Universitat Autònoma de Barcelona
Facultat de Ciències
Departament de Física

BACHELOR'S THESIS

Degree in Physics

EDGE DETECTION ON QUANTUM CHAINS

Marcel Morillas Rozas

1528024

Year

2021-22

Supervisor

Dr. Emili Bagan Capella

Call

September

Declaració d'Autoria del Treball de Final de Grau

Jo, Marcel Morillas Rozas, amb Document Nacional d'Identitat 47751254B, i estudiant del Grau en Física de la Universitat Autònoma de Barcelona, en relació amb la memòria del Treball de Final de Grau presentada pera a la seva defensa i avaluació durant la convocatòria de Setembre del curs 2021-2022, declara que

- El document presentat és original i ha estat realitzat per la seva persona.
- El treball s'ha dut a terme principalment amb l'objectiu d'avaluar l'assignatura de treball de grau en física en la UAB, i no s'ha presentat prèviament per ser qualificat en l'avaluació de cap altre assignatura ni en aquesta ni en cap altra universitat.
- En el cas de continguts de treballs publicat per terceres persones, l'autoria està clarament atribuïda, citant les fonts degudament.
- En els casos en els que el meu treball s'ha realitzat en col·laboració amb altres investigadors i/o estudiants, es declara amb exactitud quines contribucions es deriven del treball de tercers i quines es deriven de la meva contribució.
- A excepció del punts esmentats anteriorment, el treball presentat és de la meva autoria.

Signat:

Marcel Morillas Rozas

Declaració d'Extensió del Treball de Final de Grau

Jo, Marcel Morillas Rozas, amb Document Nacional d'Identitat 47751254B, i estudiant del Grau en Física de la Universitat Autònoma de Barcelona, en relació amb la memòria del Treball de final de Grau presentada pera a la seva defensa i avaluació durant la convocatòria de Setembre del curs 2021-2022, declara que:

– El nombre total de paraules (segons comptatge proposat) incloses en les seccions des de la introducció a les conclusions es de **5829** paraules.

– El nombre total de figures es de **8**.

En total el document comptabilitza:
5829 paraules + **8** figures × 200 paraules/figura = **7429** paraules
Que compleix amb la normativa al ser inferior a 10000.

Signat:

Marcel Morillas Rozas

Abstract

Abrupt changes are very common in nature and being able to identify them is essential for applications in a variety of fields. This work takes this identification problem to the quantum world. A source that prepares spin $\frac{1}{2}$ states is considered to emit particles at an unknown pure state $|\eta\rangle$. At a certain point the source feels a perturbation which makes it start preparing and emitting other unknown pure states that point in an arbitrary direction $|\phi\rangle$. Alternatively, the problem can be considered as a chain of spins with two domains. In one of them the state of the spins is $|\eta\rangle$ and in the other one the state is $|\phi\rangle$, being both states unknown. The goal is to effectively be able to detect where the abrupt change, also refered as the edge of the domain, occurs. The success probabilities of detecting the change point or edge are computed alongside a quantification of the average error made when guessing where the change does happen using the Hamming distance. In order to perform all the calculations, projective measurements on the whole chain of spins are allowed to be performed. It is shown that the edge is best detected when the states $|\eta\rangle$ and $|\phi\rangle$ are orthogonal.

Resum

Els canvis bruscs son comuns a la natura i ser capaç d'identificar-los és essencial en molts camps del coneixement. Aquest treball proposa aquesta tasca d'identificació de canvis sobtats en el món de la mecànica quàntica. Es considera una font que prepara estats d'espí $\frac{1}{2}$. Es considera que els estats preparats són són purs i corresponen a un estat desconegut $|\eta\rangle$, fins que la font pateix una pertorbació i comença a preparar i emetre uns altres estats quàntics purs desconeguts $|\phi\rangle$ orientats en una direcció arbitrària. Alternativament es pot considerar una cadena d'espins amb dos dominis. En un d'ells l'estat dels espins és $|\eta\rangle$ mentre que en el segon l'estat dels espins és $|\phi\rangle$, essent els dos estats desconeguts. L'objectiu d'aquest treball és ser capaç d'identificar de forma efectiva on es produeix aquest canvi brusc donada una cadena d'espins. S'obtenen les probabilitats d'èxit així com una quantificació de l'error comès en endevinar on es produeix aquest canvi utilitzant la distància de Hamming. Per a fer-ho possible, es permet fer mesures projectives sobre la totalitat de la cadena d'espins. Es mostra que el canvi brusc es detecta de forma més exitosa quan els estats $|\eta\rangle$ i $|\phi\rangle$ són ortogonals.

Acknowledgements

I would like to take some lines to acknowledge and thank all the people that have been somehow involved in this Bachelor's Thesis. First and foremost, I would like to give my warmest thanks to my supervisor Dr. Emili Bagan Capella who has made this work possible. Thanks for all the guidance and patience over the last year with me. Thanks for all the long and clarifying meetings either in your office during the university year or via video call in Summer, which have been very clarifying and have made me learn a lot during all this process of the BSc Thesis.

I would also like to express my gratitude to PhD students Walther Gonzalez and Yago Llorens who were always available for me whenever I needed to check results and they even got the patience to explain me some core concepts used in this BSc Thesis even though they were not directly involved in this work. I will be forever grateful for that.

I wish to extend my special thanks to all my friends and colleagues in the Physics degree, specially Diego Contreras, Arnau Diebra, Joaquim Castro, Felipe Díaz, Sergio Iglesias, Nerea Río, Alex Girón, Marc Alonso and Joel Cardero for always being supportive in the process of the BSc Thesis in all the ups and downs I have gone through.

Last but not least I would like to thank all my family for always supporting and listening to my progresses in the BSc Thesis despite their lack of knowledge on the topic. Special thanks to my parents and all my gratitude to my aunt Librada Rozas and my uncle Marcel Pagès for treating me as their own son for the last six years and my cousin Carlos Pagès for somehow being like an older brother to me all this time.

Contents

List of Figures

List of Tables

Glossary

This work will mainly follow the same notation as in Nielsen & Chuang [1]. This glossary is mainly intended for those readers who are not familiar with the notation and formalism used in quantum mechanics. Moreover, some abbreviations used in the work have been included.

$\|\psi\rangle$	Ket. Column Vector in $\mathbb{C}^n$.
$\langle\psi\|$	Bra. Dual vector to $\|\psi\rangle$. $\langle\psi\| = \|\psi\rangle^\dagger = \left(\|\psi\rangle^T\right)^*$.
$\langle\psi\|\phi\rangle$	Braket. Inner product between $\|\psi\rangle$ and $\|\phi\rangle$.
$\|\psi\rangle\langle\phi\|$	Ketbra. Outter product between $\|\psi\rangle$ and $\|\phi\rangle$.
$\|\psi\rangle \otimes \|\phi\rangle$	Tensor (or Kronecker) product between $\|\psi\rangle$ and $\|\phi\rangle$. Abbreviated as $\|\psi\rangle\|\phi\rangle$.
$A^\dagger$	Hermitian adjoint of matrix A. $A^\dagger = \left(A^T\right)^*$.
$\langle\phi\|A\|\psi\rangle$	Inner product between ϕ and $A\|\psi\rangle$ or, equivalently, between $A^\dagger\|\phi\rangle$ and $\|\psi\rangle$.
ρ	Density matrix. It describes the quantum state of a system.
$\mathbb{I}_n$	Identity matrix of dimension $n \times n$.
$\mathcal{H}$	Hilbert Space.
$\mathcal{S}_n$	Symmetric group obtained from the $n!$ permutations of n elements.
$[A, B]$	Commutator of A, B. $[A, B] = AB - BA$

Abbreviations

SW	Schur-Weyl
IRREP	Irreducible Representation
CG	Clebsch-Gordan
POVM	Positive Operator Valued Measure
SRM	Square Root Measurement
SDP	Semidefinite Programming

1 Introduction

Artificial Intelligence (AI) and Machine Learning (ML) algorithms are ruling the World nowadays. ML algorithms are present in almost any process that humans undergo on a daily basis, from self driving cars to recommending new songs on an application based on songs that have been previously listened by the user, going through applications of AI in economics, medicine, or physics amongst others.

Although AI is in the middle of a revolution in which everybody base their decisions on what these algorithms predict, there is another revolution going on, the quantum revolution. Quantum computers are starting to be a reality instead of a dream. When these quantum computers are more evolved, they will need algorithms to work with and, with time, they will be able to run complex quantum machine learning (QML) algorithms. Therefore, it is important to develop these algorithms, at least from a theoretical point of view in order to be able to apply them as soon as quantum computers allow to do so.

Even though there are already some QML algorithms that have successfully been implemented in current quantum computers, not all the existing algorithms can be applied nowadays.

In this work, a source that always emits pure qubit states $|\eta\rangle$ pointing in an unknown direction is going to be considered, as in Fig. 1. All of a sudden, the source suffers a perturbation causing it to emit particles in a different unknown quantum state $|\phi\rangle$ [2] which point in an arbitrary direction given by an angle with respect to the original direction.

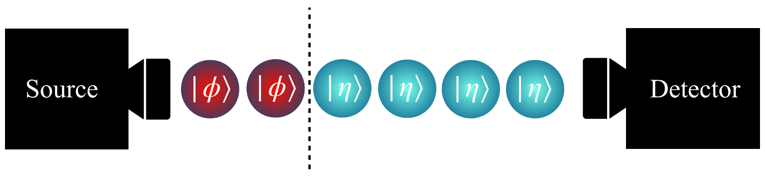


Figure 1: Problem Illustration.

Alternatively, a chain of spin $\frac{1}{2}$ states with two domains might be considered; each of them in an unknown product state $|\eta\rangle^{\otimes n}$ and $|\phi\rangle^{\otimes \bar{n}}$ respectively, where n and $\bar{n} = N - n$ are the number of spin states in each domain. The goal of this work is to correctly predict when the change in the chain, also referred as the change point or edge, has taken place. These type of sudden changes are usual in nature and being able to identify them is essential for some applications across a variety of fields. However in this case the change takes place in the quantum domain and in order to detect it, some measurements on the whole emitted sequence of quantum states are allowed to be performed.

This work presents two different main types of measurements, from which different success probabilities are going to be obtained. Furthermore, a quantification of the average "variance" made on predicting the edge's position is presented, alongside with a probability distribution for guessing where the edge is located, but previously some necessary key elements, specially regarding group theory and quantum mechanics are needed to understand the whole work are presented.

2 Representation Theory of the Symmetric Group

Quantum Mechanics (QM) postulates a theory in which the systems are described using linear algebra elements in Hilbert spaces. When considering a system with N particles, the dimensions of the objects in the Hilbert space of the system $\mathcal{H}^{\otimes N}$ scale exponentially, which makes them difficult to be dealt with. However there is an important tool, the Schur-Weyl duality, which simplifies the analysis of these problems.

2.1 Partitions

Partitions and their graphical representation, Young Tableaus, are a key element in representation theory of groups and in combinatorics. A partition is defined as a sequence

$$\lambda = (\lambda_1, \lambda_2, ..., \lambda_N) \tag{1}$$

of non-negative integer numbers in non-increasing order i.e $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$ [3]. The non-zero elements from λ are called the parts of λ and the number of parts of λ are the length of the partition $l(\lambda)$. A partition λ of the integer N is denoted as $\lambda \vdash N$ where $N = \sum_i \lambda_i$.

