NTNU – Trondheim
Norwegian University of
Science and Technology

Positive Partial Transpose States in Multipartite Quantum Systems

Øyvind Steensgaard Garberg

Master of Science in Physics and Mathematics
Submission date: June 2012
Supervisor: Jan Myrheim, IFY

Norwegian University of Science and Technology
Department of Physics

Abstract

In this master thesis I study the extremal positive partial transpose (PPT) states of the three qubit ($2 \times 2 \times 2$) system using numerical methods. Using two algorithms which locate PPT states of a specified rank and extremal PPT states respectively, I have located numerical examples of extremal PPT states with a variety of ranks. These numerical results confirm the analytical result that all PPT states of rank less than four are separable. I also derive an upper limit on the ranks of extremal PPT states. The extremal PPT states of lowest rank, the rank four states, were studied in more detail. These states were confirmed to be biseparable in accordance with both previous analytical and numerical results. The range and kernel of these states were examined for product vectors, but none were found. In an attempt to parametrize the SL$\otimes$SL equivalence classes of these extremal rank four states I have studied an analytical method to construct such states based on unextendible product bases (UPBs). This method can be used to create PPT states from a single equivalence class where, by design, the kernel of all states contain a UPB and the range contains no product vectors. All states where the range is not spanned by a basis of product vectors are necessarily entangled. I also present a numerical method for creating extremal rank four states that are symmetric under various combinations of partial transposes. Numerical examination of these states reveal no product vectors in neither range nor kernel.

The existence of rank four states with and without product vectors in their kernel implies the existence of at least two equivalence classes. To get a better impression of these equivalence classes I construct quantities that are invariant under SL$\otimes$SL transformations and must therefore have the same value for all states in the same equivalence class. Calculating the values of these invariants for all the rank four extremal states I have generated gives a seemingly continuous range of values. This indicates that there is an infinite number of equivalence classes likely described by one or more continuous variables. The invariants also revealed an interesting set of states that may belong to a single equivalence class, where one invariant is zero and the others have identical values. This was the only equivalence class where more than one of my states were included. There is obviously something special about this class, but I do not know what it is.

Sammendrag

Målet for denne masteroppgaven har vært å utføre numeriske studier av de ekstremale tilstandene med positive deltransponerte (PPT-tilstander) i systemer bestående av tre qubits (dimensjon $2 \times 2 \times 2$). Ved å bruke to algoritmer som finner henholdsvis tilstander med en spesifisert rang og ekstremale tilstander, har jeg funnet numeriske eksempler på ekstremale tilstander for en rekke forskjellige rangkombinasjoner. Disse numeriske resultatene bekrefter det analytiske resultatet som sier at alle PPT-tilstander i dette systemet som har rang lavere enn fire er separable. Jeg utleder også en øvre grense på rangen til ekstremale PPT-tilstander. De ekstremale PPT-tilstandene med lavest rang, rang fire, ble studert i mer detalj. Disse tilstandene ble i tråd med tidligere numeriske og analystiske resultater bekreftet å være biseparable. Bilde- og nullrommet til disse tilstandene ble gjennomsøkt etter produktvektorer, men ingen ble funnet hverken for tilstandene selv eller deres deltransponerte. I et forsøk på å parametrisere SL⊗SL ekvivalensklassene til disse ekstremale tilstandene har jeg sett på en analytisk metode for å konstruere slike tiltstander som bygger på ikke utvidbare produktbasiser (UPB). Denne metoden kan brukes til å konstruere tilstander fra en enkelt ekvivalensklasse hvor tilstandene er konstruert slik at de har en fullstendig basis av produktvektorer i nullrommet og ingen produktvektorer i bilderommet. Alle tilstander hvor bilderommet ikke utspennes av en produktbasis er nødvendigvis sammenfiltret. Jeg presenterer også en numerisk metode for å konstruere ekstremale rang fire tilstander som er symmetriske under ulike kombinasjoner av deltransponering. Numeriske undersøkelser av bilde- og nullrommet til disse tilstandene viste heller ingen produktvektorer.

Eksistensen av rang fire tilstander med og uten produktvektorer i nullrommet implisererer at det finnes minst to ulike ekvivalensklasser. For å undersøke disse ekvivalensklassene nærmere konstruerte jeg størrelser som er invariante under SL⊗SL-transformasjoner og derfor har de samme verdiene for alle tilstandene i en ekvivalensklasse. Beregning av disse verdiene for alle de ekstremale rang fire tilstandene jeg har generert gir et tilnærmet kontinuerlig spektrum av verdier. Dette indikerer at det finnes et uendelig antall ekvivalensklasser som er beskrevet av en eller flere kontinuerlige variabler. Disse invariante størrelsene avslørte også en spesielt interessant type tilstander som kan tilhøre samme ekvivalensklasse, hvor en invariant var null mens de andre hadde identiske verdier. Dette var den eneste ekvivalensklassen som inneholdt mer enn en av mine numeriske eksempler. Det er åpenbart noe spesielt med denne ekvivalensklassen, men jeg vet ikke hva det er.

Contents

Abstract i

Sammendrag ii

List of Symbols v

Acknowledgments vii

1 Introduction 1

2 Fundamentals 3
- 2.1 Composite systems . . . 3
- 2.2 Pure and mixed states . . . 3
- 2.3 Some linear algebra . . . 4

3 Entanglement 5
- 3.1 A short introduction to entanglement . . . 5
- 3.2 The Peres criterion . . . 7
- 3.3 Convex sets and extremal points . . . 8
- 3.4 Multipartite entanglement . . . 9

4 Numerical studies 13
- 4.1 Searching for extremal points in a convex set . . . 13
  - 4.1.1 Locating the edge . . . 16
  - 4.1.2 Upper limit on the ranks of extreme points . . . 16
  - 4.1.3 Application as separability check for states with low ranks . . . 17
- 4.2 Searching for states with specified ranks . . . 18
- 4.3 Initial results . . . 19

5 A closer look at extremal rank (4,4,4,4) PPT states 21
- 5.1 General considerations . . . 21
- 5.2 UPB construction . . . 23
- 5.3 States symmetric under partial transpositions . . . 25
- 5.4 Invariants . . . 26

6 Summary and further work **33**

A Matlab source code **35**
A.1 extremalSearch 35
A.1.1 createSigma 35
A.1.2 approachEdge 37
A.1.3 hmat2rvec 38
A.1.4 rvec2hmat 39
A.2 rankSearch 39
A.2.1 rankSearchMain.m 39
A.2.2 rankSearch.m 41
A.2.3 cSV 43
A.2.4 genGellMann 44
A.3 Gathering Symmetric States 44
A.3.1 gatherSymmetric 44
A.3.2 findSymFun 45
A.3.3 createSymmetric 45

B Optimization methods **47**
B.1 trippleEigMin 47
B.2 Conjugate gradient method 49
B.2.1 Line minimize 51
B.3 SQP 53
B.3.1 SQPfun 55
B.3.2 SQPconstr 55

List of Symbols

Symbol	Meaning
T_i	Partial tranpose with respect to subsystem i
P	Partial transpose with respect to subsystem two, only used in the bipartite case
$r(\cdot)$	The rank of a matrix
$R(\cdot)$	The range of a matrix
$K(\cdot)$	The kernel of a matrix
$k(\cdot)$	The dimension of the kernel
$\mathcal{D}$	The set of all density matrices
$\mathcal{P}$	The set of all PPT states
$\mathcal{S}$	The set of all separable states
$\mathbf{r}(\cdot)$	The sum of the rank of a state and the ranks of its partial transposes

Acknowledgments

I would like to thank my supervisor Jan Myrheim for all the invaluable help and inspiration he has provided during my work on this thesis. I would also like to thank Per Øyvind Sollid who let me use his source code as a starting point and Andreas Hauge whose guidance during my project work helped me better understand much of the theory behind this thesis.

Trondheim, June 2012
Øyvind S. Garberg

Chapter 1

Introduction

Quantum entanglement is one of the most fascinating features of quantum mechanics. It is a truly non-classical phenomenon and has several interesting applications in information theory, for example teleportation and super dense coding. It also has applications in cryptography. One of the fundamental problems in the study of entanglement is the separability problem or, in other words, the task of determining the separability or entanglement of an arbitrary state.

One of the powerful tools that can be used to help solve this problem is the Peres criterion, which states that the partial transpose of all separable states must be postive semidefinite matrices. This criterion is in general not sufficient to determine separability because the convex set of positive partial transpose states (PPT states) is in general larger than the convex set of separable states. One approach to the separability problem is to study the differences between these two sets. Because these sets are convex they are described completely by their extreme points, which has led to the study of extremal PPT states. These extremal PPT states have been studied numerically in the bipartite case [1, 2]. The goal for this thesis has been to perform a similar study in the multipartite case, mainly focused on the three qubit system i.e. a composite system with three two dimensional subsystems. I focus on this particular system because it is the lowest dimensional multipartite system and it is not obvious that going to higher dimensions would yield any additional insight into the general properties of extremal PPT states in multipartite systems. This particular system is also interesting due to the application of systems of qubits in quantum information theory.

The foundation for the numerical studies I conducted are the algorithms rankSearch and extremalSearch which search for PPT states of specified ranks and extremal PPT states respectively. The consecutive application of these algorithms were used to generated a multitude of extremal PPT states of varying ranks. Among all the PPT states generated using this method one group is particularly interesting and were studied in more detail. These are the extremal PPT states of rank $(r(\rho), r(\rho^{T_1}), r(\rho^{T_2}), r(\rho^{T_3})) = (4, 4, 4, 4)$. There are no mixed entangled PPT states with lower ranks than these and they are also the only rank where both rank is preserved under all partial transposes and we find extremal states. In an attempt to parametrize these states I studied the construction of such states using a

method involving unextendible product bases (UPBs). While this method was successful in producing extremal rank (4,4,4,4) states, a search for product vectors in the kernel of the original states showed that not all such states can be generated using the UPB method implying the existence of multiple equivalence classes of extremal rank (4,4,4,4) states.

Searching for another way to parametrize these equivalence classes I looked into the creation of extremal rank (4,4,4,4) states with special symmetries, in particular different combinations of symmetry under the various partial transposes. To be able to compare the equivalence classes of the states generated with these properties and the equivalence classes of the original states I looked into the creation of quantities that have the same value for all states in an equivalence class. An equivalence class is a set of states that can be transformed into each other using SL⊗SL transformations followed by normalization to unit trace. Four such quantities where found and their values calculated for all states, which indicate that there is an infinite number of equivalence classes described by one or more continuous variables.

I will start this thesis by repeating some fundamental quantum mechanics and linear algebra that will be useful later. These basics are covered in Chapter 2. In Chapter 3 I give the theoretical foundation for the thesis by going through the basics of entanglement and discussing the Peres criterion and the theory of convex sets in more detail. The algorithms extremalSearch and rankSearch are presented in Chapter 4 followed by a discussion of the states found by running them. The more detailed study of the extremal rank (4,4,4,4) states are presented in Chapter 5 and I finish with a summary and some avenues for further study in Chapter 6. MATLAB source code for the rankSearch and extremalSearch algorithms is found in Appendix A along with the code for creating rank four states symmetric under all partial transposes. MATLAB source code for the optimization algorithms used to search for product vectors can be found in Appendix B.

Chapter 2

Fundamentals

2.1 Composite systems

A composite system consists of several separate parts or subsystems for example a system of multiple particles. If the states $|\psi_a\rangle, |\psi_b\rangle, ..., |\psi_k\rangle$ of all subsystems $a, b, ..., k$ are known, the state of the composite system is $|\psi\rangle = |\psi_a, \psi_b, ..., \psi_k\rangle$, just as you are used to from your quantum mechanics courses. States on this form are called product states and are formed mathematically by taking the tensor product of the individual substates, i.e. $|\psi_a\rangle \otimes |\psi_b\rangle \otimes ... \otimes |\psi_k\rangle$. All possible states of the composite system can be written as superpositions of product states.

2.2 Pure and mixed states

When the state of a system is exactly known it is a pure state, but if you only know that your system has a probability p_i to be in the pure state $|\psi_i\rangle$, we say that the state is mixed and is described by the density matrix $\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i|$. That a density matrix can be written in this form, i.e. a convex combination of normalized projection operators, is equivalent to the requirements

$$\rho \geq 0, \quad \mathrm{Tr}\rho = 1, \quad \rho^\dagger = \rho. \tag{2.1}$$

All states, including pure states, can be described by density matrices. If your system is in the pure state $|\psi_k\rangle$, it has the probability distribution $p_i = \delta_{ik}$, which gives the density matrix $\rho = |\psi_k\rangle \langle\psi_k|$. This is the projection operator onto the state $|\psi_k\rangle$, and like all projection operators it has the property $\rho^2 = \rho$. All one dimensional projection operators are rank one matrices and all rank one matrices can be written as one dimensional projection operators. This means that rank(ρ)=1 is equivalent to the statement that ρ is a pure state. All states are either pure or mixed. I will denote the set of all density matrices by $\mathcal{D}$.

2.3 Some linear algebra

Here I will introduce some basic linear algebra terms you may have forgotten. This theory can be found in any basic textbook on linear algebra. The image space or range of a matrix A, denoted by $R(A)$, is the vector space spanned by the set of vectors, $\{\psi\}$, where

$$\psi = A\varphi \tag{2.2}$$

and φ is an arbitrary vector. The rank of A, denoted by $r(A)$, is the dimension of $R(A)$, the number of non-zero eigenvalues and both the number of linearly independent columns and rows. The null space or kernel of a matrix A, denoted by $K(A)$, is the vector space spanned by the set of vectors, $\{\chi\}$, where

$$A\chi = 0. \tag{2.3}$$

I will denote the dimension of $K(A)$ by $k(A)$. If A is an $m \times n$ matrix the sum of $k(A)$ and $r(A)$ is always n.

If A is Hermitian $K(A)$ is the orthogonal complement of $R(A)$. This means that all the vectors in $R(A)$ are orthogonal to all the vectors in $K(A)$. Here is the one line proof using the vector definitions given above:

$$\chi^\dagger\psi = \chi^\dagger A\varphi = (A\chi)^\dagger\varphi = 0 \tag{2.4}$$

Chapter 3

Entanglement

In this chapter I will present the theory that forms the foundation and motivation for this thesis. I will first introduce some of the theory behind entanglement in particular some of the physical consequences and applications of entanglement as well as some of the most rudimentary mathematical formalism. I will then discuss the Peres criterion and the partial transpose before moving on to the theory of convex sets and extreme points and explaining how this theory can help us solve the separability problem. I will finish this chapter by discussing the classification of entanglement in multipartite systems.

3.1 A short introduction to entanglement

Entanglement is a quantum mechanical property of a composite system, of importance comparable to that of energy. Just like energy it is hard to describe exactly what entanglement is. It is more convenient to describe how we see its effects. Entanglement can manifest itself in three major ways:

- As strong correlations between results of measurements on individual subsystems.
- As a resource in information theory.
- It allows us to have more information about the system as a whole, than the combined information about the individual subsystems.

