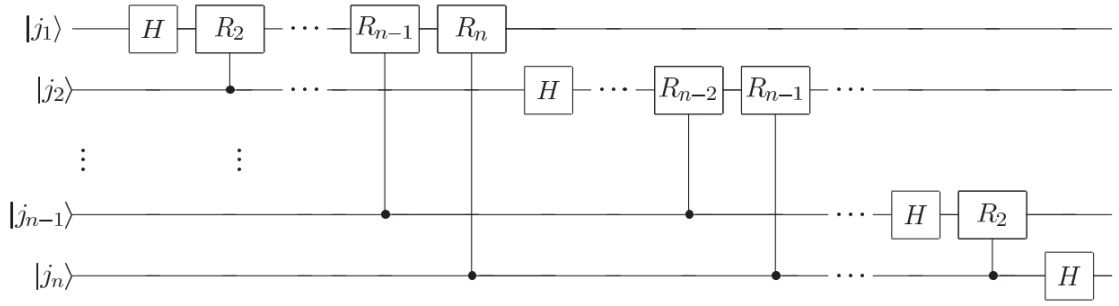


Figure 2.1: Quantum Circuit for Quantum Fourier transform. Swap gates are not shown in the circuit.

Application of Hadamard gate on first qubit of state $|j_1 j_2 ... j_n\rangle$ produces the state:

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi\iota 0.j_1}|1\rangle\right) |j_2 ... j_n\rangle \tag{2.13}$$

since $e^{2\pi\iota 0.j_1} = (-1)^{j_1}$. The application of controlled-R_2 produces the state:

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi\iota 0.j_1 j_2}|1\rangle\right) |j_2 ... j_n\rangle \tag{2.14}$$

The further application of controlled-$R_3, R_4, ..., Rn$ gates keep appending bits to the exponent of the phase factor $|1\rangle_1$, finally leading to:

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi\iota 0.j_1 j_2 ... j_n}|1\rangle\right) |j_2 ... j_n\rangle \tag{2.15}$$

A similar procedure is performed on second qubit. The Hadamard gate generates:

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi\iota 0.j_1 j_2 ... j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi\iota 0.j_2}|1\rangle\right) |j_3 ... j_n\rangle \tag{2.16}$$

The application of controlled-R_2, controlled-R_3 ....controlled-R_{n-1} produces the state:

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi\iota 0.j_1 j_2 ... j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi\iota 0.j_2 j_3 ... j_n}|1\rangle\right) |j_3 ... j_n\rangle \tag{2.17}$$

The procedure is continued for each qubits, giving the final state:

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi\iota 0.j_1 j_2 ... j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi\iota 0.j_2 j_3 ... j_n}|1\rangle\right) ... \left(|0\rangle + e^{2\pi\iota 0.j_n}|1\rangle\right) \tag{2.18}$$

This state is identical to the state (2.11) except for the order of qubits. The order

of the qubits is reversed using the swap operations and we get the desired result. The total number of operations/gates can be easily counted. The first qubit is acted upon one Hadamard gate and $(n-1)$ controlled-R gates, making a total of n gates. The next qubit needs one controlled-R gate less and so on. Thus the total number of Hadamard and controlled-R gates $= n + (n-1) + (n-2) + ... + 1 = \frac{n(n+1)}{2}$. Also at last one need about $n/2$ swap gates and each swap gate can be made using three CNOT gates. Thus the circuit needs $O(n^2)$ gates/operations which is exponentially better than FFT which needs $O(n2^n)$ steps.

2.3 Phase Estimation

2.3.1 Introduction

Quantum Phase estimation algorithm is a key procedure for many other important algorithms [NC00] [SS08]. Suppose we are given a unitary operator U on n-qubits with a known eigenvector $|u\rangle$ and an unknown eigenvalue $e^{2\pi\iota\phi}$. The algorithm can then be used to find the phase ϕ with of precision of m-bits.

$$U|u\rangle = e^{2\pi\iota\phi}|u\rangle \tag{2.19}$$

where $0 < \phi < 1$.

It is assumed that two black-boxes are given which can prepare the state $|u\rangle$ and perform the controlled - U^{2^j} operations (for j = 0, 1, ...). The algorithm requires two registers. The first register consists of 'm' number of qubits which decides the accuracy and the success probability of the algorithm. It is initialized in the state $|0\rangle$. The second register holds the state $|u\rangle$ and contains as many qubits as are necessary to store the vector $|u\rangle$.

2.3.2 Algorithm

Step1: Hadamard Transformation

Hadmard-Transformation is applied on the first register to produce equal-weight and equal-phase superposition state.

$$\mathbf{H}^{\otimes m}|\vec{0}\rangle = \frac{1}{\sqrt{2^m}}\sum_{x=1}^{2^m}|x\rangle \tag{2.20}$$

Step2: Controlled Rotations

In the next step, controlled -U^{2^k} operations are applied on register 2, using the k_{th} qubit of the first register as control.
If the k_{th} qubit is $|0\rangle$:

$$|u\rangle \Longrightarrow |u\rangle \tag{2.21}$$

If the k_{th} qubit is $|1\rangle$:

$$|u\rangle \Longrightarrow e^{2\pi\iota 2^k\phi}|u\rangle \tag{2.22}$$

It can be seen that even after the application of controlled -U^{2^k} operations, register 2 stays in the state$|u\rangle$ upto some phase factors. These phase factors can be collected next to the qubits of register 1. The state of the system can be written as :

$$\begin{aligned}|\psi_1\rangle &= \left(\frac{1}{2^{m/2}}\left(|0\rangle + e^{2\pi\iota 2^{m-1}\phi}|1\rangle\right)\left(|0\rangle + e^{2\pi\iota 2^{m-2}\phi}|1\rangle\right)..\left(|0\rangle + e^{2\pi\iota 2^0\phi}|1\rangle\right)\right)|u\rangle \\ &= \left(\frac{1}{2^{m/2}}\sum_{k=0}^{2^{m-1}} e^{2\pi\iota\phi k}|k\rangle\right)|u\rangle \end{aligned} \tag{2.23}$$

Suppose that ϕ is a m-bit binary fraction, $\phi = 0.\phi_1\phi_2\phi_3....\phi_m$. The state of the system can then be rewritten as:

$$\left(\frac{1}{2^{m/2}}\left(|0\rangle + e^{2\pi\iota 0.\phi_m}|1\rangle\right)\left(|0\rangle + e^{2\pi\iota 0.\phi_{m-1}\phi_m}|1\rangle\right)..\left(|0\rangle + e^{2\pi\iota 0.\phi_1\phi_2..\phi_{m-1}\phi_m}|1\rangle\right)\right)|u\rangle \tag{2.24}$$

Step 3: Inverse Fourier Transform

In the final step in which we do the inverse Fourier transform on the first register. Equation (2.24) is similar to the product form of the Fourier transform (Equation (2.11)). Thus after application of inverse Fourier transform the state of the first register will be: $|\phi_1\phi_2\phi_3....\phi_m\rangle$. The state of the system is thus:

$$|\psi_2\rangle = |\phi_1\phi_2\phi_3....\phi_m\rangle|u\rangle = |\phi\rangle|u\rangle \tag{2.25}$$

A measurement on the register 1 thus gives us exactly the phase ϕ. If ϕ is longer than m-bits , then we only get an estimate instead of exact value.

Suppose we use some state $|\psi\rangle$ instead of eigenstate $|u\rangle$ in the second register. $|\psi\rangle$ can be written as linear combination U-eigenstates:

$$|\psi\rangle = \sum_u c_u |u\rangle, \tag{2.26}$$

where

$$U|u\rangle = e^{2\pi\iota\phi_u}|u\rangle \tag{2.27}$$

At the end of phase estimation algorithm, we will get the state of the system as:

$$\sum_u c_u |\phi_u^{/}\rangle|u\rangle, \tag{2.28}$$

where $\phi_u^{/}$ is an approximation to the phase ϕ_u.

2.3.3 Circuit Diagram

Fig 2.2 shows the circuit for the Phase estimation [NC00]:

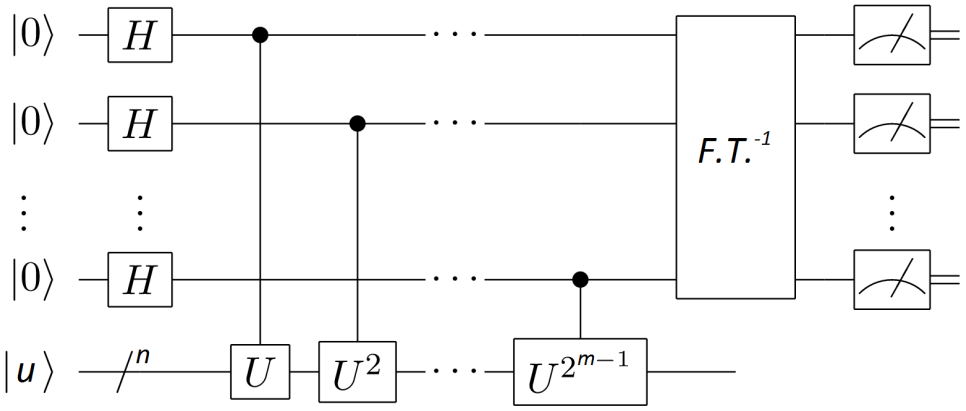


Figure 2.2: Quantum Circuit for Phase Estimation

H here represents Hadamard Transformation and $F.T^{-1}$ represents inverse Fourier Transform. As described before, there are two registers of length m and n respectively.

2.4 Algorithm for Numerical gradient estimation

2.4.1 Introduction

Recently, Stephen P. Jordan proposed a fast quantum algorithm for estimating numerical gradients [Jor05]. Jordan's method needs a black box /oracle that can compute the value of fuction *f* for an arbitrary input. In general, the efficiency of an algorithm is often related to the black box complexity. The function evaluation steps are generally the most time consuming steps of an algorithm. Lesser the number of the function evaluation steps, better is the algorithm. Regardless of the number of the dimensions *d* of the domain of *f*, Jordan's algorithm can evaluate the gradient using a single query to *f*. In contrast, the classical algorithm requires at least *d+1* queries (when simple classical finite-difference scheme is used). The speed-up of the quantum algorithm over the classical one is achieved by being able to sample all the *d* dimensions in superposition.

2.4.2 Classical Algorithm

Numerical derivative techniques rely on computing the function at several discrete points, and then using those values to estimate the true derivative [KAG09]. In the finite- difference scheme, the first derivative in one dimension is given by the formula:

$$\frac{df}{dx} \approx \frac{f(x+l) - f(x)}{l} \tag{2.29}$$

Here l is assumed to be very small such that the quadratic and higher terms in f can be neglected. For simplicity, now onwards we will discuss the gradient estimation at the origin only, since the gradient at other points can be obtained by trivially redefining f.

$$\frac{df}{dx} \approx \frac{f(l) - f(0)}{l} \tag{2.30}$$

This can be easily generalized to d- dimensions. To estimate $\bigtriangledown f$ in d-dimensions , one needs to evaluate f at least $d+1$ times, once at the origin and once at a distance l along each axis.

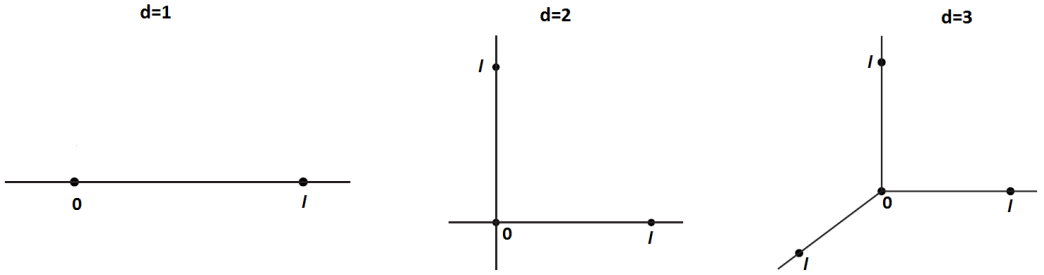


Figure 2.3: Classical Gradient Estimation using $d+1$ points : $\frac{\partial f}{\partial x_i} \simeq \left(\frac{f(\vec{x}+l\vec{e_i})-f(\vec{x})}{l}\right)$ where $\vec{e_i}$ is the ith normalized vector

In d-dimensions, evaluations of f at each point $\mathbf{x}_i$ gives a linear equation of the form $f(\mathbf{x}) = f(\mathbf{0}) + \mathbf{x}. \bigtriangledown f$. The equation has $d+1$ unknowns , thus evaluating f at $d+1$ distinct points is required.

$$\frac{\partial f}{\partial x_1} = \frac{f(l, 0...0) - f(0, 0...0)}{l} \tag{2.31}$$

similarly other partial derivatives can be calculated.

The error is linear with respect to l and is contributed by the quadratic and the higher terms in f.

For a lower magnitude of error, one can use a better method in which functions are evaluated at $\pm l/2$ from origin along each dimension. In this case $2d$ evaluations are required. $\frac{\partial f}{\partial x_1}$ will be then given by:

$$\frac{\partial f}{\partial x_1} = \frac{f(+l/2, 0...0) - f(-l/2, 0...0)}{l} \tag{2.32}$$

similarly other partial derivatives can be calculated.

Using the Taylor expansion of f it can be seen that the error is of the order of l^2 in this case, which is better than the previous method.

2.4.3 Quantum Algorithm

The black box takes d binary strings as input. Each of the binary string is of length n, where n is the precision we want in the final gradient. Apart from these $n \times d$ qubits, there are n_0 ancilla qubits which are all initialized to zero. The output of the black box is written in these ancilla qubits using addition modulo $N_0 \equiv 2^{n_0}$. The input qubit string represent the components of $\mathbf{x}$ in fixed point notation. Positive integers

$\boldsymbol{\delta}$ represented by the input string and values of $\mathbf{x} \in R^d$ are related by:

$$\boldsymbol{x} = \frac{l}{N}\left(\boldsymbol{\delta} - \frac{\boldsymbol{N}}{2}\right) \tag{2.33}$$

here components of $\boldsymbol{\delta}$ are n-bit integers ranging from 0 to $N \equiv 2^n$ and $\mathbf{N}$ is the d-dimensional vector $(N, N, N.....N)$ which serves to center the sampling region on the origin.

Similar to the input , the output of the black-box represents the value of f in fixed point notation. To achieve maximum precision with fewest qubits, one must have an order of magnitude estimate of the range of f. In the present case f ranges from $f(\mathbf{0}) - M\frac{l}{2}$ to $f(\mathbf{0}) + M\frac{l}{2}$ where M is the largest value of any of the first partial derivatives of f. The value of f is scaled and the integer which gets added modulo N_0 to the output register is:

$$\lceil \frac{NN_0}{ml} f(\boldsymbol{x}) \rfloor \tag{2.34}$$

where $\lceil \rfloor$ denotes rounding to the nearest integer and m is the estimate of M.

Step 1 : Hadamard Transformation

The algorithm starts with Hadamard transformation on the input registers to create a uniform superposition. The state of the input registers is:

$$\frac{1}{\sqrt{N^d}} \sum_{\delta_1=0}^{N-1} \sum_{\delta_2=0}^{N-1} ... \sum_{\delta_d=0}^{N-1} |\delta_1\rangle|\delta_2\rangle...|\delta_d\rangle \tag{2.35}$$

which can be written in vector notation as:

$$\frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{\delta}} |\boldsymbol{\delta}\rangle \tag{2.36}$$

Step 2 : Plane wave

In the next step, '1' is written into the output register and then the Fourier transform is performed on it to create a "plane wave". The state of the output register can written as:

$$\frac{1}{\sqrt{N_0}} \sum_{a=0}^{N_0-1} e^{\frac{\iota 2\pi a}{N_0}} |a\rangle \tag{2.37}$$

The overall state of both the registers can be written as:

$$\frac{1}{\sqrt{N^d N_0}} \sum_{\delta_1=0}^{N-1} \sum_{\delta_2=0}^{N-1} ... \sum_{\delta_d=0}^{N-1} |\delta_1\rangle|\delta_2\rangle...|\delta_d\rangle \sum_{a=0}^{N_0-1} e^{\frac{\iota 2\pi a}{N_0}} |a\rangle \tag{2.38}$$

$$= \frac{1}{\sqrt{N^d N_0}} \sum_{\boldsymbol{\delta}} |\boldsymbol{\delta}\rangle \sum_{a=0} e^{\frac{\iota 2\pi a}{N_0}} |a\rangle \tag{2.39}$$

Step 3 : Phase Kickback

In this step, we use the black box to compute f at $\boldsymbol{x} = \frac{l}{N}\left(\boldsymbol{\delta} - \frac{\boldsymbol{N}}{2}\right)$ for every integer vector $|\boldsymbol{\delta}\rangle$ and then add (modulo N_0) the integer $\lceil \frac{NN_0}{ml} f(\boldsymbol{x}) \rfloor$ into the output register. The addition of x corresponds to eigenvalue $e^{\frac{\iota 2\pi x}{N_0}}$. Thus we obtain a phase proportional to f by writing into the output register via modular addition. This technique is known as "Phase Kickback". The state of the system becomes:

$$\frac{1}{\sqrt{N^d N_0}} \sum_{\boldsymbol{\delta}} e^{\frac{\iota 2\pi N}{ml} f[(\frac{l}{n})(\boldsymbol{\delta} - \frac{\boldsymbol{N}}{2})]} |\boldsymbol{\delta}\rangle \otimes \sum_{a=0} e^{\frac{\iota 2\pi a}{N_0}} |a\rangle \tag{2.40}$$

We can Taylor expand the phase of the above equation.The terms which are quadratic or higher order in l can be neglected for sufficiently small l. The state can be thus written as:

$$\approx \frac{1}{\sqrt{N^d N_0}} \sum_{\boldsymbol{\delta}} e^{\frac{\iota 2\pi N}{ml}(f(\mathbf{0}) + (l/n)[\boldsymbol{\delta} - \boldsymbol{N}/2].\bigtriangledown f)} |\boldsymbol{\delta}\rangle \otimes \sum_{a=0} e^{\frac{\iota 2\pi a}{N_0}} |a\rangle \tag{2.41}$$

Writing out the vector components, and ignoring the global phase, we can rewrite the state of the input registers as:

$$= \frac{1}{\sqrt{N^d}} \sum_{\delta_1=0}^{N-1} \sum_{\delta_2=0}^{N-1} ... \sum_{\delta_d=0}^{N-1} e^{(\frac{\iota 2\pi}{m})[\delta_1(\frac{\partial f}{\partial x_1}) + \delta_2(\frac{\partial f}{\partial x_2}) + ... + \delta_d(\frac{\partial f}{\partial x_d})]} \times |\delta_1\rangle|\delta_2\rangle...|\delta_d\rangle \tag{2.42}$$

The above state is separable and equals:

$$\frac{1}{\sqrt{N^d}} \times \left(\sum_{\delta_1} e^{(\frac{\iota 2\pi}{m})\delta_1(\frac{\partial f}{\partial x_1})} |\delta_1\rangle \right) ... \left(\sum_{\delta_d} e^{(\frac{\iota 2\pi}{m})\delta_d(\frac{\partial f}{\partial x_d})} |\delta_d\rangle \right) \tag{2.43}$$

The global phase associated with the above state is:

$$\phi(\mathbf{0}) = 2\pi \left(\frac{N}{lm} f(\mathbf{0}) - \frac{\boldsymbol{N}}{2m} . \bigtriangledown f \right) \tag{2.44}$$

Step 4: Inverse Fourier Transform

This is the final step in which the inverse Fourier transform is applied to each of the d registers. The resultant state is :

$$= e^{\iota\phi(\mathbf{0})} \left|\frac{N}{m}\frac{\partial f}{\partial x_1}\right\rangle \left|\frac{N}{m}\frac{\partial f}{\partial x_2}\right\rangle ... \left|\frac{N}{m}\frac{\partial f}{\partial x_d}\right\rangle \tag{2.45}$$

$$= e^{\iota\phi(\mathbf{0})} \left|\frac{N}{m} \bigtriangledown f\right\rangle \tag{2.46}$$

A simple measurement in the computational basis can be used to obtain the components of $\bigtriangledown f$ with n bits of precision.

2.4.4 Computational Resources

In general , when r is the range of some quantity and θ is the minimum quantity which one can resolve, then number of bits required to represent the quantity are given by:

$$n = log_2\left(\frac{r}{\delta}\right) \tag{2.47}$$

In the classical case, if the gradient is desired to n bits of precision, then the function must be evaluated to:

$$log_2\left(\frac{max(f) - min(f)}{ml/2^n}\right) \tag{2.48}$$

bits of precision.

For the quantum case, phase acquired by the system after the "Phase kickback" step is $\frac{2\pi N}{ml} f$. The function f must be evaluated to be within $\pm\frac{ml}{2\pi N}\theta$ for the phase to be accurate to within $\pm\theta$.If the gradient is desired to n bits of precision, the function must be evaluated to:

$$n_0 = log_2\left(\frac{max(f) - min(f)}{(ml/2^n)(\theta/2\pi)}\right) \tag{2.49}$$

$$= n_{classical} + log_2\left(\frac{2\pi}{\theta}\right) \tag{2.50}$$

qubits of precision.

It can be shown that the inner product between the ideal state and the actual state after the inverse Fourier transform is atleast $cos^2\theta$ if the phase is accurate to within $\pm\theta$. Therefore the success probability of the algorithm is atleast $cos^2\theta$. n_0 differs from the classical required precision by just an additive constant and in the limit of large

n the difference becomes negligible. For example, for $\theta = \pi/8$, the algorithm succeeds 85% of the time and n_0 will exceed the classically required precision by 4 bits.

2.5 Algorithm to solve Linear Equations

2.5.1 Introduction

Recently a quantum algorithm to obtain certain information about the solution of a linear system $A\vec{x} = \vec{b}$ was presented by Aram W. Harrow, Avinatan Hassidim and Seth Lloyd[AWHL09]. The algorithm considers the case where one does not need to know the solution $\vec{x}$ itself, but rather an approximation of the expectation value of some operator associated with $\vec{x}$, e.g.,$\vec{x}^{\dagger}M\vec{x}$ for some matrix M. Such functions can be approximated on a quantum computer in a time which scales logarithmic in N (where N is the total number of variables appearing in the system) and polynomial in the condition number (κ) and desired precision (ϵ). The dependence on N is exponentially better as compared to the classical algorithms while the dependence on the error is worse. The dependence on the condition number is comparable.

2.5.2 Algorithm

In the problem of solving linear equations , we have a main matrix A, a vector of unknown variables x and a vector of free values b, such that $A|x\rangle = |b\rangle$. Here $|x\rangle$ and $|b\rangle$ are normalized quantum states and $|b\rangle$ can be written as $|b\rangle = \sum_{i=1}^{N} b_i|i\rangle$. Matrix A is assumed to be $N \times N$ Hermition matrix with spectral decomposition $\sum_{j=1}^{N} \lambda_j|u_j\rangle\langle u_j|$. If A is not a Hermition matrix , we can build a Hermition matrix $\widehat{A}$ out of the original matrix A as :

$$\widehat{A} = \begin{pmatrix} 0 & A \\ A^{\dagger} & 0 \end{pmatrix} \tag{2.51}$$

and then solve for the following equation instead:

$$\widehat{A}\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix} \tag{2.52}$$

where

$$\vec{y} = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} \tag{2.53}$$

For the rest of the report we will simply assume that A is Hermition.

The state $|b\rangle$ and the unknown vector $|x\rangle$ representing the solution are represented in the eigen-basis of matrix A. We have:

$$|b\rangle = \sum_{j=0}^{N-1} \beta_j |u_j\rangle, \tag{2.54}$$

$$|x\rangle = \frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|} \tag{2.55}$$

where $\beta_j = \langle u_j|b\rangle$

The condition number κ, is defined as the ratio between A's largest and smallest eigenvalues.The κ term is a measure as to how difficult the matrix A is to invert. For large κ A becomes closer to a matrix which can not be inverted.

The algorithm can be broadly divided into five steps: Encoding vector $\vec{b}$ into quantum state $|b\rangle$, Phase estimation subroutine, Non unitary operation on an ancillary qubit (implemented by controlled rotation), Reverse phase estimation and finally, measurement of an ancillary qubit.

State preparation of $|b\rangle$

For the algorithm to be efficient, one needs an efficient procedure to prepare state $|b\rangle$. Currently there is no explicit method known to efficiently prepare a arbitrary state for general cases. It is considered to be a very hard problem [YCK13]. To prepare the state $|b\rangle$ from state $|0....0\rangle$, one needs a unitary U such that $U|0....0\rangle = |b\rangle$. As an arbitrary unitary operation can be decomposed into a product of elementary quantum gates (single qubit rotations and CNOT gate),in general one needs O (poly(N)) elementary gates to prepare an arbitrary N-dimensional quantum state .

Regardless of the above stated fact, there are certain type of states which can be efficiently prepared. Few notable examples are given in [GR02] [SS]. For the rest of the report, we assume that certain techniques can be used to efficiently prepare the $|b\rangle$ state. Also the algorithm could be used a subroutine in a larger quantum algorithm of which some other component is responsible for producing $|b\rangle$.

Phase estimation

In this step, the technique of phase estimation is used to move the eigenvalues of the matrix A into a quantum register and the vector $|b\rangle$ is decomposed in the eigenvector

basis of A (equivalently $e^{\iota At}$). Conditional Hamiltonian simulation of A is applied to the $|b\rangle$ state controlled by some register $|\psi_0\rangle$ which is defined as:

$$|\psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} sin \frac{\pi(\tau + \frac{1}{2})}{T} |\tau\rangle \tag{2.56}$$

where T is some large number. Conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{\frac{\iota A \tau t_0}{T}}$ is applied on $|\psi\rangle \otimes |b\rangle$ (where $t_0 = O(\kappa/\epsilon)$ (ϵ is the error)) and the inverse Fourier transform on the first register is done, which in turn gives the state:

$$\sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j} \beta_j |k\rangle |u_j\rangle \tag{2.57}$$

$|k\rangle$ are the Fourier basis states here.

The coefficient $\alpha_{k|j}$ is a complex number (Please see appendix for detailed mathematical derivation) and its magnitude $|\alpha_{k|j}|$ is large if and only if $\lambda_j \approx \frac{2\pi k}{t_0}$. By defining $\tilde{\lambda} := \frac{2\pi k}{t_0}$, the register $|k\rangle$ is relabelled to get:

$$\sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j} \beta_j |\tilde{\lambda}_k\rangle |u_j\rangle \tag{2.58}$$

For perfect phase estimation,$\alpha_{k|j} = 1$ if $\tilde{\lambda}_k = \lambda_j$ and zero otherwise. Assuming this for now, we obtain:

$$|\psi_1\rangle = \sum_{j=1}^{N} \beta_j |\lambda_j\rangle |u_j\rangle \tag{2.59}$$

Thus the first register in the above equation now contains the eigenvalues of the matrix A.

