2016

Computational Progress towards Maximum Distinguishability of Bell States by Linear Evolution and Local Measurement

Victor Shang
Harvey Mudd College

Computational Progress towards Maximum Distinguishability of Bell States by Linear Evolution and Local Measurement

Victor Shang

Theresa W. Lynn, Advisor

Andrew J. Bernoff, Reader

Department of Physics

May, 2016

Abstract

Many quantum information protocols rely on the ability to distinguish between entangled quantum states known as Bell states. However, theoretical limits exist on the maximal distinguishability of these entangled states using linear evolution and local measurement (LELM) devices. In the case of two particles entangled in multiple qubit variables, the maximum number of distinguishable Bell states is known. However, in the more general case of two particles entangled in multiple qudit variables, only an upper bound is known under additional assumptions. I have written software in Matlab and Mathematica to explore computationally the maximum number of Bell states that can be distinguished in the case of two particles entangled in a qutrit variable, and the case of two particles entangled in both a qutrit and qubit variable. Using code I have written in Mathematica, I have reduced the number of cases to check for sets of 9 qubit⊗qutrit Bell states from $94,143,280$ to $10,365$. Further work needs to be done to computationally check these cases for distinguishability by an LELM apparatus.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Professor Theresa Lynn for her assistance and guidance in advising my undergraduate research. I would also like to acknowledge Lucas Brady and Julien Devin, whose previous work contributed greatly to my thesis work. Finally, I would like to thank family and friends for their unconditional support throughout my time at Harvey Mudd. I never could have done this without them.

Chapter 1

Introduction

One of the key differences between quantum physics and classical physics is the existence of situations where a group of particles is in a perfectly defined state, but the individual states of each particle are undefined. This phenomenon, known as *entanglement*, has applications to many quantum information protocols, such as teleportation [1], dense coding [2, 3], quantum repeaters [4], and entanglement swapping [5**?**]. These protocols take advantage of the properties of quantum states to encode information more securely and efficiently than can be achieved using classical methods. A key component of each of these protocols is the ability to distinguish between entangled states. Thus, limitations on the number of entangled states that can be distinguished also limit the effectiveness of these quantum information protocols.

Of particular interest is the measurement of maximally entangled states known as *Bell states* using only linear evolution and local measurement (LELM) devices since these devices are simple to construct and are reliable at detection compared to non-linear devices. Linear evolution means that the evolution of each particle does not depend on any other particle, and local measurement means that detection of each particle is registered as a "click" that occurs locally at a detector in the device. Previous theoretical upper bounds have been established [6, 7]. However, for some of these results it is not known whether the upper bounds are achievable.

This thesis describes work to explore the achievable upper limit on the maximum number of Bell states that can be distinguished using LELM devices in the case of two particles entangled in either a two-state variable, a three-state variable, or both a two-state and three-state variable. Most of the work involves computational approaches using code written in Matlab

and Mathematica. This work builds heavily upon the computational work of Julien Devin [8] and Lucas Brady [9] who worked on similar projects for their summer research. Chapter 2 will describe the mathematical foundation of entanglement as well as previous results on maximal Bell state distinguishability. Chapter 3 will explain the concept of equivalence classes and how this mathematical idea can be applied to make computational approaches to determining maximal Bell state distinguishability more efficient. Chapter 4 will focus on attempts to computationally determine the number of maximal Bell states that can be distinguished for two particles entangled in a three-state variable. Chapter 5 will conclude the thesis by summarizing the work done in my research and the current state of the problem of maximal Bell state distinguishability.

Chapter 2

Background

2.1 Introduction

First, we shall provide an overview of the mathematics involved in entangled states, as well as definitions of Bell states for two particles entangled in multi-state variables. We will also discuss previous theoretical results on the maximal distinguishability of Bell states using an LELM apparatus for n qubit and qudit variables, as well as the mathematical formalism behind the distinguishability conditions that allows us to check whether a given set of Bell states is distinguishable.

2.2 Quantum States

In quantum mechanics, information about particles can be encoded in vectors known as *quantum states*. These states are usually represented in braket notation, where a ket such as $|\phi\rangle$ is a vector signifying a quantum state ϕ while the corresponding bra $\langle\phi|$ is the co-vector to the vector $|\phi\rangle$. The magnitude squared of the inner product $\langle\psi|\,\phi\rangle$ between a bra and a ket denotes the probability that a particle in the state $|\phi\rangle$ will be measured in the state $|\psi\rangle$. The inner product $\langle\psi|\,\phi\rangle$ is referred to as the *probability amplitude*. The vector space generated by the ket vectors forms a Hilbert space whose dimension is the number of mutually orthogonal states in a particular variable. These mutually orthogonal states form a basis for the Hilbert space that spans the entire space of states for the particle. For example, in the case of a two-variable system whose basis states are the spin up and down states $\{|\uparrow\rangle, |\downarrow\rangle\}$ of spin angular momentum, any state $|\phi\rangle$ in the system can

be described as

$$|\phi\rangle = a\,|\uparrow\rangle + b\,|\downarrow\rangle\,, \tag{2.1}$$

where a and b are complex coefficients whose amplitude squared signifies the probability of a particle in the state $|\phi\rangle$ to be in the state $|\uparrow\rangle$ or $|\downarrow\rangle$, respectively. All quantum states must be normalized so that the total probability of being in any single state is 1, which can be described in this example by the condition that $|a|^2 + |b|^2 = 1$.

These states can also be represented in matrix notation in a suitable basis, where quantum states are denoted by column vectors and their co-vectors are denoted by row vectors corresponding to the conjugate transpose of the column vector. For example, the vector $|\phi\rangle$ in Equation 2.1 can also be written as the column vector

$$|\phi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}, \tag{2.2}$$

where the basis is given by $\{|\uparrow\rangle\,, |\downarrow\rangle\}$, and the corresponding co-vector is given by the row vector

$$\langle\phi| = \begin{pmatrix} a^* & b^* \end{pmatrix}, \tag{2.3}$$

where a^* and b^* are the complex conjugate to a and b, respectively. The inner product in matrix notation is just given by matrix multiplication between column and row vectors.

The quantum states we have just been exposed to are single particle states of one variable. To write quantum states of multiple particles, we take the direct product of the basis states of each particle in order to determine the joint-particle basis states of the multi-particle Hilbert space. For example, if we were to write the state of two particles $|\psi\rangle$, both of whose basis states are the spin up and down states, the corresponding basis is given by $\{|\uparrow\rangle|\uparrow\rangle\,, |\uparrow\rangle|\downarrow\rangle\,, |\downarrow\rangle|\uparrow\rangle\,, |\downarrow\rangle|\downarrow\rangle\}$, which we shall denote as the *joint-particle basis*. The most general two-particle quantum state in braket notation can be written as

$$|\psi\rangle = a\,|\uparrow\rangle_L|\uparrow\rangle_R + b\,|\uparrow\rangle_L|\downarrow\rangle_R + c\,|\downarrow\rangle_L|\uparrow\rangle_R + d\,|\downarrow\rangle_L|\downarrow\rangle_R\,, \tag{2.4}$$

where $a, b, c, d \in \mathbb{C}$ and $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$. The subscripts L and R denote whether the state belongs to the left or right particle. The corresponding column vector in the basis $\{|\uparrow\rangle_L|\uparrow\rangle_R\,, |\uparrow\rangle_L|\downarrow\rangle_R\,, |\downarrow\rangle_L|\uparrow\rangle_R\,, |\downarrow\rangle_L|\downarrow\rangle_R\}$

is given by

$$|\psi\rangle = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}. \tag{2.5}$$

Entanglement between two particles occurs when a joint-particle state cannot be written as the direct product of two single-particle states. These states are called *entangled states*. Entanglement can also occur in more than two particles, but for our purposes it is sufficient to restrict entanglement to 2−particle systems. For example, consider the state

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|\uparrow\rangle_L |\uparrow\rangle_R + |\uparrow\rangle_L |\downarrow\rangle_R \right). \tag{2.6}$$

We can rewrite this state as

$$|\psi\rangle = |\uparrow\rangle_L \otimes \frac{1}{\sqrt{2}} \left(|\uparrow\rangle_R + |\downarrow\rangle_R \right), \tag{2.7}$$

which is a direct product of two single-particle states. Thus, the state in Equation 2.6 is not an entangled state. However, the state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} \left(|\uparrow\rangle_L |\uparrow\rangle_R + |\downarrow\rangle_L |\downarrow\rangle_R \right) \tag{2.8}$$

cannot be separated into the direct product of two single-particle states, so $|\Phi^+\rangle$ is an entangled state.

