IMPLEMENTATION OF QUANTUM GATE OPERATIONS USING A DYNAMIC LEARNING ALGORITHM

A Dissertation by

Rudrayya Chowdary Garigipati

Master of Science, Wichita State University, 2008

Bachelor of Technology, JNT University, 2006

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

May 2015

IMPLEMENTATION OF QUANUM GATE OPERATIONS USING A DYNAMIC LEARNING ALGORITHM

The following faculty members have examined the final copy of this dissertation for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Doctor of Philosophy with a major in Electrical Engineering.

DEDICATION

To My Grand Parents

ACKNOWLEDGEMENTS

The pleasure of completing any task successfully would be done only by showing gratitude to the people who made it achievable. The dissertation work that I submit today could not have been completed without the support of many people. There is a lot of inspiration which made the effort come out. I want to thank and express gratitude to one and all who made this work successful and ingenious. I acknowledge and am grateful for everyone who have guided and encouraged me in this effort.

First and foremost, I would like to thank Almighty God for giving me the strength to complete my research in time.

I take this opportunity to articulate my appreciation and thanks to my advisor Dr. Preethika Kumar who has forever been a wonderful source of motivation and precious help in channeling my hard work in the right path. I learned a lot from just being around her. She has always made time to help me despite her busy schedules and has always been very supportive and understanding. I truly feel blessed to work under her, and will forever be thankful to her for helping me to finish this work.

I would like to thank Dr. Steven Skinner for his valuable guidance, support and help. Without his guidance, this work would not have been possible. His valuable suggestions and steady support helped me in giving wholeness to this research work.

I would like to thank my other committee members, Dr. Vinod Namboodiri, Dr. Visvakumar Aravinthan and Dr. Janet Twomey for all the encouragement, help and support they have provided me both professionally and personally to complete this work.

I am grateful to all my professors who have made valuable contributions to my professional and personal life during my stay at Wichita State University. I would like to thank

Dr. John Watkins for giving me the opportunity to work as a Teaching Assistant in the Department of Electrical Engineering and Computer Science, which helped me to realize one of my childhood dreams. I would also like to thank Judie Dansby for giving me the opportunity to work for her, and also for assisting me with the administrative tasks necessary for completing my doctoral program.

I would like to thank my parents, my sister, my brother-in-law, my nephew and my extended family (especially to ammamma, nannamma, attha, mavayyagaru, pedanannagaru, peddamma and cousins) for all of their love, prayers and constant support. Thanks for always being there for me and encouraging me to keep following my dreams. Their support helped me to achieve all that I could in my life. I would like to thank all my cousins and friends back home, Surya, Vamsi, Ramakrishna, Kiran, Narayana, who were always there cheering me. They have always stood by me through the good and bad times. My time at Wichita was made enjoyable in large part due to my friends that became a part of my life. I am grateful for time spent with my present and past roommates, just to name a few, SRK, Siva, Sreedhar, Sandeep, Sailesh, Suman, Kamesh, Ram and Venu. Thanks for all the support and well wishes in the entire process.

I would also like to take this opportunity to thank my late grandfather, Dr. Rudrayya Chowdary Garigipati, whom I admire the most in each and every task I take up in my life, for inspiring me, and also, for always being there with me with all his teachings and memories.

Last but not the least, I would like to convey my gratefulness to all those who have directly or indirectly assisted me on any occasion throughout this research.

ABSTRACT

Quantum circuits can be implemented by breaking them into a series of elementary gates, and several methods have been proposed to decompose a quantum circuit into elementary gate operations. The total number of gates required to implement the circuit is called the gate count, which gives a measure of the computational overhead of the circuit. In order to take advantage of the quantum effects of a quantum system, it is desired that all gate operations required to implement the quantum circuit be performed before the system loses its coherence. However, as the complexity of a quantum circuit increases, the gate count increases, and it becomes difficult to implement the circuit before losing its coherence. Therefore, strategies need to be designed that minimize the total gate count, and hence, the total time of operation relative to the system.

Quantum gate operations can be realized by finding the parameters of the system Hamiltonian. (The Hamiltonian represents total energy of the system). This can be done by equating the matrix representations for the desired gate operations to the unitary transformation under the Hamiltonian. Here, we proposed a method for finding system parameters for implementing gate operations along a linear array of qubits. A dynamic learning algorithm has been used as a tool for finding the system parameters to implement the desired gate operations *directly* wherein gate operation need not be decomposed into a sequence of elementary gate operations belonging to a universal set. Therefore, the main advantage of this scheme is that the computational overhead of a quantum circuit can be reduced. Moreover since all qubits are involved in the gate operation, and no qubits are idle, our scheme also eliminates other problems like relative phases, which are picked up due to individual qubit precessions during idle times. Another advantage is that all the parameters found using this scheme are scalable and, therefore, can be adjusted to the requirements of a given experimental realization.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF TABLES

LIST OF TABLES (continued)

LIST OF TABLES (continued)

CHAPTER 1

MOTIVATION

1.1 Introduction

A quantum computer is a device that uses quantum mechanical phenomena, such as superposition and entanglement, to perform operations on data. The most fundamental building block of a classical computer is the bit. A single classical bit is capable of storing one piece of information; it can be in one of the two binary values, 0 or 1. A *quantum bit* or *qubit* is the quantum analogue of a classical bit. Like a classical bit, a qubit can also be in one of the two states, $|0\rangle$ or $|1\rangle$ (known as computational basis states). However, the difference between bits and qubits is that a qubit can exist in a linear combination of the two states. This state is called as 'superposition state':

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

where α and β are the probability amplitudes of finding the qubit in the $|0\rangle$ and $|1\rangle$ states, respectively. Here, the two basis states, $|0\rangle$ and $|1\rangle$, form a computational basis for the two-dimensional vector space representing the qubit, and they are defined as follows:

$$|0\rangle = \binom{1}{0}; \quad |1\rangle = \binom{0}{1}; \tag{1.2}$$

When measured, the superposition state given by equation (1.1) collapses to one of the two basis states, $|0\rangle$ or $|1\rangle$, with the probabilities of $|\alpha|^2$ and $|\beta|^2$, respectively, where $|\alpha|^2 + |\beta|^2 = 1$. This means that if we measure identically prepared quantum states, we might get different results ($|0\rangle$ or $|1\rangle$) for each measurement. However, the probabilities of measuring $|0\rangle$ or $|1\rangle$ converge to $|\alpha|^2$ and $|\beta|^2$. As a result, we can only predict the probability of finding a quantum state in one of the basis states. Superposition states allow quantum computers to perform the same operation on multiple input states at the same time, which is known as *quantum parallelism*. In general, a quantum computer with N qubits can simultaneously manipulate 2^N numbers. This is one of the advantages of quantum computers over classical computers, which allows them to efficiently solve certain problems, like factoring [1], which are very hard to solve on a classical computer. An example of a qubit is electron spin, where the spin states (spin up and spin down) form the two basis states. Recently, many systems with different technologies have been proposed. Some of these systems use nuclear spins [2-4], electrons and spin states [5-9], ion traps [10, 11], atoms [12] and charge and flux states of super conducting interface devices (SQUIDS) [13-21].

A quantum computer comprises of several qubits interacting with each other. Manipulations are performed on single and multiple qubits to realize different unitary operations. These unitary operations are called "gates". Following are examples of some of the basic gate operations used in quantum computing.

Single Qubit Gate Operations

NOT Gate or X-Gate:

A quantum NOT gate is the quantum analogue of the classical NOT gate. This is a single qubit gate operation. Under the NOT gate operation, the qubit flips its state from $|0\rangle$ to $|1\rangle$, and vice versa. Hence, the NOT gate on a qubit, Q, can be described as:

$$X\,|q\rangle = |q'\rangle, \tag{1.3}$$

where $|q\rangle$ is the state of a qubit Q, and $|q'\rangle$ is the complement state of $|q\rangle$, where a $|0\rangle$ is replaced by $|1\rangle$, and vice versa. The matrix representation of the NOT gate is

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.4}$$

Phase-flip or Z-Gate:

Under the quantum phase-flip gate operation, the qubit gains a phase of $\pm 180°$ if it is in the $|1\rangle$ state. When the qubit is in the $|0\rangle$ state, then it does not change its phase. Hence, the Phase-flip gate on a qubit, Q can be described as:

$$Z|0\rangle = |0\rangle$$

$$Z|1\rangle = -|1\rangle \qquad (1.5)$$

This gate will leave the state $|0\rangle$ unchanged while a phase factor of '$e^{i\pi}$' is picked up by the state $|1\rangle$. The matrix representation of the NOT gate is

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad (1.6)$$

Hadamard or H-Gate:

A quantum Hadamard gate is one of the most useful single qubit gate operations, since it is able to create a super-position state. This gate can be described as a 'square-root of NOT' gate. The gate operation can be described as:

$$H\,|\mathrm{q}\rangle = \begin{cases} \frac{|0\rangle+|1\rangle}{\sqrt{2}}, & if\ |\mathrm{q}\rangle = |0\rangle \\ \frac{|0\rangle-|1\rangle}{\sqrt{2}}, & if\ |\mathrm{q}\rangle = |1\rangle \end{cases} \qquad (1.7)$$

where $|q\rangle$ is the state of the qubit Q. If the initial state is $|0\rangle$, then the Hadamard gate will change it to state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, which is a superposition state of the basis states $|0\rangle$ and $|1\rangle$. Likewise, if the initial state is $|1\rangle$, then it will change it to state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, which is also a superposition state of the basis states $|0\rangle$ and $|1\rangle$. The matrix representation of the Hadamard gate is

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad (1.8)$$

Multiple Qubit Gate Operations

Controlled-NOT (CNOT) Gate:

A CNOT is a two qubit gate operation where one qubit (Q_1) acts as a control and the other (Q_2) acts as a target. Under the CNOT gate operation (Figure 1.1), the target qubit flips its state when the control qubit is in the $|1\rangle$ state. When the control qubit is in the $|0\rangle$ state, the target does not change its state.

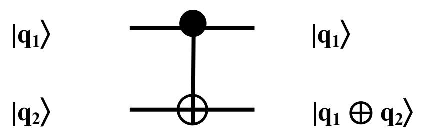

Figure 1.1. Implementation of CNOT gate operation between qubits, Q_1 and Q_2, with qubit Q_1 as control and qubit Q_2 as target. Here, $|q_1\rangle$ and $|q_2\rangle$ correspond to the states of qubits Q_1 and Q_2.

The CNOT gate operation on qubits Q_1 and Q_2, with Q_1 as a control and Q_2 as a target, can be described as:

$$|q_1q_2\rangle \rightarrow \begin{cases} |q_1q_2\rangle, \ if \ |q_1\rangle = |0\rangle \\ |q_1q_2{}'\rangle, \ if \ |q_1\rangle = |1\rangle \end{cases} \tag{1.9}$$

where $|q_i\rangle$ corresponds to state of qubit Q_i, i = 1 and 2, and $|q_2'\rangle$ is the complement of state $|q_2\rangle$ (for instance, if $|q_2\rangle = |1\rangle$, $|q_2'\rangle = |0\rangle$). Here, the matrix representation of the CNOT gate is

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.10}$$

Swap Gate:

A swap gate is a two qubit gate operation which is used to interchange the states of the two qubits, Q_1 and Q_2. The transformation is:

$$|q_1q_2\rangle \rightarrow |q_2q_1\rangle \tag{1.11}$$

where $|q_i\rangle$ corresponds to state of qubit Q_i, i = 1 and 2. Figure 1.2 shows the implementation of the swap gate.

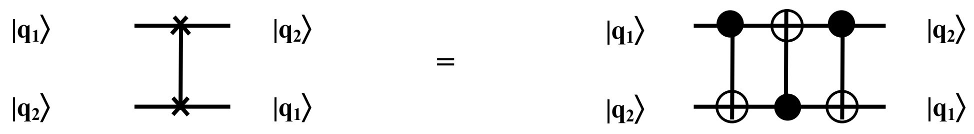

Figure 1.2. Implementation of the Swap gate operation between qubits Q_1 and Q_2 using conventional methods where the states of qubits Q_1 and Q_2 are swapped using three CNOT gates. Here, $|q_1\rangle$ and $|q_2\rangle$ correspond to the states of qubits Q_1 and Q_2, respectively.

As shown in Figure 1.2, a swap gate can be implemented by using three CNOT gate operations. From Figure 1.2, we can see that for the first and third CNOT gates, qubit Q_1 is the control and qubit Q_2 is the target. For the second CNOT gate, qubit Q_2 is the control and qubit Q_1 is the target. For instance, suppose we want to swap the states of qubits Q_1 and Q_2 which are in the initial states $|1\rangle$ and $|0\rangle$, respectively. Figure 1.3 shows the output states of qubits Q_1 and Q_2 after each gate operation, and we can see that after the implementation of the third CNOT gate, the qubits are now in states $|0\rangle$ and $|1\rangle$, respectively.

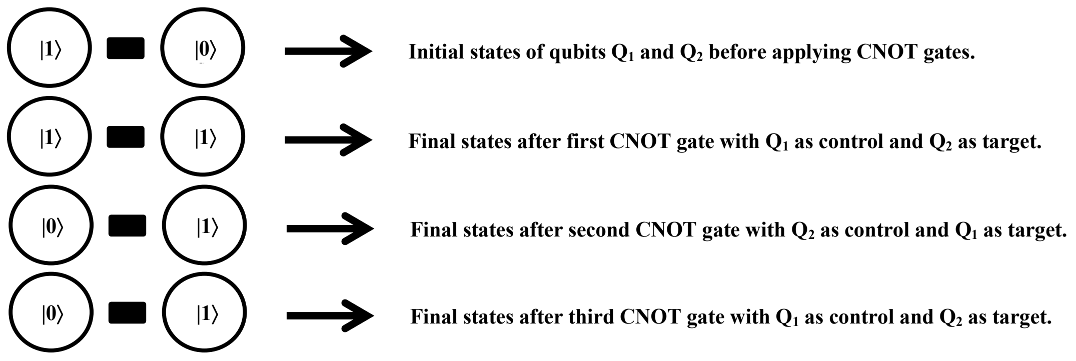
$|1\rangle$ $|0\rangle$
Initial states of qubits Q_1 and Q_2 before applying CNOT gates.
$|1\rangle$ $|1\rangle$
Final states after first CNOT gate with Q_1 as control and Q_2 as target.
$|0\rangle$ $|1\rangle$
Final states after second CNOT gate with Q_2 as control and Q_1 as target.
$|0\rangle$ $|1\rangle$
Final states after third CNOT gate with Q_1 as control and Q_2 as target.

Figure 1.3. Implementation of Swap gate operation using 3 CNOT gates.

The matrix representation of a swap gate can be described as

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

Toffoli Gate:

A Toffoli gate is a three qubit gate operation with two control qubits (Q_1 and Q_2) and one target qubit (Q_3). Here, the target qubit Q_3 flips its state only when both the control qubits Q_1 and Q_2 are in the $|1\rangle$ state. The transformation is:

$$|q_1 q_2 q_3\rangle \to \begin{cases} |q_1 q_2 q_3'\rangle, & if\ q_1 = q_2 = |1\rangle \\ |q_1 q_2 q_3\rangle, & otherwise \end{cases} \tag{1.13}$$

where $|q_i\rangle$ corresponds to state of qubit Q_i, i = 1 to 3, and $|q_3'\rangle$ is the complement of state $|q_3\rangle$ (for instance, if $|q_3\rangle = |1\rangle$, $|q_3'\rangle = |0\rangle$). Figure 1.4 shows the implementation of a Toffoli gate operation on qubits Q_1, Q_2 and Q_3, with qubits Q_1 and Q_2 as controls and Q_3 as target.

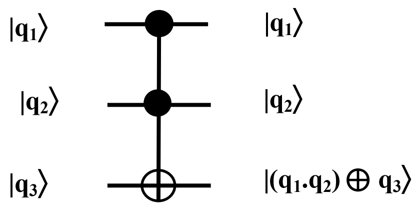

Figure 1.4. Implementation of Toffoli gate operation between qubits, Q_1, Q_2 and Q_3, with qubits Q_1 and Q_2 as controls, and qubit Q_3 as target. Here, $|q_1\rangle$, $|q_2\rangle$ and $|q_3\rangle$ correspond to the states of qubits Q_1, Q_2 and Q_3, respectively.

1.2 Research Objective and Organization of the Dissertation

A quantum circuit can be implemented by breaking it into a series of elementary gate operations, and in 1995, Barenco, *et al.,* showed that arbitrary gates can be constructed with a set of universal gates [22]. They showed that any two-qubit controlled-unitary operation can be achieved using a combination of single qubit gates and CNOT gates. In [23], the authors' found the minimal number of elementary gate operations (six elementary gates) required to implement any controlled-unitary operation. Since then, other methods have been proposed to decompose quantum circuits into these elementary gate operations [24-31]. Also, for any quantum system, gate operations must be performed before the system loses its coherence (quantum nature). As the number of gate operations to implement the quantum circuit increases, the gate count, and therefore, the associated time taken to complete the operation, increases. Hence, it is desired to minimize the total gate count of an operation, which may be achieved by minimizing the number of elementary gates used during the operation. Therefore, methods for implementing gate operations efficiently in quantum systems is an important area of research, and a number of schemes have been proposed to reduce the gate count. (Throughout this work, we use gate count, which is the number of elementary gates needed to implement a computation, as a measure of the computational overhead).

In this dissertation, we propose a circuit reduction scheme that allows us to implement a series of gate operations *directly*. For this, we use a dynamic learning algorithm based on back propagation technique to train a neural network (NN), to find system parameters for implementing the desired operation. Here, the main aim of our scheme is to find the system parameters to realize an equivalent single gate operation for the desired quantum algorithm.

In our scheme, we first choose a Hamiltonian, and then train its parameters to implement the desired gate operation within a chosen time duration. Once we know the parameter values (after training using the dynamic learning algorithm), to implement the desired gate operation, we simply set the system parameters to the values solved for, and then allow the system to evolve for a chosen interval of time (gate time obtained from training). For instance, to implement a swap gate using conventional methods, we need 3 CNOT gates. Hence, by implementing a swap gate as a single gate without breaking it into 3 CNOTs, the gate count can be reduced by one-third. This will especially be significant in moving qubits along linear nearest neighbor (LNN) arrays using swap operations. Likewise, by implementing a CNOT gate directly between two uncoupled next-to-nearest-neighbor qubits, we can realize the gate operation by skipping over the qubit in between them. Since a CNOT gate forms a universal gate set with single qubit unitary operations, the gate count for arbitrary unitary two and three-qubit operations along the array can be reduced. Thus, by combining these two operations, the overall computational overhead of implementing a gate operation using our scheme can be greatly reduced. Another key advantage of this scheme is that, since all qubits along the chain participate in the gate operation, we can avoid idle times that can give rise to relative phases.

In our research, we focus on the following types of operations:

(a) Implementing multi-qubit gate operations like CNOT, Toffoli and Swap between two next-to-nearest neighbor qubits *directly* without requiring the qubits to be adjacent to each other.

(b) Using our dynamic learning algorithm to derive an analytical solution for implementing gate operations. For this, we chose mirror inverse gate operations in LNN arrays with Ising-type interactions. Initially, we showed that our learning algorithm can be used to find the parameters to implement different mirror inverse gate operations *directly* in an LNN array, without decomposing them into series of gate operations. Later, by using these parameters, we found an analytical solution to implement a mirror inverse gate operation in an LNN array of arbitrary length.

(c) Using our dynamic learning algorithm to find the parameters to implement harder quantum circuits (like the Quantum Fourier Transform (which require training of phases in addition to probabilities)) *directly* without decomposing them into series of elementary gate operations.

The remainder of this dissertation is organized as follows. Chapter 2 gives a brief literature review about our research. Chapter 3 describes the training methodology and the dynamic learning algorithm which was used to find the parameters to implement the desired gate operations. In Chapter 4, method to implement different gate operations in a linear nearest neighbor qubit arrays with both Ising and Heisenberg couplings are discussed. Chapter 5 shows how to implement different mirror inverse gate operations (MI) in systems with Ising type of interactions. Chapter 6 discusses how the parameters found by using the training algorithm can be used to generate an analytical solution for implementing an MI gate operation in an Ising-coupled LNN system of arbitrary length (where length is the number of qubits). Chapter 7 shows how our dynamic algorithm can be used to find the parameters for implementing harder quantum circuits like the 2-qubit QFT and the encoded-Hadamard gate for the 3-qubit error correction code, directly, without having to decompose them into a series of gate operations. This chapter also discusses the changes made to our dynamic learning algorithm to improve its performance. Chapter 8 presents the summary of this dissertation.

CHAPTER 2

LITERATURE REVIEW

Like classical gates, quantum gates are the building blocks of quantum circuits. Physical realizations of logic gates in a quantum computer require interactions between two or more qubits. Based on how each qubit is allowed to interact with other qubits in a physical system, different architectures have been proposed to build a quantum computer. Most of them use nearest neighbor (NN) architectures, in which qubits are arranged along arrays, and *only* the adjacent qubits interact with each other [4, 32-41]. Such NN restrictions present different challenges in implementing quantum algorithms, which are usually designed without taking the NN restriction into consideration. For instance, in NN architectures, we cannot directly perform multi-qubit gate operations on non-NN qubits, also known as remote qubits, as there is no interaction between them. To perform an operation, the remote qubits need to be brought adjacent to each other by using swap gates before performing the desired gate operation. However, this method increases the computational overhead, because as the separation between remote qubits in an NN layout increases, the number of elementary gates (gate count) required to perform an operation between them also increases. For instance, in [42], the authors showed that if CNOT gate operations are restricted to only nearest-neighbors, the number of CNOT gates used increases up to a factor of 9. In [43], the authors showed that translating an arbitrary circuit to an LNN architecture requires a linear increase in quantum cost with respect to the number of qubits. Therefore, methods for implementing gate operations efficiently in LNN arrays is an important area of research, and a number of schemes have been proposed. For instance, heuristic methods for converting a quantum circuit to its equivalent linear NN (LNN) architecture have been proposed [44-46], where an arbitrary quantum circuit is converted into a LNN circuit by inserting swap gates and minimizing their number. Template matching techniques for CNOT and Toffoli gate operations on remote qubits have been proposed where new minimization techniques are used to reduce the number of gate operations [47, 48]. The drawback with these techniques is that they still require swap operations, and in some cases, additional ancillas (qubits that are prepared in $|0\rangle$ state) [44-46]. In [49], the authors' proposed a new gate called the $C^2(-I)$ gate that was used to perform CNOT gate operations between two remote qubits without any interaction between them. However, in all these methods, the gate count is greater than 1. In this research we will show how to implement some of these gates as a single gate.

As mentioned in Chapter 1, any quantum algorithm can be implemented by breaking it into a sequence of gate operations belonging to a universal set (elementary gates). Several methods have been proposed to decompose the quantum circuits into these elementary gate operations [22-31]. However, for any quantum system, it is desired that gate operations be performed before the system loses its coherence. As the number of gate operations to implement the quantum algorithm increases, the time taken to complete the operation before the system loses coherence increases. Hence, it is desired to minimize the total time of an operation, which can be achieved by minimizing the number of elementary gates used during the operation (called the gate count). Also, decomposing quantum circuits into the elementary gate operations can lead to other problems like relative phases, which are picked up due to individual qubit precessions during idle times. This is because, in any system, not all qubits may be involved in a gate operation. Qubits that are idle can pick up relative phases that lead to decoherence. One method is to use special encoding techniques where qubits are grouped together to form logical qubits, which are immune to certain types of errors [50-53]. For instance, the 3-qubit code is a simple quantum error correction code to correct single qubit bit-flip (or phase-flip) errors. Here, the information qubit is encoded using two extra ancillas to form a logical qubit. As with physical qubits, in order to perform gate operations on logical qubits, a universal set of "encoded" gates is required. Even though it is preferred that these encoded gates be "transversal", where the i^{th} qubit in an encoded block interacts either with itself, or the i^{th} qubits in other blocks, not all encoding schemes allow this (simply stated, transversal gates allow a gate operation to be performed directly on the encoded state without decoding). For instance, a Hadamard gate cannot be implemented transversally on the 3-qubit code. Hence, to implement a Hadamard gate on this code, we have to first decode the encoded qubit. This will, in turn, increase the computational overhead, and the possibility of picking up random errors in the decoded state.