The expectation value of simultaneous measurements of an observable A on subsystem a and an observable B on subsystem b is given as $\mathrm{Tr}[(A \otimes B)\rho]$. If the system is in a product state $\rho = \rho_a \otimes \rho_b$, the expectation value of the measurements is

$$\mathrm{Tr}[(A \otimes B)(\rho_a \otimes \rho_b)] = \mathrm{Tr}[(A\rho_a) \otimes (B\rho_b)] = \mathrm{Tr}[A\rho_a]\mathrm{Tr}[B\rho_b]. \tag{3.1}$$

Since the result is simply multiplication of the two individual results, they are completely uncorrelated. If the state is a convex combination of product states,

$$\rho = \sum_i p_i \, (\rho_{a,i} \otimes \rho_{b,i}), \tag{3.2}$$

the expectation value is

$$\sum_i p_i \operatorname{Tr}[A\rho_{a,i}]\operatorname{Tr}[B\rho_{b,i}]. \tag{3.3}$$

It is now possible for the results to be correlated, but all correlation stems from the distribution $\{p_i\}$ which can be generated by classical mechanisms. A convex combination of product states can always be written as a convex combination of pure product states. States that can be written as convex combinations of pure product states are called separable states. I will denote the set of separable states by $\mathcal{S}$. There are states which cannot be written as convex combinations of product states and these states are called entangled [3].

One of the great discussion between physicists in the early days of quantum mechanics was the Einstein–Podolsky–Rosen paradox, which was first presented in a paper from 1935 [4]. The authors argued that qunatum mechanics is an incomplete theory because it is not compatible with the principles of realism and locality. Realism means that all measurement results are determined before they are performed and locality only allows local interactions essentially limiting the speed of interaction between spatially separated systems to the speed of light. Based on these assumptions it is possible to derive upper limits on the correlation of measurement results. These limits are often called Bell inequalities after John S. Bell who published the first limit of this kind in a paper from 1964 [5]. Violations of Bell inequalities have been confirmed experimentally.

Because Bell inequalities limit the correlations between measurement results they require a significant amount of measurements to disprove. In the multipartite case it is possible to construct "Bell equalities" which can be disproved with a single measurement. One such experiment is described in [6] and employs the tripartite GHZ state, which will be mentioned again in Section 3.4.

In general the relationship between entanglement and Bell inequalities is not a simple one, as both the type of state in question and what measurements are allowed has an impact on the result. For a more detailed discussion of the relationship between entanglement and Bell inequalities see Chapter IV in [7], and references therein.

As an example of how a state can contain more information about the system as a whole than about the states of the subsystems I will use the Bell state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|+-\rangle - |-+\rangle), \tag{3.4}$$

where $|+\rangle$ and $|-\rangle$ represent the states with with spin up and down in the z direction respectively. Physically this is a state where the total spin in any direction is zero i.e. if the spin of both particles is measured in an arbitrary direction the sum will be zero. Even though the state provides information about the system as a whole, it does not contain any information about the result of a measurement on an individual subsystem. This can be seen in the mathematical formalism by calculating the reduced density matrices of the subsystems,

$$\rho_a = \operatorname{Tr}_b[\rho] = \frac{1}{2}I \quad \rho_b = \operatorname{Tr}_a[\rho] = \frac{1}{2}I. \tag{3.5}$$

Reduced density matrices are representations of the state of a subsystem. In this case they are both proportional to the identity matrix which provides only the information that any outcome of any measurement on one subsystem is equally likely.

Entanglement has some fascinating applications in quantum information theory. Quantum teleportation and superdense coding are perhaps the best known. Entanglement is an important part of some protocols in quantum cryptography and will play a key role in any future quantum computers. You can read more about the role of entanglement in quantum information theory and quantum computing in [8].

3.2 The Peres criterion

The Peres criterion, which was first published in a paper by Asher Peres in 1996 [9], is an important separability criterion because it requires very little computational power to apply. It states that the partial transpose of any separable state must be a positive semidefinite matrix. The partial transpose is a mathematical operation that treats the state as a product state and transposes one subsystem, i.e. a bipartite state $\rho = \rho_A \otimes \rho_B$ has a partial transpose $\rho^P = \rho_A \otimes \rho_B^T$.

Bipartite states have two partial transposes, one transposes subsystem A and the other subsystem B, but these can be converted into each other via the regular transpose operation. The regular transpose preserves eigenvalues which implies that both rank and positivity is preserved. It is therefore sufficient to check the positivity of one partial transpose. In multipartite systems the different partial transposes are not redundant in this way. In the tripartite system that has been the focus of this thesis, there are three independent partial transposes, independent meaning they cannot be created from each other via a regular transpose. While the third partial transpose is equivalent to applying the previous two partial transposes and a regular transpose, the consecutive application of the different partial transpose operation does not necessarily preserve positivity even though each of them applied alone does. This fact was checked numerically by looking for positive matrices where two of the partially transposed matrices were positive while the third had at least one negative eigenvalue. I will denote partial transposition with respect to subsystem i by T_i, but in the bipartite case I will use P instead of T_2.

The Peres criterion is easily proven since all separable states can be written as convex combinations of pure product states:

$$\rho = \sum_i p_i \, \rho_{A,i} \otimes \rho_{B,i}. \tag{3.6}$$

The partial transpose of this state is then written as

$$\rho^P = \sum_i p_i \, \rho_{A,i} \otimes \rho_{B,i}^T. \tag{3.7}$$

which is also a separable state, since the transpose operation preserves all the requirements that ensure that $\rho_{B,i}$ is a state (positivity, unit trace and Hermiticity). This proves that partial transposition preserves separability.

The Peres criterion is in general only a necessary separability criterion and not a sufficient one. The set of states with positive partial transposes (PPT states), which I will denote as $\mathcal{P}$, is only identical to the set of separable states, $\mathcal{S}$, in 2x2 and 2x3 dimensions [10]. In higher dimensions there are entangled PPT states, which implies that $\mathcal{P}$ and $\mathcal{S}$ are not identical. Due to the computational ease of the Peres criterion the separability problem is essentially reduced to identifying these entangled PPT states.

Different entangled states may not contain the same amount of entanglement. The process of converting a set of entangled states into a smaller set of maximally entangled states is called distillation. Entanglement that can not be concentrated in such a way is called bound. It is known that the entanglement of entangled PPT states is always bound, though this fact is not directly relevant for this thesis. You can read more about distillation and bound entanglement in Chapter XII of [7].

3.3 Convex sets and extremal points

All the previously mentioned sets, $\mathcal{D}$, $\mathcal{P}$ and $\mathcal{S}$, are convex. In a convex set all points on the straight line between two arbitrary points in the set are also included in the set. It is possible to uniquely define a convex set in two equivalent ways. It can be described by a set of algebraic inequalities or other similar conditions or by its extreme points. An extreme point does not lie on any straight line joining two other points of the set. Extreme points can intuitively be interpreted as the corners of the set. Any point in a convex set can be written as a convex combination of extreme points, but extreme points can not be written as convex combinations of other points in the set.

The set of all density matrices, $\mathcal{D}$, can easily be defined in both ways. Either by the conditions stated in equation (2.1) or by its extreme points: the pure states. It is easy to confirm that the pure states are the extreme points of $\mathcal{D}$. Any mixed state can be written as a convex combination of pure states, but a pure state can not be written as a convex combination of other pure states.

The two other sets, $\mathcal{P}$ and $\mathcal{S}$, only have easy definitions in one of these two ways. The set of PPT states is defined by the requirement that all partial transposes must be density matrices. In the bipartite case this means that $\mathcal{P}=\mathcal{D}\cap\mathcal{D}^P$. However the extreme points of $\mathcal{P}$ are not completely known.

Separable states are defined as convex combinations of pure product states and because pure states cannot be written as convex combinations of other pure states the pure product states must be the extreme points of $\mathcal{S}$. It is possible to make an alternative description of $\mathcal{S}$ using entanglement witnesses, but such a description would be highly complicated and unwieldy. Entanglement witnesses are operators with positive expectation values in all separable states and can be used as an alternative approach to the separability problem. You can read more about entanglement witnesses in Chapter VI of [7].

While the extreme points of $\mathcal{P}$ are not completely known, the relationship between $\mathcal{D}$, $\mathcal{P}$ and $\mathcal{S}$, which is described by

$$\mathcal{S}\subset\mathcal{P}\subset\mathcal{D}, \tag{3.8}$$

can be used to learn something about them. Because $\mathcal{P}$ is a subset of $\mathcal{D}$ all the extreme points of $\mathcal{D}$ that are also members of $\mathcal{P}$ must be extreme points of $\mathcal{P}$. This means that since all pure product states, which are members of $\mathcal{S}$ and therefore $\mathcal{P}$ and are extreme points of $\mathcal{D}$, must be extreme points of $\mathcal{P}$. Because all the extreme points of $\mathcal{S}$ are extreme points of $\mathcal{P}$ the differences between the two sets are completely described by the extreme points of $\mathcal{P}$ that are not pure product states. Attaining a parametrization of the extremal PPT states would give great insight into the differences between $\mathcal{S}$ and $\mathcal{P}$.

If a pure state is not a product state, it is not in $\mathcal{P}$ and must therefore be entangled. Since this might not be obvious to everyone I will give a proof in the case of a 2x2 system. Similar proofs can be done for systems of arbitrary dimensions. Using Schmidt decomposition a pure state of a 2x2 system can always be written as

$$\psi = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.9}$$

where a and b are real positive constants. The density matrix corresponding to this pure state is

$$\begin{pmatrix} a^2 & 0 & 0 & ab \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ ab & 0 & 0 & b^2 \end{pmatrix}$$

which has a partial transpose

$$\begin{pmatrix} a^2 & 0 & 0 & 0 \\ 0 & 0 & ab & 0 \\ 0 & ab & 0 & 0 \\ 0 & 0 & 0 & b^2 \end{pmatrix}$$

with eigenvalues a^2, b^2 and $\pm ab$. If ψ is not a product state neither a nor b will be zero and the partial transpose will have a negative eigenvalue. QED.

Since all pure PPT states are product states all entangled PPT states must have ranks higher than one, but this is not the strongest known lower limit on the ranks of extremal PPT states. In the bipartite case it is known that there are no PPT states of rank three containing bound entanglement [11], which means that there are no rank three extremal PPT states. We also know that all states of rank N and lower are separable in the $2 \times 2 \times N$ system [12].

3.4 Multipartite entanglement

In the multipartite case, i.e. composite systems with more than two components, the concepts of entanglement and separability become more nuanced. The bipartite definition of a separable state is easily generalized i.e. an n-partite state is separable if it can be written as

$$\rho = \sum_i p_i \left(\rho_{a,i} \otimes \rho_{b,i} \otimes ... \otimes \rho_{n,i} \right), \tag{3.10}$$

but these states are now called fully separable. In addition we also have the notion of partial separability or k-separability. A pure state is k-separable if it is possible to partition the subsystems so that the state is a product state when regarded as the corresponding k-partite state. A mixed state is k-separable if it can be written as a convex combination of pure k-separable states. For an n-partite system we must naturally have $k \leq n$. It is worth noting that even though a state is k-separable for all possible k-partitions, it might not be $(k+1)$-separable e.g. a tripartite system might be biseparable for all three possible partitions and still not be fully separable.

Just as there are different types of separability there are different categories of entanglement in multipartite systems. There are five inequivalent classes of entangled pure states in a three qubit system [13]. All the states in one equivalence class can be transformed into each other using stochastic local operations and classical communication (SLOCC), stochastic meaning that the operation does not succeed with unit probability. The transformations corresponding to local operations and classical communication (LOCC) can be represented mathematically as SL⊗SL transformations followed by normalization to unit trace. By SL⊗SL transformations I mean transformations on the form

$$\rho \to (V_a \otimes V_b \otimes ... \otimes V_n)\rho(V_a \otimes V_b \otimes ... \otimes V_n)^\dagger \tag{3.11}$$

i.e. product transformations made up of SL transformations of dimensions corresponding to the dimensions of the subsystems.

Three of these classes consist of biseparable states where we have bipartite entanglement between two subsystems. These three classes are identified by which state does not share in the bipartite entanglement. The two remaining classes consist of states with true tripartite entanglement and are exemplified and named after two famous states. The GHZ state,

$$|\psi_{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle), \tag{3.12}$$

and the W-state,

$$|\psi_W\rangle = \frac{1}{\sqrt{3}}(|100\rangle + |010\rangle + |001\rangle). \tag{3.13}$$

A characteristic difference between these two states is the behavior of the inherent entanglement when one qubit is removed from the system. If the system is in the GHZ state and a qubit is removed the remaining to qubits are separable. If a qubit is removed while the system is in the W state the remaining qubits are still entangled.

Removing a qubit is represented mathematically by the partial trace. Tracing out a subsystem in this way leaves us with the reduced density matrix. The partial trace with respect to subsystem C is defined as

$$\rho_{AB} = \mathrm{Tr_C}(\rho) = \sum_{\mathrm{i}} (\mathrm{I_A} \otimes \mathrm{I_B} \otimes |\mathrm{i}\rangle_{\mathrm{C}})^\dagger \rho (\mathrm{I_A} \otimes \mathrm{I_B} \otimes |\mathrm{i}\rangle_{\mathrm{C}}) \equiv \sum_{\mathrm{i}} \langle \mathrm{i}| \, \rho \, |\mathrm{i}\rangle \, . \tag{3.14}$$

If we remove the third qubit of a system in the GHZ state we get

$$\rho_{GHZ,AB} = \frac{1}{2}(|00\rangle \langle 00| + |11\rangle \langle 11|) \tag{3.15}$$

which is a convex combination of product states and therefore separable. Doing the same for the W state we get

$$\rho_{W,AB} = \frac{1}{3}(|00\rangle\langle 00| + |01\rangle\langle 01| + |01\rangle\langle 10| + |10\rangle\langle 10| + |10\rangle\langle 01|). \tag{3.16}$$

It is not quite as easy to determine if this state is separable just by looking at it, but because this is the state of 2×2 system the Peres criterion is sufficient to determine separability. The partial transpose of this state is

$$\rho^P_{W,AB} = \frac{1}{3}(|00\rangle\langle 00| + |01\rangle\langle 01| + |00\rangle\langle 11| + |10\rangle\langle 10| + |11\rangle\langle 00|), \tag{3.17}$$

which can be explicitly written as

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{3.18}$$

If you calculate the eigenvalues of this matrix you will find that one of them is negative and $\rho_{W,AB}$ must therefore be an entangled state. In fact it is shown in [13] that the W state is the state where the maximal amount of entanglement is preserved under the loss of a qubit. The fact that there are two separate classes of true tripartite entanglement is a sign of fundamental differences between bipartite and multipartite entanglement.

These equivalence classes of pure states can be used as the foundation for a complete classification of all $2 \times 2 \times 2$ states into a hierarchy of four convex sets. The first is the set of separable states $\mathcal{S}$. The second set is the class of biseparable states which contains all states that can be written as a convex combination of pure product states and pure biseparable states. This set is denoted by $\mathcal{B}$. The third set is denoted by $\mathcal{W}$ and contains all convex combinations of pure product states, pure biseparable states and W type pure states. The fourth set and class, the GHZ class, also allows GHZ type pure states to be included in the convex combination which means that this set is $\mathcal{D}$ the set of all states because convex combinations of all pure states are allowed. The internal relationships between these sets are obviously enough

$$\mathcal{S} \subset \mathcal{B} \subset \mathcal{W} \subset \mathcal{D}. \tag{3.19}$$

This classification was first introduced in [14] by A. Acin et. al. who also conjectured that all entangled PPT states in three qubit systems are at most in the W class.

The entanglement of pure states in a system of four qubits was classified by F. Verstraete et. al in [15]. They conclude that there are nine inequivalent classes of entangled states. Eight of these represent states where bipartite or tripartite entanglement is distributed among the four subsystems, while the last one contains states which behave like a 4-partite equivalent to the tripartite GHZ state. This indicates that increasing the number of subsystems might not introduce new types of entanglement in the way going from two to

three subsystem introduced the distinction between W- and GHZ type entanglement. This indication makes it tempting to focus numerical studies of multipartite entanglement on the tripartite case because of the increased computational requirements of studying larger systems.

