Are the Calculations of Vibrational Spectra of Molecules a Good Candidate for Practical Quantum Advantage?

Thomas Burton

4th Year Project Report
Computer Science and Physics
School of Informatics
University of Edinburgh

2023

Abstract

Quantum computers promise to solve certain problems which are classically hard to compute. One such problem is the calculation of vibrational spectra, a problem which appears to take exponential time on classical computers. The best classical algorithms for this problem truncate the computation, providing a meaningful speed up. However, their full computational scaling is not entirely understood for physical systems, and the assumptions involved may be optimisable in such a way as to challenge the practical usefulness of any quantum advantage claimed for this problem.

In this project I investigate the algorithm of Santoro et al. [34] for computing vibrational spectra, using the Strawberry Fields [23] library for Python to implement the calculations involved. In particular, I investigate the effects of altering chemical parameters on the viability and computation time of the algorithm and demonstrate that by exploiting the block diagonality of Duschinsky matrices computation time can be greatly reduced.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Thomas Burton*)

Acknowledgements

I would like to thank my supervisor, Raúl, for proposing a project that I thoroughly enjoyed on a topic I am greatly interested in. Our discussions were always thought-provoking and encouraging, and this project would not have been possible without his feedback.

I would also like to thank Joonsuk Huh from Sungkyunkwan University for providing me with the pre-processed molecular data for thymine. Without this, many hours would have been spent trying to work out how to draw the data from other available resources.

Finally, I would like to thank my parents, Lesley and Simon, for providing me with support for the last 4 years in all areas and always asking after my studies; my sister, Purdey, for keeping me fuelled with her baked goods; and Alexa, for motivating me to keep going even when things felt unachievable.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Quantum computation is a rapidly developing area of research, with improvements being made and proofs of quantum advantage/supremacy becoming more frequent. For example, in 2019 research led by engineers at Google claimed their Sycamore quantum computer could sample a 53 qubit system one million times in 200 seconds in order to produce a probability distribution of a random number generator, and that it would take an equivalent classical algorithm 10,000 years [4] to achieve the same results. However, in 2022 Zhang et al. created a new classical algorithm that was able to outperform its quantum counterpart [36]. The field is full of presentations of new algorithms that push the boundaries further on both sides.

The key goal of the field is to demonstrate quantum advantage, or that problems which are intractable on classical machines are indeed solvable in polynomial time on quantum machines [9]. Whilst this has been well studied theoretically in a number of areas, practical quantum advantage, or an experimental implementation that successfully demonstrates quantum advantage, has yet to be shown. Some algorithms, such as Shor's algorithm for factorising large numbers [35], have been shown to be computable on quantum systems, and in some cases the potential for scalability has been discussed as well, for example in the work of Monz et al. [28]. However there has yet to be an experimental implementation of such an algorithm that has been proven to outperform classical machines for the same task.

Quantum computers themselves can be broadly split into two categories: continuous variable (CV) systems and discrete systems. Discrete systems are conceptually closer to classical systems: one can consider qubits analogous to classical bits (with the key difference being that qubits can be in a superposition of states). CV systems, however, revolve around operators which have a continuous spectra acting on qumodes. It is possible to embed discrete systems into CV systems [41], meaning the two are equally computationally powerful. Particular CV systems seeing current success are photonic devices [25] [47], which involve the manipulation and measurement of light to perform calculations.

Photonic devices can generally be thought of as being split into three parts: n single photon sources, a circuit of beam splitters and phase shifters, and photon detectors [24]. The potential success of photonic devices began to be recognised in 1998 with the proof of unconditional quantum teleportation [16]. This is the proof that it is possible to transfer an arbitrary unknown quantum state over any distance [45]. If information is encoded into these states, this allows for instantaneous long-range communication that cannot be achieved classically.

The natural qualities of light makes it well suited for use in, for example, communication. The idea of transmitting a wave to encode information is already well accepted and understood, being the basis behind internet communication (and more) [2]. It therefore makes sense that a promising avenue for quantum advantage is photonic devices. They have been studied extensively over the last two decades, for example as noisy intermediate scale quantum (NISQ) devices. These are devices capable of outperforming classical machines but only on very small scales. At larger scales, too much noise appears in the system for the result to be unusable. Despite this limitation, tests have been created to verify the usefulness of photonic devices [27].

The algorithm that appears to have the most potential for quantum advantage currently is Gaussian boson sampling (GBS). Boson sampling involves using detectors to measure the entangled photon number and path states from non-classical light injected into a linear optical detector by simultaneously emitting single photon sources [1]. GBS is an extension of this, for which the input states are squeezed states instead [18]. This sampling problem can involve either counting single photons (i.e. observing whether a photon is present or not in an output mode) or counting the number of photons at each mode. The photon counting problem is significant and was chosen for quantum advantage as it has been proven to be linked to the calculation of the hafnian of a matrix (which is linked to the permanent) [32], which appears to be classicaly computationally hard.

Zhong et al. [47] recently developed a 50 spatial mode interferometer and used it to generate as many samples in 200 seconds as the Fugaku supercomputer [15] could generate in 0.6 billion years, before expanding their machine to 144 modes only a year later [46]. Whilst significant, the work of Zhong et al. was incredibly specialised, requiring a highly technical setup that could not easily be altered or modified to apply their results to other problems. Thus, the work of Zhong et al. did not fully demonstrate practical quantum advantage.

Since the work of Zhong et al., a team working at Xanadu, Canada, developed a system based on optical fibre loops that went further. Their system allowed for full programmability of quantum gates [25], meaning it could (theoretically) implement any circuit that could be designed. Again using GBS, the team produced a result in 36μs that would take the best classical algorithms approximately 9000 years. This suggests GBS is a serious avenue towards quantum advantage, but again did not fully show a practical quantum advantage.

The proposal of Boson sampling (and its Gaussian extension) was originally designed not to create a universal quantum computer, but rather demonstrate advantage in one specific way [1]. Whilst it may seem that this would not demonstrate practical quantum

advantage, Huh et al. [20] demonstrated that GBS can be set up in such a way as to be equivalent to the calculation of the vibrational spectrum of a molecule. This is a problem that is classically hard to solve and has many applications in a broad range of sciences [14], so creating a machine that can solve it in an exponentially sped up amount of time would be groundbreaking.

The classical method for computing the vibrational spectrum of a molecule involves computing transition probabilities from the starting state to all possible combinations of states at all possible excitation values/quantum numbers. The most naïve method therefore has a scaling which is exponential in the number of modes. However, there are a number of ways that the computation can be truncated and due to their wide scientific importance, many classical algorithms exist that aim to reduce the computation time of vibrational spectra.

One such algorithm is that of Santoro et al. [34], which truncates the computation based on the number of excited modes of the system as well as a maximum quantum number. The algorithm will first look at the case of 0 excited modes, then individual excited modes, then pairs etc. until a desired precision in the spectra is reached. This algorithm has been shown to perform well for a number of real molecules, so any claim of practical quantum advantage would have to outperform it. Moreover, the algorithm has not been fully explored in all edge cases where different performances may arise, nor has it been fully investigated in conjunction with other speed-ups, hence suggesting classical algorithms may well have the potential to be considerably improved and hence making the claims of practical quantum advantage harder to prove.

1.2 Aims

This work will focus on analysing the algorithm of Santoro et al. [34], and exploring the ability of vibrational spectra calculations to demonstrate quantum advantage. Their algorithm works well for practical situations, making use of exploitable aspects of the calculations involved to solve many-mode problems to a high degree of accuracy. However, they have not considered the asymptotic run cases as that fell beyond the scope of their research. Here, I develop an implementation of their algorithm and analyse its scaling to see the regimes in which it computes in better than exponential time. I use the Strawberry Fields Python library [23] developed by Xanadu to simulate GBS and hence investigate if using GBS to calculate vibrational spectra is a potential avenue for quantum advantage.

1.3 Report Structure

Chapter 2 describes a broad overview of some physical concepts and terminologies that may be unfamiliar to the reader. Chapter 3 describes the algorithm of Santoro et al., including some of the physical intuitions behind it and linking it to how I implemented it as described in chapter 4. Chapter 5 presents the results obtained from experiments involving my implementation of the algorithm. Chapter 6 summarises the findings of this project, and discusses areas in which ideas discussed here could be taken further.

1.4 Contributions

In the process of completing this project, I achieved the following:

- Gained a thorough understanding of how Gaussian boson sampling works, and more generally the basic ideas behind quantum photonics.
- Deepened my understanding of what causes vibrational spectra, and learned the basics behind Duschinsky mixing and how this effects the spectra produced.
- Became thoroughly familiar with the Strawberry Fields library, how to implement the functions contained within and how to navigate their documentation to find required information.
- Developed an implementation of the algorithm of Santoro et al., and learned its strengths and limitations through experiments I designed and conducted upon it.
- Investigated the effectiveness of exploiting block-diagonality of Duschinsky matrices to speed up computations, and framed this from the quantum computation perspective.
- Investigated the effectiveness of the algorithm of Santoro et al. on more complex systems, and thus concluded that the claims of practical quantum advantage for vibrational spectra are not clear-cut.

The main files used to perform the experiments are available at this github repository.

Chapter 2

Technical Background

This chapter shall present some technical (physical) background required for the understanding of this project, elaborating further on ideas introduced in chapter 1, and also discuss some state of the art implementations and algorithms in use currently. Readers more experienced with quantum physics may find they can skip over some sections.

2.1 Molecular Excitations

When molecules gain energy they are said to be excited, and when excited they vibrate. Molecules vibrate in specific ways and at specific frequenices known as vibrational modes. A non-linear molecule made of N atoms contains $3N-6$ vibrational modes. These modes arise from each atom being able to move in three dimensions, hence giving it 3N modes, with 6 removed to account for the centre of mass and momentum which can be set to 0 via a change of reference frame. Equivalently, the 6 removed modes are composed of 3 rotations and 3 translations and are therefore not vibrational modes. A linear molecule has $3N-5$ vibrational modes, as it only has 2 rotational degrees of freedom (a rotation about its own axis leaves it unchanged) [38]. A clear visualisation of these modes for formic acid, which has 9 vibrational modes, is available at the following website: https://www.chem.purdue.edu/jmol/vibs/facid.html [29].

A molecule can be excited by receiving energy in the form of light. Light is quantised, meaning this energy can only be delivered in certain steps, and one quanta of light is called a photon. Similarly, modes can only be excited to certain levels, meaning only photons with the correct amount of energy can be absorbed by modes. The different excited states of molecules are sometimes called energy levels.

2.2 Dirac Notation and Quantum Mechanics

Dirac notation was developed by Paul Dirac in the 20th century to simplify the complex mathematics that came with quantum mechanics. Fundamentally, it consists of two key symbols: the bra: $\langle A|$ and ket: $|A\rangle$. The simplest way to think of these is kets as vectors and bras as the complex conjugate of the transpose of a ket, $\langle A| = |A\rangle^{\dagger}$. The product of

a bra and ket is the inner product and is a scalar, written as $\langle A|B\rangle$. The product of a ket and bra is the outer product and equal to a matrix $|B\rangle\langle A|$.

In quantum mechanics, Dirac notation is used to represent quantum states of the form:

$$|\psi\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} \tag{2.1}$$

where the state has n degrees of freedom and the probability amplitude of degree i is given by $p(i) = |\psi_i|^2$.

When an operator acts on a state the state is transformed. An eigenvector of an operator is a state on which the action of an operator leaves the vector unchanged, as below:

$$A|\psi\rangle = a|\psi\rangle \tag{2.2}$$

where A is the operator with eigenvector $|\psi\rangle$ and eigenvalue a. In quantum computers, computations are generally represented as circuits, which consist of a series of gates (operators) acting on collections of states. Collections of N states are typically represented using the notation $|\psi_1, \psi_2 ... \psi_N\rangle$, which is the tensor product of states ψ_1 to ψ_N.