For quantum computation in NN arrays, in addition to efficient gate implementation, we also need methods to efficiently transporting qubits down NN arrays. Several methods for transferring quantum states efficiently have been proposed [54-67]. In [68], the author proposed a scheme using a spin chain as a channel, where the desired quantum state is placed at one end of the chain, and is propagated to the other end of the spin chain by evolution under a suitable time-dependent Hamiltonian. Here, the fidelity of state transfer for short-length chains is close to unity. However, as the length of the chain increases, the fidelity of the state transfer decreases significantly. Several schemes have been proposed to overcome this problem. One such technique is by transporting entire qubit registers via 'quantum mirror wires' [59-67]. Here, an unknown multi-qubit quantum state, when encoded at one end of the wire is transmitted to the other end, in reverse order [60-64], and the process is called "mirror inversion" (MI). One method of implementing MI processes in spin chains is by pre-engineering spin-spin couplings with a specific pattern. Another method is to repeatedly apply global signal pulses to the system during its dynamical evolution [65-67]. The advantage of such globally-controlled MI schemes is that a regular uniformly coupled spin chain can be used, without the necessity to manipulate individual couplings between qubits. However, since in some schemes using global control special block encoding of qubits using additional ancillary qubits are required, the storage density can be less than unity.

Going back to methods for implementing quantum gates, some different techniques have been proposed based on optimal control theory (OCT) and neural networks [69-104]. In [69-84], different methods were proposed to implement the desired gate operation by using optimal control theory to obtain the driving field that generates the target unitary transformation. This can be achieved by finding the control fields which minimize or maximize a given performance index, often called cost function. In [69], the authors proposed a scheme using OCT to calculate the field which can produce a desired transformation used for quantum computation. Here, the goal is to obtain the optimal pulse that generates a given unitary transformation, irrespective of the initial state of the system. Later, in [70], they showed how to use vibrational modes of molecules as a set, with the help of OCT, for implementing quantum computation processes. In [71], the authors presented a generalized OCT approach based on equation of motion of the unitary transformation, to obtain the optimal pulse that generates a target unitary transformation irrespective of the initial state. In [72], the authors used QFT in a molecular environment, as a case study to analyze the performance of various OCT schemes. In [73], optimal control techniques have been applied in order to obtain high fidelity quantum gates in 2-level quantum systems, in the presence of coupling to environment. Similarly, several other methods [74-84] have been proposed where OCT is applied to a broad range of quantum mechanical problems.

Recently, many researchers have been focusing on using quantum neural networks (QNN) as another alternative for efficient implementation of quantum gate operations. A QNN is developed by combining concepts of quantum mechanics with artificial neural network models to develop more powerful tools to implement quantum algorithms. In 1995, Kak [85], proposed the concept of quantum neural computer where the measurement operator will force the wave function into the desired eigen-function with the corresponding measurement that represents the computation. Thus far, many QNN models have been proposed and different researchers use different analogies in establishing a connection between quantum mechanics and artificial neural networks [86-104]. For instance, in [86] decoherence of a quantum state was used as an analog to the wave function to develop a quantum neural network. In [87], Chrisely developed a quantum neural net based on the positions of the slits in an interface experiment. Here, the positions of the slits in the interface experiment represent the neuron states. Similarly, in [88], the authors have used "single item networks in many universes" approach to design their quantum neural networks, whereas Ventura et.al [89, 90] used "single item modules in many universes" approach for their quantum neural networks. In [91], the authors have used simulations for both classical and various approaches to quantum neural networks for comparing their performances. Their work suggests that there are indeed certain types of problems for which QNNs are more efficient and more powerful than classical NNs. Similar work has been shown with alternative learning models in [92], where the authors demonstrate that quantum learning algorithms are theoretically more powerful than the classical ones in some situations. In [93], the authors have proposed a new model combining both classical and quantum neural networks for quantum decision making, and quantum associative memories [94-96]. In [97], the author proposed that quantum phenomenon like entanglement can be used to combine quantum computation with neural computation. In [98], the authors designed a learning simulator for the design of quantum algorithms. They have shown that their learning simulator can learn appropriate elements of a quantum algorithm for solving the Deutsch-Jozsa problem. In [99, 100], Behrman, *et al.,* proposed a temporal model for a quantum neural computer using the real time evolution of quantum dot molecules, and show by simulation that such an architecture can perform any classical logic gate. Since the time evolution is quantum mechanical, it can compute backwards or forwards in time. Moreover, it can calculate a purely quantum mechanical gate, such as a phase shift, for which there is no classical equivalent. In [100, 101], they developed a similar model called spatial net using quantum dots. Here, the information will be forwarded in "space" to calculate the output vectors. In [102], they proposed a quantum adaptive computer which can be trained to solve general problems on a quantum computer. Here, they used artificial neural network techniques to learn and compute the desired unitary gate operations. Later, they developed a dynamic learning algorithm [103] for "programming" a general quantum computer that uses superconducting quantum interference devices (SQUIDs), and demonstrate learning of both the classical gates XOR and XNOR. In [104], they used their algorithm to implement a 2-qubit CNOT gate operation, and also showed that this algorithm can be used to compute entanglement.

We used a method similar to the dynamic learning paradigm proposed by Behrman, *et al.,* in our work for implementing a desired gate operation along a linear array of qubits with different types of interactions. However, while Behrman, *et al.,* used measurement operators for training, we use the quantum fidelity condition. This makes our scheme more general as we will later discuss. Our learning algorithm is also similar to 'Gradient Ascent Pulse Engineering technique' (GRAPE) [105-107], where they use gradient ascent technique to find the pulse sequences that implement the desired gate operations. However, our method allows relatively quick learning since for each iteration the derivatives of the Lagrangian with respect to parameters are calculated at the end of the total time, T_f. The system parameters are then correspondingly updated using a weight update rule. In GRAPE, for each iteration, since the total time is divided into a number of equal time steps, a derivative of the cost function with respect to the control amplitudes are calculated *at the end of each time step*. As such, the system parameters are updated for each time step.

Another advantage with our scheme is that all the parameters found by using our method are scalable, and therefore can be adjusted to the requirements of a given experimental realization. For instance, suppose we found the parameters to implement a gate operation using the learning algorithm within some time "T" and want to reduce the total operation time by half (T/2), we do not need have to re-train the network to find a new set of parameters. Instead, we can simply multiply the system parameters by a factor of 2 to find the new set of system parameters that implements the same gate operation, without any change in the fidelity of the corresponding gate operation (both worst-case and overall fidelities). Note that fidelity is a measure of the performance of the gate operation. The maximum fidelity is 1, which implies the gate operation is being realized perfectly.

CHAPTER 3

DYNAMIC LEARNING ALGORITHM

A dynamic learning algorithm based on gradient descent technique can be used to find the system parameters for implementing any desired quantum gate operation. The gradient descent technique is used to find the local minimum of a function by taking steps proportional to the negative of the gradient of the function. Here, we use back propagation technique to train the system. During the training process, we force the output state of the system for a given input state, to the desired state by controlling the system parameters.

3.1 Training Methodology

In this section, the training process, which will be used to find the system parameters for realizing the desired gate operations, will be discussed. Following are the steps that define the training process:

Step 1: Choose a Hamiltonian for an N-qubit system. Since there can be different types of interactions between qubits, the training can be done over a wide range of quantum systems.

Step 2: Choose a learning rule that will be used to train the quantum system. Here, the dynamic learning algorithm based on back propagation technique is used. The method is similar to the dynamic learning paradigm proposed by Behrman, *et al.,* in [103], which was used to train a 2-qubit Ising coupled system to implement a CNOT gate. However, while Behrman, *et al.,* used measurement operators for training, we use the quantum fidelity condition. This makes our scheme more general as we will later discuss. The training method uses a weight update rule based on gradient descent to adjust the device parameters, which are the weights of the quantum system. The time evolution of the quantum system can be found by integrating the Schrodinger equation in MATLAB Simulink. For this, the ODE4 (ordinary differential equation) fixed size step solver will be used. (Simulink provides a set of explicit fixed-step continuous solvers. Explicit solvers compute the value of a state at the next time step as an explicit function of the current values of both the state and the state derivative. The solvers differ in the specific numerical integration technique that they use to compute the state derivatives of the model. The ODE4 solver uses the Fourth-Order Runge-Kutta (RK4) formula).

Step 3: Form training pairs for a specific gate operation, **U**. Here, **U** is the gate operation we are interested in finding system parameters for. A training pair will comprise of an initial state, $|\psi\rangle_{in}$, and the corresponding output state, $|\psi\rangle_{des}$, that would result when applying the gate operation, i.e., $|\psi\rangle_{des} = \mathbf{U}|\psi\rangle_{in}$. Thus, $|\psi\rangle_{in}$ and $|\psi\rangle_{des}$ comprise a training pair. The corresponding density matrices for the two states are $\rho_{in} = |\psi\rangle_{in}\langle\psi|_{in}$ and $\rho_{des} = |\psi\rangle_{des}\langle\psi|_{des} = \mathbf{U}|\psi\rangle_{in}\langle\psi|_{in}\mathbf{U}^{-1}$. The density matrices will be used in training. A minimum of 2^N vector pairs will be used for an N-qubit system, corresponding to all the basis vectors. For instance, for N = 2, there will be 4 training pairs, one each corresponding to the initial states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ (in the computational basis).

Step 4: Train the network for '*m*' iterations. Here, we are trying to find parameters of the system Hamiltonian that will implement the gate operation, **U**, within a time duration, t_f. Note that, t_f is the time within which we want to implement the gate operation, **U**, in a physical quantum system. As such, the duration of each iteration is equal to t_f. At the start of training, i.e., the first iteration, we randomly assign values to the parameters such that they are within experimental limits for a given quantum system. Next, the following procedure is followed during each iteration:

(i) Input one of the 2^N input states as the initial state, $\rho_{in} = |\psi\rangle_{in}\langle\psi|_{in}$, corresponding to a training pair to the network. Allow the state to evolve under the Schrödinger equation for a time t_f. The density matrix for the "actual" output state, ρ_{out}, by propagation of the input state through the network, is calculated.

(ii) Calculate the quantum fidelity between the "actual" output, ρ_{out}, and the desired output, ρ_{des} (from the training pair). Calculate the "error" for the training pair, E_p, by taking the difference between the desired quantum fidelity (which is '1' in all cases) and the calculated quantum fidelity [1, 108].

(iii) Back-propagate the error (in time) through the network, and adjust the parameters. This can be done by integrating the Schrödinger equation from the final time, t_f, to 0 with the help of a change of variable, $t' = t_f - t$. The network can be set up such that different parameters are adjusted at different "learning" rates.

(iv) Repeat steps (i), (ii), and (iii) until all training pairs are exhausted for the iteration.

At the end of each iteration, the root mean square (RMS) error will be calculated by using the following equation:

$$RMS = \frac{1}{2^{\frac{N}{2}}}\sqrt{\sum_{p=1}^{2^N} E_p^2} \tag{3.1}$$

where 2^N is the number of training pairs and E_p is the error corresponding to each training pair. The training is stopped (after *m* iterations) when the RMS error falls below a certain threshold.

Step 5: A successfully trained network will have RMS error below a certain threshold. If the network successfully trains, we test the accuracy of the results by substituting these parameters in the Hamiltonian, and checking if the desired gate operation, **U**, is implemented for different input states. If not, we continue to train the network for more iterations, with the parameter values set at the values obtained from the previously trained network.

When training, as a first step, we will assume all the parameters of the system Hamiltonian to be variables. The network will then be trained to find the system parameters that realize **U** within time t_f. Once the network can be successfully trained for the case when all the parameters are variable, we will then train the network subject to more constraints with some of the parameters fixed. This will be practical since in any quantum system not all parameters can be tuned. Even if all parameters can be tuned, in a physical implementation, it is not desirable to treat all parameters as variables, as doing so can increase the complexity of the external control circuitry.

3.2 Learning Algorithm for Training

In this section, we will show how the dynamic learning algorithm is constructed, using the methodology described by Behrman, *et al*., in [103]. The main differences between their scheme and ours are:

(i) For training, Behrman, *et al*., used a measurement operator. However, one limitation with this approach is the selection of measurement operators. We might not be able to use the same measurement operator to train for different gate operations, and we have to define a suitable operator in each case. In our scheme, we use the quantum fidelity condition for training, where the error is calculated by finding the fidelity of the final state (corresponding to a given input state). As such, the scheme is more general.

(ii) In [103], when training, one of the parameters (bias) was treated as a variable (with time). As such, the gate operation was executed, in three time steps ($3t_s$) where the bias was changed at the start of each time step (t_s). In our scheme, none of the parameters are treated as variables (with time) while training. As such, the gate operation is implemented within a single time step (t_f).

To construct the dynamic learning algorithm, we use the time evolution of the Schrödinger equation which is given by

$$\frac{\partial \rho}{\partial t} = -\frac{i}{\hbar}[H, \rho] \tag{3.2}$$

Here, ρ is the density matrix, ħ is Planck's constant divided by 2π, and H is the Hamiltonian of the system. In the dynamic leaning algorithm, the basic idea is to force the output state, $|\psi\rangle_{out}$, of the system for a given input state, $|\psi\rangle_{in}$, to the desired state, $|\psi\rangle_{des}$, under a chosen gate operation, **U**, by controlling the parameters of the Hamiltonian (for instance, tunneling, bias and coupling). Initially, the system parameters are chosen randomly (while confining to experimental limits). They are then trained by using the gradient descent learning rule,

$$W_{new} = W_{old} - \alpha \frac{dL}{dW} \tag{3.3}$$

where α is the learning rate, W is the parameter we are training, and L is the Lagrangian which is used to find the local minimum of the error function. Here, the density matrix ρ is constrained to satisfy the Schrödinger equation for the time interval of 0 to t_f. Hence, the Lagrangian can be defined as

$$L = \frac{1}{2}E_p^{\ 2} + \int_0^{t_f} \lambda(t)^\dagger \left(\frac{\partial \rho}{\partial t} + \frac{i}{\hbar}[H, \rho]\right)\gamma(t)\,dt \tag{3.4}$$

where E_p is the error corresponding to a training pair p, where p = 1 to 2^N, $\lambda^\dagger(t)$ and γ(t) are the Lagrange multipliers. Here, $\lambda^\dagger(t)$ is a row vector and γ(t) is a column vector. By making the first variation of the Lagrangian, L, with respect to ρ equal to zero, and then by integrating by parts, we can calculate these Lagrangian multipliers. The resulting equation can be written as

$$\lambda_i \frac{\delta \gamma_j}{\delta t} + \frac{\delta \lambda_i}{\delta t}\gamma_j - \frac{i}{\hbar}\sum_k \lambda_k H_{ki}\gamma_j + \frac{i}{\hbar}\sum_k \lambda_i H_{jk}\gamma_k = 0 \tag{3.5}$$

with the boundary conditions at the final time, t_f:

$$\lambda_i(t_f)\gamma_j(t_f) = (E_p)(\rho_{des})_{ji} \tag{3.6}$$

Here, $\lambda(t_f)$ and $\gamma(t_f)$ are the Lagrange multipliers at the final time, $(\rho_{des})_{ji}$ is the ji^th element in the "desired" density matrix (defined in step 3 of the training process).

The error, E_p, is calculated as:

$$E_p = 1 - F(\rho_{des}, \rho_{out}) \tag{3.7}$$

where,

$$F(\rho_{des}, \rho_{out}) = Tr(\sqrt{\sqrt{\rho_{des}}\rho_{out}\sqrt{\rho_{des}}}) \tag{3.8}$$

Here, F (ρ_{des}, ρ_{out}) is the quantum fidelity [1, 108], and ρ_{des} and ρ_{out} are as previously described. The quantum fidelity gives a measure of the closeness of the two density matrices. Here, F (ρ_{des}, ρ_{out}) varies from 0 to 1 depending on how close ρ_{out} is to ρ_{des}. If the two matrices are identical, i.e., if $\rho_{out} = \rho_{des}$, then F (ρ_{des}, ρ_{out}) = 1. As can be seen from equation (3.7), the error, E_p, is calculated by taking the difference between the desired fidelity, 1, and fidelity of the trained network, for a given training pair.

3.3 Choosing Training Pairs for the Dynamic Learning Algorithm

In our learning algorithm, we used quantum fidelity condition [1, 108], to calculate the trained network fidelity. This value is compared to the desired fidelity to calculate the error, which is then back propagated through the network to update the system parameters. One advantage of using quantum fidelity condition over the measurement operator is that, we can take '+1' as the target for every network as long as the proper input and output vectors are chosen as training pairs. This makes our scheme more general.

In choosing the training set, we basically use all the basis vectors spanning the space. Typically, in most cases, we use the computational basis states as our input vectors. However, this approach will not work if the computational basis states also happen to be the eigen vectors of the matrix that represents the gate operation we are trying to train the system for. In general, for any gate operation, A, if the training set comprises of the eigen vectors of A, then the algorithm is not able to train the system parameters to implement the gate. To demonstrate, suppose we represent the gate operation, A, in its eigen basis:

$$A = \tau_1|V_1\rangle\langle V_1| + \tau_2|V_2\rangle\langle V_2| + \tau_3|V_3\rangle\langle V_3| + \tau_4|V_4\rangle\langle V_4| \tag{3.9}$$

where $|V_1\rangle$, $|V_2\rangle$, $|V_3\rangle$, and $|V_4\rangle$ are the eigen vectors, and τ_1, τ_2, τ_3, and τ_4 are the corresponding eigen values of matrix A. Note that $|\tau_1|$, $|\tau_2|$, $|\tau_3|$, $|\tau_4| = 1$, since eigen values of a unitary matrix have unit magnitude. If we choose one of the eigen vectors as our input vector, then the final state becomes:

$$A|V_1\rangle = \tau_1|V_1\rangle\langle V_1||V_1\rangle + \tau_2|V_2\rangle\langle V_2||V_1\rangle + \tau_3|V_3\rangle\langle V_3||V_1\rangle + \tau_4|V_4\rangle\langle V_4||V_1\rangle \tag{3.10}$$

$$A|V_1\rangle = \tau_1|V_1\rangle \tag{3.11}$$

From equation (3.11), we can see that the desired final state vector will be the same as the initial state, but scaled by its corresponding eigen value, τ_1. Therefore, the desired final density matrix, ρ_{des}, will be:

$$\rho_{des} = A|V_1\rangle\langle V_1|A^\dagger = |\tau_1|^2|V_1\rangle\langle V_1| = \rho_{in} \tag{3.12}$$

From equation (3.12) we can see that both the desired density matrix and the input density matrix are equal which means that the fidelity will be 1. As such, from equation (3.7), the error E_p will be zero. Hence, the dynamic learning algorithm will not train the network for the desired gate operation. However, we can overcome this by choosing input and output vectors as the training pairs, which are not eigen vectors. As an example, suppose we want to find the system parameters for the Controlled-Phase (C-Ph) gate. The C-Ph gate is a two-qubit controlled-unitary gate, which is diagonal in the computational basis, i.e., the computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, are eigen vectors of this gate with corresponding eigen values +1, +1, +1, and +1, respectively. To train this network, we considered a 2-qubit system with Ising type interactions. When we tried to train this system using the computational basis states as input vectors, we were not able to train the network. However, when we chose training pairs as that shown in Table 3.1, we were able to train the network successfully, and the RMS error decreased to zero. The C-Ph gate can be realized with the following parameter values: $\Delta_A = \Delta_B = 0$, $\varepsilon_A = \varepsilon_B = 62.5$ MHz and $\xi = 37.5$ MHz for $t_f = 10$ns. Later, by substituting these parameters in the original Hamiltonian, we confirmed the realization of C-Ph gate operation (with an overall global phase of −135 degrees that can be ignored).

TABLE 3.1

TRAINING DATA FOR THE CONTROLLED-PHASE GATE OPERATION.

Input State, $|\psi\rangle_{in}$ ($|q_1q_2\rangle$)	**Desired Output State, $|\psi\rangle_{des}$ ($|q_1q_2\rangle$)**
$(|00\rangle+|01\rangle+|10\rangle+|11\rangle)/2$	$(|00\rangle+|01\rangle+|10\rangle-|11\rangle)/2$
$(|00\rangle-|01\rangle+|10\rangle-|11\rangle)/2$	$(|00\rangle-|01\rangle+|10\rangle+|11\rangle)/2$
$(|00\rangle+|01\rangle-|10\rangle-|11\rangle)/2$	$(|00\rangle+|01\rangle-|10\rangle+|11\rangle)/2$
$(|00\rangle-|01\rangle-|10\rangle+|11\rangle)/2$	$(|00\rangle-|01\rangle-|10\rangle-|11\rangle)/2$

CHAPTER 4

IMPLEMENTATION OF QUANTUM GATE OPERATIONS BETWEEN UNCOUPLED QUBITS

Consider a quantum system of N qubits arranged in an LNN array as shown in Figure 4.1. Here, each qubit only interacts with its nearest neighbors. The Hamiltonian, H_N, where 'N' represents the number of qubits in the system, is:

$$H_N = \sum_{i=1}^{N} (\Delta_i \sigma_{Xi} + \varepsilon_i \sigma_{Zi}) + \sum_{i=1}^{N-1} \xi_{i,i+1} \sigma_{Zi} \sigma_{Zi+1} \qquad (4.1)$$

Here, the terms Δ_i, and ε_i for i =1, 2… N, are the tunneling and bias parameters, respectively for qubit Q_i, I = 1, 2… N, and $\xi_{i,i+1}$ is the coupling parameter between qubits Q_i and Q_{i+1}. Also, σ_{Xi} and σ_{Zi} are the Pauli matrices corresponding to qubit Q_i. Here, we use Ising type interactions which are typical of Josephson-junction qubits [13-15]. We will now discuss how to implement gate operations for the architecture shown in Figure 4.1.

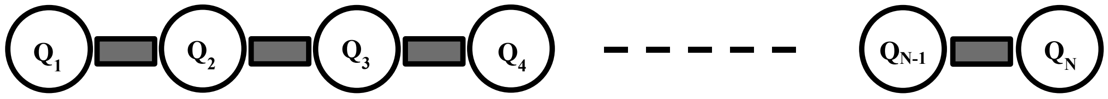

Figure 4.1. Linear nearest neighbor array comprising of N-qubits. Circles represent qubits and rectangles represent couplings between adjacent qubits. Each qubit can only interact with its nearest neighbors.

4.1 CNOT Gate Operation between Next-To-Nearest Neighbor Qubits

Suppose we want to implement a CNOT gate between two next-to-nearest neighbor qubits Q_1 and Q_3 (Figure 4.1), without requiring the qubits to be adjacent to each other. Here, Q_3 is the target qubit. The CNOT gate operation between qubits Q_1 and Q_3 can be described as:

$$|q_1 q_2 q_3\rangle \rightarrow \begin{cases} |q_1 q_2 q_3\rangle, if\ q_1 = |0\rangle \\ |q_1 q_2 q_3{}'\rangle, if\ q_1 = |1\rangle \end{cases} \tag{4.2}$$

where $|q_i\rangle$ corresponds to state of qubit Q_i, i = 1 to 3, and $|q_3'\rangle$ is the complement of state $|q_3\rangle$ (for instance, if $|q_3\rangle = |1\rangle$, $|q_3'\rangle = |0\rangle$). To implement the gate operation as in Figure 4.2, two additional swap gates are required to bring qubits Q_1 and Q_3 adjacent to each other. If each swap gate is decomposed into 3 CNOT gates, the gate count can increase to 7. In addition to increasing the computational overhead, such multi-gate decompositions can also increase the probability of propagating errors through the circuit.