Chapter 4

Numerical studies

The foundation for the numerical studies I have conducted are two algorithms, which I will refer to as extremalSearch and rankSearch. The first algorithm, extremalSearch, is used to search for extreme points in a convex set and the second, rankSearch, searches for PPT states of a specified rank. Both algorithms will be presented in detail in the subsequent sections. I will also describe how extremalSearch can be used as a separability test for PPT states with low ranks and how the theory behind this algorithm can be used to derive an upper limit on the ranks of extremal PPT states. I finish this chapter by discussing what states were found using these algorithms.

4.1 Searching for extremal points in a convex set

To search for extremal states I have used a modified version of the method presented in [2], but I will present it here as well. I will first show how the method can be used to find extremal points of $\mathcal{D}$ as a simple example and then explain how it must be modified to find extremal points in $\mathcal{P}$.

An arbitrary state, ρ, is either an extremal point of $\mathcal{D}$ or can be written as

$$\rho = x\rho_1 + (1-x)\rho_2, \quad 0 < x < 1 \tag{4.1}$$

where ρ_1 and ρ_2 are two different points in $\mathcal{D}$. For an arbitrary vector ψ we have, using the positivity of ρ, ρ_1, and ρ_2 and equation (4.1), that

$$\rho\psi = 0 \Leftrightarrow \psi^\dagger\rho\psi = 0 \Leftrightarrow \psi^\dagger\rho_1\psi = \psi^\dagger\rho_2\psi = 0 \Leftrightarrow \rho_1\psi = \rho_2\psi = 0 \tag{4.2}$$

which means that ψ is in the null space of ρ if and only if it is in the null spaces of both ρ_1 and ρ_2, i.e.

$$K(\rho) = K(\rho_1) \cap K(\rho_2). \tag{4.3}$$

The complementary statement is that the range of ρ contains all vectors that can be written as the sum of a vector from $R(\rho_1)$ and a vector from $R(\rho_2)$, i.e.

$$R(\rho) = R(\rho_1) + R(\rho_2). \tag{4.4}$$

Defining P as the projection operator onto $R(\rho)$ the following relations hold:

$$P\rho P = \rho, \quad P\rho_1 P = \rho_1, \quad P\rho_2 P = \rho_2. \tag{4.5}$$

The matrix $\sigma = \rho - \rho_1$ is now nonzero, Hermitian, has Tr $\sigma = 0$ and fulfills $P\sigma P = \sigma$. Since σ is Hermitian and is traceless it must have both positive and negative eigenvalues, which means that so must $\tau(x)$ for sufficiently large $|x|$, where $\tau(x)$ is defined as

$$\tau(x) = \rho + x\sigma. \tag{4.6}$$

The null space of $\tau(x)$ will contain the null space of ρ for all values of x, because if $\rho\psi = 0$ then $P\psi = 0$ which implies that $\sigma\psi = P\sigma P\psi = 0$. This also implies that $\tau(x)$ will always share the zero eigenvalues of ρ. Since the eigenvalues of $\tau(x)$ change continuously with x, $\tau(x)$ will stay positive in a finite interval around $x = 0$ and at the end points of this interval $\tau(x)$ has at least one more zero eigenvalue than ρ i.e. $r(\tau) < r(\rho)$.

This is the basis of the method. Starting at an arbitrary state ρ_0, find a Hermitian matrix σ_0 that fulfills $P_0\sigma_0 P_0 = \sigma_0$, where P_0 is the projection operator onto $R(\rho_0)$. Define

$$\sigma = \sigma_0 - (\mathrm{Tr}\sigma_0)\rho_0 \tag{4.7}$$

to obtain a traceless matrix. Using this matrix define

$$\tau(x) = \rho_0 + x\sigma. \tag{4.8}$$

Define ρ_1 as $\tau(x_1)$ where x_1 is one of the two values of x for which $r(\tau(x))) < (r(\rho_0))$ and $\tau(x)$ is still positive semidefinite. Repeat the procedure using ρ_1 as a starting point. Continue this process until you can no longer find a nonzero σ. This means that it is no longer possible to find a direction where it is possible to move both ways without leaving the convex set i.e. you have reached an extreme point. Geometrically this method is equivalent to moving to flat faces on the boundary of $\mathcal{D}$ with decreasing dimensions. This geometric interpretation is very intuitive if we use a cube in $\mathbb{R}^3$ as our convex set. We start at a point inside the cube and move in a random direction until we reach one of the faces. We then move in a random direction along this surface until we reach an edge, where we chose a new random direction along the edge which we follow until we reach a corner i.e. an extreme point.

Using this method to find extremal points of $\mathcal{P}$ is very similar, but the method must in essence be applied to both ρ and its partial transposes simultaneously. Practically this means more constraints on valid σs. In the bipartite case you get the additional constraint

$$Q\sigma^P Q = \sigma^P \tag{4.9}$$

where Q is the projection operator onto the image space of ρ^P. In the tripartite case you get three constraints similar to equation (4.9), one for each partial transpose, in addition to the original constraints $P\sigma P = \sigma,\ \sigma = \sigma^\dagger,\ \sigma \neq 0$, and Tr $\sigma = 0$.

When searching for a valid σ it is useful to operate in the N^2 dimensional real vector space of $N \times N$ Hermitian matrices with the scalar product

$$\langle A, B\rangle = \langle B, A\rangle = \mathrm{Tr(AB)}. \tag{4.10}$$

On this space the transpose of a linear transformation L is defined by

$$\langle A, L^T B\rangle = \langle LA, B\rangle. \tag{4.11}$$

In this space the constraints on σ are easier to handle. Define the following linear transformation on this space

$$\mathbf{P}\sigma = P\sigma P, \quad \mathbf{Q_1}\sigma = (Q_1\sigma^{T_1}Q_1)^{T_1}, \quad \mathbf{Q_2}\sigma = (Q_2\sigma^{T_2}Q_2)^{T_2}, \quad \text{etc.} \tag{4.12}$$

It is easy to show that all these linear operators are projection operators, for example

$$\mathbf{P}^2\sigma = P(P\sigma P)P = P\sigma P = \mathbf{P}\sigma, \tag{4.13}$$

and that they are symmetric, for example

$$\langle A, \mathbf{P}B\rangle = \mathrm{Tr(APBP)} = \mathrm{Tr(PAPB)} = \langle \mathbf{P}\mathrm{A}, \mathrm{B}\rangle. \tag{4.14}$$

Proving that the other operators in equation (4.12) are symmetric is not quite as simple without knowing that

$$\mathrm{Tr(AB)} = \sum_{\mathrm{ij}} \mathrm{A_{ij}B_{ji}} = \sum_{\mathrm{ij}} \mathrm{A^*_{ji}B_{ji}} = \mathrm{Tr(A^{T_i}B^{T_i})}. \tag{4.15}$$

This is true because partial transposition merely rearranges the elements of a matrix, which does not change the value of the sum. The proof for the symmetry of $\mathbf{Q}_i$ is now

$$\langle A, \mathbf{Q}_i B\rangle = \mathrm{Tr(A(Q_iB^{T_i}Q_i)^{T_i})} = \mathrm{Tr(Q_iA^{T_i}Q_iB^{T_i})} = \mathrm{Tr((Q_iA^{T_i}Q_i)^{T_i}B)} = \langle \mathbf{Q}_\mathrm{i}\mathrm{A}, \mathrm{B}\rangle. \tag{4.16}$$

All constraints on the form of equation (4.9) can now be written as

$$\mathbf{Q_i}\sigma = (Q_i\sigma^{T_i}Q_i)^{T_i} = \sigma. \tag{4.17}$$

In the tripartite case the constraints on σ can now be described in a single eigenvalue equation:

$$(\mathbf{P} + \mathbf{Q}_1 + \mathbf{Q}_2 + \mathbf{Q}_3)\sigma = 4\sigma \tag{4.18}$$

This compound operator is symmetric because it is a sum of symmetric operators and we are therefore guaranteed that its eigenvalues are real. Since all this is done before normalizing to zero trace, you can now choose σ as an arbitrary linear combination of the eigenvectors that fulfill equation (4.18). There is always at least one eigenvector fulfilling equation (4.18) and that is $\sigma = \rho_0$. When only this eigenvector remains there is no nonzero σ with Tr $\sigma = 0$ and we have reached an extreme point.

When searching for extreme points of $\mathcal{P}$ we are no longer guaranteed that $r(\rho)$ decreases in each iteration. We are only guaranteed that the sum of ranks,

$$\mathbf{r}(\rho) = r(\rho) + r(\rho^{T_1}) + r(\rho^{T_2}) + r(\rho^{T_3}) \tag{4.19}$$

in the tripartite case, decreases.

To sum up the extremalSearch algorithm is described by the simple pseudo code below. The full code can be found in Appendix A.1.

Initial ρ
$\sigma \leftarrow createNormalizedSigma(\rho)$
while $\sigma \neq 0$ **do**
  $\rho \leftarrow findEdge(\rho, \sigma)$
  $\sigma \leftarrow createNormalizedSigma(\rho)$
end while
Return ρ

4.1.1 Locating the edge

When a direction σ has been calculated you need to find a value of x such that $\tau(x)$ is still a PPT state, but $\mathbf{r}(\tau(x)) < \mathbf{r}(\rho)$. I did this by first increasing x until $\tau(x)$ is no longer PPT using large increments and then approaching the desired value x from the outside of $\mathcal{P}$. One approach is to first choose a step size and make steps of this size while monitoring the lowest eigenvalue, λ_k, of ρ and its partial transposes. A step is valid if $\lambda_k < \lambda_{k+1} < 0.5\lambda_k < 0$. Continue making steps until the current step size no longer allows a valid step and then divide the step size by two. Continue iterating until the step size reaches some extremely small limit or the value of λ no longer changes. This method should converge to the desired value of x relatively quickly.

4.1.2 Upper limit on the ranks of extreme points

By looking at the number of constraints equation (4.18) places on σ it is possible to derive an upper limit on the ranks of extremal PPT states. The state ρ has ranks $(m, n, u, v) = (r(\rho), r(\rho^{T_1}), r(\rho^{T_2}), r(\rho^{T_3}))$. The ranks of the linear projection operators are the squares of the ranks of the corresponding matrices e.g. $r(\mathbf{P}) = m^2$. Each projection operator taken alone places a number of constraints on σ equal to the dimension of its kernel e.g. $N^2 - m^2$ for $\mathbf{P}$. Summing the the number of constraints gives an upper limit on the number n_c of linearly independent constraints: $n_c \leq 4N^2 - m^2 - n^2 - u^2 - v^2$. The dimension of the space spanned by valid solutions of equation (4.18) is the dimension of the space, N^2, minus the number of independent constraints. For an extreme point this space has dimension one which means that $n_c = N^2 - 1$. Using this we get an upper limit on the square sum of ranks for an extremal PPT state:

$$m^2 + n^2 + u^2 + v^2 \leq 3N^2 + 1 \tag{4.20}$$

We have equality when all the constraints from the various projection operators are linearly independent, but in general this rarely happens. The highest rank combinations that are allowed by this limit in the $2 \times 2 \times 2$ system, where $3N^2 + 1 = 193$, are listed in Table 4.1.

1888	193
4788	193
5688	189
5778	187
6678	185
6777	183

Table 4.1: A list of the highest rank combinations allowed by equation (4.20), and the corresponding square sums

4.1.3 Application as separability check for states with low ranks

A slightly expanded version of this method can be used to write an arbitrary PPT state as a convex combination of extremal PPT states of lower total rank. Given a starting point ρ and a search direction σ it is possible to write ρ as a convex combination of the two edge points i.e. the two points where the line defined by ρ and σ hits the surface of $\mathcal{P}$. These points are easily found using the method described above. If these edge points are not extremal the procedure can be repeated for them and so on until extreme points are reached. If this algorithm returns a convex combination of only pure states the state in question is separable. This is not unlikely for low rank PPT states where few mixed extremal states of lower rank exist.

Pseudo code for a recursive implementation of this algorithm that I have called extremalDecomp is shown below. Note that this implementation only locates the extremal states used in the convex combination and not the weights, but it could easily be modified to do so.

$\rho \leftarrow Input$
Create empty list
$\sigma \leftarrow createNormalizedSigma(\rho)$
if $\sigma \neq 0$ **then**
  $\rho_1 \leftarrow findEdge(\rho, \sigma)$
  $\rho_2 \leftarrow findEdge(\rho, -\sigma)$
  Add $extremalDecomp(\rho_1)$ *to list*
  Add $extremalDecomp(\rho_2)$ *to list*
else
  Add ρ *to list*
end if
Return list

4.2 Searching for states with specified ranks

My algorithm to find a PPT state with specified ranks, rankSearch, is an adapted version of the algorithm presented in the article [1]. We continue to utilize the N^2 dimensional real vector space of Hermitian matrices. As an orthonormal basis for this space we introduce a set of matrices, $\{M_i\}$, which obey the equations

$$\mathrm{Tr}(\mathrm{M_i M_j}) = \delta_{\mathrm{ij}}. \tag{4.21}$$

The relationship between the real vector $\mathbf{x}$ representing the matrix ρ is now given by

$$x_i = \mathrm{Tr}(\rho \mathrm{M_i}) \tag{4.22}$$

and ρ can be expressed as

$$\rho = \rho(\mathbf{x}) = \sum_i x_i M_i. \tag{4.23}$$

The rank of a matrix is equal to the number of non-zero eigenvalues. If we are looking for a PPT state with $r(\rho) = k$ and $r(\rho^P) = l$ the $N - k$ lowest eigenvalues of ρ and $N - l$ lowest eigenvalues of ρ^P must be zero. The same applies to the rest of the partial transposes in the multipartite case. We now create a vector μ where the components are all the eigenvalues that should be zero i.e. the $N - k$ lowest eigenvalues of ρ, the $N - l$ lowest eigenvalues of ρ^P etc. The state we are looking for is now the solution of the problem $\mu(\mathbf{x}) = 0$. This problem can be solved by repeatedly solving the first order Taylor expansion:

$$\mu(\mathbf{x}) + (\Delta\mathbf{x} \cdot \nabla)\mu(\mathbf{x}) = 0. \tag{4.24}$$

The problem can be reformulated as

$$B\Delta\mathbf{x} = -\mu \tag{4.25}$$

where the components of B are given by $B_{ij} = \partial\mu_i/\partial x_j$. To simplify the problem even further we introduce the positive, real and symmetric matrix $A = B^T B$ and the vector $\mathbf{b} = -B^T\mu$, which gives the formulation

$$A\Delta\mathbf{x} = \mathbf{b}, \tag{4.26}$$

which can easily be solved with the conjugate gradient method. The components of B can be calculated using first order perturbation theory

$$\frac{\partial\lambda_k}{\partial x_j} = \psi_k^\dagger \frac{\partial\rho}{\partial x_j}\psi_k = \psi_k^\dagger M_j \psi_k. \tag{4.27}$$

The vector ψ_k is the eigenvector corresponding to the eigenvalue λ_k. The corresponding formulas are valid for the partial transposes. This expression is based on nondegenerate perturbation theory and we are actually looking for a degenerate state, but this method nevertheless works well in practice. MATLAB code for an implementation of this algorithm can be found in Appendix A.2

It is possible for this algorithm to return a state with lower ranks than requested i.e. there is a larger number of zero eigenvalues than we require. However if the specified combination of ranks exist changing the starting point will likely solve this problem.

4.3 Initial results

Since I operate with three identical subsystems they can always be relabeled to interchange the roles of ρ and its partial transposes. This makes all permutations of a particular set of ranks equivalent e.g. (5,5,6,6) and (6,6,5,5). I ran rankSearch several times for all possible rank combinations and then ran extremalSearch using these states as starting points. I also did repeated searches using the identity matrix as the starting point.

