2012

Quantum Codes and Computation

Sankalpa Khadka
Bard College

Quantum Codes and Computation

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Sankalpa Dhoj Khadka

Annandale-on-Hudson, New York
May, 2012

Abstract

In 1982, Richard Feynman first suggested that in order to simulate quantum mechanical system one needs quantum computer. In classical computer, data are store in the form of binary digits called bits. In quantum computer, information is stored in the form of quantum bit or qubit which lives in two dimensional complex vector space. Similarly, a classical computer uses logical gates to manipulate bits. Quantum analog of classical gates are unitary transformations which manipulate qubits.

The project presents quantum codes and computation process for error correction and quantum algorithms. The project focuses especially on Deutsch's algorithm which was proposed by David Deutsch in 1985 as an example of quantum algorithm that is significantly faster than classical algorithm for a system involving single qubit. Deutsch's Problem determines whether a function is constant or not for a single qubit system. The project extends Deutsch's algorithm for multiple qubits and explores properties of various unitary transformations and information they encode.

Contents

List of Figures

Dedication

To my grandmothers, Ganga Devi Khadka and Ram Maya Shrestha.

I have tried to emulate hard work and perseverance you have shown all your lives.

Acknowledgments

I would like to thank my research advisors, Professor Matthew Deady and Professor Greg Landweber, for providing excellent guidance for this project. I am very thankful for their tremendous support and enthusiasm in my research. I would also like to thank Professor Deady for his great advices over the years and Professor Landweber for inspiring me to become a Physics and Mathematics major.

The completion of this project marks an end of a journey which started four years ago. I would like to thank everyone who has enriched these four years of my life. I would like to thank Professor Christian Bracher for invigorating my interest in Physics through his lively lectures. I would like to thank Professor Sven Anderson for his important contribution towards my Computer Science education. I would also like to thank Dr. Abhay Deshpande and Dr. Robert Lourie for the opportunity of participation in cutting edge research at Brookehaven National Laboratory in the summer of 2010.

I would like to express my deepest gratitude to my Mom and Dad for all the sacrifices they have made for my education. I would like to thank my two wonderful sisters, Bhabana and Namuna, for their love and encouragement. I would also like to thank my aunt Vijaya for her support in the last four years. I would like to thank all my friends for a great learning experience at Bard.

1
Introduction

In 1965 Gordon Moore made the prediction that the number of transistors in integrated circuit doubles every two years [7]. This prediction, known as Moore's law, has hold for more than 40 years. According to Moore's law as time increases the number of atoms to implement bits decreases. If Moore's law were to hold then in future the number of atoms per bit will reach a level where quantum mechanics rather than classical mechanics will be needed to explain computational processes. In 1982, Richard Feynman suggested that a classical computer is incapable of efficiently simulating a quantum mechanical system and proposed the need of quantum computers based on principles of quantum mechanics [8]. A classical computer uses bits, {0,1}, to represent data. In quntum computer, information is stored in the form of quantum bit or qubit which lives in two dimensional complex vector space. Similarly, a classical computer uses logical gates to manipulate bits. Quantum analog of classical gates are unitary transformations which manipulate qubits.

In 1985, David Deutsch laid out the first quantum algorithm that outperforms classical algorithms [9]. Deutsch's Problem uses quantum computation to determine if a function is constant or not for a single qubit input system. This project extends Deutsch's problem for the 2 qubit input system and the 3 qubit input system. The project concludes with a generalized solution to the problem of discriminating functions which have all but one output values equal to each other for the n qubit input system.

The project begins with background information on linear algebra in Chapter 2. Here I introduce the definitions of vector space, tensor product and unitary transformation. In Chapter 3, I present essential tools for understading quantum computations such qubits, quantum gates and quantum entanglement. Chapters 2 and Chapter 3 build the conceptual framework for understanding the remaining chapters of the project.

In Chapter 4, I briefly present Quantum Error Correction. I have included this expository chapter for two reasons. First of all, during the research period, I spent a good amount of time learning about quantum error correction. Secondly, Quantum Error Correction is an important component in quantum computation which needs discussion in any quantum computation related works. I present groundbreaking works in quantum error correction such as Stablizer formalism developed by Daniel Gottesman [6].

In Chapter 5, I introduce quantum algorithms. The research has mostly focused on quantum algorithms, and more specifically on Deutsch's algorithm. Deutsch's algorithm predicts if a function is constant or not in fewer number of steps than classical algorithm for the 1 qubit input system. After learning the original Deutsch's problem, the first reseach question that I got interested in was how Deutsch's Algorithm behaved for the 2 qubit input system. For the 1 qubit system, there are two kinds of functions, one which have both output equal to each other and other which has two different outputs. The single qubit is simple since the standard Hadamard gate can discriminate both kinds of function. However, in the two qubit input problem, the Hadamard gate discriminates functions which have two output value equal to each other. The other kind of functions which have three out of four output value equal to each other are not distinguished by the Hadamard gate. Finding a gate which can discriminate these functions was my first major challenge in the research. After making observations of the gates and functions, I was able to find relationship between entries of the gate and outputs of the functions. This information helped me find a new gate called the GregMatt[1] gate, which discriminates the functions which have all but one output equal to each other.

After expanding Duestch's Problem for the 2 qubit system and completely solving it, my natural instinct was to expand the problem for the 3 qubit input system. In the 3 qubit input too, the Hadamard gate can discriminate the funtions that have half of the output values equal to each other. One of the important result that I discovered here was that the functions which have half of the output values equal to each other and which also form a group structure under mod 2 addition can be discriminated by a Hadamard-like gate whose entries has a direct relation with the value of the functions. However, for the 3 qubit input system, there are other kinds of functions such as the ones which have all but one output values equal, ones which have two outputs equal to each other and remaining six equal to each other, and ones which have three outputs equal to each other and remaining five equal to each other. The functions which have all but one value equal is simple enough to tackle because there are only 8 such functions for the 3 qubit system. Therefore, I set out to find a gate that can discriminate the functions which have all but one value equal to each other. This challenge seemed bigger than what I first thought. However, I found a method to discriminate these function by using the GregMatt gate with slight modification in the algorithm. This finding is a great trumph of the research because I could expand the same method for a general n qubit input problem to discriminate the functions which have all but one value equal to each other.

[1] The gate is named after my advisors Greg Landweber and Matthew Deady.

2

Background

2.1 Linear Algebra

The following standard definitions from Linear Algebra are taken from [3].

Definition 2.1.1. Let F be a field, whose elements are referred to as scalars. A **vector space** over F is a nonempty set V, whose elements are referred to as vectors together with addition and scalar multiplication. Addition assigns to each pair (u, v) of vectors in V a vector $u+v$ in V and scalar multiplication assigns to each pair $(r, u) \in F \times V$ a vector ru in V. In addition, elements of vector space forms abelian group under addition. △

Definition 2.1.2. Let V be a vector space over $F = \mathbb{R}$ or $F = \mathbb{C}$. An **inner product** on V is a function $\langle , \rangle \colon V \times V \longrightarrow F$ with the following properties:

1. **(Positive definiteness)** For all $v \in V$, the inner product $\langle v, v\rangle$ is real and

$$\langle v, v\rangle \geq 0 \quad \text{and} \quad \langle v, v\rangle = 0 \Leftrightarrow 0$$

2. For $F = \mathbb{C}$: **(Conjugate symmetry)**

$$\langle u, v\rangle = \overline{\langle v, u\rangle}$$

For $F = \mathbb{R}$: **(Symmetry)**

$$\langle u, v\rangle = \langle v, u\rangle$$

3. **(Linearity in the first coordinate)** For all $u, v \in V$ and $r, s \in F$

$$\langle ru + sv, w\rangle = r\langle u, w\rangle + s\langle v, w\rangle$$

A real (or complex) vector space V, together with an inner product, is called a **real**(or **complex**) **inner product space**. △

Definition 2.1.3. A inner product space that is complete under the metric induced by the inner product is said to be a **Hilbert space**. △

Definition 2.1.4. Let V and W be a vector spaces over F and let $\{v_1, v_2, ...v_n\}$ be basis for V and $\{w_1, w_2, ...w_m\}$ be basis for W, then the **tensor product** of V and W, denoted by $V \otimes W$, is the vector space spanned by $\{v_i \otimes w_j : 1 \leq i \leq n, 1 \leq j \leq m\}$. △

Let $A = (a_{i,j})$ be a matrix with respect to the ordered basis $\mathcal{A} = (u_1, ..., u_n)$ and $B = (b_{i,j})$ be a matrix with respect to the ordered basis $\mathcal{B} = (v_1, ..., v_m)$. Consider the ordered basis $\mathcal{C} = (u_i \otimes v_j)$ ordered by lexicographic order, that is $u_i \otimes v_j \leq u_l \otimes v_k$ if if $i < l$ or $i = l$ and $j < k$. The matrix of $A \otimes B$ with respect to $\mathcal{C}$ is :

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & ... & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & ... & a_{2,n}B \\ . & . & ... & . \\ a_{n,1}B & a_{n,2}B & ... & a_{n,n}B \end{bmatrix}.$$

This matrix is called the tensor product of the matrix A with the matrix B.

Definition 2.1.5. A **unitary transformation** is an isomorphism between two Hilbert spaces such that it preserves the inner product. Let H_1 and H_2 be Hilbert spaces, then a unitary transformation is a bijective function $U : H_1 \longrightarrow H_2$ such that

$$\langle Ux, Uy\rangle = \langle x, y\rangle.$$

△

3 Quantum Computing

3.1 Classical Computing: Bits and Logical Operations

In classical computing, data are stored in the form of binary digits or bits. *Bit* is the basic unit of information stored in a computer which in one of the two possible distinct states. For example: two distinct voltages, on and off state of electric switch, two direction of magnetization etc. The two possible values/states of a system are represented as binary digits, 0 and 1.

A Logical operation is an instruction that takes input bit/s and return an output bit under certain rule. Logical operations, also known as logical gates, are the basis of computation in classical computers. Computers are built with circuit that is made up of logical gates. The examples of logical gates are AND, OR, NOT, NOR, XOR etc.

AND		
	0	**1**
0	0	0
1	0	1

OR		
	0	**1**
0	0	1
1	1	1

XOR		
	0	**1**
0	0	1
1	1	0

NOR		
	0	**1**
0	1	0
1	0	0

3.2 Qubits

In a quantum computer, data are stored in the form of qubits or quantum bits. A qubit is the unit of information in quatum computation. A quantum system with n-qubits has a Hilbert space of 2^n dimensions, and therefore has 2^n mutually orthogonal quantum states which can be written as $\{|i\rangle\}$ where i is an n bit binary number.

For example: A 3 qubit system has 8 orthogonal states represented as $\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$.

A classical bit has only two states, either 0 or 1. Similarly a qubit has states $|0\rangle$ and $|1\rangle$, or any linear combination of states also known as a superposition. Hence, a qubit can have an infinitely many states,

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{where } \alpha \text{ and } \beta \text{ are complex numbers.}$$

A single qubit lives in vector space over complex number, $\mathbb{C}^2$. An n-qubit lives in $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 = \mathbb{C}^{2n}$.

3.3 Quantum Gates

A quantum gate is an operation which is a unitary transformation on qubits. The quantum gates are represented by matrices, and a gate acts on n qubits is represented by $2^n \times 2^n$ unitary matrix. Analogous to the classical computer which is built from an electrical circuit containing wires and logic gates, quantum computers are built from quantum circuits containing “wires”(mostly photons) and quantum gates to carry out the computation.