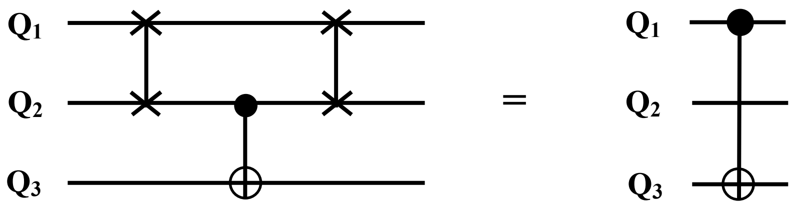

Figure 4.2. Implementation of CNOT gate operation between qubits Q_1 and Q_3 using conventional methods where the states of qubits Q_1 and Q_2 are swapped before and after the CNOT gate.

Here, the system parameters that realize the CNOT gate operation on qubits Q_1 and Q_3 *directly* without requiring swap gates to bring them adjacent to each other can be found by using the dynamic algorithm as a tool. To demonstrate this, a 3-qubit system with Ising type interactions was chosen where the Hamiltonian is defined by equation (4.1) with N = 3. The training set comprised of eight input-output pairs. The inputs were the eight computational basis states, $|000\rangle$ through $|111\rangle$. The outputs were the expected state vectors after applying the transformation given by equation (4.2) to the corresponding inputs. For instance, the output states for the input states $|010\rangle$ and $|110\rangle$ were $|010\rangle$ and $|111\rangle$, respectively. For training, we followed the procedure described in Chapter 3. The step size in the ODE4 was 0.001ns, which is the size of the steps taken by the solver when computing the numerical integration, and total time, t_f = 34.5ns, which is the time step for the gate operation. Training was stopped when the RMS error was 0.0038. Table 4.1 shows the fidelity of the output state corresponding to each input for the trained network (the optimal fidelity in each case should be 1). Table 4.2 lists the parameters of the trained network.

TABLE 4.1

FIDELITY OF THE FINAL STATE OF THE TRAINED NETWORK UNDER A CNOT GATE BETWEEN QUBITS Q_1 AND Q_3.

Input State, $\|\psi\rangle_{in}$ ($\|q_1q_2q_3\rangle$)	**Desired Output State, $\|\psi\rangle_{des}$** ($\|q_1q_2q_3\rangle$)	**Fidelity, F (ρ_{des}, ρ_{out})**	**Error** **(1 – F)**
$\|000\rangle$	$\|000\rangle$	1	0
$\|001\rangle$	$\|001\rangle$	1	0
$\|010\rangle$	$\|010\rangle$	0.9995	0.0005
$\|011\rangle$	$\|011\rangle$	0.9995	0.0005
$\|100\rangle$	$\|101\rangle$	0.9924	0.0076
$\|101\rangle$	$\|100\rangle$	0.9924	0.0076
$\|110\rangle$	$\|111\rangle$	0.9999	0.0001
$\|111\rangle$	$\|110\rangle$	0.9999	0.0001
RMS error for the network = 0.0038			

To verify if these parameters compute a CNOT gate between two next-to-nearest neighbor qubits, these parameters were substituted in the Hamiltonian given by equation (4.1) for N = 3, and the system was allowed to evolve for 34.5ns (t_f). The simulations confirmed the CNOT gate operation with Q_1 as control and Q_3 as target, which was realized up to an overall global phase of 90 degrees that can be ignored.

TABLE 4.2

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A CNOT GATE OPERATION BETWEEN QUBITS Q_1 AND Q_3.

,

Δ_1	5 MHz
Δ_2	12.6 MHz
Δ_3	1.8842 GHz
ε_1	1 GHz
ε_2	395 MHz
ε_3	119.7 MHz
ξ_{12}	395 MHz
ξ_{23}	113.1 MHz

As an example, Figure 4.3 shows the probabilities of the qubits A, B, and C in state $|1\rangle$ when the input state was a superposition of $|000\rangle$ and $|100\rangle$ states. After 34.5ns, the input state switches to the superposition of $|000\rangle$ and $|101\rangle$ states, as expected under the CNOT gate operation between qubits Q_1 and Q_3. From the figure, we can see that the final probabilities of the qubits A, B, and C to be in state $|1\rangle$ are 0.5, 0, and 0.5, respectively, which shows the implementation of CNOT gate operation between qubits Q_1 and Q_3. In this case, the worst-case fidelity of the gate operation was 99.24%, and the overall fidelity was 99.795%. The overall fidelity can be improved by training the network further.

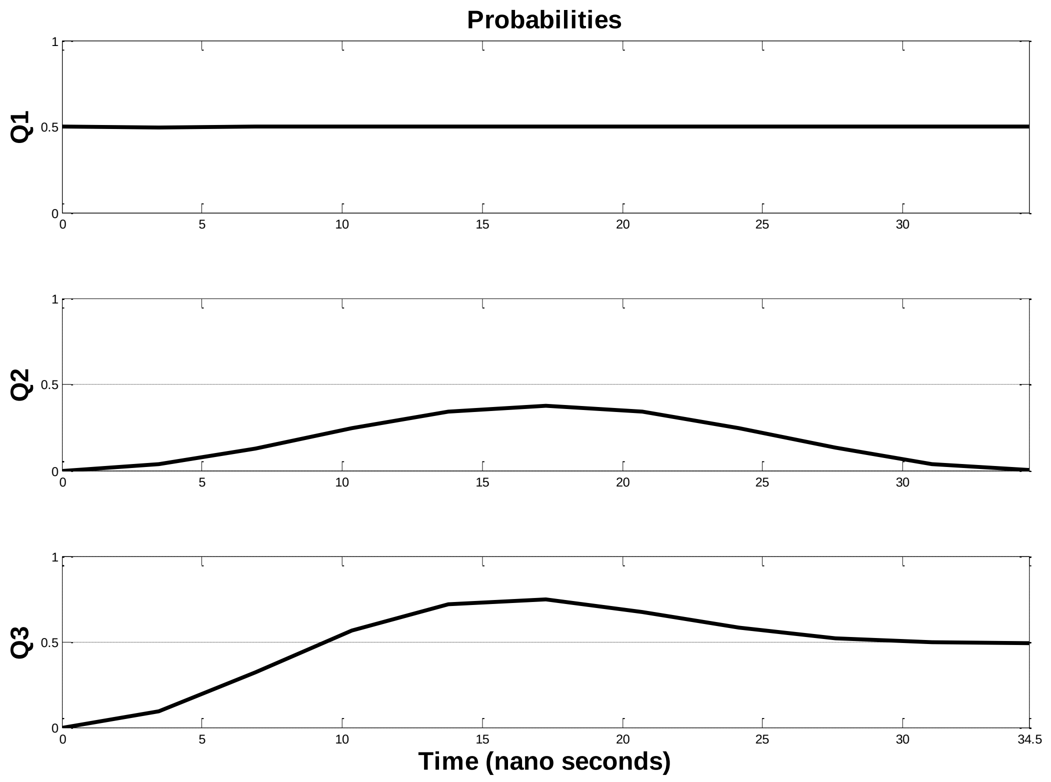

Figure 4.3. CNOT gate operation on qubits Q_1, Q_2 and Q_3, with qubit Q_1 as control, and qubit Q_3 as target. Here, the initial and final states of the system are $(1/\sqrt{2})$ $(|000\rangle+|100\rangle)$ and $(1/\sqrt{2})$ $(|000\rangle+|101\rangle)$, respectively. The probability of the qubits Q_1, Q_2 and Q_3 to be in state $|1\rangle$ has been plotted, which shows the implementation of CNOT gate operation for the given input.

4.2 Toffoli Gate Operation

To implement a Toffoli gate on a 3-qubit system using a scheme like the pulsed bias scheme [109], we need the target qubit to be coupled to both controls. As a result, to implement the Toffoli gate in an LNN system (Figure 4.1), we need to perform swap gates to bring the target qubit Q_3 between the two controls. However, using the dynamic learning algorithm as a tool, we can find system parameters such that the gate operation can be directly implemented without requiring qubit Q_3 to be between the two controls. As with the CNOT, the training set comprised of all 8 computational basis states as input vectors, and their corresponding output states under the transformation given by equation (1.13) as the output vectors. The same Hamiltonian as that for the CNOT was used. The step size in the ODE4 was 0.05ns, and total time, $t_f = 10$ns. Training was stopped when the RMS error was 0.007.

Table 4.3 shows the fidelity of the output state corresponding to each input state for the trained network. Table 4.4 lists the parameters of the trained network.

TABLE 4.3

FIDELITY OF THE FINAL STATE OF THE TRAINED NETWORK UNDER A TOFFOLI GATE BETWEEN QUBITS Q_1, Q_2 AND Q_3.

Input State, $\|\psi\rangle_{in}$ ($\|q_1q_2q_3\rangle$)	**Desired Output State, $\|\psi\rangle_{des}$** ($\|q_1q_2q_3\rangle$)	**Fidelity, F (ρ_{des}, ρ_{out})**	**Error** **(1 – F)**
$\|000\rangle$	$\|000\rangle$	0.9988	0.0012
$\|001\rangle$	$\|001\rangle$	0.9976	0.0024
$\|010\rangle$	$\|010\rangle$	0.9979	0.0021
$\|011\rangle$	$\|011\rangle$	0.9977	0.0023
$\|100\rangle$	$\|100\rangle$	0.9885	0.0115
$\|101\rangle$	$\|101\rangle$	0.9921	0.0079
$\|110\rangle$	$\|111\rangle$	0.9905	0.0095
$\|111\rangle$	$\|110\rangle$	0.9905	0.0095
RMS error for the network = 0.007			

As for the CNOT, to test whether the trained parameters implement a Toffoli gate, the values of these parameters were substituted in the 3-qubit Hamiltonian (equation (4.1) with N = 3) and the system was allowed to evolve for 10ns. The simulations confirmed the Toffoli gate operation. As an example, Figure 4.4 shows the evolution of the system for the initial state $|110\rangle$. The figure shows the evolution of the probabilities of the computational basis states $|110\rangle$ and

|111⟩ (the probabilities of the remaining states have not been shown, because at the final time, t_f = 10ns, they are zero). From the figure, we can see that the |110⟩ state evolves to the |111⟩ state, which shows the implementation of the expected Toffoli gate operation for the given input. The overall fidelity of the gate operation was 99.44%.

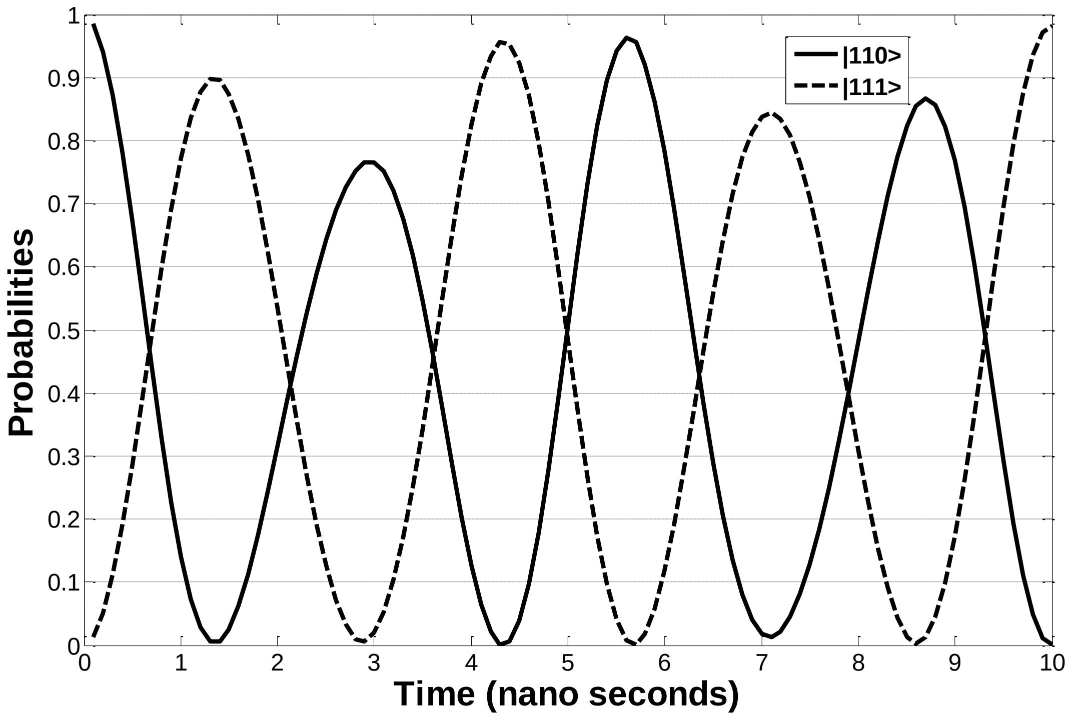

Figure 4.4. Toffoli gate operation on qubits Q_1, Q_2 and Q_3, with qubits Q_1 and Q_2 as controls, and qubit Q_3 as target. The initial and the final states of the system are |110⟩ and 111⟩, respectively. The evolution of the probabilities of the basis states |110⟩ and |111⟩ have been plotted, which shows the implementation of the Toffoli gate operation for the given input.

TABLE 4.4

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A TOFFOLI GATE OPERATION BETWEEN QUBITS Q_1, Q_2 AND Q_3.

Δ_1	10 MHz
Δ_2	70 MHz
Δ_3	182.5 MHz
ε_1	1 GHz
ε_2	116.7 MHz
ε_3	387.3 MHz
ξ_{12}	337.7 MHz
ξ_{23}	380.3 MHz

4.3 Swap Gate Operation between Two NN Qubits Without Decomposing into 3 CNOTs

Here, we use the dynamic learning algorithm to find the system parameters that implements the swap gates directly in a 2-qubit system with Ising type interactions. The Hamiltonian for this system can be defined using equation (4.1) with N = 2. Training pairs were formed by choosing the four computational basis states as input vectors, and their respective expected final states after the swap operation, as output vectors. As before, the ODE4 fixed size step solver was used with the step size of 0.1ns, and total time, t_f = 10ns. Training was stopped when the RMS error was 0. Table 4.5 shows the fidelity of the output state corresponding to each input for the trained network. Table 4.6 lists the parameters of the trained network.

TABLE 4.5

FIDELITY OF THE FINAL STATE OF THE TRAINED NETWORK UNDER A SWAP GATE BETWEEN QUBITS Q_1 AND Q_2.

Input State, $\|\psi\rangle_{in}$ ($\|q_1q_2\rangle$)	**Desired Output State, $\|\psi\rangle_{des}$ ($\|q_1q_2\rangle$)**	**Fidelity, F (ρ_{des}, ρ_{out})**	**Error (1 – F)**
$\|00\rangle$	$\|00\rangle$	1	0
$\|01\rangle$	$\|10\rangle$	1	0
$\|10\rangle$	$\|01\rangle$	1	0
$\|11\rangle$	$\|11\rangle$	1	0
RMS error for the network = 0			

TABLE 4.6

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A SWAP GATE OPERATION BETWEEN QUBITS Q_1 AND Q_2.

Δ_1	35.4 MHz
Δ_2	35.4 MHz
ε_1	27.8 MHz
ε_2	27.8 MHz
ξ	37.3 MHz

To check if these parameters compute a 2-qubit swap operation, these parameters were substituted in the 2-qubit system Hamiltonian, given by equation (4.1) for N = 2. The simulations confirmed the swap operation. Here, both the worst-case fidelity and the overall fidelity of the gate operation were 100%. As an example, Figure 4.5 shows the evolution of a 2-qubit quantum system under the swap gate operation, when the initial state is $|01\rangle$. The figure shows the evolution of the probabilities of all the four computational basis states. From Figure 4.5, we can see that the final state after 10ns is $|10\rangle$, which shows the implementation of the swap operation between qubits 1 and 2.

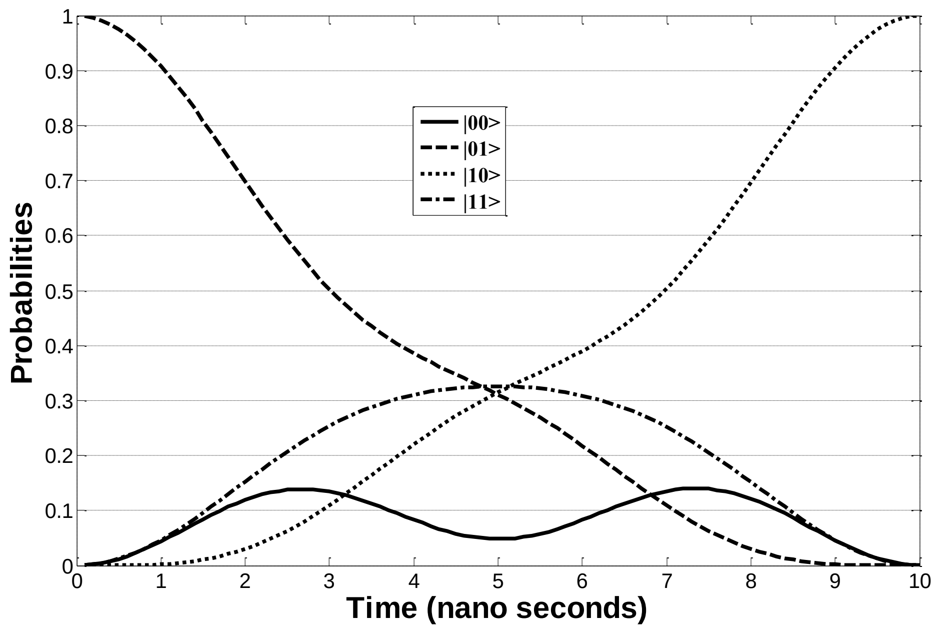

Figure 4.5. Direct implementation of a swap gate in a 2-qubit system without decomposing it into 3 CNOT gates (as shown in Figure 1.2). The initial and final states of the system are $|01\rangle$ and $|10\rangle$, respectively. The evolution of the probabilities of the 4 basis states have been plotted, which shows the implementation of the swap gate operation for the given input.

4.4 Swap Gate Operation between Next-To-Nearest Neighbor Qubits

Suppose we want to interchange the states of the next-to-nearest neighbor qubits Q_1 and Q_3. For this, three swap gate operations are needed (or 9 CNOT gates). However, by using the training algorithm, parameters can be found to implement the gate operation directly.

For this, a 3-qubit, Ising coupled system with the same Hamiltonian as that for the CNOT and Toffoli was considered. The training set comprised of the eight computational basis states, and their respective expected final states after the swap operation between qubits Q_1 and Q_3, where the transformation for the gate operation is given as:

$$|q_1 q_2 q_3\rangle \rightarrow |q_3 q_2 q_1\rangle \tag{4.3}$$

Using the training process explained in Chapter 3, the system was trained until the RMS error value was 0. As before, the ODE4 fixed size step solver was used with the step size of 0.1ns, and total time, $t_f = 10$ns. Table 4.7 shows the fidelity of the output state corresponding to each input for the trained network. Table 4.8 lists the parameters of the trained network.

TABLE 4.7

FIDELITY OF THE FINAL STATE OF THE TRAINED NETWORK UNDER A SWAP GATE BETWEEN QUBITS Q_1 AND Q_3.

Input State, $\|\psi\rangle_{in}$ ($\|q_1q_2q_3\rangle$)	**Desired Output State, $\|\psi\rangle_{des}$** ($\|q_1q_2q_3\rangle$)	**Fidelity, F (ρ_{des}, ρ_{out})**	**Error** **(1 – F)**
$\|000\rangle$	$\|000\rangle$	1	0
$\|001\rangle$	$\|100\rangle$	1	0
$\|010\rangle$	$\|010\rangle$	1	0
$\|011\rangle$	$\|110\rangle$	1	0
$\|100\rangle$	$\|001\rangle$	1	0
$\|101\rangle$	$\|101\rangle$	1	0
$\|110\rangle$	$\|011\rangle$	1	0
$\|111\rangle$	$\|111\rangle$	1	0
RMS error for the network = 0			

To confirm the gate operation for the parameters listed in Table 4.8, these parameters were substituted in a 3-qubit Hamiltonian (equation (4.1) with N = 3). Here, both the worst-case fidelity and the overall fidelity of the gate operation were 100%. As an example, Figure 4.6 shows the evolution of a 3-qubit quantum system under the swap gate, when the initial state is $|110\rangle$. The figure shows the evolution of the probabilities of the computational basis states $|110\rangle$ and $|011\rangle$ (the probabilities of the remaining states were not plotted as they are zero at the final time, $t_f = 10$ns). From Figure 4.6, we can see that the final state after 10ns is $|011\rangle$, which shows the implementation of the swap operation between qubits Q_1 and Q_3 for the given input.

TABLE 4.8

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A SWAP GATE OPERATION BETWEEN QUBITS Q_1 AND Q_3.

Δ_1	43.3 MHz
Δ_2	50 MHz
Δ_3	43.3 MHz
ε_1	33.1 MHz
ε_2	0.1 MHz
ε_3	33.1 MHz
ξ_{12}	48.4 MHz
ξ_{23}	48.4 MHz

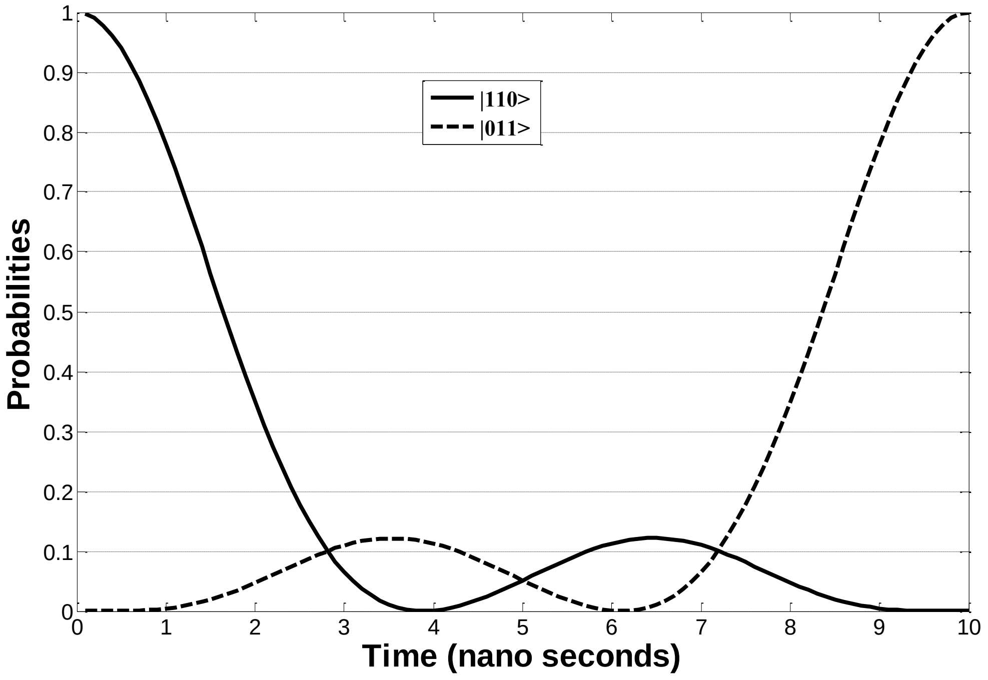

Figure 4.6. Direct implementation of a swap gate between next-to-nearest neighbors Q_1 and Q_3 in a 3-qubit system without decomposing it into 9 CNOT gates. The initial and final states of the system are $|110\rangle$ and $|011\rangle$, respectively. The evolution of the probabilities of the basis states $|110\rangle$ and $|011\rangle$ have been plotted, which shows the implementation of the swap gate operation for the given input.