We can extend our description of quantum states to states in more than one variable, which involves taking a separate direct product for each of the basis states for each variable. To make the direct product between variables distinct from the direct product between separate particles, we usually denote a multiple variable state for a single particle by a linear combination of vectors of the form $|\phi_1, \phi_2, \ldots, \phi_n\rangle$, where each ϕ_i denotes a basis state for the ith variable. *Hyper-entangled states* are 2−particle states that are entangled in multiple variables.

For example, consider two particles entangled in both a two-state spin up and down variable and a three-state orbital angular momentum variable whose basis is given by the states $\{|-\hbar\rangle, |0\rangle, |\hbar\rangle\}$. An example of a hyper-entangled state ϕ between these two particles is given by

$$|\phi\rangle = a\, |\uparrow, \hbar\rangle_L |\downarrow, -\hbar\rangle_R + b\, |\downarrow, 0\rangle_L |\uparrow, \hbar\rangle_R, \tag{2.9}$$

where $a, b \in \mathbb{C}$ and $|a|^2 + |b|^2 = 1$.

2.3 Qubit and Qudit Bell States

A *qudit* is a quantum system with d basis states. These states are usually eigenstates of an observable property such as spin or orbital angular momentum. For example, a qubit variable has two possible eigenstates while a qutrit variable has three possible eigenstates. Bell states are special quantum states that are formed by entangling two or more particles in one or more qudit variables. We shall focus on Bell states for two particles entangled in either a qubit or qutrit variable. Since each two-particle quantum state can be written as a superposition of Bell states, these Bell states form an orthonormal basis that spans the entire Hilbert space for the two-particle quantum states. Hence, working in either the joint-particle basis or the Bell-state basis is just a matter of convenience.

For the qubit case, where the basis states are given by $\{|0\rangle, |1\rangle\}$, the canonical Bell states in Dirac notation are given by the four vectors

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_L |0\rangle_R + |1\rangle_L |1\rangle_R\right) \tag{2.10a}$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_L |0\rangle_R - |1\rangle_L |1\rangle_R\right) \tag{2.10b}$$

$$\left|\Psi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_L |1\rangle_R + |1\rangle_L |0\rangle_R\right) \tag{2.10c}$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_L |1\rangle_R - |1\rangle_L |0\rangle_R\right), \tag{2.10d}$$

where L and R refer to the left and right particles, respectively. We shall refer to these Bell states as *qubit Bell states*. The symbols Φ and Ψ distinguish between the correlation of the two particles in the $\{|0\rangle, |1\rangle\}$ basis, and the superscripts $+$ and $-$ distinguish the relative phase between the first and second term of each Bell state.

For the qutrit case, where the basis states are given by $\{|0\rangle, |1\rangle, |2\rangle\}$, we

use the following entangled basis of Bell states given in [10]:

$$|\psi_{00}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|0\rangle_R + |1\rangle_L|1\rangle_R + |2\rangle_L|2\rangle_R\right) \tag{2.11a}$$

$$|\psi_{01}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|0\rangle_R + e^{2\pi i/3}|1\rangle_L|1\rangle_R + e^{-2\pi i/3}|2\rangle_L|2\rangle_R\right) \tag{2.11b}$$

$$|\psi_{02}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|0\rangle_R + e^{-2\pi i/3}|1\rangle_L|1\rangle_R + e^{2\pi i/3}|2\rangle_L|2\rangle_R\right) \tag{2.11c}$$

$$|\psi_{10}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|1\rangle_R + |1\rangle_L|2\rangle_R + |2\rangle_L|0\rangle_R\right) \tag{2.11d}$$

$$|\psi_{11}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|1\rangle_R + e^{2\pi i/3}|1\rangle_L|2\rangle_R + e^{-2\pi i/3}|2\rangle_L|0\rangle_R\right) \tag{2.11e}$$

$$|\psi_{12}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|1\rangle_R + e^{-2\pi i/3}|1\rangle_L|2\rangle_R + e^{2\pi i/3}|2\rangle_L|0\rangle_R\right) \tag{2.11f}$$

$$|\psi_{20}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|2\rangle_R + |1\rangle_L|0\rangle_R + |2\rangle_L|1\rangle_R\right) \tag{2.11g}$$

$$|\psi_{21}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|2\rangle_R + e^{2\pi i/3}|1\rangle_L|0\rangle_R + e^{-2\pi i/3}|2\rangle_L|1\rangle_R\right) \tag{2.11h}$$

$$|\psi_{22}\rangle = \frac{1}{\sqrt{3}}\left(|0\rangle_L|2\rangle_R + e^{-2\pi i/3}|1\rangle_L|0\rangle_R + e^{2\pi i/3}|2\rangle_L|1\rangle_R\right). \tag{2.11i}$$

We shall refer to these Bell states as *qutrit Bell states*. The first index of each qutrit Bell states distinguishes the correlation between the three particles in the $\{|0\rangle, |1\rangle, |2\rangle\}$ basis, while the second index distinguishes the relative phases between the joint-particle states. Thus, we can divide the qutrit Bell states into the three classes $\{|\psi_{0i}\rangle\}, \{|\psi_{1i}\rangle\}, \{|\psi_{2i}\rangle\}$ based on their correlation properties, or we can divide them into the three classes $\{|\psi_{i0}\rangle\}, \{|\psi_{i1}\rangle\}, \{|\psi_{i2}\rangle\}$ based on the relative phases between the joint-particle states.

For two particles hyper-entangled in a qubit and qutrit variable, such as spin and orbital angular momentum, an entangled basis can simply be generated by taking the tensor product of each of the qubit Bell states $|\Psi_i\rangle$ with the qutrit Bell states $|\psi_j\rangle$, yielding the state $|\Psi_i\rangle \otimes |\psi_j\rangle$. Since there are four qubit Bell states and nine qutrit Bell states, this tensor product results in a total of $4\times9 = 36$ hyper-entangled Bell states for two particles entangled in both a qubit and qutrit variable, which we shall denote as qubit⊗qutrit Bell states.

Case	Maximum Number of Distinguishability Classes
$D \equiv 0 \pmod 2$	$2D - 1$
$D \equiv 1 \pmod 2$, fermions	$2D - 2$
$D \equiv 1 \pmod 2$, bosons	$2D - 3$
$D = 3$, bosons	4

Table 2.1 Upper bounds on Bell state distinguishability for two particles entangled in n qudit variables, assuming disjoint detection signatures. D is the number of possible single particle input states for each particle.

2.4 Previous Results

Neal Pisenti [6, 11] showed that for two particles entangled in n qubit variables, it is only possible to distinguish at most $2^{n+1} - 1$ of the 4^n possible Bell states using an LELM apparatus. Neal and Philip Gaebler also showed that this upper bound is achievable [11], proving that the maximal distinguishability of Bell states for n qubit variables is exactly $2^{n+1} - 1$.