Controlled Rotation

In this step an ancilla qubit initialized at $|0\rangle$ state is introduced and $|\lambda_j\rangle$ state is used to perform a controlled Y-rotation $R_y(\theta_j) = e^{\frac{-\iota\theta_j Y}{2}}$ (Y is the Pauli Y operator) onto the ancilla qubit such that the state of the system is brought to:

$$|\psi_2\rangle = \sum_{j=1}^{N} \beta_j |\lambda_j\rangle |u_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} + \frac{C}{\lambda_j} |1\rangle \right) \tag{2.60}$$

with the rotation angles $\theta_j = 2arcsin(\frac{C}{\lambda_j})$. C is a constant satisfying the condition $C \leq min_j|\lambda_j| = O(1/\kappa)$.

Reverse Phase Estimation

In this step,the inverse of the phase estimation subroutine in the beginning is applied to effectively uncompute the eigenvalues in the first register. After undoing the phase estimation we get:

$$|\psi_3\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} + \frac{C}{\lambda_j}|1\rangle \right) \tag{2.61}$$

Measurement

This is the last step in which the ancilla qubit is measured . If the outcome is 1, the state collapses to the state:

$$|\psi_4\rangle = \sqrt{\frac{1}{\sum_{j=1}^{N} C^2|\beta_j|^2/|\lambda_j|^2}} \sum_{j=1}^{N} \beta_j \frac{C}{\lambda_j}|u_j\rangle \tag{2.62}$$

which is nothing but $|x\rangle = \sum_{j=1}^{N} \beta_j \lambda^{-1}|u_j\rangle$ upto a normalization factor. The probability of outcome to be 1 is $\sum_j |\beta_j|^2 . |\frac{C}{\lambda_j}|^2$ which scales as $O(\frac{1}{\kappa^2})$. If the outcome is 0 , the algorithm has failed and is to be repeated again.

Finally a measurement M is done whose expectation value $\langle x|M|x\rangle$ corresponds to the feature of $\vec{x}$ we wanted to evaluate.

2.5.3 Circuit

The circuit diagram for solving linear equations is given below [YCK]:

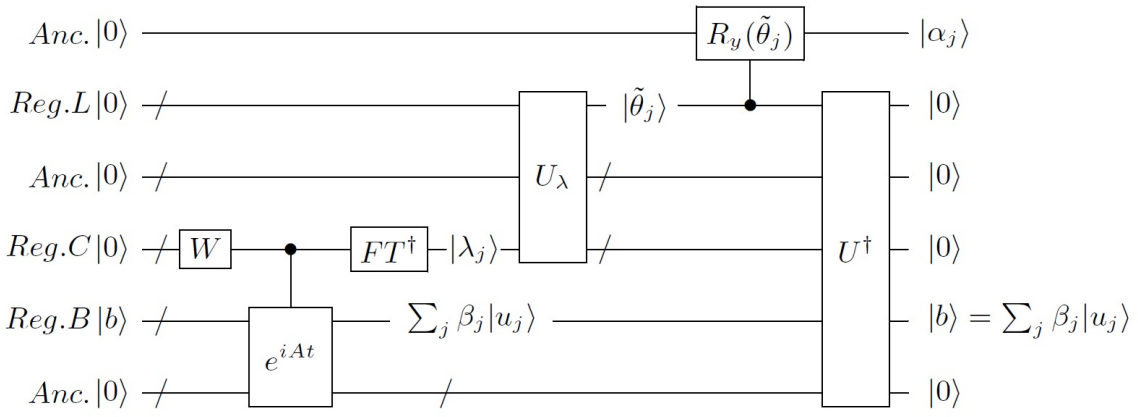


Figure 2.4: Quantum Circuit for solving the linear System$A\vec{x} = \vec{b}$

In the above figure, W represents Hadamard transform, , FT represents Fourier Transform, U_λ represents the subroutine to compute the state $|\tilde{\theta}_j\rangle$ with $\tilde{\theta}_j$ =2arcsin$\frac{C}{\lambda_j}$ for each $|\lambda_j\rangle$.

2.6 Algorithm to solve Non Linear Differential Equations

2.6.1 Introduction

A quantum algorithm was proposed by Sarah K. Leyton and Tobias J. Osborne to solve system of non-linear differential equations whose non-linear terms are polynomials [LO08]. The resource requirement for the algorithm are polylogarithmic in the number of variables and exponential in the number of time-steps over which to perform the simulation. The algorithm provide an exponential improvement over the classical algorithms which run in time scaling linear with number of variables.

The algorithm is concerned with n first order ordinary differential equations (ODEs) :

$$\begin{aligned} \frac{dz_1(t)}{dt} &= f_1(z_1(t), z_2(t), ..., z_n(t)) \\ \frac{dz_2(t)}{dt} &= f_2(z_1(t), z_2(t), ..., z_n(t)) \\ &... \\ &... \end{aligned} \tag{2.63}$$

$$\frac{dz_n(t)}{dt} = f_n(z_1(t), z_2(t), ..., z_n(t))$$

with boundary condition $\boldsymbol{z}(0) = \boldsymbol{b}$. The non-linear terms are given by n polynomials $f_\alpha(\boldsymbol{z})$ in n variables z_j, $j = 1, 2, ..., n$. The variables $z_j(t)$ are encoded in the amplitudes of a quantum state $|\phi\rangle$:

$$|\phi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}\sum_{j=1}^{n} z_j|j\rangle \tag{2.64}$$

where $\sum_{j=1}^{n} |z_j|^2 = 1$ ensures that the state is normalized. The state $|\phi\rangle$ can be encoded on a quantum computer using $\log(n)$ qubits. It is assumed that the initial state $|\phi\rangle$ can be efficiently prepared using certain techniques/methods.

2.6.2 Algorithm

The algorithm consists of two subroutines:

- Quantum Algorithm to effect a non-linear transformation of the amplitudes
- Quantum Implementation of Euler's Method

Non-linear Transformation of the Amplitudes of a Quantum State

It is a non-deterministic algorithm to prepare a quantum state whose amplitudes are non-linear function of the amplitudes of a input quantum state. Algorithm describes the quadratic systems only. The extension to the higher degrees is a straightforward task.

Suppose we have two copies of $|\phi\rangle$. The product state is given by:

$$|\phi\rangle|\phi\rangle = \frac{1}{2}\sum_{j,k=0}^{n} z_j z_k|jk\rangle \tag{2.65}$$

where $z_0 = 1$.In this expansion, every monomial $z_j^{l_j} z_k^{l_k}$, $l_j, l_k \leq 1$, of degree less than 2 appears (more than once).

We want to iterate the following transformation:

$$\boldsymbol{z} \longrightarrow F(\boldsymbol{z}) \tag{2.66}$$

where:

$$F(\boldsymbol{z}) = \begin{pmatrix} f_1(\boldsymbol{z}) \\ f_2(\boldsymbol{z}) \\ .. \\ .. \\ f_n(\boldsymbol{z}) \end{pmatrix} \tag{2.67}$$

Here f_α (where $\alpha = 1, 2, ..., n$) are quadratic polynomials:

$$f_\alpha(\boldsymbol{z}) = \sum_{k,l=0}^{n} a_{kl}^{(\alpha)} z_k z_l \tag{2.68}$$

with $a_{kl}^{(\alpha)} = a_{lk}^{(\alpha)}$ and $f_0(\boldsymbol{z}) = 1$. The main aim is to prepare the quantum state:

$$|\phi'\rangle = \frac{1}{\sqrt{2}} \sum_{\alpha=0}^{n} f_\alpha(\boldsymbol{z})|\alpha\rangle \tag{2.69}$$

where for simplification, it is assumed that the transformation is measure preserving, i.e.,

$$\sum_{j=1}^{n} |z_j|^2 = \sum_{\alpha=1}^{n} f_\alpha^*(\boldsymbol{z}) f_\alpha(\boldsymbol{z}) = 1 \tag{2.70}$$

Beyond the measure preserving assumption, there are several other assumptions which are made in order to ensure the efficiency of the method. These assumptions are:

- The coefficient in the expansion of function is of the order of one , i.e.,

$$|a_{kl}^{\alpha}| = O(1), \qquad k, l, \alpha = 1, 2, ..., n. \tag{2.71}$$

- Map F is sparse. Each $f_\alpha(\boldsymbol{z})$ can involve at most $s/2$ monomials and each variable can appear in at most $s/2$ polynomials $f_\alpha(\boldsymbol{z})$ i.e., :

$$|\{(k,l)|a_{kl}^{(\alpha)} \neq 0\}| \leq s/2, \qquad \alpha = 1, 2, ..., n \tag{2.72}$$

$$|\{\alpha|a_{kl}^{(\alpha)} \neq 0\}| \leq s/2, \qquad k, l = 1, 2, ..., n \tag{2.73}$$

- Lipschitz constant in our system is of the order of one, i.e.. the number λ such that:

$$||F(\boldsymbol{x} - \boldsymbol{y})|| \leq \lambda ||(\boldsymbol{x} - \boldsymbol{y})||$$

in the ball $||\boldsymbol{x}||^2 \leq 1$ and $||\boldsymbol{y}||^2 \leq 1$, is $O(1)$.

Now consider an operator A:

$$A = \sum_{\alpha,k,l,=0}^{n} a_{kl}^{\alpha} |\alpha 0\rangle\langle kl| \tag{2.74}$$

Hamiltonian is set up using operator A and adjoining a qubit "pointer" P :

$$H = -\iota A \otimes |1\rangle_P\langle 0| + \iota A^{\dagger} \otimes |0\rangle_P\langle 1| \tag{2.75}$$

System is initialized in the state:

$$|\psi_0\rangle = |\phi\rangle|\phi\rangle|0\rangle_P \tag{2.76}$$

Hamiltonian simulation is done for the time $t = \epsilon$. The evolution time depends on the the sparsity of H and the desired error. After the evolution , state of the system is:

$$|\psi_1\rangle = e^{\iota\epsilon H}|\phi\rangle|\phi\rangle|0\rangle \tag{2.77}$$

$$= \sum_{j=0}^{\infty} \frac{(\iota\epsilon H)^j}{j!}|\phi\rangle|\phi\rangle|0\rangle \tag{2.78}$$

$$= |\phi\rangle|\phi\rangle|0\rangle + \epsilon A|\phi\rangle|\phi\rangle|1\rangle - ... \tag{2.79}$$

As we know:

$$A|\phi\rangle|\phi\rangle = \frac{1}{2}\sum_{\alpha,k,l=0}^{n} a_{kl}^{(\alpha)} z_k z_l |\alpha\rangle|0\rangle \tag{2.80}$$

$$= \frac{1}{\sqrt{2}}|\phi'\rangle|0\rangle \tag{2.81}$$

Thus the state of the system after evolution is:

$$|\psi_1\rangle = |\phi\rangle|\phi\rangle|0\rangle + \frac{\epsilon}{\sqrt{2}}|\phi\rangle|0\rangle|0\rangle - ... \tag{2.82}$$

Now a measurement is made on the ancilla qubit. If the outcome is 0 , then the algorithm has failed and the resulting state is discarded. If the outcome is 1, we get our desired state. The probability of the outcome to be $1 \approx \frac{1}{2}\epsilon^2$ and the state of the

computational register in this case collapses to:

$$|\psi_2\rangle \approx |\phi'\rangle|0\rangle \tag{2.83}$$

Quantum Implementation of Euler's Method

To integrate the equations described in (2.63) with initial condition $z(\mathbf{0}) = \boldsymbol{b}$, Euler's method is used.A small step size is picked to iterate the map:

$$z_j \longrightarrow z_j + hz'_j = z_j + hf_j(\boldsymbol{z}) \tag{2.84}$$

Given a state $|\phi(t)\rangle$, we aim to prepare:

$$|\phi(t+h)\rangle = |\phi(t)\rangle + h|\phi'(t)\rangle + O(h^2) \tag{2.85}$$

where:

$$|\phi'(t)\rangle = \frac{1}{\sqrt{2}}\sum_{\alpha=0}^{n} f_\alpha(\boldsymbol{z}(t))|j\rangle = \frac{1}{\sqrt{2}}\sum_{\alpha,k,l=0}^{n} a_{kl}^{(\alpha)} z_k(t) z_l(t)|\alpha\rangle \tag{2.86}$$

To implement the above transformation, we assumed that we have two copies of $|\phi(t)\rangle$ and then the method of the previous section is used to implement the polynomial transformation $z_j \longrightarrow z_\alpha + hf_\alpha(\boldsymbol{z}(t))$.

2.6.3 Computational Resources

The probability to get the desired state $|\phi'\rangle$ after the measurement step of the first section is $\approx \frac{1}{2}\epsilon^2$. To ensure high success probability , the process should be repeated on roughly $\frac{16}{\epsilon^2}$ fresh $|\phi\rangle|\phi\rangle$ pairs. The success probability will be 80% in this case.

To integrate the set of equations in (2.63) using Euler's method, time is discretized into m steps(Step size = t/m). Thus we need to start with $\left(\frac{16}{\epsilon^2}\right)^m$ copies of initial state $|\phi(\mathbf{0})\rangle$. Now each state require $\log(n)$ qubits. Thus the total spatial resources required by the algorithm scale as $\left(\frac{16}{\epsilon^2}\right)^m \log(n)$.

2.6.4 Extension to Cubic or Higher Systems

In the algorithm, only quadratic systems were explained. The extension to the system with cubic or higher non-linearity is very simple. To implement the cubic transformation (or higher transformations) one has to start with three (or more for higher

transformations) copies of state $|\phi\rangle$ and rest of the procedure will be same.

2.7 Summary

- In this chapter , we have discussed three quantum algorithms: algorithm for numerical gradient estimation, to solve linear system of equations and to solve non-linear differential equations. All these algorithms provide a speed up over their classical counterparts.

- The initial state is assumed to be given or efficiently preparable in the last two algorithms. In practice it is hard problem to produce these initial states, however in certain cases these states can be efficiently prepared.

- The algorithm to solve non-linear differential equations has a very poor scaling in time. The resources consumed by the algorithm scale exponentially with the inverse step size and the integration time. The dependence on degree of nonlinearity is also exponential as well. So it will be hard to see a near future experimental implementation of the algorithm without any modification on a quantum computer such as NMR system where the number of qubits are limited.

- The algorithm for numerical gradient estimation can't be run recursively to efficiently obtain higher derivatives as there is a phase factor linked to the output state. The technique for eliminating the global phase would require one more black box query. This additional query necessitates 2^n queries for evaluation of nth partial derivative.

- The algorithm to solve linear system of equations (Harrow's algorithm) provides an exponential speedup over classical algorithms when the condition number κ is a poly-logarithmic function of N, while the speeed up is only polynomial when κ is polynomial in N.

- In a slightly modified form, Harrow's algorithm was experimentally implemented on a NMR quantum computer using 4-qubits [JP14]. In the experimental realization, eigenvalue inversion subroutine was simplified in order to keep a check on the number of qubits used. The matrix A in the experimental realization was a simple 2×2 matrix and its eigenvalues were power of 2.

- In Harrow's algorithm, non-square matrices can also be inverted using the procedure to go from non-Hermition case to the Hermition one. The algorithm has

many potent applications such as to solve Poisson equation. One has to first discretize the equation and then algorithm can be used to solve the resulting set of linear equations.

Chapter 3

Quantum Simulation of Quantum Tunneling

"Nature isnt classical, dammit, and if you want to make a simulation of nature, youd better make it quantum mechanical, and by golly its a wonderful problem, because it doesnt look so easy."

-Richard P. Feynman

3.1 Quantum Simulation

Simulation of physical systems is one of the most important practical application of computation. As far as simulation of a quantum system is concerned, the exponential increase of the Hilbert space with the system size forbids its efficient simulation on a classical computer. In 1981, Feynman delivered his famous visionary lecture "Simulating Physics with computers" where he suggested the use of quantum system to simulate the behavior of a another quantum system [Fey09]. The level of efficiency reached in this case is way beyond the capability of a classical computer. In a quantum simulation, one has an access to a controllable quantum system whose Hamiltonian can be easily manipulated. NMR information processor is an example of such a system. This controllable quantum system is then used to investigate the behavior and properties of another less accessible or controllable quantum. Much progress has been made in the field in last few years and Quantum simulation has been realized in various situations [Tra12].

A simulator is a physical device that reveal information about a mathematical function which represents a physical model. There are two types of simulators [THJJ14]

Digital Simulator:

In a digital quantum simulation, the universal unitary time evolution is replicated via Trotter decompositions, to a circuit which can then be made arbitrarily accurate. In other words, the whole time evolution of system is decomposed into a sequence of simple local operations. In 1996, Lloyd showed that a quantum computer based on this principle, can indeed act as a universal quantum simulator [S.L92]. By universality it is meant that except for changes in the programs, the same machine can tackle vastly different problems and the accuracy with which a model is simulated can be arbitrarily controlled.

Analogue Simulator:

Analogue simulators are devices whose Hamiltonians can be can be easily controlled and tuned as desired. To perform quantum simulation, controlling parameters are tuned to achieve a Hamiltonian which is equivalent to the Hamiltonian of model system under study. Hilbert space of model system is directly mapped onto the Hilbert space of simulator. Once the system is initialized in a desired state, the evolution of state in simulator directly corresponds to the evolution of model system. These type of simulators can act as a universal simulator, but rather are designed to explore specific problems in quantum physics. Advantages of a analogue quantum simulator is that it is much easier to implement as compared to a digital simulator.

3.2 Theoretical protocol for quantum simulation of quantum tunneling

Quantum tunneling is a quantum mechanical phenomena where a particle tunnels through a barrier which can't be surmounted classically. It plays an essential role in many important quantum phenomena and is widely used in modern devices such as tunnel diode, scanning tunneling microscope and so on. Simulation of quantum

tunneling is of great significance as it is a unique fundamental concept in quantum mechanics. In this section, the digital simulation protocol for tunneling is presented [Sor][GF].

The Schrodinger equation for a particle moving in one dimension can be written as:

$$\iota\frac{\partial}{\partial t}\left|\psi(x,t)\right\rangle = \left[\frac{\widehat{P}^2}{2m} + V(\widehat{X})\right]\left|\psi(x,t)\right\rangle \tag{3.1}$$

where $\widehat{X}$ and $\widehat{P}$ are position and momentum operators respectively. Here $\hbar$ is considered to be unity. If the Hamiltonian is independent of time, then the evolution of wave function with time can be straightforwardly written as:

$$\left|\psi(x,t+\triangle t)\right\rangle = e^{-\iota\left[\frac{\widehat{P}^2}{2m}+V(\widehat{X})\right]\triangle t}\left|\psi(x,t)\right\rangle \tag{3.2}$$

The standard digital quantum simulation encodes the continuous coordinate x in $n = \log_2 N$ qubits, where N is the number of discretized particle locations [Zal98]. Suppose the wave function is continuous in the region $0 < x < L$ and have a periodic boundary condition $\psi(x+L,t) = \psi(x,t)$. The wave function is stored in the n qubit quantum register as:

$$\left|\psi(x,t)\right\rangle \longrightarrow \sum_{k=0}^{2^n-1}\psi(x_k,t)\left|k\right\rangle \tag{3.3}$$

where $x_k = \left(k+\frac{1}{2}\right)\triangle l$ and $\triangle l = \frac{L}{2^n}$. $\left|k\right\rangle$ is the lattice basis vector corresponding to the binary representation of number k. Potential operator $V(\widehat{X})$ is a function of the coordinate operator $\widehat{X}$. Thus it is a diagonal matrix in the coordinate representation. It's construction for the different cases are described in the last paragraph of this section.

Similarly, the kinetic energy operator is a diagonal matrix in momentum representation. It is constructed in the coordinate representation with the help of a quantum Fourier transformation (QFT). Suppose $\phi(p,t)$ is the wave function in momentum representation. Similar to equation(3.3), we get:

$$\left|\phi(p,t)\right\rangle \longrightarrow \sum_{j=0}^{2^n-1}\phi(p_j,t)\left|j\right\rangle \tag{3.4}$$

where p_js are the eigenvalues of the momentum operator $\widehat{P}_P$ in the momentum representation:

$$p_j = \left\{ \begin{array}{ll} \frac{2\pi j}{2^n} & \text{for } 0 \leq j \leq 2^{n-1} \\ \frac{2\pi\left(2^{n-1}-j\right)}{2^n} & \text{for } 2^{n-1} \leq j \leq 2^n \end{array} \right\}. \tag{3.5}$$

$\widehat{P}_P$ can be thus written as:

$$\widehat{P}_P = \sum_{j=0}^{2^{n-1}} \frac{2\pi j}{2^n} j|j\rangle\langle j| + \sum_{j=2^{n-1}+1}^{2^n-1} \frac{2\pi}{2^n}\left(2^{n-1}-j\right)|j\rangle\langle j| \tag{3.6}$$

The kinetic energy operator in the coordinate representation is calculated using QFT as:

$$\frac{\widehat{P}^2}{2m} = \mathbf{F}^{-1}\frac{\widehat{P}_P^2}{2m}\mathbf{F} \tag{3.7}$$

where $\mathbf{F}$ is the Fourier Transformation Operator which can be implemented in a quantum circuit via a series of Hadamard gates and controlled phase gates [Cop94].

With the expressions for discretized forms of the wave function, the potential operator and the kinetic energy operator in hand, time evolution of system within a small interval $\triangle t$ can be efficiently implemented using the Trotter formulae, which is correct upto $\triangle t$. Equation (3.2) can be then approximated as:

$$|\psi(x, t+\triangle t)\rangle = e^{-\iota V(\widehat{X})\triangle t} e^{-\iota\frac{\widehat{P}^2}{2m}\triangle t}|\psi(x,t)\rangle \tag{3.8}$$

Equation(3.7) can be used to calculate the first exponential term in above equation:

$$\begin{aligned} e^{-\iota\frac{\widehat{P}^2}{2m}\triangle t} &= \mathbf{F}^{-1} e^{-\iota\frac{\widehat{P}_P^2}{2m}\triangle t}\mathbf{F} \\ &= \mathbf{F}^{-1}\mathbf{D}\mathbf{F} \end{aligned} \tag{3.9}$$

Combining equation(3.3), 3.8 and 3.9 we can write:

$$\sum_{k=0}^{2^n-1}|\psi(x_k, t+\triangle t)\rangle = \mathbf{F}^{-1}\mathbf{D}\mathbf{F}\mathbf{Q}\sum_{k=0}^{2^n-1}|\psi(x_k,t)\rangle \tag{3.10}$$

where $\mathbf{Q} = e^{-\iota V(\widehat{X})\triangle t}$. This gives us the quantum circuit for the one time step of the evolution:

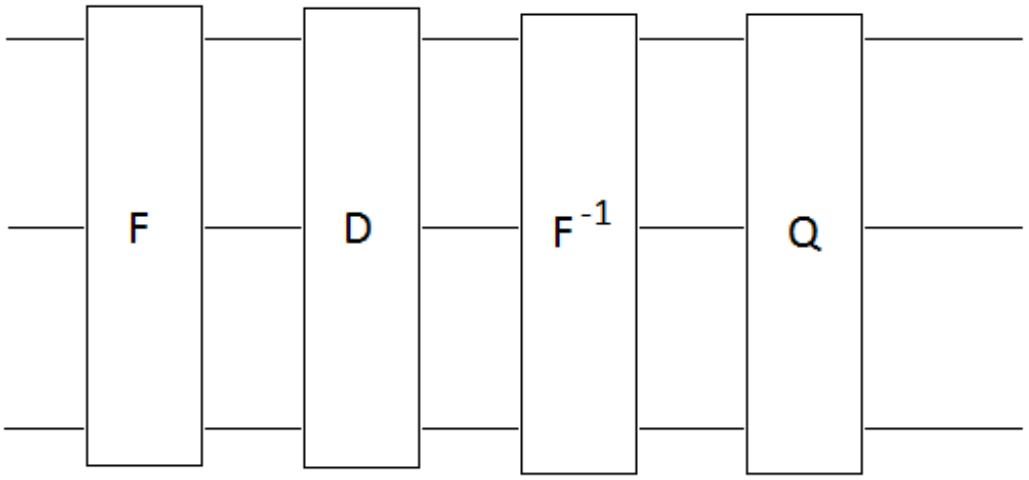


Figure 3.1: Quantum Circuit for one time step of the simulation.

POTENTIAL OPERATOR :

The theoretical protocol developed here is not for some specific case, but is applicable to all the systems in one dimension. By changing the structure of gate **Q**, different kind of potentials can be implemented. A square- well potential can be implemented with a single-qubit operator. A Z-rotation of the the highest order qubit can be written as:

$$e^{-\iota \mathbf{V} \triangle t} = e^{-\iota v \sigma_z \triangle t} \otimes I \otimes I ...$$

where σ_z is the z-matrix, $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and v is a parameter. The operator $e^{-\iota \mathbf{V} \triangle t} = \begin{pmatrix} e^{-\iota v \triangle t} & 0 \\ 0 & e^{\iota v \triangle t} \end{pmatrix} \otimes I \otimes I ...$ rotates the qubit states with $|0\rangle$ highest order qubit, with a positive phase velocity v. Similarly it rotates the qubit states with $|1\rangle$ highest order qubit, with a negative phase velocity v. Thus relative to the state in the second half of all the quantum states in the register, states in first half get a phase velocity of $+2v$. If the $z-$ rotation is implemented on second qubit instead of first, then a double well potential is implemented. Similarly if it is acted upon the last qubit, a Dirac-comb like potential will be implemented. To get the delta potential like structure, a negative phase velocity is given to the only state where the potential barrier is desired.

3.3 Results and Discussion

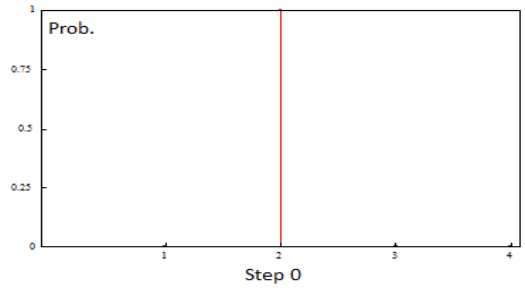
3.3.1 Double Well Potential

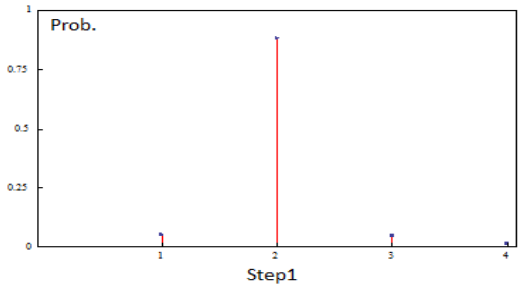
As explained in the previous section, simulations for double well potential are done by implementing Z-rotation on the second qubit. The mass of particle 'm' and height of potential barrier 'v' are taken to be 0.5 and 10 respectively. The evolution is calculated for ten steps with a time step of 0.1.