Pauli Operators

The Pauli operators are the special single qubit gates which are represented by the Pauli matrices $\{I, X, Y, Z\}$ as follows

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

For example, the application of X causes bit-flip in following ways:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

$$X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle.$$

Hadamard Gate

The Hadamard gate is defined by the matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The Hadamard gate maps the computational basis states into superpostion of states, which are anti-symmetric. The Hadamard gate is significant since it produces maximally entangled states from basis states in the following ways:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Controlled-U Gates

A controlled-U gate is the quantum gate in which the U operator acts on the n^{th} qubit n qubit only if the value of the preceeding qubit is 1.

For example: In a Controlled-NOT gate, the NOT operator flips the second qubit if the first qubit is 1.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$CNOT|00\rangle = |00\rangle$$

$$CNOT|01\rangle = |01\rangle$$

$$CNOT|10\rangle = |11\rangle$$

$$CNOT|11\rangle = |10\rangle.$$

3.4 Quantum Entanglement

Entanglement is a purely quantum mechanical property not exhibited by classical system. Entanglement occurs when subsystems interact in such a way that the resulting state of the whole system cannot be expressed as the direct product of states for its parts. When a quantum system is in such an entangled state, actions performed on one sub-system will have a side effect on another sub-system even though that sub-system is not acted upon directly.

If a pure state, $|\Psi^{(AB)}\rangle$, of a composite quantum system defined on a Hilbert Space $H_A \otimes H_B$ can be written as $|\Psi^{(AB)}\rangle = |\Psi^A\rangle \otimes |\Psi^B\rangle$, then $|\Psi^{(AB)}\rangle$ is said to be *separable state.*

A state, $|\Psi^{(AB)}\rangle$, of a composite quantum system defined on a Hilbert Space $H_A \otimes H_B$ is not a separable state, it is called an *entangled state.*

For example: Consider the state $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$. If this state were separable, it could be written in the form of $|\Psi^A\rangle \otimes |\Psi^B\rangle$ where $|\Psi^A\rangle = a_0|0\rangle + a_1|1\rangle$ $|\Psi^A\rangle = b_0|0\rangle + b_1|1\rangle$

$$|\Psi^A\rangle \otimes |\Psi^B\rangle = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

which implies, $a_0b_0 = 0, a_0b_1 = \frac{1}{\sqrt{2}}, a_1b_0 = -\frac{1}{\sqrt{2}}, a_1b_1 = 0$. However, this solution is not possible. Hence, the state $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$ is an entangled state.

4 Quantum Error Correction

4.1 Introduction

Quantum computers have extraordinary potential compared to its classical counterpart because some quantum algorithms are incredibly faster than classical algorithms. However, one of the main challenges for building reliable quantum computers would be to protect quantum information from errors. A quantum computer will interact with the environment causing *decoherence* and hence loss of quantum information stored in the system. One of the primary goals of quantum error correction is to prevent interaction with the environment. Furthermore, quantum gates are unitary transformations, which cannot be implemented perfectly. The effects of such imperfections in the gates can build up resulting in a failure in the computation. Therefore, quantum error correction aims at preventing the error in the quantum information by protecting it against error in the quantum circuit as well as from the surrounding environment.

The main idea in error correction is that in order to protect a message, it is first encoded by addition of some redundant information so that if an error occurs in the encoded message, there will be sufficient information left in the encoded message that will enable recovery of the original message by decoding the error afflicted message. In quantum computation, error is prevented in the quantum information with the help of quantum error correcting codes.

4.2 Theory of Quantum Error-Correction

A quantum error-correcting code is a mapping of k qubits (a Hilbert space of dimension 2^k) into n qubits (a Hilbert space of dimension 2^n), where $n > k$ and $n - k$ are ancillae qubits. The general idea of Quantum Error Correcting Codes is to encode k qubits whose state we want to protect within a set of n qubit(i.e. within 2^n dimensional Hilbert space) such that there is a special sub-space $\mathcal{C}$, called the codespace, that is spaced by a set of quantum states, span($\{|\psi_i\rangle\}$), i.e. the quantum codewords. The codewords are carefully chosen so that we can guarantee, for a given set of error operators, $\mathcal{E}$, that the error is detectable and the error correctability criteria is met. The key idea is to entangle k logical qubits we want to protect with $n-k$ ancillae qubits such that a subsequent measurement of just the $n-k$ ancillae qubits will project the (n) qubit state into a different orthogonal subspace depending on which type of error has afflicted which of the n qubits.

4.3 Condition for Quantum Error Correction

Consider a single qubit initially in a pure state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ interacting with its environment in the state $|E\rangle$ in an arbitary manner. As the qubit and the environment start off independently, they are in separable states which is represented as:

$$|\psi\rangle \otimes |E\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes |E\rangle. \tag{4.3.1}$$

The evolution of the qubit and its environment under a unitary transformation can be described as follows:

$$U(|0\rangle \otimes |E\rangle) = |0\rangle \otimes |E_{00}\rangle + |1\rangle \otimes |E_{01}\rangle \quad \text{and} \quad U(|1\rangle \otimes |E\rangle) = |0\rangle \otimes |E_{10}\rangle + |1\rangle \otimes |E_{11}\rangle.$$

Hence, a qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ evolves as:

$$\begin{aligned}
U((\alpha|0\rangle + \beta|1\rangle) \otimes |E\rangle)) &= \alpha(|0\rangle \otimes |E_{00}\rangle + |1\rangle \otimes |E_{01}\rangle) + \beta(|0\rangle \otimes |E_{10}\rangle + |1\rangle \otimes |E_{11}\rangle) \\
&= (\alpha|0\rangle + \beta|1\rangle) \otimes \frac{|E_{00}\rangle + |E_{11}\rangle}{2} \quad \text{(no error)} \\
&+ (\alpha|0\rangle - \beta|1\rangle) \otimes \frac{|E_{00}\rangle - |E_{11}\rangle}{2} \quad \text{(phase flip)} \\
&+ (\alpha|1\rangle + \beta|0\rangle) \otimes \frac{|E_{01}\rangle + |E_{10}\rangle}{2} \quad \text{(bit flip)} \\
&+ (\alpha|1\rangle - \beta|0\rangle) \otimes \frac{|E_{01}\rangle - |E_{10}\rangle}{2} \quad \text{(phase and bit flip).}
\end{aligned}$$

The action of U can be expanded in terms of the Pauli operators $\{I, X, Y, Z\}$ since these are the basis for the vector space of 2×2 matrices. One can extend this error model to multiple qubits by assuming the various error types that afflict each qubit independently. Thus, the operators which describes all the independent errors that might afflict n qubits are the elements of the Pauli group, $\mathcal{P}\ _n$=$\{I, X, Y, Z\}^{\otimes n}$, which is the group consisting of all direct products of the Pauli operators $\{I, X, Y, Z\}$ having overall phase 1 or i.

In devising quantum error-correcting codes, we identify a subset $\mathcal{E}$ of all the Pauli group,

$$\mathcal{E} \subset \{E_\alpha\} = \{I, X, Y, Z\}^{\otimes n}$$

It is our aim to perform a collective measurement of the n qubits in the code block that will enable us to diagnose which error $E_\alpha \in \mathcal{E}$ occured.

Criterion of Errors to be Detectable

For an error $E_a \in \mathcal{E}$ that afflicts a quantum codeword to be detectable, every pair of valid quantum codewords $|\psi_i\rangle$ and $|\psi_j\rangle$ that span the encoding space, need to be meet following criterion:

$$\langle\psi_j|E_\alpha|\psi_i\rangle = c_\alpha\delta_{ij}. \qquad (4.3.2)$$

This will guarantee that an error afflicted codeword $E_\alpha|\psi_i\rangle$ will be distinguishable from all the other valid codewords $|\psi_j\rangle$

Criterion for Errors to be Correctable

For an error $E_a \in \mathcal{E}$ that afflicts a quantum codeword to be correctable, it needs to be distinguishable from all the error afflicting all other codewords. That is if $|\psi_i\rangle$ and $|\psi_j\rangle$ are codewords, the following criterion needs to be met:

$$\langle\psi_j|E_\beta E_\alpha|\psi_i\rangle = c_\alpha\beta\delta_{ij} \quad E_\alpha, E_\beta \in \mathcal{E}. \tag{4.3.3}$$

This will guarantee that an error E_α afflicting one codeword will be distinguishable from an error E_β afflicting a different codeword.

4.4 Stabilizer Formalism

The following discussion on Stabilizer has been taken from [1].

A *stablilizer* $\mathcal{S}$=$\{S_1, S_2, ...S_k\}$ is a subgroup of tensor products of the Pauli operator, $S_i \in \{\mathbf{1}, X, Z\}^{\otimes n}$ whose elements are required to have a simultaneous eigenvalue of +1. Hence, the stabilizer is a finite Abelian sub-group of the Pauli group.

Stabilizer for the 5 qubit code

The set of tensor product that satisfy the properties of the stablizer for a 5 qubit system is as follows:

$$S_1 = X \otimes X \otimes Z \otimes X \otimes 1$$

$$S_2 = X \otimes Z \otimes X \otimes 1 \otimes X$$

$$S_3 = Z \otimes 1 \otimes X \otimes X \otimes Z$$

$$S_4 = Z \otimes X \otimes 1 \otimes Z \otimes X$$

$$S_5 = 1 \otimes Z \otimes Z \otimes Z \otimes Z$$

$$S_6 = 1 \otimes 1 \otimes 1 \otimes 1 \otimes 1.$$

It is sufficient to define the stabilizer only with the generator of the group. Here, we can drop S_5 and S_6 since the square of any element of the stabilizer is S_6 and similarly, $S_1 \cdot S_2 \cdot S_3 \cdot S_4 = S_5$. Therefore, the group created by $\{S_1, S_2, S_3, S_4\}$ meets all the criteria to be a stabilizer.

Relation between Error Operator and Stabilizer

The stabilizer is choosen in such a way that every error operator we want to protect against, $E_\alpha \in \mathcal{E}$, commutes with an element of the stabilizer, $S_i \in S$ and hence has an eigenvalue of $+1$ and hence:

$$S_i \cdot E_\alpha = E_\alpha \cdot S_i,$$

$$S_i \cdot E_\alpha |\psi\rangle_L = E_\alpha \cdot S_i |\psi\rangle_L = +1 E_\alpha |\psi\rangle_L.$$

However, this means that when we measure the eigenvalue of the operator S_i, whether the input is error afflicted or not, they both have eigenvalue of $+1$. So, a error afflicted state is not distinguishable from the pure state. However, if the error operator $E_\alpha \in \mathcal{E}$ anti-commutes with an element of the stabilizer $S_i \in S$, and it will have an eigenvalue of -1 and hence:

$$S_i \cdot E_\alpha = -E_\alpha \cdot S_i,$$

$$S_i \cdot E_\alpha |\psi\rangle_L = -E_\alpha \cdot S_i |\psi\rangle_L = -1 E_\alpha |\psi\rangle_L.$$

In this way, the affliction of the error on the pure state is indicated by the fact that the eigenvalue of the operator S_i has become -1. Hence, for a 5 qubit state, the are 16 possible error operator that afflict one of the 5 qubits. Therefore, 4 stabilizers with two distinct eigenvalues on each error afflicted state, guarantees us that each error is distinct and distinguishable.

5
Quantum Algorithms

5.1 Introduction

A quantum Algorithm is a computation process that exploits quantum computing to solve a problem. The representation of a quantum computation process requires an input register, output register and unitary transformation that takes a computational basis states into linear combination of computational basis states. If x represents an n qubit input register and y represents an m qubit output register, then the effect of a unitary transformation U_f on the computational basis $|x\rangle_n|y\rangle_m$ is represented as follows:

$$U_f(|x\rangle_n|y\rangle_m) = |x\rangle_n|y \oplus f(x)\rangle_m, \tag{5.1.1}$$

where f is a function that takes an n qubit input register and returns an m qubit output and $\oplus$ represents mod-2 bitwise addition. If the initial state of the output register is $|0\rangle$ then we get,

$$U_f(|x\rangle_n|0\rangle_m) = |x\rangle_n|f(x)\rangle_m. \tag{5.1.2}$$

The input register remains in its initial state $|x\rangle_n$ and output register gives $|f(x)\rangle_m$.