2.2 Permutations

A permutation of a set can be understood as the process in which the order of its elements is changed. A permutation of n elements is described as a 2 row list

$$\pi = \begin{pmatrix} i \\ p_i \end{pmatrix} = \begin{pmatrix} 1 & 2 & ... & i & ... & N \\ p_1 & p_2 & ... & p_i & ... & p_N \end{pmatrix} \tag{2}$$

where the first row contains the elements that are going to be permuted and the second row their new positions. In the cases where $i = p_i$, the corresponding column from π can be omitted from the permutation since it is equivalent to act on the i-th element with the identity. In general, the action of two permutations π_1, π_2 does not commute.

Given N elements, the $N!$ permutations of the elements form the so-called Symmetric Group $\mathcal{S}_N$.

2.3 Representation Theory of $\mathcal{S}_N$ and $GL(d, \mathbb{C})$ on the tensor product space $(\mathbb{C}^d)^{\otimes N}$

SW duality is an incredibly powerful tool that connects the irreducible representations (irreps) of the linear group $GL(d, \mathbb{C})$ of $d \times d$ invertible matrices over the complex numbers and the symmetric group $\mathcal{S}_N$. Let $R^{\otimes N}$ and $\mathcal{U}_\pi$ be transformations on the tensor product space $(d, \mathbb{C})^{\otimes N}$, where $R \in GL(d, \mathbb{C})$ and $\mathcal{U}_\pi$ permutes the N spaces of $(d, \mathbb{C})$ according to the permutation $\pi \in \mathcal{S}_N$ [3]. In the case of QM, $GL(d, \mathbb{C})$ can simply be restricted to the special unitary group of d dimensions $SU(d)$.

Suppose a system of N particles and the corresponding Hilbert space spanned $\mathcal{H} = (\mathbb{C}^d)^{\otimes N}$ in which the following basis is defined

$$|i_1\rangle \otimes |i_2\rangle \otimes ... \otimes |i_N\rangle = |i_1, i_2, ..., i_N\rangle, \quad i_j \in \{0, 1, 2, ..., d-1\} \tag{3}$$

The action of $SU(d)$ and $\mathcal{S}_N$ onto this basis is given by

$$R^{\otimes N} |i_1, i_2, ..., i_N\rangle = R|i_1\rangle \otimes R|i_2\rangle \otimes ... \otimes R|i_N\rangle \tag{4}$$

$$\mathcal{U}_\pi |i_1, i_2, ..., i_N\rangle = \left|i_{\pi^{-1}(1)}, i_{\pi^{-1}(2)}, ..., i_{\pi^{-1}(N)}\right\rangle \tag{5}$$

The fact that R and $\mathcal{U}_\pi$ commute, i.e $[R,\mathcal{U}_\pi] = 0$ allows the Hilbert space to be expressed as

$$\mathcal{H} = \bigoplus_{\lambda \vdash N} V_{(\lambda)}(R) \otimes D_{\{\lambda\}}(\mathcal{U}_\pi) \tag{6}$$

by virtue of Schur's Lemma. The indices in brackets (λ) refer to the irreps of $SU(d)$ and the indices in curly braces $\{\lambda\}$ refer to the irreps of $\mathcal{S}_N$. The resulting block diagonal structure provides a decomposition of $\mathcal{H}^{\otimes N}$ into invariant subspaces under the actions of $SU(d)$ and $\mathcal{S}_N$. The basis in which $\mathcal{H}^{\otimes N}$ has the block diagonal form is known as the Schur basis.

2.4 Schur-Weyl Duality for Qubits

In the case where the states of the system are qubits, the group $SU(d)$ reduces to $SU(2)$, meaning that the Schur transformation on the density operator of the state ρ_n will decouple the actions of $SU(2)$ and $\mathcal{S}_N$ over product states. The partitions of N into two elements will give as a result partitions $\lambda = (n, \bar{n}) \vdash N$ and the density operator of the system ρ_n in the Schur basis is expressed as

$$\rho_n = c_n \bigoplus_{\lambda} \mathbb{I}_{(\lambda)} \otimes \Omega^n_{\{\lambda\}} \tag{7}$$

when integrated over all the possible directions in each domain of the spin chain. λ are the irreps of the angular momentum J and c_n is a normalization factor given by

$$c_n = \frac{1}{(n+1)(\bar{n}+1)} = \frac{1}{(n+1)(N-n+1)} \tag{8}$$

The $\Omega^n_{\{\lambda\}}$ operators from Eq. 7 are rank-1 projectors which carry all the relevant information about the edge's location and are equal to 0 for irreps λ outside the support of ρ_n. Since these projectors have rank 1, it will be enough to obtain their eigenstate in order to obtain all the necessary information. These eigenstates will be denoted as $|\Omega^J_n\rangle$.

3 Addition of Angular Momenta

When dealing with a system of multiple particles which carry angular momentum it is fundamental to know the angular momentum of the whole system. The Hilbert space spanned by two subsystems is

$$\mathcal{H}_{12} = \mathcal{H}_1 \otimes \mathcal{H}_2 \tag{9}$$

where $\mathcal{H}_1$ corresponds to the first subsystem and it is spanned by the basis $\{|j_1, m_1\rangle\}$. Analogously, $\mathcal{H}_2$ is spanned by $\{|j_2, m_2\rangle\}$. Therefore, the system can be described in terms of tensor product states of the decoupled basis as

$$|j_1, m_1\rangle \otimes |j_2, m_2\rangle = |j_1, m_1; j_2, m_2\rangle \tag{10}$$

However, there exists a change of basis which transforms the decoupled basis elements into the coupled angular momentum eigenstates $|J, M\rangle$ and the other way around [4, 5]. This transformation is given by

$$|J, M\rangle = \sum_{m_1, m_2} |j_1, m_1; j_2, m_2\rangle \langle j_1, m_1; j_2, m_2|J, M\rangle = \sum_{m_1, m_2} C^{J,M}_{j_1,m_1;j_2,m_2} |j_1, m_1; j_2, m_2\rangle \tag{11}$$

where the fact that

$$\sum_{m_1, m_2} |j_1, m_1; j_2, m_2\rangle\langle j_1, m_1; j_2, m_2| = \mathbb{I} \tag{12}$$

has been used. $C^{J,M}_{j_1,m_1;j_2,m_2} = \langle j_1, m_1; j_2, m_2|J, M\rangle$ are the Clebsch-Gordan (CG) coefficients [6, 7], which transform the states from the decoupled basis to the coupled basis and viceversa.

CG coefficients are real, thus

$$C^{J,M}_{j_1,m_1;j_2,m_2} = \langle j_1, m_1; j_2, m_2 | J, M\rangle = \langle J, M | j_1, m_1; j_2, m_2\rangle \tag{13}$$

The possible total angular momentum values from coupling two systems with angular momenta j_1 and j_2 respectively are:

$$J = \{j_1 + j_2,\ j_1 + j_2 - 1, \dots, |j_1 - j_2|\} \tag{14}$$

and $M = \{-J, -J+1, \dots, 0, \dots, J-1, J\}$ adding up to a total of $2J+1$ possible values for M.

When dealing with N particles, it is important to note that the addition of angular momenta is carried out in pairs, meaning that for larger systems, the coupled basis is obtained by coupling the first two particles, then coupling the resulting system with the third and so on.

Since this work is carried using qubits, the value of j_2 will always be $1/2$. Therefore, the possible values for the CG coefficients are restricted to four possible options, encapsulated in the following two expressions [6]:

$$\left\langle j_1, \left(M - \frac{1}{2}\right); \frac{1}{2}, \frac{1}{2} \middle| \left(j_1 \pm \frac{1}{2}\right), M \right\rangle = \pm\sqrt{\frac{1}{2}\left(1 \pm \frac{M}{j_1 + 1/2}\right)} \tag{15}$$

$$\left\langle j_1, \left(M + \frac{1}{2}\right); \frac{1}{2}, \frac{-1}{2} \middle| \left(j_1 \pm \frac{1}{2}\right), M \right\rangle = \sqrt{\frac{1}{2}\left(1 \mp \frac{M}{j_1 + 1/2}\right)} \tag{16}$$

Moreover, the possible values of J given a system composed by N qubits range from $J_{max} = N/2$ to $J_{min} = 0$ if N is even or $J_{min} = 1/2$ otherwise.

4 Quantum Measurements

According to the third postulate of quantum mechanics, when an observable $\mathcal{A}$ is measured on a quantum system, the only possible outcomes are the eigenvalues a_i of $\mathcal{A}$ [8]. The state of the system after the measurement is the eigenvector $|a_i\rangle$ of the corresponding eigenvalue a_i. In quantum information, measurements are characterised by a set of operators $\{M_i\}$ on a Hilbert space $\mathcal{H}$ such that

$$\sum_i M_i^\dagger M_i = \mathbb{I} \tag{17}$$

where the i index accounts for the different possible outcomes of the measurement. Therefore, the probability corresponding to obtaining the result i given the input $|\psi\rangle$ is given by

$$p(i|\psi) = \langle\psi| M_i^\dagger M_i |\psi\rangle = \| M_i |\psi\rangle \|^2 \tag{18}$$

4.1 Positive Operator Valued Measurement (POVM)

In this work, aside from performing measurements on the angular momentum of the system of particles, quantum measurements on $\mathcal{S}_N$ are going to be performed. These measurements on the symmetric group are going to be carried out using POVMs, which are a generalization of projective measurements characterised by a set of operators $\{E_i\}$ over a Hilbert space $\mathcal{H}$ such that

1. E_i are positive operators.
2. $\{E_i\}$ resolve the identity i.e. $\sum_i E_i = \mathbb{I}$.

3. The probability of obtaining the outcome i given a state $|\psi\rangle$ is $p(i|\psi) = \langle\psi|E_i|\psi\rangle$.

If knowledge of the post measurement state is not required, POVMs can be used instead of the generalised measurements described above, defining $E_i \equiv M_i^\dagger M_i$. Note that they resolve the identity and lead to the same probability distribution as in Eq. 18.

In this work the two types of quantum measurements considered are the Square-Root Measurement and an optimal POVM obtained using Semidefinite Programming

4.2 Square Root Measurement (SRM)

In the task of distinguishing between an arbitrary set of pure states, it is important to have the most optimal measurement in order to attain the highest success probability possible. The SRM is suboptimal in the case of low and finite values of N and its result is the success probability P_s of distinguishing each set of states. However, even though this measurement is suboptimal for low and finite values on N, it is close to optimal in many cases and sometimes it is assimptotically optimal, as in [2]. Given the Gram matrix G of a family of states, the SRM is obtained as

$$P_s = \sum_n \left[\sqrt{G}_{nn}\right]^2 \tag{19}$$

5 Semidefinite Programming

Semidefinite Programming (SDP) is a type of convex optimization which deals with linear objective functions of positive semidefinite matrices subject to a linear set of constraints.

A semidefinite program is defined by the triplet (Φ, A, B), where Φ is a Hermiticity-preserving linear map and $A \in \mathcal{X}, B \in \mathcal{Y}$ are hermitian operators for some $\mathcal{X}, \mathcal{Y}$ Euclidean complex spaces [9, 10]. This triplet is associated to two different optimization problems, the primal and the dual SDP problems. However, only the primal SDP problem definition is going to be given since this work only deals with that type of optimization.

The primal SDP problem is defined as follows:

$$\begin{aligned} \text{maximize } & \langle A, X\rangle \\ \text{subject to } & \Phi(X) = B, \\ & X \geq 0 \end{aligned} \tag{20}$$

where X is the positive matrix and A is a given matrix. $\langle A, X\rangle = \mathrm{Tr}\big(A^T X\big)$ is the inner matrix product. Since A is a symmetric matrix [11], $A = A^T$ and the inner product in this case can be rewritten as $\langle A, X\rangle = \mathrm{Tr}(AX)$.