For hyperentanglement between two particles in n variables with $d_1, \ldots, d_n$ states, respectively, more recent group work by Andrew Turner [7] has established upper bounds for hyper-entangled Bell state distinguishability using an LELM apparatus under the assumption that certain projections called detection signatures are disjoint. However, we do not know if these upper bounds are achievable, and we also do not know whether the assumption of disjoint detection signatures holds. Thus, the exact upper bound for the case of n qudits is unknown. The results are given in Table 2.1, where $D = d_1 d_2 \cdots d_n$.

In particular, the maximum number of distinguishable qutrit and qubit⊗qutrit Bell states is unknown. Since previous experimental work on implementing detection schemes using LELM devices has been done for qubit and qutrit Bell states[12, 13, 14, 15], determining the maximum number of distinguishable qutrit and qubit⊗qutrit Bell states would improve our understanding of the fundamental limits to current quantum information protocols. As a result, this thesis will describe computational work towards determining the achievable limit in Bell state distinguishability for the qutrit and qubit⊗qutrit case.

2.5 Conditions for Distinguishability

One approach to determining the actual maximum number of distinguishable Bell states is to systematically check all possible $k-$subsets of the Bell states computationally to see if they are distinguishable based on the following conditions outlined in [16]: let the vector $\vec{a} = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_d)^T$ represent the annihilation operators of the single-particle input modes, and let the vector $\vec{c} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_d)^T = U\vec{a}$ represent the annihilation operators of the detector modes of the LELM apparatus, where U is a $d \times d$ unitary matrix. In order for a given set of Bell states $\{|\psi_1\rangle, \ldots, |\psi_n\rangle\}$ to be distinguishable, the left-over states after the first detection by the LELM apparatus must remain orthogonal. Thus, for a system of two particles the set of Bell states $|\psi_i\rangle$ are distinguishable if and only if the following condition is satisfied: [16]

$$\langle\psi_k| \hat{c}^\dagger \hat{c} |\psi_l\rangle = 0 \qquad \forall k \neq l, \forall j = 1, \ldots, d \tag{2.12}$$

In other words, a set of Bell states is distinguishable by an LELM apparatus if and only if there exists a $d \times d$ unitary matrix whose coefficients satisfy Equation 2.12. If we treat the coefficients of this $d \times d$ unitary matrix as free variables, then Equation 2.12 generates a system of equations in these variables that has a complex solution if and only if the set of Bell states that generated these equations are distinguishable using an LELM apparatus. Since we can search for solutions to a system of equations using computational methods, we can use Equation 2.12 to computationally determine whether a set of Bell states is distinguishable by an LELM apparatus.

This set of conditions turns out to be sufficient for distinguishability of the Bell states $|\psi_i\rangle$. In particular, if the system of equations for a particular $\hat{c}_j$ has no solutions, no first detector 'click' will leave the Bell states orthogonal, so the Bell states $|\psi_i\rangle$ cannot be completely distinguished. Hence, the equation above must be satisfied for a particular detector $\hat{c}_j$ as a necessary condition. Julien [8] and Lucas [9] implemented an algorithm in Mathematica developed in [17] based which imposes this necessary condition on the distinguishability of a given set of Bell states. My work expands on the work of Lucas and Julien by improving the efficiency of the code as well as implementing the full set of conditions outlined in [16].

Chapter 3

Equivalence Classes

3.1 Introduction

For a pair of entangled particles with a total of n Bell states, there are $\binom{n}{k}$ sets of k Bell states to check for distinguishability. As n increases, checking all possible $k-$subsets quickly becomes computationally unfeasible. To reduce the number of cases to check, Julien Devin and Lucas Brady used the concept of *equivalence classes*. This mathematical concept allows us to define certain sets of Bell states as 'equivalent' to each other, which means that the sets of Bell states are distinguishable if and only if each of the other equivalence sets of Bell states are also distinguishable. By defining an equivalence relation between sets of Bell states, we can then simplify the problem of checking each set of Bell states to checking one representative set of Bell states from each equivalence class that we have defined. This method turns out to reduce the number of cases to check drastically in the qutrit and qubit⊗qutrit case.

3.2 Definition

An *equivalence relation* $\sim$ on a set X is a binary operation that is reflexive, symmetric, and transitive. In other words, for all $a, b, c \in X$, we have $a \sim a$, $a \sim b$ if and only if $b \sim a$, and if $a \sim b$ and $b \sim c$, then $a \sim c$. One of the most common equivalence relations is the '=' relation, which satisfies all of the properties above. The set of elements $A \subset X$ that are all equivalent to an element $a \in X$ is called an *equivalence class*, denoted by $[a]$. For example, under the relation '=' in the set of real numbers, the elements 4 and 2^2 fall

under the same equivalence class [4]. Thus, elements of a single equivalence class are identical in the sense that they can be treated as the same element when considering operations on the element and their properties, such as the fact that both 4 and 2^2 are even numbers.

In the case of Bell states, since any unitary operator on a single particle's state can be realized using linear optics [18], we can compose any LELM apparatus with single-particle unitary operators on the left and/or right particles to create another LELM apparatus, as in Figure 3.1. Thus, if a set of Bell states is distinguishable using an LELM apparatus, then any set of states resulting from a unitary transformation on the left and/or right particles must be distinguishable using an LELM apparatus constructed by composing the original distinguishing apparatus with the unitary operators. Since all unitary matrices are invertible, any set of Bell states that can be transformed into a set of distinguishable Bell states by left and/or right particle unitary transformations is also distinguishable using an LELM apparatus.

These properties allow us to define an equivalence relation between sets of Bell states, where two sets of Bell states are equivalent if and only if there exist single-particle unitary operations on the left and/or right particles that transforms one set of Bell states into the other. Since all the sets of Bell states in a given equivalence class have the same distinguishability characteristic, we only need to check one member from each distinguishability class in order to check for all the possible $k-$subsets.

To determine these equivalence classes, we use a base set of unitary matrices in the Bell state basis corresponding to single-particle unitary operations on the left and/or right particles. Since the composition of unitary operations is also unitary, we can find more unitary matrices corresponding to single-particle unitary operations on the left and/or right particles by multiplying the matrices in our base set together. Once we have found all the possible unitary matrices that can be generated by our base set, we can apply these unitary matrices all at once to a particular set of Bell states to generate an entire equivalence class for that set of Bell states. The larger the equivalence classes generated, the better of a reduction in the number of cases we need to check. Thus, it is important for us to generate large sets of unitary matrices with our base set in order to simplify the number of cases to check as much as possible.

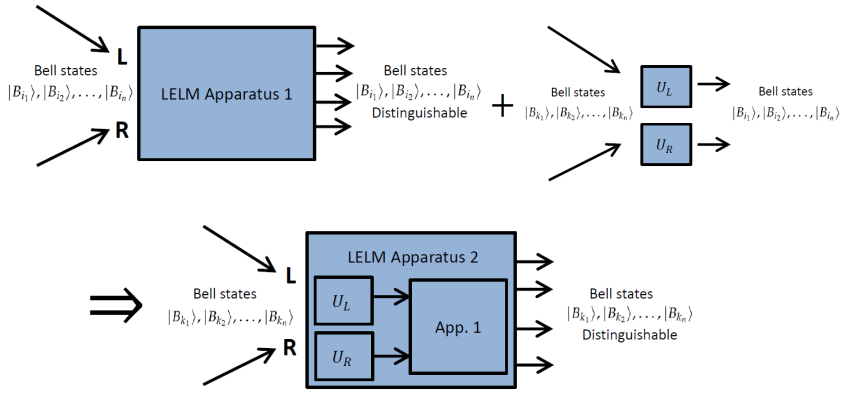

Figure 3.1 If a set of n Bell states $\{|B_i\rangle\}$ are distinguishable and there exists a single-particle unitary transformation from $\{|B_i\rangle\}$ to another set of n Bell states $\{|B_k\rangle\}$, then $\{|B_k\rangle\}$ is also distinguishable. This defines an equivalence relation.