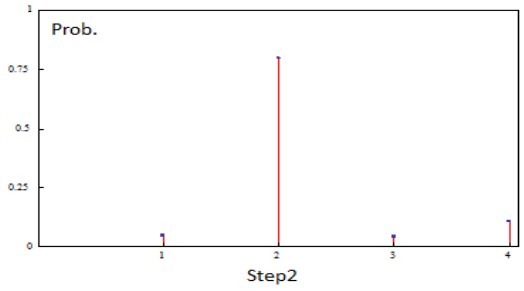
(Programming codes are given in the appendix. While writing codes, we have used the package QDENSITY [BD06].)

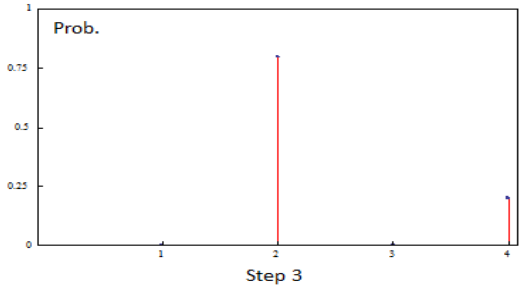
Two Qubit Case

The four basis states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ represent the four lattice sites 1,2,3,4 as shown in the graph of two qubit case. The two potential wells are located at position '2' and position '4' marked in the graph.









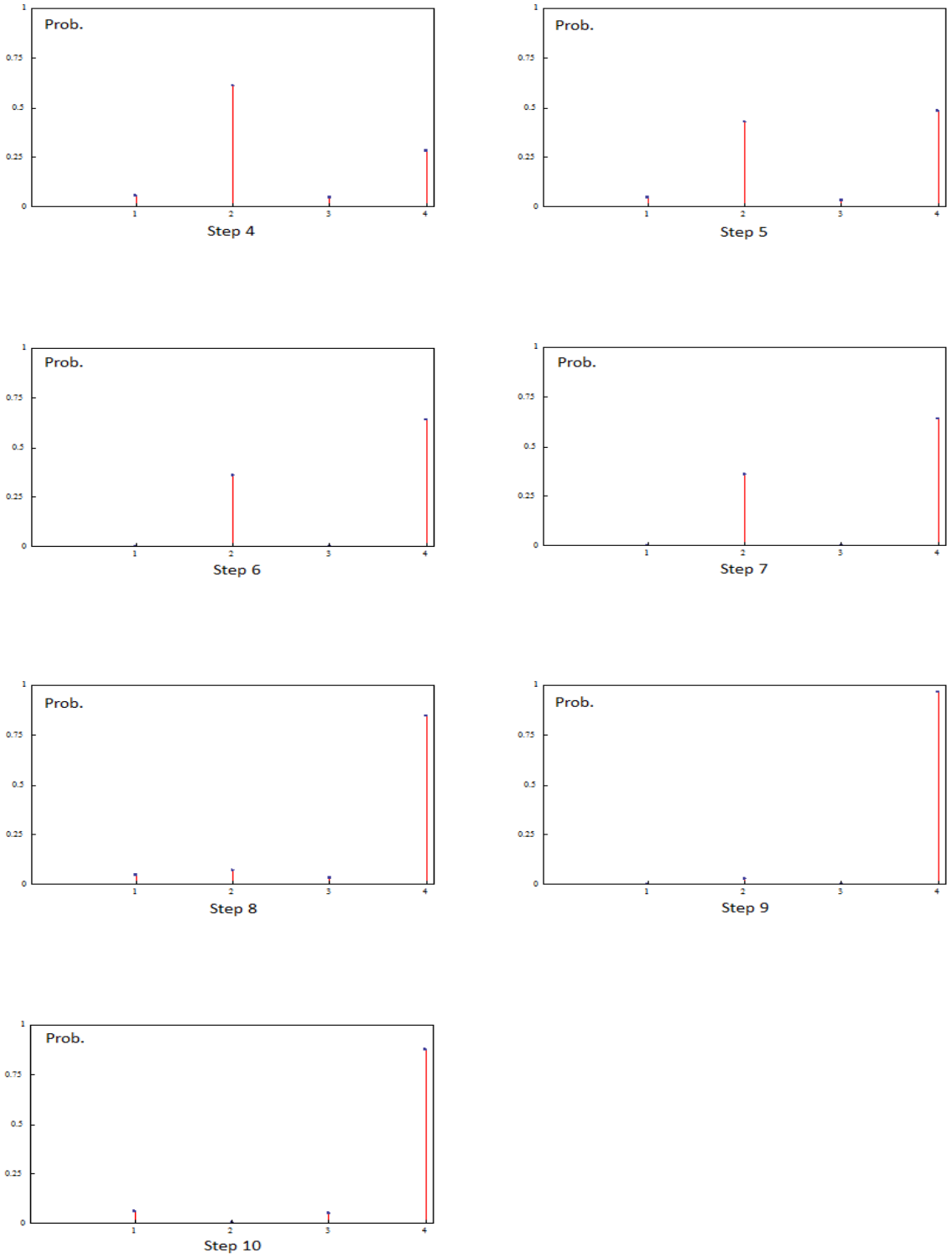


Figure 3.2: Probability distribution of the particle in a double well potential as a function of time for the first ten steps of two qubit simulation.

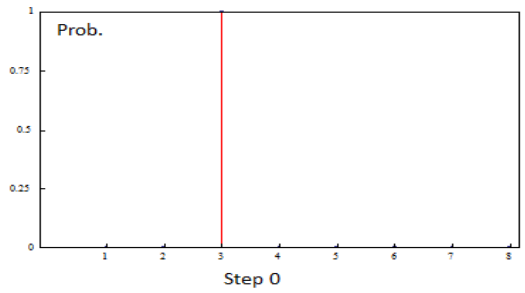
Numerical values for the probabilities shown in the Figure[3.2] are given in the table below:

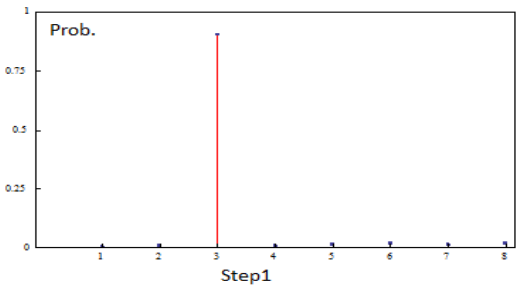
	Position 1	Position 2	Position 3	Position 4
Time Step 0	0.0000	1.0000	0.0000	0.0000
Time Step 1	0.0517	0.8836	0.0484	0.0163
Time Step 2	0.0502	0.7994	0.0434	0.1069
Time Step 3	0.0002	0.7987	0.0003	0.2009
Time Step 4	0.0583	0.6123	0.0477	0.2817
Time Step 5	0.0503	0.4272	0.0366	0.4859
Time Step 6	0.0006	0.3570	0.0012	0.6413
Time Step 7	0.0613	0.2095	0.0501	0.6791
Time Step 8	0.0466	0.0733	0.0333	0.8468
Time Step 9	0.0014	0.0306	0.0024	0.9656
Time Step 10	0.0611	0.0047	0.0551	0.8791

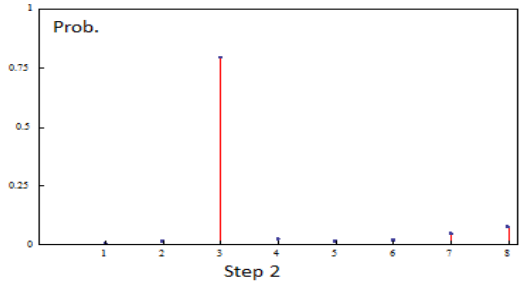
Table 3.1: Probabilities at four positions for different time steps (Double Well Potential- 2 Qubits)

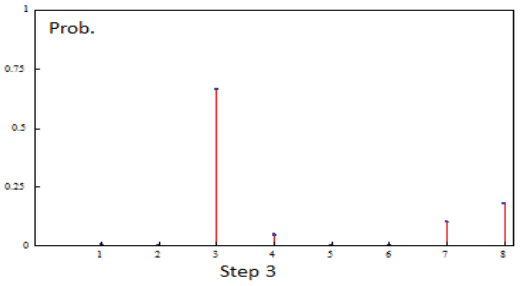
Three Qubit Case

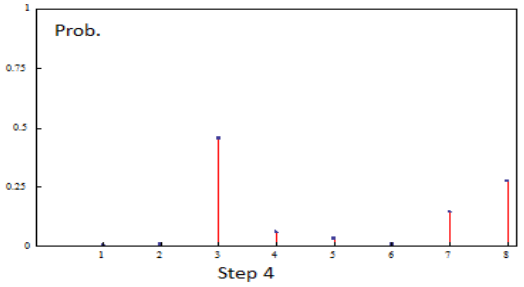
A approach similar to the two qubit case is followed for the three qubit states. The eight basis states: $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$ represent the eight lattice sites 1,2,3,4,5,6,7, 8 respectively. The two wells are located at position 3-4 and position 7-8. All other parameters such as mass, time step and potential height are taken to be same.

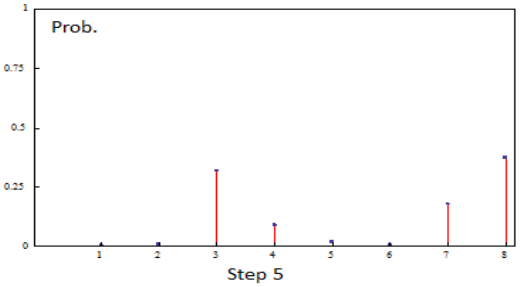


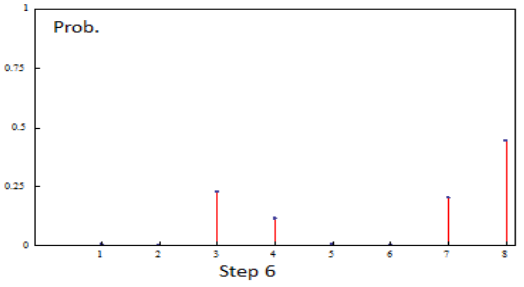


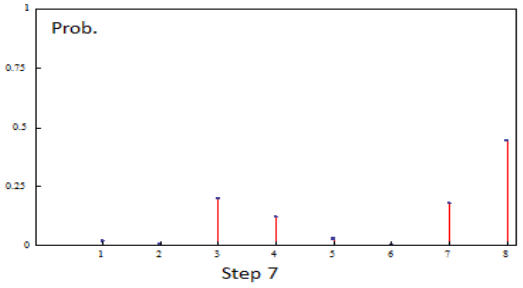

Prob.
Step 2

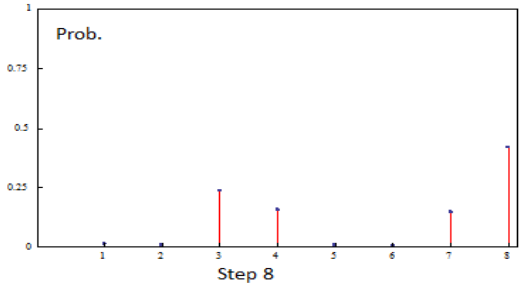

Prob.
Step 3

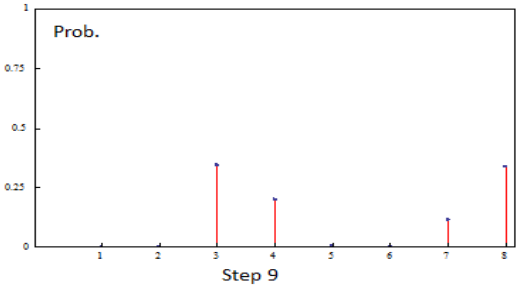

Prob.
Step 4


Prob.
Step 5


Prob.
Step 6


Prob.
Step 7


Prob.
Step 8


Prob.
Step 9

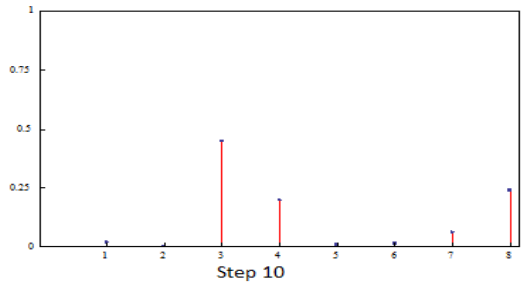


Figure 3.3: Probability distribution of the particle as a function of time for the first ten steps of three qubit simulation.

Numerical values for the probabilities shown in the Figure[3.3] are given in the table below:

	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.0045	0.0117	0.9046	0.0085	0.0150	0.0218	0.0140	0.0199
Time Step 2	0.0048	0.0137	0.7948	0.0270	0.0154	0.0216	0.0474	0.0752
Time Step 3	0.0034	0.0002	0.6667	0.0467	0.0024	0.0003	0.1023	0.1779
Time Step 4	0.0049	0.0093	0.4590	0.0621	0.0349	0.0085	0.1465	0.2749
Time Step 5	0.0058	0.0125	0.3164	0.0904	0.0197	0.0062	0.1778	0.3712
Time Step 6	0.0046	0.0005	0.2272	0.1141	0.0058	0.0002	0.2029	0.4448
Time Step 7	0.0222	0.0061	0.1974	0.1206	0.0299	0.0020	0.1783	0.4435
Time Step 8	0.0175	0.0089	0.2360	0.1576	0.0101	0.0067	0.1467	0.4164
Time Step 9	0.0010	0.0005	0.3424	0.2011	0.0047	0.0006	0.1144	0.3353
Time Step 10	0.0203	0.0026	0.4504	0.1989	0.0085	0.0162	0.0644	0.2387

Table 3.2: Probabilities at eight positions for different time steps (Double Well Potential- 3 Qubits)

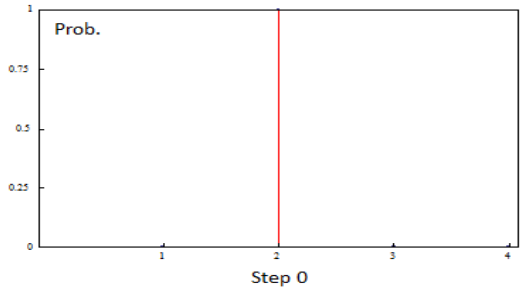
In both two qubit as well as three qubit case, it can be seen that the wave function tunnels from one well to another in a matter of time. In the three qubit case, the resolution is better as the dynamics of particle within the well can also be seen. As expected, the probability of particle on the potential crests is negligible.

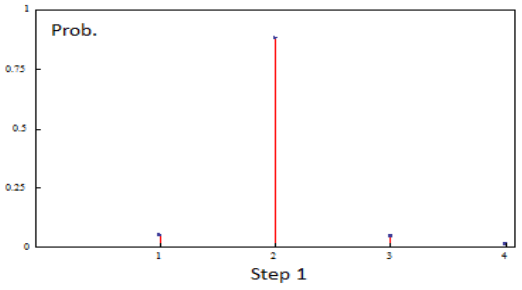
3.3.2 Free Particle

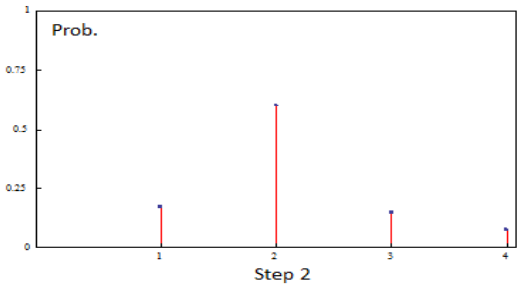
To understand the behavior in a double well potential better, a control simulation is run where the potential is kept zero i.e., $v = 0$. In other words the potential matrix **Q** is equal to identity.

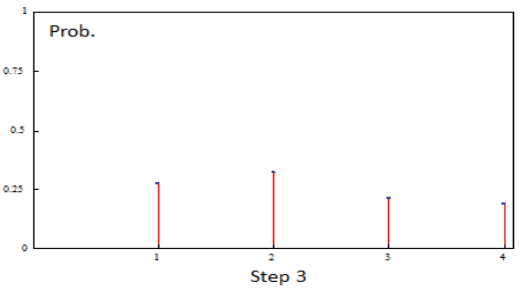
Two qubit case

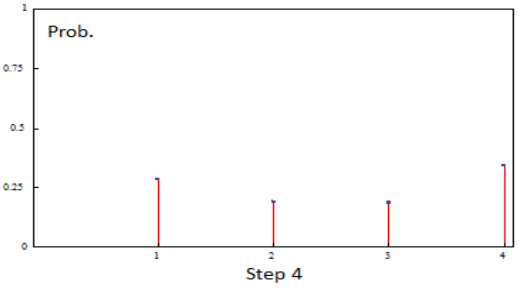
All the parameters such as mass, time step and potential height are taken to be same are same as in the two qubit case of Double well potential.

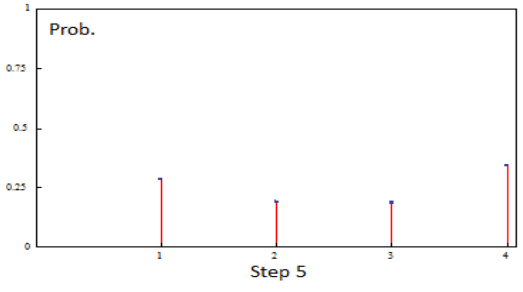












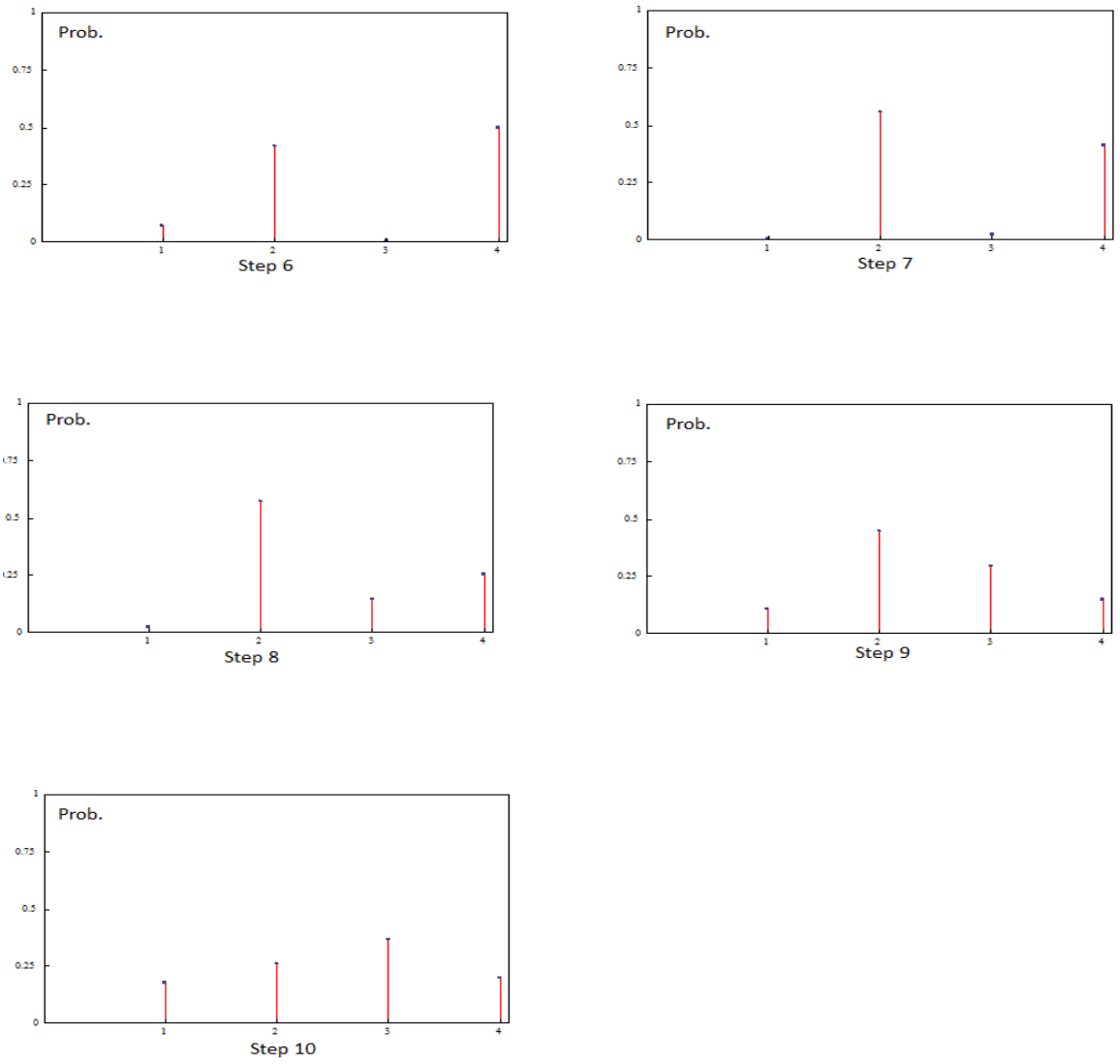


Figure 3.4: Probability distribution of the particle as a function of time for a free particle (Two qubits).

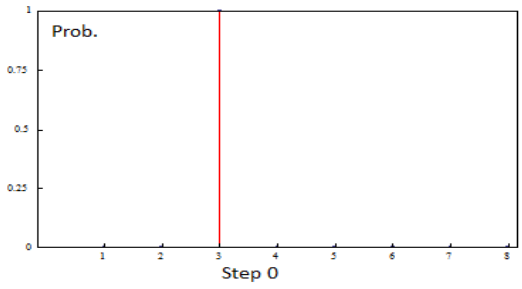
Numerical values for the probabilities shown in the Figure[3.4] are given in the table below:

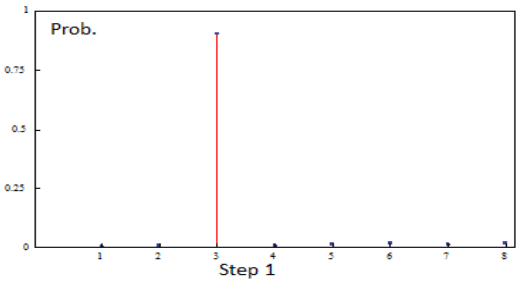
	Position 1	Position 2	Position 3	Position 4
Time Step 0	0.0000	1.0000	0.0000	0.0000
Time Step 1	0.0517	0.8836	0.0484	0.0163
Time Step 2	0.1721	0.6039	0.1488	0.0751
Time Step 3	0.2745	0.3230	0.2132	0.1893
Time Step 4	0.2841	0.1898	0.1854	0.3408
Time Step 5	0.1950	0.2479	0.0882	0.4688
Time Step 6	0.0732	0.4181	0.0077	0.5010
Time Step 7	0.0036	0.5619	0.0240	0.4105
Time Step 8	0.0257	0.5759	0.1450	0.2535
Time Step 9	0.1077	0.4510	0.2939	0.1475
Time Step 10	0.1768	0.2608	0.3656	0.1968

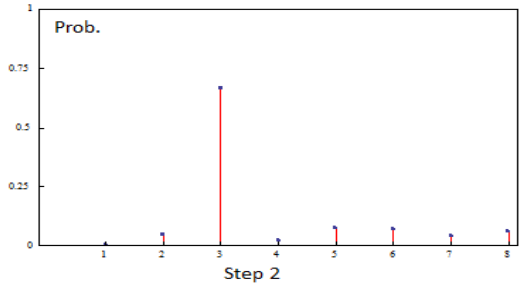
Table 3.3: Probabilities at four positions for different time steps (Free Particle - 2 Qubits)

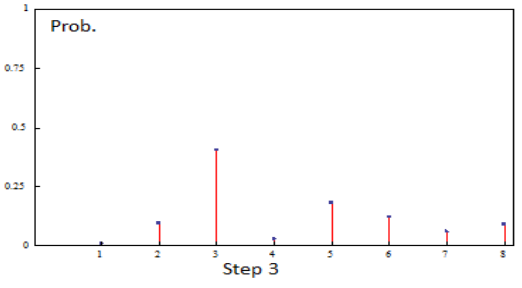
Three qubit case

All the parameters such as mass, time step and potential height are taken to be same are same as in the three qubit case of Double well potential.