This type of optimization can also be expressed as a minimization problem. The only thing to bear in mind is that if the primal problem performs a minimization, the dual problem will be a maximization and contrariwise. All the SDPs in this work have been carried out using the python library CVXPY [11] and the performance of the optimization depends on the used solver, which might vary depending on the programming language and the software used to perform the optimization.

6 Success Probability

Due to the probabilistic nature of quantum systems, perfectly guessing the location of the edge is not possible. Therefore it is necessary to know what is the probability of getting a correct guess inferred from the results of a measurement. Previously to the calculation of the success probability, it is convenient to define what from now on is going to be called a hypothesis. Given a partition $\lambda = (n, \bar{n}) \vdash N$, a hypothesis is the situation in which the system is composed by n particles in state $|\eta\rangle$and $\bar{n}$ particles in state $|\phi\rangle$, and will be refered to hypothesis n, as shown in Fig. 2.

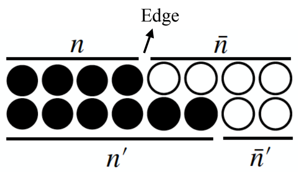


Figure 2: Two possible hypotheses n and n' given a system of $N = 8$ particles.

Having a success is defined as guessing the hypothesis n given that the system is found in the partition $\lambda = (n, \bar{n}) \vdash N$, and the success probability is defined as $P_s = p(n|n)$. In order to obtain P_s, the Gram matrix of the system has to be obtained in order to be able to perform both the SRM and SDP, which are the two types of measurements considered in this work, as it was previously mentioned.

6.1 Gram Matrix

Once the $|\Omega_n^J\rangle$ have been obtained[1] for a given N, the Gram matrix can also be obtained for a given value of J. This matrix will have components

$$\tilde{G}_{nn'}^J = \langle \Omega_n^J | \Omega_{n'}^J \rangle \tag{21}$$

where n, n' correspond to the different hypotheses. However, not all $|\Omega_n^J\rangle$ are valid in order to obtain the matrix since some of them have no projection in the angular momentum subspace we are interested in, i.e all the components of the vector are null. Therefore if the overlaps with these null vectors are calculated, the result will be null columns and rows in the Gram matrix, which will not be included in the matrix.

Note that this Gram matrix does not contain the weights c_n that appear in Eq. 7 or the prior probabilities of each of the hypotheses, which are assumed to be equiprobable. Therefore the weighted Gram matrix G^J is expressed as

$$G_{nn'}^J = \frac{1}{N}\sqrt{c_n}\sqrt{c_{n'}}\,\mathrm{Tr}(\mathbb{I}_J)\,\langle \Omega_n^J | \Omega_{n'}^J \rangle \tag{22}$$

where $\mathrm{Tr}(\mathbb{I}_J)$ is the dimension of the irreducible representation J which in $SU(2)$ is given by $\mathrm{Tr}(\mathbb{I}_J) = 2J + 1$.

The Gram matrix in the case where the states are spin-$\frac{1}{2}$ states, can also be obtained using

$$G_{nn'}^J = \frac{2J+1}{\sqrt{(n+1)(\bar{n}+1)(n'+1)(\bar{n}'+1)}}\sqrt{\frac{\binom{n}{\frac{N}{2}-J}\binom{\bar{n}'}{\frac{N}{2}-J}}{\binom{n'}{\frac{N}{2}-J}\binom{\bar{n}}{\frac{N}{2}-J}}} \tag{23}$$

where it is assumed that $n \leq n'$. With this expression, only the upper triangular part of the matrix is filled, but that is enough as the Gram matrix is symmetric, $G_{nn'} = G_{n'n}$. This expression becomes really useful as the number of particles N in the system starts growing since the process described in Eq. 22 becomes computationally costly.

[1]A detailed process for obtaining $|\Omega_n^J\rangle$ is given in Appendix. A.

6.2 Square-Root Measurement

In order to perform the SRM, $\sqrt{G}$ is required, as shown in Section. 4.2. But since the dimension of G, as it has been previously stated, grows as N and J increase, performing the square-root of G directly is not efficient from the computational point of view. Therefore it is convenient to compute $\sqrt{G}$ as

$$\sqrt{G} = \left[G^{-1}\right]^{-1/2} \tag{24}$$

since it is known that G^{-1} is tri-diagonal; i.e the only no null entries of the matrix are the diagonal terms and the ones found in its immediate upper and lower diagonal. This procedure is then used to compute all the matrices $\sqrt{G^J}$ of the system.

Once the $\sqrt{G^J}$ matrices are obtained, the SRM (P_J) of the matrix is given by Eq. 19 and the success probability will therefore be given by the sum over all the values of J as expressed below

$$P_s = \sum_J \left(\sum_i \left[\sqrt{G^J}_{ii}\right]^2 \right) \tag{25}$$

The expression for the SRM can be expanded in a Taylor series around $\varphi = 0$. In this expansion it can be seen that the quantity $(N/2)P_J$ converges to a limiting function as $N \to \infty$ provided J scales as N [12]. Therefore it is convenient to use the normalised variable $\varphi = J/(N/2)$. Using the previously mentioned Taylor series, $(N/2)P_J$ can be expressed as

$$\frac{N}{2}P(\varphi) = 2\varphi^2 - \sum_{k=3}^{\infty} a_k \varphi^{2k} \tag{26}$$

where the coefficients a_k up to order 30 are found in the Appendix. B.1. The Taylor series in Eq. 26 has no closed formula for its terms a_k or its sum. However, Eq. 26 can be treated as a “perturbative expansion” of the success probability of the SRM and in order to obtain accurate values of $(N/2)P(\varphi)$, convergence acceleration methods need to be used, specifially the Padé Approximant method [13].

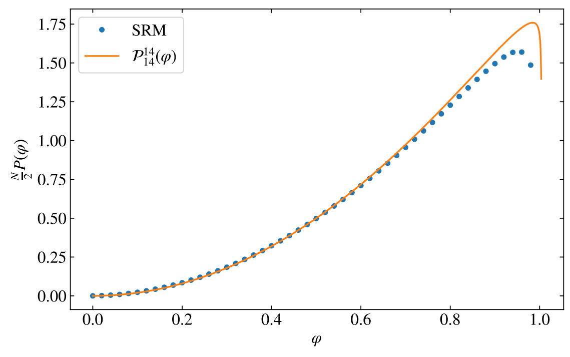
The Padé Approximant method consists of approximating a function around a specific point by a rational function of a given order. This method often provides a better approximation than the Taylor series and it may converge in the regions where a series is divergent.

The Padé Approximant is for $(N/2)P(\varphi)$ is defined as

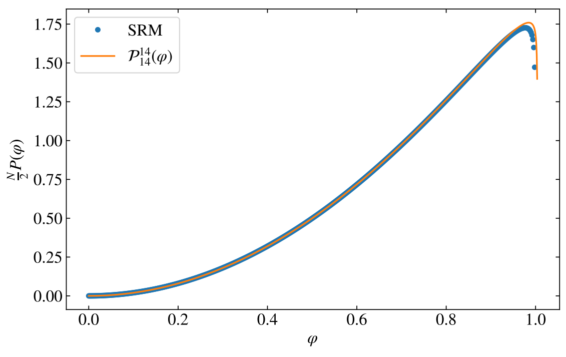
$$[n/m]_P(\varphi) = \mathcal{P}^n_m(\varphi) = \frac{\sum_{k=0}^{n} A_k \varphi^k}{\sum_{k=0}^{m} B_k \varphi^k} \tag{27}$$

where B_0 is chosen to be 1 without any loss of generality. The Taylor series[2] has been computed up to order 30 and the coefficients of $\mathcal{P}^n_m(\varphi)$ (except B_0) are chosen so that the first $n+m+1$ terms of the Padé Approximant match the $n+m+1$ terms of Eq. 26. Since $(N/2)P(\varphi)$ is an even function of φ, $\{\mathcal{P}^{2n+1}_{2n+1}\}_{n=1}^{\infty} = 0$ for $n \in \mathbb{N}$. The first terms of the diagonal sequence $\{\mathcal{P}^{2n}_{2n}\}_{n=1}^{\infty}$ have been computed[3] up to $n = 7$. Plotting the SRM, which has been obtained numerically, and the Padé Approximant as in Fig. 3 shows how good of an approximation this method yields. The larger N is, the better SRM fits $\mathcal{P}^{14}_{14}(\varphi)$.

[2]The Taylor series coefficients are available at Appendix. B.1.
[3]The coefficients can be found at Appendix. B.2.



(a) $\frac{N}{2}P(\varphi)$ and $\mathcal{P}_{14}^{14}(\varphi)$ for $N = 100$.



(b) $\frac{N}{2}P(\varphi)$ and $\mathcal{P}_{14}^{14}(\varphi)$ for $N = 1000$.

Figure 3: Comparison of how the Padé Approximant fits $\frac{N}{2}P(\varphi)$ as N grows.

Once obtained an expression for the Padé Approximant, the total success probability at leading order 1/N is obtained by integrating $(N/2)P(\varphi)$ after performing the Padé acceleration. The best approximation will be achieved with the Padé approximant of maximum degree, i.e $\mathcal{P}_{14}^{14}(\varphi)$, obtaining

$$P_s = \int_0^1 \frac{N}{2}P(\varphi)d\varphi = \int_0^1 \mathcal{P}_{14}^{14}(\varphi)d\varphi = 0.649757 \approx 0.6498 \tag{28}$$

In order to check whether this result is consistent or not, the success probability P_s is also obtained by first integrating Eq. 26, obtaining

$$P_s = \int_0^1 \frac{N}{2}P(\varphi) = \left[\frac{2}{3}\varphi^3 - \sum_{k=3}^{\infty} \frac{a_k}{2k+1}\varphi^{2k+1}\right|_0^1 \tag{29}$$

and by defining

$$Q(\varphi) \coloneqq \frac{2}{3}\varphi^3 - \sum_{k=3}^{\infty} \frac{a_k}{2k+1}\varphi^{2k+1} \tag{30}$$

Eq. 29 reduces to $Q(1)$, since $Q(0) = 0$. In order to be able to evaluate $Q(1)$ accurately, Padé acceleration is applied to $Q(\varphi)$. The Padé Approximants of $Q(\varphi)$ will be denoted as $\mathcal{Q}_m^n(\varphi) = [n/m]_Q(\varphi)$.

The first thing to note is that due to the fact that $Q(\varphi)$ is odd, the diagonal Padé sequence $\{\mathcal{Q}_n^n(\varphi)\}$ can not be constructed. Then, the convergence of $\{\mathcal{Q}_m^{2n-1}(\varphi), \mathcal{Q}_m^{2n+1}(\varphi)\}_{n=2}^7$ has to be checked, and it has been seen to rapidly converge to $P_s = 0.6501$, which is consistent with the result in Eq. 28. Since the two results are not identical, they can be averaged, hence summarizing all the previous analysis in

$$P_s = 0.6499 \pm 0.0002 \tag{31}$$

In order to show how powerful and valuable the Padé analysis is, a Toy Model for the SRM is presented next.