3.3 Unitary Matrices

The unitary matrices used for the qubit case, in the basis given by Equations 2.10a through 2.10d, are given by [8]

$$QB1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.1}$$

$$QB2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.2}$$

$$QB3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.3}$$

$$QB4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{3.4}$$

These matrices represent unitary transformations on the left and right qubit particles that Julien developed. $QB2$ represents the transformations $|1\rangle_L \rightarrow i\,|1\rangle_L$ and $|1\rangle_R \rightarrow i\,|1\rangle_R$. This transformation takes $|\Phi^+\rangle$ to $|\Phi^-\rangle$, $|\Phi^-\rangle$ to $|\Phi^+\rangle$, and leaves the states $|\Psi^+\rangle$ and $|\Psi^-\rangle$ unchanged, which is represented by swapping the first two columns of $QB1$. Similarly, $QB1$ represents the identity transformation, $QB3$ represents the transformation $|0\rangle_L \rightarrow \frac{1}{\sqrt{2}}(|0\rangle_L + i\,|1\rangle_L)$, $|0\rangle_R \rightarrow \frac{1}{\sqrt{2}}(|0\rangle_L + i\,|1\rangle_L)$, $|1\rangle_L \rightarrow \frac{1}{\sqrt{2}}(|0\rangle_L - i\,|1\rangle_L)$, and $|1\rangle_L \rightarrow \frac{1}{\sqrt{2}}(|0\rangle_L - i\,|1\rangle_L)$, while $QB4$ represents the transformation $|0\rangle_L \rightarrow \frac{(1+i)}{2}\,|0\rangle_L + \frac{(1-i)}{2}\,|1\rangle_L$, $|0\rangle_R \rightarrow \frac{(-1+i)}{2}\,|0\rangle_L + \frac{(1+i)}{2}\,|1\rangle_L$, $|1\rangle_L \rightarrow \frac{(1-i)}{2}\,|0\rangle_L + \frac{(1+i)}{2}\,|1\rangle_L$, and $|1\rangle_R \rightarrow \frac{(-1-i)}{2}\,|0\rangle_L + \frac{(1-i)}{2}\,|1\rangle_L$.

The unitary matrices used for the qutrit case, in the basis given by $2.11a$ through $2.11i$, are given by [9]

$$QT1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.5}$$

$$QT2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \tag{3.6}$$

$$QT3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{3.7}$$

$$QT4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{3.8}$$

$$QT5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.9}$$

$$QT6 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}. \tag{3.10}$$

These matrices represent unitary transformations on the left and right

qutrit particles that Julien developed, in addition to the matrix $QT6$ that Lucas developed. $QT1$ represents the identity transformation, $QT2$ represents the transformation $|0\rangle_R \rightarrow |1\rangle_R$, $|1\rangle_R \rightarrow |2\rangle_R$, and $|2\rangle_R \rightarrow |0\rangle_R$, $QT3$ represents the transformation $|1\rangle_L \rightarrow e^{2\pi i/3}|1\rangle_L$ and $|2\rangle_R \rightarrow e^{-2\pi i/3}|2\rangle_R$, $QT4$ represents the transformation $|0\rangle_L \rightarrow e^{2\pi i/3}|0\rangle_L$ and $|0\rangle_R \rightarrow e^{-2\pi i/3}|0\rangle_R$, and $QT5$ represents the change of basis transformation $|0'\rangle = \frac{1}{\sqrt{3}}(|0\rangle + |1\rangle + |2\rangle)$, $|1'\rangle = \frac{1}{\sqrt{3}}(|0\rangle + e^{2\pi i/3}|1\rangle + e^{-2\pi i/3}|2\rangle)$, and $|2'\rangle = \frac{1}{\sqrt{3}}(|0\rangle + e^{-2\pi i/3}|1\rangle + e^{2\pi i/3}|2\rangle)$, followed by the transformation performed in $QT4$.

Lucas did not specify the unitary transformations for the matrix $QT6$, so I wrote code in the Mathematica notebook QT6.nb in order to try to determine the exact unitary transformations that it represented. However, I was unable to find valid single-particle unitary transformations on the left and/or right particles that would results in a matrix of the form $QT6$, and it has been recently discovered by another student in our group that these unitary transformations cannot exist. However, I have included the matrix $QT6$ as part of the set of qutrit unitary matrices for comparison's sake between my code and previous work done by Julien and Lucas. Thus, the code described in Section 3.6 must eventually be re-run in order to see if not including $QT6$ increases the number of cases to check for the qubit⊗qutrit case.

Notice that for both the qubit and qutrit unitary matrices, we have ignored overall phase factors for each individual Bell states. Since overall phases do not matter when characterizing each individual Bell state, we have simplified each unitary matrix in Equations 3.1-3.10 so that the overall phases for each Bell state is 0.

We can change the basis of these matrices into the joint-particle state basis with the following transformation matrices:

$$T_{B\rightarrow S} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \end{pmatrix}, \tag{3.11}$$

$$T_{T\to S} = \frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & e^{2\pi i/3} & e^{-2\pi i/3} \\ 1 & e^{2\pi i/3} & e^{-2\pi i/3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & e^{2\pi i/3} & e^{-2\pi i/3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & e^{-2\pi i/3} & e^{2\pi i/3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & e^{-2\pi i/3} & e^{2\pi i/3} \\ 1 & e^{-2\pi i/3} & e^{2\pi i/3} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.12}$$

The matrix $T_{B\to S}$ transforms a matrix from the qubit Bell state basis to the $2\otimes 2$ joint-particle state basis while the $T_{T\to S}$ matrix a matrix from the qutrit Bell state basis to the $3\otimes 3$ single-particle state basis. These matrices were used to generate equivalence classes in VictorBellEquivalences.nb, which will be explained in more detail later.

Since the unitary matrices are given in the Bell state basis, each unitary matrix is simply a permutation matrix of the Bell states. Hence, in order to generate an equivalence class for a particular set of Bell states, we just have to use matrix multiplication to multiply the unitary matrix by a column vector which represents the set of Bell states being used to generate the equivalence class. Since this matrix multiplication is being done in the Bell state basis, the vector for a set of Bell states simply consists of a column vector $\{e_1, e_2, \ldots, e_n\}$, where $e_i = 1$ if the ith Bell state is a part of the set and $e_i = 0$ otherwise. For example, the set of qubit Bell states $\{|\Phi^+\rangle, |\Psi^+\rangle, |\Psi^-\rangle\}$ can be represented in the qubit Bell state basis as

$$\{|\Phi^+\rangle, |\Phi^-\rangle, |\Psi^-\rangle\} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}. \tag{3.13}$$

The set of Bell states that results from the transformation given by $QB2$ is then given by

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \tag{3.14}$$

which is the set $\{|\Phi^-\rangle, |\Psi^+\rangle, |\Psi^-\rangle\}$.

The unitary matrices given in 3.1-3.10 are by no means an exhaustive set of matrices that generates the largest equivalence class possible. However,

since additional basis unitary matrices were difficult to find, and for ease of comparison between my code and previous code, I decided to stick with the unitary matrices given in Equations 3.1-3.10.

For the base set of unitary matrices for the qubit⊗qutrit case, we include the non-identity qubit unitary matrices 3.2-3.4 tensored with the qutrit identity matrix 3.5, the qubit identity matrix 3.1 tensored with the non-identity qutrit matrices 3.6-3.10, and the 36×36 identity matrix corresponding to the identity transformation on the qubit⊗qutrit Bell states.