This method of implementing swap operations directly without decomposing into CNOTs can be used to realize efficient quantum computing in LNN arrays. For instance, in Figure 4.1, to implement a gate operation between qubits Q_1 and Q_N, a maximum of 2(N-2) swap gates are required to bring qubit Q_1 adjacent to qubit Q_N. If each swap gate is broken into 3 CNOTs, 6(N-2) CNOTs are required. However, using the swap gate operation between NN qubits proposed in section 4.3, only 2(N-2) operations are required. Moreover, using the swap gate operation between next-to-nearest neighbor qubits proposed in section 4.4, the gate count can be further reduced.

Throughout this work, when implementing gate operations, it is assumed that all the pulses are ideal. Typically, in an experimental system, pulses are non-ideal with finite rise and fall times. As a result, the switching process itself might give rise to errors: decrease in the expected probabilities of the output state, and introduction of relative phases in the output state. The expected probabilities can be raised by changing the total time of the gate operation, t_f, the value for which can be obtained from simulations. On the other hand, random relative phases as a result of rise and fall times are harder to keep track of. To overcome such phases, an architecture similar to that presented in [110] can be used.

4.5 Training while Keeping Some Parameters Fixed

In physical quantum systems, even though the ability to tune all parameters may exist, it may be beneficial to treat some parameters as fixed parameters in order to reduce the number of external control lines. This is because from a practical standpoint we typically do not vary all the system parameters when implementing two or more different gate operations in the same system. The purpose of this part of the study was to examine if a network could be trained for a gate operation by keeping some parameters fixed throughout the training process. For training, the 3-

qubit system with Ising type couplings was considered. The tunneling parameters were fixed at 36 MHz. The network was then trained for a swap gate between Q_1 and Q_3, and a CNOT gate between qubits Q_1 and Q_3. This is because from a practical standpoint we typically do not vary all the system parameters when implementing two or more different gate operations in the same system. The parameters to realize the two gate operations are listed in Table 4.9 and Table 4.10, respectively.

TABLE 4.9

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A SWAP GATE BETWEEN QUBITS Q_1 AND Q_3 AFTER FIXING THE TUNNELING PARAMETERS.

Δ_1	36 MHz
Δ_2	36 MHz
Δ_3	36 MHz
ε_1	24.9 MHz
ε_2	0 MHz
ε_3	24.9 MHz
ξ_{12}	36.7 MHz
ξ_{23}	36.7 MHz

From Table 4.9, we can see that we were able to successfully train the parameters for a swap gate between qubits Q_1 and Q_3, by keeping all 3 tunneling parameters fixed at 36 MHz. Here, the step size in the ODE4 was 0.1ns, and total time, $t_f = 13$ns. Training was stopped when the RMS error was 0.0189. The overall fidelity of the gate operations was 98.25%. From the Table 4.10, for the CNOT gate between qubits Q_1 and Q_3, we can see that the tunneling parameter Δ_3 had to be treated as a variable for training. Here, the step size in the ODE4 was

0.01ns, and total time, t_f = 12ns. Training was stopped when the RMS error was 0.0244. The overall fidelity of the gate operation was 98.6%.

TABLE 4.10

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A CNOT GATE BETWEEN QUBITS Q_1 AND Q_3 AFTER FIXING THE TUNNELING PARAMETERS Δ_1 AND Δ_2.

Δ_1	36 MHz
Δ_2	36 MHz
Δ_3	834.2 MHz
ε_1	1 GHz
ε_2	447.9 MHz
ε_3	133.9 MHz
ξ_{12}	448.2 MHz
ξ_{23}	131.9 MHz

4.6 Extending the Learning Algorithm to Non-Diagonal Hamiltonians

The gate operations described until now were all implemented using an LNN array with diagonal Ising type interactions. The purpose of this part of the research was to study if the learning algorithm could be extended to other non-diagonal coupling schemes. To demonstrate, a two-qubit system with Heisenberg interactions, which is typical of spin-coupled quantum dots [6, 7], has been considered. The Hamiltonian is

$$H = \sum_{i=1,2} (\Delta_i \sigma_{Xi} + \varepsilon_i \sigma_{Zi}) + \xi(\sigma_{Z1}\sigma_{Z2} + \sigma_{X1}\sigma_{X2} + \sigma_{Y1}\sigma_{Y2}) \qquad (4.4)$$

Here, as before, the terms Δ_i, and ε_i are the tunneling and bias parameters, respectively, and ξ is the coupling parameter. The network has been trained to implement a swap gate between qubits Q_1 and Q_2. While training, the tunneling parameters were fixed, i.e., $\Delta_1 = \Delta_2$ = 36 MHz. The other parameters trained through the network were $\varepsilon_1 = \varepsilon_2 = 33.7$ MHz, and $\xi = 37.5$ MHz (t_f = 10ns). The training was stopped when the RMS error was 0.001. Simulations confirmed the gate operation upto an overall global phase of 135 degrees, which can be ignored. The overall fidelity of this gate operation was 99.9%.

4.7 Conclusions

A new scheme to implement gate operations in a one dimensional LNN array, by using dynamic learning algorithm has been introduced. This was accomplished by training the quantum system using a back propagation technique, to find the system parameters that implement gate operations directly. We showed how the training algorithm can be used as a tool for finding the parameters for implementing CNOT and Toffoli gates between next-to-nearest neighbor qubits in an Ising-coupled LNN system. We then showed how the scheme can be used to find parameters for realizing swap gates, first between two adjacent qubits, and then, between two next-to-nearest-neighbor qubits, in each case without decomposing it into 3 CNOT gates. Finally, we showed how the scheme can be extended to systems with non-diagonal interactions. The main advantage of our scheme is that, we can reduce the computational overhead of a quantum circuit by finding the parameters to implement the desired gate operations directly, without decomposing them into a sequence of elementary gate operations. We can also reduce the time taken to realize any desired gate operation using this scheme. For instance, using the pulsed bias scheme that was proposed by Kumar, *et al.*, in [109], we need 30ns to realize a 2-qubit swap gate operation (10ns for each CNOT gate operation). However, using the gate operation proposed in section 4.3, we only need 10ns to implement the swap gate operation. Another advantage is that, all the parameters found using our scheme are scalable, and therefore, can be adjusted to the requirements of a given experimental realization.

CHAPTER 5

IMPLEMENTATION OF QUANTUM MIRROR INVERSE GATE OPERATIONS USING A DYNAMIC LEARNING ALGORITHM

As discussed in Chapter 2, LNN restrictions present different challenges in implementing quantum circuits, which are usually designed without taking the NN restriction into consideration. Typically, to perform a gate operation between two non-NN qubits, also known as remote qubits, the qubits need to be brought adjacent to each other. As the separation between remote qubits increases, the number of gates required to perform an operation between them can also increase. Hence, finding ways for efficiently transporting qubits down NN arrays, to implement quantum circuits efficiently, has become an important area of research in quantum computing. One method is using the mirror inverse operations (MI) where an unknown multi-qubit quantum state, when encoded at one end of the wire is transmitted to the other end, in reverse order [60-64].

In this chapter, we use the dynamic learning algorithm presented in Chapter 3 to find the system parameters to implement MI directly in a linear array of qubits with Ising type interactions. In section 5.1, we show how the training algorithm can be used as a tool for finding the parameters for implementing MI between nearest neighbor qubits in an Ising-coupled LNN system. Unlike non-diagonal XY interactions where a SWAP gate is easily implemented, in Ising coupled systems, each swap operation needs to be decomposed into 3 CNOTs. As such, methods for finding system parameters that allow us to transmit qubit states along LNN arrays without requiring swap gates can greatly reduce the overall computational overhead in these systems. These Ising coupled systems are commonly seen in superconducting Josephson junctions, dipole-

dipole or J-coupled systems [13−15, 111, 112]. We used our learning algorithm to find the parameters to implement MI directly without decomposing them into a sequence of gate operations in four, five, six, seven, and eight-qubit systems. In section 5.2, we investigate the efficiency of our MI scheme in the presence of unwanted couplings. In section 5.3, we validate our scheme by comparing our results against those presented in [60, 61] where the authors' show how to implement MI in an XY coupled system. In section 5.4, we present the conclusions.

5.1 Training Results for Implementing Mirror Inverse Gate Operations

Consider a quantum system of N qubits arranged in an LNN array as shown in Figure 4.1. Here, each qubit only interacts with its nearest neighbors through Ising type interactions, which are diagonal in the interaction basis, typical of Josephson-junction qubits [13-15]. The Hamiltonian, H_N, where 'N' represents the number of qubits in the system, is given by equation (4.1). We will now provide training results for MI in 4-, 5-, 6-, and 7-qubit LNN systems. The training set comprised of 2^N basis states, and their respective expected states after the MI operation. Here, the MI gate operation, **U**, is defined by the following transformation:

$$|q_1 q_2 q_3 \cdots q_{N-1} q_N\rangle \xrightarrow{\mathbf{U}} |q_N q_{N-1} \cdots q_3 q_2 q_1\rangle \tag{5.1}$$

Using the training process explained in Chapter 3, we trained the system for the desired MI operations until the RMS error values for MI in 4-, 5-, 6- and 7-qubit LNN systems were 0.0006, 0.0003, 0.0026 and 0.0023, respectively. The ODE4 fixed size step solver was used with step size of 0.1ns was used for all MI operations.

Table 5.1 lists the parameters of the trained network in a 4-qubit LNN system. To confirm the gate operation for the parameters listed in Table 5.1, we substituted these parameters in a 4-qubit Hamiltonian (equation (4.1) with N = 4) and ran simulations. The worst-case fidelity of the MI operation was 99.92%, and the overall fidelity of the MI operation was 99.94%.

Simulations confirmed the MI operation on qubits Q_1 through Q_4, which was realized up to an overall global phase of 45 degrees that can be ignored.

As an example, Figure 5.1 shows the evolution of the 4-qubit quantum system under the MI operation, when the initial state is $|1010\rangle$. The figure shows the evolution of the probabilities of the computational basis states $|1010\rangle$ and $|0101\rangle$ (the probabilities of the remaining states have not been shown because at the final time they are zero). From Figure 5.1, we can see that the final state after 12.5ns is $|0101\rangle$, which shows the implementation of the MI operation between qubits Q_1 through Q_4 for the given input state.

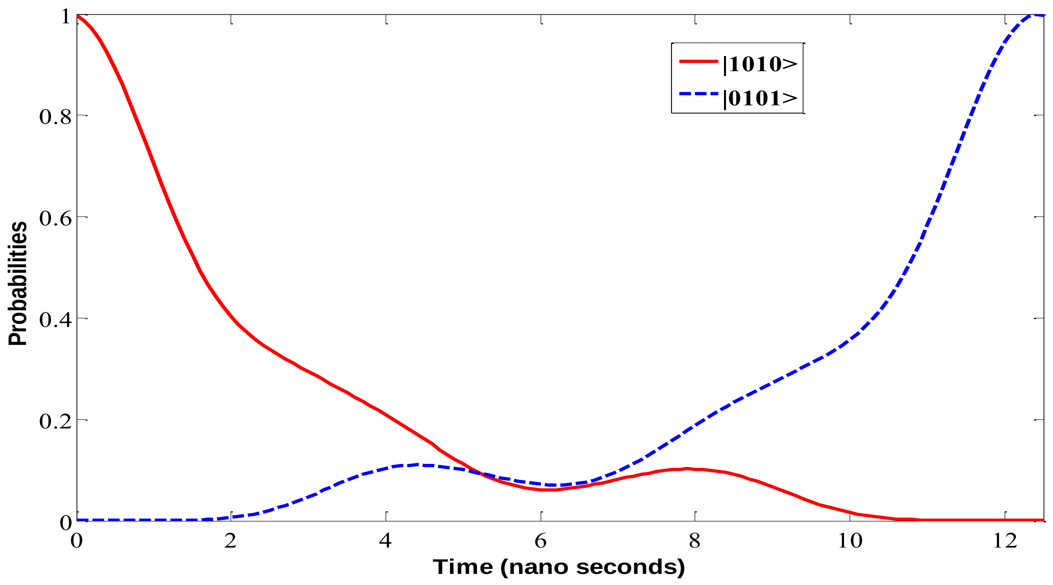

Figure 5.1. Implementation of an MI gate operation on qubits Q_1 through Q_4, using parameters of the trained network. The initial and final states of the system are $|1010\rangle$ and $|0101\rangle$, respectively. The evolution of the probabilities of the basis states $|1010\rangle$ and $|0101\rangle$ have been plotted, which shows the implementation of the desired 4-qubit MI gate operation for the given input.

Table 5.2 lists the parameters of the trained network in a 5-qubit LNN system. Here, both the worst-case fidelity and the overall fidelity of the gate operation were 100%. Simulations confirmed the MI gate operation on qubits Q_1 through Q_5, which was realized up to an overall global phase of 180 degrees that can be ignored.

TABLE 5.1

PARAMETERS OF THE TRAINED NETWORK TO REALIZE AN MI GATE OPERATION BETWEEN QUBITS Q_1 THROUGH Q_4.

Tunneling	Bias	Coupling
Δ_1 = 40.3 MHz	ε_1 = 29.7 MHz	ξ_{12} = 45.3 MHz
Δ_2 = 49.4 MHz	ε_2 = 0 MHz	ξ_{23} = 49.4 MHz
Δ_3 = 49.4 MHz	ε_3 = 0 MHz	ξ_{34} = 45.3 MHz
Δ_4 = 40.3 MHz	ε_4 = 29.7 MHz	Time, T_f = 12.5ns

TABLE 5.2

PARAMETERS OF THE TRAINED NETWORK TO REALIZE AN MI GATE OPERATION BETWEEN QUBITS Q_1 THROUGH Q_5.

Tunneling	Bias	Coupling
Δ_1 = 42 MHz	ε_1 = 31.2 MHz	ξ_{12} = 48.8 MHz
Δ_2 = 53.2 MHz	ε_2 = 0 MHz	ξ_{23} = 55.7 MHz
Δ_3 = 56.2 MHz	ε_3 = 0 MHz	ξ_{34} = 55.7 MHz
Δ_4 = 53.2 MHz	ε_4 = 0 MHz	ξ_{45} = 48.8 MHz
Δ_5 = 42 MHz	ε_5 = 31.2 MHz	Time, T_f = 13.3ns
RMS error for the network = 0.0003		

As an example, Figure 5.2 shows the evolution of the 5-qubit quantum system under the MI operation, when the initial state is $|01001\rangle$. The figure shows the evolution of the probabilities of the computational basis states $|01001\rangle$ and $|10010\rangle$ (the probabilities of the remaining states have not been shown because at the final time they are zero). From Figure 5.2, we can see that the final state after 13.3ns is |10010⟩, which shows the implementation of the MI operation between qubits Q_1 through Q_5 for the given input state.

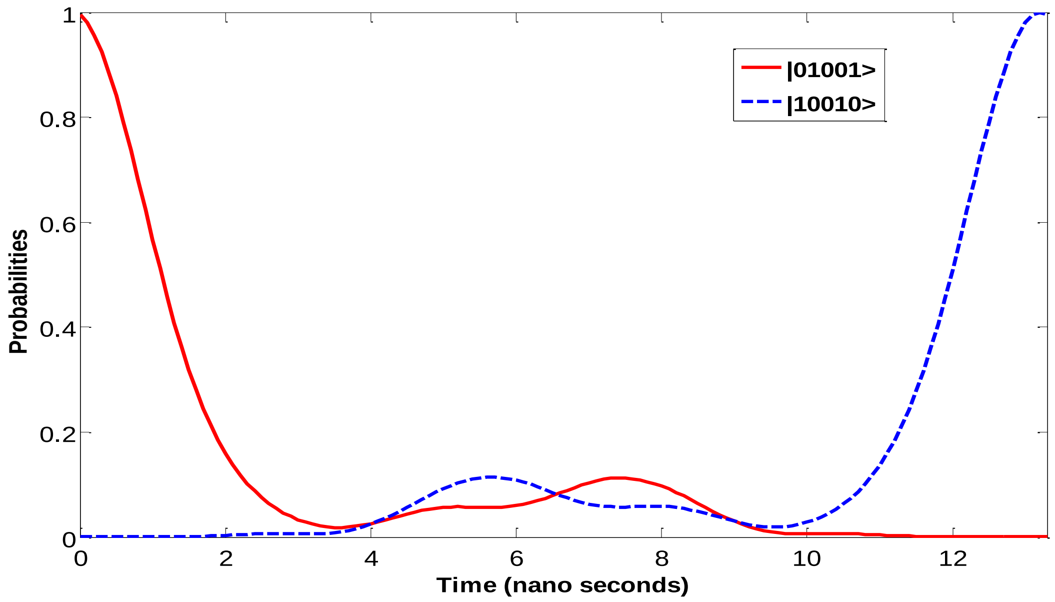

Figure 5.2. Direct implementation of an MI gate operation on qubits Q_1 through Q_5. The initial and final states of the system are |01001⟩ and |10010⟩, respectively. The evolution of the probabilities of the basis states |01001⟩ and |10010⟩ have been plotted, which shows the implementation of the desired 5-qubit MI gate operation for the given input.

Table 5.3 lists the parameters of the trained network for a 6-qubit LNN system. Here, the worst-case and overall fidelities of the gate operation were 99.64% and 99.74%, respectively. Simulations confirmed the MI gate operation on qubits Q_1 through Q_6, which was realized up to an overall global phase of 45 degrees that can be ignored.

As an example, Figure 5.3 shows the evolution of the 6-qubit quantum system under the MI operation, when the initial state is |010101⟩. The figure shows the evolution of the probabilities of the computational basis states |010101⟩ and |101010⟩ (the probabilities of the remaining states have not been shown, because at the final time they are zero). From Figure 5.3, we can see that the final state after 15ns is $|101010\rangle$, which shoes the implementation of the MI operation between qubits Q_1 through Q_6 for the given input state.

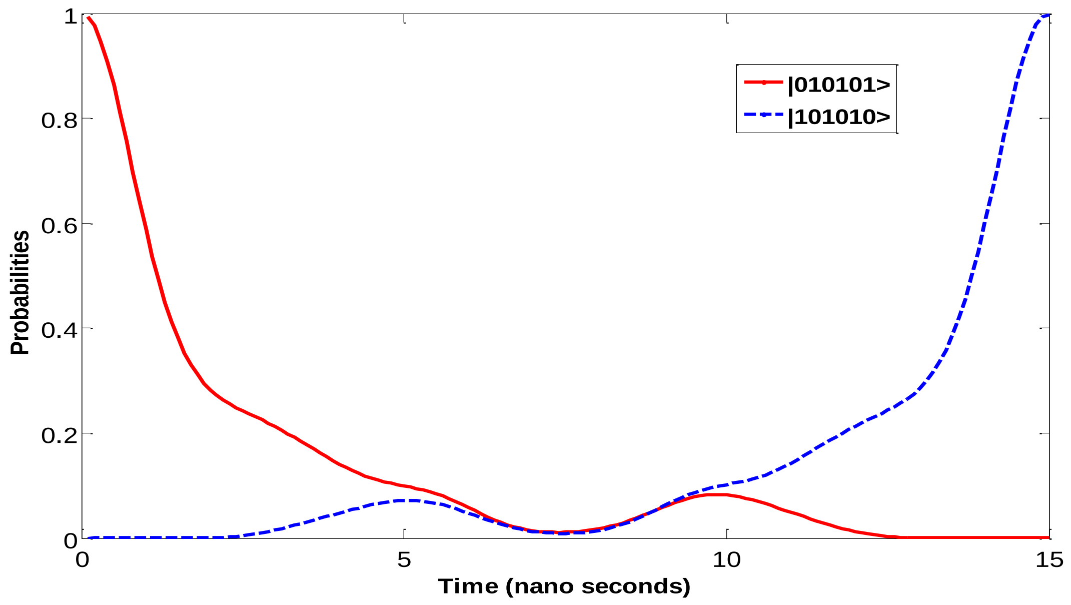

Figure 5.3. Direct implementation of an MI gate operation on qubits Q_1 through Q_6. The initial and final states of the system are $|010101\rangle$ and $|101010\rangle$, respectively. The evolution of the probabilities of the basis states $|010101\rangle$ and $|101010\rangle$ have been plotted, which shows the implementation of the desired 6-qubit MI gate operation for the given input.

TABLE 5.3

PARAMETERS OF THE TRAINED NETWORK TO REALIZE AN MI GATE OPERATION BETWEEN QUBITS Q_1 THROUGH Q_6.

Tunneling	Bias	Coupling
Δ_1 = 41.1 MHz	ε_1 = 29.8 MHz	ξ_{12} = 48.2 MHz
Δ_2 = 51.6 MHz	ε_2 = 0 MHz	ξ_{23} = 56.4 MHz
Δ_3 = 58 MHz	ε_3 = 0 MHz	ξ_{34} = 59.1 MHz
Δ_4 = 58 MHz	ε_4 = 0 MHz	ξ_{45} = 56.4 MHz
Δ_5 = 51.6 MHz	ε_5 = 0 MHz	ξ_{56} = 48.2 MHz
Δ_6 = 41.1 MHz	ε_6 = 29.8 MHz	Time, T_f = 15ns
RMS error for the network = 0.0026		

Table 5.4 lists the parameters of the trained network in a 7-qubit LNN system. Here, the worst-case fidelity of the gate operation was 99.68% and the overall fidelity of the gate operation was 99.77%. Simulations confirmed the MI gate operation on qubits Q_1 to Q_7 for the given input state.

TABLE 5.4

PARAMETERS OF THE TRAINED NETWORK TO REALIZE AN MI GATE OPERATION BETWEEN QUBITS Q_1 THROUGH Q_7.

Tunneling	Bias	Coupling
$\Delta_1 = 40.9$ MHz	$\varepsilon_1 = 29.3$ MHz	$\xi_{12} = 48$ MHz
$\Delta_2 = 52.2$ MHz	$\varepsilon_2 = 0$ MHz	$\xi_{23} = 56.6$ MHz
$\Delta_3 = 59.7$ MHz	$\varepsilon_3 = 0$ MHz	$\xi_{34} = 60.8$ MHz
$\Delta_4 = 61.4$ MHz	$\varepsilon_4 = 0$ MHz	$\xi_{45} = 60.8$ MHz
$\Delta_5 = 59.7$ MHz	$\varepsilon_5 = 0$ MHz	$\xi_{56} = 56.6$ MHz
$\Delta_6 = 52.2$ MHz	$\varepsilon_6 = 0$ MHz	$\xi_{67} = 48$ MHz
$\Delta_7 = 40.9$ MHz	$\varepsilon_7 = 29.3$ MHz	Time, $T_f = 16.3$ns
RMS error for the network = 0.0023		

As an example, Figure 5.4 shows the evolution of a 7-qubit quantum system under the MI gate operation, when the initial state is $|0100101\rangle$. The figure shows the evolution of the probabilities of the computational basis states $|0100101\rangle$ and $|1010010\rangle$ (the probabilities of the remaining states have not been shown because at the final time they are zero). From Figure 5.4, we can see that the final state after 16.3ns is $|1010010\rangle$, which shows the implementation of the MI gate operation between qubits Q_1 through Q_7 for the given input state.