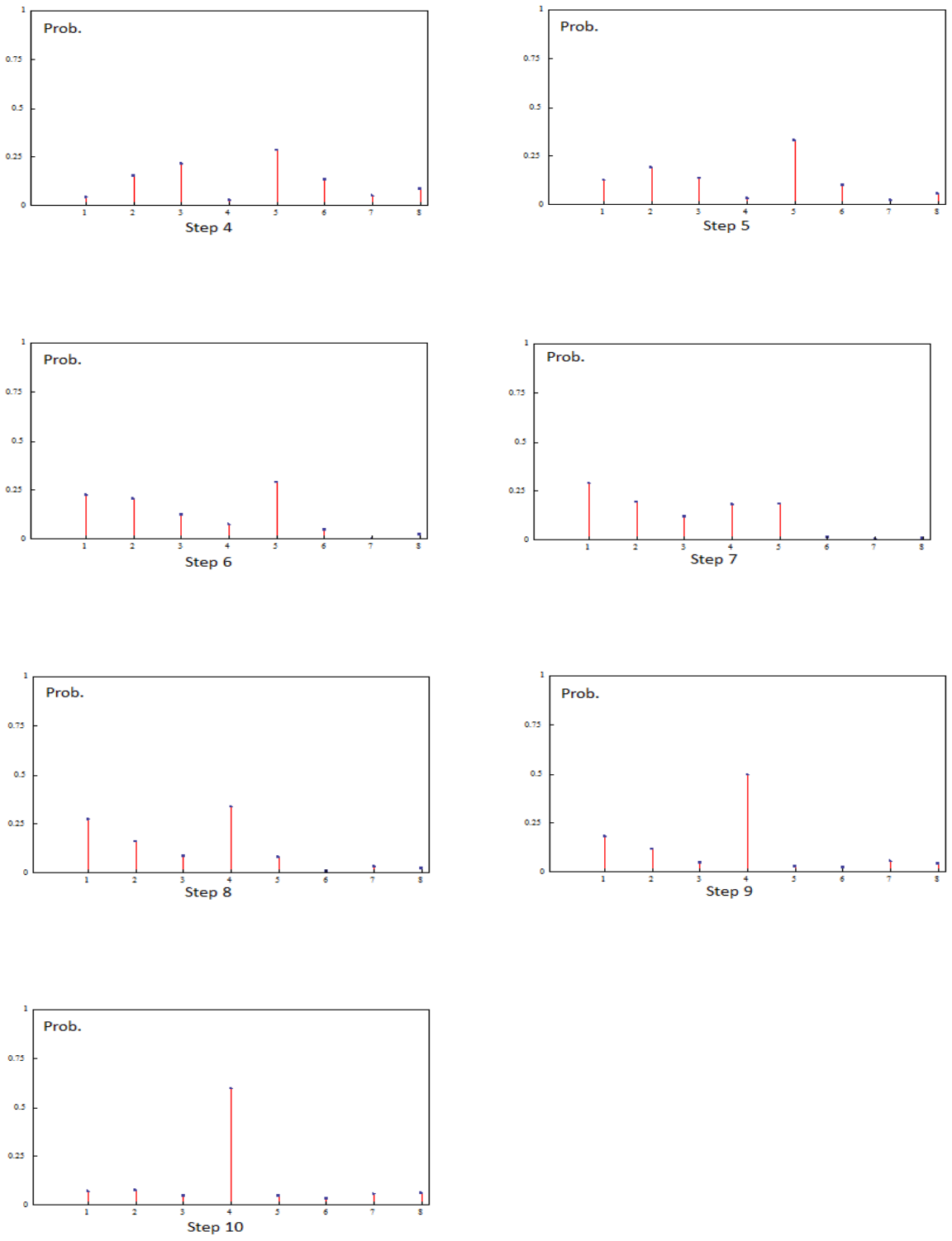


Figure 3.5: Probability distribution of the particle as a function of time for a free particle (Three qubits).

Numerical values for the probabilities shown in the Figure[3.5] are given in the table below:

	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.0045	0.0117	0.9046	0.0085	0.0150	0.0218	0.0140	0.0199
Time Step 2	0.0061	0.0471	0.6677	0.0237	0.0762	0.0742	0.0435	0.0616
Time Step 3	0.0111	0.0987	0.4032	0.0312	0.1803	0.1219	0.0620	0.0916
Time Step 4	0.0462	0.1521	0.2148	0.0286	0.2841	0.1325	0.0529	0.0888
Time Step 5	0.1256	0.1915	0.1356	0.0334	0.3300	0.1007	0.0244	0.0589
Time Step 6	0.2246	0.2062	0.1251	0.0762	0.2888	0.0509	0.0020	0.0261
Time Step 7	0.2885	0.1934	0.1184	0.1800	0.1847	0.0163	0.0059	0.0127
Time Step 8	0.2723	0.1593	0.0846	0.3365	0.0799	0.0119	0.0326	0.0229
Time Step 9	0.1808	0.1162	0.0464	0.4985	0.0305	0.0253	0.0580	0.0442
Time Step 10	0.0724	0.0778	0.0492	0.5997	0.0482	0.0324	0.0595	0.0609

Table 3.4: Probabilities at four positions for different time steps (Free Particle - 3 Qubits)

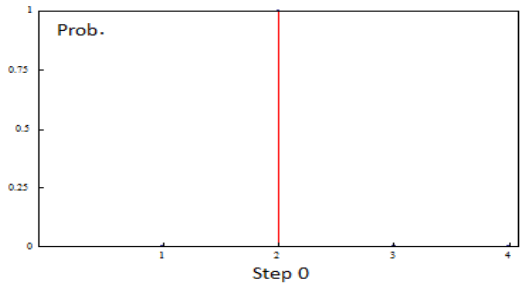
In both the two qubit as well as three qubit case, the results show a remarkable difference between the present scenario where the particle is free to move and the previous scenario where a double well potential was implemented. In the previous case the probability of particle was confined to two wells, while here the probability to find particle at one of the four positions is more random.

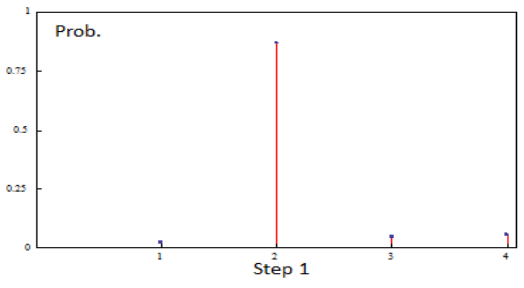
3.3.3 Single potential barrier in the path

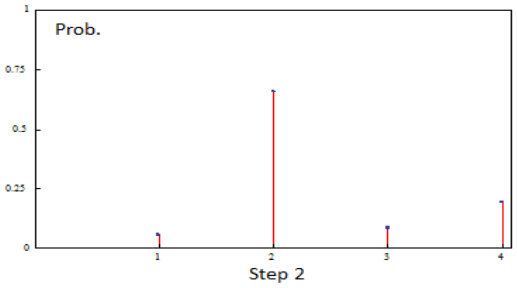
In this section, we have shown the simulation of particle moving in one dimension with a potential barrier in its path. Potential barrier is created by giving an appropriate phase velocity to to the state where barrier is located.

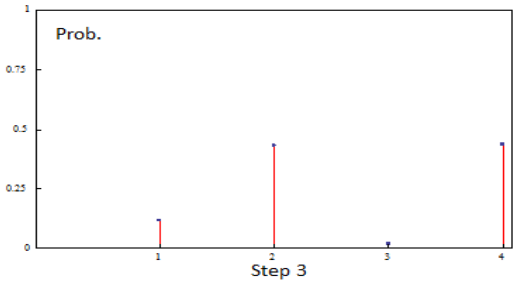
Two Qubit

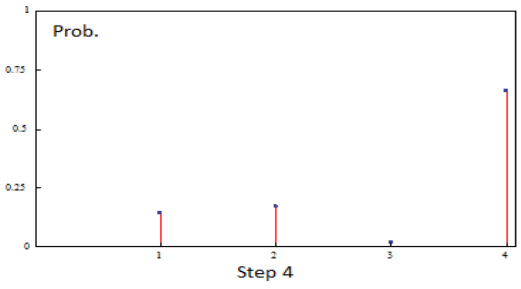
The barrier is placed at the location '3' and all the other parameters are kept same. The height of the barrier is equal to the height of potential wells considered in the two qubit case of the previous section.

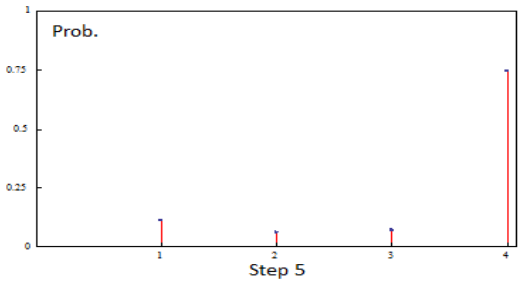

Prob.
Step 0

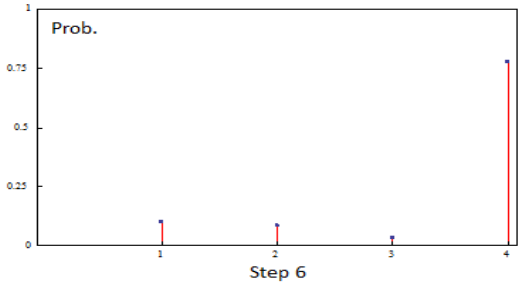

Prob.
Step 1

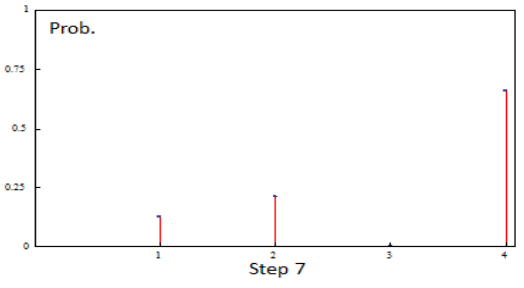

Prob.
Step 2


Prob.
Step 3


Prob.
Step 4


Prob.
Step 5


Prob.
Step 6


Prob.
Step 7

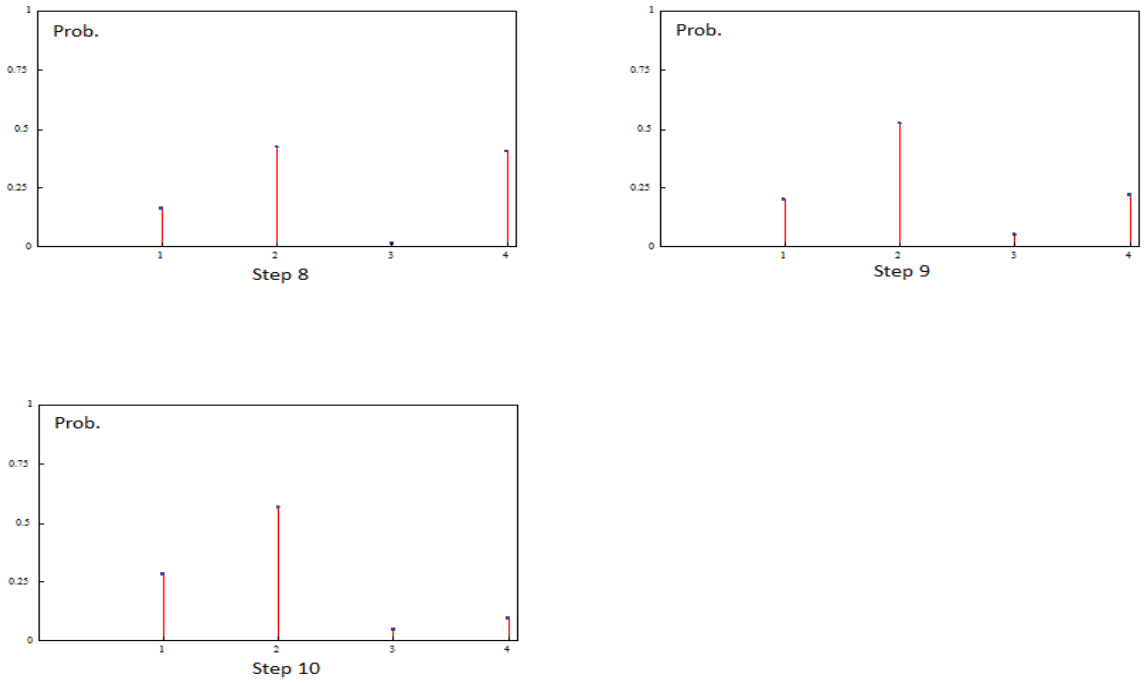


Figure 3.6: Probability distribution of the particle as a function of time (2 Qubits). There is a potential barrier at site 3.

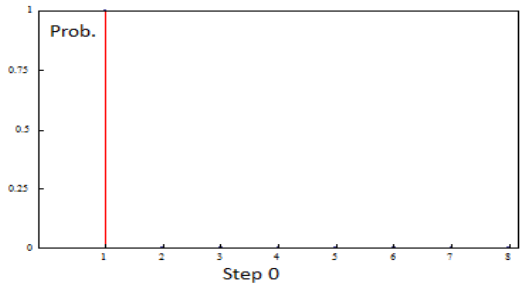
Numerical values for the probabilities shown in the Figure[3.6] are given in the table below:

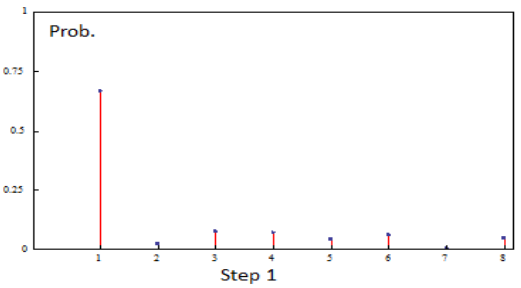
	Position 1	Position 2	Position 3	Position 4
Time Step 0	0.0000	1.0000	0.0000	0.0000
Time Step 1	0.0260	0.8709	0.0470	0.0561
Time Step 2	0.0605	0.6604	0.0869	0.1922
Time Step 3	0.1159	0.4299	0.0188	0.4353
Time Step 4	0.1434	0.1705	0.0217	0.6645
Time Step 5	0.1131	0.0653	0.0741	0.7476
Time Step 6	0.1004	0.0857	0.0360	0.7780
Time Step 7	0.1251	0.2131	0.0016	0.6601
Time Step 8	0.1610	0.4217	0.0145	0.4028
Time Step 9	0.1993	0.5279	0.0534	0.2194
Time Step 10	0.2823	0.5691	0.0509	0.0978

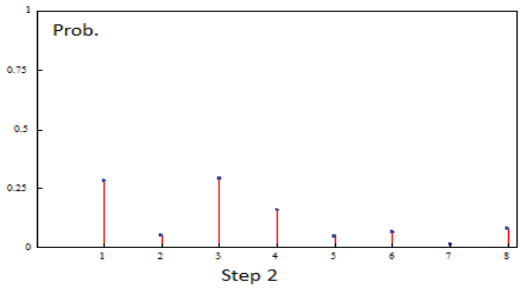
Table 3.5: Probabilities at four positions for different time steps (Potential Barrier at Position 3)

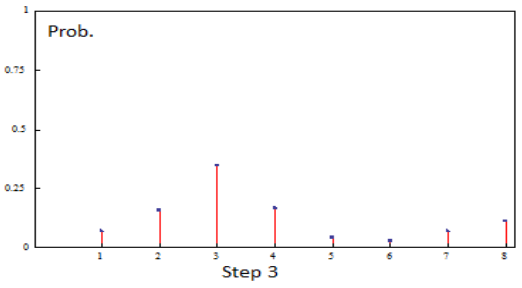
Three qubits

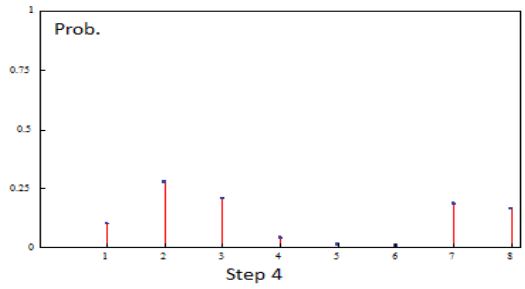
There are eight locations and the potential barrier is placed at the location '5'. The height of the barrier is ten times the height of potential wells considered in the three qubit case of the previous section i.e, v=100. Mass is kept same. Time step is doubled (t=0.2).

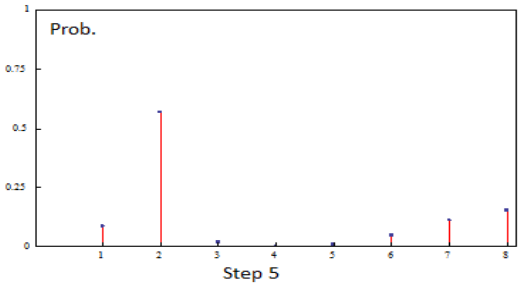












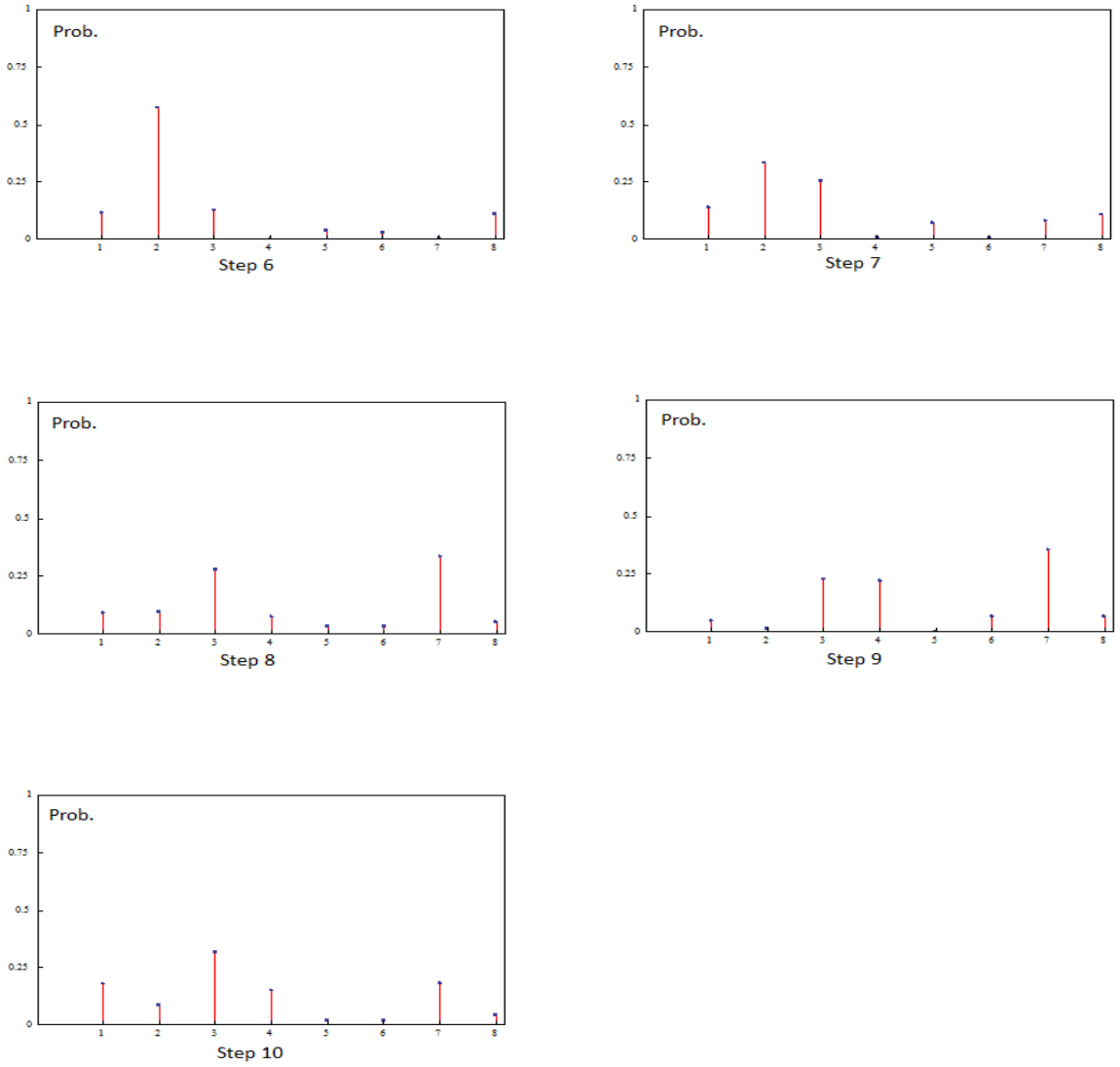


Figure 3.7: Probability distribution of the particle as a function of time with a barrier at location 5(3 Qubits).

Numerical values for the probabilities shown in the Figure[3.7] are given in the table below:

	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.6677	0.0237	0.0762	0.0742	0.0435	0.0616	0.0061	0.0471
Time Step 2	0.2819	0.0533	0.2905	0.1594	0.0467	0.0664	0.0173	0.0845
Time Step 3	0.0707	0.1563	0.3463	0.1664	0.0460	0.0316	0.0719	0.1108
Time Step 4	0.1018	0.2755	0.2073	0.0421	0.0133	0.0094	0.1866	0.1640
Time Step 5	0.0892	0.5710	0.0211	0.0006	0.0082	0.0465	0.1112	0.1522
Time Step 6	0.1141	0.5759	0.1267	0.0018	0.0368	0.0295	0.0051	0.1101
Time Step 7	0.1386	0.3315	0.2539	0.0086	0.0713	0.0062	0.0839	0.1061
Time Step 8	0.0920	0.0965	0.2777	0.0771	0.0333	0.0338	0.3347	0.0549
Time Step 9	0.0500	0.0142	0.2274	0.2197	0.0027	0.0657	0.3530	0.0672
Time Step 10	0.1783	0.0883	0.3145	0.1505	0.0221	0.0191	0.1819	0.0454

Table 3.6: Probabilities at eight positions for different time steps (Potential barrier at site 5.)

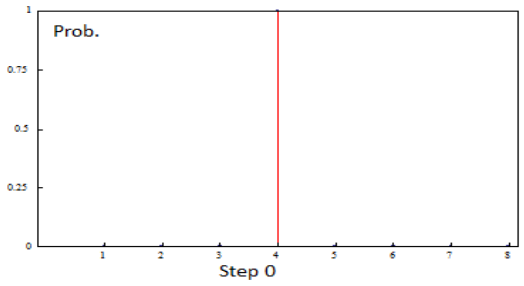
In both the two qubit as well as three qubit case, the particle tunnels from one side to another despite the presence of potential barrier in its way. The increased time step in the second case was necessary to find a simulation that captured the significant tunneling.

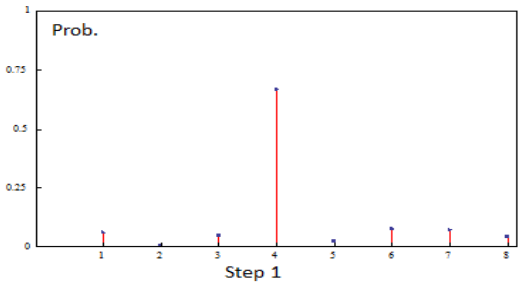
3.3.4 Two potential barriers in the path

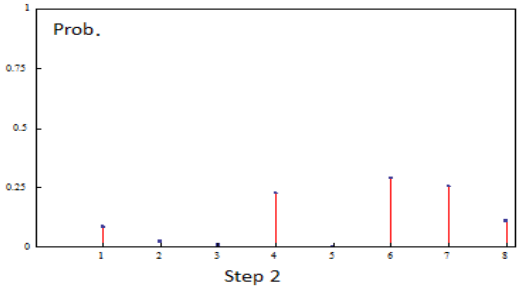
Here we have created two potential barriers in the path of particle. The barriers are located at position '3' and postion '5'. Here also the step size is taken to be 0.2 to cover a significant amount of tunneling. The potential height and mass are same (v=10 and m=0.5). Here we have considered two cases:

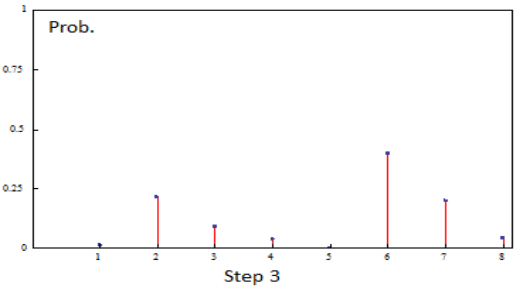
Case 1

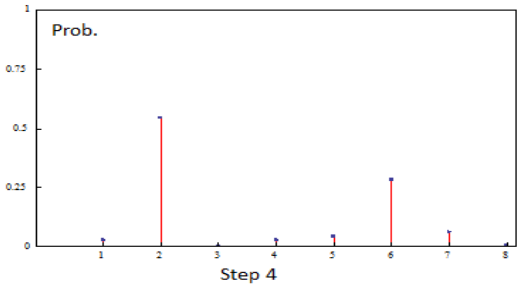
Here the initial state of particle is between the two potential barriers i.e., position 4. It is observed that the probability oscillates. In three time steps, probability to find particle on either sides of barriers increases significantly at the cost of probability at location between the two barriers. After few time steps, the probability at center again increases. As expected, the probability at two barriers is negligible.

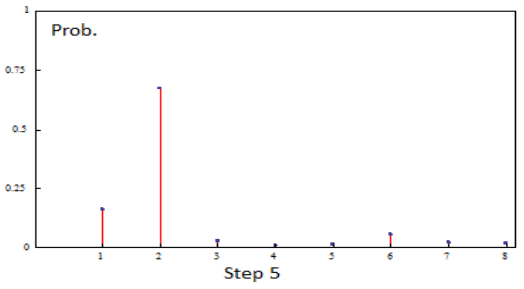

Prob.
Step 0

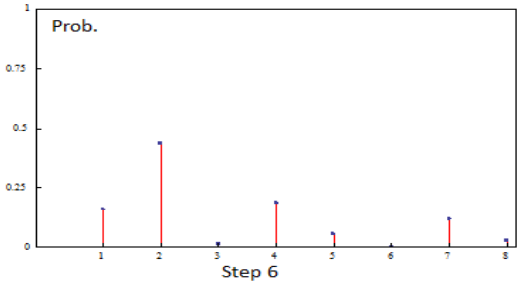

Prob.
Step 1

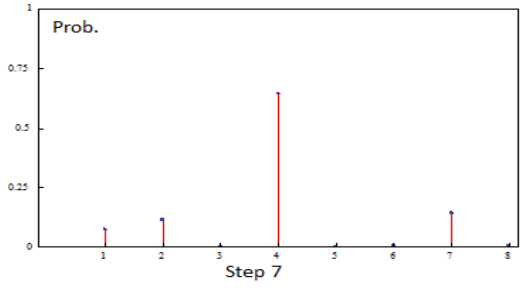

Prob.
Step 2


Prob.
Step 3


Prob.
Step 4


Prob.
Step 5


Prob.
Step 6


Prob.
Step 7

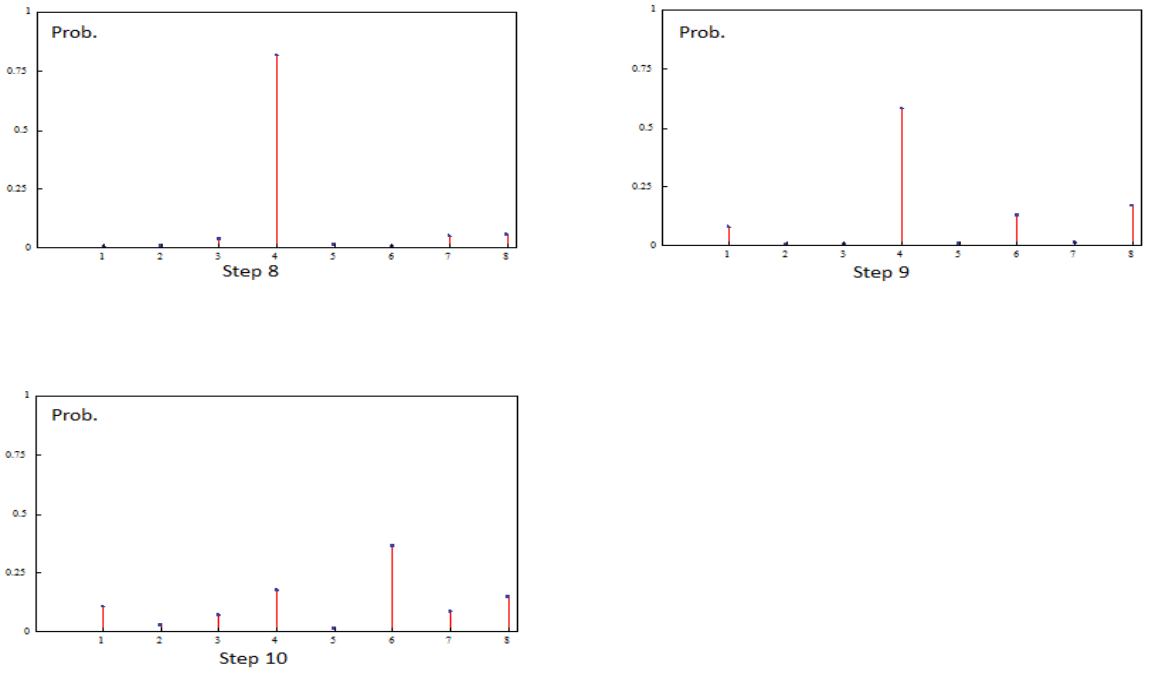


Figure 3.8: Probability distribution of the particle as a function of time (3 Qubits) with barriers at location 3 and location 5.