2.3 Harmonic Oscillators

A harmonic oscillator is any object which undergoes periodic motion [30]. It turns out that many concepts in physics, from pendula to hydrogen atoms, can be accurately modelled as a simple harmonic oscillator. The Hamiltonian (the mathematical operator that defines the total energy of a system) of a simple harmonic oscillator with mass m and oscillating at frequency ω is:

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}M\omega^2\hat{x}^2 \tag{2.3}$$

Where $\hat{p} = -i\hbar\nabla$ is the momentum operator and $\hat{x}$ is the position operator. The energy eigenstates of such a system are known as Fock states, and are given by the equation:

$$\psi_n(x) = \langle x|\psi_n\rangle = \frac{1}{\sqrt{2^n n!}}\left(\frac{m\omega}{\pi\hbar}\right)^{1/2} e^{\frac{-m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right) \tag{2.4}$$

Where H_n are the Hermite polynomials. $\psi_n(x)$ is the wavefunction in the position basis and hence describes the probability amplitude of observing the particle in position n (equivalent to at energy level n). The Fock states can be interpreted as the presence of n quanta of light in the corresponding vibrational mode.

Equivalently, one can define the creation operator $a^\dagger$ as:

$$a^\dagger = \sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} - \frac{i}{m\omega}\hat{p}\right) \tag{2.5}$$

The operator † represents the Hermitian conjugate, and is defined as $\langle A^\dagger \phi | \psi \rangle = \langle \phi | A \psi \rangle$. This is equivalent to taking the complex conjugate and transpose of the matrix representing the operator A. This allows the Fock states to be written as:

$$|n\rangle = \frac{(a^\dagger)^n}{\sqrt{n!}} |0\rangle \tag{2.6}$$

The number n is also known as the quantum number of the mode, and hence the excitation of a mode can be defined by its quantum number. The significance of the creation operator (and it's corresponding annihilation operator a) is to represent the addition or subtraction of individual quanta of energy to or from the system. This leads to the relation:

$$a^\dagger |n\rangle = \sqrt{n+1}\,|n+1\rangle \tag{2.7}$$

In the cases considered in this report, the addition or subtraction of a quanta from a state $|n\rangle$ is to add or remove one photon from that mode.

2.4 Spectroscopy and Frank Condon Profiles

Spectroscopy is the study of the emission of light by matter. By analysing the observed spectra created by certain materials, the molecules contained within can be identified [14]. An example of one such spectra is shown in Figure 2.1.

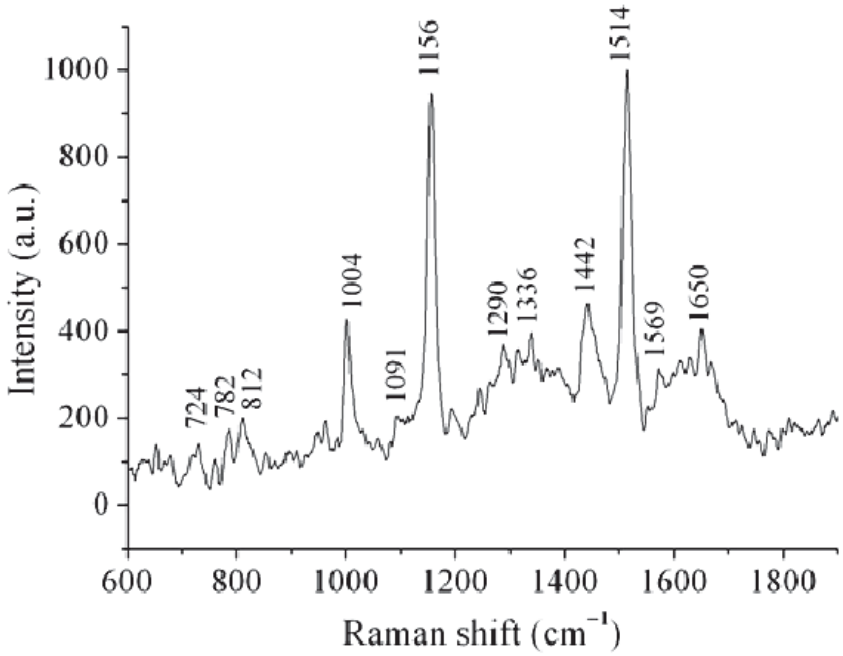


Figure 2.1: An example of the spectra produced when investigating the bacteria Planococcus sp.NJ41. The spectra was produced when a 20mW laser shone light with a wavelength of approximately 785nm at the bacteria. The units cm^{-1} describe the wave number which identifies the light produced. Figure reproduced from [40].

Spectroscopy is commonly employed to identify unknown substances within a target, and hence is highly important experimentally, with applications from biomedicine [12] to planetary exploration [5]. However, in order to match observed spectra to known spectra, the spectra must first be known. As such, a method of computing the spectra for a given molecule is a highly useful tool.

2.4.1 Mathematical Representation of Molecules

The different energy levels of vibrational modes can be modelled as simple harmonic oscillators [22]. Thus, the movement from one state to another is the movement from one Hamiltonian corresponding to the electronic ground state to the Hamiltonian of an excited state. This can be described by a linear transform from one set of coordinates to another, as first proposed by Duschinsky [13]. Specifically, the transformation is

$$\boldsymbol{q'} = U_d \boldsymbol{q} + \boldsymbol{d} \tag{2.8}$$

where $\boldsymbol{q'}$ and $\boldsymbol{q}$ are the final and initial coordinates, $\boldsymbol{d}$ a displacement vector and U_d the orthogonal Duschinsky matrix for that molecule.

Duschinsky matrices are unique for molecules (examples for molecules considered in this project are in appendix B) and the composition of the matrices reveals information about the transition. Crucially, Duschinksy matrices that are more sparse represent transitions with less 'Duschinsky mixing', meaning the excitations of individual modes are more significant than the correlation between those excited modes.

2.4.2 Mathematics of Vibrational Transitions

From the coordinate translation relationship (2.8), Doktorov [11] demonstrated the transformation from one energy to another can be described by the application of one operator, called the Doktorov operator:

$$U_{Dok} = DS'^{\dagger}RS \tag{2.9}$$

where R, S and D are rotation, squeeze and coherent displacement operators respectively. The full composition of these three operators is not required for this project, but they relate to the chemical properties of the molecule as such: R comes from the Duschinsky matrix, S and S' from the initial and final frequencies respectively, and D from the displacement vector. Thus, the transition probability to a specific mode from one mode to another, known as the Frank Condon factor (FCF) [20], is given by:

$$|\langle \boldsymbol{m}| \hat{U}_{dok} |\boldsymbol{n}\rangle|^2 \tag{2.10}$$

where $\langle \boldsymbol{m}|$ and $|\boldsymbol{n}\rangle$ are the states $\langle m_1, m_2 ... m_N|$ and $|n_1, n_2 ... n_N\rangle$ of a system with N vibrational modes, and each $\langle m_i|$ represents a Fock state/the number of photons in the ith mode. By sampling the FCFs, a probability distribution called the Frank Condon Profile (FCP) is generated for each frequency, defined for a transition from the ground state (with zero vibrations at zero temperature) as:

$$FCP(\omega) = \sum_{\boldsymbol{m}}^{\inf} (|\langle \boldsymbol{m}| \hat{U}_{dok} |\boldsymbol{0}\rangle|^2) \delta(\omega - \sum_{k}^{N} \omega'_k m_k) \tag{2.11}$$

where ω_k and ω'_k are the initial and final frequencies of mode k respectively.

The spectra computed using such a method is the ideal spectra. However, there is often noise present in the spectra produced experimentally, for example due to the Doppler effect [1]. Therefore, in order to account for any noise, once the FCP has been computed it may then be convoluted with a Gaussian or Lorentzian function [34] over specific energies to generate a vibrational spectrum that is closer to experimentally produced spectra.

2.4.3 Limitations and Exploits

Despite their usefulness, the calculations of vibrational spectra is directly linked to the Hafnian of a matrix and hence is theoretically a #P-complete problem (discussed further in sections 2.5 and 2.6). For a system of n vibrational modes, there are $O(2^n)$ combinations of excited modes and for each of these the modes can be excited up to different values (theoretically to infinity), meaning there are $\Omega(2^n)$ FCFs to compute. In practice, the computation time can be reduced substantially as true molecules do not (for example) have spectra that depend highly on the contribution from modes with very high excitations. However, there does not exist a universal polynomial time algorithm for all molecular spectra.

Algorithms have been developed to attempt to speed up such calculations by making use of these experimental observations. One such algorithm is that of Santoro et al. [34], which was designed to work on more complex systems such as molecules in solution. This algorithm is discussed in greater detail in chapter 3, but in short exploits the fact that the majority of states in which more than a few modes of the system are excited contribute very little to the overall spectra. Moreover, the authors exploit the fact that the contribution to the spectra of excitations which are at higher energy levels will decrease. The combination of these two observations allow calculations to be truncated considerably earlier than the most naïve methods would, hence greatly decreasing computation time. They compared the output of their algorithm to known results and found that, for example, their computation of the spectra of anthracene (a molecule with 66 vibrational modes) computed 87% of the total spectra in only 26 seconds.

2.5 Gaussian Boson Sampling

Boson sampling is a quantum linear optics scheme which involves using a linear interferometer to cause the mixing of a series of single-photon emitting sources [1]. The output is then the probability of observing a single photon at each detector, and this probability distribution can be sample to give interesting results [42]. In the terminology used in section 2.4, the output state is $|n_1, n_2 ... n_N\rangle$ for an input of $|m_1, m_2 ... m_N\rangle$. However, this is experimentally difficult to scale as the simultaneous emission of single photons generally relies on a non-deterministic method of photon production called spontaneous

[1] The Doppler effect is the apparent shift in wavelength of light produced by a source that is moving relative to the observer. As spectroscopy involves the observation of moving (oscillating) particles, the spectra produced will be broadened by this effect.

parametric down conversion [8]. This means that as the size of the system increases the average time for the emission of N single photons also increases.

Mathematically, the output of a boson sampling circuit is related to the permanent of the matrix describing the interferometer:

$$\langle n_1, n_2 ... n_N | \psi' \rangle = \frac{|Per(U)|^2}{\prod_{i=1}^{N} m_i! n_i!} \tag{2.12}$$

where $|\psi'\rangle = W|\psi\rangle$ is the output state when W, a homomorphism of U, is applied to the input states $|n_1, n_2 ... n_N\rangle$.

Gaussian Boson sampling is an extension of boson sampling that does not require the input to be single-photon Fock states. Hamilton et al. [18] demonstrated that Gaussian states (states whose characteristic functions are Gaussian in phase-space [33]) can be used as the inputs instead of the single-photon Fock states without loss of generality. This is significant because single-mode squeezed states, which are Gaussian states, can be deterministically generated by applying the same squeezing gate to N input vacuum states simultaneously [43]. A circuit diagram of a GBS circuit is shown in figure 2.2.

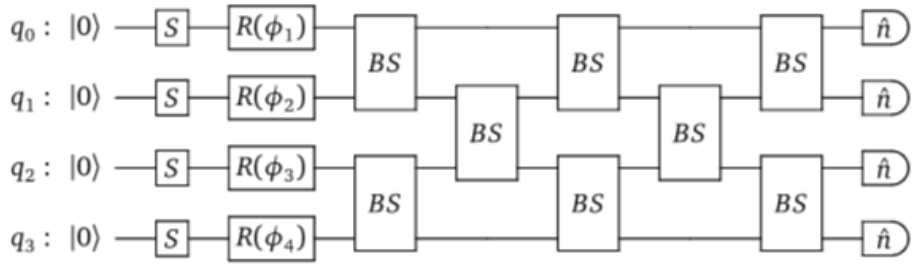


Figure 2.2: A circuit diagram showing Gaussian boson sampling. Without the initial 4 squeeze gates the circuit would be equivalent to boson sampling. From left to right the gates are: squeezing gates, S; rotation gates parameterised by some ϕ, $R(\phi_n)$; beamsplitters BS which together represent the unitary acting on the system; photon counters $\hat{n}$. Reproduced from [43].