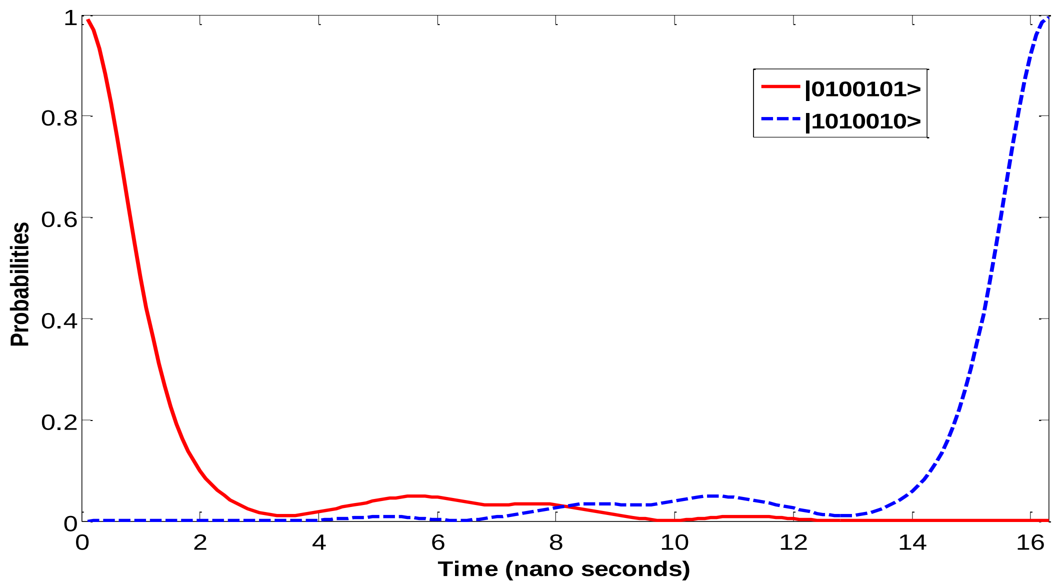

Figure 5.4. Implementation of an MI gate operation on qubits Q_1 through Q_7, using the parameters of the trained network. The initial state of the system is $|0100101\rangle$, and the final state is $|1010010\rangle$. The evolution of probabilities of the basis states $|0100101\rangle$ and $|1010010\rangle$ have been shown, which shows the implementation of the 7-qubit MI gate operation for the given input.

From the above results, we can observe that, to realize MI operations, we only need to change the bias on the first and last qubits, and fix the bias on the middle qubits to zero. Also, the tunneling parameters are mirror symmetric about the central qubit/s. That is, when N is even, $\Delta_i = \Delta_{N-i+1}$ for i = 1, 2, 3, … N/2; when N is odd, $\Delta_i = \Delta_{N-i+1}$ for i = 1, 2, 3, … ((N-1)/2)+1). Likewise, the coupling parameters follows a similar mirror symmetry ($\xi_{1,2} = \xi_{N-1,N}$, etc.).

Note that, swap gate operations between remote qubits Q_1 and Q_N in an N qubit LNN system can be achieved by applying two successive MI operations. The first MI operation is performed on qubits Q_1 through Q_N. The second MI operation is performed on qubits Q_2 through Q_{N-1}, which reverses back the states of qubits Q_2 through Q_{N-1}. As such, at the end of both MI operations only the states of Q_1 and Q_N are interchanged.

5.2 Training the Quantum System for Mirror Inversion in the Presence of Unwanted Couplings

In section 5.1, we have shown that our dynamic learning algorithm can be used to perform MI operations on nearest neighbor qubits. In this section, we use the learning algorithm to find parameters for implementing MI in the presence of unwanted couplings. Consider Figure 5.5(a), which shows five qubits, Q_1 through Q_5. Here, qubits Q_1 through Q_4 comprise the LNN chain along which we want to perform MI. Qubit Q_5 is an additional qubit that is "unwantedly" coupled to qubit Q_1. The Hamiltonian, H is:

$$H = \sum_{i=1}^{5} (\Delta_i \sigma_{Xi} + \varepsilon_i \sigma_{Zi}) + \sum_{i=1}^{3} \xi_{i,i+1} \sigma_{Zi} \sigma_{Zi+1} + \xi_{1,5} \sigma_{Z1} \sigma_{Z5} \tag{5.2}$$

where the parameters have the usual meaning. Here, we want to perform MI operation on qubits Q_1 through Q_4 without having to switch off the coupling between Q_1 and Q_5. For training, initially, all the couplings are set to 5MHz. Also, while training, all the parameters, except the coupling between Q_1 and Q_5, which is fixed at 5MHz, are treated as variables.

The training set comprises of the thirty two computational basis states, and their respective expected final states after the mirror inverse operation between qubits Q_1 through Q_4. That is, the transformation for the gate operation is given as:

$$q_1 q_2 q_3 q_4 q_5\rangle \rightarrow |q_4 q_3 q_2 q_1 q_5\rangle \; q_1 q_2 q_3 q_4 q_5\rangle \rightarrow |q_4 q_3 q_2 q_1 q_5\rangle \tag{5.3}$$

Using the training process explained in Chapter 3, we trained the system until the RMS error value was 0.0034. Table 5.5 lists the parameters of the trained network.

TABLE 5.5

PARAMETERS OF THE TRAINED NETWORK TO REALIZE AN MI GATE OPERATION BETWEEN QUBITS Q_1 THROUGH Q_4 WITH QUBIT Q_1 "UNWANTEDLY" COUPLED TO AN ADDITIONAL QUBIT Q_5.

Tunneling	Bias	Coupling
Δ_1 = 100 MHz	ε_1 = 75 MHz	ξ_{12} = 114.6 MHz
Δ_2 = 122.5 MHz	ε_2 = 0 MHz	ξ_{23} = 125.1 MHz
Δ_3 = 122.5 MHz	ε_3 = 0 MHz	ξ_{34} = 114.6 MHz
Δ_4 = 100 MHz	ε_4 = 75 MHz	ξ_{45} = 5 MHz
Δ_5 = 25 MHz	ε_5 = 96.9 MHz	Time, T_f = 5ns
RMS error for the network = 0.0034		

To confirm the gate operation for the parameters listed in Table 5.5, we substituted these parameters in a 5-qubit Hamiltonian (equation (5.2)), and ran simulations. Here, the worst-case and overall fidelities of the gate operation were 99.55% and 99.67%, respectively. The simulations confirmed the MI gate operation on qubits Q_1 through Q_4, which was realized up to an overall global phase of -134 degrees that can be ignored. To demonstrate that the MI operation is indeed a quantum operation, we considered the case when qubits Q_1 and Q_5 are entangled, initially. As an example, we considered the 5-qubit system to be in a superposition of states $|00101\rangle$ and $|10100\rangle$ with probability amplitudes of ½ and √(3)/2, respectively. Here, qubits Q_1 and Q_5 are in the entangled state $\frac{1}{2}(|01\rangle+\sqrt{(3)}|10\rangle)$; qubits Q_2, Q_3, Q_4 are in the state of $|010\rangle$. The final state was found to be a superposition of the states of $|01001\rangle$ and $|01010\rangle$ with probability amplitudes of ½ and √(3)/2 respectively, which confirms both the MI gate operation between qubits Q_1 through Q_4, and also the transfer of entanglement between qubits Q_1 and Q_5 to qubits Q_4 and Q_5.

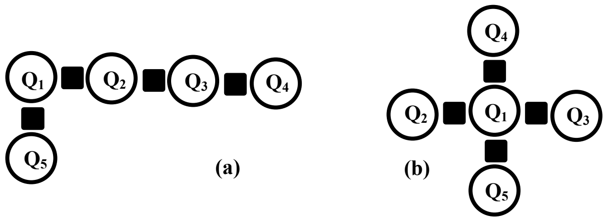

Figure 5.5(a). Linear nearest neighbor array comprising of four qubits Q_1, Q_2, Q_3, Q_4. Here, qubit Q_5 is an additional qubit that is "unwantedly" coupled to qubit Q_1. Figure 5.5(b) Five-qubit system where qubit Q_1 is coupled to four qubits Q_2, Q_3, Q_4 and Q_5.

Figure 5.6 shows the simulation results where the probabilities of the basis states $|00101\rangle$, $|10100\rangle$, $|01001\rangle$ and $|01010\rangle$ have been plotted. The simulations confirmed the MI operation for the given input state. Moreover, the simulations also confirmed that MI is achieved between qubits Q_1 through Q_4 using the same set of parameters, irrespective of which one of the four qubits qubit Q_5 is coupled to. Also, simulations showed that, for the parameters shown in the Table 5.6, if the coupling between qubits Q_1 and Q_5 is increased, the RMS error increases (i.e., if the value, ξ_{15} is changed from 5MHz to 10MHz, the RMS error value changes from 0.0034 to 0.013). However, the error can be decreased by further training.

As a special case to this discussion, we considered a new 5-qubit Ising coupled system with qubit Q_1 coupled to four qubits Q_2, Q_3, Q_4 and Q_5, as shown in Figure 5.5(b). The Hamiltonian is:

$$H = \sum_{i=1}^{5} (\Delta_i \sigma_{Xi} + \varepsilon_i \sigma_{Zi}) + \sum_{i=2}^{5} \xi_{1,i} \sigma_{Z1} \sigma_{Zi} \tag{5.4}$$

Here, we want to perform a MI operation between qubits Q_2 and Q_3 without switching off the couplings between qubits Q_1, Q_4, and Q_1, Q_5 (i.e., without making ξ_{14} and ξ_{15} zero). Note that the MI operation is equivalent to a swap operation between qubits Q_2 and Q_3. The transformation for the gate operation is given as:

$$|q_1 q_2 q_3 q_4 q_5\rangle \rightarrow |q_1 q_3 q_2 q_4 q_5\rangle \tag{5.5}$$

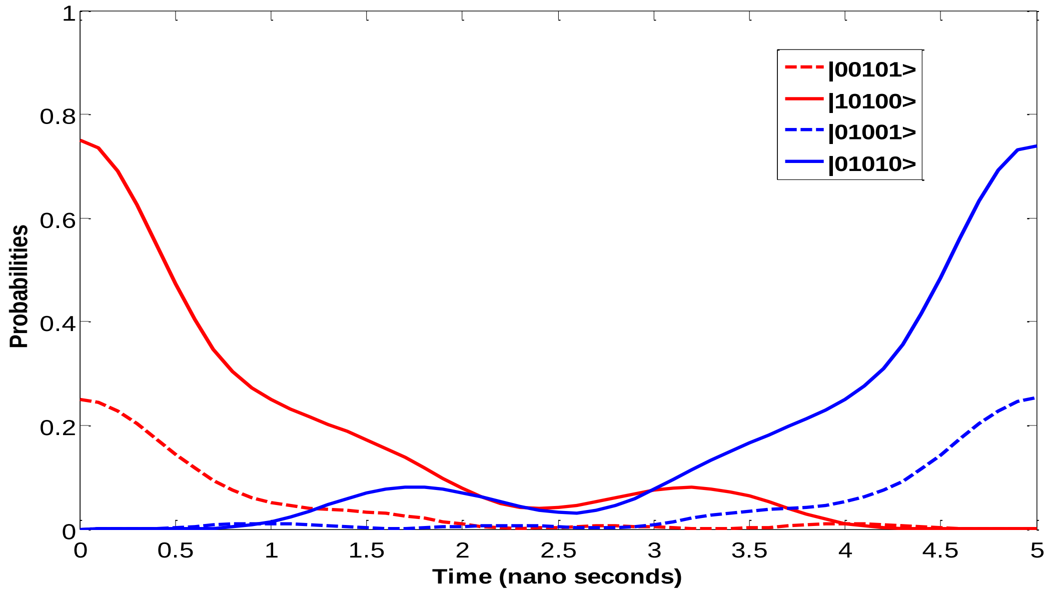

Figure 5.6. Implementation of an MI gate operation on qubits Q_1 through Q_4 with an additional qubit Q_5 "unwantedly" coupled to qubit Q_1. The initial state is (1/2) $|00101\rangle$ + ($\sqrt{3}/2$) $|10100\rangle$ and the final state is (1/2) $|01001\rangle$ + ($\sqrt{3}/2$) $|01010\rangle$. The evolution of the probabilities of the basis states $|00101\rangle$, $|10100\rangle$, $|01001\rangle$ and $|01010\rangle$ have been plotted, which shows the implementation of the desired MI gate operation for the given input.

At the start of training, we initialized all the tunneling parameters to 100MHz, bias parameters to 75MHz, and the coupling parameters to 5MHz. During the training process, we only trained the tunneling parameter of qubit Q_1, bias parameters of qubits Q_1, Q_2, Q_3, and the coupling between qubits Q_1, Q_2 and qubits Q_1, Q_3. We did not train the remaining parameters. The training set comprised of the 32 computational basis states and their respective expected final states after the swap gate operation between qubits Q_2 and Q_3.

Using the training process explained in Chapter 3, we trained the system until the RMS error value was 0.0047. Table 5.6 lists the parameters for the trained network. As can be seen from the training results, the network trains the couplings ξ_{12} and ξ_{13} to be values much greater than couplings ξ_{14} and ξ_{15}, which in this case, are "unwanted" couplings.

TABLE 5.6

PARAMETERS OF THE TRAINED NETWORK TO REALIZE AN MI GATE OPERATION BETWEEN QUBITS Q_2 AND Q_3 WITH QUBIT Q_1 COUPLED TO QUBITS, Q_2, Q_3, Q_4 AND Q_5.

Tunneling	Bias	Coupling
Δ_1 = 113.5 MHz	ε_1 = 0 MHz	ξ_{12} = 109.5 MHz
Δ_2 = 100 MHz	ε_2 = 75 MHz	ξ_{13} = 109.5 MHz
Δ_3 = 100 MHz	ε_3 = 75 MHz	ξ_{14} = 5 MHz
Δ_4 = 100 MHz	ε_4 = 54 MHz	ξ_{15} = 5 MHz
Δ_5 = 100 MHz	ε_5 = 54 MHz	Time, T_f = 4.4ns
RMS error for the network = 0.0047		

5.3 Validation of Our Learning Algorithm

In [60], the authors showed that MI gate operations can be implemented by modulating the couplings of a quantum spin chain with XY interactions. In [61], the authors used the GRAPE algorithm to find the parameters to implement MI gate operations in spin chains with XY interactions. Here, we want to use our learning algorithm to do the same and see how it performs when compared to the GRAPE algorithm. For this, we chose an N-qubit LNN system with XY interactions, as shown in Figure 4.1, whose Hamiltonian, $H_{N'}$ where 'N' represents the number of qubits in the system, is:

$$H_{N'} = \sum_{i=1}^{N} (\Delta_i \sigma_{Xi} + \varepsilon_i \sigma_{Zi}) + \sum_{i=1}^{N-1} \xi_{i,i+1} (\sigma_{Xi}\sigma_{Xi+1} + \sigma_{Yi}\sigma_{Yi+1}) \tag{5.6}$$

Using the training process explained in Chapter 3, we were able to train the system until the RMS error value was 0. Here, initially, we set all the parameters to be equal to 25MHz. The trained parameters are shown in Table 5.7. We can observe that the values of all parameters except the couplings are zero. This is similar to the results shown in [60, 61], where the authors showed that MI operations can be implemented directly by modulating the couplings in quantum systems with XY interactions (and by setting both bias and tunneling parameters to zero), which validates our learning algorithm.

TABLE 5.7

PARAMETERS OF THE TRAINED NETWORK TO REALIZE AN MI GATE OPERATION BETWEEN QUBITS Q_1 THROUGH Q_4 IN A 4-QUBIT LNN SYSTEM WITH XY INTERACTIONS.

Tunneling	Bias	Coupling
Δ_1 = 0 MHz	ε_1 = 0 MHz	ξ_{12} = 43.3 MHz
Δ_2 = 0 MHz	ε_2 = 0 MHz	ξ_{23} = 50 MHz
Δ_3 = 0 MHz	ε_3 = 0 MHz	ξ_{34} = 43.3 MHz
Δ_4 = 0 MHz	ε_4 = 0 MHz	Time, T_f = 5ns
RMS error for the network = 0		

5.4 Conclusions

In this chapter, we used our dynamic learning algorithm to implement MI gate operations in an LNN array. This was accomplished by training the quantum system using a back propagation technique, to find the system parameters that implement MI gate operations directly. The main advantage of our scheme is that by implementing the MI operation as a single operation, without decomposing it into a sequence of CNOT gates, the overall gate count can be reduced. Secondly, since all qubits along the chain participate in the MI operation, we can avoid idle times that can give rise to relative phases, and also avoid propagation errors due to one/more faulty CNOT gates. Thirdly, our method does not require additional ancillas and pre-engineered mirror-periodic Hamiltonians for implementing any of these MI operations. Lastly, all parameters found using our scheme are scalable, and therefore, can be adjusted to the requirements of a given experimental realization.

CHAPTER 6

FINDING ANALYTICAL SOLUTION FOR MIRROR INVERSE GATE OPERATIONS IN QUANTUM SYSTEMS WITH DIAGONAL INTERACTIONS

In this chapter, we present an analytical solution for implementing mirror inverse (MI) gate operations in quantum systems with diagonal (Ising type) interactions. Ising coupled systems are commonly seen in superconducting Josephson junctions, dipole-dipole or J-coupled systems [13-15, 111, 112]. Unlike non-diagonal XY interactions where a SWAP gate is easily implemented, in Ising coupled systems each swap operation needs to be decomposed into 3 controlled-NOT (CNOT) gates. As such, methods for finding system parameters that allow us to transmit qubit states along LNN arrays without requiring swap gates can greatly reduce the overall computational overhead in these systems. In Chapter 5, we showed that MI gate operations can be implemented directly in Ising coupled systems without having to decompose the operation into a sequence of CNOT gates. To achieve this, we used the dynamic learning algorithm as a tool for finding parameters of the Ising-coupled system Hamiltonian that realizes the operation. Here, we first choose a Hamiltonian, and then train its parameters to implement the desired MI gate operation within a chosen time duration (t_f). Once the parameter values are known, to implement the MI operation we simply set the system parameters to the values solved for, and then allow the system to evolve for time, t_f. However, as the number of qubits increases, the number of system parameters also increase, which makes the training process complex and hard to converge. By carefully analyzing the parameter values obtained from training systems with fewer qubits, we try to find analytical solutions that can be extended to calculate parameter values to achieve MI in systems with large number of qubits without the need for training.

The chapter is divided as follows. In section 6.1, we show how we can use the parameters found by using the training algorithm for implementing MI between nearest neighbor qubits in an Ising-coupled LNN system to generate an analytical solution. In section 6.2, we present the conclusions.

6.1 Analytical Solution for Mirror Inverse Gate Operations in Quantum Systems with Diagonal Interactions

Consider a quantum system of N qubits arranged in an LNN array as shown in Figure 4.1. Here, each qubit only interacts with its nearest neighbors through Ising type interactions, which are diagonal in the interaction basis. The Hamiltonian, H_N, where 'N' represents the number of qubits in the system, is given by equation (4.1).

In [60, 61], the authors presented that in a quantum system with XY interactions, the MI operations can be implemented *directly* by tuning the coupling values to $J_i = \sqrt{(i(N-i)}$, where J is the coupling constant, and N is the number of qubits. Following their approach, we carefully analyzed the system parameters we obtained from training Ising coupled systems for MI operations (system of 4, 5, 6 and 7 qubits), and generated similar equations that can be used to calculate parameters to implement MI in systems with larger numbers of qubits without having to train the system. To achieve this, we used our training results presented in Chapter 5, for implementing MI operations in 4, 5, 6, and 7 qubit systems. Since each of the four MI operations used different time steps to implement the MI operation, we scaled all the parameters so that the MI operation in each system was implemented in 10ns (i.e., t_f = 10ns). Tables 6.1 and 6.2 show the tunneling and coupling parameters for MI gate operations after scaling the time step to 10 ns. By observing the parameters presented in Tables 6.1 and 6,2, we noticed a symmetry between the parameters with respect to the centre of the chain similar to that presented in [60, 61]. Using this symmetry, we derived the following expressions for the tunneling and coupling parameters for an arbitrary "N" number of qubits:

$$\Delta_i = \delta_{\Delta i} D_i \tag{6.1}$$

$$\xi_i = \delta_{\xi i} J_i \tag{6.2}$$

where D_i and J_i can be calculated by direct substitution:

$$D_i = \sqrt{i(N+i-1)} \tag{6.3}$$

$$J_i = \sqrt[3]{i(N-i)} \tag{6.4}$$

Here, i =1, 2… N, and $\delta_{\Delta i}$ and $\delta_{\xi i}$ are the tunneling and coupling multiplication factors, respectively. For different values of N and i, these factors are calculated using the ratios $\delta_{\Delta i} = \Delta_i/D_i$, and $\delta_{\xi i} = \xi_i/J_i$, respectively. The tunneling multiplication factor was found to be 25 in all cases. That is, the tunneling parameters for any arbitrary N qubit system can be found using equation (6.1), where $\delta_{\Delta i} = 25$. The coupling multiplication factor varied, and the values have been listed in Table 6.3. Note that, we have also included the parameters for implementing 2-qubit and 3-qubit swap gate operations (shown in Chapter 4) in this table as these gate operations can be considered as special cases of the MI gate operations.

TABLE 6.1

TUNNELING PARAMETERS FOR MI IN 4-, 5-, 6-, 7-QUBIT SYSTEMS (FOUND USING THE DYNAMIC LEARNING ALGORITHM) AFTER SCALING THEM FOR A TOTAL TIME OF 10NS.

4-Qubits	**5-Qubits**	**6-Qubits**	**7-Qubits**
		Δ_1=61.5MHz	Δ_1= 66.67MHz
Δ_1=50.38MHz	Δ_1=55.86MHz	Δ_2=77.85MHz	Δ_2= 85.1MHz
Δ_2=61.75MHz	Δ_2=70.76MHz	Δ_3=86.25MHz	Δ_3= 97.31MHz
Δ_3=61.75MHz	Δ_3=74.75MHz	Δ_4=86.25MHz	Δ_4=100.1MHz
Δ_4=50.38MHz	Δ_4=70.76MHz	Δ_5=77.85MHz	Δ_5= 97.3MHz
	Δ_5=55.86MHz	Δ_6=61.5MHz	Δ_6= 85.1MHz
			Δ_7= 66.67MHz

TABLE 6.2

COUPLING PARAMETERS FOR MI IN 4-, 5-, 6-, 7-QUBIT SYSTEMS (FOUND USING THE DYNAMIC LEARNING ALGORITHM) AFTER SCALING THEM FOR A TOTAL TIME OF 10NS.

4-Qubits	**5-Qubits**	**6-Qubits**	**7-Qubits**
		ξ_1=72.3MHz	ξ_1= 78.24MHz
	ξ_1=64.9MHz	ξ_2=84.45MHz	ξ_2= 92.26MHz
ξ_1=56.63MHz	ξ_2=74.1MHz	ξ_3=88.35MHz	ξ_3=99.1MHz
ξ_2=61.75MHz	ξ_3=74.1MHz	ξ_4=84.45MHz	ξ_4=99.1MHz
ξ_3=56.63MHz	ξ_4=64.9MHz	ξ_5=72.3MHz	ξ_5= 92.26MHz
			ξ_6= 78.24MHz

TABLE 6.3

MULTIPLICATION FACTORS FOR THE COUPLING PARAMETERS FOR MI IN 4-, 5-, 6-, 7-QUBIT SYSTEMS (FOUND USING THE DYNAMIC LEARNING ALGORITHM) AFTER SCALING THEM FOR A TOTAL TIME OF 10NS.