5.2 Deutsch's Problem

Deutsch's Problem is an example of a quantum algorithm that performs computation in fewer steps than a classical computer. In 1985, David Deutsch first proposed a quantum algorithm that outperform a classical computer. The following derivation of Deutsch's problem is taken from [4].

Let's consider that both input and output registers contain a single qubit, and there is some function f that takes a single bit into a single bit such that

$$U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle. \tag{5.2.1}$$

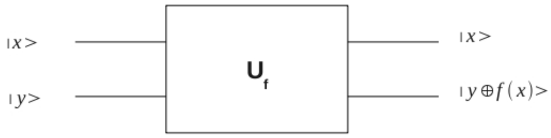

Figure 5.2.1. Circuit Diagram of Deutsch's Problem

Suppose that we are given a box which executes the unitary transformation U_f. There are four possibilities for what U_f can be depending on values of $f(x)$ for $x = 0$ and $x = 1$. This can observed in the table below:

f	x=0	x=1
f_0	0	0
f_1	0	1
f_2	1	0
f_3	1	1

.

Suppose that the box executes U_f once for one of the four funtions but we are not told which one of the four operations it performed. Classically, we can have the box act on one of the four computational basis elements and find out value of either $f(0)$ or $f(1)$. If we let the box act on $|0\rangle|0\rangle$ or $|0\rangle|1\rangle$, we can find out the value of $f(0)$, and similarly if we let the box act on $|1\rangle|0\rangle$ or $|1\rangle|1\rangle$, we can find out the value of $f(1)$. If we find that

$f(0) = 0$ then the function could be f_0 or f_1, and likewise if $f(0) = 1$, then the function could be f_2 or f_3. Hence, in one operation, it is inconclusive which U_f the box executed. In order to tell precisely which one of the four functions it is, we need to perform the unitary tranformation U_f at least twice, once for $x = 0$ and for $x = 1$ and compare the results with the table above.

However, suppose we are interested to find out the relation between $f(0)$ and $f(1)$, whether $f(0) = f(1)$ which is true for f_0 and f_3 or $f(0) \neq f(1)$ which is true for f_1 and f_2. With a classical computer, we need to perform U_f twice for $x = 0$ and $x = 1$ and compare the results. Quite significantly with a quantum computer it can be determined by performing U_f only once. A quantum computer gets this privilege due to an important property called superpostion which says that a qubit can exist in superposition of all possible states or simply stated, a qubit can exist as a linear combination of $|0\rangle$ and $|1\rangle$ states instead of purely one of them. One way to obtain superpostion is by applying the Hadamard transformation to input and output registers before apply U_f. The Hadamard gate produces maximally a entangled state or a superposition of all possible states from a pure state.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\begin{aligned} (H \otimes H)(|1\rangle|1\rangle) &= (\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle) \\ &= \frac{1}{2}(|0\rangle|0\rangle - |1\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|1\rangle). \end{aligned} \tag{5.2.2}$$

After achieving the superposition of all possible states, we apply U_f to get

$$\frac{1}{2}(U_f|0\rangle|0\rangle - U_f|1\rangle|0\rangle - U_f|0\rangle|1\rangle + U_f|1\rangle|1\rangle).$$

From 4.2.1, it follows that application of U_f leads to following state.

$$\frac{1}{2}(|0\rangle|0 \oplus f(0)\rangle - |1\rangle|0 \oplus f(1)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|1 \oplus f(1)\rangle).$$

We know that $\overline{x} = 1 \oplus x$ since $\overline{1} = 0$ and $\overline{0} = 1$. Hence $\overline{f}(x) = 1 \oplus f(x)$

$$\frac{1}{2}(|0\rangle|f(0)\rangle - |1\rangle|f(1)\rangle - |0\rangle|\overline{f}(0)\rangle + |1\rangle|\overline{f}(1)\rangle).$$

On applying the condition, $f(0) = f(1)$ and $\overline{f}(0) = \overline{f}(1)$, we get

$$\frac{1}{2}(|0\rangle - |1\rangle)(|f(0)\rangle - |\overline{f}(0)\rangle). \tag{5.2.3}$$

Similarly, if $f(0) \neq f(1)$ then $\overline{f}(1) = f(0)$ $f(1) = \overline{f}(0)$

$$\frac{1}{2}(|0\rangle + |1\rangle)(|f(0)\rangle - |\overline{f}(0)\rangle). \tag{5.2.4}$$

Hence on applying H back on input register, we get

$$|1\rangle\frac{1}{\sqrt{2}}(|f(0)\rangle - |\overline{f}(0)\rangle), \text{ when } f(0) = f(1), \tag{5.2.5}$$

$$|0\rangle\frac{1}{\sqrt{2}}(|f(0)\rangle - |\overline{f}(0)\rangle), \text{ when } f(0) \neq f(1). \tag{5.2.6}$$

Therefore, we get either $|0\rangle$ or $|1\rangle$ in the input register depending on whether or not $f(0) = f(1)$. In the output register, we get entangled state, $\frac{1}{\sqrt{2}}(|f(0)\rangle - |\overline{f}(0)\rangle)$ for both cases. Therefore, the output register does not give us any useful information about the function.

However, there is an important aspect of this computation that needs some discussion. As we have observed that even though we were able to deduce if the function is constant ($f(0) = f(1)$) or not, we are not able to tell precisely values of $f(0)$ and $f(1)$, since $f(0) = f(1)$ for both f_0 and f_3. In order words, we are not able to pin-point which one of the two functions it is. This is an important feature of quantum computation, where there is a tradeoff between a specific information and relational information. For the Deutsch's problem, to determine the specific information, both classical and quantum compuatation take two different operations whereas for relational information, quantum computation takes one unitary transformation.

Schema of Deutsch's Algorithm

Deutsch's Algorithm can be summarized in three steps.

Step 1 Prepare a maximally entangled state from pure states using the Hadamard gate.

$$(H \otimes H)|1\rangle|1\rangle$$

Step 2 Apply U_f on the maximally entangled state.

$$U_f((H \otimes H)|1\rangle|1\rangle)$$

Step 3 Apply a Hadamard transformation on input register.

$$(H \otimes I)U_f((H \otimes H)|1\rangle|1\rangle)$$

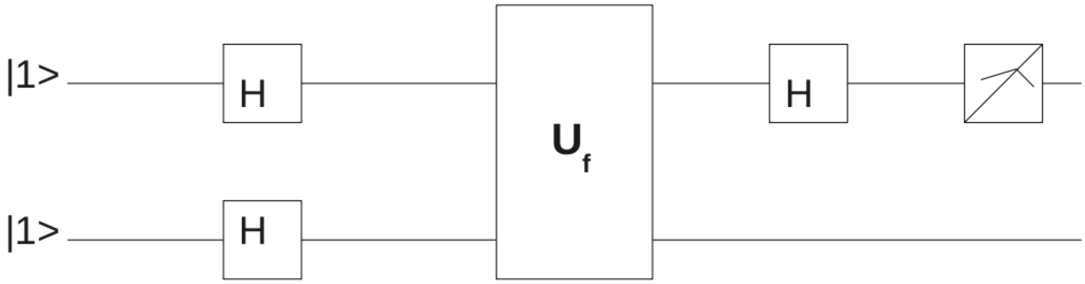

Figure 5.2.2. Schema of Deutsch's Algorithm

5.3 Extension of Deutsch's Problem

In Deutsch's problem involving one qubit input and one qubit output, to obtain relational information, the computational steps are reduced by half. Instead of two applications of U_f, which is the required for a classical computer, simply one computation is sufficient to find if $f(0) = f(1)$ or not. I have found the extension of Deutsch's Algorithm for multiple qubits input a surprisingly interesting problem to investigate. My original contribution to this project begins with extension of Deutsch's problem for multiple qubits.

5.4 Deutsch's Problem with 2 qubits input

Let $|xy\rangle$ be a two qubit input register and $|z\rangle$ be a one qubit output register. Suppose U_f is a unitary transformation on the output register defined as follows:

$$U_f(|xy\rangle|z\rangle) = |xy\rangle|z \oplus f(xy)\rangle. \tag{5.4.1}$$

Suppose we are given a box that executes unitary transformation U_f. There are altogether 16 possibilities for what U_f can be depending on value of $f(xy)$ for each computational basis $|xy\rangle(|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle)$. All 16 possible cases are shown in the table below.

f	xy=00	xy=01	xy=10	xy=11
f_0	0	0	0	0
f_1	1	0	0	0
f_2	0	1	0	0
f_3	0	0	1	0
f_4	0	0	0	1
f_5	1	1	0	0
f_6	0	1	1	0
f_7	0	0	1	1
f_8	1	0	0	1
f_9	1	0	1	0
f_{10}	0	1	0	1
f_{11}	1	1	1	0
f_{12}	0	1	1	1
f_{13}	1	0	1	1
f_{14}	1	1	0	1
f_{15}	1	1	1	1

Suppose that the box executes U_f for one of 16 functions but we are not told which one of the 16 operations it performed. If we let the box act on any composite state with input register $|00\rangle$ then we can find out value of $f(00)$ which has two possible values, 0 and 1. But, there are in total eight functions f which have $f(00) = 0$ while the other eight have $f(00) = 1$. The same is true for all the composite states with input state $|01\rangle$, $|10\rangle$ and

$|11\rangle$. Hence, one action of the box on a initial composite state cuts total possibilities in half for what U_f is. Clearly, we need to perform U_f on initial state when the input state is $|00\rangle, |01\rangle, |10\rangle,$ and $|11\rangle$ and compare all values in order to deduce which one of the 16 operations the box executes.

However, suppose we are interested in the relation between $f(00), f(01), f(10), f(11)$. For example, we are interested to know whether $f(00) = f(01) = f(10) = f(11)$ (true for f_0 and f_{15}) or $f(00) = f(01)$ and $f(10) = f(11)$ (true for f_5 and f_7)or $f(00) = f(11)$ and $f(01) = f(10)$ (true for f_6 and f_8), etc. With a classical computer, we need to perform the unitary transformation, U_f on initial states with input register in states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ and compare their values. However, with a quantum computer, we can determine such relational information in only one action of U_f. In order to do this, we need to prepare states entangling of all 3 qubits. One way of obtaining such entangled state is by applying the Hadamard transformation. Before applying U_f, we apply the Hadamard transformation in order to produce such entangled state. The Hadamard gate for 2 qubits is simply the tensor product of two Hadmard gates for single qubit:

$$H_2 = H_1 \otimes H_1 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

$$\begin{aligned}(H_2 \otimes H_1)(|11\rangle|1\rangle) =& \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|00\rangle|0\rangle - |00\rangle|1\rangle - |01\rangle|0\rangle + |01\rangle|1\rangle \\ & - |10\rangle|0\rangle + |10\rangle|1\rangle + |11\rangle|0\rangle - |11\rangle|1\rangle).\end{aligned} \tag{5.4.2}$$

After we have superposition of all possible state, we apply U_f,

$$\frac{1}{2}(U_f|0\rangle|0\rangle - U_f|1\rangle|0\rangle - U_f|0\rangle|1\rangle + U_f|1\rangle|1\rangle).$$

From 4.4.1, it follows that application of U_f leads to following state:

$$\frac{1}{2\sqrt{2}}(|00\rangle|f(00)\rangle - |00\rangle|1 \oplus f(00)\rangle - |01\rangle|f(01)\rangle + |01\rangle|1 \oplus f(01)\rangle$$
$$-|10\rangle|f(10)\rangle + |10\rangle|1 \oplus f(10)\rangle + |11\rangle|f(11)\rangle - |11\rangle|1 \oplus f(11)\rangle).$$