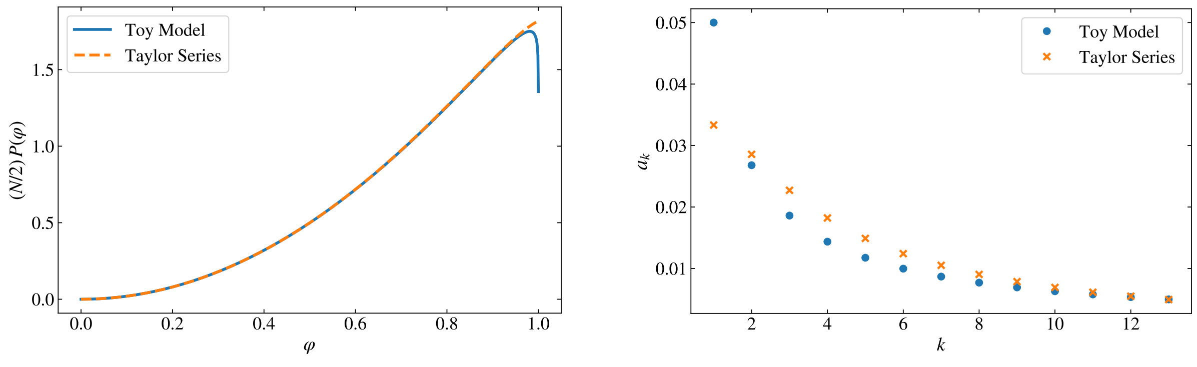
6.2.1 Toy Model

Padé approximants do not have rigorous results on all types of functions. In fact, rigorous results on Padé approximants only exist for very restricted families of functions, to where the function obtained for the SRM in Eq. 26 does not seem to belong [12]. However, Padé approximants are applied in more general situations across a variety of fields including physics, where they have been proved to be very valuable and powerful.

The following toy model for the Taylor series of $(N/2)P(\varphi)$ in Eq. 26 is considered:

$$f(\varphi) = 2\varphi^2 - \sum_{k=3}^{\infty} \frac{\varphi^{2k}}{20k^{\frac{9}{10}}} \tag{32}$$

whose coefficients are very similar to those in Eq. 26, as seen in Fig. 4b. As shown, the coefficients of the two functions are very similar showing no significant difference in the coefficients a_k with $k \geq 8$.



(a) Toy model and Taylor series.

(b) Toy model and Taylor series coefficients.

Figure 4: Comparison of the Toy Model and the Taylor series.

Contrairwise to Eq. 26, the series in the function defined for the toy model can be summed up, resulting in the function to be

$$f(\varphi) = 2\varphi^2 - \frac{\varphi^4}{20}\mathrm{Li}_{\frac{9}{10}}(\varphi^2) \tag{33}$$

where $\mathrm{Li}_n(x)$ is the polylogarithm [14]. The behaviour of $f(\varphi)$ is shown at Fig. 4a, in comparison with the Taylor series of the SRM. It describes incredibly well the behaviour of $(N/2)P(\varphi)$ but the goal of this toy model is to show how useful and powerful the Padé analysis used previously can be. In order to do so, some information will be extracted from the toy model once it has been summed up. This information cannot be obtained directly for small values of k from the series in Eq. 32. In particular, what's desired to be found is f_{max}, the place where φ_{max} takes the value $f(\varphi_{max}) = f_{max}$, both quantities in the closed interval $[0, 1]$ and $\int_0^1 f(\varphi)d\varphi$. All these values can be obtained from Eq. 33 and to four significant digits, they happen to be

$$f_{max} = 1.750; \quad \varphi_{max} = 0.9810, \quad \int_0^1 f(\varphi)d\varphi = 0.6480 \tag{34}$$

If the Padé analysis is performed assuming that the only known terms of the sequence are the corresponding to $k \leq 10$, it can be seen that even though the Taylor Series does not provide accurate results, the Padé approximants successfully provide accurate results for the desired quantities. The result of this analysis can be seen in Table. 1.

Table 1: Results of the quantities in Eq. 34 using the Taylor series and Padé approximants.

k	f_{max}		φ_{max}		$\int_0^1 f(\varphi)d\varphi$	
	Taylor	Padé	Taylor	Padé	Taylor	Padé
3	4.869	1.839	1.911	1.000	0.6595	0.6512
4	3.244	1.793	1.503	1.000	0.6565	0.6498
5	2.677	1.766	1.333	0.990	0.6549	0.6492
6	2.394	1.756	1.240	0.985	0.6538	0.6488
7	2.228	1.753	1.181	0.983	0.6530	0.6486
8	2.120	1.751	1.142	0.982	0.6524	0.6484
9	2.046	1.751	1.113	0.981	0.6519	0.6484
10	1.992	1.751	1.092	0.981	0.6516	0.6483

As it can be seen from Table. 1 the Padé approximant method converges to the value of f_{max} from $k = 8$ whereas the value obtained using the Taylor series is far from being precise and far from converging to the correct value. In the case of φ_{max}, the Padé analysis provides an accuracy of two decimal places since $k = 6$ and 3 decimal places from $k = 9$. However, the Taylor series does not even get an accuracy of a first decimal place in all the analysis. Lastly, the integral value coincides to three decimal places from $k = 6$ in the case of the Padé approximant, contrary to the Taylor series which only gets to the first decimal place of precision.

Therefore, the Padé approximant method proves to be very useful and powerful and due to the similarity of the used function with the function that appears in the SRM, it makes sense to use the Padé acceleration method within the SRM to approximate $(N/2)P(\varphi)$.

6.3 Semidefinite Programming

The other type of measurement that is going to be performed on the system is the optimal measurement, characterised by a POVM, whose elements will be obtained encoding the measurement process as a SDP problem[4].

In order to reduce the dimensionality of the problem, each column of $\sqrt{G^J}$ will be taken as a new state $|\omega_n\rangle$ with which a set of density matrices $\{\rho_n = |\omega_n\rangle\langle\omega_n|\}$ is constructed, where ρ_n corresponds to the density operator when the system is at hypothesis n. This density operator encodes all the relevant information of the problem. The measurement elements will be characterised by a set of positive operators $\{\Pi_n\}$ where Π_n corresponds to measuring the hypothesis n. Then, the success probability in terms of a primal SDP problem can be expressed as

$$P_J = \sum_n \max\{\mathrm{Tr}(\Pi_n \rho_n)\} \tag{35}$$

subject to $\sum_n \Pi_n = \mathbb{I}$; $\Pi_n \geq 0$, which are the conditions required for $\{\Pi_n\}$ to be a POVM. The result of the SDP will yield both the optimal measurement elements and the maximum success probability.

Since $\rho_n = |\omega_n\rangle\langle\omega_n|$, the primal SDP problem can be rewritten as

$$P_J = \sum_n \max\{\mathrm{Tr}(\Pi_n |\omega_n\rangle\langle\omega_n|)\} = \sum_n \max\{\mathrm{Tr}(\langle\omega_n|\Pi_n|\omega_n\rangle)\} = \sum_n \max\{\langle\omega_n|\Pi_n|\omega_n\rangle\} \tag{36}$$

where the cyclic property of the trace has been used. Then, the total success probability for a given N will

[4]All the POVM elements obtained using SDP, alongside with the Gram matrices are available at the following OneDrive Folder.

be given by the sum of the success probabilities for each J as

$$P_s = \sum_J P_J \tag{37}$$

where, as with the SRM, the sum over J is over the possible angular momentum values.

SDP gives as a result the optimal measurement and the maximum success probability. However, as seen in Fig. 5 as N grows the success probability obtained by the SDP and the SRM tend to the same value since in this case, as in [2], the SRM is assimptotically optimal. However, for finite values of N the SDP will always obtain a better P_s than the SRM.

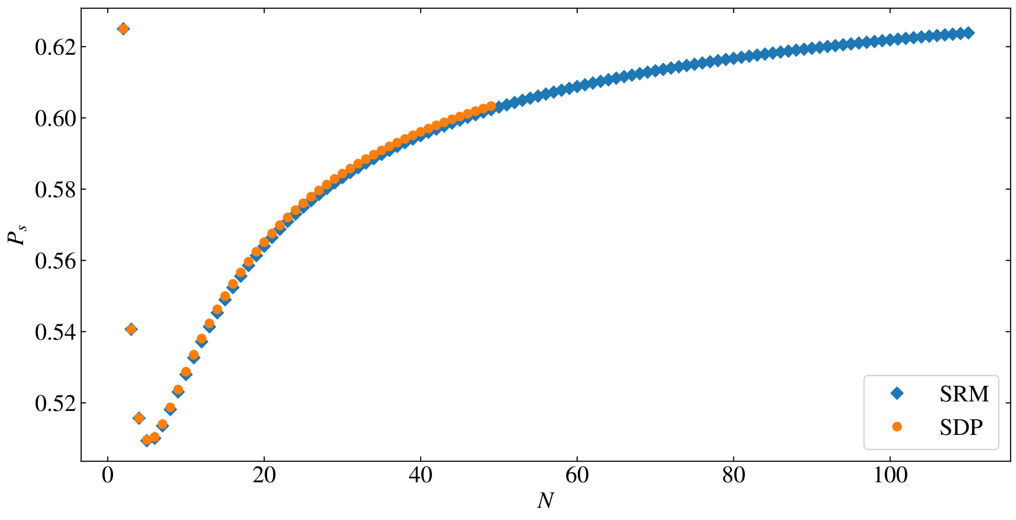


Figure 5: Success probability up to $N = 49$ obtained with the SDP and up to $N = 110$ with the SRM.

The SDP optimization has only been obtained up to $N = 49$ since the optimization becomes computationally costly from that value of N.

After the success probability is known, it is also interesting to be able to gain knowledge on the error that is made when making a prediction of where the edge is located, understanding by error the number of particles by which the guess is wrong from the real edge.

7 Expected Value of the Hamming Distance

The Hamming distance can be used to quantify the average error made when guessing a hypothesis. Given two binary arrays $\mathbf{x}, \mathbf{x}'$ of an arbitrary size, the Hamming distance between them is defined as

$$h(\mathbf{x}, \mathbf{x}') = |\mathbf{x} - \mathbf{x}'| \tag{38}$$

in this case, $h(n, \hat{n}) = |n - \hat{n}|$. The expected value of the Hamming distance given the hypothesis n and the guess $\hat{n}$ is

$$\langle h \rangle = \sum_{n,\hat{n}} h(n, \hat{n}) p(\hat{n}|n) \tag{39}$$

where $p(\hat{n}|n)$ is the probability of guessing the hypothesis $\hat{n}$ given that the system is in the hypothesis n. Since the work is carried with a set of density matrices $\{\rho_n = |\psi_n\rangle\langle\psi_n|\}$, the probability can be expressed as

$$p(\hat{n}|n) = \mathrm{Tr}(\Pi_{\hat{n}} \rho_n) = \mathrm{Tr}(\Pi_{\hat{n}} |\psi_n\rangle\langle\psi_n|) = \langle\psi_n|\Pi_{\hat{n}}|\psi_n\rangle \tag{40}$$

where $\{\Pi_{\hat{n}}\}$ are the measurement operators for each guess $\hat{n}$ given the system is made of N particles. As with the success probability, $\langle h\rangle$ is going to be computed using the SRM and SDP.

7.1 $\langle h\rangle$ using Square Root Measurement

When computing the success probability, the SRM consisted on the sum of the diagonal elements of the squared elements on the diagonal of $\sqrt{G}$, since those terms correspond to guessing hypothesis n given the system is in that same hypothesis. Therefore, in order to take into account all the possible hypotheses and guesses, all the terms in $\sqrt{G}$ have to be included, hence obtaining

$$\langle h\rangle_{SRM} = \sum_J \left[\sum_{n,\hat{n}} |n-\hat{n}| \left[\sqrt{G^J}_{n,\hat{n}}\right]^2\right] \tag{41}$$

where, as in the previous sections, the sum over J is over all the possible angular momentum values.

Contrary to the success probability, $\langle h\rangle$ using the SRM does not converge to any limiting function as $N \to \infty$ and therefore it is not possible to apply the Padé Approximant method to see if $\langle h\rangle$ converges to a specific value as N grows. However there is work in progress in order to see if there is any other quantity that converges to a specific value.