2-Qubits	3-Qubits	4-Qubits	5-Qubits	6-Qubits	7-Qubits
				$\delta_{\xi 1}$= 42.3	$\delta_{\xi 1}$= 42.98
			$\delta_{\xi 1}$= 40.82	$\delta_{\xi 2}$= 42.23	$\delta_{\xi 2}$= 42.98
	$\delta_{\xi 1}$= 38.4	$\delta_{\xi 1}$= 39.2	$\delta_{\xi 2}$= 40.7	$\delta_{\xi 3}$= 42.47	$\delta_{\xi 3}$= 43.27
$\delta_{\xi 1}$= 37.3	$\delta_{\xi 2}$= 38.4	$\delta_{\xi 2}$= 38.83	$\delta_{\xi 3}$= 40.7	$\delta_{\xi 4}$= 42.23	$\delta_{\xi 4}$= 43.27
		$\delta_{\xi 3}$= 39.2	$\delta_{\xi 4}$= 40.82	$\delta_{\xi 5}$= 42.3	$\delta_{\xi 5}$= 42.98
					$\delta_{\xi 6}$= 42.98

Observing Table 6.3, since the coupling multiplication factor is not constant in value for a given value of N except for N = 2 and N = 3 (unlike the tunneling factor, which was 25 in all cases), in order to find an equation for predicting this factor for any N qubit system, we plotted these factors for N = 2 through 7 qubit systems to generate an equation. Note that for a given N qubit system, since there were N-1 factors, we took the average of these values when plotting. For instance, for N = 4, there were 3 values 39.2, 38.83, and 39.2, and we considered the average of these 3 values in plotting. From the plot, we were able to generate a general expression for finding the multiplication factor for an arbitrary N qubit system:

$$\delta_{\xi} = 1.2096*N + 34.709 \tag{6.5}$$

where δ_{ξ} is the average coupling multiplication factor. Figure 6.1 shows the plot that was used to generate equation (6.5). Note that using equation (6.5), we can find the coupling factor for any N qubit system, and then use equation (6.2) to find individual coupling parameters.

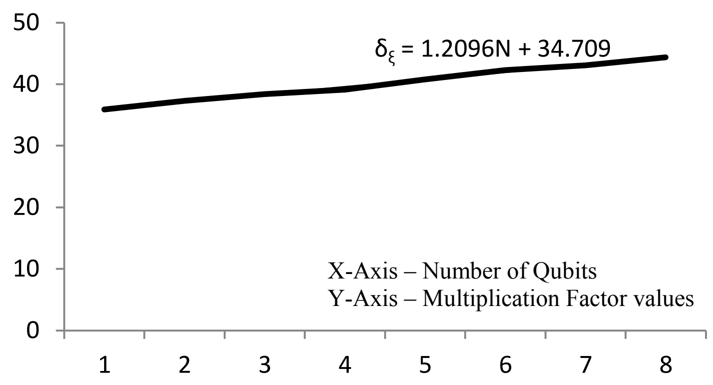

Figure 6.1. Finding the expression to generate the multiplication factors to calculate the coupling parameters for any MI gate operation. Here, for each value of N (number of qubits), we calculated the average value of the (N-1) multiplication factors.

For calculating the bias parameters, we use a similar technique. From the results shown in Chapter 5, we observe that for the MI operation, a bias is only applied on the first and last qubits, while the bias on the remaining N-2 qubits is zero. Moreover, the biases on the first and last qubits are equal in magnitude. Therefore, to generate an expression for the bias value on the first and last qubits, we directly plotted the bias as a function of N (after scaling them to a time step of 10 ns). Figure 6.2 shows the plot of the bias as a function of N for N = 2 through 7 qubits. Using the plot, we generated the following equation for calculating the bias for an arbitrary value on N:

$$\varepsilon = 3.9832*N + 20.766 \quad (6.6)$$

Therefore, using equations (6.1) through (6.6), we are able to implement MI operations in any N qubit system.

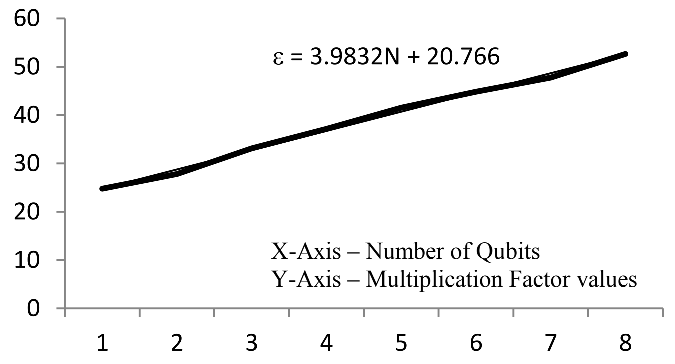

Figure 6.2. Finding the expression to generate the bias parameters for the any MI gate operation.

Next, to verify these equations, we used them to find parameters for an 8-qubit system. Here, the multiplication factor for the tunneling parameters is 25, and the corresponding tunneling parameters can be calculated using equation (6.1) with D_1=D_8=2.83 MHz, D_2=D_7=3.74 MHz, D_3=D_6=4.24 MHz, and D_4=D_5=4.47 MHz, respectively. Similarly, the multiplication factor for the coupling parameters is 44.4 (equation (6.5)), and the corresponding coupling parameters are calculated using equation (6.2) with J_1=J_7=1.91 MHz, J_2=J_6=2.29 MHz, J_3=J_5=2.47 MHz, and J_4=2.52 MHz, respectively. Table 6.4 shows the parameters to implement the 8-qubit MI gate operation. Simulations were run on the 8-qubit system with T = 10ns (Hamiltonian given by equation (4.1) with N = 8). The simulations confirmed the MI gate operation on qubits Q_1 through Q_8 with an overall fidelity of 99.7%. As an example, Figure 6.3 shows the evolution of the system for the initial state $|01010101\rangle$. The figure shows the evolution of the probabilities of the computational basis states $|01010101\rangle$ and $|10101010\rangle$ (the probabilities of the remaining states have not been shown because at the final time they are zero). From the figure, we can see that the $|01010101\rangle$ state evolves to the $|10101010\rangle$ state, which shows the implementation of the expected MI gate operation for the given input.

TABLE 6.4

PARAMETERS TO REALIZE 8-QUBIT MI GATE OPERATION BETWEEN QUBITS Q_1 THROUGH Q_8 PREDICTED USING THE ANALYTICAL SOLUTION.

Tunneling	**Bias**	**Coupling**
Δ_1 = 70.7 MHz	ε_1 = 52.63 MHz	ξ_{12} = 84.8 MHz
Δ_2 = 93.5 MHz	ε_2 = 0 MHz	ξ_{23} = 101.7 MHz
Δ_3 = 106 MHz	ε_3 = 0 MHz	ξ_{34} = 109.7 MHz
Δ_4 = 111.75 MHz	ε_4 = 0 MHz	ξ_{45} = 111.9 MHz
Δ_5 = 111.75 MHz	ε_5 = 0 MHz	ξ_{56} = 109.7 MHz
Δ_6 = 106 MHz	ε_6 = 0 MHz	ξ_{67} = 101.7 MHz
Δ_7 = 93.5 MHz	ε_7 = 0 MHz	ξ_{78} = 84.8 MHz
Δ_8 = 70.7 MHz	ε_8 = 52.63 MHz	Time, T_f = 10nS
Fidelity of the gate operation = 99.7%		

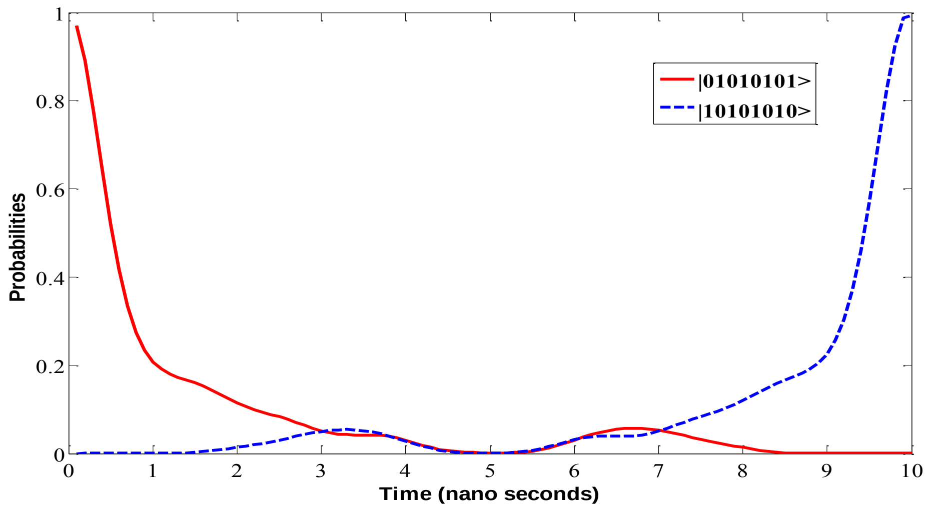

Figure 6.3. Direct implementation of an MI gate operation between qubits Q_1 through Q_8. The initial and final states of the system are $|01010101\rangle$ and $|10101010\rangle$, respectively. The evolution of the probabilities of the basis states $|01010101\rangle$ and $|10101010\rangle$ have been plotted, which shows the implementation of the desired 8-qubit MI gate operation for the given input.

6.2 Conclusions

In this chapter, we introduced an analytical solution to implement MI gate operations in an LNN array with Ising type interactions. This was accomplished by carefully analyzing the parameter values obtained in Chapter 5, and deriving analytical solutions that can be generalized for any LNN system with arbitrary number of qubits. This analytical solution can be used to calculate parameter values to achieve MI gate operations in systems with any numbers of qubits without the need for training.

CHAPTER 7

REDUCTION OF QUANTUM CIRCUITS USING A DYNAMIC LEARNING ALGORITHM

The purpose of this chapter is two-fold. One is to show how our dynamic algorithm can be used to implement complicated quantum circuit operations directly without having to decompose them into a series of gate operations. To this end, we choose two examples.

(a) Quantum Fourier Transform (QFT): The QFT is the quantum analog of the classical Fourier transform which is used in many quantum algorithms like finding the prime factors of an integer [113], and quantum phase estimation algorithms [114]. QFT can be implemented by decomposing the operation into a sequence of single-qubit Hadamard, 2-qubit controlled-phase shift and swap gates [1, 115]. Here, we will show that a 2-qubit QFT can be implemented directly without decomposing it into a series of gate operations using our dynamic learning algorithm. In this example, training can be hard as phases are involved.

(b) Three-Qubit Encoded-Hadamard Gate: The implementation of this gate will help with fault tolerant quantum computation. An encoded-Hadamard (e-H) gate operation on the 3-qubit code is the application of the equivalent of Hadamard gate operation on the logical qubit without decoding. To our knowledge, no method of implementing the e-H gate on a 3-qubit logical qubit exists. The 3-qubit code is a simple error correction code to correct single qubit bit flip (or phase flip) errors. In order to perform gate operations on encoded (or logical) qubits, a universal set of "encoded" gates is required. However, not all the encoded schemes will allow these gates to be implemented without decoding the actual encoded qubit. For instance, to perform an

H-gate operation on the encoded qubit, we first need to decode it and then apply the H-gate on the physical data qubit (each encoded qubit consists of one data qubit and two ancillas (qubits in $|0\rangle$ state)). This not only increases the computational overhead, but also negates all the advantages of encoded qubits. Here, we show how we can implement an e-H gate *directly* on the 3-qubit error correction code, without decoding the encoded qubit using our learning algorithm.

The second motivation of this work is to improve upon our dynamic learning technique by making improvements to the learning algorithm [116-118] proposed in Chapter 3. Previously, we used the learning rate, α, for controlling the rate of gradient descent, i.e., the parameter, α, indicates how far to go along the direction of the current gradient to update the weights after each iteration. However, a disadvantage is that the value chosen for the parameter α, affects the convergence time of the training process. If we choose the learning rate to be small, the update of weights will also be small from one iteration to the next, which effects the convergence time of the training process. If we choose α, to be large, in order to speed up the learning, the resulting large changes in the weights may affect the learning procedure adversely by making it unstable, i.e., it can lead to steps that skip the minimum, resulting in a large increase in error. To overcome this dependence of the learning rate on α, several modifications can be made to the back-propagation algorithm. One such technique is to use an alternative weight update procedure by adding a momentum factor to the weight update rule which helps the neural network to converge faster [119-122]. This momentum factor determines the amount of influence on the present iteration from the results of the previous one, by changing the weights in the direction that is a combination of current and previous gradients. In addition to speeding up the training process, in some cases, the addition of the momentum factor also avoids the training process from being trapped in a local minimum, by helping it to skip over those regions without training the network in these regions. To demonstrate the improvement in our learning algorithm after including the momentum parameter, we use the QFT and the e-H gates as examples for comparison.

The chapter is divided as follows. In section 7.1, we present the training algorithm. In section 7.2, we apply our training process to find the parameters for the 2-qubit QFT, and the e-H gate operation on the 3-qubit error correction code. In section 7.3, we compare the performance of the learning algorithms with and without the momentum factor. In section 7.4, we present the conclusions.

7.1 Learning Algorithm for Training

As before, to construct the dynamic learning algorithm, we use the time evolution of the Schrödinger equation which is given by

$$\frac{\partial \rho}{\partial t} = -\frac{i}{\hbar}[H, \rho] \tag{7.1}$$

Here, ρ is the density matrix, ħ is plank's constant divided by 2π, and H is the Hamiltonian of the system. In the dynamic leaning algorithm, the basic idea is to force the output state, $|\psi\rangle_{\text{out}}$, of the system for a given input state, $|\psi\rangle_{\text{in}}$, to the desired state, $|\psi\rangle_{\text{des}}$, under a chosen gate operation, **U**, by controlling the parameters of the Hamiltonian (for instance, tunneling, bias and coupling). Initially, the system parameters are chosen randomly (while confining to experimental limits). They are then trained by using the gradient descent learning rule,

$$W(t+1) = W(t) - \alpha \frac{dL}{dw} + \mu \Delta W(t) \tag{7.2}$$

where

$$\Delta W(t) = W(t) - W(t-1) \tag{7.3}$$

Here, α is the learning rate, and μ is the momentum parameter, and the values of both are in the range 0 to 1. The terms W(t+1), W(t) and W(t-1) are the parameter values at training steps t+1, t, and t-1, respectively, and L is the Lagrangian which is used to find the local minimum of the error function. Previously, in our learning algorithm (described in Chapter 3), the weight change was always in the direction of the current gradient. However, from equation (7.3), we can observe that now it will be in a direction that is a combination of current and previous gradients. This can be done by saving the weights from one or more previous iterations, and the new weights are updated based on the value of the weights at training steps t and t-1, as shown in equation (7.3). Note that the dependence of the value of W(t+1) on the values at W(t-1) is through the momentum factor, μ. We randomly choose the α and μ values to train the network initially, and α and μ are fixed throughout the training. However, in some cases, the algorithm stops training after a few iterations and the training process gets trapped in a local minimum. To overcome this, we can start the training process with a different value for α and μ. For instance, while training for a 2-qubit QFT, both the momentum and learning factors were set to 0.085, initially. After 50 iterations, the training stopped, i.e., the RMS error failed to converge below 0.008. However, by changing the momentum factor to 0.1, we are able to reduce the RMS error value to 0.0008 (shown in section 7.4).

7.2 Training Results

In this section, we will demonstrate how our dynamic learning algorithm can be used as a tool for both circuit reduction and implementation of complex gate operations. First, we explain how this scheme can be used to implement quantum algorithms directly, without decomposing them into a series of gate operations, and consider the 2-qubit QFT as an example. Next, we explain how this scheme can be used to help with fault-tolerant quantum computation by implementing logic gate operations directly on encoded qubits without having to decode the qubit. As an example, we show how to implement an encoded-Hadamard (**e-H**) gate operation on a logical qubit for the 3-qubit error correction code.

Consider a quantum system of N qubits arranged in an LNN array as shown in Figure 4.1. Here, each qubit only interacts with its nearest neighbors, where we are considering Heisenberg-type interactions, which are typical of spin-coupled quantum dots [9, 112]. The Hamiltonian, H_2, for a system of two qubits A and B is given by equation (4.4).

Quantum Fourier Transforms

Quantum Fourier Transform (QFT) is the quantum analog of the classical Fourier Transform which is used in many quantum algorithms like finding the prime factors of an integer [113], and quantum phase estimation algorithms [114]. The mathematical representation of QFT is given by

$$|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle \tag{7.4}$$

where $|j\rangle$ and $|k\rangle$ are the computational basis of states an N (in this case, N = 2) qubit quantum system. The circuit for implementing the QFT, as shown in Figure 7.1, is obtained by decomposing the operation into a sequence of single-qubit Hadamard, two-qubit controlled-phase shift and swap gates [1, 115].

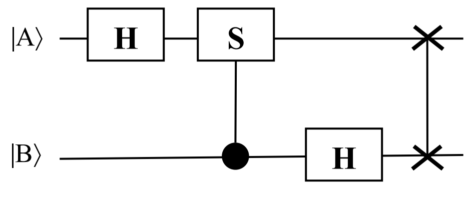

Figure 7.1. Circuit for implementing the QFT on a two qubit system.

$$QFT = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \tag{7.5}$$

From Figure 7.1, we can see that to realize a 2-qubit QFT, 4 gates are used. (Note that for a Heisenberg-coupled system, a swap gate can be directly implemented as a single operation. This may not be possible in other systems. For instance, in systems coupled through diagonal Ising-type interaction, a swap gate needs to be implemented as 3 successive CNOTs. As such, 6 gates are required to implement the QFT). Here, we will show how to combine these gates into a single unitary operation. To do this, we train the system parameters of a 2-qubit Hamiltonian (equation (4.4)), using the dynamic learning algorithm as a tool. The training set comprises of four input-output pairs. The inputs are the four computational basis states, $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. Output states are the expected state vectors, after applying the QFT matrix given by equation (7.5) to the corresponding input vectors. The training was carried out using the process explained in chapter 3 with the modification to the weight update rule as given by equations (7.2) and (7.3). (Step size of the ODE4 fixed size step solver was 0.1ns, and the total time, t_f is 10ns, which is the time step for the gate operation). Training was stopped when the RMS error was 0.0008. Table 7.1 shows the fidelity of the output state corresponding to each input for the trained network (the optimal fidelity in each case should be 1). Table 7.2 lists the parameters of the trained network.

TABLE 7.1

FIDELITY OF THE FINAL STATE OF THE TRAINED NETWORK UNDER A QFT ALGORITHM BETWEEN QUBITS Q_1 AND Q_2.

Input State, $\vert\psi\rangle_{in}$ ($\vert q_1q_2\rangle$)	**Desired Output State, $\vert\psi\rangle_{des}$ ($\vert q_1q_2\rangle$)**	**Fidelity, F (ρ_{des}, ρ_{out})**	**Error (1 – F)**
$\vert 00\rangle$	$(\vert 00\rangle+\vert 01\rangle+\vert 10\rangle+\vert 11\rangle)/2$	0.9988	0.0012
$\vert 01\rangle$	$(\vert 00\rangle+i\vert 01\rangle-\vert 10\rangle-i\vert 11\rangle)/2$	0.9992	0.0008
$\vert 10\rangle$	$(\vert 00\rangle-\vert 01\rangle+\vert 10\rangle-\vert 11\rangle)/2$	0.9998	0.0002
$\vert 11\rangle$	$(\vert 00\rangle-i\vert 01\rangle-\vert 10\rangle+i\vert 11\rangle)/2$	0.9992	0.0008
RMS error for the network = 0.0008			

TABLE 7.2

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A QFT ALGORITHM BETWEEN QUBITS Q_1 AND Q_2.

Δ_A	127.7 MHz
Δ_B	23.5 MHz
ε_A	23.5 MHz
ε_B	127.7 MHz
ξ	18.7 MHz

To verify if these parameters compute the QFT between Q_1 and Q_2, we substituted these parameters in the Hamiltonian, H, given by equation (4.4) and allowed the system to evolve for 10ns (t_f). The simulations confirmed the QFT for a 2-qubit system, which was realized up to an overall global phase of 112 degrees that can be ignored. Table 7.3 shows the desired and final output states for each of the four input states.

TABLE 7.3

COMPARISON OF THE DESIRED FINAL STATE AND THE SIMULATED FINAL STATE FOR A 2-QUBIT QFT.

Input State, $\vert\psi\rangle_{in}(\vert q_1q_2\rangle)$	**Desired Output State, $\vert\psi\rangle_{des}$ ($\vert q_1q_2\rangle$)**	**Actual Output State, $\vert\psi\rangle_{out}$ ($\vert q_1q_2\rangle$)**
$\vert 00\rangle$	$(\vert 00\rangle+\vert 01\rangle+\vert 10\rangle+\vert 11\rangle)/2$	$(0.5e^{1.23°i}\vert 00\rangle+0.52e^{-2.085°i}\vert 01\rangle+0.5e^{1.5°i}\vert 10\rangle+0.48e^{3.4°i}\vert 11\rangle)e^{112°i}$
$\vert 01\rangle$	$(\vert 00\rangle+i\vert 01\rangle-\vert 10\rangle-i\vert 11\rangle)/2$	$(0.52e^{-2.08°i}\vert 00\rangle+i0.48e^{0.5°i}\vert 01\rangle-0.49e^{1.6°i}\vert 10\rangle-i0.5e^{0.13°i}\vert 11\rangle)e^{112°i}$
$\vert 10\rangle$	$(\vert 00\rangle-\vert 01\rangle+\vert 10\rangle-\vert 11\rangle)/2$	$(0.5e^{1.44°i}\vert 00\rangle-0.49e^{1.3°i}\vert 01\rangle+0.5e^{0.43°i}\vert 10\rangle-0.51e^{-0.06°i}\vert 11\rangle)e^{112°i}$
$\vert 11\rangle$	$(\vert 00\rangle-i\vert 01\rangle-\vert 10\rangle+i\vert 11\rangle)/2$	$(0.48e^{3.43°i}\vert 00\rangle-i0.5e^{-0.14°i}\vert 01\rangle-0.51e^{-0.94°i}\vert 10\rangle+i0.51e^{1.81°i}\vert 11\rangle)e^{112°i}$

As examples, Figures 7.2 and 7.3 shows the evolution of the system under a 2-qubit QFT when the initial state is $|01\rangle$. Figure 7.2 shows the evolution of the probabilities of the four computational basis states, while Figure 7.3 shows the evolution of phases for each computational state. The relative phases for computational states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ are 109.45, −157.51, −66.74, and 22.83, respectively. After ignoring the overall global phase, the relative phases can be written as −2.55, −269.5 (or 90.5), −178.75 and −89.17 which are close to the expected relative phases for the input state $|01\rangle$. The overall fidelity of the gate operation was 99.88%.

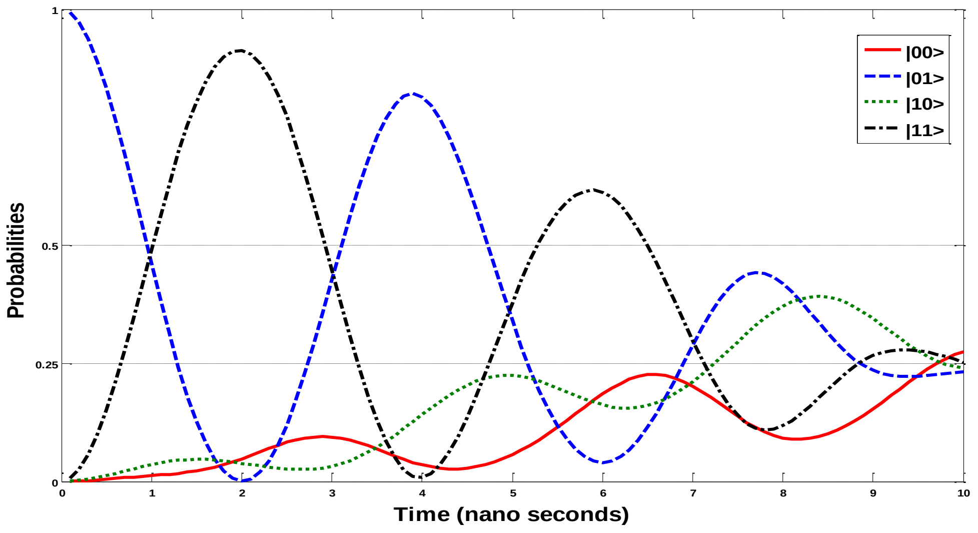

Figure 7.2. Two-qubit QFT – evolution of the probabilities of the four basis states of the two qubit system for the initial state |01⟩.