PPT states where found for most high rank combinations, but if at least one rank was four or lower only states where all ranks are equal were found i.e. rank (4,4,4,4), (3,3,3,3), (2,2,2,2) and (1,1,1,1). The rank one states are pure product states. The rank two and three states were tested for separability and found to be separable. Of the combinations where all ranks are greater than four only (5,5,6,8), (5,5,8,8), and (5,8,8,8) were not found. These are among the high rank combinations with the greatest difference between ranks. These combinations may not exist, but it is also possible that they are only difficult to find using the rankSearch algorithm. I found extremal PPT states for all combinations with ranks equal or greater then four up to the limit derived in Section 4.1.2, with one exception (5,6,8,8). I consider it likely that extremal states of this rank combination exist. Its absence from the data is likely due to an insufficient number of applications of extremalSearch using a state of suitable rank as the starting point.

The searches starting at the identity all resulted in ranks (6,7,7,7) and (6,6,7,8) as the result of five iterations, each reducing a single rank by one. These are among the extremal states with highest total rank $\mathbf{r}$. Only bested by (5,7,8,8) and equaled by (5,7,7,8). The only other state reachable by five reductions of a single rank by one is (5,6,8,8), but this combination was not found. I also did searches starting at the identity for systems of dimensions $2 \times 2 \times 3$ and $3 \times 3 \times 3$, which resulted in ranks (10,10,10,11) and (9,10,11,11) and (23,23,23,24) and (22,23,24,24) respectively. Like in the $2 \times 2 \times 2$ case these are among the extremal states with highest total rank $\mathbf{r}$ allowed by the upper limit in equation (4.20).

The extremal $2 \times 2 \times 2$ states generated by starting at the identity were also examined for extremality in the sets $\mathcal{D} \cap \mathcal{D}^{T_i} \cap \mathcal{D}^{T_j}$, where $i, j = 1, 2, 3$ and $i \neq j$, in addition to $\mathcal{P} = \mathcal{D} \cap \mathcal{D}^{T_1} \cap \mathcal{D}^{T_2} \cap \mathcal{D}^{T_3}$. I found that states where $r(\rho^{T_i}) = 8$ were also extremal in $\mathcal{D} \cap \mathcal{D}^{T_j} \cap \mathcal{D}^{T_k}$ where $j \neq i$, $k \neq i$, and $j \neq k$.

These results are similar to what was found in [1], where similar methods were used to study bipartite systems of low dimensions. In both cases the states with the lowest ranks, up to some rank k, are states where all the ranks are equal. PPT states are found for almost all combinations of ranks higher than k, excluding only the combinations where the difference between ranks are greatest for some systems. And also here extremal states where found for all states with ranks above k. The separability properties for the states with rank k and $k-1$ vary for systems with different dimensions, but the 3×3, 3×4, and 3×5 systems all share the same structure as my $2 \times 2 \times 2$ system i.e. all states with ranks below k are separable while states of rank k can be extremal.

Chapter 5

A closer look at extremal rank (4,4,4,4) PPT states

In this chapter I study the extremal rank (4,4,4,4) PPT states in more detail. The extremal rank four states generated using the algorithms described in the previous chapter were confirmed to be biseparable. The range and kernel of the states themselves and their partial transposes were examined for product vectors, but none were found. I then show how an equivalence class of extremal rank four states can be created analytically using a UPB construction. Because these states contain product vectors in their kernel, but not in their range by design we know that there must be more than one equivalence class of extremal rank four states. I continue with a method to construct extremal rank four states that are symmetric under various combinations of partial transposes. Finally I construct a set of quantities that are invariant under SL⊗SL transformations and therefore have the same value for all states in the same equivalence class. Calculating the values of these quantities for all generated states show that there is likely an infinite number of equivalence classes described by one or more continuous variables. These invariants also reveal an interesting set of states which may belong to a single equivalence class, where one invariant quantity is zero and the others equal.

5.1 General considerations

The rank four extremal PPT states are of special interest because they are the mixed extremal states of lowest rank. As was shown in the section presenting the extremalSearch algorithm, all non extremal PPT states can be written as convex combinations of lower rank extremal PPT states. The only extremal states with rank lower than the rank four states are the pure product states. This means that an arbitrary rank four PPT state is either extremal or separable.

According to the numerical results of [1] all rank (4,4) states of a bipartite 2×4 or 4×2 system are separable. If this is true all rank (4,4,4,4) states must be biseparable. You can find a theoretical derivation of this result in [12]. I have checked the biseparability of all

my extremal rank four states and they are biseparable as expected.

An important property of a state ρ is the number of product vectors that can be found in $R(\rho)$ and $K(\rho)$. Among other things it is related to a separability criterion known as the range criterion. The bipartite version of the range criterion states that the range, $R(\rho)$, of any separable state contains and is spanned by a set of product vectors $\{\psi_i \otimes \varphi_i\}$ and the range of the partial transpose, $R(\rho^P)$, must be spanned by the corresponding set of product vectors $\{\psi_i \otimes \varphi_i^*\}$. The generalization to multipartite systems is easily done by adding similar criteria for all partial transposes.

By using parameter counting it is possible to give give an educated guess as to the existence of product vectors in a generic subspace of dimension d. Let us look at a composite system of dimension N with subsystems of dimensions N_A, N_B and N_C. Each subsystem contributes $N_i - 1$ free complex parameters after normalization resulting in a total number of $N_A + N_B + N_C - 3$ complex parameters for an arbitrary product vector. The restriction to a particular subspace implies a number of constraint equations

$$\psi_k^\dagger(\varphi \otimes \chi \otimes \eta) = 0, \quad k = 1, 2, ..., N - d, \tag{5.1}$$

where $\{\psi_k\}$ is a set vectors spanning the orthogonal complement of the chosen d-dimensional subspace. Subtracting a complex parameter for each complex constraint equation leaves us with p free complex parameters with

$$p = N_A + N_B + N_C - 3 - N + d. \tag{5.2}$$

If $p > 0$ the problem is underdetermined i.e. there is an infinite number of product vectors in the d dimensional subspace characterized by p free complex parameters. If $p < 0$ the problem is overdetermined and we expect no solutions in the generic case. However it is still possible for special d dimensional subspaces to contain product vectors. When $p = 0$ the number of constraints and available parameters matches exactly and we have a finite set of solutions. For our three qubit system this means that we expect a finite set of solutions in subspaces of dimension $d = 5$ e.g. the range of rank 5 states or the kernel of rank 3 states. Based on parameter counting we do not expect that the range or kernel of the rank four states and their partial transposes contain product vectors.

All product vectors in a specified subspace are global solutions of the problem

$$\begin{aligned} \min_{\varphi,\chi,\eta} \ & (\varphi \otimes \chi \otimes \eta)^\dagger P (\varphi \otimes \chi \otimes \eta) \\ \text{s.t.} \ & \varphi^\dagger \varphi = 1 \\ & \chi^\dagger \chi = 1 \\ & \eta^\dagger \eta = 1 \end{aligned} \tag{5.3}$$

where P is the projection operator onto the orthogonal complement of the specified subspace. Since P is a positive semidefinite operator the global minima correspond to a function value of zero and all these minima are product vectors in the specified subspace. I tried solving this problem with three different numerical methods: generalizations of the

sequential quadratic programming (SQP) and conjugate gradient methods presented in my project [16] and a generalization of the double eigenvalue problem presented in Appendix A of [1]. You can find the MATLAB code for these methods in Appendix B.

None of these methods found any product vectors in the range or kernel of any of the rank four extremal states or their corresponding partial transposes.

5.2 UPB construction

Of the bipartite systems studied in [1] the "rank structure" of the $2 \times 2 \times 2$ system is most similar to the bipartite 3×3 system. In both systems the extremal states with lowest nontrivial rank are the rank 4 states. In the 3×3 case the range of the rank 4 states contained no product vectors while the kernel contained a complete set. This property led to the parametrization of the rank 4 states using unextendible product bases (UPBs).

An unextendible product basis is a product basis of some space $\mathcal{H}_S$, which is a subspace of a Hilbert space $\mathcal{H}$, which makes it impossible to find a single product vector in the orthogonal complement of the space spanned by the UPB, $(\mathcal{H}_S)^\perp = \mathcal{H} - \mathcal{H}_S$, i.e. it is impossible to find a product vector that is orthogonal to all the vectors in the UPB. It is shown in [17] that states with kernels spanned by a UPB contain bound entanglement. Entanglement of this kind is also found in extremal PPT states. The same article also describes the properties and construction of UPBs in more detail.

The UPB parametrization of the extremal rank four states of the 3×3 system is discussed in detail in [18], but I will repeat the highlights here. A five dimensional orthogonal UPB for the 3×3 system is parametrized by four real, positive and continuous parameters. While the UPBs in the kernels of the rank four states are not necessarily orthogonal, all of them can be transformed to states where the UPB is orthogonal using SL$\otimes$SL transformations. This means that equivalence classes of the extremal rank (4,4) states are parametrized by the four previously mentioned parameters.

Hoping to find a similar structure for the extremal rank (4,4,4,4) states I looked into the construction of a four dimensional orthogonal UPB for the $2 \times 2 \times 2$ system. For two product vectors to be orthogonal only the vectors representing one of the subsystems need to be orthogonal i.e.

$$(\varphi \otimes \chi \otimes \eta)^\dagger (u \otimes v \otimes w) = (\varphi^\dagger u)(\chi^\dagger v)(\eta^\dagger w) = 0 \tag{5.4}$$

is true if

$$\varphi^\dagger u = 0 \ \vee \ \chi^\dagger v = 0 \ \vee \ \eta^\dagger w = 0. \tag{5.5}$$

In addition to the requirement of orthogonality the vectors of all three subsystems must be linearly independent respectively for two arbitrary chosen product vectors. This implies that the same vector cannot be used twice for one subsystem in two different product vectors. Using the qubit states

$$|1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad |+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{5.6}$$

I constructed the UPB

$$\{\psi_i\} = \{|1,1,1\rangle, |0,+,+\rangle, |+,0,-\rangle, |-,-,0\rangle\}. \tag{5.7}$$

This is not the only UPB you can construct, but any four dimensional three qubit UPB can be transformed into all other such UPBs using an SL⊗SL transformation. I will give an explicit example and show how the UPB presented in [17],

$$\{\varphi_i\} = \{|0,1,+\rangle, |1,+,0\rangle, |+,0,1\rangle, |-,-,-\rangle\}, \tag{5.8}$$

can be transformed into the one I constructed. In this case both UPBs used the same four qubit vectors, but even if two UPBs were constructed using different vectors they would still be SL⊗SL equivalent. I am now looking for matrices $A, B, C \in \mathrm{SL}(2,\mathbb{C})$ such that

$$\{\psi_i\} = \{k_i(A \otimes B \otimes C)\varphi_i\}, \tag{5.9}$$

where k_i is a complex constant.

To determine A we have the equations

$$A|0\rangle = \alpha_1|1\rangle \quad A|1\rangle = \alpha_2|0\rangle \quad A|+\rangle = \alpha_3|+\rangle \quad A|-\rangle = \alpha_4|-\rangle \tag{5.10}$$

where α_i are arbitrary complex constants. The first two equations limit A to the form

$$\begin{pmatrix} 0 & \alpha_1 \\ \alpha_2 & 0 \end{pmatrix}. \tag{5.11}$$

The third equation implies $\alpha_1 = \alpha_2 = \alpha_3$ and the fourth equation requires $\alpha_1 = -\alpha_4$. Choosing $\alpha_1 = 1$ we get

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{5.12}$$

To determine B we have the equations

$$B|1\rangle = \beta_1|1\rangle \quad B|+\rangle = \beta_2|+\rangle \quad B|0\rangle = \beta_3|0\rangle \quad B|-\rangle = \beta_4|-\rangle \tag{5.13}$$

which are easily solved by choosing $\beta_i = 1$ and $B = I$. The equations determining C are

$$C|+\rangle = \gamma_1|1\rangle \quad C|0\rangle = \gamma_2|+\rangle \quad C|1\rangle = \gamma_3|-\rangle \quad C|-\rangle = \gamma_4|0\rangle \tag{5.14}$$

The second and third equations limit C to the form

$$\frac{1}{\sqrt{2}}\begin{pmatrix} \gamma_3 & \gamma_2 \\ -\gamma_3 & \gamma_2 \end{pmatrix}. \tag{5.15}$$

The remaining equations imply $\gamma_1 = \gamma_2 = \gamma_3$ and $\gamma_1 = -\gamma_4$. Choosing $\gamma_1 = 1$ we get

$$C = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}. \tag{5.16}$$

The constants k_i in equation (5.9) are given by $k_i = \alpha_i\beta_i\gamma_i$, which means that $k_i = 1$. We have now showed that the UPB $\{\varphi_i\}$ can be transformed into the UPB $\{\psi_i\}$ with the transformation

$$\frac{1}{\sqrt{2}}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\otimes\begin{pmatrix}1 & 1\\ -1 & 1\end{pmatrix}. \tag{5.17}$$

To construct a state with a UPB, $\{\psi_i\}$, contained in its kernel simply create the normalized projection operator onto the orthogonal complement of the UPB i.e.

$$\rho = \frac{1}{N-d}\left(I - \sum_{i=1}^{d}\psi_i\psi_i^\dagger\right). \tag{5.18}$$

The kernels of the partial transposes of ρ will contain the corresponding UPBs where the vectors of the transposed subsystem are complex conjugated. Because I have chosen a real basis the kernels of ρ and its partial transposes contain identical UPBs. The state ρ and all states created from ρ via SL⊗SL transformations are extremal rank four PPT states.

Numerical examinations of the extremal rank four states found via the systematic search presented in the previous chapter showed no product vectors in any kernels. This implies that there are at least two equivalence classes of extremal rank four states that cannot be converted into each other using SL⊗SL transformations. An interesting question is what the fundamental difference between these classes are.

5.3 States symmetric under partial transpositions

Another way to classify states are which symmetries they possess. It is possible that the extremal rank (4,4,4,4) states could belong to equivalence classes exemplified by states with particular symmetries. The first symmetries that come to mind are symmetry under one or more of the partial transposes. I looked at symmetry under all three partial transposes first, which also implies symmetry under the regular transpose and makes the matrices real. A general $2\times2\times2$ state is described by 64 real parameters before normalization. Imposing these symmetry requirements reduces that number to 27. Creating PPT states with these symmetry properties is not difficult using numerical methods. Simply minimize the square of the four lowest eigenvalues of a generic state with the required symmetry properties. MATLAB code for an implementation of this method can be found in Appendix A.3. I created several hundred rank four states using this method and they were all extremal.

It is also possible to relax the symmetry requirements a bit and only require symmetry under two partial transposes. This raises the number of free parameters describing a generic state to 36 and allows the matrices to be complex, but states generated using the method described above are still guaranteed to be PPT because the excluded partial transpose can be created by sequentially applying the other two partial transposes and a regular transpose. The state is still symmetric under the first two operations and the regular transpose does not change eigenvalues so we still end up with a state with four identical sets of eigenvalues. Only about 75 % of the states generated using this method

and these relaxed symmetry requirements are extremal. All states generated using this method were examined for product vectors in both range and kernel using the numerical methods mentioned in Section 5.1, but none were found.

If we only require symmetry under a single partial transpose we are no longer guaranteed that the produced states will be PPT using the method presented above. I did not pursue this case.