7.2 $\langle h\rangle$ using Semidefinite Programming

As for the success probability, $\langle h\rangle$ can also be written in terms of a primal SDP problem in order to obtain an optimal value for the expected hamming distance value.

In order to obtain the optimal $\langle h\rangle$, the SDP problem must be a minimization due to the fact that in order for $\langle h\rangle$ to be optimal, it needs to be as low as possible, meaning that the error made when guessing is the smallest possible. Therefore, expressing Eq. 39 as a primal SDP problem lays

$$\langle h\rangle_{SDP} = \min\left\{\sum_{n,\hat{n}} |n-\hat{n}| \operatorname{Tr}(\Pi_{\hat{n}}\rho_n)\right\} = \min\left\{\sum_{n,\hat{n}} |n-\hat{n}| \langle\psi_n|\Pi_{\hat{n}}|\psi_n\rangle\right\} \tag{42}$$

subject to

$$\begin{aligned} &\rho_n \geq 0 \\ &\Pi_{\hat{n}} \geq 0 \\ &\sum_{\hat{n}} \Pi_{\hat{n}} = \mathbb{I} \end{aligned} \tag{43}$$

The results of $\langle h\rangle$ both for the SRM and the SDP are shown in Fig. 6 below. The SDP has only been performed up to $N = 88$ since the computational cost increases significantly.

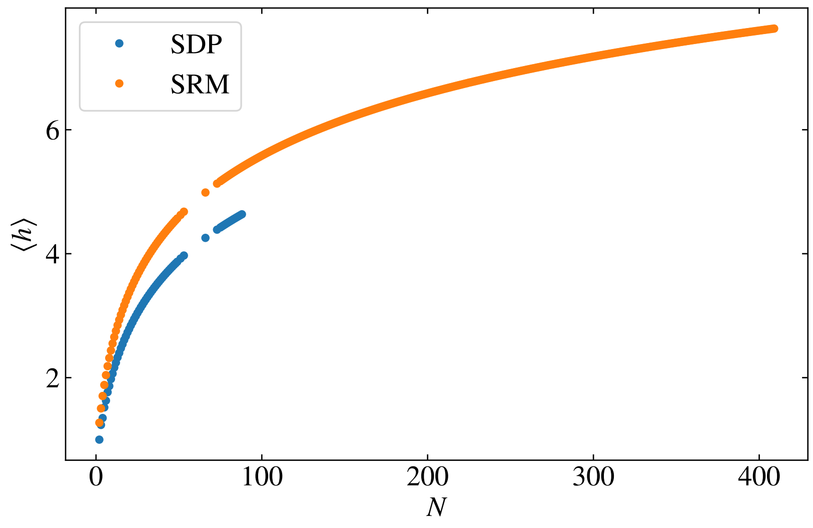


Figure 6: $\langle h \rangle$ using SDP and SRM.

As it was expected, the error made when using the POVM obtained with the SDP is lower than the result obtained with the SRM since the SDP gives as an output the optimal measurement.

8 Simulating Edge Detection

The main goal of this work is to be able to predict where the sudden change on the states happens. In order to do so, as mentioned in the introduction, some measurements are performed on the whole chain of states. After the change, once the edge is crossed the states $|\phi\rangle$ point differ from the initial states by an angle θ. Without loss of generality, the initial states $|\eta\rangle$ can be taken to be $|0\rangle$ and the states $|\phi\rangle$ can then be expressed as $|\phi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\varphi}\sin(\theta/2)\,|1\rangle$. Since all the states $|\phi\rangle$ are pointing in the same direction, instead of coupling in pairs as suggested in Section. 3, all the n states $|\eta\rangle$ can be considered a single state of $J = M = n/2$ and the $\bar{n}$ $|\phi\rangle$ states can be considered a single state of $J = M = \bar{n}/2$. Therefore, the system can be expressed as

$$\left|\frac{n}{2},\frac{n}{2}\right\rangle_z \otimes \left|\frac{\bar{n}}{2},\frac{\bar{n}}{2}\right\rangle_\theta \tag{44}$$

where the sub-index θ accounts for the direction the $|\phi\rangle$ states are pointing towards in the Bloch-Sphere with respect to the $|\eta\rangle$ states. Then the Wigner $\mathcal{D}$-matrices [4, 5, 15] can be used in order to rotate the $|\phi\rangle$ states and express them in their projections with the direction where the $|\eta\rangle$ states are pointing towards. Once it has done, they are coupled using CG coefficients obtaining the projections in the coupled angular momentum basis. This whole process is given by

$$\begin{aligned}|\psi\rangle = \left|\frac{n}{2},\frac{n}{2}\right\rangle_z \otimes \left|\frac{\bar{n}}{2},\frac{\bar{n}}{2}\right\rangle_\theta &= \left|\frac{n}{2},\frac{n}{2}\right\rangle_z \otimes \sum_{m'=-\bar{n}/2}^{\bar{n}/2} d^{\bar{n}/2}_{m',\bar{n}/2}(\theta)\left|\frac{\bar{n}}{2},m'\right\rangle_z \\ &= \sum_{J=\left|\frac{n-\bar{n}}{2}\right|}^{\frac{n+\bar{n}}{2}} \left[\sum_{m'=-\bar{n}/2}^{\bar{n}/2} d^{\bar{n}/2}_{m',\bar{n}/2}(\theta)\, C^{J,\frac{n}{2}+m'}_{\frac{n}{2},\frac{n}{2},\frac{\bar{n}}{2},m'}\left|J,\frac{n}{2}+m'\right\rangle_z\right]\end{aligned} \tag{45}$$

In order to see how good the detection is, an edge is assigned at random. With that, the first measurement that is performed is on the angular momentum, where the probability of obtaining angular momentum J is given by

$$P(J) = \sum_{m=-J}^{J} |\langle J,m|\psi\rangle|^2 = \sum_{m=-\frac{\bar{n}}{2}}^{\frac{\bar{n}}{2}} \left|d^{\bar{n}/2}_{m,\bar{n}/2}(\theta)\, C^{J,\frac{n}{2}+m}_{\frac{n}{2},\frac{n}{2},\frac{\bar{n}}{2},m}\right|^2 \tag{46}$$

which will ultimately depend on the angle that the two states form in the Bloch sphere θ since $|\psi\rangle$ depends on the angle.

Once this measurement is done, the second measurement is performed in order to obtain the probability of guessing n using either the SRM or the POVM elements obtained with the SDP. In the case of a system of $N = 20$ particles and having chosen $n = 10$, the probability distributions for n for 4 different angles θ are shown in Fig. 7.

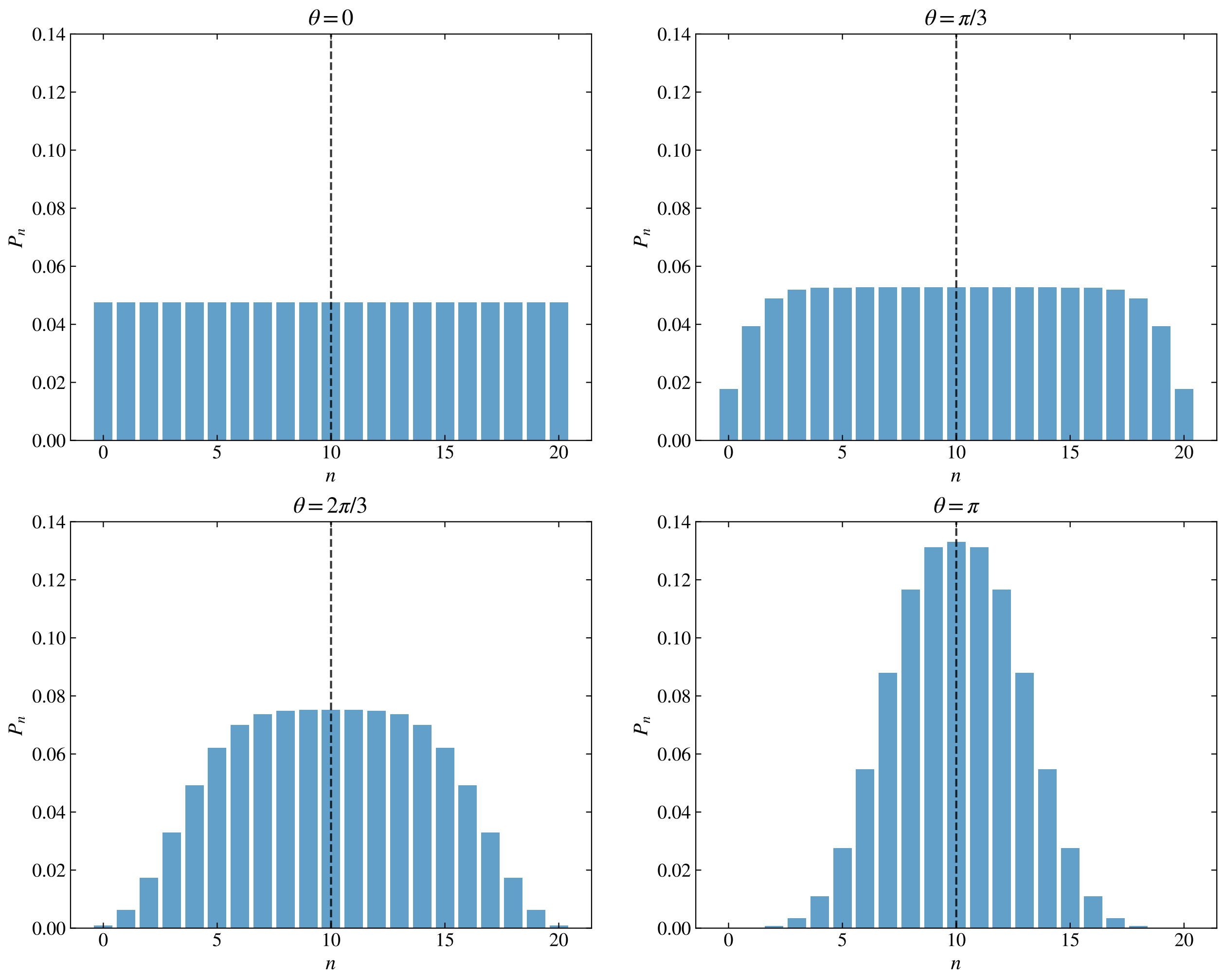


Figure 7: Probability distribution of the edge detection given $N = 20$ and $n = 10$. The dashed line corresponds to the location of the real edge.

As it was expected, when the angle $\theta = 0$, the states $|\phi\rangle$ correspond to $|\eta\rangle$ meaning that there is no real edge and the probability of correctly guessing becomes uniform. As θ starts growing, the probability of guessing where the edge is increases until $\theta = \pi$, where $|\eta\rangle$ is orthogonal to $|\phi\rangle$ and the performance of the detection is maximum.

9 Conlusions

Throughout this work the necessary elements required to perform edge detection on a chain of quantum states have been presented and calculated using two different types of measurements: an optimal measurement and a suboptimal one which asymptotically becomes optimal.

In the case of the non optimal measurement, the Square Root Measurement, it has been shown with a consistent and powerful method that the success probability converges to a value of $P_s = 0.6499 \pm 0.0002$ as the number N of states in the chain tends to infinity.

Moreover a Toy Model for the Square Root Measurement has been presented in order to test the performance of the convergence acceleration method used, due to the resemblance of the model to the Square Root Measurement's Taylor expansion. It has been found that the use of the Padé Approximant in this toy model provides precise results where the evaluation of the truncated Taylor expansion fails to do so. This motivates the use of this convergence acceleration method for the success probability using the Square Root Measurement.