Mathematically, the output state of a Gaussian boson sampling is related to the Hafnian of the matrix describing the interferometer:

$$\langle n_1, n_2 ... n_N | \psi' \rangle = \frac{|Haf(UU^T \tanh(r))_s|^2}{(\prod_{i=1}^{N} n_i!) \cosh^N(r)} \tag{2.13}$$

where r is the squeezing parameter applied to all input states and ψ' is again the output state. It should be noted that the hafnian calculates the number of perfect matchings in a graph, meaning its computation is #P-complete. The full description of the calculation of a hafnian is not required for this project, but broadly speaking the size of the matrix considered increases with the number of photons observed in each mode. Specifically, for the hafnian $Haf(U)_s$ of a matrix U, each row and column corresponding to a mode is deleted if the number of photons in that mode is 0, or repeated m times if the it contains $m > 0$ photons. As with boson sampling, by sampling the output distribution of a GBS circuit, interesting results can be obtained, for example sampling dense subgraphs [3] and vibrational spectra, as discussed below.

2.6 Mapping Spectra to Gaussian Boson Sampling

2.6.1 Implementation

Using the tools described in sections 2.1 - 2.5, Huh et al. showed that the problem of computing vibrational spectra can be mapped directly to Gaussian boson sampling [20]. By modelling the vibrational modes of a molecule as simple harmonic oscillators, the Frank Condon profile can be sampled to calculate the vibrational spectra using equation 2.10, which depends on the Doktorov operator $\hat{U}_{dok}$. As shown in equation 2.9, the Doktorov operator can be decomposed into displacement, squeezing and rotation operators, all of which together can form part of a quantum GBS circuit.

If the initial temperature is $0K$, i.e. the transitions are occurring from the ground state, the inputs are all vacuum state modes (equal to $|0\rangle$). Therefore, by preparing a set of N squeezed vacuum states and inputting them to an interferometer equivalent to $DS^{\dagger}R$ and then sampling the output, the vibrational spectra of a molecule can be obtained. This equivalence is shown in figure 2.3. In this figure, it is clear that both problems involve shining light on a system and sampling the light that is subsequently re-emitted.

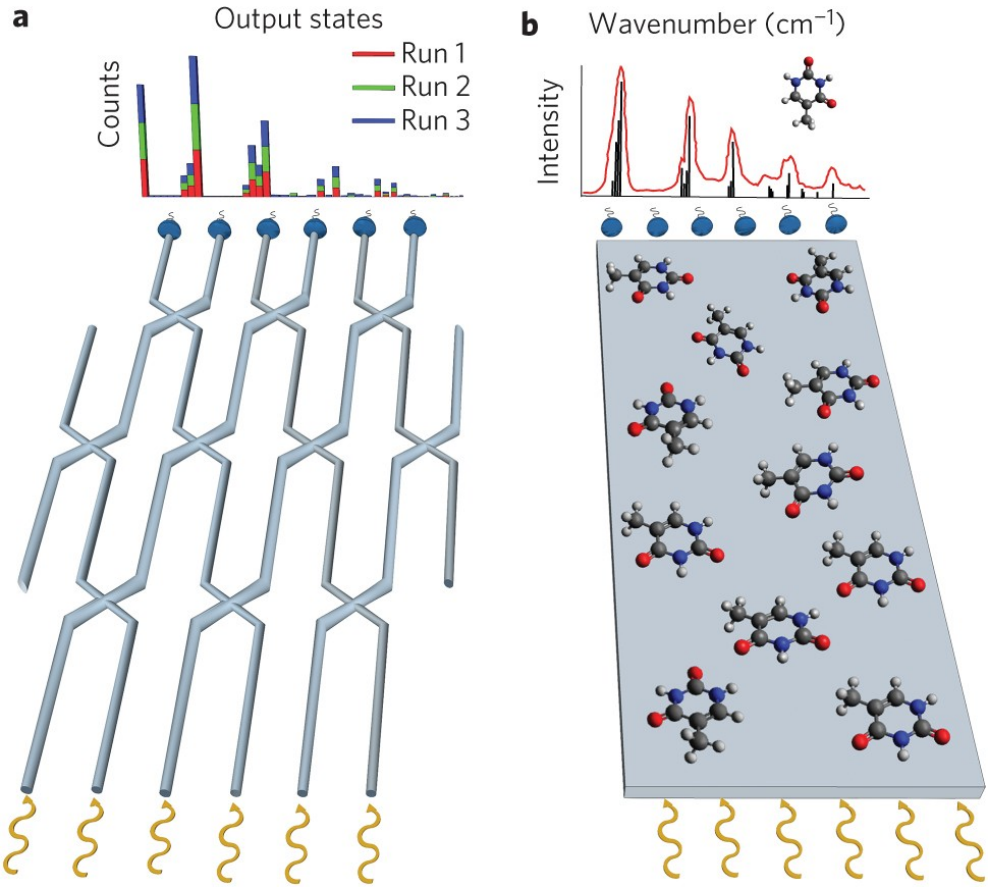


Figure 2.3: **a** a representation of sampling a GBS circuit, with 6 modes of light interferring wiht each other in an interferometer before sampling occurs. **b** A visualisation of the emission of vibrational spectra, with 6 modes of light incident on a molecule which then re-emits light in the form of a spectra. Reproduced from [20].

A subtle but important distinction must be noted here between classical and quantum computers. In classical systems, computers tend to be generalised and capable of computing a variety of different algorithms using the same basic underlying hardware and processes. The GBS system, however, is a sampling problem that, in this case, is designed in such a way to have its output probabilities match the FCP of specified molecules.

There exist various forms of the Doktorov operator, all of which are equivalent. For example, for molecules investigated at temperatures greater than $0K$, a modified experimental apparatus is used which applies a displacement operator first rather than last. However, this project focuses on simulations rather than implementations on actual quantum devices and hence the exact form of the Doktorov operator is not important.

2.6.2 Experimental and Theoretical Limitations

The work of Huh et al. undoubtedly provides a potential avenue for quantum advantage; they have shown that the computation of FC profiles can be reduced to a problem of GBS consistently. However, there are some limitations to their work.

Firstly, their work involved simulations focusing on molecules which require at most three photons per mode in the FC factors. This number was chosen due to technical limitations of the ability of photon detectors to distinguish more than 3 photons [6]. Whilst this allowed their work to be shown to be experimentally implementable, for larger/more complex systems the experimental apparatus must be able to distinguish enough photons at a greater number of modes. They do, however, show through further simulations that their methodology can work for larger molecules.

Moreover, whilst it is possible to create artificial problems which are incredibly computationally complex to compute, most real spectra seem to be sparse and have structure. This raises the question of whether approximations such as that developed by Santoro et al. [34] are actually in need of replacement, or whether such approximations are good enough for this problem. If the classical approximations are good enough on their own, the advantage proposed may be purely theoretical and practically unimportant. This distinction between theoretical and practical quantum advantage is hugely relevant as achieving theoretical proof of quantum advantage would not have an immediate effect on the scientific community [31], but achieving practical quantum advantage would have far reaching consequences.

2.6.3 Current Implementations

As mentioned previously, there are experimental challenges involved with creating a quantum device, such as keeping the amount of noise in the system low [27], or controlling a high number of photons [6]. However, progress is being made and since the publication of the work by Huh et al. a quantum device has been developed that can calculate vibrational spectra for two mode systems with up to 15 photons per mode [39]. This system relies on the principle of superconducting, which is when materials lose all electrical resistance at low temperatures, to manipulate the modes of microwave cavities. This introduces another limitation (keeping the system at sufficiently low

temperatures), and demonstrates the technical challenges that can arise depending on the implementation chosen for a quantum device. However, whilst this is only a small number of modes, it suggests advantages in the field are coming closer and closer to experimentally demonstrating practical quantum advantage.

Another subsequent implementation focusing on a two mode system was that of Clements et al. [7] in 2018. This was an optical implementation of GBS building upon the work of Huh et al. [20]. Whilst the size was again small and hence not of great practical significance in itself, the key results of the paper was to demonstrate the sources of errors in computations and how they can be reliably accounted for. This would have significant implications for larger-scale systems as no apparatus could be reliably scaled up if errors could not be removed or accounted for.

Chapter 3

The Algorithm of Santoro et al.

This chapter presents a summary of the algorithm of Santoro et al. [34] and presents some of the physical intuition behind the choices made within it.

3.1 The Algorithm

In their original paper, Santoro et al. describe the steps of their computation for the FCP of a molecule with N vibrational modes in solution. The key idea behind their methodology is not to compute the whole FCP, but rather split the FCP into a set of N Frank Condon classes (FCCs), denoted C_n, where each C_n contains n excited modes.

The first class, C_0 is the class for which no modes are excited (equivalent to all being excited to $w_k = 0$) and hence contains only a single value which can be computed analytically. Specifically, C_0 is a Gaussian integral representing the overlap of two Gaussian states.

C_1 is then computed, consisting of the states which only have a single excited oscillator. Each of these oscillators can be excited up to some maximum quantum number $w_k >= 1$ beyond which the state's contribution to the spectra (i.e. FCF) is less than some value that defines negligibility. These FCFs are computed iteratively, moving from one excited mode to the next when the quantum number of that mode leads to an FCF with a negligible contribution.

C_2 is then computed. This class consists of pairs of excited oscillators. As in C_1 these pairs are looked at iteratively, and once the contribution becomes negligible the next pair is examined.

This description of the earlier classes leads to the intuitive observation that each class C_n contains $\binom{N}{n}$ different combinations of excited modes, and each of these excitations can be up to any quantum number $w_k >= 1$, where k represents the mode being excited. The algorithm iterates over these classes until a desired precision of the FCP or maximum computation time is reached.

Santoro et al. exploit the fact that the calculations can be truncated on the size of the quantum number, as for each mode within C_n increasing w_k will lead to an increasing

FCF up to some maximum value, before decreasing from thereon. The point after which increasing w_k has only negligible effects is deemed w_k^{max}. w_k^{max} is unique for each oscillator and for each class, so for each class C_n the vector $\boldsymbol{W}_n$ can be defined, with values equal the different w_k^{max} values of each oscillator. To try and help visualise the excitations considered, an example of the possible excitation values for a 3 mode system is shown in table 3.1. This vector can then be used to estimate the number of FCFs required for each class: the approximate number of FCFs to be computed for the current class is equal to the average of $\boldsymbol{W}_n$ multiplied by $\binom{N}{n}$. If this estimate is greater than some cutoff value the threshold for negligibility is increased and $\boldsymbol{W}_n$ recalculated.

C_n	Possible Excited Modes	$\boldsymbol{W}_n$	Excitation Values
1	[1], [2] or [3]	[1,2,1]	[1,0,0], [0,1,0], [0,2,0], [0,0,1]
2	[1,2], [1,3] or [2,3]	[2,1,1]	[1,1,0], [2,1,0], [1,0,1], [2,0,1], [0,1,1]
3	[1,2,3]	[1,2,2]	[1,1,1], [1,2,1], [1,1,2], [1,2,2]

Table 3.1: Example description of the FCCs for a molecule with 3 modes. Note modes are numbered from 1 to N in this example.

The estimates for the w_k^{max} values are obtained using the results from C_1 and C_2. This screening is accomplished by choosing two threshold values, ε_1 for C_1 and ε_2 for C_2, for each class with $n > 2$. The previously calculated FCFs of all modes are then iterated over until the maximum quantum number for which one of the two inequalities $C_1(k, w_k) < \varepsilon_1$ and $C_2(k, l, [w_k, w_k]) < \varepsilon_2$ are not fulfilled. In order to do this screening, the FCFs are stored relative to their quantum number and the mode(s) excited.