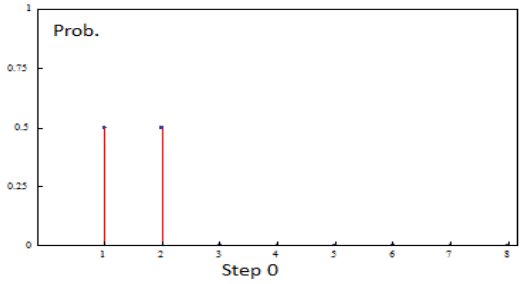
Numerical values for the probabilities shown in the Figure[3.8] are given in the table below:

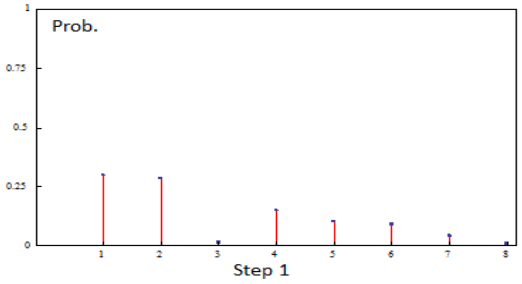
	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.0616	0.0061	0.0471	0.6677	0.0237	0.0762	0.0742	0.0435
Time Step 2	0.0868	0.0228	0.0087	0.2265	0.0024	0.2886	0.2548	0.1094
Time Step 3	0.0131	0.2142	0.0909	0.0396	0.0002	0.3962	0.2004	0.0454
Time Step 4	0.0277	0.5466	0.0025	0.0275	0.0447	0.2822	0.0648	0.0040
Time Step 5	0.1623	0.6762	0.0307	0.0128	0.0169	0.0590	0.0230	0.0191
Time Step 6	0.1602	0.4350	0.0138	0.1868	0.0561	0.0018	0.1180	0.0283
Time Step 7	0.0757	0.1140	0.0030	0.6478	0.0019	0.0081	0.1423	0.0072
Time Step 8	0.0070	0.0086	0.0370	0.8180	0.0143	0.0042	0.0520	0.0589
Time Step 9	0.0818	0.0073	0.0053	0.5836	0.0113	0.1288	0.0134	0.1685
Time Step 10	0.1079	0.0305	0.0718	0.1767	0.0151	0.3621	0.0885	0.1474

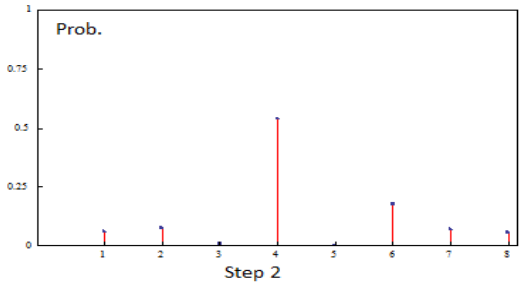
Table 3.7: Probabilities at eight positions for different time steps (Potential barrier at site 3 and 5.)

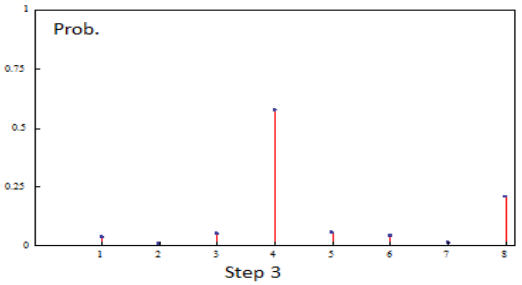
Case 2

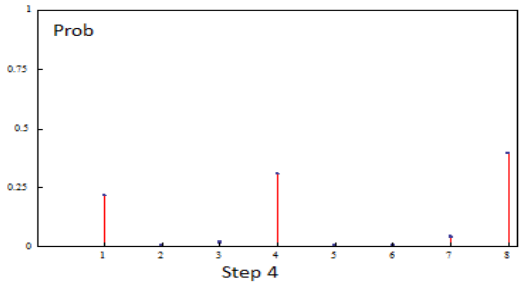
Here we start with a state which is equal superposition of $|000\rangle$ and $|001\rangle$. It can be seen that the rate of spread of wave function is lower. Within two time steps, the probability at position '4' increases considerably and in next two steps probability at the right side of second barrier also becomes significant.

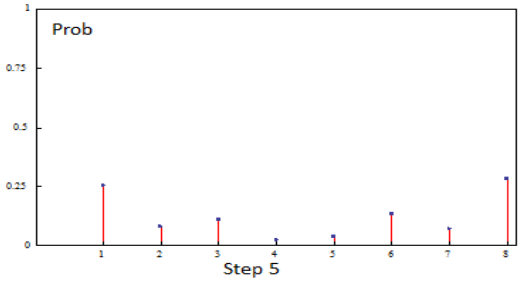


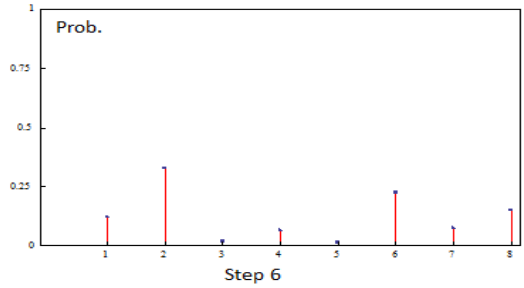


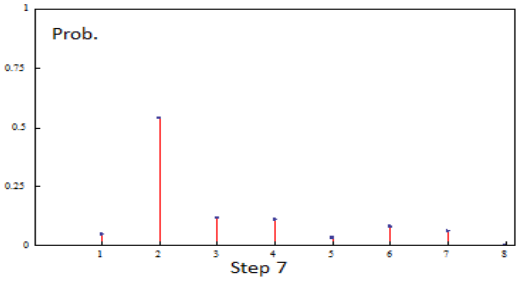












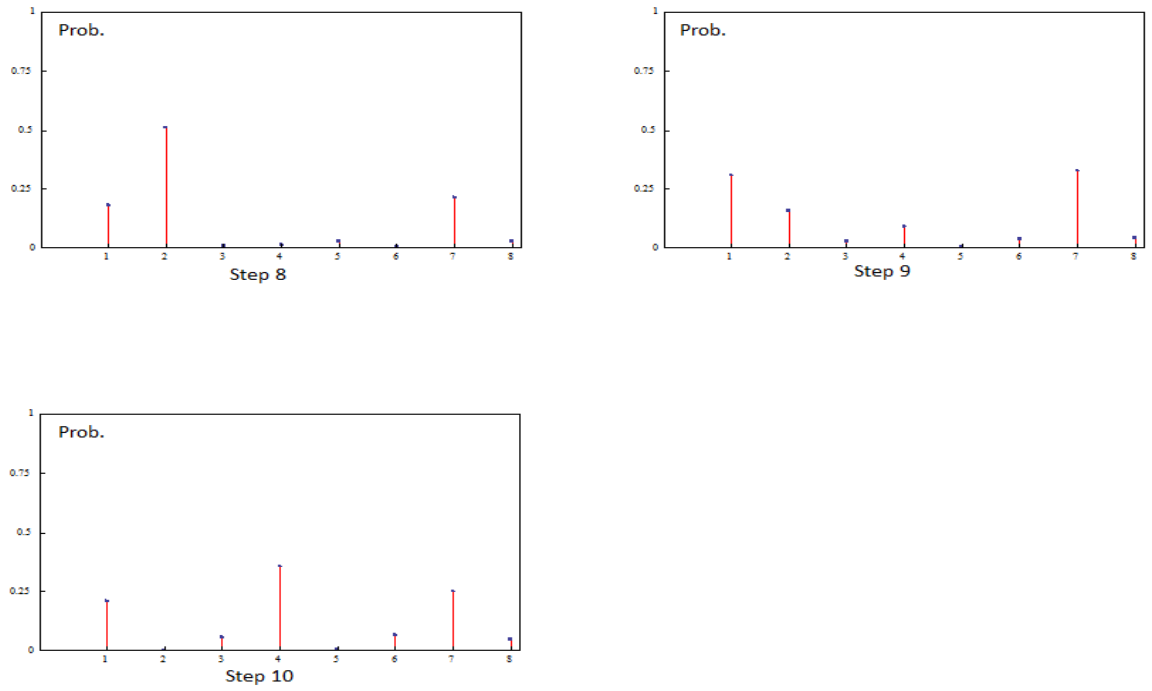


Figure 3.9: Probability distribution of the particle (superposition initial state) as a function of time (3 Qubits) with a barriers at location 3 and 5.

Numerical values for the probabilities shown in the Figure[3.9] are given in the table below:

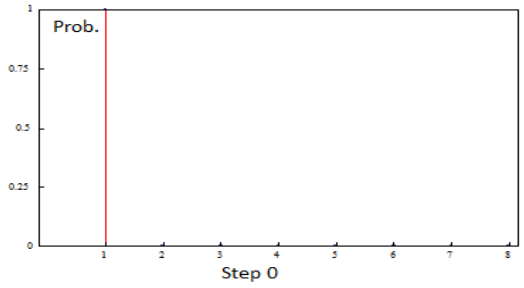
	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	0.5000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.2990	0.2845	0.0169	0.1490	0.1021	0.0939	0.0434	0.0110
Time Step 2	0.0625	0.0791	0.0092	0.5413	0.0019	0.1773	0.0718	0.0567
Time Step 3	0.0380	0.0110	0.0514	0.5776	0.0581	0.0417	0.0139	0.2083
Time Step 4	0.2177	0.0056	0.0218	0.3071	0.0062	0.0049	0.0429	0.3938
Time Step 5	0.2521	0.0839	0.1085	0.0270	0.0408	0.1331	0.0734	0.2812
Time Step 6	0.1208	0.3274	0.0219	0.0662	0.0139	0.2239	0.0761	0.1496
Time Step 7	0.0506	0.5420	0.1177	0.1091	0.0321	0.0812	0.0647	0.0026
Time Step 8	0.1818	0.5136	0.0085	0.0135	0.0300	0.0068	0.2140	0.0317
Time Step 9	0.3082	0.1567	0.0284	0.0938	0.0054	0.0400	0.3259	0.0417
Time Step 10	0.2102	0.0023	0.0571	0.3556	0.0062	0.0678	0.2513	0.0494

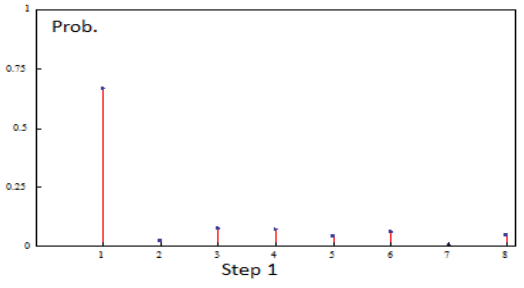
Table 3.8: Probabilities at eight positions for different time steps(Potential barrier at site 3 and 5.)

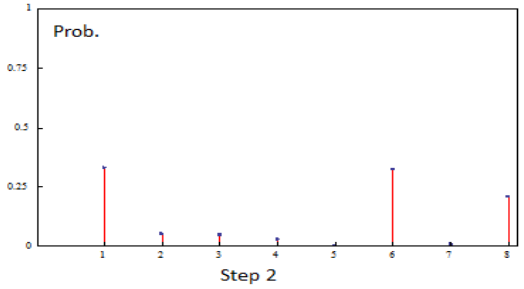
3.3.5 Three potential barriers in path

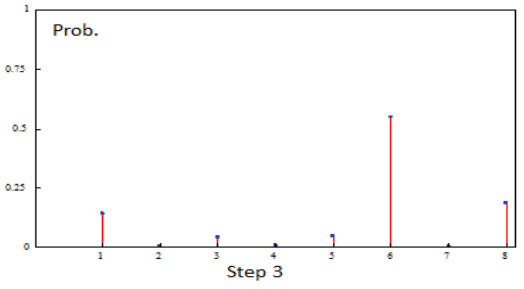
We have increased the number of barriers to three. The additional barrier is added to position '4' (other two being at location '3' and '5'). The height of the three barriers

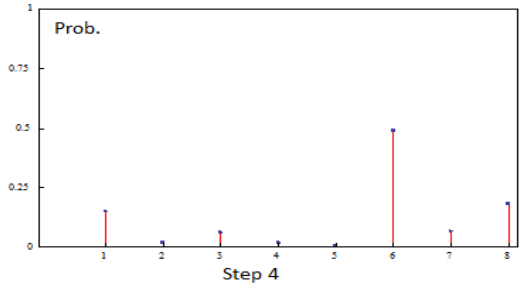
is equal to each other as well as to the barriers considered in the previous section. These three barriers can be considered as a single barrier of larger width. Time step and mass are 0.2 and 0.5 respectively. Here also we have done simulations for the similar two cases: one with a localized initial state and other with initial state as superposition of two states:

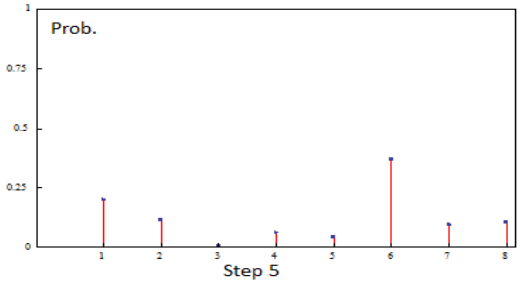












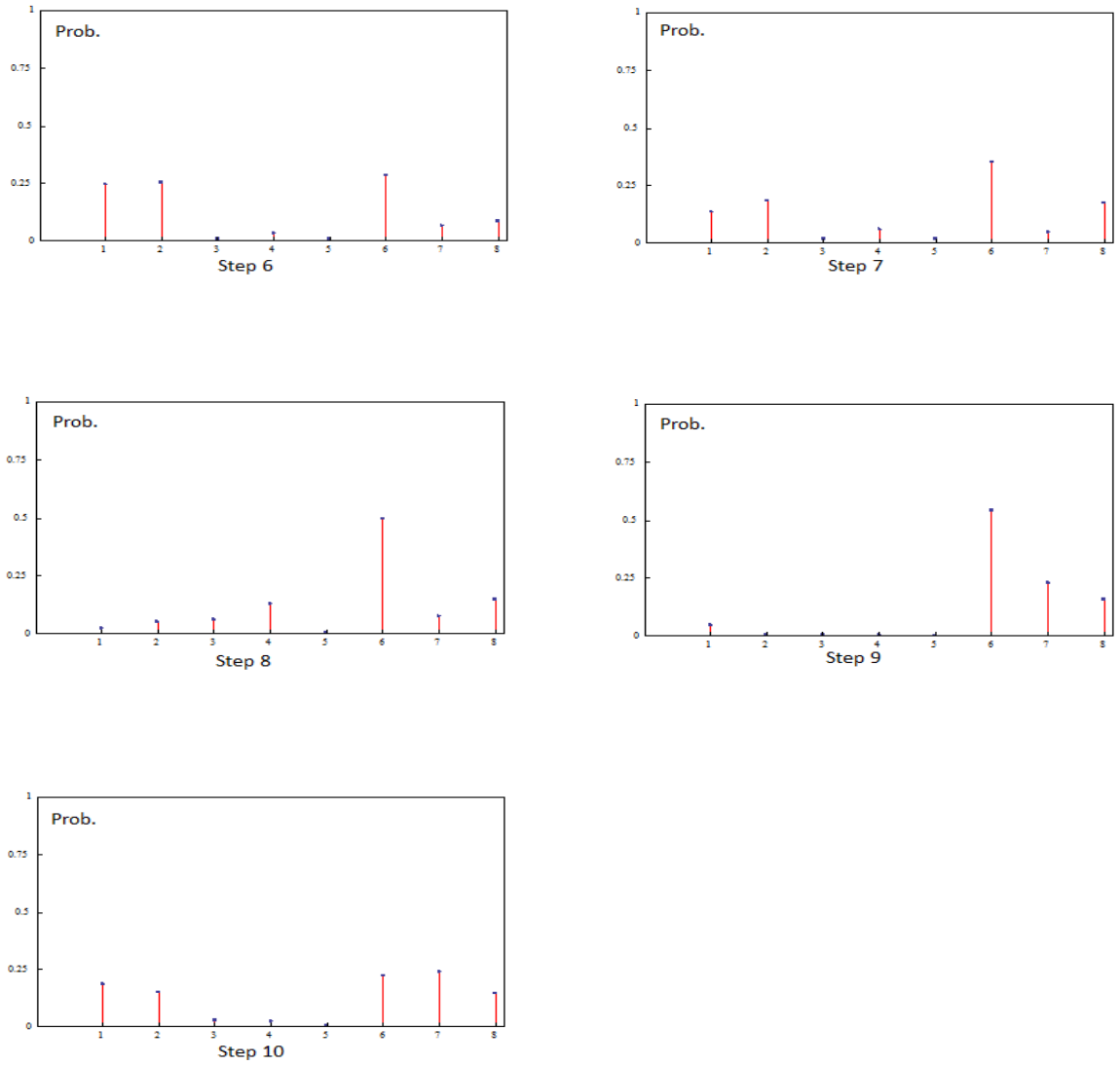


Figure 3.10: Probability distribution of the particle as a function of time (3 Qubits) with three barriers at locations 3, 4 and 5.

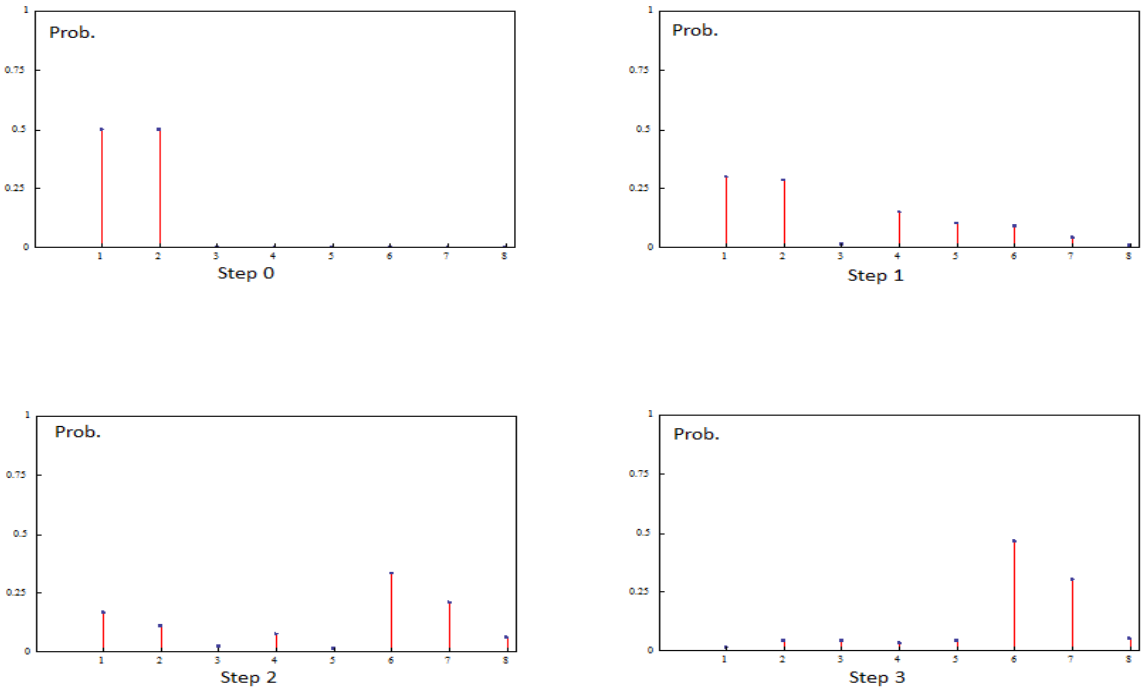
Numerical values for the probabilities shown in the Figure[3.10] are given in the table below:

	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.6677	0.0237	0.0762	0.0742	0.0435	0.0616	0.0061	0.0471
Time Step 2	0.3281	0.0539	0.0505	0.0299	0.0002	0.3213	0.0095	0.2066
Time Step 3	0.1437	0.0070	0.0456	0.0105	0.0487	0.5504	0.0071	0.1871
Time Step 4	0.1501	0.0217	0.0623	0.0193	0.0052	0.4914	0.0683	0.1815
Time Step 5	0.2008	0.1150	0.0068	0.0649	0.0449	0.3667	0.0955	0.1054
Time Step 6	0.2457	0.2522	0.0128	0.0367	0.0116	0.2847	0.0699	0.0864
Time Step 7	0.1349	0.1839	0.0224	0.0615	0.0218	0.3505	0.0502	0.1748
Time Step 8	0.0258	0.0533	0.0624	0.1291	0.0049	0.4978	0.0788	0.1479
Time Step 9	0.0483	0.0076	0.0066	0.0060	0.0017	0.5432	0.2296	0.1570
Time Step 10	0.1869	0.1502	0.0285	0.0252	0.0052	0.2212	0.2376	0.1453

Table 3.9: Probabilities at eight positions for different time steps (Potential barriers at site 3, 4 and 5.)

Case B: Superposition state

We start with a state which is equal superposition of state $|000\rangle$ and state $|001\rangle$. Here the picture is much more clear and tunneling is much more clearly demonstrated.



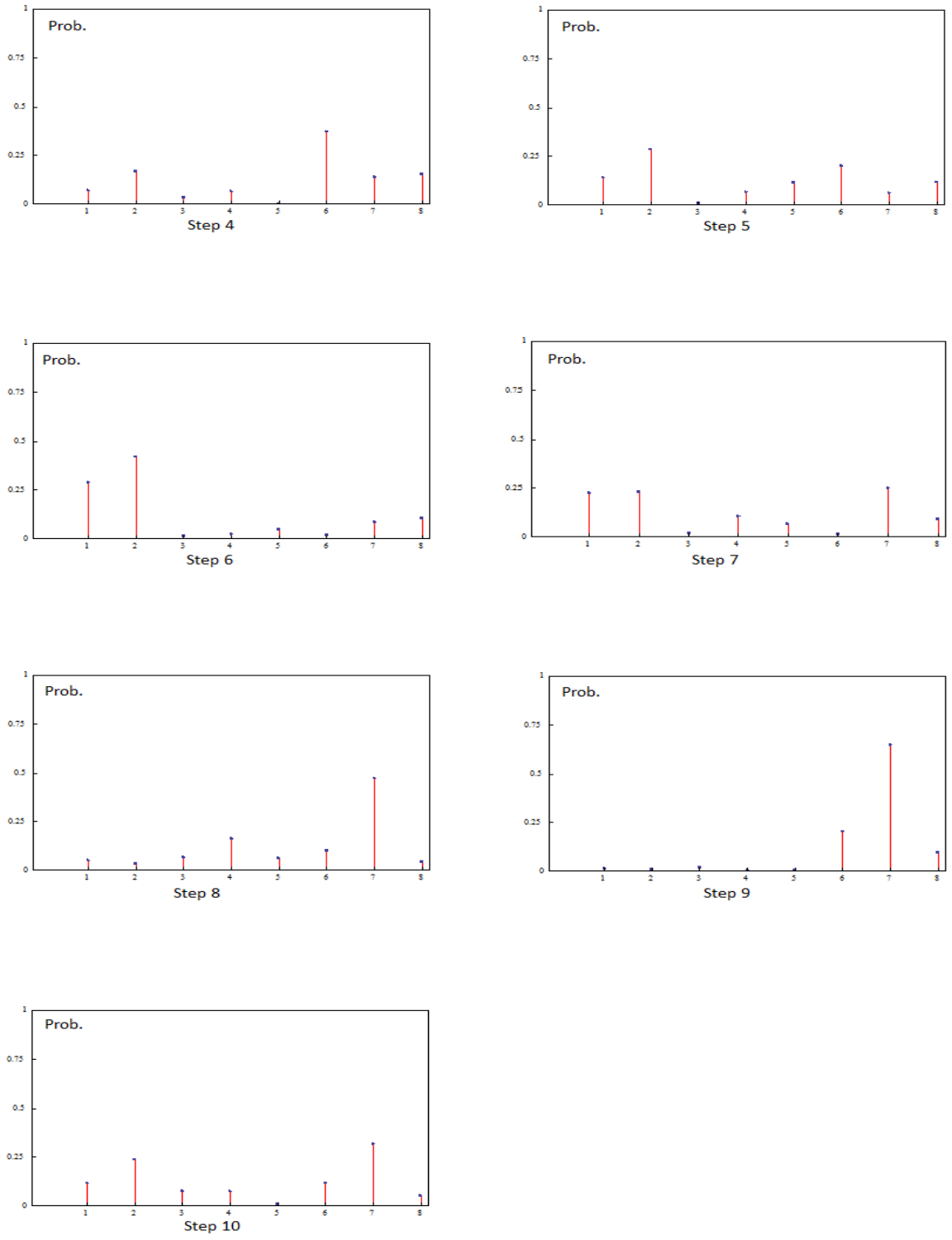


Figure 3.11: Probability distribution of the particle (superposition initial state) as a function of time (3 Qubits) with a barriers at locations 3, 4 and 5.