3.4 Qubit and Qutrit Equivalence Classes

Since the unitary matrices given by Equations 3.1 through 3.4 are permutation matrices of the Bell states, we can represent the qubit unitary matrices in cycle notation, where 1 corresponds to $|\Phi^+\rangle$, 2 corresponds to Φ^-, 3 corresponds to $|\Psi^+\rangle$, and 4 corresponds to $|\Psi^-\rangle$. For example, the cycle form of $QB2$ is given by (1 2) since $QB2$ swaps $|\Phi^+\rangle$ and Φ^- while leaving $|\Psi^+\rangle$ and $|\Psi^-\rangle$ unchanged. Under this notation, the unitary matrices given by Equations 3.1 through 3.4 are a subset of the symmetric group S_4, which is the group of all permutations of four elements. Thus, any combination of $QB1$ through $QB4$ will result in another permutation matrix whose cycle form is an element of S_4.

It is well known that any $n-$cycle and a $2-$cycle of adjacent elements in the $n-$cycle is a generator for S_n, the symmetric group of n elements. Thus, since the cycle form of $QB4$ is (1 2 3 4) and the cycle form of $QB2$ is (1 2), the permutations $QB2$ and $QB4$ generate all of S_4. Hence, every permutation of the qubit Bell states represents single-particle unitary operations on the left and/or right particles, so every set of qubit Bell states is equivalent to any other set of qubit Bell states. Therefore, for any k, there is only one equivalence class of k qubit Bell states, so one only needs to check one particular set of k qubit Bell states to determine if k qubit Bell states are distinguishable by LELM devices. It has been well-established by the no-go theorem that the maximum number of qubit Bell states that can be distinguished using an LELM apparatus is three [19, 1].

3.5 Matlab Code

In the case of the qutrit and qubit⊗qutrit Bell states, it was not immediately obvious which permutations were generated. In addition, previous code

written by Lucas Brady in Mathematica [9] could generate equivalence classes for sets of 7 and 8 qubit⊗qutrit Bell states within a week, but failed for the case of sets of 9 qubit⊗qutrit Bell states. Thus, I decided to write my own code to generate the equivalence classes for both the qutrit and qubit⊗qutrit cases. Since generating equivalences classes involved many matrix operations, Matlab seemed like a good place to start. The scripts that actually generated the equivalence classes were FindAllEquivalences.m and FindAllEquivalences2.m. Pseudocode for both scripts are shown in Figures 3.2 and 3.3.

Since Lucas's code already generated equivalence classes for sets of 7 and 8 qubit⊗qutrit Bell states, I initially focused my efforts on determining the equivalence classes for sets of 9 qubit⊗qutrit Bell states. In FindAllEquivalences, the basic idea was to represent sets of 9 qubit⊗qutrit Bell states as a vector of 9 ones and $36 - 9$ zeroes, where the ones corresponded to the qubit⊗qutrit Bell states included in the set and the zeroes corresponded to qubit⊗qutrit Bell states left out of the set. By looping through 9 indices that were appropriately spaced apart, I could loop through all $\binom{36}{9}$ sets of 9 qubit⊗qutrit Bell states to apply the unitary matrices generated by $QT1 - QT6$ and generate the equivalence classes for sets of 9 qubit⊗qutrit Bells states.

The rate at which Matlab was able to pass through loop statements became a limiting factor in generating equivalence classes for the qubit⊗qutrit case. For example, by testing my code, I estimated FindAllEquivalences.m

```
function FindAllEquivalences(Matrices)
    FirstStates = {}
    i = 0
    for i1 = 1 : 28, i2 = i1 + 1 : 29, . . . , i9 = i8 + 1 : 36 do
        CurrentStates = Multiply vector(i1, i2, i3, i4, i5, i6, i7, i8, i9) by
Matrices
        if member(FirstStates,CurrentStates) == 0 then
            FirstStates[end+1] = CurrentStates[1]
        end if
        i + +
    end for
    return FirstStates
end function
```

Figure 3.2 Pseudocode for the Matlab script FindAllEquivalences.m

```
function FindAllEquivalences2(Matrices,nchoosek)
    FirstStates = {}
    i = 0
    for n = 1 :length(nchoosek) do
        CurrentStates = Mutiply vector2(nchoosek[end],nchoosek[n, :]) by
Matrices
        if member(FirstStates,CurrentStates) == 0 then
            FirstStates[end+1] = CurrentStates[1]
        end if
        i + +
    end for
    return FirstStates
end function
```

Figure 3.3 Pseudocode for the Matlab script FindAllEquivalences2.m

to take over 100 days to generate equivalence classes for sets of 9 Bell states in the qubit⊗qutrit case. Thus, I decided to write FindAllEquivalences2.m in an attempt to speed up FindAllEquivalences.m using a different method of generating equivalence classes. FindAllEquivalences2 takes in a list of unitary matrices and an $\binom{n}{k}$ matrix and generates an equivalence class by applying the list of unitary matrices to $n \times 1$ vectors with k ones and $n - k$ zeros generated by the $\binom{n}{k}$ matrix in order to find all equivalence classes of sets of k out of n Bell states. This method avoids the problem of looping through 9 indices, while also being generalizable to the general qudit case.

Both these scripts involved helper functions in order to make the process more efficient. These helper functions were SetMatrices.m, GenerateGroup.m, GenerateTransform.m, member.m, vector.m, and vector2.m SetMatrices.m defines the sets of unitary matrices given by 3.1 through 3.10, and the transformation matrices $T_{B\to S}$ and $T_{T\to S}$ for the other functions to use. It also defines the set of qubit or qutrit Bell states that I want to generate an equivalence class for in both the Bell states basis and the single-particle state basis using the transformation matrices. GenerateGroup.m takes in our FullTransform list of unitary matrices and finds all possible unitary matrices that can be generated by them. We can then immediately find the entire equivalence class for a set of Bell states by hitting it with the entire group. GenerateTransform.m takes in two $d \times d$ unitary matrices representing linear transformations on the left and right single particle $d-$qudit states and a $d^2 \times d^2$ change of basis matrix from the single particle basis to

the qudit bell state basis, and returns the transformation matrix associated with the kronecker product of the two matrices in the qudit bell state basis. member.m takes in two list of matrices, Matrices1 and Matrices2, and returns 1 if any element of Matrices1 is in Matrices2 and 0 otherwise. vector.m takes in a list of 9 coordinates and inserts a 1 into a vector of 36 zeros at each of the coordinates given. This function allows us to store the sets of Bell states as a smaller list of coordinates rather than a large vector of size 36. vector2.m takes in a positive integer n and a row matrix of k coordinates and returns a vector with a 1 at each of the n coordinates and 0 elsewhere.

FindAllEquivalences2.m also failed to produce equivalence classes for sets of 9 qutrit states in a timely manner as its runtime was not significantly different from FindAllEquivalences.m. Thus, I decided to move onto Mathematica since Mathematica is far more efficient than Matlab at running through loops. Mathematica also contained a large variety of built-in functions that I could implement to make my code more efficient.

3.6 Mathematica Code

My work in Mathematica directly expands upon Lucas's summer work. Lucas had previously developed a probabilistic algorithm to generate equivalence classes for the qutrit⊗qutrit case, which was able to successfully generate equivalence classes for sets of $k = 7$ and $k = 8$ qubit⊗qutrit Bell states within a period of a week. However, the $k = 9$ case proved too inefficient for his algorithm to halt within a reasonable time frame. I developed a new algorithm that not only improves the run time of calculating equivalence classes significantly, but also generates them using a deterministic method. The code for the algorithm is contained in the Mathematica notebook VictorBellEquivalences.nb.