We know $\overline{x} = 1 \oplus x$ and $\overline{f}(xy) = 1 \oplus f(xy)$.

$$\begin{aligned}&\frac{1}{2\sqrt{2}}(|00\rangle|f(00)\rangle - |00\rangle|\overline{f}(00)\rangle - |01\rangle|f(01)\rangle + |01\rangle|\overline{f}(01)\rangle - \\ &|10\rangle|f(10)\rangle + |10\rangle|\overline{f}(10)\rangle + |11\rangle|f(11)\rangle - |11\rangle|\overline{f}(11)\rangle).\end{aligned} \tag{5.4.3}$$

$$\text{Condition I, } f(00) = f(01) = f(10) = f(11).$$
$$\overline{f}(00) = \overline{f}(01) = \overline{f}(10) = \overline{f}(11).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle) \tag{5.4.4}$$

$$\text{Condition II,} f(00) = f(10) = \overline{f}(01) = \overline{f}(11).$$
$$f(01) = f(11) = \overline{f}(00) = \overline{f}(10).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle) \tag{5.4.5}$$

$$\text{Condition III,} f(00) = f(01) = \overline{f}(10) = \overline{f}(11).$$
$$f(10) = f(11) = \overline{f}(00) = \overline{f}(01).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle) \tag{5.4.6}$$

$$\text{Condition IV,} f(00) = f(11) = \overline{f}(01) = \overline{f}(10).$$
$$f(01) = f(10) = \overline{f}(00) = \overline{f}(11).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle \tag{5.4.7}$$

On applying H back on the input register for each condition we obtain:

$$|11\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) = f(01) = f(10) = f(11), \tag{5.4.8}$$

$$|10\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) = f(10) \text{ and } f(01) = f(11), \tag{5.4.9}$$

$$|01\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) = f(01) \text{ and } f(10) = f(11), \tag{5.4.10}$$

$$|00\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) = f(11) \text{ and } f(01) = f(10). \tag{5.4.11}$$

Hence we get either $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|11\rangle$ in the input register depending on the relation between $f(00)$, $f(01)$, $f(10)$ and $f(11)$. In the output register, we get entangled state, $\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle)$ in all cases. Therefore, the output register does not give us useful information about the function.

In the one qubit problem, broadly speaking there are two kinds of functions: one where $f(0) = f(1)$ and another where $f(0) \neq f(1)$. More importantly, the Hadamard gate is able to discriminate between these functions. In the two qubit problem, there are four functions alone for which two out of four output values are equal. Using the Hadamard gate, we are able to discriminate all four such functions. In addition to that, there are four other functions for which three out of four output are equal namely $f_1, f_2, f_3,$ and f_4. Note that there are four other functions $f_{11}, f_{12}, f_{13}, f_{14}$, but they also represent the same information.

f	xy=00	xy=01	xy=10	xy=11
f_1	1	0	0	0
f_2	0	1	0	0
f_3	0	0	1	0
f_4	0	0	0	1

First, let's see what happens if we try to apply conditions for f_1 on the entangled state prepared using the Hadamard gate in 4.4.4.

$$\text{Conditions } \overline{f}(00) = f(01) = f(10) = f(11).$$

$$f(00) = \overline{f}(01) = \overline{f}(10) = \overline{f}(11).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle). \tag{5.4.12}$$

On applying H_2 on the input register, we get

$$\frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

On applying, H_2 we get an entangled state, which means that following the schema of Deutsch's algorithm we cannot discriminate functions that are like f_1. In fact, we cannot discriminate all four functions f_1, f_2, f_3 and f_4 since we cannot get a pure state on applying H_2 on the input register as we found for f_1. Therefore we need a different gate for step 1 to prepare entangled state so that in step 3, it can also give pure state. Incidently, there is one such gate which fulfills the above conditions, the GregMatt[1] gate.

$$GM = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix}$$

Again, we follow the same schema where we first prepare the entangled state using the GregMatt gate.

$$\begin{aligned}(GM \otimes H_1)(|11\rangle|1\rangle) =& \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|00\rangle|0\rangle - |00\rangle|1\rangle + |01\rangle|0\rangle - |01\rangle|1\rangle \\ & + |10\rangle|0\rangle - |10\rangle|1\rangle - |11\rangle|0\rangle + |11\rangle|1\rangle)\end{aligned} \tag{5.4.13}$$

After we have superposition of all possible state, we can apply U_f on the qubits,

$$\begin{aligned}&\frac{1}{2\sqrt{2}}(|00\rangle|f(00)\rangle - |00\rangle|1 \oplus f(00)\rangle + |01\rangle|f(01)\rangle - |01\rangle|1 \oplus f(01)\rangle \\ &+|10\rangle|f(10)\rangle - |10\rangle|1 \oplus f(10)\rangle - |11\rangle|f(11)\rangle + |11\rangle|1 \oplus f(11)\rangle)\end{aligned}$$

We know $\overline{x} = 1 \oplus x$ and $\overline{f}(xy) = 1 \oplus f(xy)$.

$$\begin{aligned}&\frac{1}{2\sqrt{2}}(|00\rangle|f(00)\rangle - |00\rangle|\overline{f}(00)\rangle + |01\rangle|f(01)\rangle - |01\rangle|\overline{f}(01)\rangle + \\ &|10\rangle|f(10)\rangle - |10\rangle|\overline{f}(10)\rangle - |11\rangle|f(11)\rangle + |11\rangle|\overline{f}(11)\rangle)\end{aligned} \tag{5.4.14}$$

[1] The gate is named after my advisors Greg Landweber and Matthew Deady.

$$\text{Condition I, } f(01) = f(10) = f(11) = \overline{f}(00).$$

$$\overline{f}(01) = \overline{f}(10) = \overline{f}(11) = f(00).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle) \tag{5.4.15}$$

$$\text{Condition II, } f(00) = \overline{f}(01) = f(10) = f(11).$$

$$\overline{f}(00) = f(01) = \overline{f}(10) = \overline{f}(11).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle) \tag{5.4.16}$$

$$\text{Condition III,} f(00) = f(01) = \overline{f}(10) = f(11).$$

$$\overline{f}(00) = \overline{f}(01) = f(10) = \overline{f}(01).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle) \tag{5.4.17}$$

$$\text{Condition IV,} f(00) = f(01) = f(10) = \overline{f}(11).$$

$$\overline{f}(00) = \overline{f}(01) = f(10) = \overline{f}(11).$$

$$\frac{1}{2\sqrt{2}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)(|f(00)\rangle - |\overline{f}(00)\rangle \tag{5.4.18}$$

On applying GM back on the input register for each condition we obtain:

$$|11\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) \neq f(01) = f(10) = f(11), \tag{5.4.19}$$

$$|10\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) = f(10) = f(11) \neq f(01), \tag{5.4.20}$$

$$|01\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) = f(01) = f(11) \neq f(10), \tag{5.4.21}$$

$$|00\rangle\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle), \text{ when } f(00) = f(01) = f(10) \neq f(11). \tag{5.4.22}$$

Hence we get either $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|11\rangle$ in the input register depending on the relation between $f(00)$, $f(01)$, $f(10)$ and $f(11)$. In the output register, we get entangled state, $\frac{1}{\sqrt{2}}(|f(00)\rangle - |\overline{f}(00)\rangle)$ in all cases.

Therefore, we are able to completely solve the two qubit Deutsch's problem. We have found that for relational information we can distinguish functions in one application of U_f. This is a significant improvement from classical computing, where at least four computations are required to find such relational information.

5.5 Some Important Observations on the 2 qubit Deutsch's Problem

We have observed that in the 2 qubit problem, there are broadly two kinds of functions. The first kind has two out of four output equal and second kind has three out of four outputs equal. In order to discriminate first kind, we need the Hadamard gate for entanglement and to discriminate second kind, we need the GregMatt gate. Therefore, there must be some kind of relationship between the gate that is used for entanglement and the functions which the gate can discriminate. In this section, we will make observations regarding the properties of entangling gates.

Entangling gate contains information about the functions it discriminates

There is an interesting relationship between the entries in the entangling gate and values for functions that are identifiable using such gate. When we look at the entries of the Hadamard gate and table of functions with their values, we can observe a direct correspondence between each function's values and entries of Hadamard gate.

f	xy=00	xy=01	xy=10	xy=11
f_0	0	0	0	0
f_{10}	0	1	0	1
f_7	0	0	1	1
f_6	0	1	1	0

$$H_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

For each 0 in the table of function's values, the Hadamard gate has 1 and for 1 it has -1 in corresponding positions. Hence, the Hadamard gate (H_2) holds information about the functions that it can identify.

Interestingly, there is no such direct relationship between functions which have three output values equal and the entangling gate.

f	xy=00	xy=01	xy=10	xy=11
f_1	1	0	0	0
f_2	0	1	0	0
f_3	0	0	1	0
f_4	0	0	0	1

$$GM = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix}.$$

However, the relationship between the functional values and GM gate is indirect. If we suppose that f_1 corresponds with first the row of the GM gate then change in the output values for each function corresponds to change in the GM gate from 1 to -1. For example, function f_0 has output values 1, 0, 0, 0 which corresponds to first row of the GregMatt gate which are all 1's. The second function f_1 has output values 0, 1, 0, 0 which means there are two places where they differ, one at $f(00)$ and another at $f(01)$. As a result, the second row in the GM gate has -1 at first two entries. Similar patterns follow for third and fourth rows of the GM gate.

An Example

The following example can shed some light on the properties of entangling gates:

$$MS = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}.$$

The MS gate is another example of an entangling gate that can distinguish functions that for which two out of four output values are equal. We can show this by using the schema of Deutsch's Algorithm that we have used in earlier problems. First, we begin by entangling the initial state using the MS gate.

$$\begin{aligned}(MS \otimes H_1)(|11\rangle|1\rangle) =& \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|00\rangle|0\rangle - |00\rangle|1\rangle + |01\rangle|0\rangle - |01\rangle|1\rangle \\ & + |10\rangle|0\rangle - |10\rangle|1\rangle - |11\rangle|0\rangle + |11\rangle|1\rangle).\end{aligned} \tag{5.5.1}$$

After we have superposition of all possible state, we can apply U_f to get,

$$\begin{aligned}&\frac{1}{2\sqrt{2}}(|00\rangle|f(00)\rangle - |00\rangle|1 \oplus f(00)\rangle + |01\rangle|f(01)\rangle - |01\rangle|1 \oplus f(01)\rangle \\ &+ |10\rangle|f(10)\rangle - |10\rangle|1 \oplus f(10)\rangle - |11\rangle|f(11)\rangle + |11\rangle|1 \oplus f(11)\rangle)\end{aligned}$$

We know $\overline{x} = 1 \oplus x$ and $\overline{f}(xy) = 1 \oplus f(xy)$, so

$$\begin{aligned}&\frac{1}{2\sqrt{2}}(|00\rangle|f(00)\rangle - |00\rangle|\overline{f}(00)\rangle + |01\rangle|f(01)\rangle - |01\rangle|\overline{f}(01)\rangle + \\ &|10\rangle|f(10)\rangle - |10\rangle|\overline{f}(10)\rangle - |11\rangle|f(11)\rangle + |11\rangle|\overline{f}(11)\rangle).\end{aligned}$$

We will use a tabular representation which simply our algebraic manipulations.