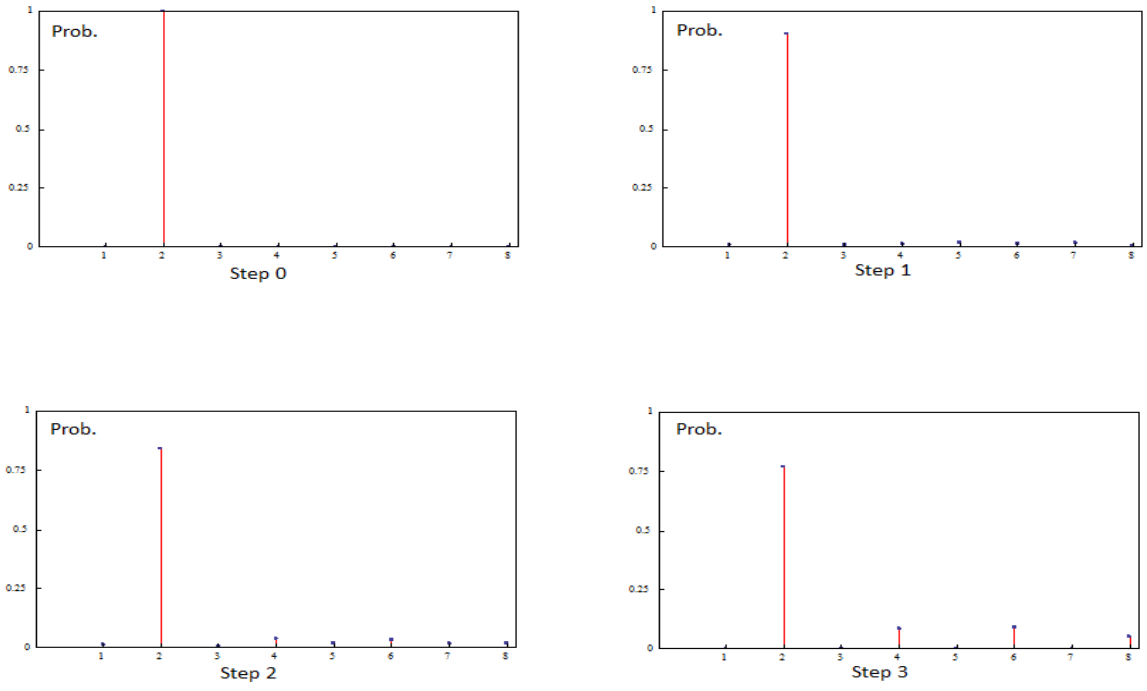
Numerical values for the probabilities shown in the Figure[3.11] are given in the table below:

	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	0.5000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.2990	0.2845	0.0169	0.1490	0.1021	0.0939	0.0434	0.0110
Time Step 2	0.1674	0.1100	0.0242	0.0776	0.0149	0.3319	0.2107	0.0634
Time Step 3	0.0160	0.0422	0.0452	0.0320	0.0418	0.4667	0.3013	0.0546
Time Step 4	0.0716	0.1662	0.0342	0.0658	0.0001	0.3701	0.1384	0.1537
Time Step 5	0.1401	0.2827	0.0094	0.0700	0.1141	0.2011	0.0651	0.1175
Time Step 6	0.2857	0.4164	0.0144	0.0266	0.0473	0.0187	0.0851	0.1058
Time Step 7	0.2233	0.2285	0.0217	0.1056	0.0660	0.0160	0.2475	0.0912
Time Step 8	0.0529	0.0338	0.0692	0.1610	0.0627	0.1002	0.4750	0.0454
Time Step 9	0.0134	0.0120	0.0193	0.0035	0.0034	0.2018	0.6497	0.0970
Time Step 10	0.1165	0.2365	0.0796	0.0760	0.0082	0.1171	0.3148	0.0512

Table 3.10: Probabilities at eight positions for different time steps (Potential barriers at site 3, 4 and 5.)

3.3.6 Dirac Comb Potential

As explained earlier, Dirac Comb potential can be achieved by applying Z-rotation on last qubit. We have done the simulations for the three- qubit case. It can be seen as a four well potential with four wells located at positions 2,4,6 and 8. Here time step is taken to be 0.1. So the tunneling is visible in the later time steps as compared to the previous section where the time step was double.



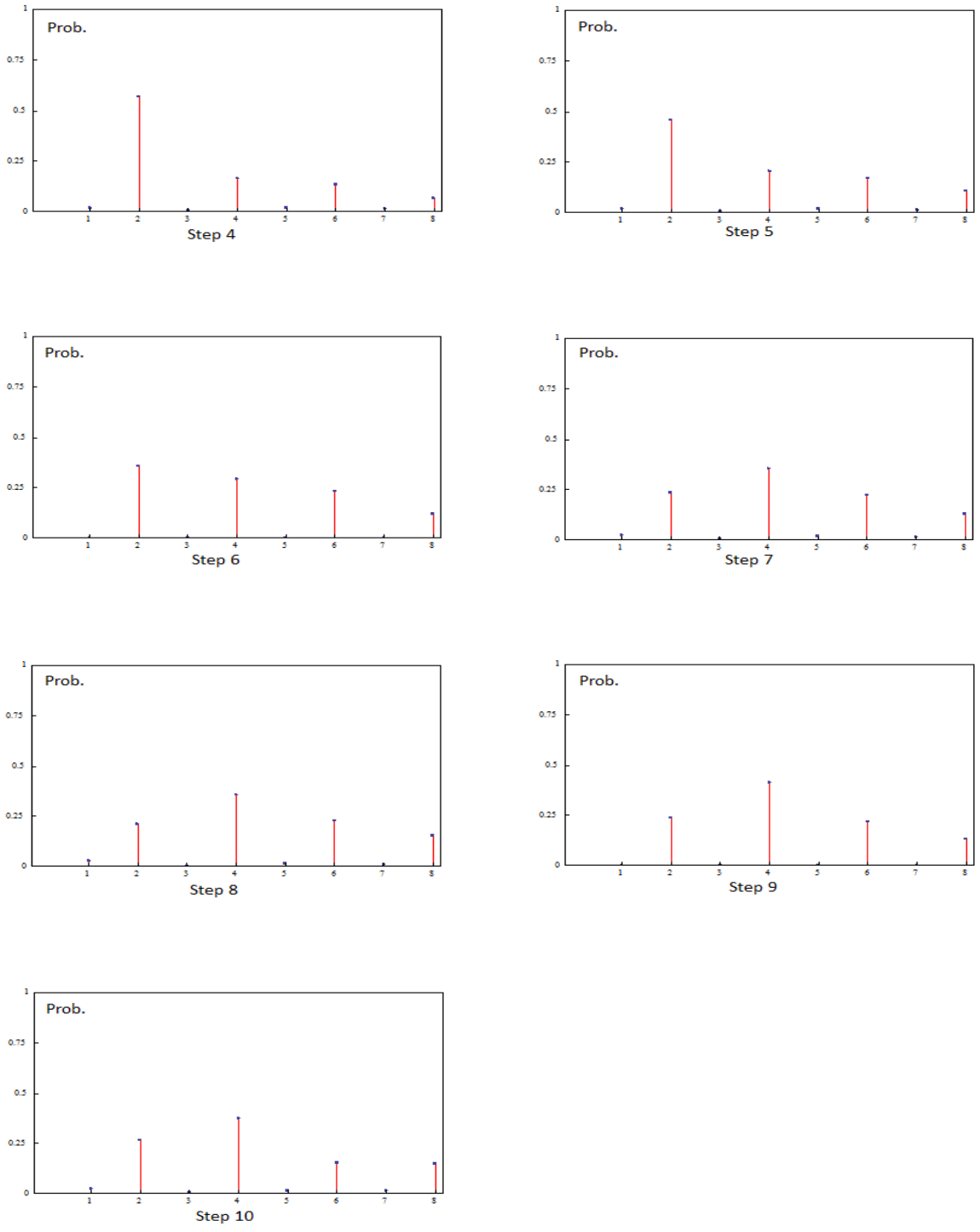


Figure 3.12: Probability distribution of the particle as a function of time for Dirac Comb Potential.

Numerical values for the probabilities shown in the Figure[3.12] are given in the table below:

	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Time Step 0	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Time Step 1	0.0117	0.9046	0.0085	0.0150	0.0218	0.0140	0.0199	0.0045
Time Step 2	0.0137	0.8436	0.0069	0.0401	0.0216	0.0336	0.0180	0.0224
Time Step 3	0.0002	0.7703	0.0002	0.0853	0.0000	0.0904	0.0000	0.0535
Time Step 4	0.0192	0.5714	0.0065	0.1639	0.0209	0.1339	0.0163	0.0679
Time Step 5	0.0224	0.4613	0.0039	0.2042	0.0185	0.1681	0.0134	0.1081
Time Step 6	0.0005	0.3559	0.0010	0.2921	0.0001	0.2305	0.0002	0.1197
Time Step 7	0.0254	0.2337	0.0048	0.3526	0.0182	0.2208	0.0153	0.1292
Time Step 8	0.0285	0.2097	0.0010	0.3554	0.0144	0.2272	0.0119	0.1520
Time Step 9	0.0004	0.2365	0.0028	0.4110	0.0003	0.2172	0.0004	0.1314
Time Step 10	0.0268	0.2654	0.0043	0.3711	0.0175	0.1518	0.0156	0.1475

Table 3.11: Probabilities at eight positions for different time steps (Dirac Comb Potential- 3 Qubits)

Chapter 4

NMR Quantum Computing

"Computers are physical objects, and computations are physical processes."

- David Deutsch

4.1 Introduction

The quantum computation can be thought of as set of unitary operations acting on a set of input qubits. To build a quantum computer, the main job is to find suitable quantum systems, set them up in pure initial states and physically achieve Hamiltonians that produce these unitary transformations. Customary designs for quantum computers consist of N two-level systems which are coupled to one another and have some particular interaction with the outside world so that they can be examined and controlled, but are otherwise isolated [DiV].

A molecule with N spin $\frac{1}{2}$ nuclei can be thought as an N-bit quantum computer, given that the spins are able to interact, one can control their states in a desired manner and there is well defined scheme of reading out the outcome of the computation. In contrast, NMR systems are rather different. An archetypal NMR sample consists of not just one spin-system, but a large number of copies, one from each molecule in the sample [Kee10]. Nuclear spins at thermal equilibrium are in a statistical mixture of pure states, whereas quantum computing is typically performed on pure input states. It was shown that quantum computing can be performed using a special kind of mixed states i.e., ensembles in a pseudo pure state, as well. It will be discussed in the next section. While Dirac bra and ket notation is usually used for quantum

computers, NMR systems are described using the density matrix approach because of the statistical nature [Jon01].

4.2 Pseudo-Pure State

An ensemble with an each member represented by a same state vector corresponds to a pure state. On the other hand, a mixed ensemble cannot be characterized by a single vector in Hilbert space. The density matter approach is used to study these situations. A density operator can be assigned to both pure as well as mixed state.

For a pure state with state vector $|\psi\rangle$, the density operator is given by:

$$\rho = |\psi\rangle\langle\psi| \tag{4.1}$$

For a mixed state, where fraction 'a' of the members of ensemble are in state $|\psi\rangle$ and the rest of members are in state $|\phi\rangle$, the density operator is given by:

$$\rho = a|\psi\rangle\langle\psi| + (1-a)|\phi\rangle\langle\phi| \tag{4.2}$$

Thus a density operator can always be associated with a general quantum mechanical system drawn from a general ensemble irrespective of whether it is pure or mixed. For pure states, a sate vector can be associated with the system. It can be seen that, for pure states we have:

$$\rho^2 = \rho \tag{4.3}$$

As discussed earlier, the quantum algorithms are usually devised in terms of pure states. A quantum computer consisting of this pure state is thus an ensemble of independent quantum computers, doing the same computation. The result in a measurement is the average of all the computers registers (states of the spins). It can be seen that for such average results, it can be problematic to work with mixed states as this averaging can erase the results of a computation. This problem can be circumvented by creating a subsystem within the whole system that behaves precisely like a pure state. These states are known as pseudo-pure states [KDK00].
Mathematically, pesudo-pure state can be written as a sum of unit operator and an operator representing a pure state:

$$\rho_{pp} \propto \alpha \mathbb{1} + \beta\rho_p \tag{4.4}$$

where ρ_{pp} is the pseudo pure state ρ_p is a pure state. An NMR signal is an average over all the spins present in the ensemble. This identity operator corresponds to equal populations of all the levels and thus does not contribute to the signal. Only the ρ_p parts contribute to the signal. Thus behavior of a pseudo pure state is exactly equal to that of pure state. The coefficient β is largely determined by the polarization of the spin system. It can be easily seen that a single spin is always in a pseudo pure state.

Different procedures have been proposed and experimentally implemented to create a pseudo-pure state in NMR [Jon11]. These schemes to produce pseudo-pure state can broadly classified into three categories: spatial averaging, temporal averaging, and logical labelling.

D. Cory, A. Fahmy and T. Havel showed that a spatially averaged pseudo-pure state can be created by a careful arrangement of rf gradients and pulses with different flip angles [DGCH97]. It is called spatial because it creates a spatial distribution of states across the ensemble whose mean density matrix is pseudo-pure. The scheme $\left[\frac{\pi}{3}\right]_x^2 - [\triangle]_z - \left[\frac{\pi}{4}\right]_x^1 - \left[\frac{1}{2J}\right] - \left[-\frac{\pi}{4}\right]_y^1 - [\triangle]_z$ leads to pseudo-pure matrix represented by $\frac{1}{2}\left[I_{1z} + I_{2z} + 2I_{1z}I_{2z}\right]$. Here J is the coupling constant, $\triangle_z$ is a gradient in the z direction and 1 & 2 are the spin labels.

Knill and co. showed that one can use several experiments with different preparation steps, whose averaged results give a pure state [EKL98]. It is called temporal averaging scheme as these unitary transformations are executed sequentially in time. For example, in equilibrium, the populations of four states for two coupled spins are:

$$\begin{aligned} |\uparrow\uparrow\rangle &\longrightarrow \frac{1}{4} + \epsilon \\ |\uparrow\downarrow\rangle &\longrightarrow \frac{1}{4} \\ |\downarrow\uparrow\rangle &\longrightarrow \frac{1}{4} \\ |\downarrow\downarrow\rangle &\longrightarrow \frac{1}{4} - \epsilon \end{aligned} \tag{4.5}$$

The populations of three states (for e.g., $|\uparrow\downarrow\rangle, |\downarrow\uparrow\rangle, |\downarrow\downarrow\rangle$) can be equalized by cyclically permuting them and adding the results. The timed average population can

be written as:

$$\frac{1}{4}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}+\epsilon\begin{pmatrix}1\\-\frac{1}{3}\\-\frac{1}{3}\\-\frac{1}{3}\end{pmatrix}=\left(\frac{1}{4}-\frac{\epsilon}{3}\right)\begin{pmatrix}1\\1\\1\\1\end{pmatrix}+\frac{4\epsilon}{3}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \tag{4.6}$$

If the corresponding average density operator is calculated, it can be seen that it corresponds to the sum of the unit operator and a pure state. Thus one effectively achieves a pseudo pure state.

4.3 Single Qubit Gates

Single qubit gates correspond to rotations in single-spin subspaces. Any rotation of this kind can be implemented by RF-pulses [JAJM]. For implementing a gate on a single nucleus, one uses selective pulses. In case of simultaneously applying the gate to large number of separate nuclei, hard pulses are used. In this case, gate can be seen as product of one qubits gates, one for each affected nucleus.

In a rotating frame, a RF pulse can be represented by $e^{-\frac{\iota H t}{\hbar}}$, where t is the duration of the pulse and H is the Hamiltonian during this time. Generalized x-pulses can be written as $e^{-\frac{\iota\phi_x \mathbf{S}_x}{\hbar}}$ and generalized y-pulses can be written as $e^{-\frac{\iota\phi_x \mathbf{S}_y}{\hbar}}$. Here ϕ is the flip angle. Using these two generalized rotations,one can implement an SU(2) operation. For example, the set of rotations around the z axis, which cannot be directly implemented by RF pulses, are employed by combing three rotations around axis in xy-plane:

$$\begin{aligned} e^{-\frac{\iota\phi \mathbf{S}_z}{\hbar}} &= e^{-\frac{\iota\frac{\pi}{2}\mathbf{S}_y}{\hbar}} e^{\frac{\iota\phi \mathbf{S}_x}{\hbar}} e^{\frac{\iota\frac{\pi}{2}\mathbf{S}_y}{\hbar}} \\ &= e^{-\frac{\iota\frac{\pi}{2}\mathbf{S}_x}{\hbar}} e^{-\frac{\iota\phi \mathbf{S}_y}{\hbar}} e^{\frac{\iota\frac{\pi}{2}\mathbf{S}_x}{\hbar}} \end{aligned} \tag{4.7}$$

We now show the construction of some important single-qubit gates:

NOT gate

NOT gate can be implemented using a $180^\circ I_x$ pulse:

$$\begin{aligned} NOT &\longrightarrow e^{-\frac{\iota\pi \mathbf{S}_x}{\hbar}} \\ &= \begin{pmatrix} 0 & -\iota \\ -\iota & 0 \end{pmatrix} \\ &= e^{-\iota\frac{\pi}{2}} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{aligned} \tag{4.8}$$

Thus this NMR implementation of NOT gate is different from the actual NOT gate by an overall phase of $-\pi/2$. But this overall phase doesn't effect any observable quantity and can not be observed itself. So, this implementation or any other such implementation with a global phase is as good as a real NOT gate.

Hadamard Gate

Hadamard gate can be implemented using the following RF pulse:

$$e^{-\iota\left(\frac{\pi}{\sqrt{2}}\right)\frac{(\mathbf{S}_x+\mathbf{S}_z)}{\hbar}} = \frac{\iota}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{4.9}$$

As usual ignoring the global phase, it has the desired form. Alternatively it can also be implemented by following three pulses:

$$45I_y - 180I_x - 45I_{-y} \tag{4.10}$$

When implementing quantum algorithms on NMR quantum computers, Hadamard is many times replaced by the pseudo-Hadamard gate which is easier to implement. The pseudo-Hadamard operator, h, which has the form:

$$\mathbf{h} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \tag{4.11}$$

This can be implemented by $90I_y$ pulse. In many algorithms a pair of Hadamards is replaced by pseudo Hadamard and its inverse. The pseudo Hadamard inverse is given by:

$$\mathbf{h}^{-1} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \tag{4.12}$$

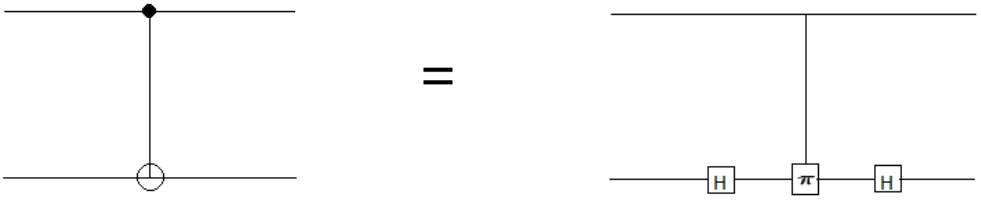
As expected, it can be implemented by $90I_{-y}$ pulse.

4.4 Two Qubit Gates

The gates described in the previous section perform rotations within the subspace corresponding to a single spin. If employed using hard pulses, these gates can affect many spins, but the overall result is that each spin rotates within its own subspace. This way, the operation can be decomposed as the product of two or more one spin operators. A true two qubit gate cannot be decomposed into a set of one qubit gates. It corresponds to rotations within a subspace corresponding to two spins. These gates are at the heart of quantum algorithms. Due to these gates the state of one spin can become dependent on the state of another spin and thus we get a conditional dynamics.

CNOT Gate

Two qubit gates require coupling between the spins and can be implemented using either "soft pulses" or "pulses plus free precession". Controlled-Not gate is a fundamental two qubit gate. As described in chapter 1, it can seen as an application of NOT gate to the target gate if the control gate is in state $|0\rangle$. There are number of methods which can be described for implementing this gate. But we will consider a general approach used in NMR to these kind of gates. It is well known that controlled gates, such as CNOT, are related to controlled phase shifts by the Hadamard transform. As an example, CNOT can be substituted by the following three gate network:

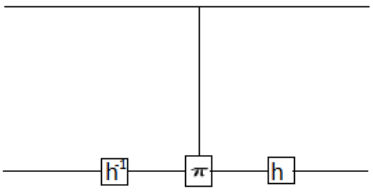


Here H are the usual one qubit Hadamard gates and π is given by:

$$\pi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{4.13}$$

It thus performs the transformation $|11\rangle \longrightarrow -|11\rangle$ and leave the other basis states unchanged.

As explained earlier the pair of Hadamard gates in the above three gate network can be replaced by pseudo Hadamard and pseudo Hadamard inverse. Hence the CNOT can be achieved in NMR using the following sequence:



Controlled Phase Shift Gate

As discussed earlier, all the controlled gates in quantum computing are related to controlled phase shifts by the Hadamard transform. An example of CNOT gate was given in the previous section. Thus if one knows how to implement a controlled phase gate, practically one knows how to implement any controlled gate.

A generalized phase shift gate can be written as:

$$\phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-\iota\phi} \end{pmatrix} \tag{4.14}$$

It thus performs the transformation $|11\rangle \longrightarrow e^{-\iota\phi}|11\rangle$ and leave the other basis states unchanged. It can be easily decomposed as a product of diagonal operators:

$$\phi = exp[-\iota \times \frac{1}{2}\phi \times \left[-\left(\frac{1}{2}E\right) + I_z + S_z - 2I_zS_z\right]] \tag{4.15}$$

The first term which requires a Hamiltonian proportional to$\frac{1}{2}E$ is not important as it simply imposes a global phase shift, and can be ignored. The last three terms are unproblematic: I_z and S_z can be implemented as periods of free precession or by using composite z-pulses, and $2I_zS_z$ is proportional to the scalar coupling Hamiltonian. The matrix π used in the implementation of CNOT gate can be implemented as:

$$\pi = (90^\circ I_z)\,(90^\circ S_z)\,(-90^\circ 2I_zS_z) \tag{4.16}$$

It can be achieved in a variety of ways. The above three Hamiltonians can be applied in any order as the three terms commute. I_z and S_z can be realized by free precession or by one of the large variety of different composite pulses.

Two qubit gates can also bring in global phase shifts just like simple one qubit gates. Such phase shifts can be disregarded as these are applied to the entire wavefunction and not just to the spins participating in the gate. Conceptually these phase shifts can be thought of as coming up from the lack of $frac$12E expression in controlled phase shift gates.

4.5 Quantum State Tomography

It is often essential to carry out a complete determination of the quantum state of a qubit or a set of qubits to understand the results or physical process involved in a quantum information processing. Quantum state tomography is the method of rebuilding the quantum state (density matrix) by measurements on the systems coming from the source. Measurements which are made on the system have to be tomographically complete in order to uniquely identify the state. In other words, measured observables must form an operator basis on the Hilbert space of the system.

For a one qubit system, the density matrix of a system can be expanded in the orthogonal pauli basis as:

$$\rho = \frac{tr(\rho)\mathbf{1} + tr(\rho\sigma_x)\sigma_x + tr(\rho\sigma_y)\sigma_y + tr(\rho\sigma_z)\sigma_z}{2} \tag{4.17}$$

It can be recalled that $tr(\rho\sigma_i)\sigma_i$ is the expectation value (not a result of single measurement) of operator σ_i. Thus performing the measurements for Pauli matrices can yield all the elements of the density operator [IOF11]. The generalization of this procedure to an arbitrary number of qubits is straight forward:

$$\rho = \sum_{\vec{v}} \frac{tr\left(\sigma_{v_1} \otimes \sigma_{v_2} \otimes .....\sigma_{v_n}.\rho\right)\sigma_{v_1} \otimes \sigma_{v_2} \otimes .....\sigma_{v_n}}{2^n} \tag{4.18}$$

here the sum run overs all the vectors$\vec{v} = v_1, v_2, ..., v_n$ whose components are chosen from the set of pauli matrices plus identity.

The measurement in an NMR system can only provide some off-diagonal matrix elements of the density matrix. In order to obtain the rest of the elements of density matrix, one has to rotate the original density matrix by applying rotational operations. For an example, lets consider the system of H and C-13 in chloroform as our two qubit system. As only one spin be measured at one time, one has to carry out the measurements separately for the two spins. One starts a computation and does a measurement on H at some required stage. Next, the computation is restarted from the beginning and one measures the signal for C. Again in the next step, the computation is restarted from beginning, but this time before making the measurement, density matrix is first rotated by applying a rotational operation and then the signals are measured for H and P in independent experiments. This process in continued until all the elements of density operator are obtained [Lee].

To make things clearer, let's assume the density operator is given by:

$$\rho = \begin{pmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \rho_{14} \\ \rho_{21} & \rho_{22} & \rho_{23} & \rho_{24} \\ \rho_{31} & \rho_{32} & \rho_{33} & \rho_{34} \\ \rho_{41} & \rho_{42} & \rho_{43} & \rho_{44} \end{pmatrix} \tag{4.19}$$

But we know that the density operator is hermitian, thus its element in the i-th row

and j-th column is equal to the complex conjugate of the element in the j-th row and i-th column. We can write it as:

$$\rho = \begin{pmatrix} \rho_{11}^* & \rho_{12} & \rho_{13} & \rho_{14} \\ \rho_{12}^* & \rho_{22} & \rho_{23} & \rho_{24} \\ \rho_{13}^* & \rho_{23}^* & \rho_{33} & \rho_{34} \\ \rho_{14}^* & \rho_{24}^* & \rho_{34}^* & \rho_{44} \end{pmatrix}$$

$$= \begin{pmatrix} x_1 & x_2 + \iota x_{11} & x_3 + \iota x_{12} & x_4 + \iota x_{13} \\ x_2 - \iota x_{11} & x_5 & x_6 + \iota x_{14} & x_7 + \iota x_{15} \\ x_3 - \iota x_{12} & x_6 - \iota x_{14} & x_8 & x_9 + \iota x_{16} \\ x_4 - \iota x_{13} & x_7 - \iota x_{15} & x_9 - \iota x_{16} & x_{10} \end{pmatrix} \tag{4.20}$$

(4.21)

In an experiment, one gets two peaks for each of the nuclear spin. In an experiment corresponding to the nuclear spin of C, left peak corresponds to the element ρ_{12} and right peak corresponds to ρ_{34}. Similarly for nuclear spin of H, left peak corresponds to the element ρ_{13} and right peak corresponds to ρ_{24}. To obtain other elements in the density matrix, we have to perform a rotation of density operator so that desired elements to be measures come to positions labelled as 12, 13, 24 and 34. After this rotation, measurement can be made to calculate the elements ρ_{12}‘, ρ_{13}‘, ρ_{24}‘, ρ_{24}‘ of the rotated density matrix. These new elements are a linear combination of the elements of original density matrix. More rotation are made so that we get enough equations to solve for all the matrix elements [GLL].