This method of truncation leads to a bounded but exponential run time: the computation time will increase exponentially with the number of permutations in the binomial distribution up to some maximum beyond which it can increase no further. In practice, the computation time per class will likely slightly decrease past this maximum point. This is because a higher class C_m will have a greater number of permutations of excitations available to it than the lower class C_n (provided $n < m <= N/2$). Thus, limiting the number of FCFs considered will limit the maximum quantum number for each mode as well. This means that the computation time is reduced as the maximum quantum number considered affects computation time (as discussed in section 2.5).

It should be noted that the exact method of calculation for FCFs employed by Santoro et al. is to recursively compute Frank Condon integrals instead of the methods outlined here, where the FCF is the square of the absolute value of a Frank Condon integral. However, this choice does not affect the structure of the algorithm.

3.2 Physical Intuitions Behind Assumptions

The calculation truncations described by Santoro et al. are based both on attempting to reduce calculation time and also on physical intuitions.

Firstly, the idea to separate the FCP into classes is physically reasonable, as the contribution of multiple excited modes is no greater than the contribution of each excited

mode individually. Physically, this can be thought of as the excitation for any given mode individually providing a significant contribution, and the consideration of other modes alongside it contributing minor corrections to the value.

The screening conducted with C_1 and C_2 is also physically reasonable, as in the limit of no Duschinsky mixing (akin to independent oscillations) the excitation of multiple oscillators will be at most the product of each oscillator individually. In the case of some Duschinsky mixing, the C_2 class (which holds information of the mixing of pairs of molecules) can be used as the screening value instead. Whilst this assumption breaks down for systems that have highly correlated oscillations (i.e. far from the limit of no mixing), the majority of physical systems appear to observe these restrictions and hence the screening is valid. This was verified in their original paper, in which Santoro et al. found for multiple molecules the values up to the C_5 class contained at least 95% of the spectra.

The effects of changes in frequency and position on the spectra produced are described by the C_1 class, again because this is (in effect) describing the effect of the excitations on each individual mode. Given C_2 describes the effect of pairs of modes interacting (Duschinsky mixing), the classes C_1 and C_2 describe the majority of the three effects causing the emission of spectra: frequency changes, position changes and Duschinsky mixing. As such, it can be expected that not only will increasing class number have a diminishing effect on the FCP, but also that the first two classes can be used to calculate an estimate of the effect of higher classes.

Finally, whilst the values for ε_1 and ε_2 are chosen solely to limit the computation time (i.e. not necessarily with physical intuition), the tests they form part of can be computed rapidly for real examples. This is because the classes C_1 and C_2 being iterated over are of size $O(N)$ and $O(N^2)$, meaning many values of ε_1 and ε_2 can be iterated over in a short time. This means that the use of ε_1 and ε_2 will not introduce systematic errors into the result obtained, as if any errors do appear to be observed they can simply be recalculated before continuing with the full calculation.

In effect, the algorithm reduces computation time through the choice of two limits: one lower bound on the FCFs below which they are considered to be negligible and not provide a meaningful contribution, and one upper limit on the number of FCFs which caps how many FCFs are computed for higher classes. However, both of these have the effect of limiting the quantum number considered: the negligibility limit caps the quantum number for the first 2 classes (as mentioned before increasing the quantum number should increase the contribution of the FCF to the spectra before then decreasing the contribution), whilst the limit on the number of factors caps the quantum number considered at later classes. Moreover, the negligibility limit affects the quantum number limit for higher classes: as the higher classes are capped based on the contribution for the first two, if a lower limit is imposed on the first two it will have an effect on the later ones.

As the limits described are all effectively limiting the quantum number of modes considered, the assumptions made all depend on that one set of variables (it is not an individual variable as there is a different w_k^{max} for each mode. This means the simplification is based on a description of the system that depends solely on the quantum

number of the modes. However, this is entirely reasonable. As discussed in chapter 2, the quantum number of a system describes its energy level. Spectra are produced by the transition of atoms between energy levels, and hence the quantum number is the variable one would expect to limit for each mode to limit the computation of a vibrational spectrum.

3.3 Pseudocode

Here, I summarise the algorithm in such a way that it links to the way I implemented it, assuming a system with N modes indexed from 0 to N-1. It should be noted that in step 2 below Santoro et al. described altering the stored value of the C_2 FCFs to remove a bias introduced by position and frequency shifts (as these are accounted for in step 1): storing $C_2(k,l,w) - C_1(k,w) \times C_1(l,w)/C_0$ as opposed to just $C_2(k,l,w)$. However, upon testing my implementation this gave negative values for the shifts (which is not physically reasonable) and results that did not agree with experimental values. I therefore stored the true value of the C_2 FCFs instead. This worked empirically (see section 4.3, and did not decrease the number of FCFs considered (as the correction is a subtraction of a number which is always positive).

Also, Santoro et al. did not explicitly describe how their algorithm deals with C_2 FCFs when the pair of modes involved are not excited to the same number, simply stating to excite both to the same number. I decided to take this to mean excite both modes to the same quantum number, then compute the FCFs for the possible permutations of the excitations of each mode up to this point.

0. Compute C_0, which can be done in constant time

1. Compute the Class 1 FCFs:

for `k in modes` **do**
  $FCF \leftarrow \varepsilon + 1$ ▷ a value below ε is considered 0
  $w_k \leftarrow 1$
  while $FCF > \varepsilon$ **do**
    $FCF \leftarrow calculateFCF(k, w_k)$
    $C_1(k, w_k) \leftarrow FCF$
    $w_k \leftarrow w_k + 1$
  end while
end for

2. Compute the Class 2 FCFs. Here, both modes are initially excited up to the same quantum number before computing the permutations within:

for `k in modes` **do**
  for `l in modes;` $l > k$ **do**
    $FCF \leftarrow \varepsilon + 1$ ▷ a value below ε is considered 0
    $w_k \leftarrow 1$
    while $FCF > \varepsilon$ **do**
      $FCF \leftarrow calculateFCF(k, l, [w_k, w_k])$
      $C_2(k, l, [w_k, w_k]) \leftarrow FCF$

$w_k \leftarrow w_k + 1$
end while
for $(i = 1;\ \ i <= w_k;\ \ i++)$ **do**
for $(j = 1;\ \ j <= w_k;\ \ j++)$ **do**
if $i \neq j$ **then**
$FCF \leftarrow calculateFCF(k, l, [i, j])$
$C_2(k, l, [i, j]) \leftarrow FCF$
end if
end for
end for
end for
end for

3. Compute the FCCs for $n > 2$, which first requires computing the $\boldsymbol{W}_n$ vector:

(a) Find the $\boldsymbol{W}_n$ vector:

while $estimate > maximum$ **do**
$\varepsilon_1 \leftarrow \varepsilon_1 + \delta\varepsilon$
$\varepsilon_2 \leftarrow \varepsilon_2 + \delta\varepsilon$
for `k in modes` **do**
$FCF_1 \leftarrow \varepsilon_1 + 1$
$FCF_2 \leftarrow \varepsilon_2 + 1$
$w_k \leftarrow 1$
while not $(FCF_1 < \varepsilon_1\ \ \&\ \ FCF_2 < \varepsilon_2)$ **do**
$FCF_1 \leftarrow C_1(k, w_k)$
$l \leftarrow 0$
while not $(FCF_2 < \varepsilon_2)\ \ \&\ \ l < N$ **do**
if $l \neq k$ **then**
$FCF_2 \leftarrow C_2(k, l, [w_k, w_k])$
end if
$l \leftarrow l + 1$
end while
$w_k \leftarrow w_k + 1$
end while
$\boldsymbol{W}_n[k] \leftarrow w_k$
end for
$estimate \leftarrow mean(\boldsymbol{W}_n) \times \binom{N}{n}$
end while

(b) Calculate the FCFs for C_n, using the $\boldsymbol{W}_n$ vector previously calculated, assuming $permutations(n, w)$ is a function that returns all permutations for a class C_n with each mode excited up to its maximum value in $\boldsymbol{W}_n$:

for `(o, e) in permutations(n,` $\boldsymbol{W}_n$`)` **do**
$C_n(o, e) \leftarrow calculateFCF(o, e)$
end for

4. Repeat step 3 until a desired precision or class is reached

Chapter 4

Method

In order to complete this study, I had to accomplish two main goals: first, learn to use the Strawberry Fields [23] library for Python to simulate quantum computations, and then implement the algorithm of Santoro et al. [34] with quantum methods. Here, I describe these two processes in the hopes that the descriptions could be of use to a future student completing a project in a similar area.

4.1 Strawberry Fields and Quantum Computation

Whilst I did have some experience with studying quantum computers before undertaking this project, my knowledge was quite limited, especially on the field of photonic quantum computers. One of the most useful tools for understanding how quantum computers can actually be simulated was the tutorials on the strawberry fields website [44]. These provided a succinct and clear introduction into how to use the library, while also giving an idea of the physical processes being represented.

In effect there are 3 steps to a simulation with Strawberry fields: declare a circuit of a certain size, apply the required gates to the required qubits, and perform any final measurements/post processing. Functions are provided for all of these aspects, including a wide range of well-studied gates. The website also allows simulations to be run on 8-mode physical quantum devices via the cloud, but this was not required for this project.

Apart from general tutorials into how to use the language, there are also specific pages dedicated to common/important problems in quantum computation. These pages included tutorials on both non-Gaussian and Gaussian boson sampling which were very helpful in getting to grips with the simulations.

There was one limitation to these tutorials, however, in that they are focused heavily on how to use the Strawberry Fields library to simulate entire circuits. Whilst for the majority of quantum simulations this would likely prove adequate, for my project I needed to be able to use certain functions without the entire circuit. The exact functions and setup used are described in detail below, but in order to learn how to use these I had to look further into the source code library and manually find definitions for the lower

level functions that I required. Morerover, the tutorials each focused on different areas, and some data was found upon analysing other pages that seemed less relevant. The most important example of this in my case was the vibrational excitations tutorial, which not only contained the data needed to run experiments for a new molecule (pyrrole), but also explained in greater detail the relationship between a Duschinsky matrix and molecular data.

4.2 Initial Implementation

Having got to grips with the basics of Strawberry Fields, I set about implementing the project. The implementation was designed and all computations run on a home desktop PC running windows 11[1]. As the Strawberry Fields library is for Python, it made sense to adopt an object-oriented approach. To this end, I initially split my code into three files: `FCPComputation.py`, `FrankCondonComputations.py` and `Constants.py`.

4.2.1 FCPComputation.py

`FCPComputation.py` contains the main implementation of the algorithm of Santoro et al.. It first generates the Gaussian state from the molecular data provided using the Strawberry Fields function `qchem.vibronic.VibronicTransition()` on a circuit with as many modes as there are in the molecule. The parameters for this function are the squeezing and displacement parameters, and two interferometers which all together represent the Doktorov transformation. The `VibronicTransition()` function takes these molecular parameters and uses them to generate the displacement gates and a single unitary that represents the action of all the beamsplitters required to represent the Duschinsky matrix.

The gate parameters are in turn generated using the `qchem.vibronic.gbs_params()` function. This function takes in data specific to each molecule consisting of the initial and final frequencies, the molecule's Duschinsky matrix, the displacement vector for the molecule and the temperature being looked at. All experiments described here used a temperature of 0 Kelvin.

Once a Gaussian state has been generated, the algorithm of Santoro et al. is followed, as described in section 3.3. The Gaussian state is used in conjunction with `FrankCondonComputations.py` to calculate the FCFs for each class C_n. Each FCC is stored as a dictionary where the keys are lists representing the quantum number each mode has been excited to and the values are the FCFs, and all FCCs are stored in a list. Technically, each key is actually of length $2N$ for a molecule with N modes, where the first N modes represent the final state and the second N modes represent the starting state. However, as this project only looked at transitions from the ground state (temperature at $0K$), the second N modes were always all 0.