Alongside the success probability, the expected Hamming distance has been calculated for the two types of measurements, providing with the average error made when predicting where the chain's edge is located. As expected, the Hamming distance using the optimal measurement is lower than the obtained using the sub optimal measurement. It would be tempting to perform the same analysis as with the success probability using Padé Approximants but it is not possible due to the fact that the power series expansion of the Hamming distance expected value in the case of the Square Root Measurement is deeply divergent. However, there is still work in progress in order to be able to perform a similar analysis as with the SRM with a quantity that behaves better than the Hamming distance.

After all the previous calculations a probability distribution has been obtained, which shows how good is the edge detection protocol given specific states for each domain and a specific location of the edge, using two measurements on the whole chain of states, one measurement on the angular momentum and a second one on the symmetric group $\mathcal{S}_N$. This detection, as expected, ultimately depends on the angle θ that the unknown states $|\eta\rangle$ and $|\phi\rangle$ form in the Bloch-Sphere and it has happened to be the most efficient when the states are orthogonal, as one could initially expect.

References

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[2] Gael Sentís, E. Bagan, John Calsamiglia, G. Chiribella, and Ramon Munoz Tapia. *Quantum Change Point*. *Physical Review Letters*, 117:150502, 10 2016.

[3] Koenraad M.R. Audenaert. A digest on representation theory of the symmetric group, 2006.

[4] Jun John Sakurai and Jim Napolitano. *Modern quantum mechanics; 3rd ed.* Addison-Wesley, San Francisco, CA, 2011.

[5] D. A. Varshalovich, A. N. Moskalev, and V. K. Khersonskii. *Quantum Theory of Angular Momentum*. World Scientific, 1988.

[6] Wikipedia contributors. Clebsch–gordan coefficients — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Clebsch-Gordan_coefficients, 2022.

[7] Particle Data Group (PDG). *Clebsch-Gordan coefficients, Spherical Harmonics and d-functions*. https://pdg.lbl.gov/2018/reviews/rpp2018-rev-clebsch-gordan-coefs.pdf.

[8] Emili Bagan. *Notes d'Informació i Computació Quàntiques*, 2007.

[9] Lieven Vandenberghe and Stephen Boyd. *Semidefinite Programming*. *SIAM Review*, 38(1):49–95, 1996.

[10] John Watrous. *Theory of Quantum Information*, 2011.

[11] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[12] Emili Bagan Capella. *Quantum Edge Detection*, 2022.

[13] C.M.Bender and S.A.Orszag. *Advanced Mathematical Methods for Scientists and Engineers I: Assymptotic Methods and Perturbation Theory*. McGraw-Hill,Inc, 1978.

[14] *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.1.6 of 2022-06-30. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

[15] Emili Bagan. *Notes de Teoria de Grups*.

A Construction of Ω_n^j Projectors

In order to construct the Ω_n^j, it is necessary to define the vector $\boldsymbol{\alpha}$ as the vector whose i-th component is the partial angular momentum from coupling i spin-$\frac{1}{2}$ particles. Therefore:

$$\boldsymbol{\alpha} = (j_1, j_{12}, j_{123}, ..., j_{1...N}) = \left(\frac{1}{2}, j_{12}, ..., J\right) \tag{47}$$

If J is fixed, each $\boldsymbol{\alpha}$ can be associated to a way of coupling the N particles in such a way that the system has angular momentum J. In order to have a better understanding of this, the simple case in which $N = 4$ and $J = 1$ will be used as an example throughout all the section. In this case, there are three possible paths that give $J = 1$. These paths are:

$$\begin{aligned} \boldsymbol{\alpha}_1 &= \left(\frac{1}{2}, 1, \frac{3}{2}, 1\right) \\ \boldsymbol{\alpha}_2 &= \left(\frac{1}{2}, 1, \frac{1}{2}, 1\right) \\ \boldsymbol{\alpha}_3 &= \left(\frac{1}{2}, 0, \frac{1}{2}, 1\right) \end{aligned} \tag{48}$$

which can be graphically represented as in Fig. 8.

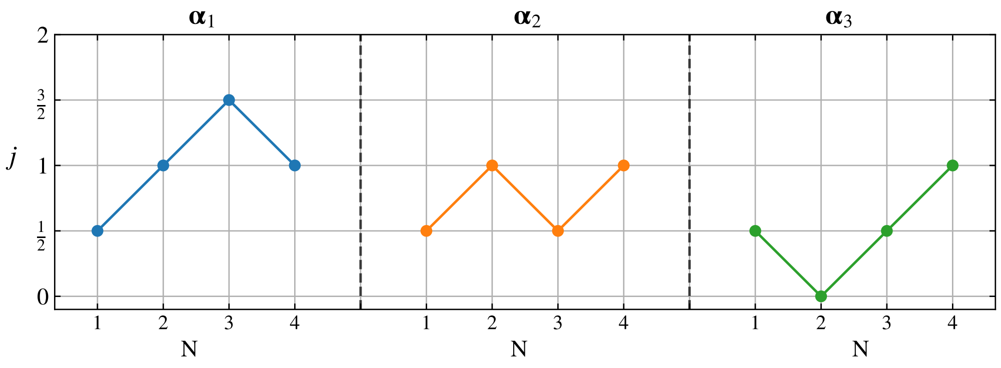


Figure 8: Different paths for $J = 1$ in the case of $N = 4$.

Since the system studied is made of $N = n + \bar{n}$ spin-$\frac{1}{2}$ particles, the basis elements of the Hilbert space spanned are expressed as

$$\left|\frac{n}{2}, m\right\rangle \otimes \left|\frac{\bar{n}}{2}, \bar{m}\right\rangle = \sum_j |j, m + \bar{m}\rangle \otimes \sum_{\boldsymbol{\alpha}'} \tilde{\Omega}_{\boldsymbol{\alpha}'}^{j,n} |\boldsymbol{\alpha}'\rangle \tag{49}$$

where the states $|j, m + \bar{m}\rangle$ act on $SU(2)$, and the part to the right of the tensor product acts on the symmetric group $\mathcal{S}_N$.

The previous expression can be thought as if the N states are organised in a row, havin the n states of the first type. first, followed by the $\bar{n}$ states of the other type. Since this work is carried for all the states to be spin-$\frac{1}{2}$ particles, the first n elements can be described as spin up states $\left|\frac{1}{2}, \frac{1}{2}\right\rangle$, and the rest as spin down states $\left|\frac{1}{2}, -\frac{1}{2}\right\rangle$. Therefore, the basis elements can be expressed as

$$\left|\frac{1}{2}, \frac{1}{2}\right\rangle^{\otimes n} \otimes \left|\frac{1}{2}, -\frac{1}{2}\right\rangle^{\otimes \bar{n}} = \sum_j |j, m + \bar{m}\rangle \otimes \sum_{\boldsymbol{\alpha}'} \tilde{\Omega}_{\boldsymbol{\alpha}'}^{j,n} |\boldsymbol{\alpha}'\rangle \tag{50}$$

The next step is to define a vector which contains the third components of spin of the N particles. That vector will be denoted as $\boldsymbol{s}$ and will be defined as

$$\boldsymbol{s} = \left(\overbrace{\frac{1}{2}, ..., \frac{1}{2}}^{n}, \overbrace{-\frac{1}{2}, ..., -\frac{1}{2}}^{\bar{n}}\right). \tag{51}$$

Another vector $\boldsymbol{w}$ will be defined by components as

$$w_i = \sum_{k=1}^{i} s_i, \tag{52}$$

which contains the cumulative sum of $\boldsymbol{s}$.

Due to the election in Eq. 50 of ordering the basis states in that way, some of the paths $\boldsymbol{\alpha}$ will not be allowed, since the partial angular momentum $\boldsymbol{\alpha}_i$ may be lower than the sum of the third components of the spin $\boldsymbol{w}_i$, as shown below continuing with the example of $N = 4$, $J = 1$ and $n = 2$.

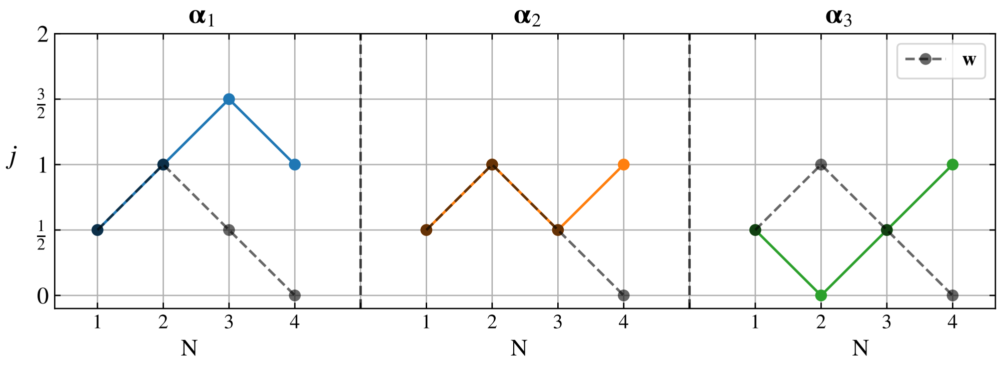


Figure 9: Different paths for $J = 1$, $N = 4$ and $n = 2$.

In this case, the only compatible paths are $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$, since for the $\boldsymbol{\alpha}_3$ path, $\boldsymbol{\alpha}_3^{(2)} < \boldsymbol{w}^{(2)}$, which can not occur. In this case, the index in brackets indicates the component of the vector. As a consequence, in Eq. 50 this path will be excluded and the sum will be over $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$. The paths which are possible, and therefore taken into account have to fulfill the condition

$$\min\{\alpha_k^{(i)} - w^{(i)}\} \geq 0 \tag{53}$$

where $\alpha_k^{(i)}$ is the i-th component of $\boldsymbol{\alpha}_k$ and $w^{(i)}$ is the i-th component of $\boldsymbol{w}$.

Since $\tilde{\Omega}^j_n$ is a rank-1 projector,instead of calculating all its elements $\langle\alpha|\tilde{\Omega}^j_n|\alpha'\rangle$ it is sufficient to obtain its diagonal elements $\langle\alpha|\tilde{\Omega}^j_n|\alpha\rangle$ which are given by

$$\tilde{\Omega}^j_{n,\alpha} = \prod_{i=0}^{N-1} \left\langle \alpha_i, w_i; \frac{1}{2}, s_i \middle| \alpha_{i+1}, w_{i+1} \right\rangle \tag{54}$$

where $\tilde{\Omega}^j_{n,\alpha}$ is a component of $|\tilde{\Omega}^j_n\rangle$. $\left\langle \alpha_i, w_i; \frac{1}{2}, s_i \middle| \alpha_{i+1}, w_{i+1} \right\rangle$ can be identified as CG coefficients.

In order to obtain $\left|\Omega^j_n\right\rangle$, the vector obtained has to be normalised, i.e

$$\left|\Omega^j_n\right\rangle = \frac{\tilde{\Omega}^j_{n,\alpha}}{\sqrt{\sum_{\alpha}\left(\tilde{\Omega}^j_{n,\alpha}\right)^2}} \tag{55}$$

B Taylor Series and Padé Approximant Coefficients

In this appendix the coefficients for the Taylor Series of P_j obtained using the SRM are given, along the coefficients for some of the Padé Approximants that have been calculated.