5.4 Invariants

As demonstrated by the existence of extremal rank (4,4,4,4) PPT states with and without product vectors in their kernel, we have at least two equivalence classes of such states. In order to explore the differences of these classes I have constructed some quantities that are invariant under SL⊗SL transformations. These quantities will be referred to as invariants, but all states in the same equivalence class will not produce the same values for these invariants. Remember that a physical transformation consists of an SL⊗SL transformation and normalization to unit trace. The normalization will change the value of the invariants, but it is possible to compensate for this by looking at the ratio of two invariants of the same order. In these ratios, which I will call normalized invariants, the normalization factor accompanying the SL⊗SL transformation cancels out and all states in an equivalence class will produce the same values.

This approach to constructing invariants utilizes a property unique to two dimensional systems and can therefore not be directly applied to higher dimensional systems. SL⊗SL transformations preserve the determinant of any state. The determinant of the Hermitian matrix

$$A = \begin{pmatrix} a+b & c-id \\ c+id & a-b \end{pmatrix} \tag{5.19}$$

is $a^2 - b^2 - c^2 - d^2$, which is equal to the inner product of a four-vector

$$x = (a, b, c, d)^T \tag{5.20}$$

with itself i.e.

$$x^2 = x^\mu x_\mu = \eta_{\mu\nu} x^\mu x^\nu = x^T \eta x = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = a^2 - b^2 - c^2 - d^2. \tag{5.21}$$

Here η is the Minkowski metric and we use the Einstein summation convention i.e. repeated Greek indices imply a sum from zero to three and repeated Latin indices imply a sum from one to three. Inner products are preserved under Lorentz transformations and it is therefore possible to represent any SL transformation of a Hermitian 2 by 2 matrix by a Lorentz transformation of the corresponding four-vector. The conversion from Hermitian matrix to

real four-vector is easily formulated using the Pauli matrices as an orthogonal basis. The Pauli matrices are

$$\sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{5.22}$$

and obey the relation

$$\sigma_i \sigma_j = \delta_{ij}\sigma_0 + i\epsilon_{ijk}\sigma_l, \quad i,j,k = 1,2,3 \tag{5.23}$$

where ϵ is the antisymmetric Levi-Cevita tensor . The matrix A can be written as

$$A = x^\mu \sigma_\mu \tag{5.24}$$

where x^μ is the equivalent four-vector whose components are given as

$$x^\mu = \frac{1}{2}\mathrm{Tr}(\mathrm{A}\sigma_\mu). \tag{5.25}$$

In an equivalent way the three qubit state ρ can be written as

$$\rho = \rho^{\mu\nu\lambda}(\sigma_\mu \otimes \sigma_\nu \otimes \sigma_\lambda) \tag{5.26}$$

where the components $\rho^{\mu\nu\lambda}$ are given by

$$\rho^{\mu\nu\lambda} = \frac{1}{8}\mathrm{Tr}(\rho\,(\sigma_\mu \otimes \sigma_\nu \otimes \sigma_\lambda)). \tag{5.27}$$

The tensor $\rho^{\mu\nu\lambda}$ will serve as the basis for my invariants, but because partial transposition is equivalent to a Lorentz transformation using $(\rho^{T_i})^{\mu\nu\lambda}$ would produce identical values. Each index represents a different subsystem and can therefore only be contracted with the corresponding index on another tensor. This leaves us with only one second order invariant:

$$I_2 = \eta_{\mu\alpha}\eta_{\nu\beta}\eta_{\lambda\gamma}\rho^{\mu\nu\lambda}\rho^{\alpha\beta\gamma} = \rho^{\mu\nu\lambda}\rho_{\mu\nu\lambda} \tag{5.28}$$

There are five independent fourth order invariants corresponding to different combinations of contracted indices, but one of these is simply the square of the second order invariant. The four new invariants can be written as

$$I_{4,12} = \eta_{\mu\alpha}\eta_{\nu\beta}\eta_{\lambda\tau}\eta_{\delta\kappa}\eta_{\epsilon\xi}\eta_{\zeta\gamma}\rho^{\mu\nu\lambda}\rho^{\alpha\beta\gamma}\rho^{\delta\epsilon\zeta}\rho^{\kappa\xi\tau} = \rho^{\mu\nu\lambda}\rho_{\mu\nu\zeta}\rho^{\delta\epsilon\zeta}\rho_{\delta\epsilon\lambda} \tag{5.29}$$

$$I_{4,21} = \eta_{\mu\alpha}\eta_{\nu\xi}\eta_{\lambda\gamma}\eta_{\delta\kappa}\eta_{\epsilon\beta}\eta_{\zeta\tau}\rho^{\mu\nu\lambda}\rho^{\alpha\beta\gamma}\rho^{\delta\epsilon\zeta}\rho^{\kappa\xi\tau} = \rho^{\mu\nu\lambda}\rho_{\mu\epsilon\lambda}\rho^{\delta\epsilon\zeta}\rho_{\delta\nu\zeta} \tag{5.30}$$

$$I_{4,22} = \eta_{\mu\alpha}\eta_{\nu\xi}\eta_{\lambda\tau}\eta_{\delta\kappa}\eta_{\epsilon\beta}\eta_{\zeta\gamma}\rho^{\mu\nu\lambda}\rho^{\alpha\beta\gamma}\rho^{\delta\epsilon\zeta}\rho^{\kappa\xi\tau} = \rho^{\mu\nu\lambda}\rho_{\mu\epsilon\zeta}\rho^{\delta\epsilon\zeta}\rho_{\delta\nu\lambda} \tag{5.31}$$

$$I_{4,23} = \eta_{\mu\alpha}\eta_{\nu\xi}\eta_{\lambda\zeta}\eta_{\delta\kappa}\eta_{\epsilon\beta}\eta_{\tau\gamma}\rho^{\mu\nu\lambda}\rho^{\alpha\beta\gamma}\rho^{\delta\epsilon\zeta}\rho^{\kappa\xi\tau} = \rho^{\mu\nu\lambda}\rho_{\mu}{}^{\beta\zeta}\rho^{\delta}{}_{\nu\zeta}\rho_{\delta\beta\lambda} \tag{5.32}$$

I have used these four fourth order invariants divided by the square of the second order invariant as my normalized invariants.

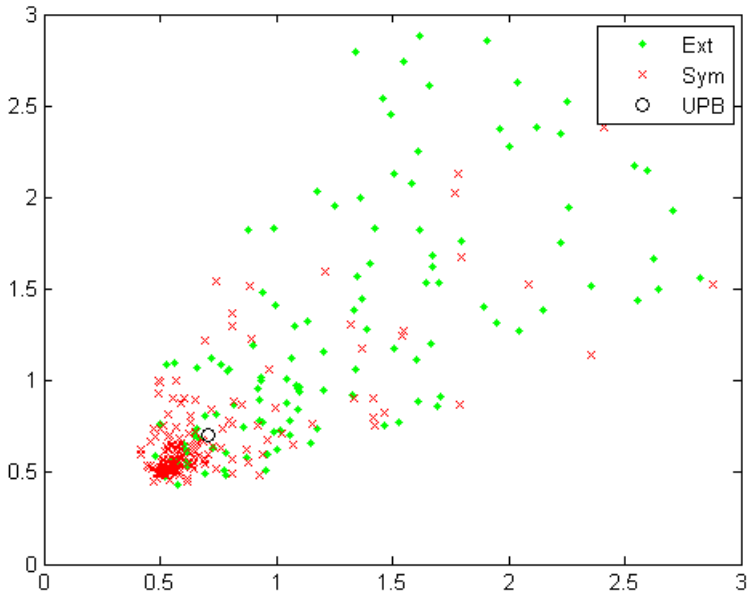
To test that my normalized invariants actually are invariant under SL⊗SL transformations I first calculated their values for the state I created using the UPB construction

and SL⊗SL transformations of this state. The greatest difference between two value of the same invariant was of the order of 10^{-9} and the greatest difference between the values of a transformed state and the original state was of the order of 10^{-12}. These small errors are likely due to numerical anomalies and I consider my normalized invariants to be unchanged by SL⊗SL transformations.

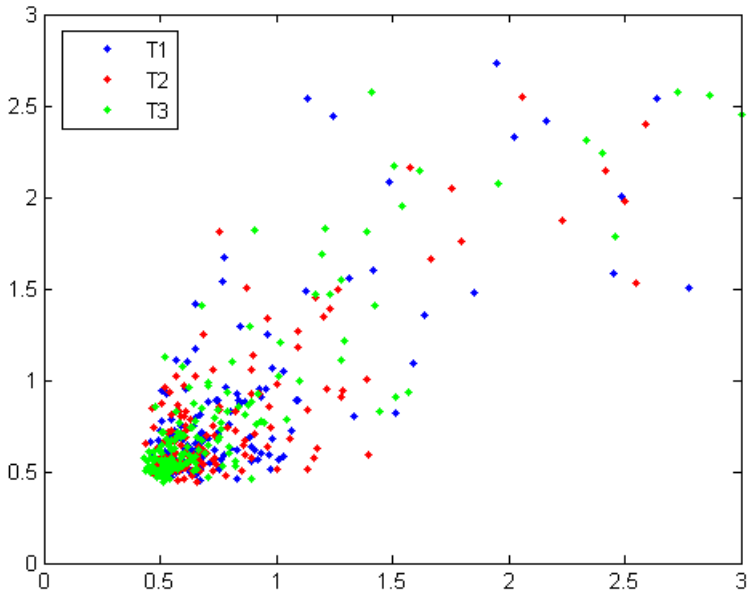
The values of the normalized invariants were calculated for the states found during my general search and with the symmetries discussed in the previous section i.e. symmetric under all partial transpositions and combinations of any two partial transpositions. Plots of the first against the second and the third against the fourth normalized invariants are shown in Figure 5.1. The invariants from the UPB state is also included in the plot as a single data point. These plots are focused on the interval where most of the data points are located, but there are also some points for higher values that are outside the interval shown in the plot in particular for the asymmetric and completely symmetric states. This is shown in the plot of all of the third against the fourth normalized invariants for the partially symmetric states is shown in Figure 5.2a. The values of the invariants seem to vary continuously. This is even more clear when looking at Figure 5.2b where the sorted values of the first normalized invariants are shown for all state types. This continuous variation of invariants suggests that we have an infinite set of equivalence classes described by one or more continuous variables.

There seems to be no obvious difference between the types of states. The invariants seem to span the same range of values and be correlated in the same way for all the different state types. The correlations between the different invariants were studied by calculating the covariance matrices and their eigenvectors and eigenvalues for each state type. For all state categories the covariance matrix had one dominant eigenvalue with the remaining ones very close to zero in comparison. The dominant eigenvalue corresponded to an eigenvector very close to (0.5 0.5 0.5 0.5). This indicates that the invariants are lie very close to a line in a four dimensional space. This fits well with the plots in Figure 5.1. Here the data is projected onto a two dimensional space where it lies roughly along a line. Due to the lack of differences between the state categories it is possible that one of the categories could be used as a standard form for all extremal rank four state, meaning that all states can be transformed to the standard form using SL⊗SL transformations.

There is a small subset of the data that does not appear in any of the figures. For nine of the states without any specific symmetry the quadratic invariant has a value of zero. This is interesting because the remaining values of the quadratic invariant do not fall below 10^{-2} so there is not a continuous descent to zero. They are also interesting because for each of these state all the fourth order invariants have the same value indicating that these states belong to the same equivalence class, something that does not occur anywhere else in the data. There is obviously something special about this equivalence class, but we do not know what, apart from the special value of the quadratic invariant.

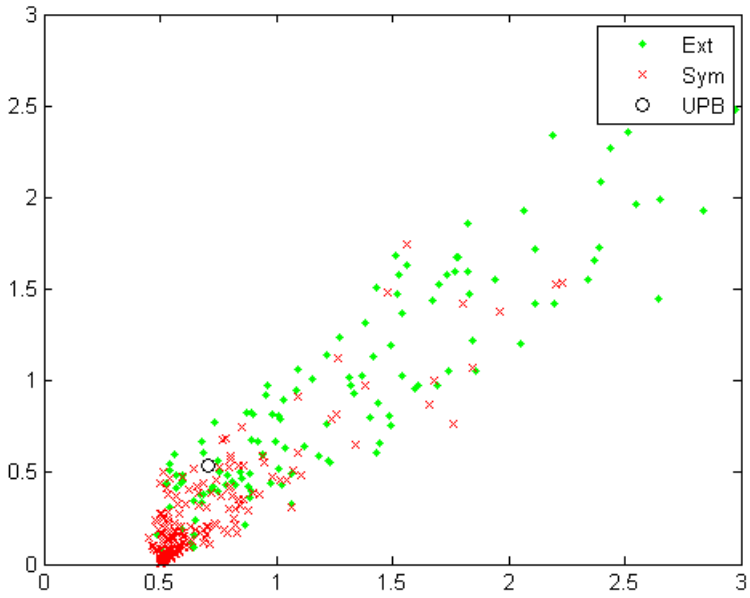


(a) Plot of the first invariant against the second invariant for the original extremal rank four states (Ext), the completely symmetric rank four states (Sym) and the UPB state (UPB).

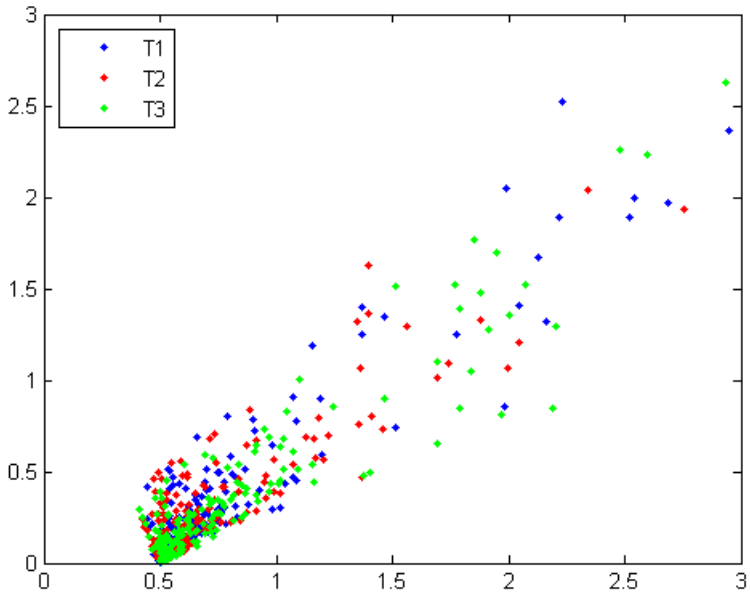


(b) Plot of the first invariant against the second invariant for the states which are symmetric under two partial transposes. They are labeled with the symmetry they do not possess.

Figure 5.1

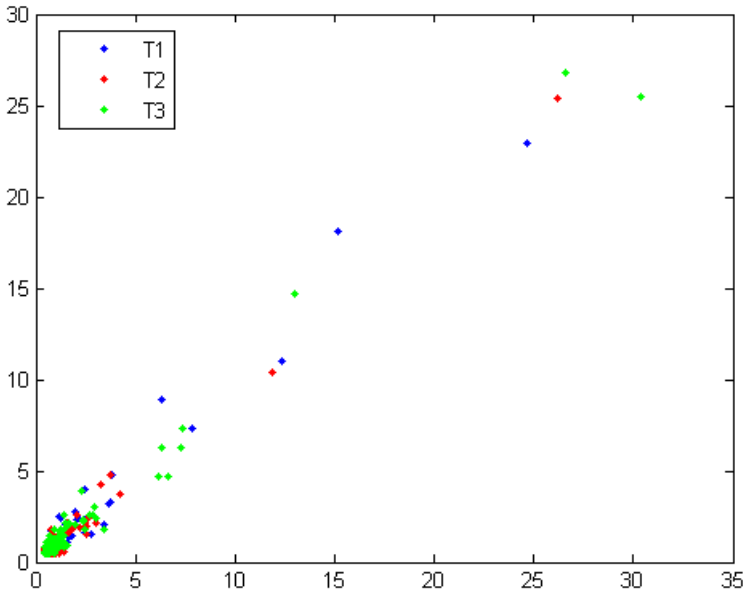


(c) Plot of the third invariant against the fourth invariant for the original extremal rank four states (Ext), the completely symmetric rank four states (Sym) and the UPB state (UPB).