$U_f(MS \otimes H_1)(\lvert 11\rangle \lvert 1\rangle)$	0000	0011	0101	0110
$\lvert 00\rangle \lvert f(00)\rangle - \lvert 00\rangle \lvert \overline{f}(00)\rangle$	1	1	1	1
$\lvert 01\rangle \lvert f(01)\rangle - \lvert 01\rangle \lvert \overline{f}(01)\rangle$	1	1	-1	-1
$\lvert 10\rangle \lvert f(10)\rangle - \lvert 10\rangle \lvert \overline{f}(10)\rangle$	1	-1	1	-1
$-\lvert 11\rangle \lvert f(11)\rangle + \lvert 11\rangle \lvert \overline{f}(11)\rangle$	-1	1	1	-1

The left most column of the table is the entangled state $U_f(MS \otimes H_1)(|11\rangle|1\rangle)$. The column on the right of it are the coefficient of $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ for functions which has values on top of the column. The column on the right also represents the entangled state on the input register after applying U_f. For example: On top of the second column we have 0000 which represents the case when $f(00) = f(01) = f(10) = f(11)$ and when this condition is applied on the entangled state, we get

$$(|00\rangle + |01\rangle + |10\rangle - |11\rangle)(f(00) - \overline{f}(00)).$$

Following the schema of Deutsch's Algorithm, we apply the MS transformation back on the input register. Therefore, we can apply the MS gate to all the different functions.

$$\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 4 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Therefore, we obtain a pure state on applying the MS on the input register. This confirm that the MS can be another gate that discriminates functions which have two out of four values equal. The result is not quite surprising if we look at that MS gate and functions with their values.

$$MS = \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

f	xy=00	xy=01	xy=10	xy=11
f_0	0	0	0	0
f_{10}	1	1	0	0
f_7	1	0	1	0
f_6	1	0	0	1

We can observe that the MS gate in fact holds information about the functions it distinguishes even though it might not be as direct as it appeared in the Hadamard gate. If we consider that the first row of the MS gate corresponds with the first function, then change in the functional values from 0 to 1 along each column corresponds with the change in the sign of the entries in the MS gate along same column. For example: In the first column of the function's values changes from 0 in first row to 1 in second, third and fourth row which corresponds to change in the first column where there -1 in the first row and

1 in second, third and fourth row of the MS gate. Thus, the MS gate holds information about all the functions.

An entangling gate cannot discriminate two different kinds of functions

The 2 qubit problem has broadly two kinds of functions. The first kind which have two out of four output values equal are distinguished using the Hadamard gate and second kind which have three out of four output values equal are distinguished using the GregMatt gate. We found out earlier that the Hadamard gate cannot discriminate functions which have three output values equal. The same rule applies for the GregMatt gate. It cannot discriminate functions which have two output values equal. However, it turns out that GregMatt gate also represents functions which have two output values equal.

$$GM = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix}$$

f	xy=00	xy=01	xy=10	xy=11
f_0	0	0	0	0
f_{10}	1	1	0	0
f_7	1	0	1	0
f_6	1	0	0	1

Clearly, the GM gate contains information about these four functions. There is direct relationship between the entries in the gate and output of the functions in the table. All the 0's and 1's in the functional table corresponds to 1 and -1 in the GM gate. We will again follow the schema of Deutsch's algorith and use tabluar representation to write the entangled states and coefficient of the input register for each functions.

$U_f(GM \otimes H_1)(\|11\rangle\|1\rangle)$	0000	1100	1010	1001
$\|00\rangle\|f(00)\rangle - \|00\rangle\|\overline{f}(00)\rangle$	1	1	1	1
$\|01\rangle\|f(01)\rangle - \|01\rangle\|\overline{f}(01)\rangle$	1	1	-1	-1
$\|10\rangle\|f(10)\rangle - \|10\rangle\|\overline{f}(10)\rangle$	1	-1	1	-1
$-\|11\rangle\|f(11)\rangle + \|11\rangle\|\overline{f}(11)\rangle$	-1	1	1	-1

For each functions, we have listed the coefficient for $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, which the state of the input register after applying U_f. Following the final step of the schema, we apply GM back on the input register.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

Therefore on applying the GM transformation, we do not get pure state, which means GM can not discriminate the functions. Although, the GM gate contains relational information about the functions, it can not distinguish between the functions.

Group Structure

One important observation that can be made about the functions that are encoded by the Hadamard gate is that they form a group under modulo-2 addition. Specifically, the functions form a Klein-4 group.

f	xy=00	xy=01	xy=10	xy=11
f_0	1	0	0	0
f_{10}	0	1	0	1
f_7	0	0	1	0
f_6	0	0	0	1

5.6 Deutsch's Problem with 3 qubits input

Deutsch's problem can be extended to 3 qubits input. In order to do this, first consider a input register containing three qubits and an output register containing one qubit.

Consider a function f that takes 3 bits input into a single bit,

$$U_f(|xyz\rangle|w\rangle) = |xyz\rangle|w \oplus f(xyz)\rangle. \tag{5.6.1}$$

There are altogether 256 possibilities for what U_f can be depending on the value of $f(x, y, z)$ for each computational basis of $|xyz\rangle$. It is convenient to divide the total possibilities into cases depending on the number of 1 and 0 each class of functions have. So, the problem can be broken down into nine cases.

0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1

There is 1 function which has all eight outputs 0. Similarly, there are 8 functions with one output 1, there are 28 functions with two outputs 1, there are 56 functions with three outputs 1, there are 70 functions with four outputs 1, there are 56 functions with five output 1, there are 28 functions with six outputs 1, there are 8 functions with seven outputs 1 and there is 1 function with all eight outputs 1. However, as we discussed earlier, the scope of Deutsch's Problem is to determine the relationship information rather than specific information. As such a function that has all eight outputs 0 and a function that has all eight outputs 1 are essential same since both represent conditions,

$$f(000) = f(001) = f(010) = f(011) = f(100) = f(101) = f(110) = f(111).$$

The same notion is true for functions that have 2 outputs 1 or 2 output 0 and so on. In both 1 qubit and 2 qubits cases, the first thing we were interested was to determine what the tranformation (U_f) represent for functions where half of the output values are equal to each other. In order to do we used the Hadamard Gate, H_1 for 1 qubit and H_2 for 2 qubits.

Therefore, our initial guess would be to used a H_3, which is simply $H_1 \otimes H_1 \otimes H_1$ to deduce the functions where 4 out of 8 outputs are equal to each other. There are essentially 36 functions which have half of the output values equal to each other which are listed below:

xyz=000	xyz=001	xyz=010	xyz=011	xyz=100	xyz=101	xyz=110	xyz=111
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	1
0	1	0	0	0	1	1	1
0	0	0	1	1	0	1	1
0	0	1	0	1	0	1	1
0	1	0	0	1	0	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	0	1	1
0	1	1	0	0	0	1	1
0	0	0	1	1	1	0	1
0	0	1	0	1	1	0	1
0	1	0	0	1	1	0	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	1	0	1
0	0	1	1	1	0	0	1
0	1	0	1	1	0	0	1
0	1	1	0	1	0	0	1
0	1	1	1	0	0	0	1
0	0	0	1	1	1	1	0
0	0	1	0	1	1	1	0
0	1	0	0	1	1	1	0
0	0	1	1	0	1	1	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	0
0	0	1	1	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	0	1	0	1	0
0	1	1	1	0	0	1	0
0	0	1	1	1	1	0	0
0	1	0	1	1	1	0	0
0	1	1	0	1	1	0	0
0	1	1	1	0	1	0	0
0	1	1	1	1	0	0	0

.

The 3 qubit Hadamard gate, H_3, is one of the candidates for gates that can entangle and disentangle the functions for which half of the outputs are equal to each other. However, it can at most deduce 8 such functions. One way to find those eight functions is to first entangle the input and output registers and apply each of 36 conditions and pick the ones that give back a pure state on apply H_3. There is an easier way to come up with the functions that H_3 can deduce. If we recall our observations in the 2 qubit Deutsch's problem, we found that for H_2, there is a direct correspondence between 1 and -1 with 0 and 1 in the table of the output values of the functions. Therefore, by looking at the entries of H_3, we can find the functions which are deduced by it. For each 1 appearing in H_3 which put 0 and for each -1 we put 1 in the corresponding position in the table of function's values.

$$H_3 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

xyz=000	xyz=001	xyz=010	xyz=011	xyz=100	xyz=101	xyz=110	xyz=111
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0
0	0	0	0	1	1	1	1
0	1	0	1	1	0	1	0
0	0	1	1	1	1	0	0
0	1	1	0	1	0	0	1

We can confirm that the above 8 functions are deduced by H_3 by carrying the detailed calculation in tabular form as we did in the 2 qubit problem.

$$\begin{aligned}(H_3 \otimes H_1)(|111\rangle|1\rangle) =& \frac{1}{2\sqrt{2}}(|000\rangle - |001\rangle - |010\rangle + |011\rangle \\ & - |100\rangle + |101\rangle + |110\rangle - |111\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{4}(|000\rangle|0\rangle - |000\rangle|1\rangle - |001\rangle|0\rangle + |001\rangle|1\rangle - |010\rangle|0\rangle + |010\rangle|1\rangle \\ & + |011\rangle|0\rangle - |011\rangle|1\rangle - |100\rangle|0\rangle + |100\rangle|1\rangle + |101\rangle|0\rangle - |101\rangle|1\rangle \\ & + |110\rangle|0\rangle - |110\rangle|1\rangle - |111\rangle|0\rangle + |111\rangle|1\rangle).\end{aligned} \tag{5.6.2}$$

On applying U_f and conditions, $\overline{x} = 1 \oplus x$ and $\overline{f}(x,y,z) = 1 \oplus f(x,y,z)$ we get,

$$\begin{aligned}&\frac{1}{4}(|000\rangle|f(000)\rangle - |000\rangle|\overline{f}(000)\rangle - |001\rangle|f(001)\rangle + |001\rangle|\overline{f}(001)\rangle \\ &-|010\rangle|f(010)\rangle + |010\rangle|\overline{f}(010)\rangle + |011\rangle|f(011)\rangle - |011\rangle|\overline{f}(011)\rangle \\ &-|100\rangle|f(100)\rangle + |100\rangle|\overline{f}(100)\rangle + |101\rangle|f(101)\rangle - |101\rangle|\overline{f}(101)\rangle \\ &+|110\rangle|f(110)\rangle - |110\rangle|\overline{f}(110)\rangle - |111\rangle|f(111)\rangle + |111\rangle|\overline{f}(111)\rangle).\end{aligned}$$

$\mathrm{U}_f(H_3 \otimes H_1)(\|111\rangle\|1\rangle)$	00000000	01010101	00110011	01100110	00001111	01011010	00111100	01101001
$\|000\rangle\|f(000)\rangle - \|000\rangle\|\overline{f}(000)\rangle$	1	1	1	1	1	1	1	1
$-\|001\rangle\|f(001)\rangle + \|001\rangle\|\overline{f}(001)\rangle$	-1	1	-1	1	-1	1	-1	1
$-\|010\rangle\|f(010)\rangle + \|010\rangle\|\overline{f}(010)\rangle$	-1	-1	1	1	-1	-1	1	1
$\|011\rangle\|f(011)\rangle - \|011\rangle\|\overline{f}(011)\rangle$	1	-1	-1	1	1	-1	-1	1
$-\|100\rangle\|f(100)\rangle + \|100\rangle\|\overline{f}(100)\rangle$	-1	-1	-1	-1	1	1	1	1
$-\|101\rangle\|f(101)\rangle + \|101\rangle\|\overline{f}(101)\rangle$	-1	-1	-1	-1	1	1	1	1
$\|110\rangle\|f(110)\rangle - \|110\rangle\|\overline{f}(110)\rangle$	1	1	-1	-1	-1	-1	1	1
$\|111\rangle\|f(111)\rangle - \|111\rangle\|\overline{f}(111)\rangle$	-1	1	1	-1	1	-1	-1	1
$\mathrm{H}_3 U_f(H_3 \otimes H_1)(\|111\rangle\|1\rangle)$	$\|111\rangle$	$\|110\rangle$	$\|101\rangle$	$\|100\rangle$	$\|011\rangle$	$\|010\rangle$	$\|001\rangle$	$\|000\rangle$

The leftmost column of the above table contains the entangled state $U_f(H_3 \otimes H_1)(|111\rangle|1\rangle)$ and each column to its right contains the coeffiecient of each pure state for the functions on top of the column. The bottom of the table has the distinct pure state that is obtained on applying H_3 back on the entangled state for each function.