Using the description from section 3.3, the first two steps did not present any particular coding challenges. In order to compute the classes for $N >= 3$, however, a permutation of excited modes had to be generated. This was achieved using the

[1]Processor: AMD Ryzen 5 2600X Six-Core Processor 3.60 GHz, Ram: 16GB

`multiset_permutations()` function from the SymPy library for Python [26]. Specifically, a list of all possible combinations of excited modes was generated and then converted to a list of indices signifying which modes are excited for a specific FCF calculation. For example, a system with 3 excited modes out of 4 total would have the list of indices: [[0,1,2],[0,1,3],[0,2,3],[1,2,3]]. The order does not matter.

An escape was also added to the loop for $N >= 3$. This allows the program to be run initially for a high cut-off value to observe approximately how many FCFs each class would contribute, aiding the decision of where to cut off calculations and hence minimising the computation time for finding the $\boldsymbol{W}_n$ vector.

`FCPComputation.py` also contains code for plotting a calculated vibrational spectra. When plotting the spectra, as the FCP produced is incomplete (because the algorithm truncates the calculation before all FCFs are computed), the probability distribution values of energy are drawn from is incomplete. To get around this, a value that cannot be observed for the energy is assigned to the missing probability.

The dictionary keys (representing the initial and final modes as discussed above) are sampled from the FCP using the FCFs as probabilities with the `random.choice()` function in Python, and then converted to energies by taking the dot product of the first N modes with the final frequencies. In order to deal with non-ground state starting states, the dot product of the second N modes and the starting frequencies is then subtracted from this result. However, as this project only looked at ground-state starting states, this was always a subtraction of 0 and had no effect. The spectrum is then plotted using the Strawberry Fields `plot.spectrum()` function, which produces a histogram and convolutes it with a Gaussian.

4.2.2 FrankCondonComputations.py

`FrankCondonComputations.py` contains the methods to calculate the FCF for a given class, as well as calculate the percentage of a spectrum that has been computed (this is simply the sum of all FCCs, since the FCP is a probability distribution and therefore sums to 1). The FCF is calculated using the Strawberry Fields method `fock_prob()` (from the state class), which takes in a list representing the excitation of the modes and a value at which to truncate computations. The cutoff must be at least as large as the sum of the excitation of the modes, so was assigned either this value or 10 (the default truncation value).

This function makes use of another library, the walrus[17], which contains methods for calculating computationally hard problems. The key function for this project is `twq.density_matrix_element()` which is called by `fock_prob()`. The exact calculation used to compute the probability is different to the Hafnian calculation described in chapter 2, but the result is equivalent and a full explanation is not required for this project.

For example, `gaussianState.fock_prob([8,3,0,0], 11)` will calculate the FCF of a 4 mode system with the first mode excited to $w = 8$ and the second excited to $w = 3$. As mentioned previously, classes for $N <= 3$ can be computed simply, but in order to account for the permutations involved for $N >= 3$ a helper function was written

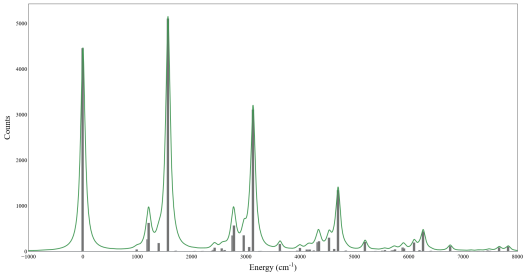
that creates a list of all the different excitations according to the WKMax vector for the provided list of excited modes. The only method for calculating FCFs currently implemented is through the Strawberry Fields library, but the code was written in such a way that it could easily be extended to include other methods of calculation (e.g. a direct alteration on the matrix maths involved) should comparisons wish to be made.

4.2.3 Constants.py

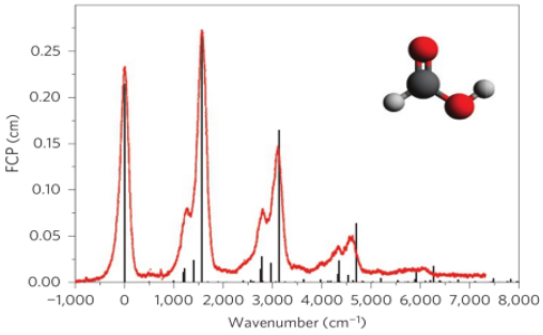
The constants contained within `Constants.py` are: the minimum quantum number allowed for any mode and the step by which to increment this when working through the algorithm (both of these were kept at 1, but were included in case experiments were done focussing on specific quantum numbers); a value below which calculation results are considered to be negligible (set to 1×10^{-10}, as suggested in [34]); the number of samples used to generate the vibrational spectra (set to 20000, as used on the Strawberry Fields tutorial); the value to assign the missing probability of an FCP to (set to -1000 as energies cannot be negative); an upper limit on the quantum number for any given excited mode (set to 30000 so as not to be used, as the negligible value provides a similar limit, but this was included in case calculation times were taking too long). The file also contains functions to access the vibrational data needed to generate the Gaussian state of any of the molecules I looked at.

4.3 Algorithm Verification

In order to verify that the algorithm worked as expected, the first spectra produced was that of formic acid. This molecule was chosen as a paper by Huh et al. [20] provides a spectra generated experimentally, whilst the Strawberry Fields tutorial on vibrational spectra provides the same result computed using a quantum simulation and shows it to be in agreement with the experimental version. Figure 4.1 shows the results of comparing the spectra calculated by my implementation compared to the experimental data, showing they are qualititively in agreement.



(a) Spectra computed using my implementation for formic acid up to class N=3. Note the value at -1000 represents the fact that the FCP is only 99.8% complete



(b) Spectra of formic acid obtained from experimental data. Inset: image rendering of formic acid. Reproduced from [20]

Figure 4.1: Comparison of experimentally and computationally determined vibrational spectra of formic acid.

A correlation coefficient for the pure Strawberry Fields data and the results from my own implementation was computed using the cosine similarity method, and gave a value of 0.999. The high correlation coefficient, along with apparent qualitative agreement, suggests my implementation worked correctly.

The spectra was also plotted for thymine, but no exact data was available to calculate the correlation coefficient and hence it cannot be used to verify the algorithm with high confidence. However, a similar qualitative result was observed and can be seen in appendix A for interest.

4.4 Changes to the Implementation

Once I had verified that my implementation gave the correct results, I focused on improving some aspects of the implementation that limited its use.

4.4.1 Algorithm Termination

Initially, my algorithm terminated after a specific class had been calculated. However, in some cases (specifically when comparing run times for different molecules) a more important termination condition was the percentage of the FCP/spectra that had been computed up to that point. To implement this an optional parameter was added to the main function which stated the desired percentage of the spectra, with a default value of 1 (corresponding to a 100% complete spectra and hence not affecting program execution). A variable was added to the main function to track how much of the total FCP had been computed after each FCF was calculated, and the program was terminated if this exceeded the completion target. This had no meaningful effect on the run time as the update was done as each FCF was computed, adding a constant $O(1)$ time.

4.4.2 Data Storage

I created a new Python file to aid with the storage of data: `DataStorage.py`. This file allows the saving of information of an overview of the calculations that took place for each FCC. Specifically after each class is calculated the molecule name, class number, time taken, approximate maximum number of FCFs and percent complete is written to a file, whilst the time taken to find a cutoff value for the number of FCFs is written to another. Timing was performed in a basic manner, simply using Python's `time()` function from the Time library before and after a specific action and then calculating the difference. A more sophisticated method was not required as exact values for computation times were not important. Rather, the patterns and relative differences were being investigated.

The C_0 class information for each molecule was not stored in this output file. Instead, I created a function to write the C_0 values for all molecules to a single file in a dictionary, using the Numpy [19] function `np.save()`. This was made possible by the addition of a constant to `Constants.py` that contained a list of all molecules I had run calculations on.

In order to be able to perform visualisations of the data generated, I added the capability to save the entire FCP after each molecule had finished computing. Initially, this was achieved by simply using Python's in-built file writing, but for larger molecules this began taking far too long to complete and led to excessively large files. To get around this, I switched to using the Numpy [19] function `np.save()` to write the list of FCCs to a file. This worked well until I began looking at thymine, a molecule with 39 vibrational modes, at which point saving the FCCs as a list of dictionaries took up too much space for np.save to work with. To solve this, I converted the list of dictionaries to a Numpy array of Numpy arrays, as np.save takes up considerably less space when using Numpy arrays compared to other Python objects.

4.4.3 Molecule Generation

Due to the difficulty in finding readily available Duschinsky matrices, and in order to investigate the performance of the algorithm under specific rare/unusual conditions, I added a function to `DataStorage.py` that generates a 'random' molecule. `generateMolecule()` generates a random orthogonal matrix that can be interpreted as a Duschinksy matrix, and either uses provided values for the displacement and frequency vectors or generates these randomly as well. The corresponding data is then written to a text file, so I added a function to `Constants.py` that could load such data.

In practice, I used formic acid as the main base for most of the generated molecules. This is because I wanted to have the molecules come from a physically accurate origin, but the size of pyrrole and thymine lead to unfeasible computation times[2].

The molecular data for formic acid was modified in three ways to investigate the effects on computation: setting the displacement vector to $\mathbf{0}$, replacing the Duschinsky matrix with an equally sized but randomly generated orthogonal matrix, and replacing the frequencies with frequencies designed to increase computation time. In order to get an unbiased view of the effect of randomising Duschinsky matrices, results were generally calculated for an average over 5 variations each identical except for their Duschinsky matrix.

In order to ensure that the data was not biased, `scipy.stats.special_ortho_group()` was used to generate random orthogonal matrices. This function is based on a known and verified method by G.W. Stewart [37], and ensures that generated matrices are drawn from a uniform distribution. The downside to this method is it meant I could not easily generate random matrices with specific features, such as high sparsity or block diagonality.

Setting the displacement vector to $\mathbf{0}$ was trivial, as all that is required is simply setting all entries in the array to 0.

The method for generating hard to compute frequencies was provided by my supervisor,

[2]The time to compute C_3 for pyrrole was 4230 seconds, whilst a randomly generated Duschinsky matrix but with all other parameters the same as pyrrole had not computed C_3 after 43200 seconds.

Raúl García Patrón. As discussed in chapter 2, the FCP is given by:

$$FCP(\omega) = \sum_{\boldsymbol{m}}^{\inf}(|\langle \boldsymbol{m}| \hat{U}_{dok} |\mathbf{0}\rangle|^2)\delta(\omega - \sum_{k}^{N} \omega'_k m_k) \tag{4.1}$$

This means that to generate a problem with hard to compute frequencies, the chosen frequencies must not be decomposable into each other. Letting each element of the final frequency matrix $\omega'_i = k\log(P_i)$, where P_i is the ith prime number leads to:

$$\begin{aligned} m_i\omega'_i &= m_i k\log(P_i) \\ &= k\log(P_i^{m_i}) \\ \implies \sum_{i=1}^{N} m_i\omega'_i &= k\log(\prod_{i=1}^{N} P_i^{m_i}) \\ &= k\log(\mathbb{N}) \end{aligned} \tag{4.2}$$

Thus, as each natural number as a unique decomposition, the frequencies are unique.

4.4.4 Data Visualisation

Finally, once I had all the data stored and ready for analysis, I created the new file `DataVisualiser.py`, which contains functions for loading data (and reformatting if required, as is the case for molecules like thymine) and visualising results. The results from these functions are described in greater detail in chapter 5, but the general methodology for the functions was to load the data using `np.load()`, iterate through and record the desired results, before plotting with Matplotlib [21]. The exact description of each function is not presented here, as the data created is more important (other methods could have achieved the same visualisations). However, they were still written as functions rather than one-off scripts to allow for the correction/recreation of any visualisations I realised were incorrect at a later date.

Chapter 5

Results

As the goal of this project was to investigate practical (rather than the well established theoretical) quantum advantage, I focused on three real molecules: formic acid, pyrrole and thymine. The molecular data for formic acid and pyrrole was obtained from the Strawberry Fields library (in the `data` module), whilst that of thymine was provided by Joonsuk Huh[1]. All other molecules generated were created based on one of these molecules, for example by taking the frequencies of formic acid but generating a random Duschinsky matrix to use instead of the known matrix.