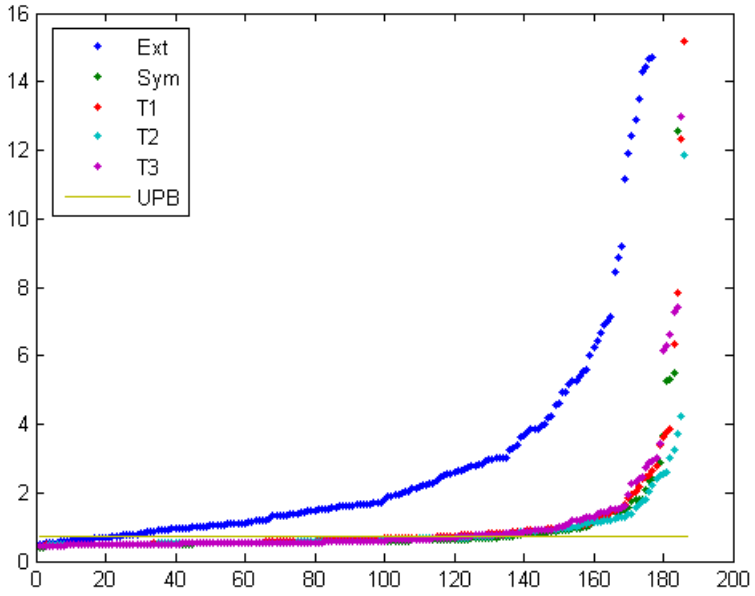


(d) Plot of the third invariant against the fourth invariant for the states which are symmetric under two partial transposes. They are labeled with the symmetry they do not possess.

Figure 5.1



(a) Plot of the first invariant against the second invariant for the states which are symmetric under two partial transposes. They are labeled with the symmetry they do not possess



(b) Plot of the first invariant against the second invariant for the states which are symmetric under two partial transposes. They are labeled with the symmetry they do not possess.

Figure 5.2

Chapter 6

Summary and further work

I have presented numerical methods that can be used to locate PPT states of specific rank and extremal PPT states. I have used these methods to systematically search for extremal PPT states in the three qubit system and found states of a variety of rank combinations. For ranks up to and including four only states of equal rank were found. All rank one states are extremal pure product states. For ranks two and three only separable states were found in concordance with theory. The first mixed extremal states were found at rank four. Above rank four I found extremal PPT states for almost all rank combinations up to an upper limit on the ranks of extremal PPT states which I derived in Section 4.1.2. The set of rank combinations where extremal states were found is similar to the corresponding sets for bipartite systems of dimension 3×3.

The most natural classification of PPT states is classification into SL$\otimes$SL equivalence classes. The defining characteristic of these equivalence classes is that all states in a class can be transformed into all other states by SL$\otimes$SL transformations followed by normalization to unit trace. I studied the extremal rank four states in more detail in the hope of determining a parametrization of the equivalence classes of these states. I first hoped to achieve this goal by using a UPB construction. This UPB construction succeeded in producing extremal rank four states, but an examination of the initial non UPB states for product vectors in their range and kernel revealed that the UPB construction could not possibly create all extremal rank four states. As an alternative approach I then constructed extremal rank four states that are symmetric under varying combinations of partial transposes. To assess whether these symmetric states could be used as a standard form for all extremal rank four states I constructed quantities that are equal for all states in the same equivalence class. The values of these invariants were calculated for all states, but the available data was not sufficient to confirm or refute that the symmetric states can be used as a standard form for all extremal rank four states. The data does indicate that there is an infinite number of equivalence classes described by one or more continuous variables. One avenue that could be pursued to determine whether the symmetric states can be used as a standard form is to look for SL$\otimes$SL transformations that can transform an arbitrary state into a symmetric state.

The calculation of these invariants revealed a small group of especially interesting states

where the quadratic invariant was zero and the fourth order invariants all had the same value. These were also the only set of states likely to belong to the same equivalence class. Discovering what makes these states special is an obvious goal for further studies.

Another possible direction for continued studies is to look at the dimension and geometry of the surfaces of PPT states of a particular rank. Such studies have been conducted in the bipartite case and are described in [19]. One could also perform studies similar to what was done in this thesis for higher dimensional systems, in particular tripartite systems where each subsystem is not of the same dimension and multipartite systems consisting of more subsystems. It is worth noting that the dimension of the complete Hilbert space increases quickly with the dimension and number of subsystems, which increases the computational power required to run the algorithms presented in this thesis.

Appendix A

Matlab source code

A.1 extremalSearch

```
n1=2;
n2=2;
n3=2;
n=n1*n2*n3;
N=n*n;
id=eye(N);
tol=1e-13;

% rho=eye(n)/n;
rho=states{1};

iter=0;
while true
    iter=iter+1;
    createSigma
    %If rho is an extremal point, stop.
    if index==(N-1)
        break
    end
    stepsize=0.1;
    approachEdge
end
```

A.1.1 createSigma

```
%Calculate the partial transposes of rho, their eigenvalues, eigenvectors
%and ranks.
rhoP1 = deltransponer(rho,n1,n2*n3);
rhoP2 = deltransponer(deltransponer(rho,n1,n2*n3),n1*n2,n3);
rhoP3 = deltransponer(rho,n1*n2,n3);
[eve ,eva ] = eig(rho);
[eveP1,evaP1] = eig(rhoP1);
[eveP2,evaP2] = eig(rhoP2);
```

```
[eveP3,evaP3] = eig(rhoP3);
rr=rank(rho,tol);
rrP1=rank(rhoP1,tol);
rrP2=rank(rhoP2,tol);
rrP3=rank(rhoP3,tol);

% For rho and its partial transposes calculate the projection operator on
% the image space.
P = eve(:,n-rr+1:n)*eve(:,n-rr+1:n)';
P = 0.5*(P+P');
PX1 = eveP1(:,n-rrP1+1:n)*eveP1(:,n-rrP1+1:n)';
PX1 = 0.5*(PX1+PX1');
PX2 = eveP2(:,n-rrP2+1:n)*eveP2(:,n-rrP2+1:n)';
PX2 = 0.5*(PX2+PX2');
PX3 = eveP3(:,n-rrP3+1:n)*eveP3(:,n-rrP3+1:n)';
PX3 = 0.5*(PX3+PX3');

% Calculate the real matrix representation of the transformation P*rho*P,
% and the respective transformations for the various partial transposes.
PP = zeros(N,N);
for ii=1:N
  AA = rvec2hmat(id(:,ii),n);
  BB = P*AA*P;
  BB = 0.5*(BB+BB');
  PP(:,ii) = hmat2rvec(BB,n);
end
PP = 0.5*(PP+PP');

PPX1 = zeros(N,N);
for ii=1:N
  AA = rvec2hmat(id(:,ii),n);
  AA = deltransponer(AA,n1,n2*n3);
  BB = PX1*AA*PX1;
  BB = 0.5*(BB+BB');
  CC = deltransponer(BB,n1,n2*n3);
  PPX1(:,ii) = hmat2rvec(CC,n);
end
PPX1 = 0.5*(PPX1+PPX1');

PPX2 = zeros(N,N);
for ii=1:N
  AA = rvec2hmat(id(:,ii),n);
  AA = deltransponer(deltransponer(AA,n1,n2*n3),n1*n2,n3);
  BB = PX2*AA*PX2;
  BB = 0.5*(BB+BB');
  CC = deltransponer(deltransponer(BB,n1,n2*n3),n1*n2,n3);
  PPX2(:,ii) = hmat2rvec(CC,n);
end
PPX2 = 0.5*(PPX2+PPX2');

PPX3 = zeros(N,N);
```

```
for ii=1:N
  AA = rvec2hmat(id(:,ii),n);
  AA = deltransponer(AA,n1*n2,n3);
  BB = PX3*AA*PX3;
  BB = 0.5*(BB+BB');
  CC = deltransponer(BB,n1*n2,n3);
  PPX3(:,ii) = hmat2rvec(CC,n);
end
PPX3 = 0.5*(PPX3+PPX3');


PPP = PP+PPX1+PPX2+PPX3;
[vecs e]=eig(PPP);

% Create sigma as a random linear combination of the eigenvectors of PPP
% that correspond to eigenvalues that are 4.
index=N;
sigma=zeros(n);
while (e(index,index)>(4-1e-12))
    sigma=sigma+rand*rvec2hmat(vecs(:,index),n);
    index=index-1;
    if index==0,
        break;
    end
end
% Ensure that that trace(sigma) is zero to preserve noramalization.
sigma=sigma-trace(sigma)*rho;
```

A.1.2 approachEdge

```
%Moves rho along the direction of sigma as far as possible without rho
%or its partial transposes gaining any negative eigenvalues.

e=zeros(n,4);
it=0;

%move until you have at least one negative eigenvalue
while true
    rho=rho+stepsize*sigma;
    e = fulleig(rho);
    eigrho=sort(e(:));
    if eigrho(1)<-1e-9
        break;
    end
end

stepsize=-1*stepsize;

%Move backwards until you are exactly at the edge. A step is allowed
%if it raises the value of the lowest eigenvalue, but does not
%increase it to more than half the value of the previous lowest
```

```
%eigenvalue. If a step is not allowed the stepsize is halfed.
%The loop continues until no change can be detected in the eigenvalues
%or the stepsize becomes lesss than 1e-20.
while true
    it=it+1;
    tau=rho+stepsize*sigma;
    eigtau = fulleig(tau);
    eigtau=sort(eigtau(:));

    if eigtau(1)<0.5*eigrho(1)&&(eigrho(1)<eigtau(1))
        rho=tau;
        e0=e;
        e = fulleig(rho);
        eigrho=sort(e(:));
        if max(max(abs(e0-e)))==0
            break;
        end
    else
        stepsize=0.5*stepsize;
        if abs(stepsize)<1e-20
            break;
        end
    end

    if it==10000,
        disp('MAX ITER REACHED')
        break;
    end
end
```

A.1.3 hmat2rvec

```
function OUT = hmat2rvec(A,n)
% A is a Hermitean (n x n) matrix
% to be represented as a real vector of dimension n^2
%
v = zeros(n,1);
for ii=1:n
  v(ii) = real(A(ii,ii));
end
kk = n;
a = sqrt(2.0);
for jj=2:n
  for ii=1:jj-1
    kk = kk+1;
    v(kk) = a*real(A(ii,jj));
    kk = kk+1;
    v(kk) = a*imag(A(ii,jj));
  end
end
%
```

```
OUT = v;
```

A.1.4 rvec2hmat

```
function OUT = rvec2hmat(v,n)
% v is a real vector of dimension n^2
% to be made into a Hermitean (n x n) matrix A
%
A = ones(n,n)+1i*ones(n,n);
for ii=1:n
  A(ii,ii) = v(ii);
end
kk = n+1;
a = sqrt(0.5);
for jj=2:n
  for ii=1:jj-1
    A(ii,jj) = a*(v(kk)+1i*v(kk+1));
    A(jj,ii) = conj(A(ii,jj));
    kk = kk+2;
  end
end
%
OUT = A;
```

A.2 rankSearch

A.2.1 rankSearchMain.m

```
% Searches for a PPT density matrix with a given set of ranks r and rP
% for the density matrix and its partially transposed rhoP. Starts a
% distance y from the maximally mixed state approximately.

%example: [rho,ra,raP] = rankSearchMain(3,3,7,6,0.01);

function [rho,ra,raP1,raP2,raP3] =...
    rankSearchMain(nA,nB,nC,r,rP1,rP2,rP3,y)

format long
warning off

t = cputime;
% Dimensions n, nA and nB for the Hilbert spaces H = H_A tensor H_B, H_A
% and H_B respectively.
n = nA*nB*nC;


% Bound for zero.
bound = 1e-15;

% Number of eigenvalues of rho and rhoP which should be equal to zero in
% the final density matrix.
```

```
zeroEVs = n - r + 1;
zeroEVP1s = n - rP1 + 1;
zeroEVP2s = n - rP2 + 1;
zeroEVP3s = n - rP3 + 1;

% eye(n)/n is the n-by-n identity matrix divided by n, so this is the
% maximally mixed state in H.
rho0 = eye(n)/n;

% M is a cell vector where each element is a basis matrix for
% traceless, hermitian n-by-n matrices.
M = genGellMann(n);

% dim is the number of dimensions of the set of normalized hermitian
% matrices.
dim = n*n - 1;

% sigma will be the traceless part of rho.
sigma = zeros(n);

x0 = randn(dim,1);

% making sure that we start inside the set of density matrices. This step
% is not really neccessary, but experience shows that it is easier to find
% density matrices with very asymmetric ranks if we start close to the
% maximally mixed state.

x0 = y*x0;
sigma = cSV(M,n,x0);
rho = rho0 + sigma;

% Do the search
[fMin, xMin] = rankSearch(rho0, x0, dim, M, zeroEVs, ...
    zeroEVP1s, zeroEVP2s, zeroEVP3s, nA, nB, nC, bound);

% Recreate rho from xmin
sigma = cSV(M,n,xMin);
rho = rho0 + sigma;

% display the eigenvalues and ranks of rho and rhoP
[eig(rho) eig(deltransponer(rho,nA,nB*nC))...
    eig(deltransponer(deltransponer(rho,nA,nB*nC),nA*nB,nC))...
    eig(deltransponer(rho,nA*nB,nC))]
ra = rank(rho,1e-12);
raP1 = rank(deltransponer(rho,nA,nB*nC),1e-12);
raP2 = rank(deltransponer(deltransponer(rho,nA,nB*nC),nA*nB,nC),1e-12);
raP3 = rank(deltransponer(rho,nA*nB,nC),1e-12);
[ra raP1 raP2 raP3]

cputime - t
```

```
end
```