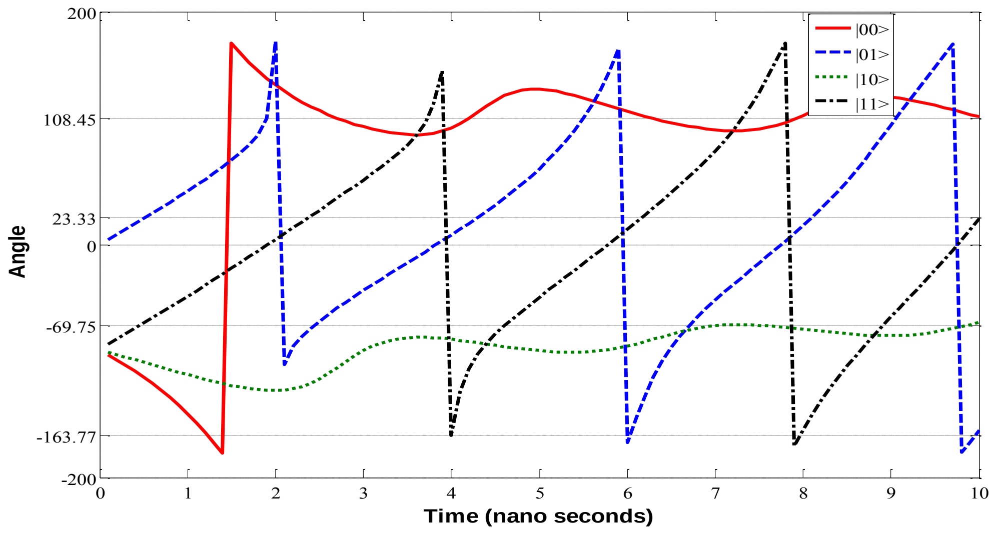

Figure 7.3. Two-qubit QFT – evolution of the phases of the four basis states of the two qubit system for the initial state |01⟩.

Since the circuit for Inverse Quantum Fourier Transform (IQFT) is simply the inverse of the circuit for the QFT (implementing the gates in the reverse order and replacing each gate by its adjoint), the IQFT can be implemented by simply using T=30ns. However, we trained the network to implement the IQFT in 10ns. Table 7.4 lists the parameters to implement IQFT. Here, the training was stopped when the RMS error value was 0.0017.

TABLE 7.4

PARAMETERS OF THE TRAINED NETWORK TO REALIZE THE IQFT ALGORITHM DIRECTLY BETWEEN QUBITS Q_1 AND Q_2.

Δ_A	283.5 MHz
Δ_B	59.9 MHz
ε_A	52.8 MHz
ε_B	282.4 MHz
ξ	55.8 MHz

In order to show that our scheme can be extended to LNN system with different types of interactions, we also trained our network to implement the 2-qubit QFT in Ising and XY interactions. Ising-coupled interactions are typical of super conducting qubits [13-15] and XY interactions are typical of quantum dot spins [123] and nuclear spins interacting via a two-dimensional electron gas [124]. Tables 7.5, and 7.6 lists the parameters of the trained network that implements QFT in a 2-qubit LNN system with Ising and XY interactions, respectively. (Here, the total time of the gate operation is 10ns in both cases, and the step size in the ODE4 was 0.01ns). The RMS errors are 0.0037 and 0.0032, respectively.

TABLE 7.5

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A QFT ALGORITHM DIRECTLY BETWEEN QUBITS Q_1 AND Q_2 IN A 2-QUBIT SYSTEM WITH ISING TYPE INTERACTIONS.

Δ_A	921.6 MHz
Δ_B	185.6 MHz
ε_A	47.6 MHz
ε_B	912.8 MHz
ξ	126 MHz

It is important to point that all the parameters solved for using our scheme are scalable relative to the time step, t_f. That is, suppose we need t_f to be 5ns, which is half the original value. The gate operation can be implemented by simply doubling all the parameter values. Thus, our parameters provide flexibility and can be adjusted to the requirements of a given experimental realization.

TABLE 7.6

PARAMETERS OF THE TRAINED NETWORK TO REALIZE A QFT ALGORITHM DIRECTLY BETWEEN QUBITS Q_1 AND Q_2 IN A 2-QUBIT SYSTEM WITH XY TYPE INTERACTIONS.

Δ_A	601.9 MHz
Δ_B	55.2 MHz
ε_A	68.4 MHz
ε_B	598.5 MHz
ξ	67.8 MHz

Encoded-Hadamard Gate for Three Qubit Error Correction Code

Quantum error correcting codes (QECCs) are used to protect quantum information against decoherence. The three qubit code is a simple error correction code to correct single qubit bit flip errors. In this code, an information qubit $\alpha|0\rangle+\beta|1\rangle$, is encoded using two ancilla qubits in the $|0\rangle$ state to create a logical qubit state $\alpha|000\rangle+\beta|111\rangle$. Here, the encoded states $|000\rangle$ and $|111\rangle$ are referred to as the 'logical 0'or $|0\rangle_L$, and 'logical 1' or $|1\rangle_L$ states, respectively. The encoding circuit to create these encoded states is shown in Figure 7.4.

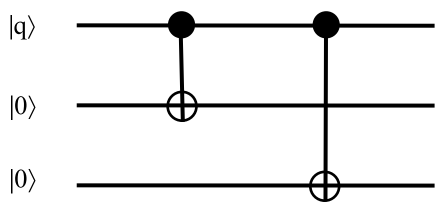

Figure 7.4. Encoder circuit for a three qubit quantum error correction code. Information qubit is encoded using 2 extra ancillas (qubits in |0⟩ state) to generate the logical qubit.

As mentioned earlier, in order to perform gate operations on logical qubits, a universal set of "encoded" gates is required. However, not all encoded schemes will allow these gates to be implemented without decoding the actual encoded qubit, first. For example, if we were to perform a Hadamard gate on the 3-qubit code without decoding (hence the name encoded-Hadamard gate or e-H gate), we would have to perform the following operation:

$$\alpha|000\rangle + \beta|111\rangle \xrightarrow{e-H} \alpha\left(\frac{|000\rangle+|111\rangle}{\sqrt{2}}\right) + \beta\left(\frac{|000\rangle-|111\rangle}{\sqrt{2}}\right) \quad (7.6)$$

To our knowledge, no method exists to implement the e-H gate without first having to decode the logical qubit. As such, we use the training algorithm to solve for system parameters of the Hamiltonian so that an e-H gate can be directly implemented on the encoded qubit.

For training, we consider a system of 3 qubits, A, B, and C. We assume the interactions between the qubits to be of the Ising type, which is typical for Josephson-junction qubits [39-41]. The Hamiltonian of the system is

$$H_3 = \sum_{i=A,B,C} (\Delta_i \sigma_{Xi} + \varepsilon_i \sigma_{Zi}) + \xi_{AB}\sigma_{ZA}\sigma_{ZB} + \xi_{BC}\sigma_{ZB}\sigma_{ZC} + \xi_{AC}\sigma_{ZA}\sigma_{ZC} \quad (7.7)$$

where, as before, Δ_i and ε_i for i = A, B, and C are the tunneling and bias parameter, respectively, and ξ_{AB} is the coupling between qubits A and B, ξ_{BC} is the coupling between qubits B and C, and ξ_{AC} is the coupling between the qubits A and C. Once again, the dynamic learning algorithm presented in Chapter 3 and section 7.1 was used to train the NN, where the system parameters were treated as weights. Here, the training set comprises of two input-output pairs. The inputs are the two computational basis states, $|000\rangle$ and $|111\rangle$. The outputs are the expected state vectors after applying the circuit shown in Figure 7.4 to the corresponding inputs (shown in Table 7.6). The step size in the ODE4 was 0.01ns, and total time, $t_f = 10$ns, which is the time step for the gate operation. Training was stopped when the RMS error was 0.0127. Table 7.7 shows the fidelity of the output state corresponding to each input for the trained network (the optimal fidelity in each case should be 1). Table 7.8 lists the parameters of the trained network.

TABLE 7.7

FIDELITY OF THE FINAL STATE OF THE TRAINED NETWORK UNDER AN ENCODED-HADAMARD GATE ON A 3-QUBIT ERROR CORRECTION CODE.

Input State, $|\psi\rangle_{in}$ ($|q_1q_2q_3\rangle$)	**Desired Output State, $|\psi\rangle_{des}$** ($|q_1q_2q_3\rangle$)	**Fidelity, F (ρ_{des}, ρ_{out})**	**Error** **(1 – F)**
$|000\rangle$	$(|000\rangle+|111\rangle)/\sqrt{2}$	0.9874	0.0124
$|111\rangle$	$(|000\rangle-|111\rangle)/\sqrt{2}$	0.9871	0.0129
RMS error for the network = 0.0127			

To check if these parameters compute an e-H gate on the 3-qubit QECC, we substituted these parameters in the Hamiltonian, H_3, given by equation (7.7) and allowed the system to evolve for 10ns (t_f). The simulations confirmed the e-H gate on the 3-qubit QECC for a 3-qubit system, which was realized up to an overall global phase of 96 degrees that can be ignored. Equations (7.8) and (7.9) give the final states that were obtained corresponding to the input states of $|000\rangle$ and $|111\rangle$, respectively.

$$|000\rangle \to 0.71e^{-96.76^\circ i}|000\rangle+0.1e^{-39.3^\circ i}|001\rangle+0.03e^{47.6^\circ i}|010\rangle+0.04e^{-132.4^\circ i}|011\rangle$$
$$+0.03e^{-52.2^\circ i}|100\rangle+0.06e^{-137^\circ i}|101\rangle+0.07e^{-101.6^\circ i}|110\rangle+0.69e^{-96^\circ i}|111\rangle$$

$$\approx (0.71|000\rangle + 0.69|111\rangle)e^{-96^\circ i} \tag{7.8}$$

$$|111\rangle \to 0.69e^{-96^\circ i}|000\rangle + 0.07e^{-116^\circ i}|001\rangle + 0.003e^{26.8^\circ i}|010\rangle + 0.03e^{-83^\circ i}|011\rangle$$
$$+0.03e^{123^\circ i}|100\rangle + 0.12e^{64^\circ i}|101\rangle + 0.076e^{73.1^\circ i}|110\rangle + 0.7e^{83.4^\circ i}|111\rangle$$

$$\approx (0.693|000\rangle - 0.703|111\rangle)e^{-96^\circ i} \tag{7.9}$$

Figures 7.5 and 7.6 shows the evolution of the system under the e-H gate on a 3-qubit QECC when the initial states are $|000\rangle$ and $|111\rangle$, respectively. The overall fidelity of the gate operation was 98.73%. From Figures 7.5 and 7.6, we can see that the initial states $|000\rangle$ and $|111\rangle$ evolve to the desired final states, which shows the implementation of the desired e-H gate operation.

TABLE 7.8

PARAMETERS OF THE TRAINED NETWORK TO REALIZE THE ENCODED-HADAMARD GATE ON A 3-QUBIT ERROR CORRECTION CODE.

Δ_A	578.5 MHz
Δ_B	172.2 MHz
Δ_C	710.4 MHz
ε_A	-183.2 MHz
ε_B	-84.6 MHz
ε_C	291.4 MHz
ξ_{AB}	1.1496 GHz
ξ_{BC}	1.3502 MHz
ξ_{AC}	1.1186 MHz

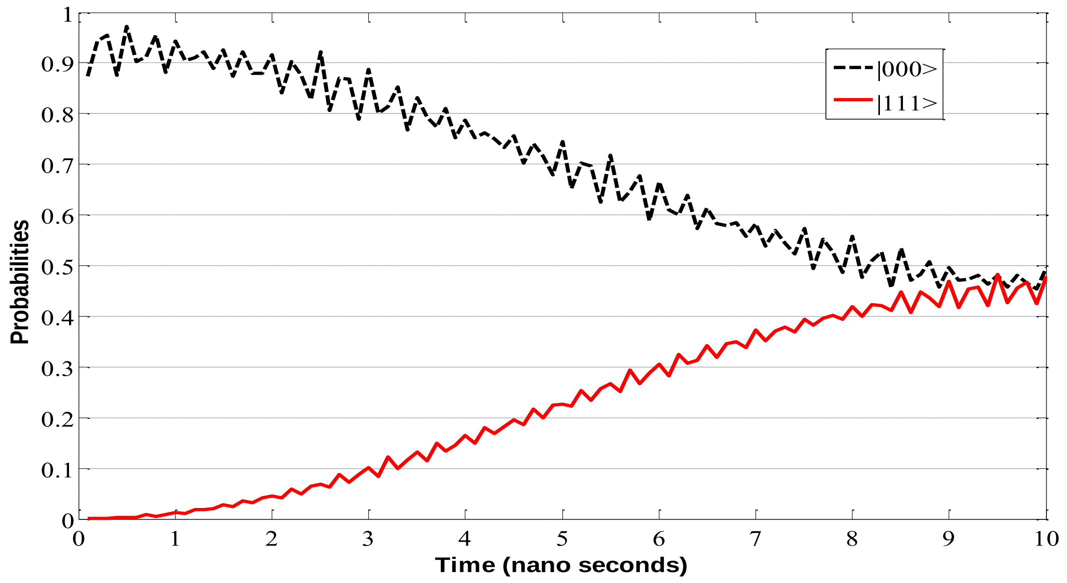

Figure 7.5. Three-Qubit Quantum Error Correction Code – evolution of the probabilities of the encoded states $|0\rangle_L$ and $|1\rangle_L$ of the three qubit system for the initial state, $|0\rangle_L = |000\rangle$.

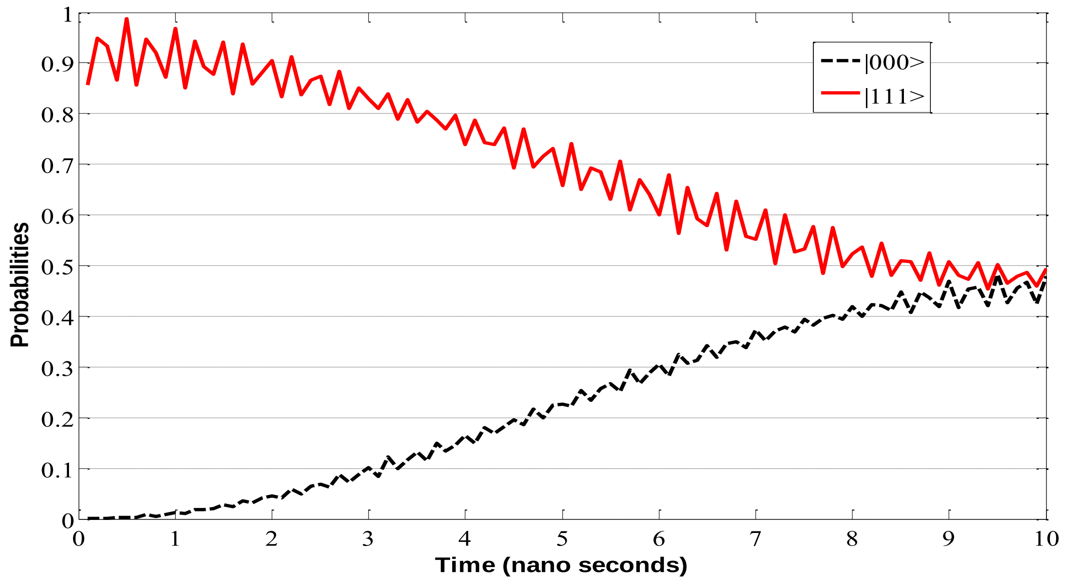

Figure 7.6. Three-Qubit Quantum Error Correction Code – evolution of the probabilities of the encoded states $|0\rangle_L$ and $|1\rangle_L$ of the three qubit system for the initial state, $|1\rangle_L = |111\rangle$.

7.3 Comparing the Performance of the Learning Algorithms with and without the Momentum Factor

In this section, we discuss the advantages of using the momentum factor. Using a momentum factor in the weight update rule improved the convergence rate of the training process. Also, in addition to speeding up the training process, in some cases, addition of the momentum factor helped avoid the training process from being trapped in a local minimum by allowed it to skip over those regions without performing any training there. To demonstrate this, we compared the performance of the learning algorithm with the momentum factor (used in this chapter) with the learning algorithm without the momentum factor (used in Chapter 3) for different cases.

As a test case, we chose the 2-qubit QFT with the Hamiltonian given by equation (4.4) for N = 2. First, we chose the dynamic learning algorithm without the momentum factor to train our system to find the parameters that implement the QFT operation. All the parameters were initially set to 50 MHz, and the total time of the gate operation was set to 10ns. The learning factor was set to 0.085, and we let the training to continue for 50 iterations. From Figure 7.7, we can see that the training process was trapped in the local minima, and the training was stopped after a few iterations. The final RMS error was 0.0714. Next, we used the dynamic learning algorithm with the momentum factor to train the system for 50 iterations with both the momentum and learning factors set to 0.085. In this case, the final RMS error was 0.008 (Figure 7.7), and can be further reduced by training the system for more iterations. Note that, while plotting the graphs, we left out the RMS error values of the first 9 iterations.

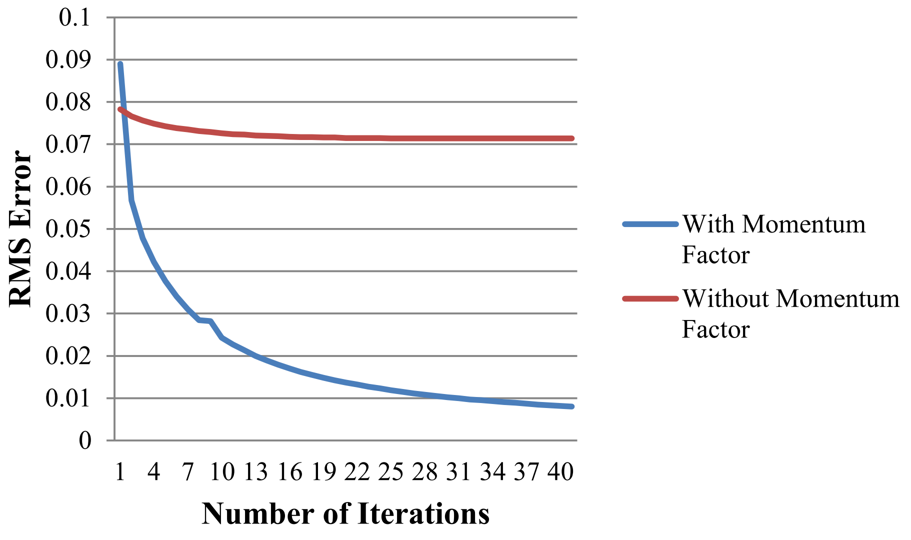

Figure 7.7. Convergence of the dynamic learning algorithm with and without momentum factor – RMS error versus number of iterations for the training of the 2-qubit Fourier Transform operation. Here, the learning factor was set to 0.085 in both the cases, and the momentum factor (when included in the training) was set to 0.085.

7.4 Conclusions

In this chapter, we have used the dynamic learning algorithm as a tool for implementing harder circuits which require training of phases in addition to probabilities (like the 2-qubit QFT and an encoded-Hadamard gate for the 3-qubit QECC). Moreover, we introduced a momentum factor in the learning algorithm to make it more efficient. To demonstrate the improvement of our learning algorithm after using the momentum factor, we use the 2-qubit QFT and the 3-qubit encoded-Hadamard gate as examples.

CHAPTER 8

CONCLUSIONS

In this work, we have proposed a new scheme for circuit reduction in quantum systems using a dynamic learning algorithm as a tool. We have shown that by using this method, we can find the system parameters to perform a series of gate operations *directly* by finding the equivalent gate operation. This is the main advantage of our scheme since it reduces the overall computational overhead as quantum circuits need not be broken down into a sequence of elementary gates. Secondly, since all qubits along the chain participate in the desired gate operation, we can avoid idle times that can give rise to relative phases, and also avoid propagation errors due to one/more faulty CNOT gates. Thirdly, our method does not require additional ancillas and pre-engineered mirror-periodic Hamiltonians for implementing any of these MI operations. Lastly, all parameters found using our scheme are scalable, and therefore, can be adjusted to the requirements of a given experimental realization. Initially, we showed how the training algorithm can be used as a tool for finding the parameters for implementing CNOT and Toffoli gates between next-to-nearest neighbor qubits in an Ising-coupled LNN system. We then showed how the scheme can be used to find parameters for realizing swap gates, first between two adjacent qubits, and then between two next-to-nearest-neighbor qubits, in each case without decomposing it into 3 CNOT gates. Finally, we showed how the scheme can be extended to systems with non-diagonal interactions. Then, we used our dynamic learning algorithm to implement MI gate operations in an LNN array by using the dynamic learning algorithm. Later, using the training results for MI presented in Chapter 5, we derived an analytical solution to implement MI gate operations in an LNN array with Ising type interactions. This analytical solution can be used to calculate parameter values to achieve MI gate operations in systems with an arbitrary of qubits without the need for training. Finally, we used our learning algorithm to implement the 2-qubit quantum Fourier Transform and the 3-qubit encoded-Hadamard gate, both of which are hard circuits to train as they require training of phases in addition to training of probabilities. We also used the 2-qubit Fourier Transform circuit to compare the convergence times of our learning algorithms with and without the momentum factor, and showed that the momentum factor will improve the performance of the learning algorithm (faster convergence to lower RMS errors).

REFERENCES

LIST OF REFERENCES

[1] Nielson, M.A., and Chuang, I.L., "Quantum Computation and Quantum Information," Cambridge University Press, Cambridge, England, 2001.

[2] Gershenfeld, N. A., and Chuang, I. L., "Bulk Spin-Resonance Quantum Computation," *Science*, Vol. 275, pp. 350, 1997.

[3] Cory, D., Fahmy,A., and Havel, T., "Ensemble Quantum Computing by NMR Spectroscopy," *Proceedingsof the national Academy of Sciences USA*, 94, pp. 1634, 1997.

[4] Kane, B. E., "A Silicon-Based Nuclear Spin Quantum Computer," *Nature*, Vol. 393, pp. 133, 1998.

[5] Ekert, A., and Jozsa, R., "Quantum Computing and Factoring Algorithm," *Reviews of modern Physics*, Vol. 68, pp. 773, 1996.

[6] Shedelbeck, G., Wegscheider, W., Bichler, M., and Abstreiter, G., "Coupled Quantum Dots Fabricated by Cleaved Edge Overgrowth: from Artificial Atoms to Molecules," *Science*, Vol. 278, pp. 1792, 1997.

[7] Oosterkamp, T. H., Fujisawa, T., van der Wiel, W. G., Ishibashi, K., Hijman, R. V., Tarucha, S., Kouwenhoven, H. V., "Microwave Spectroscopy of a Quantum-Dot Molecules," *Nature*, Vol. 395, pp. 873, 1998.

[8] Bonadeo, N. H., Erland, J., Gammon, D., Park, D., Katzer, D. S., and Steel, D. G., "Coherent Optical Control of the Quantum State of a Single Quantum Dot," *Science*, Vol. 282, pp. 1473, 1998.

[9] Loss. D., and DiVincenzo, D. P., "Quantum Computation with Quantum Dots," *Physical Review A,* Vol. 57, pp. 120, 1998.

[10] Cirac, J. J., and Zoller, P., "A Scalable Quantum Computer with Ions in an Array of Microtraps," *Nature*, Vol. 404, pp. 579, 2000.

[11] Gulde, S., Riebe, M., Lancaster, G. P. T., Becher, C., Eschner, J., Haffner, H., Schmidt-Kaler, F., Chuang, I. L., and Blatt, R., "Implementation of the Deutsch-Jozsa Algorithm on an Ion-Trap Quantum Computer," *Nature*, Vol. 421, pp. 48, 2003.

[12] Haroche, S., Brune, M., and Raimond, J. M., "Experiments with Single Atoms in a Cavity: Entanglement, Schrodinger's Cats and Decoherence," The Royal Society of *Philosophical Transactions London A*, Vol. 335, pp. 2367, 1997.