The main work of the code is contained in the function FindEquiv, which takes in a list of matrices representing an equivalence class and finds equivalence classes of sets of k out of n Bell states represented by vectors of length n with k ones and $n - k$ zeros. It returns a list whose first element is a list of a representative set from each equivalence class and whose second element is a list of the size of each equivalence class. The matrices used in the argument of FindAllEquiv were imported from the text file GenerateMatrices.txt, which contained the unitary matrices for the qubit⊗qutrit Bell states generated by the Matlab script GenerateGroup.m. The pseudocode for FindEquiv is shown in Figure 3.4.

```
function FindAllEquiv(matrices,n,k)
    statesToCheck = List of (n choose k) 1's
    i = 0
    for i < (n choose k) do
        subset = UnrankKSubset(i, k,Table(p, {p, n}))
        currentState = MakeState(subset, n)
        equivClass = Apply matrices to currentState
        Append currentState to outputStates
        Append length(equivClass) to outputLengths
        Get equivIndex for equivClass
        Set statesToCheck[equivIndex[j]] = 0 for all j =
1, . . . ,length(equivIndex)
    end for
    return outputStates and outputLengths
end function
```

Figure 3.4 Pseudocode for the Mathematica function FindAllEquiv, found in VictorBellEquivalences.nb

One of the main issues I encountered while trying to generate equivalence classes for the qubit⊗qutrit case using a deterministic algorithm was the need to store the Bell states that I already checked in some sort of list. Since the Bell states for the qubit⊗qutrit case are vectors of length 36, storing $\binom{36}{k}$ of these Bell states directly was unfeasible with the memory I had available. Thus, I decided to order the Bell states and store them in a list of $\binom{36}{k}$ boolean values, with which I would mark the boolean value of the ith element of the list as 1 if the ith Bell state had been checked and 0 otherwise. In this way, I could keep track of which Bell states I had checked already so that my algorithm did not rely on randomly generating and checking Bell states like previous code from Lucas's work.

In order to convert between the position of a boolean value on a list and a specific qubit⊗qutrit Bell state, FindAllequiv uses two helper functions MakeState and GetSubset. MakeState takes in a $k-$subset of the list of integers from 1 to N representing the coordinates of a state and creates a corresponding vector of length N with a 1 at each coordinate and 0 everywhere else. We use MakeState to generate the vectors being multiplied in FindEquiv. GetSubset takes in a state of length n with k ones and $n-k$ zeros and returns a list of the coordinates of the ones in the state corresponding to a particular $k-$ subset of the list of integers from 1 to N. We use GetSubset to

transform from the vector form of the state to the subset form in FindEquiv.

Using FindEquiv, I was able to generate equivalence classes for sets of 8 qubit⊗qutrit Bell states in a couple hours on my Windows 8 laptop, which took a week with Lucas's code, and I was also able to generate equivalence classes for sets of 9 qubit⊗qutrit Bell states in about half a day, which Lucas's code was unable to generate. For the $k = 9$ case, FindEquiv was able to reduce the number of cases to check from $\binom{36}{9} = 94,143,280$ to $10,365$ using Lucas' matrix $QT6$. The equivalence classes are stored in the text file All9Classes.txt as vectors of 1's and 0's. These vectors were converted to Bell state representations using Mathematica code found in LucasConvertEquivalence.nb, which had been previously created by Lucas to convert between vector and Bell state notations. The result of this conversion is stored in the text file InitialStates9.txt. Now that I had successfully written code to generate equivalence classes for the qubit⊗qutrit case, I was ready to move on to checking members from each equivalence class in order to determine the maximal number of Bell states that could be distinguished in the qutrit and qubit⊗qutrit case.

Chapter 4

Qutrit Bell State Distinguishability

4.1 Introduction

Julien in his summer work had previously determined that sets of 3 qutrit Bell states were distinguishable using LELM apparatuses and that sets of 5 were not[8]. For the $n = 4$ case, Julien was able to separate the sets of Bell states into two equivalence classes using only the matrices $QT1 - QT5$. One equivalence class had no solutions to the necessary conditions in Equation 2.12 for a particular detector, but Julien found a solution to Equation 2.12 in one detector for the other equivalence class, which is represented by the set of Bell states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$. However, Julien was unable to rule out if the equivalence class given by the set $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$ satisfied all the necessary conditions outlined in Equation 2.12 for all six detectors. I decided to explore this issue in greater detail in order to determine if the maximal distinguishability of qutrit Bell states was 3 or 4. All tests of runtime were completed on a Windows 8 laptop with an Intel Core i7-4810MQ processor, 2.80 GHz processing speed, and 16.0 GB of RAM.

4.2 Initial Approaches

I began by implementing the full set of distinguishability conditions in the Mathematica notebook QutritBellStates.nb. Using this notebook, I can generate the full system of Equations given by Equation 2.12 using the function EquationSystem. EquationSystem takes in a list of Bell states and

```
function EquationSystem(StateList)
    for i, j = 1 :length(StateList) do
        if i < j then
            Equation = {WriteEquation0(StateList[i],Statelist[j]),...,
WriteEquation5(Statelist[i],Statelist[j]}
            EqRe = Real part of Equation
            EqIm = Imaginary part of Equation
            System = UnitaryCondition + EqRe + EQIm
        end if
    end for
    return System
end function
```

Figure 4.1 Pseudocode for the Mathematica function EquationSystem, found in QutritBellStates.nb

returns a list of equations corresponding to Equation 2.12, where $|\Phi_k\rangle$ are the Bell states in the list given to EquationSystem. I can then use Mathematica's built-in function Solve in order to attempt to determine whether the system of equations produced by EquationSystem has a solution. The pseudocode for EquationSystem is shown in Figure 4.1.

I began QutritBellStates.nb by defining the qutrit Bell states in the list BellList. For the qutrit case, there are total of $2 \times 3 = 6$ single-particle states, so we require 6 detectors corresponding to a 6×6 unitary matrix. Thus, I defined a unitary condition for a general 6×6 matrix by first defining the 6×6 matrix $\{\text{UnitaryMatrix}\}_{kl}$ =akl+b$kl \cdot i$, where both $a$kl and $b$kl are real coefficients. Then, the unitary condition is given by the Equation UnitaryMatrix·UnitaryMatrix$^\dagger = I_6$, where UnitaryMatrix† is the conjugate transpose of UnitaryMatrix and I_6 is the 6×6 identity matrix. To make the code more efficient, I restricted the solution search space to the real numbers by separating the unitary condition into its real and imaginary parts, which results in 72 individual equations from the condition that UnitaryMatrix·UnitaryMatrix$^\dagger = I_6$ which I stored in UnitaryConditions.

I then wrote the helper function FindEquation, which takes in two states and generates the equations in Equation 2.12 by following the method in the Lutkenhaus simple criteria paper [19]. The helper functions WriteEquation$i$ for $i = 0, \ldots, 5$ converts the equations given by FindEquation into symbolic form for the $i-$th detector mode. To do this, FindEquation utilizes the Switch function in Mathematica, which replaces certain characters such as

```
function FindUnitaryMatrix(StateList)
    distinguishable = false
    while distinguishable = false do
        RR = random real number between 0 and 1
        RC = RR + RR·i
        unitaryMatrix = Orthogonalize(6 × 6 matrix of RC)
        unitarySystem = UnitaryEquationSystem(StateList, unitarymatrix)
        if unitarySystem does not contain false then
            distinguishable = True
            apparatus = unitaryMatrix
        end if
    end while
    return apparatus
end function
```

Figure 4.2 Pseudocode for the Mathematica function EquationSystem, found in QutritBellStates.nb

0, 1, and 2 with the appropriate variables corresponding to detection of the Bell states at the detection modes.

After applying EquationSystem to the set of Bell states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$, I obtained a system of 144 equations of second degree polynomials with real coefficients. After saving the resulting system of equations in the variable SystemFour, I ran Mathematica's built-in method Solve on SystemFour in order to try to determine if there existed a solution to SystemFour. If a solution exists to a given system of equations, Solve will return a specific solution satisfying the system, and Solve will return null if Mathematica cannot find a numerical solution. However, Mathematica was unable to return a result after running for several days. Thus, I decided to attempt to solve SystemFour using another approach in the Mathematica notebook FindUnitaryMatrix.nb, which contained the function FindUnitaryMatrix. The pseudocode for FindUnitaryMatrix is shown in Figure 4.2.