Experimental Results:

We prepared a pseudo pure state which can be resolved as a sum of Identity part and a pure state part. The pure states part corresponds to $|01\rangle$ in our case. The procedure presented in the previous sections to prepare a spatially averaged pseudo pure state is used to prepare state $|00\rangle$ and then a single qubit NOT gate is implemented on second qubit to achieve the state $|01\rangle$. Here we show the experimental results for state tomography of the state in detail. After the state preparation, we acquired the spectra (labelled as II in the Figure [4.1] & Figure [4.3]and calculated the intensity of peak by integrating the area under curve. Then we added a phase of $90^{\text{ø}}$ in the spectra and recalculated the intensities. We have shown it in Figure [4.2] and

Figure [4.4]. Both these steps were performed for both carbon as well as hydrogen. In this way we got the details of elements $\rho_{12}, \rho_{13}, \rho_{24}, \rho_{24}$ of the density matrix. To get the details about other elements we restarted the process from beginning and before making the observation, we performed the "IX' rotation on density operator(here IX means identity is acting upon spin under observation and NOT gate is acting upon the other spin) and repeated the same procedure for both the spins. Similarly, we performed the same procedure for "IY" rotation and "XX" rotation. The spectra for these four kind of rotations (including II) are given in the Figure [4.1],Figure [4.2],[4.3] and Figure [4.4].

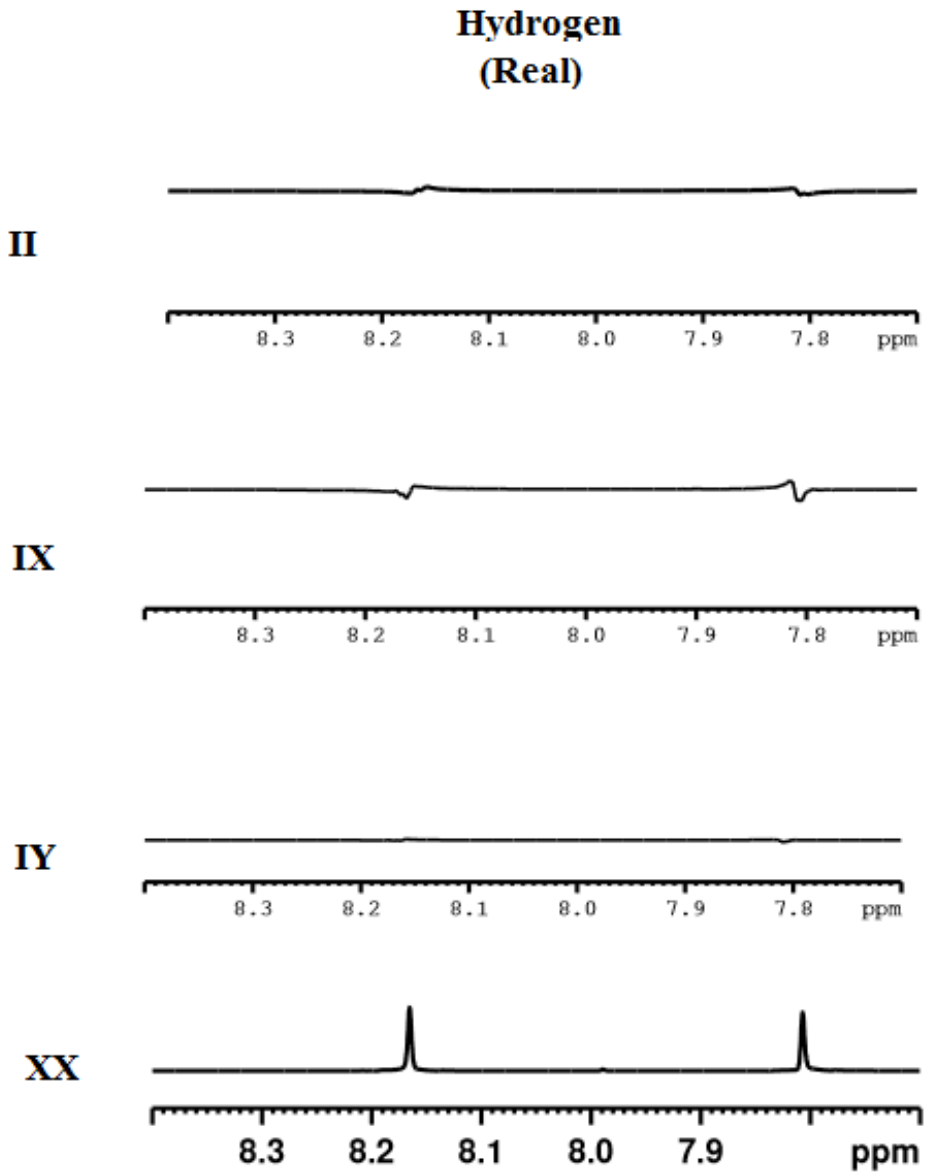


Figure 4.1: Spectra for Hydrogen after the four rotations: II, IX, IY, XX

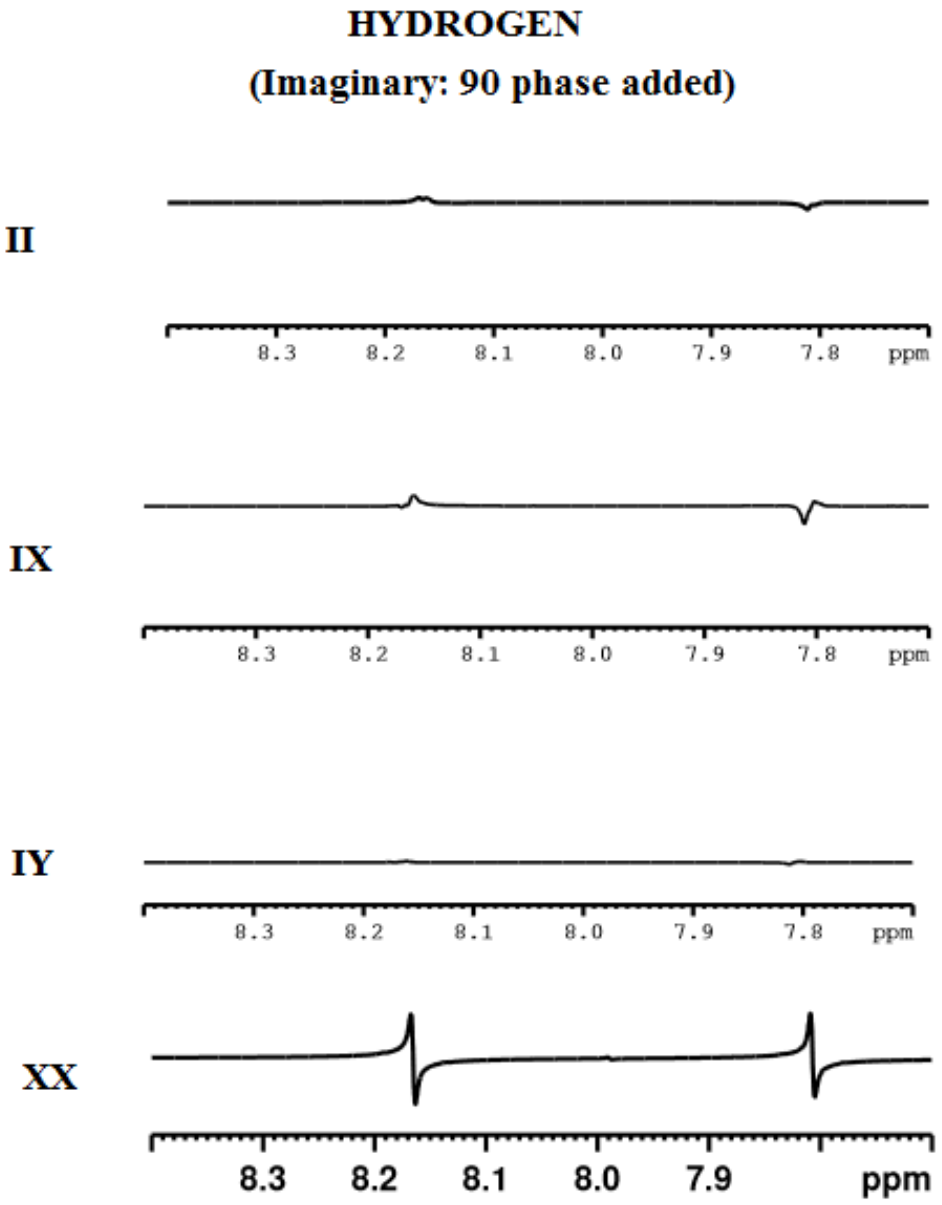


Figure 4.2: Spectra of Hydrogen after adding a phase of 90^0 for the four rotations: II, IX, IY, XX

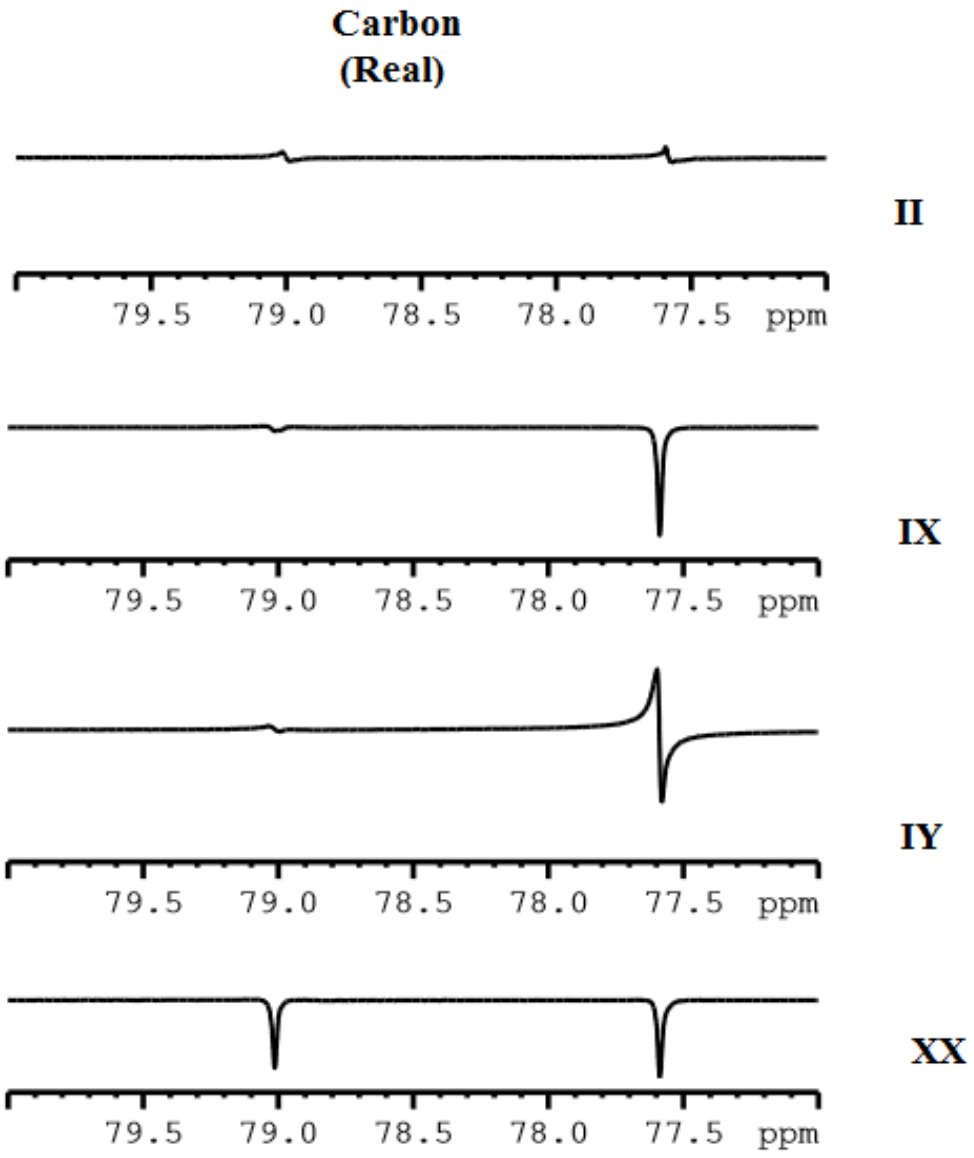


Figure 4.3: Spectra for Carbon after the four rotations: II, IX, IY, XX

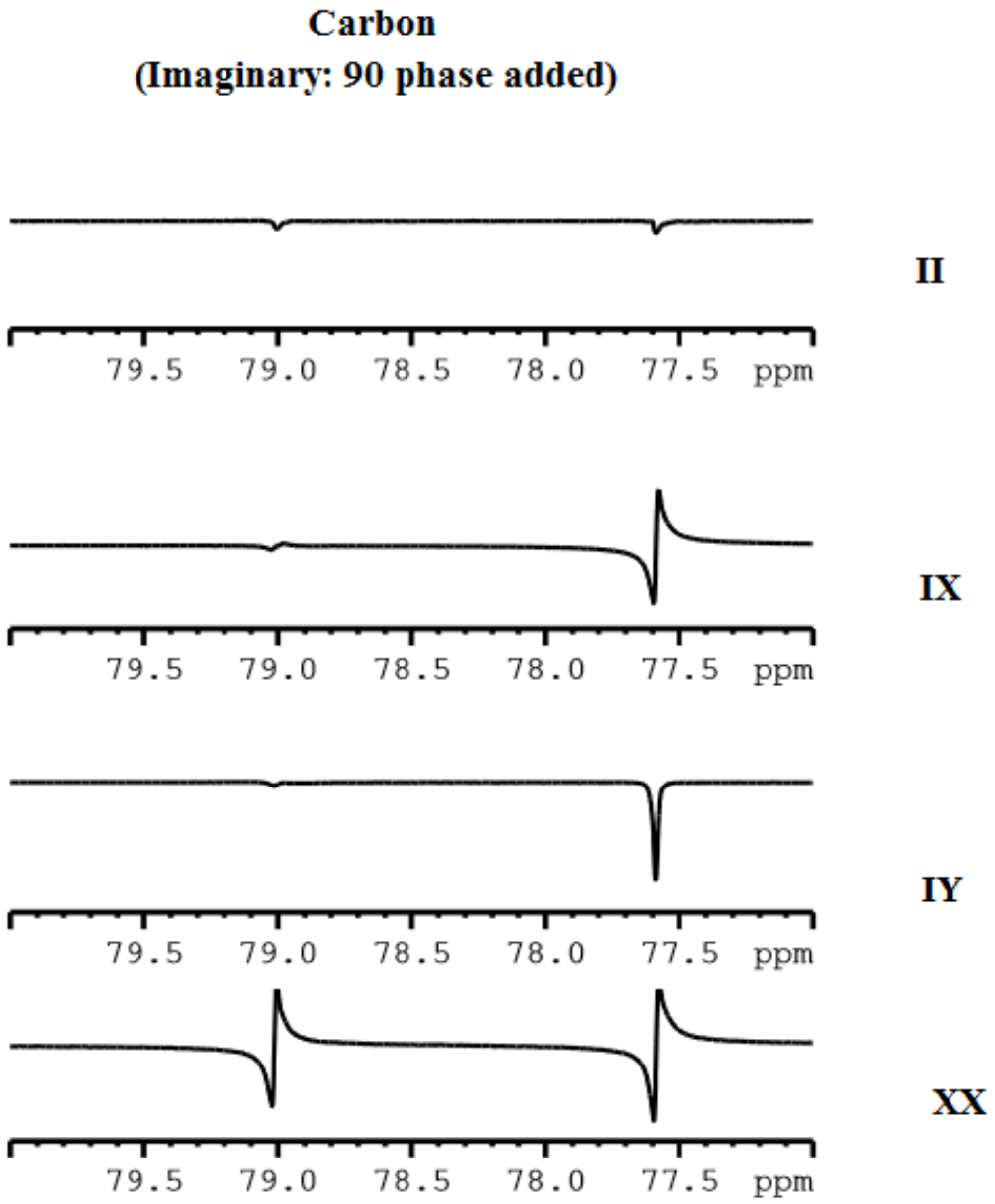


Figure 4.4: Spectra of Carbon after adding a phase of 90^0 for the four rotations: II, IX, IY, XX

Using the intensity values from the experiments mentioned above, we got the required number of equations to solve for the all the elements of density matrix. We

solved these equations numerically and got the following density matrix:

$$\rho = \begin{pmatrix} 0.0141 & 0.0020 + 0.0011i & 0.0023 - 0.0005i & 0 + 0.0032i \\ 0.0020 - 0.0011i & 0.9777 & 0 - 0.0109i & 0.0023 + 0.0005i \\ 0.0023 + 0.0005i & 0 + 0.0109i & 0.0037 & 0.0096 + 0.0114i \\ 0 - 0.0032i & 0.0023 - 0.0005i & 0.0096 - 0.0114i & 0.0046 \end{pmatrix} \tag{4.22}$$

On a 3D bar chart, the elements of density matrix can be visualized as:

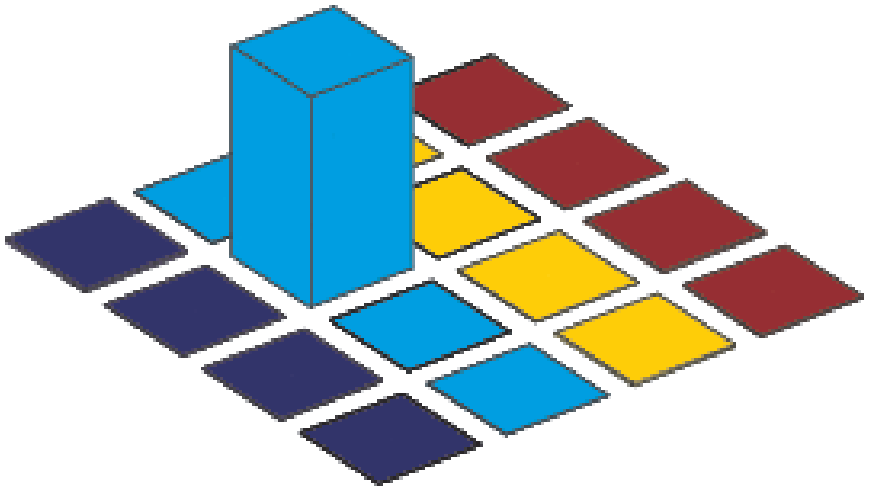


Figure 4.5: Quantum State Tomography: Real part of the elements of Density matrix

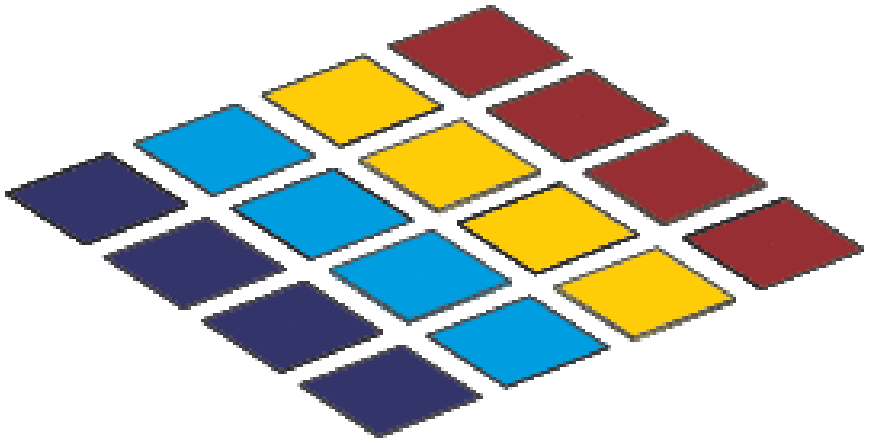
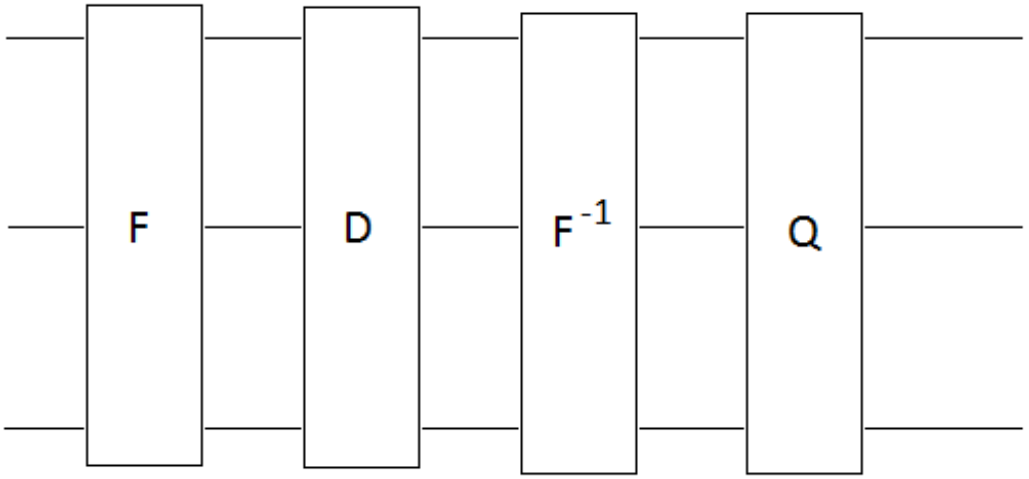


Figure 4.6: Quantum State Tomography: Imaginary part of the elements of Density matrix

Thus complete determination of the quantum state of both qubits is achieved.

4.6 Simulation of quantum tunneling on an NMR information processor

The theoretical protocol for simulation of tunneling on an NMR quantum computer is same as the one given in section 3.2. The figure 3.1 gives the circuit diagram for simulation quantum tunneling on a general n-qubit quantum computer. For ease, we have redrawn it here :



For a two qubit system we can write the quantum circuit for double well potential as:

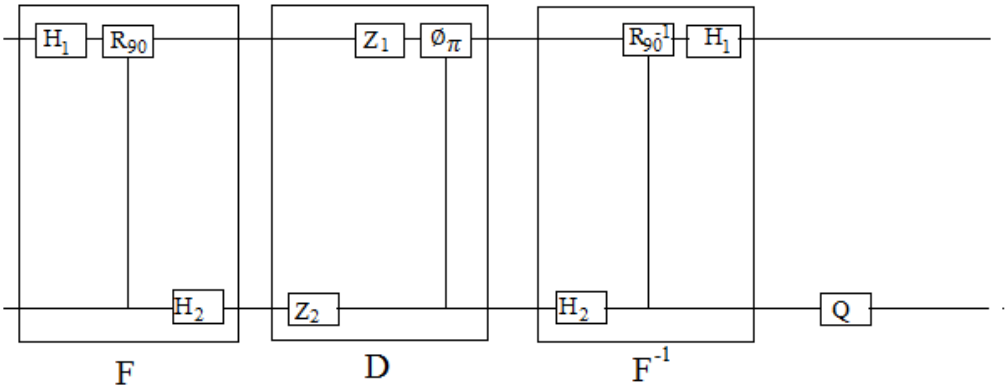


Figure 4.7: Quantum Circuit for two qubit simulation of quantum tunneling.

Here $Q = e^{-\iota v \sigma_z \triangle t}$ (notation are similar to the one used in chapter 3). The first block in the diagram is Fourier transform where R_{90} means conditional $90^{\o}$ rotation.

The third block is just the inverse of first block. The second block is the kinetic operator in the momentum basis. The three gates used in this block are [GF]:

$$\phi_\pi = e^{-\iota\frac{\pi^2}{2}\left(R_{\frac{\pi}{2}}\right)^2\triangle t} \tag{4.23}$$

$$Z_1 = e^{\iota\frac{\pi^2}{8}\sigma_z\otimes I\triangle t} \tag{4.24}$$

$$Z_2 = e^{\iota\frac{\pi^2}{2} I\otimes\sigma_z\triangle t} \tag{4.25}$$

Now comes the part of implementing the above circuit diagram on an NMR quantum computer. The radio frequency pulses can be used to implement the single qubit gates and J-coupling evolution combined with refocusing pulses can be used to implement two qubit gates. Q.F.T can be implemented by following sequence of pulses and time evolutions:

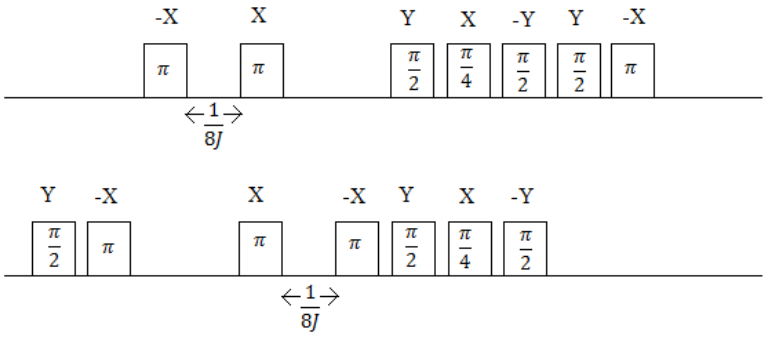


Figure 4.8: Pulse sequence for Quantum Fourier Transform gate.

The pulse sequence for the second block, i.e., Kinetic energy operator, is given by:

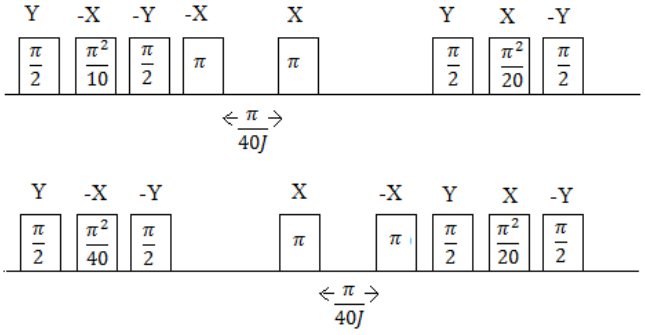


Figure 4.9: Pulse sequence for Kinetic energy gate.

The potential gate for double well potential can be implemented as:

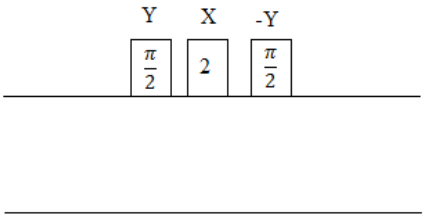


Figure 4.10: Pulse sequence for Potential energy gate.

For, other potentials (in 2 qubit cases) the structure of the gates other than the potential gate will be the same. The pulse sequences for the delta like potentials is more complicated as it requires control rotations. For potential gate in Dirac comb potential in 3 qubits, the same sequence needs to be applied on last qubit.