[13] Makhlin, Y., Schon, G., and Shnirman, A., "Quantum State Engineering with Josephson Junction Devices," *Reviews of Modern Physics*, Vol. 73, pp. 357, 2001.

[14] Makhlin, Y., Schon, G., and Shnirman, A., “Josephson-Junction Qubits with Controlled Couplings,” *Nature*, Vol. 398, pp. 305, 1999.

[15] Averin, D. V., “Josephson-Junction Qubits with Controlled Couplings,” *Solid State Communication*, Vol. 105, pp. 659, 1998

[16] Orlando, T. P., Lloyd, S., Levitov, L. S., Berggren, K. K., Feldman, M. J., Bocko, M. F., Mooij, J. E.,Harmans, C. J. P., and van der Wal, C. H., “Flux-Based Superconducting Qubits for Quantum Computation,” *Physica C: Superconductivity*, Vol. 372, pp. 194, 2001.

[17] Mooij, J. E., Orlando, T. P., Levitov, L. S., Tian, L., van der Wal, C. H., and Lloyd, S.,” Josephson Persistent-Current Qubit,” *Science*, Vol. 285, pp. 1036, 1999.

[18] Orlando, T. P., Mooij, J. E., Tian, L., van der Wal, C. H., Levitov, L. S., and Lloyd, S., ”Superconducting Persistent-Current Qubit,” *Physical Review B*, Vol. 60, pp. 15398, 1999.

[19] Chiarello, F., “Quantum computing with superconducting quantum interference devices: a possible strategy,” *Physics Letters A*, Vol. 277, pp. 189, 2000.

[20] Averin, D. V., Friedman, J. R., and Lukens, J. E., “Macroscopic Resonant Tunneling of Magnetic Flux,” *Physical Review B*, Vol. 62, pp. 11802, 2000.

[21] Averin, D. V., “Quantum Computing and Quantum Measurement with Mesoscopic Josephson Junctions,” *Fortscritte der Physik*, Vol. 48, pp. 1055, 2000.

[22] Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A. and Weinfurter, H. "Elementary Gates for Quantum Computation." *Physical Review A* 52, no. 5, 1995, 3457.

[23] Song, G., and Klappenecker, A., “Optimal Realizations of Controlled Unitary Gates,” *quantum Information and Computation,* Vol. 3, no. 2, pp. 139, 2003.

[24] Kumar, P., and Skinner, S. R., “Simplified Approach to Implementing Controlled-Unitary Operations in a Two-Qubit System,” *Phys. Rev A*, vol. 76, pp. 022335-022345, 2007.

[25] Vidal, G., and Dawson, C. M., “A Universal Quantum Circuit for Two-Qubit Transformations with Three CNOT Gates,” *Phys. Rev. A*, Vol. 69, 010301, 2004.

[26] Vatan, F., and C. Williams, C., “Optimal Quantum Circuits for General Two-Qubit Gates,” *Phys. Rev. A*, Vol. 69, 032315, 2004.

[27] Vartiainen, J. J., Mottonen, M., and Salomaa, M. M., “Efficient Decomposition of Quantum Gates,” *Phys. Rev. Let.*, Vol. 92, 177902, 2004.

[28] Bullock, S. S., and Markov, I. L., “Arbitrary Two-Qubit Computation in 23 Elementary Gates,” *Phys. Rev. A,* Vol. 68, 012318, 2003.

[29] Mottonen M, and Vartiainen, J. J., “Decompositions of general quantum gates. Ch. 7 in Trends in Quantum Computing Research," 2006.

[30] Deutsch, D., “Quantum Computational Networks,” *Proceedings of the Royal Society of London A*, 425:73–90, 1989.

[31] Yong-Sheng Zhang, Ming-Yong Ye, and Guang-Can Guo, “Conditions for Optimal Construction of Two-Qubit Nonlocal Gates,” *Phys. Rev. A*, Vol.71, 062331, 2005.

[32] Hollenberg, L.C.L., Dzurak, A.S., Wellard, C., Hamilton, A.R., Reilly, D.J., Milburn, G.J., and Clark, R.G., “Charge-Based Quantum Computing Using Single Donors in Semiconductors,” *Phys. Rev. B*, Vol. 69, 113301–113304, 2004.

[33] Pachos, J.K., and Knight, P.L., “Quantum Computation with a One-Dimensional Optical Lattice,” *Phys. Rev. Lett.,* Vol. 91, 107902–107905, 2003.

[34] Friesen, M., Rugheimer, P., Savage, D.E., Lagally, M.G., van der Weide, D.W., Joynt, R., and Eriksson, M.A., “Practical Design and Simulation of Silicon-Based Quantum-Dot Qubits,” *Phys. Rev. B*, Vol. 67, 121301–121304 (R), 2003.

[35] Ladd, T.D., Goldman, J.R., Yamaguchi, F., and Yamamoto, Y., “All-Silicon Quantum Computer,” *Phys. Rev. Lett*., Vol. 89, 017901–017904, 2002.

[36] Novais, E., and Castro Neto, A.H., “Nuclear Spin Qubits in a Pseudo spin Quantum Chain,” *Phys. Rev. A*, Vol. 69, 062312–062317, 2004.

[37] Tian, L., and Zoller, P., “Quantum Computing with Atomic Josephson-Junction Arrays,” *Phys. Rev. A*, Vol. 68, 042321–042330, 2003.

[38] van der Ploeg, S.H.W., Izmalkov, A., van den Brink, A.M., Hübner, U., Grajcar, M., Il’ichev, E., Meyer, H.-G., and Zagoskin, A.M., “Controllable Coupling of Superconducting Flux Qubits,” *Phys. Rev. Lett*., Vol. 98, 057004–057007, 2007.

[39] Lantz, J., Wallquist, M., Shumeiko, V.S., and Wendin, G., “Josephson Junction Qubit Network with Current-Controlled Interaction,” *Phys. Rev. B*, Vol. 70, 140507–140510 (R), 2004.

[40] Stock, R., and James, D.F.V., “Scalable, High-Speed Measurement-Based Quantum Computer using Trapped Ions,” *Phys. Rev. Lett*., Vol. 102, 170501–170504, 2009.

[41] Van Meter, R., Ladd, T.D., Fowler, A.G., and Yamamoto, Y., “Distributed Quantum Computation Architecture Using Semiconductor Nano photonics,” *Int. J. Quantum Inf.,* Vol. 8, 295–323, 2010.

[42] Shende, V.V. Bullock, S.S., and Markov, I.L., “Synthesis of Quantum-Logic Circuits,” *IEEE Trans. On CAD*, Vol. 25, No. 6, 1000–1010, 2006.

[43] Cheung, D., Maslov, D., and Severini, S., “Translation Techniques Between Quantum Circuit Architectures,” Workshop on Quantum Information Processing, December 2007.

[44] Chakrabarti, A., Sur-Kolay, S., “Nearest Neighbor Based Synthesis of Quantum Boolean Circuits,” *Engineering Letters*, Vol. 15, 356–361, 2007.

[45] Khan, M.H.A., “Cost Reduction in Nearest Neighbor Based Synthesis of Quantum Boolean Circuits,” *Engineering Letters*, Vol. 16, 1–5, 2008.

[46] Hirata, Y., Nakanishi, M., Yamashita, S., and Nakashima, Y., “An Efficient Method to Convert Arbitrary Quantum Circuits to Ones on a Linear Nearest Neighbor Architecture,” In *International Conference on Quantum, Nano and Micro Technologies*, 26–33, 2009.

[47] Chakrabarti, A., and Sur-Kolay, S., “Rules for Synthesizing Quantum Boolean Circuits Using Minimized Nearest-Neighbor Templates,” In *International Conference on Advanced Computing and Communications*, 183–189, 2007.

[48] Saeedi, M., Wille, R., and Drechsler, R., “Synthesis of Quantum Circuits for Linear Nearest Neighbor Architectures,” *Quantum Information Processing*, Vol. 10, 355–377, 2011.

[49] Kumar, P., “Efficient Quantum Computing between Remote Qubits in Linear Nearest Neighbor Architectures,” *Quantum information processing*, Vol. 12, no. 4, 1737-1757, 2013.

[50] Kumar, P., Skinner, S. R., Daraeizadeh, S.,”A Nearest Neighbor Architecture to Overcome Dephasing,” *Quantum Information Processing*, Vol. 10, 1570-0755, 2012.

[51] Duan, L. M.,Guo, G. C., “Preserving Coherence in Quantum Computation by Pairing Quantum Bits,” *Phys. Rev. Lett*., Vol. 79, 1953–1956, 1997.

[52] Zurek, W. H., Laflamme, R., “Quantum Logical Operations On Encoded Qubits,” *Phys. Rev. Lett*., Vol. 77, 4683, 1996.

[53] Shor, P. W., “Fault-Tolerant Quantum Computation,” *Proc. 37th Sympos. Foundations of Computer Science*, IEEE Computer Society Press, (1996).

[54] Bose, S., "Quantum Communication through an Unmodulated Spin Chain," *Phys. Rev. Lett.,* Vol. 91, pp. 207901-1–207901-4, 2003.

[55] Christandl, M., Datta, N., Dorlas, T. C., Ekert, A., Kay, A., and Landahl, A. J., "Perfect Transfer of Arbitrary States in Quantum Spin Networks," *Phys. Rev. A*, Vol. 71, pp. 032312-1–032312-11, 2005.

[56] Wang, Y., Shuang, F., and Rabitz, H., "All Possible Coupling Schemes in XY Spin Chains for Perfect State Transfer," *Phys. Rev. A*, vol. 84, pp. 012307-1–012307-4, 2011.

[57] Burgarth, D., Giovannetti, V., and Bose, S., "Efficient and perfect state transfer in quantum chains," *J. Phys. A: Math. Gen*., Vol. 38, 6793–6802, 2005.

[58] Burgarth, D., and Bose, S., "Perfect Quantum State Transfer with Randomly Coupled Quantum Chains," *New J. Phys*., Vol. 7, 135-1–135-14, 2005.

[59] Karbach, P., and Stolze, J., "Spin Chains as Perfect Quantum State Mirrors," *Phys. Rev. A*, Vol. 72, pp. 030301-1–030301-4, 2005.

[60] Albanese, C., Christandl, M., Datta, N., and Ekert, A., "Mirror Inversion of Quantum States in Linear Registers," *Phys. Rev. Lett*., Vol. 93, 230502-1–0230502-4, 2004.

[61] Rao, K. R. K., Mahesh, T. S., and Kumar, A., "Efficient Simulation of Unitary Operations by Combining Two Numerical Algorithms: An NMR Simulations of the Mirror-Inversion Propagator of an XY Spin Chain", *Phys. Rev. A*, Vol. 90, pp. 012306-1–030303-9, 2014.

[62] Clark, S. R., Alves, C. M., and Jaksch, D., "Efficient Generation of Graph States for Quantum Computation," *New J. Phys.,* Vol. 7, 124**,** 2005.

[63] Yung, M. H., "Quantum Speed Limit for Perfect State Transfer in One Dimension," *Phys. Rev. A*, Vol. 74, pp. 030303-1–030303-4, 2006.

[64] Nguyen, B. H., and Nguyen, V. H., "Engineered Spin Chains for Perfect Quantum State Transfer," *Adv. Nat. Sci.: Nanosci. Nanotechnol*, Vol. 1, pp. 025003-1–025003-9, 2010.

[65] Fitzsimons, J., Xiao, L., Benjamin, S. C., and Jones, J. A., "Quantum Information Processing with Delocalized Qubits under Global Control," *Phys. Rev. Lett.,* Vol. 99, pp. 030501-1–030501-4, 2007.

[66] Fitzsimons, J., and Twamley, J., "Globally Controlled Quantum Wires for Perfect Qubit Transport, Mirroring, and Computing," *Phys. Rev. Lett*., Vol. 97, 090502-1–090502-4, 2006.

[67] Raussendorf, R., “Quantum Computation via Translation-Invariant Operations on a Cchain of Qubits,” *Phys. Rev. A*, Vol. 72, pp. 052301-052306, 2005.

[68] Bose, S., “Quantum Communication through an Unmodulated Spin Chain,” *Physical Review Letters*, Vol. 91, pp. 207901, 2003.

[69] Tesch, Carmen M., Lukas Kurtz, and Regina de Vivie-Riedle., "Applying Optimal Control Theory for Elements of Quantum Computation in Molecular Systems." *Chemical Physics Letters,* Vol. 343, no. 5, pp. 633-641, 2001.

[70] Tesch, Carmen M., and Regina de Vivie-Riedle., "Quantum Computation with Vibrationally Excited Molecules," *Physical review letters*, Vol. 89, no. 15, 157901, 2002.

[71] Palao, J. P., and Koslo, R., “Molecular Quantum Computing by an Optimal Control Algorithm for Unitary Transformations” *Phys. Rev. Lett*. Vol. 89, 188301, 2002.

[72] Bisio, A., Chiribella, G., D’Ariano, G. M., Facchini, S., and Perinotti, P., “Optimal Quantum Learning of a Unitary Transformation,” *Physical Review A*, Vol. 81, no. 3, 032324, 2010.

[73] Grace, M., Brif, C., Rabitz, H., Walmsley, I. A., Kosut, R. L., and Lidar, D. A., “Optimal Control of Quantum Gates and Suppression of Decoherence in a System of Interacting Two-Level Particles,” *J. Phys. B: At. Mol. Opt. Phys*., Vol. 40, 103, 2007.

[74] Borzi, A., Stadler, G., and Hohenester, U., “Optimal quantum control in nanostructures: theory and application to a generic three-level system,” *Phys. Rev. A*, Vol. 66, 053811, 2002.

[75] Hohenester, U., Rekdal, P. K., Borzi, A., and Schmied-mayer, J., “Optimal Quantum Control of Bose-Einstein Condensates in Magnetic Microtraps,” *Phys. Rev. A*, Vol. 75, 023602, 2007.

[76] Jirari, H., and Potz, W., “Quantum Optimal Control Theory and Dynamic Coupling in the Spin-Boson Model,” *Phys. Rev. A*, Vol. 74, no. 2, 022306, 2006.

[77] Jirari, H., and Potz, W., “Optimal Coherent Control of Dissipative N-Level Systems,” *Phys. Rev. A*, Vol. 72, 013409, 2005.

[78] Tesch, C. M., and deVivie Riedle, R., “Quantum Computation with Vibrationally Excited Molecules,” *Phys. Rev. Lett*. Vol. 89, 157901, 2002.

[79] Rolo, R., and Potz, W., “Bell-State Preparation for Electron Spins in a Semiconductor Double Quantum Dot,” *Phys. Rev. B*, Vol. 76, 075333, 2007.

[80] Wenin, M., and Potz, W., "Optimal Control of a Single Qubit by Direct Inversion," *Phys. Rev. A*, Vol. 74, 022319, 2006.

[81] Montangero, S., Calarco, T., and Fazio, R., "Robust Optimal Quantum Gates for Josephson Charge Qubits," *Phys. Rev. Lett.*, Vol. 99, 170501, 2007.

[82] Wenin, M., and Potz, W., "Minimization of Environment-Induced Decoherence in Quantum Subsystems and Application to Solid-State-Based Quantum Gates," *Phys. Rev. B*, Vol. 78, 165118, 2008.

[83] Rolo, R., and Potz, W., "Time-Optimal Performance of Josephson Charge Qubits: A Process Tomography Approach," *Phys. Rev. B*, Vol. 79, 224516, 2009.

[84] Schulte-Herbruggen, T., Sporl, A., Khaneja, N., & Glaser, S. J., "Optimal Control for Generating Quantum Gates in Open Dissipative Systems," *Journal of Physics B: Atomic, Molecular and Optical Physics,* Vol. 44, no. 15, 154013, 2011.

[85] Kak, S. C., "On Quantum Neural Computing," *Advances in Imaging and Electron Physics*, Vol. 94, 259, 1995.

[86] Perus, M., "Neuro-Quantum Parallelism in Brain-Mind and Computers", *Informatica*, Vol. 20, pp. 173-183, 1996.

[87] Chrisley, R.L., "Learning in Non-Superpositional Quantum Neurocomputers," *Brain, Mind and Physics. IOS Press, Amsterdam*, pp. 126-139, 1997.

[88] Menneer, T., and Narayanan, A., "Quantum-inspired neural networks," Technical report R329, Department of Computer Science, University of Exeter, UK, 1995.

[89] Ventura, D., "Implementing Competitive Learning in a Quantum System", *Proceedings of the International Joint Conference on Neural Networks*, paper #513, 1999.

[90] Ventura, D., "Learning Quantum Operators", *Proceedings of the International Conference on Computational Intelligence and Neuroscience,* pp. 750-752, 2000.

[91] Narayanan, A., and Meneer, T., "Quantum Artificial Neural Network Architectures and Components", *Information Sciences*, Vol. 128, pp. 231-255, 2000.

[92] Bshouty, N. H., and Jackson, J., "Learning DNF over the Uniform Distribution Using a Quantum Example Oracle", *Proceedings of the Annual Conference on Computational Learning Theory, ACM Press*, pp. 118-127, 1995.

[93] Zak, M., "Quantum Decision-Maker", *Information Sciences*, Vol. 128, pp. 199-215, 2000.

[94] Ventura D., and Martinez, T., "Quantum Associative Memory", *Information Sciences*, Vol. 124, pp. 273-296, 2000.

[95] Ezhov, A., Nifanova, A., and Ventura, D., "Distributed Queries for Quantum Associative Memory", *Information Sciences*, Vol. 128, pp. 271-293, 2000.

[96] Howell, J., Yeazell. J., and Ventura, D., "Optically Simulating a Quantum Associative Memory", *Physical Review A*, Vol. 62, article W2303, 2000.

[97] Ventura, D. A., "On the Utility of Entanglement in Quantum Neural Computing." 2001.

[98] Bang, J., Ryu, J., Yoo, S., Pawłowski, M., and Lee, J., "A Strategy for Quantum Algorithm Design Assisted by Machine Learning," *New Journal of Physics*, Vol. 16, no. 7, 073017, 2014.

[99] Behrman, E. C., Niemel, J., Steck, J. E., and Skinner, S. R., "A Quantum Dot Neural Network." *Proceedings of the 4th Workshop on Physics of Computation*, pp. 22-24, 1996.

[100] Behrman, E.C., Nash, L.R., Steck, J. E., Chandrashekar, V. G., and Skinner, S. R., "Simulations of Quantum Neural Networks", *Information Sciences*, Vol. 128, pp. 257-269, 2000.

[101] Behrman, E. C., Steck, J. E., and Skinner, S. R., "A Spatial Quantum Neural Computer," *Proceedings of the 1999 International Joint Conference on Neural Networks (IJCNN'99)*, Washington, DC, July 10-16, published on CD, ISBN 0-7803-5532-6, #2113, 1999.

[102] Behrman, E. C., Chandrashekar, V., Wang, Z., Belur, C. K., Steck, J. E., and Skinner, S. R., "A quantum neural network computes entanglement." *APS Meeting Abstracts*. Vol. 1, 2003.

[103] Behrman, E. C., Steck, J. E., Kumar, P., and Walsh, K. A., "Quantum algorithm design using dynamic learning." *Quantum Information & Computation* 8.1, 12-29, 2008.

[104] Behrman, E. C., Steck, J. E., Gagnebin, P. K., and Skinner, S. R., "Towards the Dynamic Learning of an Experimental Entanglement Witness," *Evolutionary Computation, CEC 2006. IEEE Congress on*. IEEE, pp. 2613-2621, 2006.

[105] Khaneja, N., Reiss, T., ehlet, C., Schulte-Herbruggen, T. and Glaser, S., J., "Optimal Control of Coupled Spin Dynamics: Design of NMR Pulse Sequences by Gradient Ascent Algorithms," *J. Magn. Reson*, Vol. 172, pp. 296–305, 2005.

[106] Rowland, B., Jones, J. A., "Implementing Quantum Logic Gates with Gradient Ascent Pulse Engineering: Principles and Practicalities," *Phil. Trans. R. Soc*. A, Vol. 370, pp. 4636–4650, 2012.

[107] Tsai, D. B., and Goan, H.S., "Gradient Ascent Pulse Engineering Approach to CNOT Gates in Donor Electron Spin Quantum Computing," *AIP Conf. Proc., Solid-State Quantum Computing,* Vol. 1074, pp. 50–54, 2008.

[108] Jozsa, R., "Fidelity for Mixed Quantum States," *Journal of Modern Optics,* Vol. 41, no. 12, 2315–2323, 1994.

[109] Kumar, P., Skinner, S.R., Behrman, E.C., Steck, J.E., Zhou, Z., Han, S, "Quantum Gates Using a Pulsed Bias Scheme," *Phys. Rev. A*, Vol. 72, 042311-042319, 2005.

[110] Kumar, P., Skinner, S.R., "Universal Quantum Computing in Linear Nearest Neighbor Architectures," *Quantum Information and Computation,* Vol. 11, 0300–0312, 2011.

[111] Zhou, X., Zhou, Z., Guo, G., and Feldman, M.J., "Quantum Computation with Untunable Couplings," *Phys. Rev. Lett*., Vol. 89, pp. 197903, 2002.

[112] Schuch, N., Siewert, J., "Natural Two-Qubit Gate for Quantum Computation Using the XY Interaction," *Phys. Rev. A*, Vol. 67, pp. 032301−032308, 2003.

[113] Shor, P., "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. Sci. Statist. Comput*., Vol. 26, 1484, 1997.

[114] Dobsicek, M., Johansson, G., Shumeiko, V., and Wendin, G., "Arbitrary Accuracy Iterative Quantum Phase Estimation Algorithm Using a Single Ancillary Qubit: A Two-Qubit Benchmark," *Phys. Rev. A*, Vol. 76, 030306(R), 2007.

[115] Weinstein, Y. S., Pravia, M. A., Fortunato, E. M., Lloyd, S., and Cory, D. G., "Implementation of the Quantum Fourier Transform. *Phys. Rev. Lett*., Vol. 86, 1889, 2001.

[116] Garigipati, R. C., and P. Kumar, P., "Implementing Gate Operations between Uncoupled Qubits in Linear Nearest Neighbor Arrays using a Learning Algorithm," *Quantum Inf. Processing*, Vol. 12, 2291–2308, (2013).

[117] Garigipati, R. C., and Kumar, P., "Mirror Inverse operations in Linear Nearest Neighbors using Dynamic Learning Algorithm," IEEE Trans. Neural Networks and Learning Systems, accepted (2015).

[118] Garigipati, R. C., and P. Kumar, P., "Finding Analytical Solution for Mirror Inverse Gate Operations in Quantum Systems with Diagonal Interactions," *SPIE Sensing Technology+Applications. International Society for Optics and Photonics*, accepted (2015).

[119] Fausett, L., "Fundamental of Neural Networks," 1st Edition, Prentice Hall, 1994.

[120] Rojas R., “Neural Networks: A Systematic Introduction,” Springer, Berlin, 1996.

[121] Riedmiller, M., and Braun, H., “A Direct Adaptive Method for Faster Back Propagation Learning: the RPROP Algorithm,” In H. Ruspini, editor, *Proceedings of the International Conference on Neural Networks*, pp 586-591, San Francisco, CA, Mar. 1993.

[122] Leonard, J., and Kramer, M. A., “Improvement of the Back Propagation Algorithm for Training Neural Networks,” *Computers in Chemical Engineering,* Vol. 14, 337-341., 1990.

[123] Imamoglu, A., Awschalom, D. D., Burkard, G., Divincenzo, D.P., Loss, D., Sherwin, M., and Small, A., “Quantum Information Processing Using Quantum Dot Spins and Cavity QED,” *Phys. Rev. Lett.* Vol.83, 4202, 1999.

[124] D. Mozyrsky, V. Privman, and M. Glasser, “Indirect Interaction of Solid State Qubits via Two-Dimensional Electron Gas,” *Phys. Rev. Lett.* Vol. 86, 5112, 2001.