5.1 Computation Time of the Algorithm of Santoro et al.

As discussed previously, the expected run time of an algorithm to compute the FCP for a molecule with N modes is $O(2^N)$. In order to investigate the effects of the speedups employed by the algorithm of Santoro et al. I calculated the time taken to reach a spectra that was at least 83.76% complete. It should be noted that FCPs and FCFs are unique for each molecule, so the exact percentage of the spectra completed after the times shown here is not 83.76% (see appendix D for the exact results). This value was chosen as it was the highest completion percentage of spectra computed for thymine, the largest molecule investigated.

Figure 5.1 shows the timing analysis, along with an exponential function fit to the data (using `scipy.optimize.curve_fit`). As can be seen, the run time is still exponential. However, it is exponential in approximately $N/2-4$ as opposed to N, verifiying that there is indeed a significant speed up introduced by the algorithm of Santoro et al. The key limitation of this data is the small data set chosen; future analysis investigating the trend for a greater range of molecules would be beneficial.

A graph of the same data but without thymine is in figure D.1 in appendix D, where a similar result appears showing the high coefficient of determination is not just due to the large difference in times taken.

[1] Joonsukhuh@gmail.com

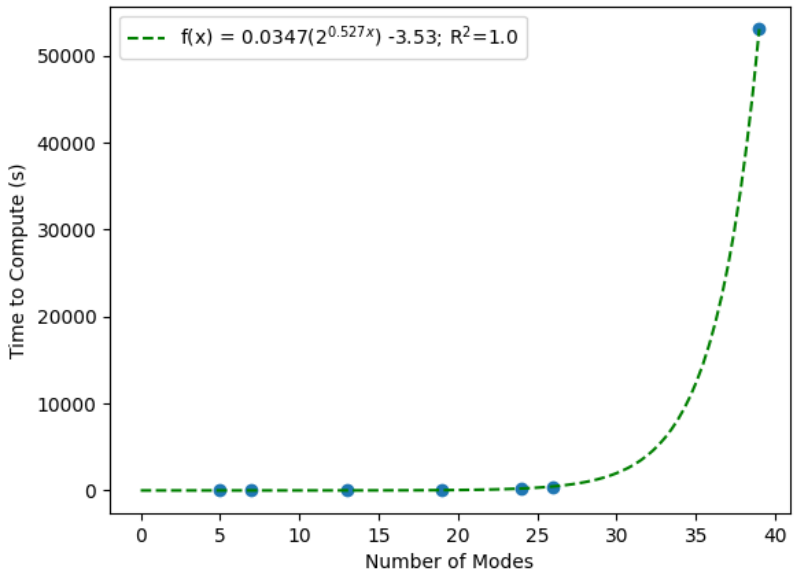
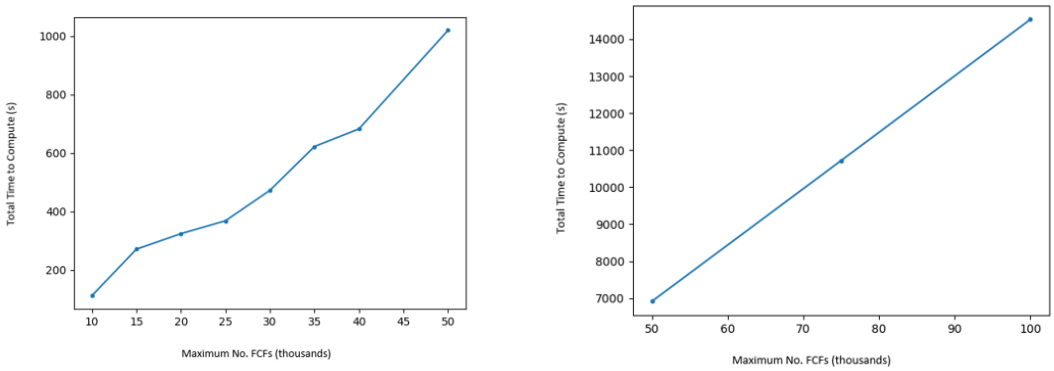


Figure 5.1: Time to compute at least 83.76% of a spectra. The molecules looked at in size order are: the smaller diagonal block of pyrrole, formic acid, the smaller diagonal block of thymine, the larger diagonal block of pyrrole, pyrrole, the larger diagonal block of thymine, and thymine. For a discussion on what the smaller/larger blocks represent see section 5.4. R^2 is the coefficient of determination.

5.1.1 Relationship Between Computation Time and Maximum Number of FCFs for Each Class

In order to verify that the choice of cut-off values didn't bring the run time out of the exponential domain, I measured the time taken to compute the FCP for varying cut-offs.



(a) Formic acid (b) Pyrrole

Figure 5.2: Relationship between total computation time and maximum number of FCFs for all classes for formic acid and pyrrole, computed up to c=7.

Figure 5.2 shows the results observed for formic acid and pyrrole. As expected, the total time to compute increases approximately linearly with the maximum computed

number of FCFs. This verifies that whilst the choice of cut-off values for the number of integrals does have a noticeable effect on the computation time, the choice of cut-off values alone cannot bring the time complexity out of the exponential domain.

5.2 Computation Time, Class Number and Number of FCFs

5.2.1 Contribution to Spectra by Class

The assumptions of Santoro et al. ultimately all boil down to one key point: that a near-enough complete FCP/spectra can be generated after only the first few FCCs have been computed. It therefore made sense to first plot the cumulative percentage of the spectra of the various molecules investigated to verify this assumption. This is shown in figure 5.3

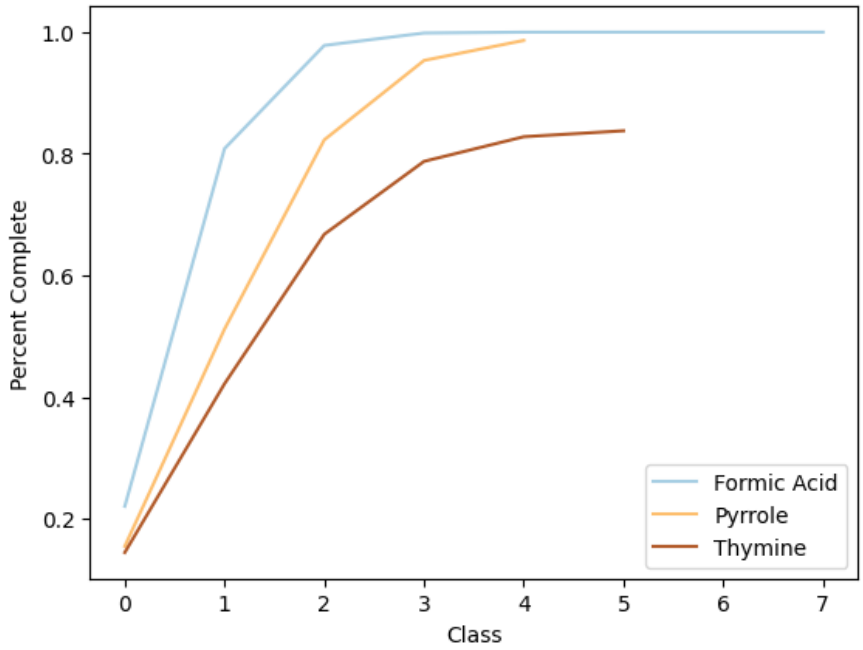


Figure 5.3: Visualisation of the relationship between percentage contribution to the spectra and time to compute. Here the program terminated once a certain class, rather than percentage, had been reached to prevent excessive computation times.

As expected, the higher classes provide a smaller contribution to the spectra of each molecule than the lower classes. It is also clear that the contribution at lower class levels decreases for larger molecules. This is to be expected: as mentioned previously the lower classes are expected to contribute more to the spectra because they describe independent oscillations. As the size of molecules increases, the number of oscillations which can occur simultaneously whilst remaining mostly independent will increase.

5.2.2 Effect of Class Number on Time and Contribution to Spectra

Figure 5.4 demonstrates the relationship between computation time, percent contribution to the spectra and class number for the 3 real molecules considered. As the spectra is not computed to 100% completion, the sum of the contributions does not add to 1 but rather the values shown in table 5.1.

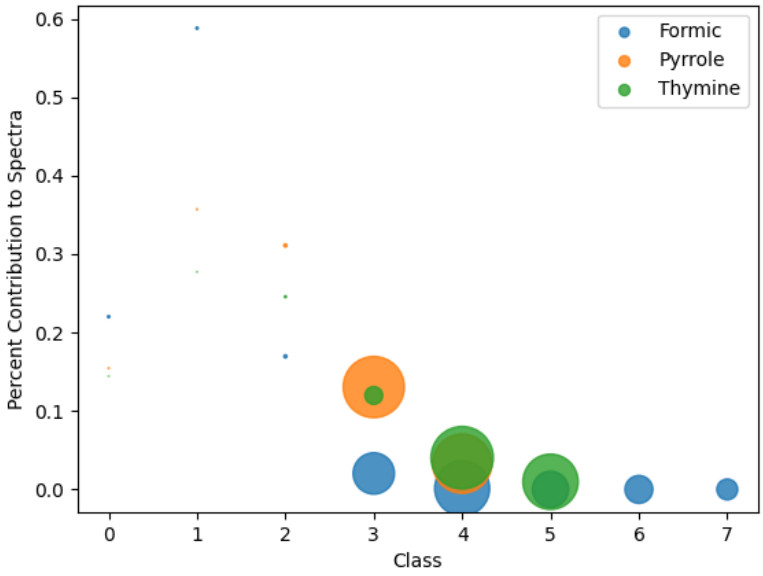


Figure 5.4: Visualisation of the relationship between percentage contribution to the spectra, class number and time to compute. The area of each bubble is proportional to the percentage of the total computation time for the spectrum of that molecule.

Molecule	Moleculuar Modes	Maximum Class	% Spectra Computed	Total Time to Compute (s)
Formic Acid	7	7	0.9998	779.13
Pyrrole	24	4	0.9861	8208.76
Thymine	39	5	0.8376	53041.58

Table 5.1: Calculation data for the three real molecules considered.

As can be seen, all three molecules follow a similar pattern: an increase in percent contribution from C_0 to C_1, followed by an exponential decrease. Physically, this makes sense: one would expect the sum of the contribution for non-zero modes to outweigh the $\mathbf{0}$ mode as the non-zero modes represent the transition. The maximum at C_1 can be attributed to the vibrations of individual modes having a greater effect than the interaction between modes.

The computation time is also in agreement with what the algorithm predicts. The low computation time for C_0 to C_3 followed by an exponential blow up that is bounded by some maximum is exactly as described by the algorithm of Santoro et al and discussed in Chapter 3

An interesting feature of Figure 5.4 is that whilst the plots for the three molecules seem to have a similar distribution, the distribution appears to be flattened and shifted for pyrrole and thymine. A possible explanation for this is that pyrrole and thymine are both larger molecules than formic acid. This may mean that the higher classes of larger molecules provide more information on the spectra than corresponding classes for smaller molecules, but more research would be required on a greater number of molecules to confirm this.

5.3 Random Molecules

Following on from the results described in section 5.2, randomly generated molecular data was used to investigate the effects of various molecular parameters on the computations completed.

For graphing purposes in this section, the suffix 'delta=0' signifies a modified displacement vector that is set to $\mathbf{0}$, the prefix 'random' signifies a randomly generated Duschinsky matrix, and the prefix 'prime' signifies a molecule with frequencies chosen to be hard to compute.