FindUnitaryMatrix attempts to find a 6×6 unitary matrix whose entries represent the coefficients for the detector modes of an apparatus that can distinguish a given set of Bell states. It does this by generating random 6×6 unitary matrices and testing the elements of the resulting matrix with the necessary and sufficient conditions as outlined in Equation 2.12. Unfortunately, if the set of Bell states are not distinguishable, FindUnitaryMatrix

will run forever. Thus it can only be used to verify sets of Bell states that are distinguishable. It turned out that FindUnitaryMatrix was unable to find a unitary matrix for a set of detector modes that could distinguish the set of Bell states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$ after running for several days.

The Mathematica notebook QutritBellStates2.nb was a third attempt at finding a unitary matrix representing an LELM apparatus that could distinguish the set of Bell states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$. The main idea was that instead of solving for the full system of equations outlined in Equation 2.12 for each detector simultaneously, it would be computationally simpler to solve for a particular detector, which had been accomplished by Julien [8]. However, in order to ensure the unitary condition outlined in [19] was maintained, the six rows of UnitaryMatrix corresponding to the coefficients of the six detector modes must be pairwise orthogonal. Thus, I decided to test this method in the simpler case for the qubit Bell states in the Mathematica notebook QubitBellStates.nb, where I wrote the function FindAllSolutions in order to find four orthogonal detectors to construct a qubit Bell State distinguishability apparatus. The pseudocode for FindAllSolutions is shown in Figure 4.3.

FindAllSolutions first attempts to find an individual detector mode satisfying Equation 2.12 for one particular detector, then finds another detector that was orthogonal to the first detector. FindAllSolutions would halt once coefficients for UnitaryMatrix corresponding to four pairwise orthogonal detector modes were found. However, if no solution existed to the set of conditions produced by the set of Bell states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$, FindAllSolutions would halt. For the qubit case, FindAllSolution was able to successfully find solutions for sets of distinguishable states and return no solutions for indistinguishable sets of states in less than a second. Hence, I decided to scale up the method for the qutrit case in the notebook QutritBellStates2.nb in order to see if this method could resolve the issue of maximal qutrit Bell state distinguishability. However, after running QutritBellStates2.nb for several days, FindAllSolution did not halt. At this point, I began to look for other methods for determining whether a solution existed to the system of equations produced by EquationSystem.

4.3 Gröbner Bases

Since the distinguishability of a set of Bell states depends only on the existence of a solution to the set of equations in 2.12, and not the exact solution

```
function FindAllSolutions(StateList)
    AllSolutions = 0 and Solutions = {}
    while AllSolutions == 0 and length(Solutions) < 4 do
        FullEquationSystem = EquationSystem(StateList)
        for i = 1 :length(Solutions) do
            Apply orthogonality conditions to states in StateList
            Separate orthogonality conditions into real and imaginary
components EqRe and EqIm
            FullEquationSystem = FullEquationSystem + EqRe + EqIm
        end for
        Result = FindInstance(FullEquationSystem)
        if Result == {} then
            AllSolutions = 1
            Append Result to Solutions
        end if
    end while
    return Solutions
end function
```

Figure 4.3 Pseudocode for the Mathematica function FindAllSolutions, found in QubitBellStates.nb

itself, I decided to look into computational methods that could determine if a system of equations had a solution without having to solve for the system directly. A key feature of the equations generated by EquationSystem was that each one was a second-degree polynomial equation with real coefficients of the form $f_1(a00, a01, \ldots, b54, b55) = f_2(a00, a01, \ldots, b54, b55)$. Since this equation is equivalent to the equation $f_1 - f_2 = 0$, the solutions to the system of equations $\{f_{i1} = f_{i2}\}$ given by EquationSystem is just the set of common complex zeroes of the set of second-degree polynomials $\{f_{i1} - f_{i2}\}$. Hence, the problem of whether a set of qutrit Bell states are distinguishable reduces to the problem of whether a common zero exists between a set of second-degree polynomials with real coefficients.

One way this can be done computationally is to use the concept of a *Gröbner basis*. The idea of Gröbner bases was created by Bruno Buchberger in his 1965 Ph.D. thesis, where he not only developed the theory of Gröbner bases but also described an algorithm to compute one [20]. In order to define what a Gröbner basis is, we must first define a few mathematical terms. Let K be a field, such as the real numbers $\mathbb{R}$, and consider the polynomial ring $K[x_1, \ldots, x_n]$ whose elements are polynomials in the variables $x_1, \ldots, x_n$ with coefficients lying in K. If $F = \{f_1, \ldots, f_k\}$ is a finite set of polynomials, then the *ideal* generated by $\mathbb{F}$ is the set given by

$$\langle F \rangle = \{\sum_{i=1}^{k} p_k f_k | g_1, \ldots, g_k \in K[x_1, \ldots, x_n]\}. \tag{4.1}$$

In other words, the ideal $\langle F \rangle$ is just the set of linear combinations of elements in F, where the coefficients are polynomials in the ring $K[x_1, \ldots, x_n]$. Before we can define a Gröbner basis, we must first give a *term ordering* to the ring $K[x_1, \ldots, x_n]$, which is a total order $\prec$ on the monomials $x^a = x^{a_1} \cdots x^{a_n}$ of $K[x_1, \ldots, x_n]$ that satisfies the following properties:

1. $x^a \prec x^b$ implies $x^{a+c} \prec x^{b+c}$ for all $a, b, c \in \mathbb{N}$.

2. $1 \prec x^a$ for all $a \in \mathbb{N}^n \backslash \{0\}$, where 1 is the constant monomial.

Then, for a polynomial $f \in F$, the *initial term*, denoted by $in_\prec(f)$, is defined as the largest monomial x^a that occurs in f under the term order $\prec$ with a non-zero coefficient. A Gröbner basis of an ideal I is a finite subset $G \subset I$ such that

$$\langle \{in_\prec(g) | g \in G\} \rangle = \langle \{in_\prec(f) | f \in I\} \rangle \tag{4.2}$$

In other words, a Gröbner basis is a finite set of polynomials from I whose initial terms generate the same ideal as the ideal generated by all the initial terms of I.

The most convenient property of Gröbner bases to my work is the fact that a Gröbner basis of an ideal $I = \langle F \rangle$ under any term ordering has the same set of common zeroes as the set of polynomials F that generated the ideal I. This property is due to the fact that the set of common zeroes to F depends only on the ideal generated by F. *Hilbert's Nullstellensatz* then implies that F has no common zeroes if and only if G contains the polynomial 1. Thus, we can determine whether a common zero exists between a set of second-degree polynomials with real coefficients by computing a Gröbner basis of the set of polynomials. Then, 1 is contained in the Gröbner basis if and only if no solution exists to the system of equations produced by EquationSystem for a particular set of Bell states, which reduces the problem of checking the distinguishability of Bell states to computing a Gröbner basis.

4.4 Application of Gröbner Bases in Mathematica and Singular

Since Mathematica had a built-in method GroebnerBasis for computing a Gröbner basis, I decided to use Mathematica's implementation of the Gröbner basis algorithm instead of creating my own. I created the Mathematica notebook QutritGroebnerBasis.nb in order to test the efficiency of Mathematica's implementation. The main modification was in the function PolynomialSystem, where instead of generating a system of equations to be solved for, PolynomialSystem in QutritGroebnerBasis.nb instead produced a list of polynomials whose common zeroes corresponded to the solution set to the original system of equations generated by EquationSystem. The output of PolynomialSystem was then passed onto the method GroebnerBasis with the ordering $a00 \prec a01 \prec \ldots \prec a54 \prec a55 \prec b01 \prec \ldots \prec b55$. Under this ordering, GroebnerBasis did not halt after running for several days.