B.1 Taylor Series Coefficients

Table 2: Taylor series coefficients for P_j up to order 30.

k	a_{2k}	a_{2k+1}
1	-	$\frac{1}{30}$
2	$\frac{1}{35}$	$\frac{229}{10\,080}$
3	$\frac{101}{5\,544}$	$\frac{5\,725}{384\,384}$
4	$\frac{5\,111}{411\,840}$	$\frac{554\,293}{52\,715\,520}$
5	$\frac{2\,137\,221}{236\,487\,680}$	$\frac{249\,919\,231}{31\,783\,944\,192}$
6	$\frac{76\,576\,105}{11\,076\,222\,976}$	$\frac{10\,870\,862\,389}{1\,772\,195\,676\,160}$
7	$\frac{8\,413\,125\,001}{1\,533\,630\,873\,600}$	$\frac{1\,506\,595\,197\,973}{304\,973\,453\,721\,600}$

B.2 Padé Approximant Coefficients

Table 3: Coefficients for the Padé approximant $\mathcal{P}_4^4(\varphi)$.

k	A_{2k}	B_{2k}
0	0	1
1	2	-0.857143
2	-1.74129	0.016667

Table 4: Coefficients for the Padé approximant $\mathcal{P}_6^6(\varphi)$.

k	A_{2k}	B_{2k}
0	0	1
1	2	-0.64639
2	-1.28078	-0.132642
3	-0.298617	0.00361255

Table 5: Coefficients for the Padé approximant $\mathcal{P}_8^8(\varphi)$.

k	A_{2k}	B_{2k}
0	0	1
1	2	-1.69541
2	-3.39083	0.803865
3	1.5744	-0.0800584
4	-0.132174	0.000536673

Table 6: Coefficients for the Padé approximant $\mathcal{P}^{12}_{12}(\varphi)$.

k	A_{2k}	B_{2k}
0	0	1
1	2	-2.56162
2	-5.12324	2.32816
3	4.62288	-0.87898
4	-1.70114	0.120603
5	0.214071	-0.00445401
6	-0.00614975	0.0000123132

Table 7: Coefficients for the Padé approximant $\mathcal{P}^{14}_{14}(\varphi)$.

k	A_{2k}	B_{2k}
0	0	1
1	2	-3.47961
2	-6.95921	4.68403
3	9.33473	-3.03572
4	-5.98403	0.951594
5	1.82375	-0.125282
6	-0.22237	0.00507884
7	0.00725633	-0.0000178728

C Codes

All the programming work in this BSc thesis has been done using Python and Wolfram Mathematica. In this appendix all the necessary codes to replicate this work can be found, except for the codes used for plotting purposes. All the codes generated using python have a cream coloured backgroud, whereas the codes created with Wolfram Mathematica have a white background.

All the codes that appear in this appendix can also be found at the following GitHub Repository.

C.1 Calculation of Ω^j_n, Gram Matrices and Success Probabilities for SDP and SRM

```python
"""
CONTENT:

- Omega Vectors
- Weighted Gram Matrix
- SQRT Measurement
- SDP
"""

import numpy as np
import sympy as smp
from scipy.linalg import sqrtm
import matplotlib.pyplot as plt
from sympy.physics.quantum.cg import CG
from sympy.utilities.iterables import multiset_permutations
import cvxpy as cp

N = 11

#List of all the possible J values given N

J_possibles = []

```

```python
if N%2 == 0:
    for i in range(int(N/2)+1):
        J_possibles.append(i)
else:
    for i in range(int(N/2)+1):
        J_possibles.append(1/2+i)

#Function that calculates the Omega Projectors

def Omega_Vector(N,n,J):
    s = []
    Alphas = []

    for i in range(n):
        s.append(smp.Rational(1,2))
    for i in range(N-n):
        s.append(smp.Rational(-1,2))

    w = np.cumsum(s)

    J_max = N/2
    Steps_down = int(J_max - J)

    A = []
    for i in range(Steps_down):
        A.append(smp.Rational(-1,2))
    for i in range(N-Steps_down):
        A.append(smp.Rational(1,2))


    S = list(multiset_permutations(A))
    for i in S:
        if ((all(x >= 0 for x in np.cumsum(i)) == True) and (all(y>=0 for y in np.cumsum(i)-w
    ))):
            Alphas.append(np.cumsum(i))

    Omega = []

    for Alfa in Alphas:
        Prod = 1
        for i in range(N-1):
            Prod = Prod*CG(Alfa[i],w[i],smp.Rational(1,2),w[i+1]-w[i],Alfa[i+1],w[i+1]).doit
    ()
        Omega.append(Prod)


    Norm = sum(i*i for i in Omega)
    if Norm == 0:
        Norm = 1

    Omega_Norm = []
    for i in Omega:
        Omega_Norm.append(i/smp.sqrt(Norm))

    return Omega_Norm[::-1]


#Array for the probabilities both on SRM and SDP

psq=[]
psdp = []

#Documents where the success probability for N will be exported for SRM & SDP
SDP_doc = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/SDP","a")
SQRTM_doc = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/SQRTM","a")

for j in range(len(J_possibles)):
```

```

#Documents where the Gram matrices, the Sqrt(G) and the POVM elements of the SDP will be exported
Gram_Sqrt = open("/Users/marcel/Desktop/TFG//Matrices/GSQ/GSQ,N={},J={}".format(N,J_possibles[j]),"w")
Gram_Mat = open("/Users/marcel/Desktop/TFG/Matrices/Gram_Matrices/Gt,N={},J={}".format(N,J_possibles[j]),"w")
POVM = open("/Users/marcel/Desktop/TFG/Matrices/POVM_elements/POVM,N={},J={}".format(N,J_possibles[j]),"w")


Omegas = []
n1 = []

for n in range(N+1):
    if all(x == 0 for x in Omega_Vector(N,n,j)) == True:
        continue
    else:
        Omegas.append(Omega_Vector(N,n,j))
        n1.append(n)

#Export an array with all the hypotheses that project on angular momentum j
np.save("/Users/marcel/Desktop/TFG/Projecting_n/N={},J={}.npy".format(N,J_possibles[j]),n1)

#Define an empty G matrix
G = smp.zeros(len(Omegas),len(Omegas))

#Fill the G matrix

for i in range(len(Omegas)):
    for m in range(len(Omegas)):
        if len(Omegas[i])==len(Omegas[m]):
            for k in range(len(Omegas[i])):
                G[i,m] = G[i,m] + Omegas[i][k]*Omegas[m][k]

        elif len(Omegas[i])<len(Omegas[m]):
            for k in range(len(Omegas[i])):
                G[i,m] = G[i,m] + Omegas[i][k]*Omegas[m][k]

        else:
            for k in range(len(Omegas[m])):
                G[i,m] = G[i,m] + Omegas[i][k]*Omegas[m][k]

#Weight the G matrix
Di = smp.zeros(len(Omegas),len(Omegas))
for i in range(len(Omegas)):
    Di[i,i] = ((np.sqrt(2*J_possibles[j]+1))/(np.sqrt(n1[i]+1)*np.sqrt(N-n1[i]+1)))

#Gt is the weighted Gram matrix
Gt = (Di*G*Di).evalf()

#Sqrt(G)
GSQ = sqrtm(np.array(Gt).astype(np.float64))

if J_possibles[j] == J_possibles[-1]:
    GSQ1 = GSQ
    GSQ = np.zeros((len(Omegas), len(Omegas)))
    for i in range(len(Omegas)):
        for k in range(len(Omegas)):
            GSQ[i][k] = GSQ1[i][k].real

#Save Sqrt(G) as an array
np.save("/Users/marcel/Desktop/TFG/Matrices/GSQ_NPY/GSQ,N={},J={}.npy".format(N,J_possibles[j]),GSQ)

#Export G, and Sqrt(G) in matrix form
```

```
    Gram_Mat.write("[")
    for i in range(len(Omegas)):
        Gram_Mat.write("[")
        for j in range(len(Omegas)):
            if j == len(Omegas)-1:
                Gram_Mat.write("{}".format(Gt[i,j]))
            else:
                Gram_Mat.write("{},".format(Gt[i,j]))
        Gram_Mat.write("]")
        if i != len(Omegas)-1:
            Gram_Mat.write(",")
    Gram_Mat.write("]")
    Gram_Mat.close()

    Gram_Sqrt.write("[")
    for i in range(len(Omegas)):
        Gram_Sqrt.write("[")
        for j in range(len(Omegas)):
            if j == len(Omegas)-1:
                Gram_Sqrt.write("{}".format(GSQ[i,j]))
            else:
                Gram_Sqrt.write("{},".format(GSQ[i,j]))
        Gram_Sqrt.write("]")
        if i != len(Omegas)-1:
            Gram_Sqrt.write(",")
    Gram_Sqrt.write("]")
    Gram_Sqrt.close()

    #SRM
    Sqrt_Measurement = 0

    for i in range(len(Omegas)):
        Sqrt_Measurement += np.round(GSQ[i,i]**(2),6)

    psq.append((1/N)*Sqrt_Measurement)


    #SDP
    omega_tilda = []
    for i in range(len(Omegas)):
        omega_tilda.append(GSQ[:,i])

    #Definition of variables and constraints of SDP
    X = []
    for i in range(len(omega_tilda)):
        X.append(cp.Variable((len(omega_tilda[0]),len(omega_tilda[0])),PSD=True))

    constraints = [sum(X) == np.identity(len(omega_tilda[0]))]

    #Define the problem and Solve it
    P = 0
    for i in range(len(omega_tilda)):
        P += (1/N)*omega_tilda[i]@ X[i]@ omega_tilda[i]

    Prob = cp.Problem(cp.Maximize(P),constraints)
    Prob.solve()

    psdp.append(Prob.value)

    #Export the POVM elements in matrix form
    for i in range(len(X)):
        POVM.write("[")
        for l in range(len(X[i].value)):
            POVM.write("[")
            for j in range(len(X[i].value)):
                if j == len(X[i].value)-1:
                    POVM.write("{}".format(X[i].value[l,j]))
                else:
                    POVM.write("{},".format(X[i].value[l,j]))
            POVM.write("]")
            if l != len(X[i].value)-1:
                POVM.write(",")
        POVM.write("]")
        POVM.write("\n\n")
    POVM.close()

psq.pop(-1)
psq.append(1/N)

#Append the SDP and SRM documents with the success probability of N
SDP_doc.write("{}\t{}\n".format(N,np.cumsum(psdp)[-1]))
SQRTM_doc.write("{}\t{}\n".format(N,np.cumsum(psq)[-1]))
SDP_doc.close()
SQRTM_doc.close()

#Documents for the success probabilities of j given N
SDP_partial = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/Parcials/SDP_N={}".format(N),"w")
SQRT_partial = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/Parcials/SQRT_N={}".format(N),"w"
    )

SDP_partial.write("J\tSDP\n")
SQRT_partial.write("J\tSQRTM\n")

for i in range(len(psq)):
    SQRT_partial.write("{}\t{}\n".format(J_possibles[i], psq[i]))
    SDP_partial.write("{}\t{}\n".format(J_possibles[i], psdp[i]))

SQRT_partial.close()
SDP_partial.close()
```