Appendix A

Derivation of Fourier coefficient α which emerges after the Phase estimation subroutine in Harrow's algorithm

Conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{\frac{\iota A\tau t_0}{T}}$ is applied on $|\psi\rangle \otimes |b\rangle$ in the "Phase estimation" subroutine of Harrow's algorithm and we get:

$$\left(\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{iA\tau t_0/T}\right)(|\psi_0\rangle \otimes |b\rangle) = \sum_{j=1}^{N}\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau|\psi_0\rangle e^{iA\tau t_0/T}\beta_j|u_j\rangle \tag{A.1}$$

$$= \sum_{j=1}^{N}\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \left(\sqrt{\frac{2}{T}}\sum_{\tau'=0}^{T-1} sin\left(\frac{\pi(\tau'+1/2)}{T}\right)|\tau'\rangle\right)\beta_j\left(e^{iA\tau t_0/T}|u_j\rangle\right) \tag{A.2}$$

$$= \sum_{j=1}^{N}\sum_{\tau=0}^{T-1}\sum_{\tau'=0}^{T-1}\sqrt{\frac{2}{T}} sin\left(\frac{\pi(\tau'+1/2)}{T}\right)|\tau\rangle\langle\tau|\tau'\rangle\beta_j e^{iA\tau t_0/T}|u_j\rangle \tag{A.3}$$

$$= \sum_{j=1}^{N}\sum_{\tau=0}^{T-1}\sqrt{\frac{2}{T}}\beta_j sin\left(\frac{\pi(\tau+1/2)}{T}\right)e^{i\lambda_j t_0/T}|\tau\rangle|u_j\rangle \tag{A.4}$$

$$\tag{A.5}$$

Inverse Fourier transform is defined as:

$$\sum_{j=1}^{N-1} x_j|j\rangle \rightarrow \sum_{k=1}^{N-1} y_k|k\rangle \tag{A.6}$$

where

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} \left(e^{-\iota 2\pi/N k_j}\right) x_j \tag{A.7}$$

Thus after applying inverse Fourier transform we get the state:

$$= \sum_{j=1}^{N} \sum_{\tau=0}^{T-1} \sqrt{\frac{2}{T}} \beta_j |u_j\rangle sin\left(\frac{\pi(\tau+1/2)}{T}\right) e^{i\lambda_j t_0/T} \frac{1}{\sqrt{T}} \left(\sum_{k=1}^{T-1} e^{\frac{-i2\pi}{T}\tau k} |k\rangle\right) \tag{A.8}$$

$$= \sum_{j=1}^{N} \sum_{\tau=0}^{T-1} \sum_{k=0}^{T-1} \sqrt{\frac{2}{T}} \frac{1}{\sqrt{T}} sin\left(\frac{\pi(\tau+1/2)}{T}\right) e^{\frac{-i2\pi\tau k}{T}} e^{i\lambda_j \tau t_0/T} \beta_j |k\rangle |u_j\rangle \tag{A.9}$$

$$= \sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j} \beta_j |k\rangle |u_j\rangle \tag{A.10}$$

where

$$\alpha_{k|j} = \sum_{\tau=0}^{T-1} \sqrt{\frac{2}{T^2}} sin\left(\frac{\pi(\tau+1/2)}{T}\right) e^{\frac{-i2\pi}{T}\tau k} e^{i\lambda_j \tau t_0/T} \tag{A.11}$$

$$\alpha_{k|j} = \sum_{\tau=0}^{T-1} \sqrt{\frac{2}{T^2}} sin\left(\frac{\pi(\tau+1/2)}{T}\right) e^{i\left(\frac{2\pi}{T}\tau k + \frac{\tau\lambda_j t_0}{T}\right)} \tag{A.12}$$

$$= \frac{\sqrt{2}}{T} \sum_{\tau=0}^{T-1} sin\left(\frac{\pi(\tau+1/2)}{T}\right) e^{\frac{i\tau}{T}(-2\pi k + \lambda_j t_0)} \tag{A.13}$$

$$= \frac{\sqrt{2}}{T} \sum_{\tau=0}^{T-1} \left(\frac{e^{i\left(\frac{\pi(\tau+1/2)}{T}\right)} - e^{-i\left(\frac{\pi(\tau+1/2)}{T}\right)}}{2i}\right) e^{\frac{i\tau}{T}(\lambda_j t_0 - 2\pi k)} \tag{A.14}$$

Put $\delta = \lambda_j t_0 - 2\pi k$

$$\alpha_{k|j} = \frac{1}{i\sqrt{2}T} \sum_{\tau=0}^{T-1} e^{i\tau\delta/T} \left(e^{i\left(\frac{\pi(\tau+1/2)}{T}\right)} - e^{-i\left(\frac{\pi(\tau+1/2)}{T}\right)}\right) \tag{A.15}$$

$$= \frac{1}{i\sqrt{2}T} \sum_{\tau=0}^{T-1} \left(e^{\frac{i2\tau\delta + i\pi 2\tau + i\pi}{2T}} - e^{\frac{i2\tau\delta - i\pi 2\tau - i\pi}{2T}}\right) \tag{A.16}$$

$$= \frac{1}{i\sqrt{2}T} \sum_{\tau=0}^{T-1} e^{\frac{\iota\pi}{2T}} e^{\iota\tau\left(\frac{\delta+\pi}{T}\right)} - e^{\frac{-\iota\pi}{2T}} e^{\iota\tau\frac{(\delta-\pi)}{T}} \tag{A.17}$$

As we know, the sum of a G.P. is given by-

$$\sum_{n=0}^{N-1} a_0 r^n = a_0 \frac{(1-r^n)}{1-r} \tag{A.18}$$

We get:

$$\alpha_{k|j} = \frac{1}{i\sqrt{2}T}\left(e^{\frac{\iota\pi}{2T}}\left(\frac{1-e^{\iota(\delta+\pi)}}{1-e^{\iota\frac{\delta+\pi}{T}}}\right) - e^{\frac{-\iota\pi}{2T}}\left(\frac{1-e^{\iota(\delta-\pi)}}{1-e^{\iota\frac{\delta-\pi}{T}}}\right)\right) \tag{A.19}$$

$$= \frac{1}{i\sqrt{2}T}\left(e^{\frac{\iota\pi}{2T}}\left(\frac{1+e^{\iota\delta}}{1-e^{\iota\frac{\delta+\pi}{T}}}\right) - e^{\frac{-\iota\pi}{2T}}\left(\frac{1+e^{\iota\delta}}{1-e^{\iota\frac{\delta-\pi}{T}}}\right)\right) \tag{A.20}$$

$$= \frac{1+e^{\iota\delta}}{i\sqrt{2}T}\left(\frac{e^{\iota\left(\frac{\delta+\pi}{2T}\right)}e^{-\iota\delta/2T}}{1-e^{\iota\frac{\delta+\pi}{T}}} - \frac{e^{\iota\left(\frac{\delta-\pi}{2T}\right)}e^{-\iota\delta/2T}}{1-e^{\iota\frac{\delta-\pi}{T}}}\right) \tag{A.21}$$

$$= \frac{1+e^{\iota\delta}}{i\sqrt{2}T}\left(\frac{e^{-\iota\delta/2T}}{e^{-\iota\left(\frac{\delta+\pi}{2T}\right)}\left(1-e^{\iota\frac{\delta+\pi}{T}}\right)} - \frac{e^{-\iota\delta/2T}}{e^{-\iota\left(\frac{\delta-\pi}{2T}\right)}\left(1-e^{\iota\frac{\delta-\pi}{T}}\right)}\right) \tag{A.22}$$

$$= \frac{1+e^{\iota\delta}}{i\sqrt{2}T}\left(\frac{e^{-\iota\delta/2T}}{e^{-\iota\left(\frac{\delta+\pi}{2T}\right)} - e^{\iota\frac{\delta+\pi}{2T}}} - \frac{e^{-\iota\delta/2T}}{e^{-\iota\left(\frac{\delta-\pi}{2T}\right)} - e^{\iota\frac{\delta-\pi}{2T}}}\right) \tag{A.23}$$

$$= \frac{e^{\iota\delta/2}\left(e^{-\iota\delta/2}+e^{\iota\delta/2}\right)e^{-\iota\delta/2T}}{\sqrt{2}T}\left(\frac{1}{2sin\left(\frac{\delta+\pi}{2T}\right)} - \frac{1}{2sin\left(\frac{\delta-\pi}{2T}\right)}\right) \tag{A.24}$$

$$= \frac{e^{\iota\delta/2(1-1/T)}}{\sqrt{2}T}cos(\delta/2)\left(\frac{sin\left(\frac{\delta-\pi}{2T}\right) - sin\left(\frac{\delta+\pi}{2T}\right)}{sin\left(\frac{\delta+\pi}{2T}\right)sin\left(\frac{\delta-\pi}{2T}\right)}\right) \tag{A.25}$$

$$= \frac{e^{\frac{\iota\delta}{2}(1-1/T)}}{\sqrt{2}T}cos(\delta/2)\left(\frac{2cos\left(\frac{\delta}{2T}\right)sin\left(\frac{\pi}{2T}\right)}{sin\left(\frac{\delta+\pi}{2T}\right)sin\left(\frac{\delta-\pi}{2T}\right)}\right) \tag{A.26}$$

$$\tag{A.27}$$

Put back $\delta = \lambda_j t_0 - 2\pi k$

$$\alpha_{k|j} = \frac{e^{\frac{\iota(\lambda_j t_0 - 2\pi k)}{2}(1-1/T)}}{\sqrt{2}T}cos((\lambda_j t_0 - 2\pi k)/2)\left(\frac{2cos\left(\frac{\lambda_j t_0 - 2\pi k}{2T}\right)sin\left(\frac{\pi}{2T}\right)}{sin\left(\frac{\lambda_j t_0 - 2\pi k + \pi}{2T}\right)sin\left(\frac{\lambda_j t_0 - 2\pi k - \pi}{2T}\right)}\right) \tag{A.28}$$

Thus

$$|\alpha_{k|j}| = \frac{\sqrt{2}}{T}\left(\frac{cos((\lambda_j t_0 - 2\pi k)/2)cos\left(\frac{\lambda_j t_0 - 2\pi k}{2T}\right)sin\left(\frac{\pi}{2T}\right)}{sin\left(\frac{\lambda_j t_0 - 2\pi k + \pi}{2T}\right)sin\left(\frac{\lambda_j t_0 - 2\pi k - \pi}{2T}\right)}\right) \tag{A.29}$$

Appendix B

Mathematica Codes

∗ProgramforQuantumTunnelingin$a$DoubleWellPotential∗

∗Programisgateresolved∗

∗TwoQubits∗

Needs["QDENSITYQdensity"]

EffectiveGate = Module[{}, $t = 0.1$; $v = 10$;

Gate1 = (had[1, 1] ⊗ Sigma0);

$$\text{Gate2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \text{Exp}\left[\frac{2*\pi*i}{4}\right] \end{pmatrix};$$

Gate3 = (Sigma0 ⊗ had[1, 1]);

$$\text{Z0} = \begin{pmatrix} \text{Exp}\left[\frac{i*\pi*\pi*t}{8}\right] & 0 \\ 0 & \text{Exp}\left[\frac{-i*\pi*\pi*t}{8}\right] \end{pmatrix};$$

Gate4 = (Z0 ⊗ Sigma0);

$$\text{Z1} = \begin{pmatrix} \text{Exp}\left[\frac{i*\pi*\pi*t}{4*8}\right] & 0 \\ 0 & \text{Exp}\left[-\frac{i*\pi*\pi*t}{4*8}\right] \end{pmatrix};$$

Gate5 = Chop[(Sigma0 ⊗ Z1)];

$$\emptyset_{01} = \text{Chop}\left[\text{MatrixExp}\left[\frac{-i*\pi*\pi*4*t}{8} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}\right]\right];$$

$\text{Gate6} = \emptyset_{01};$

$\text{Gate7} = \text{Gate3};$

$\text{Gate8} = \text{Adj}[\text{Gate2}];$

$\text{Gate9} = \text{Gate1};$

$$\text{Pot} = \begin{pmatrix} \text{Exp}[-i*v*t] & 0 \\ 0 & \text{Exp}[i*v*t] \end{pmatrix};$$

$\text{Gate10} = \text{Chop}[(\text{Sigma0} \otimes \text{Pot})];$

$\text{GateArray} = \{\text{Gate1}, \text{Gate2}, \text{Gate3}, \text{Gate4}, \text{Gate5}, \text{Gate6}, \text{Gate7}, \text{Gate8}, \text{Gate9}, \text{Gate10}\};$

$\text{SuperGate} = \text{Chop}[\text{Gate2.Gate1}];$

$\text{For}[j = 1, j \leq 8, j{+}{+},$

$\text{SuperGate} = \text{Chop}[\text{GateArray}[[j+2]].\text{SuperGate}]$

];

SuperGate

];

$\text{Module}[\{\}, \text{state0} = (\text{Ket}[0] \otimes \text{Ket}[1]); \text{density0} = (\text{state0} \otimes \text{Adj}[\text{state0}]);$

$$\text{Endstates} = \left\{\begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\end{pmatrix}\right\};$$

Endstates[[1]] = state0;

For[$l = 1, l \leq 10, l{+}{+}$,

Endstates[[$l + 1$]] = Chop[SuperGate.Endstates[[l]]]

];

$$\text{Enddensity} = \left\{ \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \right.$$
$$\begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix},$$
$$\left. \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \right\};$$

Enddensity[[1]] = density0;

For[$l = 1, l \leq 10, l{+}{+}$,

Enddensity[[$l + 1$]] = Chop[SuperGate.Enddensity[[l]].Adj[SuperGate]]

];

];

Module[{},

$$\text{prob} = \left\{ \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \right.$$

$$\left\{\begin{pmatrix}0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\end{pmatrix}\right\};$$

For[$j = 1, j \leq 11, j{+}{+}$,

For[$k = 1, k \leq 4, k{+}{+}$,

prob[[j]][[k]] = (Abs[Endstates[[j]][[k]]])^2

]

];

]

For[$i = 1, i \leq 11, i{+}{+}$,

p = ListPlot[Transpose[prob[[i]]], Filling → Axis, PlotRange → $\{0, 1\}$,

Frame → True, FillingStyle → {Red},

FrameTicks → $\{\{\{0, 0.25, 0.5, 0.75, 1\}, \text{None}\}, \{\{1, 2, 3, 4, 5, 6, 7, 8\}, \text{None}\}\}$;

Print[p]]

Three qubits

∗ProgramforQuantumTunnelinginaDoubleWellPotential∗

∗Three Qubits∗

Needs["QDENSITYQdensity"]

Module[{}, $L = 3$; PotQubit = 2; $v = 10$; $t = 0.1$; $m = 0.5$];

Module[{}, state0 = Ket[1];

For[$i = 2, i \leq L, i{+}{+}$, If[i == PotQubit, state0 = (state0 ⊗ Ket[1]),

state0 = (state0 ⊗ Ket[0])]]; density0 = (state0 ⊗ Adj[state0])];

ControlP[L_, q1_, q2_, k_]:=

Module $\left[\{\phi\}, \phi = \text{Exp}\left[2 * \pi * i \,/ 2^{k}\right];\right.$

```
1/4 * ((3 + ϕ) * TwoOp[L, q1, q2, 0, 0] + (1 - ϕ) * TwoOp[L, q1, q2, 3, 0]+
(1 - ϕ) * TwoOp[L, q1, q2, 0, 3] + (ϕ - 1) * TwoOp[L, q1, q2, 3, 3])];

QFTGate = Module[{},
Ω = IdentityMatrix[2^L]; For[i = 1, i ≤ L, i++,
Ω = had[L, i].Ω;
For[j = i + 1, j ≤ L, j++,
    Ω = ControlP[L, j, i, j - i + 1].Ω
]
]; Ω];

InverseQFTGate = Adj[QFTGate];

PotentialGate = Module [{}, Pot = ( Exp[-i * v * t]      0
                                    0                    Exp[i * v * t] ) ;
PotGate = IdentityMatrix[2];
If[PotQubit==1, For[i = 1, i ≤ L, i++,
If[i == 1, PotGate = Pot, PotGate = (PotGate ⊗ Sigma0)]
], For[i = 2, i ≤ L, i++,
If[i == PotQubit, PotGate = (PotGate ⊗ Pot), PotGate = (PotGate ⊗ Sigma0)]
]]; Chop[PotGate]
];

KineticEnergyGate = Module[{}, MomOp = IdentityMatrix[2^L];
For[i = 0, i ≤ 2^(L - 1), i++,
MomOp[[i + 1]][[i + 1]] = 2* π / 2^L * i] ;
For[i = 2^(L - 1) + 1, i ≤ (2^L) - 1, i++,
MomOp[[i + 1]][[i + 1]] = 2* π / 2^L * (2^(L - 1) - i)] ;
MatrixExp [-i * Chop [MatrixPower[MomOp,2] / 2*m] * t]
];
```

```
SuperGate = Chop[PotentialGate.InverseQFTGate.KineticEnergyGate.QFTGate];

Module[{}, EndVectorTemp = Range[2^L]; EndVectorTemp = EndVectorTemp - EndVectorTemp;
Endstates = {EndVectorTemp, EndVectorTemp, EndVectorTemp, EndVectorTemp,
EndVectorTemp, EndVectorTemp, EndVectorTemp, EndVectorTemp, EndVectorTemp,
EndVectorTemp, EndVectorTemp};
Endstates[[1]] = state0;
For[l = 1, l <= 10, l++,
Endstates[[l + 1]] = Chop[SuperGate.Endstates[[l]]]
];
EndMatrixTemp = IdentityMatrix[2^L];
Enddensity = {EndMatrixTemp, EndMatrixTemp, EndMatrixTemp, EndMatrixTemp,
EndMatrixTemp, EndMatrixTemp, EndMatrixTemp, EndMatrixTemp, EndMatrixTemp,
EndMatrixTemp, EndMatrixTemp};
Enddensity[[1]] = density0;
For[l = 1, l <= 10, l++,
Enddensity[[l + 1]] = Chop[SuperGate.Enddensity[[l]].Adj[SuperGate]]
];
];

Module[{},
prob = {EndVectorTemp, EndVectorTemp, EndVectorTemp, EndVectorTemp,
EndVectorTemp, EndVectorTemp, EndVectorTemp, EndVectorTemp, EndVectorTemp,
EndVectorTemp, EndVectorTemp};
For[j = 1, j <= 11, j++,
For[k = 1, k <= 2^L, k++,
prob[[j]][[k]] = (Abs[Endstates[[j]][[k]]])^2
]
```

```
];
]
For[i = 1, i ≤ 11, i++,
p = ListPlot[Transpose[prob[[i]]], Filling → Axis, PlotRange → {0, 1},
Frame → True, FillingStyle → {Red},
FrameTicks → {{{0, 0.25, 0.5, 0.75, 1}, None}, {{1, 2, 3, 4, 5, 6, 7, 8}, None}}];
Print[p]]
```

Most of the structure of the programs for other potentials is same as that of program for Double Potential except for the important changes in the potential gate structure which are listed below:

```
(*Single Dirac Delta Potental *)
Module[{}, L = 2; PotState = 3; v = 10; t = 0.1; m = 0.5];
PotentialGate = Module[{}, Pot = IdentityMatrix[2^L];
For[i = 1, i ≤ 2^L, i++, If[i == PotState, Pot[[i]][[i]] = 1, Pot[[i]][[i]] = −1]];
Chop[MatrixExp[−i ∗ v ∗ t ∗ Pot]]
];
(*Double Dirac Delta Potental *)
Module[{}, L = 3; PotQubit = 0; PotState1 = 3; PotState2 = 5; v = 10; t = 0.2; m = 0.5];
PotentialGate = Module[{}, Pot = IdentityMatrix[2^L];
For[i = 1, i ≤ 2^L, i++, If[i == PotState1||i == PotState2, Pot[[i]][[i]] = 1,
Pot[[i]][[i]] = −1]];
Chop[MatrixExp[−i ∗ v ∗ t ∗ Pot]]
];
(*Tripple DiracDeltaPotental − SuperpositionStart*)
```

```
Module[{}, L = 3; PotQubit = 0; PotState1 = 3; PotState2 = 4; PotState3 = 5;
pm1 = 1; pm2 = 1; pm3 = 1; v = 10; t = 0.2; m = 0.5];
Module [{}, state0 = (KetV[{0,0,0}]+KetV[{0,0,1}])/Sqrt[2]; density0 = (state0 ⊗ Adj[state0])]
PotentialGate = Module[{}, Pot = -1 * IdentityMatrix[2^L];
Pot[[PotState1]][[PotState1]] = pm1;
Pot[[PotState2]][[PotState2]] = pm2;
Pot[[PotState3]][[PotState3]] = pm3;
Chop[MatrixExp[-i * v * t * Pot]]
];

(*Dirac Comb Potential*)
Module[{}, L = 3; PotQubit = 3; v = 10; t = 0.1; m = 0.5];
PotentialGate = Module [{}, Pot = ( Exp[-i * v * t]      0
                                    0                    Exp[i * v * t] );
PotGate = IdentityMatrix[2];
If[PotQubit==1, For[i = 1, i ≤ L, i++,
If[i == 1, PotGate = Pot, PotGate = (PotGate ⊗ Sigma0)]
], For[i = 2, i ≤ L, i++,
If[i == PotQubit, PotGate = (PotGate ⊗ Pot), PotGate = (PotGate ⊗ Sigma0)]
]]; Chop[PotGate]
];
```

Bibliography

[AWHL09] A. Hassidim, A. W. Harrow, and S. Llyod, *Quantum algorithm for linear systems of equations*, Phys. Rev. Lett. **103** (2009), 150502.

[BD06] F. Tabakin, B.J. Diaz, J.M. Burdis, *Qdensity- a mathematica quantum computer simulation*, Comp.Phy.Comm. **174** (2006), 914–934.

[Cop94] D. Coppersmith, *An approximate fourier transform useful in quantum factoring*, IBM Research Report (1994), RC19642.

[DGCH97] A. F. Fahmy, D. G. Cory, and T.F. Havelr, *Ensemble quantum computing by NMR-spectroscopy*, Proc. Nat.Ac. of Sci., USA **94** (1997), 1634–1639.

[DiV] D. P. DiVincenzo, *The physical implementation of quantum computation*, Physik **48**, 771–783.

[EKL98] I. L. Chuang, E. Knill and R. Laflamme, *Effective pure states for bulk quantum computation*, Phys. Rev. A **57** (1998), 3348.

[Fey09] R.P Feynman, *Simulating physics with computers*, Int. J. Theor.Phys. **21** (2009), 224102.

[GF] L. Hao, F.H. Zhang, G.L. Long, G.R. Feng, Y. Lu, *Experimental simulation of quantum tunneling in small systems*, Sci.Rep. **3**, 2232.

[GLL] Y. Sun, G. L. Long, H. Y. Yan, *Analysis of density matrix reconstruction in NMR quantum computing*, J. Opt. B: Quantum Semiclass. Opt. **3**.

[GR02] L. Grover and T. Rudolph, *Creating superpositions that correspond to efficiently integrable probability distributions*, arXiv:[quant-ph] (2002), 0208112v1.

[IOF11] T. Bonagamba, E. Azevedo, I. Oliveira, R. Sarthour Jr. and J. C. C. Freitasr, *NMR quantum information processing*, 2011.

[JAJM] R. H. Hansen, J. A. Jones, and M. Mosca, *Quantum logic gates and nuclear magnetic resonance pulse sequences*, J. Magn. Reson. **135**.

[Jon01] J. A. Jones, *Quantum computing and nuclear magnetic resonance*, PhysChemComm **11** (2001).

[Jon11] J. A. Jones, *Quantum computing with NMR*, Prog. NMR Spectrosc. **59** (2011), 91–120.

[Jor05] S. P. Jordan, *Fast quantum algorithm for numerical gradient estimation*, Phys. Rev. Lett. **95** (2005), 050501.

[JP14] X. Yao, Z. Li, C. Ju, H. Chen, X. Peng, S. Kais, J. Du J. Pan, Y. Cao, *Experimental realization of quantum algorithm for solving linear system of equations*, Phys. Rev. A. **89** (2014), 022313.

[KAG09] I. Kassal and A. Aspuru-Guzik, *Quantum algorithm for molecular properties and geometry optimization*, J. Chem. Phys. **131** (2009), 224102.

[KDK00] K. Dorai, T. S. Mahesh, Arvind and A. Kumar, *Quantum computation using NMR*, Curr. Sci. **79** (2000), 10.

[Kee10] J. Keeler, *Understanding NMR spectroscopy*, John Wiley and Sons, 2010.

[Lee] J. S. Lee, *The quantum state tomography on an NMR system*, Phy. Lett. A. **305**, 349–353.

[LO08] S. K. Leyton and T. J. Osborne, *A quantum algorithm to solve nonlinear differential equations*, arXiv:[quant-ph] (2008), 0812.4423v1.

[NC00] M. A. Nielsen and I. L. Chuang, *Quantum computation and Quantum information*, Cambridge University Press, Cambridge, England, 2000.

[Sak94] J. J. Sakurai, *Modern quantum mechanics*, Pearson Education, Inc., 1994.

[S.L92] S.Llyod, *Universal quantum simulators*, Science **273** (1992), 1073.

[Sor] A.T. Sornborgor, *Quantum simulation of tunneling in small systems*, Sci.Rep. **2**, 597.

[SS] A. N. Soklakov and R. Schack, *Efficient state preparation for a register of quantum bits*, Phys. Rev. A **73**, 012307.

[SS08] J. Stolze and D. Suter, *Quantum computing : A short course from theory to experimentl*, WILEY-VCH, Weinheim, 2008.

[Ste97] A. Steane, *Quantum computing*, arXiv:[quant-ph] (1997), 9708022v2.

[THJJ14] S. R. Clark, T. H. Johnson and D. Jaksch, *What is quantum simulator?*, arXiv:[quant-ph] (2014), 1405.2831v1.

[Tra12] A. Trabesinger, *Quantum simulation*, N. Physics **8** (2012), 263.

[YCK] S. Frankel Y. Cao, A. Daskin and S. Kais, *Quantum circuits for solving linear systems of equations*, arXiv:[quant-ph], 1110.2232v3.

[YCK13] I. Petras, J. Traub, Y. Cao, A. Papageorgiou and S. Kais, *Quantum algorithm and circuit design solving the poisson equation*, New J. Phys. **15** (2013), 0130210.

[Zal98] C. Zalka, *Simulating quantum systems on quantum computer*, Proc. R. Soc. Lond. A **454** (1998), 313–322.