We have found 8 functions that the Hadamard gates encodes and decodes. However, we do not know the gates that can encode and decode the remaining 28 functions.

5.7 Some Important Observations on the 3 qubit Deutsch's Problem

In the 2 qubit problem we have noticed that H_2 encodes the information about the functions and is also self adjoint. In the 3 qubit problem too, H_3 encodes information about the functions that it deduces and is self adjoint. There is another important observation that we made about the functions that H_2 encoded and decoded, it was the function group a Klein-4 group under modulo 2 addition. In the problem too, it turns out that the functions that H_3 encodes form a Klein-4 group under modulo 2 addition. In fact, this property of the functions plays an important role in determining the gates that we need to encode and decode the remaining 28 functions that H_3 is not able to encode.

Group Structure

Let's look back at the functions that H_3 was able to encode.

xyz=000	xyz=001	xyz=010	xyz=011	xyz=100	xyz=101	xyz=110	xyz=111
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0
0	0	0	0	1	1	1	1
0	1	0	1	1	0	1	0
0	0	1	1	1	1	0	0
0	1	1	0	1	0	0	1

It turns out that the functions here form a group structure under mod 2 addition. This information is key for finding the gates that can encode and decode functions. In order to figure out the gates other than H_3 that can encode and decode the remaining 28 functions we need to follow two rules. 1. The functions that we want to encode must altogether form a group structure. 2. The entries of the gate that encodes such group of functions has one to one relatoinship with the functional values. More precisely, the 0 in the functional table corresponds to 1 in the gate and 1 in the functional table corresponds to -1 in the gate.

Let's take an example of the functions that form a group and find the gate that encodes the information about the function.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The gate that encodes these functions follows the two rules that mentioned earlier.

$$AD = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 \end{pmatrix}.$$

We can confirm that the above 8 functions are deduced by H_3 by carrying the detailed calculation in tabular form as we did in the 2 qubit problem.

$$\begin{aligned}(AD \otimes H_1)(|111\rangle|1\rangle) =& \frac{1}{2\sqrt{2}}(|000\rangle - |001\rangle + |010\rangle - |011\rangle \\ & + |100\rangle + |101\rangle - |110\rangle - |111\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{4}(|000\rangle|0\rangle - |000\rangle|1\rangle - |001\rangle|0\rangle + |001\rangle|1\rangle + |010\rangle|0\rangle - |010\rangle|1\rangle \\ & - |011\rangle|0\rangle + |011\rangle|1\rangle + |100\rangle|0\rangle - |100\rangle|1\rangle + |101\rangle|0\rangle - |101\rangle|1\rangle \\ & - |110\rangle|0\rangle + |110\rangle|1\rangle - |111\rangle|0\rangle + |111\rangle|1\rangle).\end{aligned} \tag{5.7.1}$$

On applying U_f and conditions, $\overline{x} = 1 \oplus x$ and $\overline{f}(x, y, z) = 1 \oplus f(x, y, z)$ we get,

$$\begin{aligned}&\frac{1}{4}(|000\rangle|f(000)\rangle - |000\rangle|\overline{f}(000)\rangle - |001\rangle|f(001)\rangle + |001\rangle|\overline{f}(001)\rangle \\ &+|010\rangle|f(010)\rangle - |010\rangle|\overline{f}(010)\rangle - |011\rangle|f(011)\rangle + |011\rangle|\overline{f}(011)\rangle \\ &+|100\rangle|f(100)\rangle - |100\rangle|\overline{f}(100)\rangle + |101\rangle|f(101)\rangle - |101\rangle|\overline{f}(101)\rangle \\ &-|110\rangle|f(110)\rangle + |110\rangle|\overline{f}(110)\rangle - |111\rangle|f(111)\rangle + |111\rangle|\overline{f}(111)\rangle).\end{aligned}$$

U$_f(AD \otimes H_1)(\|111\rangle\|1\rangle)$	00000000	01010101	00110011	01100110	00001111	01011010	00111100	01101001
$\|000\rangle f(000) - \|000\rangle \overline{f}(000)$	1	1	1	1	1	1	1	1
$-\|001\rangle f(001) + \|001\rangle \overline{f}(001)$	-1	1	-1	1	-1	1	-1	1
$\|010\rangle f(010) - \|010\rangle \overline{f}(010)$	-1	-1	1	1	-1	-1	1	1
$-\|011\rangle f(011) + \|011\rangle \overline{f}(011)$	1	-1	-1	1	1	-1	-1	1
$\|100\rangle f(100) - \|100\rangle \overline{f}(100)$	-1	-1	-1	-1	1	1	1	1
$\|101\rangle f(100) - \|101\rangle \overline{f}(101)$	-1	-1	-1	-1	1	1	1	1
$-\|110\rangle f(110) + \|110\rangle \overline{f}(110)$	1	1	-1	-1	-1	-1	1	1
$-\|111\rangle f(111) + \|111\rangle \overline{f}(111)$	-1	1	1	-1	1	-1	-1	1
H$_3U_f(H_3 \otimes H_1)(\|111\rangle\|1\rangle)$	$\|111\rangle$	$\|110\rangle$	$\|101\rangle$	$\|100\rangle$	$\|011\rangle$	$\|010\rangle$	$\|001\rangle$	$\|000\rangle$

Hence, the AD gate successfully discriminates the above functions which formed an abelian group. This can be generalized for all the functions having 4 out of 8 values equal to each other. Any set of functions that form a subgroup can be discriminated by a gate that has a direct correspondence with the functions. At this point, we know that all functions that have 4 out of 8 values equal can be distinguished in single operaiton of U_f. With this, we are left with functions that have 7 output values equal, functions that have 6 output values equal and the ones with 5 output values equal.

5.8 Functions which have all but one output values equal to each other

One of the kinds of function that the Hadamard gate cannot discriminate are the ones with 7 output values equal. We can list them in a table.

xyz=000	xyz=001	xyz=010	xyz=011	xyz=100	xyz=101	xyz=110	xyz=111
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

The most challenging part of solving this problem is to find a gate that can entangle the initial states and for each of these functions return a distinct pure state, so that each of thesefunctions is identifiable.After making close observations for the 2 qubit case and many trials and errors, we have finally come up with a gate that can distinguish these functions with a slight variation in the schema of the Deutsch's algorithm. This gate is called the GregMatt1 as it is simply the tensor product of Idenitity matrix with the GregMatt gate.

$$GM_1 = I \otimes GM = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

One thing we have notice here is that the GM_1 is the first instance of the entangling gate which has 0 as its entries. In order to discriminate all of these functions with the GM_1 gate, we need to divide this problem into two parts. The first part handles the first four functions, and the second deals with the last four functions.

Part I

In our schema of Deutsch's algorithm, we have always choosen states like $|1\rangle$ for 1 qubit and $|11\rangle$ for 2 qubits as the initial state of the input register. However, in the first part of this particular problem we will choose $|011\rangle$ as the initial state of the input register. Hence we use the GM_1 gate to entangle the input state and the Hadamard gate to entangle the output register to get,

$$\begin{aligned}(GM_1 \otimes H_1)(|011\rangle|1\rangle) =& \frac{1}{2}(|000\rangle + |001\rangle + |010\rangle - |011\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|000\rangle|0\rangle - |000\rangle|1\rangle + |001\rangle|0\rangle - |001\rangle|1\rangle \\ & + |010\rangle|0\rangle - |010\rangle|1\rangle - |011\rangle|0\rangle + |011\rangle|1\rangle).\end{aligned} \tag{5.8.1}$$

Notice that the GM_1 gate does not maximally entangle the state of input register. Once we have the entangled state, we apply U_f transformation which gives us,

$$\frac{1}{2\sqrt{2}}(|000\rangle|f(000)\rangle - |000\rangle|1 \oplus f(000)\rangle + |001\rangle|f(001)\rangle - |001\rangle|1 \oplus f(001)\rangle$$
$$+ |010\rangle|f(010)\rangle - |010\rangle|1 \oplus f(010)\rangle - |011\rangle|f(011)\rangle + |011\rangle|1 \oplus f(011)\rangle).$$

We know, $\overline{x} = 1 \oplus x$ and $\overline{f}(xyz) = 1 \oplus f(xyz)$, so

$$\frac{1}{2\sqrt{2}}(|000\rangle|f(000)\rangle - |000\rangle|\overline{f}(000)\rangle + |001\rangle|f(001)\rangle - |001\rangle|\overline{f}(001)\rangle$$
$$+ |010\rangle|f(010)\rangle - |010\rangle|\overline{f}(010)\rangle - |011\rangle|f(011)\rangle + |011\rangle|\overline{f}(011)\rangle).$$

We can use the tabular form to confirm our assumptions.

$U_f(GM_1 \otimes H_1)(\|011\rangle\|1\rangle)$	10000000	01000000	00100000	00010000
$\|000\rangle\|f(000)\rangle - \|000\rangle\|\overline{f}(000)\rangle$	1	1	1	1
$\|001\rangle\|f(001)\rangle - \|001\rangle\|\overline{f}(001)\rangle$	-1	-1	1	1
$\|010\rangle\|f(010)\rangle - \|010\rangle\|\overline{f}(010)\rangle$	-1	1	-1	1
$-\|011\rangle\|f(011)\rangle + \|011\rangle\|\overline{f}(011)\rangle$	1	-1	-1	1
$0 \cdot \|100\rangle$	0	0	0	0
$0 \cdot \|101\rangle$	0	0	0	0
$0 \cdot \|110\rangle$	0	0	0	0
$0 \cdot \|111\rangle$	0	0	0	0
$GM_1 U_f(GM_1 \otimes H_1)(\|011\rangle\|1\rangle)$	$\|011\rangle$	$\|010\rangle$	$\|001\rangle$	$\|000\rangle$

Therefore, part I successfully discriminates the first four functions.

Part II

For the other four functions, we will use $|111\rangle$ as the initial state of input register.

$$\begin{aligned}(GM_1 \otimes H_1)(|111\rangle|1\rangle) =& \frac{1}{2}(|100\rangle + |101\rangle + |110\rangle - |111\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|100\rangle|0\rangle - |100\rangle|1\rangle + |101\rangle|0\rangle - |101\rangle|1\rangle \\ & + |110\rangle|0\rangle - |110\rangle|1\rangle - |111\rangle|0\rangle + |111\rangle|1\rangle).\end{aligned} \tag{5.8.2}$$

Notice that the GM_1 gate does not maximally entangle the state of input register. Once we have the entangled state, we apply U_f transformation which gives us,

$$\frac{1}{2\sqrt{2}}(|100\rangle|f(100)\rangle - |100\rangle|1 \oplus f(100)\rangle + |101\rangle|f(101)\rangle - |101\rangle|1 \oplus f(101)\rangle$$
$$+ |110\rangle|f(110)\rangle - |110\rangle|1 \oplus f(110)\rangle - |111\rangle|f(111)\rangle + |111\rangle|1 \oplus f(111)\rangle).$$

We know, $\overline{x} = 1 \oplus x$ and $\overline{f}(xyz) = 1 \oplus f(xyz)$, so

$$\frac{1}{2\sqrt{2}}(|100\rangle|f(100)\rangle - |100\rangle|\overline{f}(100)\rangle + |101\rangle|f(101)\rangle - |101\rangle|\overline{f}(101)\rangle$$
$$+ |110\rangle|f(110)\rangle - |110\rangle|\overline{f}(110)\rangle - |111\rangle|f(111)\rangle + |111\rangle|\overline{f}(111)\rangle).$$

We can use the tabular form to confirm our assumptions.