A.2.2 rankSearch.m

```
function [f, xc] = rankSearch(rho0, x0, dim, M, zeroEVs, zeroEVP1s,...
    zeroEVP2s, zeroEVP3s, nA, nB, nC, fbound)

% output to screen
disp('Starting search:')

% calculating the dimension of the full Hilbertspace
n = nA*nB*nC;

% xc contains the current x-values
xc = x0;

% B will contain the change in the eigenvalues we want to minimize
% as we vary x-values. zeroEVs+zeroEVP1s+zeroEVP2s+zeroEVP3s is the #
% of eigenvalues we want to be equal to zero when we're done
B = zeros(zeroEVs+zeroEVP1s+zeroEVP2s+zeroEVP3s,dim);

sigma = cSV(M,n,x0);
rho = rho0 + sigma;

rhoP1 = deltransponer(rho,nA,nB*nC);
rhoP2 = deltransponer(deltransponer(rho,nA,nB*nC),nA*nB,nC);
rhoP3 = deltransponer(rho,nA*nB,nC);

E = eig(rho);
EP1 = eig(rhoP1);
EP2 = eig(rhoP2);
EP3 = eig(rhoP3);

[E,I] = sort(E);
[EP1,IP1] = sort(EP1);
[EP2,IP2] = sort(EP2);
[EP3,IP3] = sort(EP3);

lambda = [E(1:zeroEVs); EP1(1:zeroEVP1s);...
    EP2(1:zeroEVP2s); EP3(1:zeroEVP3s)];

f = sqrt(lambda'*lambda);

eigOK = sign(E(1));

if f < fbound && eigOK == 1
    return
else
    MP1 = cell(length(M),1);
    MP2 = cell(length(M),1);
    MP3 = cell(length(M),1);
```

```
for it=1:length(M)
    MP1{it} = deltransponer(M{it},nA,nB*nC);
    MP2{it} = deltransponer(deltransponer(M{it},nA,nB*nC),nA*nB,nC);
    MP3{it} = deltransponer(M{it},nA*nB,nC);
end

iter = 0;
while f > fbound
    iter = iter + 1;
% calculating the partially transposed of the current best final state
    rhoP1 = deltransponer(rho,nA,nB*nC);
    rhoP2 = deltransponer(deltransponer(rho,nA,nB*nC),nA*nB,nC);
    rhoP3 = deltransponer(rho,nA*nB,nC);
 % vecs (vecsP) is a matrix with the eigenvectors of rho (rhoP) as
 % column vectors D (DP) is a diagonal matrix with the eigenvalues
 % of rho (rhoP) on the diagonal
    [vecs,D] = eig(rho);
    vecs = vecs(:,I');

    [vecsP1,DP1] = eig(rhoP1);
    vecsP1 = vecsP1(:,IP1');

    [vecsP2,DP2] = eig(rhoP2);
    vecsP2 = vecsP2(:,IP2');

    [vecsP3,DP3] = eig(rhoP3);
    vecsP3 = vecsP3(:,IP3');

 % V is a matrix with the eigenvectors of rho and rhoP corresponding
 % to the eigenvalues which we are trying to minimize to zero as
 % column vectors
    V = [vecs(:,1:zeroEVs) vecsP1(:,1:zeroEVP1s)...
        vecsP2(:,1:zeroEVP2s) vecsP3(:,1:zeroEVP3s)];

    for it=1:(zeroEVs)
        B(it,:) = transpose(cellfun(@(A) V(:,it)'*A*V(:,it),M));
    end
    for it=(zeroEVs+1):(zeroEVs+zeroEVP1s)
        B(it,:) = transpose(cellfun(@(A) V(:,it)'*A*V(:,it),MP1));
    end
    for it=(zeroEVs+zeroEVP1s+1):(zeroEVs+zeroEVP1s+zeroEVP2s)
        B(it,:) = transpose(cellfun(@(A) V(:,it)'*A*V(:,it),MP2));
    end
    for it=(zeroEVs+zeroEVP1s+zeroEVP2s+1):(zeroEVs+zeroEVP1s+...
            zeroEVP2s+zeroEVP3s)
        B(it,:) = transpose(cellfun(@(A) V(:,it)'*A*V(:,it),MP3));
    end

    % calculating the left and right hand sides of A*dx = b
    B = real(B);
    A = B'*B;
```

```
        b = -B'*lambda;



        % solving A*dx = b for dx with the conjugate gradient method
        [dx, flagg] = cgs(A,b);

        % updating our current best estimate for x
        xc = xc + dx;

        sigma = cSV(M,n,xc);

        rho = rho0 + sigma;
        E = eig(rho);
        [E,I] = sort(E);

        rhoP1 = deltransponer(rho,nA,nB*nC);
        EP1 = eig(rhoP1);
        [EP1,IP1] = sort(EP1);

        rhoP2 = deltransponer(deltransponer(rho,nA,nB*nC),nA*nB,nC);
        EP2 = circshift(EP2,1);
        [EP2,IP2] = sort(EP2);

        rhoP3 = deltransponer(rho,nA*nB,nC);
        EP3(n+1) = 0;
        EP3 = circshift(EP3,1);
        [EP3,IP3] = sort(EP3);

        lambda = [E(1:zeroEVs); EP1(1:zeroEVP1s); EP2(1:zeroEVP2s);...
            EP3(1:zeroEVP3s)];

        f = sqrt(lambda'*lambda);

        % output to screen
        if mod(iter,500) == 0
            disp(['Iter: ' num2str(iter) '    ' ...
           'f = ' num2str(f,'%1.15g')])
        end
    end
end
```

A.2.3 cSV

```
function sigma = cSV(M, n, x)
% cSV returns sigma which is a traceless hermitian n-by-n matrix whose
% coefficients are given by the vector x.
dim = length(M);
sigma = zeros(n);
for it = 1:dim
    sigma = sigma + x(it)*M{it};
```

```
end
```

A.2.4 genGellMann

```
function D = genGellMann(d)
%Creates a complete basis for traceless, hermitian n-by-n matrices.

D = cell(d*d-1,1);

w = 0;
for it=1:d
    for jt=it+1:d
        w = w+1;
        E = zeros(d);
        E(it,jt) = 1;
        D{w} = E + E';
        D{w} = D{w}/sqrt(2);
        w = w+1;
        D{w} = -1i*(E'-E);
        D{w} = D{w}/sqrt(2);
        clear E
    end
end

for it=2:(d-1)
    w = w+1;
    A = sqrt(2/(it*(it-1)))*blkdiag(eye(it-1),1-it);
    D{w} = blkdiag(A,0);
    clear A
    for jt=1:(d-it-1)
        D{w} = blkdiag(D{w},0);
    end
    D{w} = D{w}/sqrt(2);
end

w = w + 1;

D{w} = sqrt(2/(d*(d-1)))*blkdiag(eye(d-1),1-d);
D{w} = D{w}/sqrt(2);
```

A.3 Gathering Symmetric States

A.3.1 gatherSymmetric

```
opts=optimset('Algorithm','sqp',...
              'MaxFunEvals',200000,...
              'MaxIter',500000,...
              'TolFun',1.0e-500,...
              'TolCon',1.0e-500,...
              'TolX',1.0e-500);
```

```
ind=1;
states=cell(1);
for ii=1:200
    x0=randn(27,1);
    [Xf fval exitflag ]=fminsearch(@(x) findSymFun(x),x0,opts);
    rho=createSymmetric(Xf);
    disp(fval)
    fulleig(rho)
    if fval<1e-14
        states{ind}=rho;
        ind=ind+1;
    end
end
```

A.3.2 findSymFun

```
function res = findSymFun(x)
rho=createSymmetric(x);
e=eig(rho);
res=(e(1)^2+e(2)^2+e(3)^2+e(4)^2);
```

A.3.3 createSymmetric

```
function rho =createSymmetric(x)
%Converts the real 27 dimensional vector x to a herimitian
%8 by 8 matrix rho that is symmetric under all partial transposes


a=diag([x(1) x(2)]);
a(2,1)=x(3);
a(1,2)=x(3);

b=diag([x(4) x(5)]);
b(2,1)=x(6);
b(1,2)=x(6);

c=diag([x(7) x(8)]);
c(2,1)=x(9);
c(1,2)=x(9);

A=[a, b; b, c];

a=diag([x(10) x(11)]);
a(2,1)=x(12);
a(1,2)=x(12);

b=diag([x(13) x(14)]);
b(2,1)=x(15);
b(1,2)=x(15);

c=diag([x(16) x(17)]);
```

```
c(2,1)=x(18);
c(1,2)=x(18);

B=[a, b; b, c];

a=diag([x(19) x(20)]);
a(2,1)=x(21);
a(1,2)=x(21);

b=diag([x(22) x(23)]);
b(2,1)=x(24);
b(1,2)=x(24);

c=diag([x(25) x(26)]);
c(2,1)=x(27);
c(1,2)=x(27);

C=[a, b; b, c];

rho=[A,B;B,C];
rho=rho/trace(rho);
```

Appendix B

Optimization methods

B.1 trippleEigMin

```
n1=2;
n2=2;
n3=2;
n=n1*n2*n3;
ftol=1e-16;
etol=1e-12;
id1=eye(n1);
id2=eye(n2);
id3=eye(n3);

%Select A
% load('extremal4444states')
% A=states{6};

%Choose randnom starting vectors
u=randn(n1,1)+1i*randn(n1,1);
v=randn(n2,1)+1i*randn(n2,1);
w=randn(n3,1)+1i*randn(n3,1);
u=u/norm(u);
v=v/norm(v);
w=w/norm(w);

%Calculate function value
psi=kron(u,kron(v,w));
fvalue=real(psi'*A*psi);
fvalue0=10;

evs=zeros(1,6);
iter=0;
while (fvalue>ftol)&&(iter<5000)&&(abs(fvalue0-fvalue)>0)

    % Solve first eigenvalueproblem and replace the first vector with the
    % eigenvector corresponding to the smallest eigenvalue
```

```
temp=kron(id1,kron(v,w));
V1=temp'*A*temp;
V1=0.5*(V1+V1');
[vec ev]=eig(V1);
mind=1;
mev=ev(1);
for ii=2:n1
    if ev(ii,ii)<mev
        mev=ev(ii,ii);
        mind=ii;
    end
end
evs(1:2)=diag(ev)';
u=vec(:,mind);

%Repeat for second and third eigenvalue problems
temp=kron(u,kron(id2,w));
V2=temp'*A*temp;
V2=0.5*(V2+V2');
[vec ev]=eig(V2);
mind=1;
mev=ev(1);
for ii=2:n1
    if ev(ii,ii)<mev
        mev=ev(ii,ii);
        mind=ii;
    end
end
v=vec(:,mind);
evs(3:4)=diag(ev)';

temp=kron(u,kron(v,id3));
A3=temp'*A*temp;
A3=0.5*(A3+A3');
[vec ev]=eig(A3);
mind=1;
mev=ev(1);
for ii=2:n1
    if ev(ii,ii)<mev
        mev=ev(ii,ii);
        mind=ii;
    end
end
w=vec(:,mind);
evs(5:6)=diag(ev)';

%Recalculate function value
psi=kron(u,kron(v,w));
fvalue0=fvalue;
fvalue=real(psi'*A*psi);
```

```
    iter=iter+1;
end
```

B.2 Conjugate gradient method

```
n1=2;
n2=2;
n3=2;
id1=eye(n1);
id2=eye(n2);
id3=eye(n3);
MaxIter = 1e5;
TolGrad = 1e-14;
TolGradC = 1e-14;

% Choose A
load('extremal4444states')
A=states{nstates};

exitflag = 0;
phi=randn(n1,1)+1i*randn(n1,1);
chi=randn(n2,1)+1i*randn(n2,1);
xi=randn(n3,1)+1i*randn(n3,1);
phi=phi/norm(phi);
chi=chi/norm(chi);
xi=xi/norm(xi);

grad1=((phi'*phi)*(chi'*chi)*(xi'*xi)*(kron(id1,kron(chi,xi))'*A*...
    kron(phi,kron(chi,xi)))-(chi'*chi)*(xi'*xi)*...
    (kron(phi,kron(chi,xi))'*A*kron(phi,kron(chi,xi)))*phi)/...
    (norm(phi)*norm(chi)*norm(xi))^4;
grad2=((phi'*phi)*(chi'*chi)*(xi'*xi)*(kron(phi,kron(id2,xi))'*A*...
    kron(phi,kron(chi,xi)))-(phi'*phi)*(xi'*xi)*...
    (kron(phi,kron(chi,xi))'*A*kron(phi,kron(chi,xi)))*chi)/...
    (norm(phi)*norm(chi)*norm(xi))^4;
grad3=((phi'*phi)*(chi'*chi)*(xi'*xi)*(kron(phi,kron(chi,id3))'*A*...
    kron(phi,kron(chi,xi)))-(phi'*phi)*(chi'*chi)*...
    (kron(phi,kron(chi,xi))'*A*kron(phi,kron(chi,xi)))*xi)/...
    (norm(phi)*norm(chi)*norm(xi))^4;
grad=[grad1;grad2;grad3];
ngrad1=grad1;
ngrad2=grad2;
ngrad3=grad3;
ngrad=[ngrad1;ngrad2;ngrad3];

u=-1*grad1;
v=-1*grad2;
w=-1*grad3;

fvalue=f(A,phi,chi,xi);
iter=0;
```

```
while (norm(ngrad)>TolGrad)
    iter=iter+1;
    fvalue0=fvalue;
    [t fvalue]=TPlineMinimize(A,phi,chi,xi,u,v,w);

    phi=phi+t*u;
    chi=chi+t*v;
    xi=xi+t*w;

    nphi=phi/norm(phi);
    nchi=chi/norm(chi);
    nxi=xi/norm(xi);

    grad0=grad;
    ngrad1=(kron(id1,kron(nchi,nxi))'*A*kron(nphi,kron(nchi,nxi))...
        -fvalue*nphi);
    ngrad2=(kron(nphi,kron(id2,nxi))'*A*kron(nphi,kron(nchi,nxi))...
        -fvalue*nchi);
    ngrad3=(kron(nphi,kron(nchi,id3))'*A*kron(nphi,kron(nchi,nxi))...
        -fvalue*nxi);
    ngrad=[ngrad1;ngrad2;ngrad3];

    grad1=((phi'*phi)*(chi'*chi)*(xi'*xi)*(kron(id1,kron(chi,xi))'*A*...
        kron(phi,kron(chi,xi)))-(chi'*chi)*(xi'*xi)*...
        (kron(phi,kron(chi,xi))'*A*kron(phi,kron(chi,xi)))*phi)/...
        (norm(phi)*norm(chi)*norm(xi))^4;
    grad2=((phi'*phi)*(chi'*chi)*(xi'*xi)*(kron(phi,kron(id2,xi))'*A*...
        kron(phi,kron(chi,xi)))-(phi'*phi)*(xi'*xi)*...
        (kron(phi,kron(chi,xi))'*A*kron(phi,kron(chi,xi)))*chi)/...
        (norm(phi)*norm(chi)*norm(xi))^4;
    grad3=((phi'*phi)*(chi'*chi)*(xi'*xi)*(kron(phi,kron(chi,id3))'*A*...
        kron(phi,kron(chi,xi)))-(phi'*phi)*(chi'*chi)*...
        (kron(phi,kron(chi,xi))'*A*kron(phi,kron(chi,xi)))*xi)/...
        (norm(phi)*norm(chi)*norm(xi))^4;
    grad=[grad1;grad2;grad3];
    gradchange=abs(norm(grad)-norm(grad0));

    beta=max(real((grad'*(grad-grad0))/(grad0'*grad0)),0);%Polak-Ribiere

    u=-1*grad1+beta*u;
    v=-1*grad2+beta*v;
    w=-1*grad3+beta*w;

    if iter>MaxIter,
        disp('MAX ITER')
        exitflag=1;
        break;
    end
    if gradchange<TolGradC,
        exitflag=2;
```

```
        break;
    end

    phi=phi/norm(phi);
    chi=chi/norm(chi);
    xi=xi/norm(xi);
end

% Standardform phi etc
normphi=norm(phi);
normchi=norm(chi);
normxi=norm(xi);
phi=phi/normphi;
chi=chi/normchi;
xi=xi/normxi;
[maxvalue index] =max(abs(phi));
phi=conj(phi(index))*phi/maxvalue;
[maxvalue index] =max(abs(chi));
chi=conj(chi(index))*chi/maxvalue;
[maxvalue index] =max(abs(xi));
xi=conj(xi(index))*xi/maxvalue;
```