5.3.1 Contribution to Spectra by Class

Figure 5.5 summarises the results of comparing randomly generated molecules to formic acid. Firstly, as can be seen, the effect of replacing frequencies with values designed to make the calculations harder is minimal. This suggests that in nature there are no two

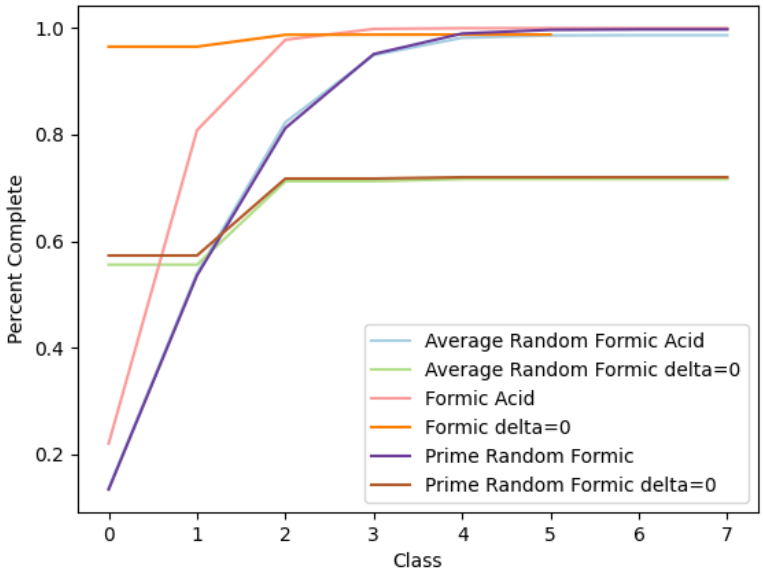


Figure 5.5: Cumulative percentage contribution to the spectra at each class for formic acid and randomly generated molecules based on formic acid.

occupation numbers that lead to the same energy value. As such there are no clusters of probabilities and hence they are naturally hard to compute.

The effect of replacing the Duschinsky matrix of a molecule with an equally sized but randomly generated orthogonal matrix is also as expected: for a given molecule, a greater number of classes must be computed in order to achieve the same precision in the spectrum computed. This is because the Duschinsky matrix describes the mixing between modes, so if the matrix is less sparse (see Appendix B for a visualisation of this) there is more mixing and hence higher classes are required.

5.3.2 Computation Time of Random Molecules

In order to compare the computation time, rather than just the class number required, for a random and real Duschinsky matrix, Figure 5.6 was produced. As is clear from this graph, the randomly generated molecules took on average much longer to compute for all classes C_3 or higher. This is to be expected as described previously. Interestingly, all the randomly generated molecules took less time than formic to compute higher classes (see Figure C.2 in appendix C for a zoomed in version of this same graph), meaning the increased computation time is solely due to the large time spent at C_3. As all classes after C_2 still have the same upper limit on the number of FCFs to compute, this suggestst that the C_3 class involves higher quantum numbers/excitation values for random Duschinsky matrices with little sparsity.

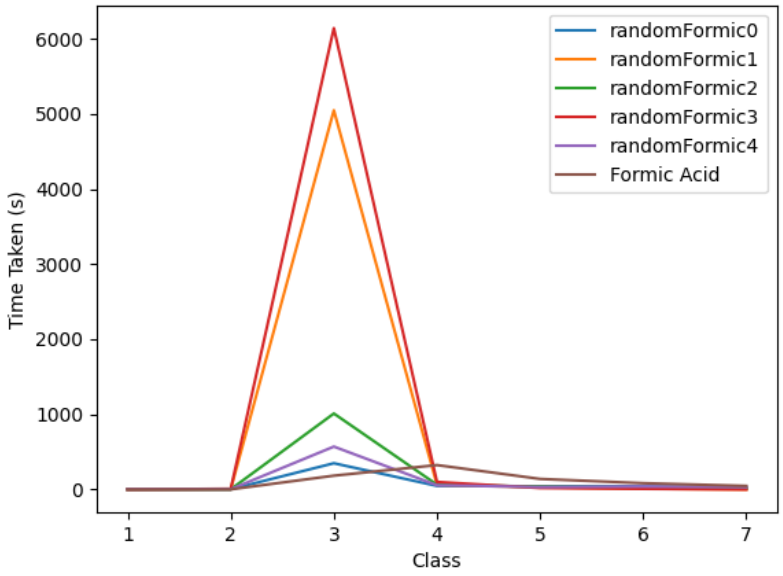


Figure 5.6: Time taken to compute each FCC for formic acid and 5 molecules with identical vibrational data but randomly generated Duschinsky matrices.

5.3.3 Effect of Removing the Displacement Vector

Another interesting result is the effect of setting the displacement vector to $\mathbf{0}$. For the true molecular data of formic acid, this leads to the spectra being described fully after just the classes C_0 and C_1 have been computed. This can be explained by the lack of displacement leading to an increased contribution of the mode $\mathbf{0}$ as there are no longer any differences in displacements, only vibrations. However, for randomly generated Duschinsky matrices, this effect is not reproduced, as shown in Figure 5.5. Whilst the spectra computed after the C_0 and C_1 class is once again greater than the contribution for the equivalent molecules with a non-zero displacement vector, the full computation fails to completely describe the spectrum produced.

At first, I thought the explanation for this would be that the cut off values chosen were limiting the results, either because all higher classes were now contributing less and hence more FCFs were being lost below the lower bound of what is considered negligible, or because a higher quantum number was required to explore the higher classes and this was limited by the upper bound on the number of FCFs to compute. I therefore re-ran the experiment but with varying values for each of these cutoffs, as shown in Figure 5.7.

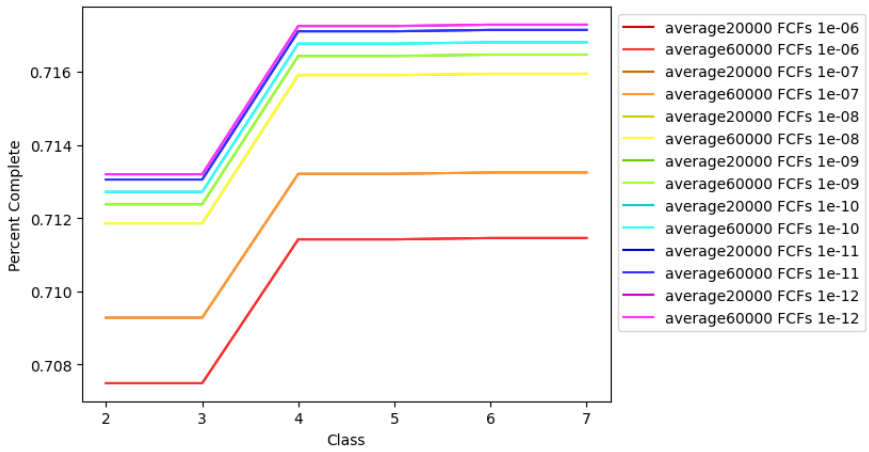


Figure 5.7: Cumulative percentage contribution to the spectra at each class for varying cut-off values averaged for 5 radonmly generated molecules. The 20000 or 60000 values represent the maximum number of FCFs computed, whilst the value in scientific notation is the threshold for negligibility. Only values of $C_n > 1$ are shown to highlight the effect, but the pattern is the same for C_1 and C_0 (see FigureC.1 in appendix C)

Changing the negligibility has only a minor and diminishing effect, suggesting this is not to blame as the trend appears unlikely to continue at a great enough pace to reach 100% completion. Changing the limit on the number of FCFs also does not appear to explain the incompleteness, as a cutoff of 20000 and 60000 had no change in the completeness. Whilst it might be thought that this simply isn't a large enough difference in the FCF limit, the time taken to find the $\boldsymbol{W}_n$ vector does not change significantly for the two different limits. If the incompleteness were due to not considering enough FCFs, it would be expected that the iteration time to find the maximum number of FCFs to compute would change with the limit. The fact that it doesn't implies the physical

limit on the number of FCFs is within both bounds.

As neither of these limits seem to have a meaningful effect on the results obtained for a molecule with **0** displacement vector, the most likely explanation for the incompleteness is that the algorithm breaks down in such cases. The most likely area for this to occur is the use of the classes C_1 and C_2 to limit the factors computed for higher classes. As the algorithm looks for the highest quantum number that the two previously discussed conditions are not satisfied for, it is possible that in the case of high Duschinsky mixing (as is the case for the randomly-generated Duschinsky matrices) the algorithm is inadvertently screening out meaningful values for higher classes.

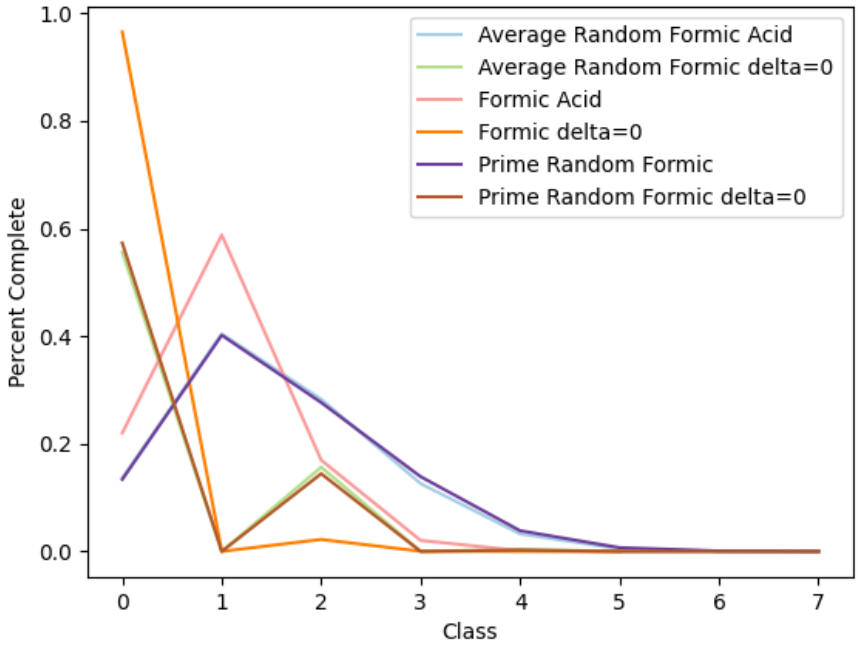


Figure 5.8: Percentage contribution to the spectra at each class for formic acid and randomly generated molecules based on formic acid.

As mentioned previously, the screening assumptions are that the mixing is accounted for in C_2, displacement and frequency shifts in C_1 and hence the majority of the system is described by these two classes. If, however, the system involves a large amount of mixing (i.e. the Duschinsky matrix is not sparse) the proportion of the spectrum that C_2 provides should decrease. If the system involves low displacement, C_1 should provide a lower proportion of the spectra. Both of these effects are more clearly visible in Figure 5.8, which shows the individual (rather than cumulative) contribution of each class. Thus, the combination of these two classes should provide less of the spectrum than they did for a non-random, non-zero displacement molecule, suggesting they cannot be accurately used to reduce computation times in non-sparse cases.

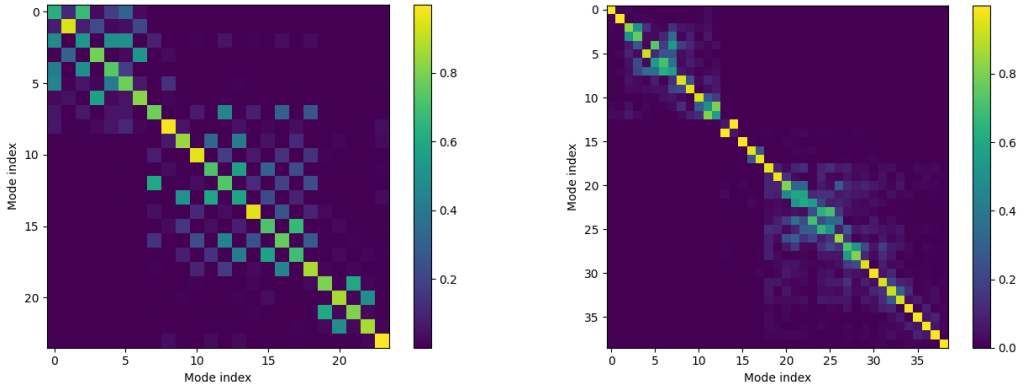
However, this does not necessarily mean that the algorithm has failed, as in physically accurate situations the Duschinsky matrices of molecules do tend to be sparse. This

means that the algorithm appears to only perform badly in non-physical cases, but more research would be required to verify this.