To test the efficiency of Mathematica's implementation on a smaller scale, I created the Mathematica notebook QubitGroebnerBasis.nb. Most of the functions were carried over from QutritBellStates, with the Bell states and helper functions appropriately redefined for use in the qubit case. After testing both GroebnerBasis and FindInstance, a built-in method for finding a particular solution to a system of equations, for the qubit case,

both methods failed to halt after several hours of run-time. Therefore, it seemed that Mathematica's implementation of the Gröbner basis algorithm was too inefficient to resolve the issue of Bell state distinguishability for the qutrit case.

After a bit of research, I discovered that the most state of the art Gröbner basis algorithms were the Faugère's F4 and F5 algorithms, both of which were developed by Jean-Charles Faugère [21, 22]. The F5 algorithm happened to be implemented in the SINGULAR computer algebra system, so I decided to try to compute a Gröbner basis for the polynomials produced by PolynomialSystem for the set of Bell states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$ in SINGULAR. However, in order to do this, I had to convert the coefficients of the polynomials into numerical values since I was unable to define exact values such as $\sqrt{3}$ in SINGULAR. After converting the polynomials from PolynomialSystem into numerical form and passing them into SINGULAR's Gröbner basis function, SINGULAR returned the set $\{1\}$ almost instantaneously, which means that the set of Bells states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$ cannot be completely distinguished by an LELM apparatus. However, after passing in small subsets of the full system of polynomials generated by PolynomialSystem, SINGULAR generated large sets of polynomials after many hours of run-time. Thus, I had reason to suspect that the result of SINGULAR's Gröbner basis function on the full set of polynomials was inaccurate since it seemed improbable that SINGULAR computed a Gröbner basis for a set of 144 polynomials nearly instantly while SINGULAR was unable to compute a Gröbner basis for a subset of 15 of these polynomials within a few hours.

Thus, though preliminary results in SINGULAR naively suggest that there exists no solution to the set of polynomials given by PolynomialSystem for the set of Bell states $\{\psi_{00}, \psi_{01}, \psi_{02}, \psi_{10}\}$, these SINGULAR results cannot be fully trusted due to the approximation of exact coefficients in the polynomials given to SINGULAR as well as SINGULAR's own erratic behavior for large systems of polynomials.

Chapter 5

Conclusion

In order to simplify the problem of computing the maximal number of Bell states in the qutrit and qubit⊗qutrit case, I have written code in Mathematica that can successfully generate equivalence classes for sets of $n < 36$ qubit⊗qutrit Bell states deterministically and with much higher efficiency than previous iterations of the code. This code has simplified the number of cases to check for sets of 9 qubit⊗qutrit Bell states from $\binom{36}{9} = 94,143,280$ to $10,365$. I have also attempted to investigate the maximal distinguishability of qutrit Bell states using Mathematica code written in QutritBellStates.nb, FindUnitarymatrix.nb, QutritBellStates2.nb, and QutritBellStates3.nb along with the assistance of the SINGULAR computer algebra system. Though preliminary results in SINGULAR suggest that 3 is the maximum number of Bell states that can be distinguished in the qutrit case, more work needs to be done to verify that the results in SINGULAR are valid given the fact that the polynomials used to check the distinguishability of Bell states in the qutrit case were numerical approximations of the exact distinguishability conditions, as well as the fact that SINGULAR's Gröbner basis function may have unintended behavior for large systems of polynomials. The code outlined in Sections 3.5 and 3.6 also need to be re-run without the use of $QT6$ as one of the qutrit unitary matrices in order to see if the number of qubit⊗qutrit cases to check increases.

Future avenues of research include verifying the results in SINGULAR for the qutrit case, as well as determining ways to simplify the system of equations given in 2.9 so that it is more computationally feasible to solve for the distinguishability conditions. Another area of development would be to generalize the method for calculating equivalence classes in VictorBellEquivalences for the qubit⊗qutrit case to two particles hyper-entangled in

multiple qudit variables. It is my hope that the work described in this thesis can guide future computational work on the problem of maximal Bell state distinguishability in the general case of hyper-entangled qudits.

Bibliography

[1] L. Vaidman and N. Yoran, "Methods for reliable teleportation," *Phys. Rev. A*, vol. 59, pp. 116–125, Jan 1999.

[2] C. H. Bennett and S. J. Wiesner, "Communication via one- and two-particle operators on einstein-podolsky-rosen states," *Phys. Rev. Lett.*, vol. 69, pp. 2881–2884, Nov 1992.

[3] J. T. Barreiro, T.-C. Wei, and P. G. Kwiat, "Beating the channel capacity limit for linear photonic superdense coding," *Nature Physics*, vol. 4, pp. 282–286, 2008.

[4] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, "Quantum repeaters: The role of imperfect local operations in quantum communication," *Phys. Rev. Lett.*, vol. 81, pp. 5932–5935, Dec 1998.

[5] M. Żukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert, ""event-ready-detectors" bell experiment via entanglement swapping," *Phys. Rev. Lett.*, vol. 71, pp. 4287–4290, Dec 1993.

[6] N. C. Pisenti, "Distinguishability of hyper-entangled bell states with linear devices." Harvey Mudd College Senior Thesis, 2011.

[7] A. Turner, "Distinguishabiilty of qudit hyperentangled states with linear evolution and local measurement." Harvey Mudd College Senior Thesis, 2014.

[8] J. Devin, "Distinguishability of qutrit bell states." Harvey Mudd College Summer Research Summary (Unpublished), 2011.

[9] L. Brady, "Distinguishability of combined qutrit and qubit states." Harvey Mudd College Summer Research Summary (Unpublished), 2012.

[10] M. Dusek, "Discrimination of the bell states of qudits by means of linear optics," *Optics Communications*, vol. 199, pp. 161–166, July 2001.

[11] N. Pisenti, C. P. E. Gaebler, and T. W. Lynn, "Distinguishability of hyperentangled bell states by linear evolution and local projective measurement," *Phys. Rev. A*, vol. 84, 2011.

[12] P. G. Kwiat and H. Weinfurter, "Embedded bell-state analysis," *Phys. Rev. A*, vol. 58, pp. R2623–R2626, Oct 1998.

[13] D. Bouwmeester, J. W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger, "Experimental quantum teleportation," *Nature*, vol. 390, p. 575, 1997.

[14] K. Mattle, H. Weinfurter, P. G. Kwiat, and A. Zeilinger, "Dense coding in experimental quantum communication," *Phys. Rev. Lett.*, vol. 76, pp. 4656–4659, Jun 1996.

[15] J.-W. Pan, D. Bouwmeester, H. Weinfurter, and A. Zeilinger, "Experimental entanglement swapping: Entangling photons that never interacted," *Phys. Rev. Lett.*, vol. 80, pp. 3891–3894, May 1998.

[16] P. van Loock and N. Lütkenhaus, "Simple criteria for the implementation of projective measurements with linear optics," *Phys. Rev. A*, vol. 69, p. 012302, Jan 2004.

[17] T.-C. Wei, J. T. Barreiro, and P. G. Kwiat, "Hyperentangled bell-state analysis," *Phys. Rev. A*, vol. 75, p. 060305, Jun 2007.

[18] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Phys. Rev. Lett.*, vol. 73, pp. 58–61, Jul 1994.

[19] N. Lütkenhaus, J. Calsamiglia, and K.-A. Suominen, "Bell measurements for teleportation," *Phys. Rev. A*, vol. 59, pp. 3295–3300, May 1999.

[20] B. Buchberger, "Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal." PhD thesis, 1965.

[21] J.-C. Faugère, "A new efficient algorithm for computing Gröbner bases (f4)," *Journal of Pure and Applied Algebra*, vol. 139, pp. 61–81, June 1999.

[22] J.-C. Faugère, "A new efficient algorithm for computing Gröbner bases without reduction to zero (f5)," *ISSAC '02*, pp. 75–83, July 2002.