$U_f(GM_1 \otimes H_1)(\|011\rangle\|1\rangle)$	00001000	00000100	00000010	00000001
$0 \cdot \|000\rangle$	0	0	0	0
$0 \cdot \|001\rangle$	0	0	0	0
$0 \cdot \|010\rangle$	0	0	0	0
$0 \cdot \|011\rangle$	0	0	0	0
$\|000\rangle\|f(000)\rangle - \|000\rangle\|\overline{f}(000)\rangle$	1	1	1	1
$\|001\rangle\|f(001)\rangle - \|001\rangle\|\overline{f}(001)\rangle$	-1	-1	1	1
$\|010\rangle\|f(010)\rangle - \|010\rangle\|\overline{f}(010)\rangle$	-1	1	-1	1
$-\|011\rangle\|f(011)\rangle + \|011\rangle\|\overline{f}(011)\rangle$	1	-1	-1	1
$GM_1 U_f(GM_1 \otimes H_1)(\|111\rangle\|1\rangle)$	$\|111\rangle$	$\|110\rangle$	$\|101\rangle$	$\|100\rangle$

Hence, part II can distinguish the remaining four functions. One thing we can notice is that the first four output values of the first four functions in the 3 qubit problem are same as the output values of the functions discriminated by the GregMatt gate in the 2 qubit problem. In this method, we have essentially divided the 3 qubit problem into two 2 qubit problems.

5.9 Deutsch's Problem with n qubits input

Our observations of the 2 qubit and the 3 qubit problem have given an impetus to extend Deutsch's Problem for the n qubit case. The Hadamard gate first appears in the original Deutsch's Problem. The Hadamard gate appeared again in the 2 in the form of $H \otimes H$ qubit and in the 3 qubit problems in the form of $H \otimes H \otimes H$. In each these cases, the Hadamard gate was able to discriminate functions which have half of the output values equal to 0 and the other half equal to 1. It is most likely the case that for the n qubit problem too, the Hardmard gate would appear in the form of $H^{\otimes n}$ and would discriminate the functions which have half of their output value equal to 0 and other half equal to 1.

However, what's more intriguing is the GregMatt gate and the functions it discriminates. For the 2 qubit problem, we observed that the GregMatt gate holds the information about the functions which have three out of four outputs equal to each other and successfully distinguishes each of them. However, when we tried to extend Deutsch's Problem for the 3 qubit input, in order to discriminate the functions which have seven out of eight output value equal to each other, we did something unique. We created a new entangling gate, the GregMatt1 gate by tensor product between Indentity matrix with the GregMatt gate($I \otimes GM$). By doing so we were able to divide the problem into two parts and for each part used different intial input states to achieve desired pure state to discriminate all 8 functions. We can extend this technique for the 4 qubit input problem and subsequently come with a general method for the n qubit case.

As usual, let's state the Deutsch's problem for 4 qubits input. Consider a function f that takes 4 bits input into a single bit,

$$U_f(|abcd\rangle|e\rangle) = |abcd\rangle|e \oplus f(abcd)\rangle. \tag{5.9.1}$$

For the 4 qubit input, there are essential 16 functions which have all but one values equal to each other. Let's list these function in the following table:

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

.

Here we used the GregMatt3 gate to entangled the initial input state. The GregMatt3 is simply $I \otimes GM_1$,

$$GM_2 = I \otimes GM_1 = \frac{1}{2} \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1
\end{pmatrix}.$$

Here we divide the problem into 4 parts where we use different initial states for the input register. Each part discrimintes 4 funcitons.

Part I

In part I, we discriminate the following four functions:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Here, we use $|0011\rangle$ as the initial state of the input register. Then, we entangle the input register with the GM_2 gate and output register with the Hadamard gate as usual. As a result we get a partially entangled state,

$$\begin{aligned}(GM_2 \otimes H_1)(|0011\rangle|1\rangle) =& \frac{1}{2}(|0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|0000\rangle|0\rangle - |0000\rangle|1\rangle + |0001\rangle|0\rangle - |0001\rangle|1\rangle \\ & + |0010\rangle|0\rangle - |0010\rangle|1\rangle - |0011\rangle|0\rangle + |0011\rangle|1\rangle).\end{aligned} \tag{5.9.2}$$

Once again, we will use tabular form to find the results:

$\mathrm{U}_f(GM_2 \otimes H_1)(\vert 0011\rangle\vert 1\rangle)$	1000000000000000	0100000000000000	0010000000000000	0001000000000000
$\vert 0000\rangle\vert f(0000)\rangle - \vert 000\rangle\vert \overline{f}(0000)\rangle$	1	1	1	1
$\vert 0001\rangle\vert f(0001)\rangle - \vert 001\rangle\vert \overline{f}(0001)\rangle$	−1	−1	1	1
$\vert 0010\rangle\vert f(0100)\rangle - \vert 010\rangle\vert \overline{f}(0010)\rangle$	−1	1	−1	1
$-\vert 0011\rangle\vert f(0011)\rangle + \vert 011\rangle\vert \overline{f}(0011)\rangle$	1	−1	−1	1
$0 \cdot \vert 0100\rangle$	0	0	0	0
$0 \cdot \vert 0101\rangle$	0	0	0	0
$0 \cdot \vert 0110\rangle$	0	0	0	0
$0 \cdot \vert 0111\rangle$	0	0	0	0
$0 \cdot \vert 1000\rangle$	0	0	0	0
$0 \cdot \vert 1001\rangle$	0	0	0	0
$0 \cdot \vert 1010\rangle$	0	0	0	0
$0 \cdot \vert 1011\rangle$	0	0	0	0
$0 \cdot \vert 1100\rangle$	0	0	0	0
$0 \cdot \vert 1101\rangle$	0	0	0	0
$0 \cdot \vert 1110\rangle$	0	0	0	0
$0 \cdot \vert 1111\rangle$	0	0	0	0
$GM_2\ \mathrm{U}_f(GM_2 \otimes H_1)(\vert 0011\rangle\vert 1\rangle)$	$\vert 0011\rangle$	$\vert 0010\rangle$	$\vert 0001\rangle$	$\vert 0000\rangle$

Hence, part I discriminates the first four functions.

Part II

In part II, we discriminate the following four functions:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Here, we use $|0111\rangle$ as the initial state of the input register. Then, we entangle the input register with the GM_2 gate and output register with the Hadamard gate as usual. As a result we get a partially entangled state,

$$\begin{aligned}(GM_2 \otimes H_1)(|0111\rangle|1\rangle) =& \frac{1}{2}(|0100\rangle + |0101\rangle + |0110\rangle - |0111\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|0100\rangle|0\rangle - |0100\rangle|1\rangle + |0101\rangle|0\rangle - |0101\rangle|1\rangle \\ &+ |0110\rangle|0\rangle - |0110\rangle|1\rangle - |0111\rangle|0\rangle + |0111\rangle|1\rangle).\end{aligned} \tag{5.9.3}$$

Once again, we will use tabular form to find the results:

$\mathrm{U}_f(GM_2 \otimes H_1)(\|0111\rangle\|1\rangle)$	0000100000000000	0000010000000000	0000001000000000	0000000100000000
$0 \cdot \|0000\rangle$	0	0	0	0
$0 \cdot \|0001\rangle$	0	0	0	0
$0 \cdot \|0010\rangle$	0	0	0	0
$0 \cdot \|0011\rangle$	0	0	0	0
$\|0100\rangle\|f(0100)\rangle - \|0100\rangle\|\overline{f}(0100)\rangle$	1	1	1	1
$\|0101\rangle\|f(0101)\rangle - \|0101\rangle\|\overline{f}(0101)\rangle$	-1	-1	1	1
$\|0110\rangle\|f(0110)\rangle - \|0110\rangle\|\overline{f}(0110)\rangle$	-1	1	-1	1
$-\|0111\rangle\|f(0111)\rangle + \|0111\rangle\|\overline{f}(0111)\rangle$	1	-1	-1	1
$0 \cdot \|1000\rangle$	0	0	0	0
$0 \cdot \|1001\rangle$	0	0	0	0
$0 \cdot \|1010\rangle$	0	0	0	0
$0 \cdot \|1011\rangle$	0	0	0	0
$0 \cdot \|1100\rangle$	0	0	0	0
$0 \cdot \|1101\rangle$	0	0	0	0
$0 \cdot \|1110\rangle$	0	0	0	0
$0 \cdot \|1111\rangle$	0	0	0	0
$GM_2\ \mathrm{U}_f(GM_2 \otimes H_1)(\|0111\rangle\|1\rangle)$	$\|0111\rangle$	$\|0110\rangle$	$\|0101\rangle$	$\|0100\rangle$

Hence, part II discriminates the second four functions.

Part III

In part III, we discriminate the following four functions:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Here, we use $|1011\rangle$ as the initial state of the input register. Then, we entangle the input register with the GM_2 gate and output register with the Hadamard gate as usual. As a result we get a partially entangled state,

$$
\begin{aligned}
(GM_2 \otimes H_1)(|1011\rangle|1\rangle) =& \frac{1}{2}(|1000\rangle + |1001\rangle + |1010\rangle - |1011\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
=& \frac{1}{2\sqrt{2}}(|1000\rangle|0\rangle - |1000\rangle|1\rangle + |1001\rangle|0\rangle - |1001\rangle|1\rangle \\
& + |1010\rangle|0\rangle - |1010\rangle|1\rangle - |1011\rangle|0\rangle + |1011\rangle|1\rangle).
\end{aligned} \tag{5.9.4}
$$

Once again, we will use tabular form to find the results:

$\mathrm{U}_f(GM2 \otimes H_1)(\|1011\rangle\|1\rangle)$	0000000010000000	000000000100000	0000000000100000	0000000000010000
$0 \cdot \|0000\rangle$	0	0	0	0
$0 \cdot \|0001\rangle$	0	0	0	0
$0 \cdot \|0010\rangle$	0	0	0	0
$0 \cdot \|0011\rangle$	0	0	0	0
$0 \cdot \|0100\rangle$	0	0	0	0
$0 \cdot \|0101\rangle$	0	0	0	0
$0 \cdot \|0110\rangle$	0	0	0	0
$0 \cdot \|0111\rangle$	0	0	0	0
$\|1000\rangle\|f(1000)\rangle - \|1000\rangle\|\overline{f}(1000)\rangle$	1	1	1	1
$\|1001\rangle\|f(1001)\rangle - \|1001\rangle\|\overline{f}(1001)\rangle$	-1	-1	1	1
$\|1010\rangle\|f(1010)\rangle - \|1010\rangle\|\overline{f}(1010)\rangle$	-1	1	-1	1
$-\|1011\rangle\|f(1011)\rangle + \|1011\rangle\|\overline{f}(1011)\rangle$	1	-1	-1	1
$0 \cdot \|1100\rangle$	0	0	0	0
$0 \cdot \|1101\rangle$	0	0	0	0
$0 \cdot \|1110\rangle$	0	0	0	0
$0 \cdot \|1111\rangle$	0	0	0	0
$GM_2\ \mathrm{U}_f(GM_2 \otimes H_1)(\|1011\rangle\|1\rangle)$	$\|1011\rangle$	$\|1010\rangle$	$\|1001\rangle$	$\|1000\rangle$

Hence, part III discriminates the third four functions.