C.2 Success Probability with SDP and SRM using the analytical expression for G

```
"""
CONTENT:
    - Weighted Gram Matrix from Formula
    - SQRT Measurement
    - SDP (Programar-lo de nuevo)
"""

import numpy as np
import math
from scipy.linalg import sqrtm
import cvxpy as cp


N = 409

#List of all the possible J values given N and the projecting n

J_possibles = []
n1 = []

if N%2 == 0:
    for i in range(int(N/2)+1):
        J_possibles.append(i)
    n1.append(int(N/2))
else:
    for i in range(int(N/2)+1):
        J_possibles.append(1/2+i)
    n1.append(int(N/2-0.5))
    n1.append(int(N/2+0.5))

psq = []
```

```python
psdp = []

#Document for the SDP and SRM results
SDP_doc = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/SDP","a")
SQRTM_doc = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/SQRTM","a")

for i in range(len(J_possibles)):

    #Create documents for G, Sqrt(G) and POVM elements
    Gram_Mat = open("/Users/marcel/Desktop/TFG/Matrices/Gram_Matrices/Gt,N={},J={}".format(N,J_possibles[i]),"w")
    Gram_Sqrt = open("/Users/marcel/Desktop/TFG/Matrices/GSQ/GSQ,N={},J={}".format(N,J_possibles[i]),"w")
    POVM = open("/Users/marcel/Desktop/TFG/Matrices/POVM_elements/POVM,N={},J={}".format(N,J_possibles[i]),"w")

    if i == 0:
        n_proj = n1
    else:
        if N%2 == 0:
            n_proj.insert(0,int(N/2)-(i))
            n_proj.append(int(N/2)+(i))
        else:
            n_proj.insert(0,int(N/2-1/2)-(i))
            n_proj.append(int(N/2+1/2)+(i))

    np.save("/Users/marcel/Desktop/TFG/Projecting_n/N={},J={}.npy".format(N,J_possibles[i]),n_proj)

    #Creation of the Gram matrix
    G = np.zeros(shape = (len(n_proj),len(n_proj)))

    c = int(N/2-J_possibles[i])

    for j in range(len(n_proj)):
        n = n_proj[j]
        for k in range(len(n_proj)):
            nprim = n_proj[k]
            if n<=nprim:
                P1 = ((2*J_possibles[i]+1))/(np.sqrt((n+1)*(N-n+1)*(nprim+1)*(N-nprim+1)))
                P2 = np.sqrt((math.comb(n,c)*math.comb(N-nprim,c))/(math.comb(nprim,c)*math.comb(N-n,c)))
                G[j][k] = P1*P2
                G[k][j] = P1*P2

    #Sqrt(G)
    GSQ = sqrtm(G)
    if J_possibles[i] == J_possibles[-1]:

        GSQ1 = GSQ
        GSQ = np.zeros((len(n_proj), len(n_proj)))
        for p in range(len(n_proj)):
            for q in range(len(n_proj)):
                GSQ[p][q] = GSQ1[p][q].real
    np.save("/Users/marcel/Desktop/TFG/Matrices/GSQ_NPY/GSQ,N={},J={}.npy".format(N,J_possibles[i]),GSQ)

    #Export G and Sqrt(G)
    Gram_Mat.write("[")
    for i in range(len(n_proj)):
        Gram_Mat.write("[")
        for j in range(len(n_proj)):
            if j == len(n_proj)-1:
                Gram_Mat.write("{}".format(G[i,j]))
            else:
                Gram_Mat.write("{},".format(G[i,j]))
        Gram_Mat.write("]")
        if i != len(n_proj)-1:
            Gram_Mat.write(",\n")
    Gram_Mat.write("]")
    Gram_Mat.close()

    Gram_Sqrt.write("[")
    for i in range(len(n_proj)):
        Gram_Sqrt.write("[")
        for j in range(len(n_proj)):
            if j == len(n_proj)-1:
                Gram_Sqrt.write("{}".format(GSQ[i,j]))
            else:
                Gram_Sqrt.write("{},".format(GSQ[i,j]))
        Gram_Sqrt.write("]")
        if i != len(n_proj)-1:
            Gram_Sqrt.write(",\n")
    Gram_Sqrt.write("]")
    Gram_Sqrt.close()

    #SRM
    Sqrt_Measurement = 0

    for i in range(len(n_proj)):
        Sqrt_Measurement += np.round(GSQ[i,i]**(2),6)

    psq.append((1/N)*Sqrt_Measurement)

    omega_tilda = []
    for i in range(len(n_proj)):
        omega_tilda.append(GSQ[:,i])

    #Definition of variables and constraints of SDP
    X = []
    for i in range(len(omega_tilda)):
        X.append(cp.Variable((len(omega_tilda[0]),len(omega_tilda[0])),PSD=True))

    constraints = [sum(X) == np.identity(len(omega_tilda[0]))]

    #Define the problem and Solve it
    P = 0
    for i in range(len(omega_tilda)):
        P += (1/N)*omega_tilda[i]@ X[i]@ omega_tilda[i]

    Prob = cp.Problem(cp.Maximize(P),constraints)
    Prob.solve()

    psdp.append(Prob.value)

    #Export the POVM elements in matrix form
    for i in range(len(X)):
        POVM.write("[")
        for l in range(len(X[i].value)):
            POVM.write("[")
            for j in range(len(X[i].value)):
                if j == len(X[i].value)-1:
                    POVM.write("{}".format(X[i].value[l,j]))
                else:
                    POVM.write("{},".format(X[i].value[l,j]))
            POVM.write("]")
            if l != len(X[i].value)-1:
                POVM.write(",")
        POVM.write("]")
        POVM.write("\n\n")
    POVM.close()


psq.pop(-1)
psq.append(1/N)

#Documents for the success probabilities of j given N
SDP_partial = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/Parcials/SDP_N={}".format(N),"w")
SQRT_partial = open("/Users/marcel/Desktop/TFG/Ps_SDP_SQRTM/Parcials/SQRT_N={}".format(N),"w"
    )

SDP_partial.write("J\tSDP\n")
SQRT_partial.write("J\tSQRTM\n")

for i in range(len(psq)):
    SQRT_partial.write("{}\t{}\n".format(J_possibles[i], psq[i]))
    SDP_partial.write("{}\t{}\n".format(J_possibles[i], psdp[i]))

SQRT_partial.close()
SDP_partial.close()
```

C.3 Hamming Distance Expected Value for SRM

```
"""
Hamming Distance expected value SRM:
    - Creates a document with the Expected Value for each J
    - Document with the total Expected value for each N
"""

import numpy as np
import cvxpy as cp

N = 40

#List of all the possible J values given N
J_possibles = []

if N%2 == 0:
    for i in range(int(N/2)+1):
        J_possibles.append(i)
else:
    for i in range(int(N/2)+1):
        J_possibles.append(1/2+i)

#Document where <h> will be exported for each N
g = open("/Users/marcel/Desktop/TFG/HammingDistance_Optimization/ExpectedValueSRM","a")

Ham_Opt = 0

for J in J_possibles:
    #Load Sqrt(G) and the corresponding n array of projections
    GSQ = np.load("/Users/marcel/Desktop/TFG/Matrices/GSQ_NPY/GSQ,N={},J={}.npy".format(N,J))
    n_proj = np.load("/Users/marcel/Desktop/TFG/Projecting_n/N={},J={}.npy".format(N,J))

    #Perform <h>
    for i in range(len(n_proj)):
        for j in range(len(n_proj)):
            Ham_Opt += (1/N)*abs(i-j)*GSQ[i][j]**2

g.write("{}\t{}\n".format(N,Ham_Opt))
g.close()
```

C.4 Hamming Distance Optimization with SDP

```python
"""
Hamming Distance Optimization:
    - Creates a document with the Expected Optimized Value of <h> for each J
    - Document with the total optimized expected value <h> for each N
"""

import numpy as np
import cvxpy as cp

N = 7

#List of all the possible J values given N
J_possibles = []

if N%2 == 0:
    for i in range(int(N/2)+1):
        J_possibles.append(i)
else:
    for i in range(int(N/2)+1):
        J_possibles.append(1/2+i)


Ham_Opt = 0

#Create document for <h> for each j given N and another document for <h> given N
f = open("/Users/marcel/Desktop/TFG/HammingDistance_Optimization/Partials/HammingOpt_N,N={}".
    format(N),"a")
g = open("/Users/marcel/Desktop/TFG/HammingDistance_Optimization/ExpectedValue_N","a")

f.write("J\tHOpt/N\n")

for J in J_possibles:

    #Load Sqrt(G) and the corresponding n array of projections
    GSQ = np.load("/Users/marcel/Desktop/TFG/Matrices/GSQ_NPY/GSQ,N={},J={}.npy".format(N,J))
    n_proj = np.load("/Users/marcel/Desktop/TFG/Projecting_n/N={},J={}.npy".format(N,J))

    #Get the columns of Sqrt(G) in order to create density matrices |a><a|
    GSQ_col = []

    for i in range(len(GSQ[0])):
        GSQ_col.append(GSQ[:][i])

    #Define Variables and Constrains of SDP
    X = []
    for i in range(len(GSQ_col)):
        X.append(cp.Variable((len(GSQ_col[0]),len(GSQ_col[0])),PSD=True))

    constraints = [sum(X) == np.identity(len(GSQ_col[0]))]

    #Define and Solve the Problem
    P = 0
    for i in range(len(n_proj)):
        for j in range(len(n_proj)):
            P += (1/N)*abs(i-j)*(GSQ_col[i]@X[j]@GSQ_col[i])

    Prob = cp.Problem(cp.Minimize(P),constraints)
    Prob.solve()

    Ham_Opt += Prob.value

    #Export <h> for each j
    f.write("{}\t{}\n".format(J,Prob.value))

#Export <h> for N
g.write("{}\t{}\n".format(N,Ham_Opt))
```

```

f.close()
g.close()
```

C.5 Padé Approximants for P_s

```
In[71]:= coeffs = {0, 0, -1/30, -1/35, -229/10080, -101/5544, -5725/384384, -5111/411840, -554293/52715520,
           -2137221/236487680, -249919231/31783944192, -76576105/11076222976, -10870862389/1772195676160,
           -8413125001/1533630873600, -1506595197973/304973453721600};

         expr = 2 x^2;
         For[i = 1, i ≤ Length[coeffs], i++, expr += coeffs⟦i⟧ x^(2 i)];
         para cada       longitud
         Padeaprox = PadeApproximant[expr, {x, 0, {8, 8}}];
                     aproximación de Pade
         Pj = Integrate[N[Padeaprox], {x, 0, 1}]
              integra   valor numérico
         N[Padeaprox]
         valor numérico
```

C.6 Projections on the Coupled Angular Momentum Basis using Wigner Matrices

```
Ntot = 20;
n = 7;
nb = Ntot - n;
angle = 180°;
P1 = {n/2, n/2};
P2 = {nb/2, nb/2};
P2Rot = {};
For[mp = -nb/2, mp ≤ nb/2, mp++, AppendTo[P2Rot, {WignerD[{P2⟦1⟧, P2⟦1⟧, mp}, angle], P2⟦1⟧, mp}]];
para cada                        añade al final    D de Wigner
Coupled = {{"Coeff", "J", "M"}};
For[i = 1, i ≤ Length[P2Rot], i++,
para cada      longitud
  For[j = Abs[(n - nb)/2], j ≤ (n + nb)/2, j++,
  para cada valor absoluto
   If[ClebschGordan[{n/2, n/2}, {P2Rot⟦i, 2⟧, P2Rot⟦i, 3⟧}, {j, n/2 + P2Rot⟦i, 3⟧}] ≠ 0,
   si  Clebsch Gordan
    AppendTo[Coupled, {P2Rot⟦i, 1⟧ * ClebschGordan[{n/2, n/2}, {P2Rot⟦i, 2⟧, P2Rot⟦i, 3⟧}, {j, n/2 + P2Rot⟦i, 3⟧}], j, n/2 + P2Rot⟦i, 3⟧}], Continue]]];
    añade al final                  Clebsch Gordan                                                                                      continúa iteración
PJ = {};
For[j = Ntot/2, j ≥ 0, j = j - 1,
para cada
  Pj = 0;
  For[i = 2, i ≤ Length[Coupled], i++, If[j == Coupled⟦i, 2⟧, Pj = Pj + Coupled⟦i, 1⟧^2, Continue]]
  para cada     longitud                si                                                  continúa iteración
   AppendTo[PJ, Pj]];
   añade al final
```