B.2.1 Line minimize

```
function [t fval] = TPlineMinimize(W,phi,chi,xi,u,v,w)

A0 = real(kron(phi,kron(chi,xi))'*W*kron(phi,kron(chi,xi)));
A1 = real(kron(u,kron(chi,xi))'*W*kron(phi,kron(chi,xi))...
   + kron(phi,kron(v,xi))'*W*kron(phi,kron(chi,xi))...
   + kron(phi,kron(chi,w))'*W*kron(phi,kron(chi,xi))...
   + kron(phi,kron(chi,xi))'*W*kron(u,kron(chi,xi))...
   + kron(phi,kron(chi,xi))'*W*kron(phi,kron(v,xi))...
   + kron(phi,kron(chi,xi))'*W*kron(phi,kron(chi,w)));
A2 = real(kron(u,kron(v,xi))'*W*kron(phi,kron(chi,xi))...
   + kron(u,kron(chi,w))'*W*kron(phi,kron(chi,xi))...
   + kron(u,kron(chi,xi))'*W*kron(u,kron(chi,xi))...
   + kron(u,kron(chi,xi))'*W*kron(phi,kron(v,xi))...
   + kron(u,kron(chi,xi))'*W*kron(phi,kron(chi,w))...
   + kron(phi,kron(v,w))'*W*kron(phi,kron(chi,xi))...
   + kron(phi,kron(v,xi))'*W*kron(u,kron(chi,xi))...
   + kron(phi,kron(v,xi))'*W*kron(phi,kron(v,xi))...
   + kron(phi,kron(v,xi))'*W*kron(phi,kron(chi,w))...
   + kron(phi,kron(chi,w))'*W*kron(u,kron(chi,xi))...
   + kron(phi,kron(chi,w))'*W*kron(phi,kron(v,xi))...
   + kron(phi,kron(chi,w))'*W*kron(phi,kron(chi,w))...
   + kron(phi,kron(chi,xi))'*W*kron(u,kron(v,xi))...
   + kron(phi,kron(chi,xi))'*W*kron(u,kron(chi,w))...
   + kron(phi,kron(chi,xi))'*W*kron(phi,kron(v,w)));
A3 = real(kron(u,kron(v,w))'*W*kron(phi,kron(chi,xi))...
   + kron(u,kron(v,xi))'*W*kron(u,kron(chi,xi))...
   + kron(u,kron(v,xi))'*W*kron(phi,kron(v,xi))...
```

```
    + kron(u,kron(v,xi))'*W*kron(phi,kron(chi,w))...
    + kron(u,kron(chi,w))'*W*kron(u,kron(chi,xi))...
    + kron(u,kron(chi,w))'*W*kron(phi,kron(v,xi))...
    + kron(u,kron(chi,w))'*W*kron(phi,kron(chi,w))...
    + kron(u,kron(chi,xi))'*W*kron(u,kron(v,xi))...
    + kron(u,kron(chi,xi))'*W*kron(u,kron(chi,w))...
    + kron(u,kron(chi,xi))'*W*kron(phi,kron(v,w))...
    + kron(phi,kron(v,w))'*W*kron(u,kron(chi,xi))...
    + kron(phi,kron(v,w))'*W*kron(phi,kron(v,xi))...
    + kron(phi,kron(v,w))'*W*kron(phi,kron(chi,w))...
    + kron(phi,kron(v,xi))'*W*kron(u,kron(v,xi))...
    + kron(phi,kron(v,xi))'*W*kron(u,kron(chi,w))...
    + kron(phi,kron(v,xi))'*W*kron(phi,kron(v,w))...
    + kron(phi,kron(chi,w))'*W*kron(u,kron(v,xi))...
    + kron(phi,kron(chi,w))'*W*kron(u,kron(chi,w))...
    + kron(phi,kron(chi,w))'*W*kron(phi,kron(u,w))...
    + kron(phi,kron(chi,xi))'*W*kron(u,kron(v,w)));
A4 = real(kron(phi,kron(chi,w))'*W*kron(u,kron(v,w))...
    + kron(phi,kron(v,xi))'*W*kron(u,kron(v,w))...
    + kron(phi,kron(v,w))'*W*kron(phi,kron(v,w))...
    + kron(phi,kron(v,w))'*W*kron(u,kron(chi,w))...
    + kron(phi,kron(v,w))'*W*kron(u,kron(v,xi))...
    + kron(u,kron(chi,xi))'*W*kron(u,kron(v,w))...
    + kron(u,kron(chi,w))'*W*kron(phi,kron(v,w))...
    + kron(u,kron(chi,w))'*W*kron(u,kron(chi,w))...
    + kron(u,kron(chi,w))'*W*kron(u,kron(v,xi))...
    + kron(u,kron(v,xi))'*W*kron(phi,kron(v,w))...
    + kron(u,kron(v,xi))'*W*kron(u,kron(chi,w))...
    + kron(u,kron(v,xi))'*W*kron(u,kron(v,xi))...
    + kron(u,kron(v,w))'*W*kron(phi,kron(chi,w))...
    + kron(u,kron(v,w))'*W*kron(phi,kron(v,xi))...
    + kron(u,kron(v,w))'*W*kron(u,kron(chi,xi)));
A5 = real(kron(phi,kron(v,w))'*W*kron(u,kron(v,w))...
    + kron(u,kron(chi,w))'*W*kron(u,kron(v,w))...
    + kron(u,kron(v,xi))'*W*kron(u,kron(v,w))...
    + kron(u,kron(v,w))'*W*kron(phi,kron(v,w))...
    + kron(u,kron(v,w))'*W*kron(u,kron(chi,w))...
    + kron(u,kron(v,w))'*W*kron(u,kron(v,xi)));
A6 = real(kron(u,kron(v,w))'*W*kron(u,kron(v,w)));

B0 = (phi'*phi)*(chi'*chi)*(xi'*xi);
B1 = (phi'*u+u'*phi)*(chi'*chi)*(xi'*xi)...
    + (phi'*phi)*(chi'*v+v'*chi)*(xi'*xi)...
    + (phi'*phi)*(chi'*chi)*(xi'*w+w'*xi);
B2 = (u'*u)*(chi'*chi)*(xi'*xi)...
    + (phi'*phi)*(v'*v)*(xi'*xi)...
    + (phi'*phi)*(chi'*chi)*(w'*w)...
    + (phi'*phi)*(chi'*v+v'*chi)*(xi'*w+w'*xi)...
    + (phi'*u+u'*phi)*(chi'*chi)*(xi'*w+w'*xi)...
    + (phi'*u+u'*phi)*(chi'*v+v'*chi)*(xi'*xi);
B3 = (phi'*u+u'*phi)*(chi'*v+v'*chi)*(xi'*w+w'*xi)...
```

```
    + (u'*u)*(chi'*v+v'*chi)*(xi'*xi)...
    + (u'*u)*(chi'*chi)*(xi'*w+w'*xi)...
    + (phi'*phi)*(v'*v)*(xi'*w+w'*xi)...
    + (phi'*u+u'*phi)*(v'*v)*(xi'*xi)...
    + (phi'*phi)*(chi'*v+v'*chi)*(w'*w)...
    + (phi'*u+u'*phi)*(chi'*chi)*(w'*w);
B4 = (u'*u)*(chi'*v+v'*chi)*(xi'*w+w'*xi)...
    + (phi'*u+u'*phi)*(v'*v)*(xi'*w+w'*xi)...
    + (phi'*u+u'*phi)*(chi'*v+v'*chi)*(w'*w)...
    + (phi'*phi)*(v'*v)*(w'*w)...
    + (u'*u)*(chi'*chi)*(w'*w)...
    + (u'*u)*(v'*v)*(xi'*xi);
B5 = (phi'*u+u'*phi)*(v'*v)*(w'*w)...
    + (u'*u)*(chi'*v+v'*chi)*(w'*w)...
    + (u'*u)*(v'*v)*(xi'*w+w'*xi);
B6 = (u'*u)*(v'*v)*(w'*w);


C = [A6*B5-A5*B6,...
     2*A6*B4-2*A4*B6,...
     3*A6*B3+A5*B4-A4*B5-3*A3*B6,...
     4*A6*B2+2*A5*B3-2*A3*B5-4*A2*B6,...
     5*A6*B1+3*A5*B2+A4*B3-A3*B4-3*A2*B5-5*A1*B6,...
     6*A6*B0+4*A5*B1+2*A4*B2-2*A2*B4-4*A1*B5-6*A0*B6,...
     5*A5*B0+3*A4*B1+A3*B2-A2*B3-3*A1*B4-5*A0*B5,...
     4*A4*B0+2*A3*B1-2*A1*B3-4*A0*B4,...
     3*A3*B0+A2*B1-A1*B2-3*A0*B3,...
     2*A2*B0-2*A0*B2,...
     A1*B0-A0*B1];

r = roots(C);
r=r(find(real(r)>0));
r=r(find(imag(r)==0));

% Choose the smallest real solution
t=min(r);
if isempty(t)
    t=2;
end

fval=f(W,phi+t*u,chi+t*v,xi+t*w);
```

B.3 SQP

```
% Supply the dimensions n1, n2, and n3 the observable A

nr1=2*n1;
nr2=2*n2;
nr3=2*n3;

id1=eye(n1);
```

```
id2=eye(n2);
id3=eye(n3);

J=[id1  1i*id1];
K=[id2  1i*id2];
L=[id3  1i*id3];
JKL=kron(J,kron(K,L));
W=real(JKL'*A*JKL);
W=0.5*(W+W');
W=0.25*(W+deltransponer(W,nr1*nr2,nr3)+deltransponer(W,nr1,nr2*nr3)'...
    +deltransponer(deltransponer(W,nr1,nr2*nr3),nr1*nr2,nr3));

% A=0.25*(A+A'+pt(A+A',nr1,nr2));

x0=randn(nr1,1);
x0=x0/norm(x0);
y0=randn(nr2,1);
y0=y0/norm(y0);
z0=randn(nr3,1);
z0=z0/norm(z0);
X0=[x0;y0;z0];
clear x0 y0 z0

opts=optimset('Algorithm','sqp',...
              'DerivativeCheck','off',...
              'Diagnostics','off',...
              'Display','off',...
              'GradConstr','on',...
              'GradObj','on',...
              'MaxFunEvals',200*(nr1+nr2),...
              'MaxIter',500,...
              'TolFun',1.0e-16,...
              'TolCon',1.0e-16,...
              'TolX',1.0e-16,...
              'ObjectiveLimit',-1.0e6,...
              'ScaleProblem','none'...
              );

[Xf fval exitflag output lambda grad]=...
    fmincon(@(X) SQPfun(nr1,nr2,nr3,W,X),X0,[],[],[],[],[],[],...
            @(X) SQPconstr(nr1,nr2,nr3,X),opts);

phi=J*Xf(1:nr1);
chi=K*Xf(nr1+1:nr1+nr2);
xi=L*Xf(nr1+nr2+1:nr1+nr2+nr3);
[maxvalue index]=max(abs(phi));
phi=conj(phi(index))*phi/maxvalue;
[maxvalue index]=max(abs(chi));
chi=conj(chi(index))*chi/maxvalue;
[maxvalue index]=max(abs(xi));
xi=conj(xi(index))*xi/maxvalue;
```

B.3.1 SQPfun

```
function [funval grad]=SQPfun(nr1,nr2,nr3,A,X)
%
% [funval grad]=f01spqeval(nr1,nr2,B,z)
%

xyz=kron(X(1:nr1),kron(X(nr1+1:nr1+nr2),X(nr1+nr2+1:nr1+nr2+nr3)));
funval=xyz'*A*xyz;
grad=2*[kron(eye(nr1),kron(X(nr1+1:nr1+nr2),X(nr1+nr2+1:nr1+nr2+nr3)))'...
    *A*xyz;
        kron(X(1:nr1),kron(eye(nr2),X(nr1+nr2+1:nr1+nr2+nr3)))'*A*xyz;
    kron(X(1:nr1),kron(X(nr1+1:nr1+nr2),eye(nr3)))'*A*xyz];
```

B.3.2 SQPconstr

```
function [c ceq gradc gradceq] = SQPconstr(nr1,nr2,nr3,X)
%
% [c ceq gradc gradceq] = SQPconstr(nr1,nr2,z)
%

c=zeros(nr1+nr2+nr3,0);
ceq=[X(1:nr1)'*X(1:nr1)-1;
     X(nr1+1:nr1+nr2)'*X(nr1+1:nr1+nr2)-1;
     X(nr1+nr2+1:nr1+nr2+nr3)'*X(nr1+nr2+1:nr1+nr2+nr3)-1];
gradc=c;
gradceq=2*[[X(1:nr1); zeros(nr2,1); zeros(nr3,1)], [zeros(nr1,1);...
    X(nr1+1:nr1+nr2); zeros(nr3,1)], [zeros(nr1+nr2,1);...
    X(nr1+nr2+1:nr1+nr2+nr3)]];
```

Bibliography

[1] J. Leinaas, J. Myrheim, and P. Ø. Sollid, "Numerical studies of entangled positive-partial-transpose states in composite quantum systems," *Physical Review A*, vol. 81, pp. 1–12, June 2010.

[2] J. Leinaas, J. Myrheim, and E. Ovrum, "Extreme points of the set of density matrices with positive partial transpose," *Physical Review A*, vol. 76, pp. 1–4, Sept. 2007.

[3] R. Werner, "Quantum states with Einstein-Podolsky-Rosen correlations admitting a hidden-variable model," *Physical Review A*, vol. 40, pp. 4277–4281, Oct. 1989.

[4] A. Einstein, B. Podolsky, and N. Rosen, "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?," *Physical Review*, vol. 47, pp. 2–5, 1935.

[5] J. S. Bell, "On the Einstein Podolsky Rosen pradox," *Physics*, 1964.

[6] N. Mermin, "What's wrong with these elements of reality?," *Physics Today*, vol. 43, no. 6, p. 9, 1990.

[7] R. Horodecki, M. Horodecki, and K. Horodecki, "Quantum entanglement," *Reviews of Modern Physics*, vol. 81, pp. 865–942, June 2009.

[8] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[9] A. Peres, "Separability criterion for density matrices," *Physical Review Letters*, vol. 77, pp. 1413–1415, Aug. 1996.

[10] M. Horodecki, P. Horodecki, and R. Horodecki, "Separability of mixed states: necessary and sufficient conditions," *Physics Letters A*, vol. 223, pp. 1–8, Nov. 1996.

[11] L. Chen and Y.-X. Chen, "Rank-three bipartite entangled states are distillable," *Physical Review A*, vol. 78, p. 5, Aug. 2008.

[12] S. Karnas and M. Lewenstein, "Separability and entanglement in 2x2xN composite quantum systems," *Physical Review A*, vol. 64, pp. 1–11, Sept. 2001.

[13] W. Dür, G. Vidal, and J. I. Cirac, "Three qubits can be entangled in two inequivalent ways," *Physical Review A*, vol. 62, pp. 1–12, Nov. 2000.

[14] A. Acín, D. Bruß, M. Lewenstein, and A. Sanpera, "Classification of Mixed Three-Qubit States," *Physical Review Letters*, vol. 87, pp. 2–5, July 2001.

[15] F. Verstraete, J. Dehaene, B. De Moor, and H. Verschelde, "Four qubits can be entangled in nine different ways," *Physical Review A*, vol. 65, pp. 1–5, Apr. 2002.

[16] Ø. Garberg, "Numerical optimization of expectation values in product states," 2012.

[17] C. Bennett, D. DiVincenzo, T. Mor, P. Shor, J. Smolin, and B. Terhal, "Unextendible Product Bases and Bound Entanglement," *Physical Review Letters*, vol. 82, pp. 5385–5388, June 1999.

[18] J. Leinaas, J. Myrheim, and P. Ø. Sollid, "Low-rank extremal positive-partial-transpose states and unextendible product bases," *Physical Review A*, vol. 81, pp. 1–6, June 2010.

[19] L. O. Hansen, A. Hauge, J. Myrheim, and P. Ø. Sollid, "Low-rank positive-partial-transpose states and their relation to product vectors," *Physical Review A*, vol. 85, pp. 1–17, Feb. 2012.