Part IV

In part IV, we discriminate the following four functions:

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

Here, we use $|1011\rangle$ as the initial state of the input register. Then, we entangle the input register with the GM_2 gate and output register with the Hadamard gate as usual. As a result we get a partially entangled state,

$$
\begin{aligned}
(GM_2 \otimes H_1)(|1111\rangle|1\rangle) =& \frac{1}{2}(|1100\rangle + |1101\rangle + |1110\rangle - |1111\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
=& \frac{1}{2\sqrt{2}}(|1100\rangle|0\rangle - |1100\rangle|1\rangle + |1101\rangle|0\rangle - |1101\rangle|1\rangle \\
& + |1110\rangle|0\rangle - |1110\rangle|1\rangle - |1111\rangle|0\rangle + |1111\rangle|1\rangle).
\end{aligned} \tag{5.9.5}
$$

Once again, we will use tabular form to find the results:

$\text{U}_f(GM2 \otimes H_1)(\|1111\rangle\|1\rangle)$	0000000000001000	0000000000000100	0000000000000010	0000000000000001
$0 \cdot \|0000\rangle$	0	0	0	0
$0 \cdot \|0001\rangle$	0	0	0	0
$0 \cdot \|0010\rangle$	0	0	0	0
$0 \cdot \|0011\rangle$	0	0	0	0
$0 \cdot \|0100\rangle$	0	0	0	0
$0 \cdot \|0101\rangle$	0	0	0	0
$0 \cdot \|0110\rangle$	0	0	0	0
$0 \cdot \|0111\rangle$	0	0	0	0
$0 \cdot \|1000\rangle$	0	0	0	0
$0 \cdot \|1001\rangle$	0	0	0	0
$0 \cdot \|1010\rangle$	0	0	0	0
$0 \cdot \|1011\rangle$	0	0	0	0
$\|1100\rangle\|f(1100)\rangle - \|1100\rangle\|\overline{f}(1100)\rangle$	1	1	1	1
$\|1101\rangle\|f(1101)\rangle - \|1101\rangle\|\overline{f}(1101)\rangle$	-1	1	-1	1
$\|1110\rangle\|f(1110)\rangle - \|1110\rangle\|\overline{f}(110)\rangle$	-1	1	-1	1
$-\|1111\rangle\|f(1111)\rangle + \|1111\rangle\|\overline{f}(1111)\rangle$	1	-1	-1	1
$GM_2\ \text{U}_f(GM_2 \otimes H_1)(\|1111\rangle\|1\rangle)$	$\|1111\rangle$	$\|1110\rangle$	$\|1101\rangle$	$\|1100\rangle$

Hence, part I discriminates the last four functions.

From above observation for the 3 qubit and the 4 qubit problem, we can generalized the method of discriminating functions which have one but all values equal to each other.

Definition 5.9.1. The **GregMatt-k gate** is represented as GM_k and defined as tensor product of $k \times k$ identity matrix and the GregMatt gate,

$$GM_k = I_k \otimes GM \text{ where,}$$

$$I_k \text{ is } k \times k \text{ identity matrix and}$$

$$GM_0 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix}.$$

The matrix representation of GM_k is a $2^{k+2} \times 2^{k+2}$ matrix,

$$GM_k = \begin{bmatrix} GM_0 & 0 & ... & 0 \\ 0 & GM_0 & ... & 0 \\ . & . & ... & . \\ 0 & 0 & ... & GM_0 \end{bmatrix}.$$

Spaced Input Method

Spaced Input Method is the process of discriminating the functions which have all but one value equal to each other for the n qubit Deutsch's Problem. In this method, we take every fourth computational basis state and divide the functions into group of four. We use each of these selected basis state as the state of the input register to discriminate a group of four functions in the same order. For example in the 3 qubit problem, there are all together 8 functions which have all but one function value equal to each other. Using the spaced input method, we divide the functions into 2 groups each consisting of 4 functions and for first group use $|011\rangle$ as the state of input register and the second group use $|111\rangle$ as the state of the input register.

Theorem 5.9.2. *For $n \geq 2$ and $k = n-2$, in the n qubit Deutsch's problem, GM_k can discriminate the functions that have all but one value equal to each other with spaced input method.*

Proof. For the n qubit problem, there are 2^n functions that have all but one value equal to each other. Following the spaced input method, we divide the functions into 2^{n-2} groups with each containing 4 functions. For each group we use different computational basis state as the state of input register.

Let's choose an arbitary group of functions and suppose the state of input register is $|m\rangle$ where m is natural number representation of the computational basis state. The group of functions will have the following output values where superscript denotes the position of 1 in the sequence of output values

$$\begin{bmatrix} 0 & 0 & 0 & 0 & \cdot\ \cdot\ \cdot\ \cdot\ \cdot & 1^{(m-3)} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot\ \cdot\ \cdot\ \cdot & 0 & 1^{(m-2)} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot\ \cdot\ \cdot\ \cdot & 0 & 0 & 1^{(m-1)} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot\ \cdot\ \cdot\ \cdot & 0 & 0 & 0 & 1^{(m)} & \cdots & 0 & 0 & 0 & 0 \end{bmatrix}$$

Using Deutsch's algorithm, first entanlge the input register using the GM_k gate and the Hadamard gate to entangle the output register,

$$\begin{aligned} (GM_k \otimes H_1)(|m\rangle|1\rangle) =& \frac{1}{2}(|m-3\rangle + |m-2\rangle + |m-1\rangle - |m\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ =& \frac{1}{2\sqrt{2}}(|m-3\rangle|0\rangle - |m-3\rangle|1\rangle + |m-2\rangle|0\rangle - |m-2\rangle|1\rangle \\ &+ |m-1\rangle|0\rangle - |m-1\rangle|1\rangle - |m\rangle|0\rangle + |m\rangle|1\rangle). \end{aligned}$$

Once we have the entangled state, we apply U_f transformation which gives us,

$$\frac{1}{2\sqrt{2}}(|m-3\rangle|f(m-3)\rangle - |m-3\rangle|\overline{f}(m-3)\rangle + |m-2\rangle|f(m-2)\rangle - |m-2\rangle|\overline{f}(m-2)\rangle$$
$$+ |m-1\rangle|f(m-1)\rangle - |m-1\rangle|\overline{f}(m-1)\rangle - |m\rangle|f(m)\rangle + |m\rangle|\overline{f}(m)\rangle).$$

$$\text{Condition I, } \overline{f}(m-3) = f(m-2) = f(m-1) = f(m)$$
$$f(m-3) = \overline{f}(m-2) = \overline{f}(m-1) = \overline{f}(m)$$

$$\frac{1}{2\sqrt{2}}(|m-3\rangle - |m-2\rangle - |m-1\rangle + |m\rangle)(|f(m-3)\rangle - |\overline{f}(m-3)\rangle). \tag{5.9.6}$$

$$\text{Condition II, } f(m-3) = \overline{f}(m-2) = f(m-1) = f(m)$$
$$\overline{f}(m-3) = f(m-2) = \overline{f}(m-1) = \overline{f}(m)$$

$$\frac{1}{2\sqrt{2}}(|m-3\rangle - |m-2\rangle + |m-1\rangle - |m\rangle)(|f(m-3)\rangle - |\overline{f}(m-3)\rangle). \tag{5.9.7}$$

$$\text{Condition III, } f(m-3) = f(m-2) = \overline{f}(m-1) = f(m)$$
$$\overline{f}(m-3) = \overline{f}(m-2) = f(m-1) = \overline{f}(m)$$

$$\frac{1}{2\sqrt{2}}(|m-3\rangle + |m-2\rangle - |m-1\rangle - |m\rangle)(|f(m-3)\rangle - |\overline{f}(m-3)\rangle). \tag{5.9.8}$$

$$\text{Condition IV, } f(m-3) = f(m-2) = f(m-1) = \overline{f}(m)$$
$$\overline{f}(m-3) = \overline{f}(m-2) = \overline{f}(m-1) = f(m)$$

$$\frac{1}{2\sqrt{2}}(|m-3\rangle + |m-2\rangle + |m-1\rangle + |m\rangle)(|f(m-3)\rangle - |\overline{f}(m-3)\rangle). \tag{5.9.9}$$

On applying GM_k back on the input register for each condition we obtain:

$$|m\rangle\frac{1}{\sqrt{2}}(|f(m-3)\rangle - |\overline{f}(m-3)\rangle), \text{ when } f(m-3) \neq f(m-2) = f(m-1) = f(m), \tag{5.9.10}$$

$$|m-1\rangle\frac{1}{\sqrt{2}}(|f(m-3)\rangle - |\overline{f}(m-3)\rangle), \text{ when } f(m-3) = f(m-1) = f(m) \neq f(m-2), \tag{5.9.11}$$

$$|m-2\rangle\frac{1}{\sqrt{2}}(|f(m-3)\rangle - |\overline{f}(m-3)\rangle), \text{ when } f(m-3) = f(m-2) = f(m) \neq f(m-1), \tag{5.9.12}$$

$$|m-3\rangle\frac{1}{\sqrt{2}}(|f(m-3)\rangle - |\overline{f}(m-3)\rangle), \text{ when } f(m-3) = f(m-2) = f(m-1) \neq f(m). \tag{5.9.13}$$

Hence, we get a distinct pure state for each function. Therefore GM_k can successfully discriminate the functions.

5.10 Conclusion

The extensions of Deutsch's problem for the 2 qubit and the 3 qubit input have given us some insightful results. One of the important results of the reseach is the discovery of the GregMatt gate for discriminating functions that have all but one value equal to each other in the 2 qubit problem. This led us to find the GregMatt gate for the 3 and the 4 qubit problem. This along with the Spaced Input Method equipped us with a new technique of discriminating functions which have all but one value equal to each other for generalized n qubit problem. For the 2 qubit and the 3 qubit problems we have noticed that the Hadamard gate successfully discriminates functions that have half of the output values equal to each other. Interestingly, the functions that the Hadamard gate discriminates form a abelian group under mod 2 addition. We have noticed that the functions which have half of the output values equal to each other and which also form a group under mod 2 addition can be discriminated by Hadamard-like gate, whose entries have direct correspondence with the values of the functions. We have also noticed that such group has surprising relation to classical error correcting codes.

Although we have solved the 2 qubit problem, we have not completely solved the 3 qubit problem. In the 3 qubit problem we successfully found ways to dicriminate functions which have 7 out of 8 output values equal to each other and functions which have 4 out of 8 values equal to each other. This leaves us with two kinds of functions, ones that have 3 output values equal each other to each other and 5 output values equal to each other and others that have 2 output values equal to each other and 6 output values each other each other. Since this research project has to be completed in two semesters, there is not enough time to investigate the 3 qubit problem in its entirety. Exploring these functions and ways of discriminating them can be an interesting question for future research.

Bibliography

[1] Colin P. Williams, *Explorations in Quantum Computing*, Spinger, New York, 2010.

[2] Michael A. Nielsen and Issac L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, New York, 2010.

[3] Steven Roman, *Advanced Linear Algebra*, Spinger, New York, 2005.

[4] N. David Mermin, *Quantum Computer Science*, Cambridge University Press, Cambridge, 2007.

[5] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek, *Perfect Quantum Error Correcting Code*, Physical Review Letters **7** (1996), 198–201.

[6] Daniel Gottesman, *Stabilizer Codes and Quantum Error Correction*, arXiv:quant-ph/9705052v1 (1997).

[7] Gordon E. Moore, *Cramming More Component onto Integrated Circuits*, Electronics **38** (1965).

[8] Richard Feynman, *Simulating Physics with Computers*, Optics News **11** (1982), 467-488.

[9] David Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computers*, Proceeding of the Royal Society of London **A400** (1985), 97-117.