5.4 Exploitation of Block Diagonality

5.4.1 Motivation

Whilst producing the visualisations of Duschinsky matrices and the corresponding vibrational spectra for molecules, I noticed that the Duschinsky matrices of real molecules tend to be highly sparse and often quite block diagonal. An example of this for pyrrole and thymine is shown in figure 5.9. This turned out to be a well known fact, for example the data I was given for thymine was described in two block diagonal sections, whilst the formic acid data used throughout is only for the first seven of the nine modes, as it is known that the first block diagonal section provides the majority of the spectra [20]. As there are current technical challenges in building large quantum computers (as mentioned previously), this provides a limit on the size of a molecule that can be investigated using GBS. I therefore decided to investigate if it is possible to compute the spectra for the diagonal blocks individually and then recombine them to obtain the full spectra, both to see if this provided a computational speed up and as it would suggest a simpler machine could be developed.



(a) Pyrrole (b) Thymine

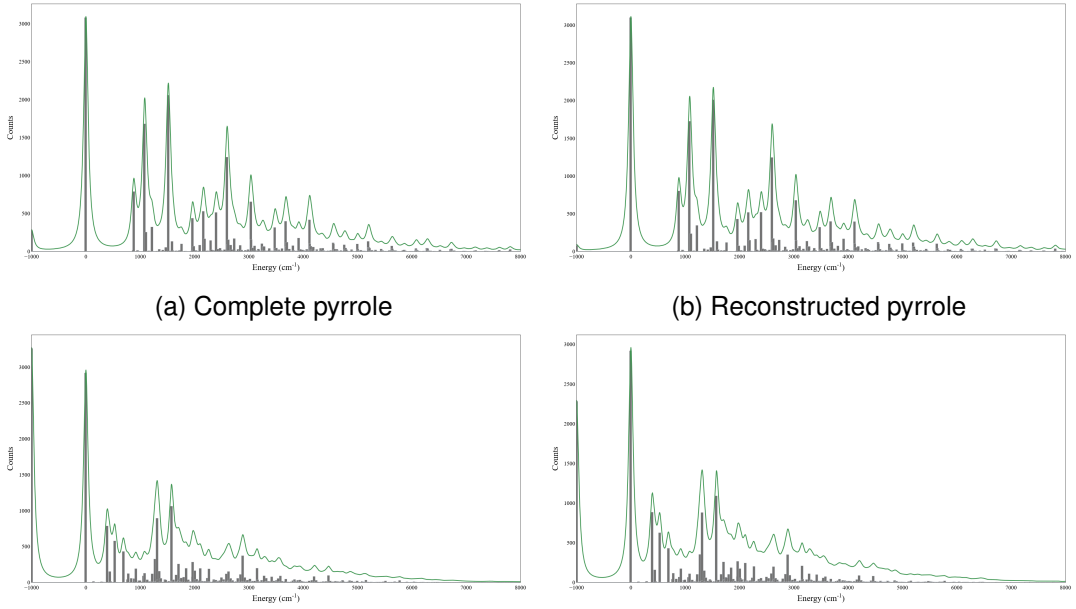
Figure 5.9: Visualisation of Duschinsky matrices for pyrrole (N=24) and thymine (N=39).

After completing this experiment, I found the work of Dierksen and Grimme [10] which was more advanced than mine. First, the Duschinsky matrix was approximaged with a more block diagonal version of itself, chosen to reach some threshold of similarity. Then the FCPs for each block were calculated and multiplied together to give the full FCP. Their method was able to accurately compute molecules even with up to 468 vibrational modes to a high degree of accuracy. Thus, my approach had a strong theoretical founding.

5.4.2 Findings

Formic acid had too few modes to investigate the block diagonality effects in detail, so I focused on pyrrole and thymine. As can be seen from Figure 5.9a, the Duschinsky matrix of pyrrole can be split into two blocks at mode 19, whilst Figure 5.9b shows thymine can be split into two blocks at mode 13.

The total spectra were reconstructed assuming the spectra could be convoluted via a simple assumption that the probabilities are independent (i.e. assuming that, for example, the probability of all modes being 0 in thymine is equal to the product of all modes in the two sub-spectra being 0). The spectra produced are shown in Figure 5.10, demonstrating qualitive agreement between the complete and reconstructed spectra. As summarised in Table 5.2, not only do the reconstructed spectra seem to highly agree with the calculated spectra of the whole system, but they can be computed much faster.



(a) Complete pyrrole

(b) Reconstructed pyrrole

(c) Complete thymine

(d) Reconstructed thymine

Figure 5.10: Comparison of complete spectra and spectra of two partial blocks convoluted together. For pyrrole the maximum class considered was C_4, for thymine it was C_5.

Molecule	Correlation Coefficient	Full Spectrum Computation Time	Partial Spectra Computation Time	Convolution Time	% Speedup
Pyrrole	0.998	8208.76	6043.82	3	20.5
Thymine	0.992	53041.58	10846.27	33	73.7

Table 5.2: Comparison of results obtained for reconstructed and full spectra computation. All times are in seconds. All values for pyrrole were for computations up to class C_4, and for thymine C_5. The correlation coefficient was calculated using the cosine method.

Initially, this speed up was less clear, as the time taken to loop over all FCFs of the partial spectra and multiply them together was also $O(2^N)$ (simply due to the sheer number of FCFs computed). However, this time was massively reduced because the FCFs are stored in descending order for each class (as a consequence of the method used to compute them). This meant the calculation could be cutoff as soon as the product of two factors was less than the defined constant for negligibility. Thus, the speed up is still observed when accounting for the time taken to convolute the two spectra.

Physically, this is to be expected: if parts of the spectra are independent of each other, their correlation described by the Duschinsky matrix should be zero leading to block diagonality. This would mean that the FCPs produced should be independent of each other meaning treating them as independent probabilities is valid.

One limitation of this method is that it leads to a less clear image on what percent of the spectra has been calculated. This was also noted by Dierksen and Grimme in their paper. For both pyrrole and thymine the reconstructed data give an FCP with a sum closer to 1 than the individual full spectra. Whilst small for this data, more work needs to be done for more molecules to see if this discrepancy grows, as if a spectra is believed to be more complete than it actually is it could lead to incorrect observations.

5.4.3 Primitive Timing Analysis

Figure 5.11 shows the relationship between time and number of modes.

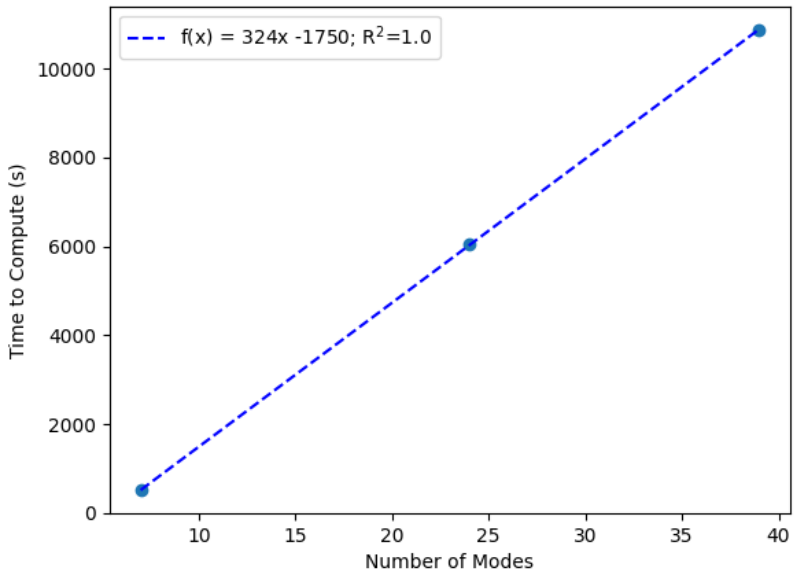


Figure 5.11: Time to compute spectra of formic acid, pyrrole and thymine. The times for the sub-blocks are not plotted as they are not physically meaningful. Note the exact percentage computed cannot be derived from the combined block-diagonal method as without knowing the full spectra beforehand the accuracy cannot be calculated.

Whilst the data available to me for investigating block diagonality was limited, I decided

to plot the time taken as a function of number of modes anyway to see if there were any visible trends. The trendline was produced using linear regression via the scipy function `scipy.stats.linregress`.

As can be seen, for the three molecules investigated in this project, a combination of the method of Santoro et al. and an exploitation of block diagonality can greatly reduce computation time. In fact, for these three molecules the scaling appears to be linear. It is unlikely that this is a true reduction to linearity as in effect the assumptions are just using truncations to reduce computation time, and rather this is just the trend for small values for a slower-growing exponential function. However, the fact that for molecules with up to 40 modes the scaling is linear suggests this may be of practical use, as molecules cannot be infinite in size, so such a hybrid method would likely be able to perform well even for larger molecules.

Chapter 6

Conclusion

6.1 Discussion

In summary, the calculations of vibrational spectra are a good candidate for quantum advantage, but the practicality of such advantage is not clear. Whilst existing classical algorithms, such as those of Santoro et al. or Dierksen and Grimme, are not able to reduce the computation time to polynomial, the assumptions made are still able to massively reduce computation time. Here, I have also demonstrated that these two known approximations can be applied together, hence further reducing computation time.

I have also shown that the assumptions these classical algorithms rely on do not always hold, for example for systems with low sparsity in their Duschinsky matrices. A quantum system would not be subjected to such slow downs, so whilst they have their technical challenges (in terms of being built), if a scalable GBS system capable of detecting more than a few photons of light in each mode was developed it would outperform classical computer in computing vibrational spectra. However, it appears these assumptions only break down for unphysical systems, as all three real molecules investigated here performed well, suggesting the quantum advantage of GBS is not in fact practical.

As discussed previously, there are technical limitations to building larger quantum systems. However, if the Duschinsky matrices were split into diagonal blocks multiple smaller circuits could be used to not only greatly speed up computation time, but enable such devices to be built in the nearer future. Whilst this suggests the exploitation of block diagonality would be useful when building quantum devices, it cannot be forgotten that the same method can greatly reduce computation time on classical machines as well.

The results obtained from looking at randomly-generated molecules are also significant in that they demonstrate the limits of current classical algorithms for solving this task. These limits exist not only in the computational complexity sense, as is well studied, but also in the requirement that classical algorithms can only speed up computation by using approximations. The demonstration of the collapse of the commonly used

algorithm of Santoro et al. when displacement is not present in a system, for example, could have applications when computing the vibrational spectra of complex systems.

The case for practical quantum advantage relies on the fact that the classical solution scales exponentially. Whilst my results do show this exponential scaling for real molecules, I have also shown that it can be reduced to approximately linear in practical terms by the assumptions mentioned previously. However, this needs to be further investigated with more molecules and combinations of methods as the data used here is limited.

6.2 Future Work

Going forwards, further work could continue to be done in the area of Gaussian boson sampling. There are still significant technical challenges in building a quantum device, so any claim of practical quantum advantage would first require technical advances.

There is also significant work that could be done to further improve the classical algorithms currently used. This is significant not only because it can improve current computation times, but also because if a classical algorithm exists that computes quickly in most physical cases the claims of practical quantum advantage become harder to prove.

Combinations of different classical speed ups are a promising avenue, for example the combination of block diagonality and the algorithm of Santoro et al. appears to provide a promising avenue for massive time reductions. Whilst the results here are promising, they are for a limited data set due to the availability of data and time constraints of the project. Further verification is required to ensure this trend continues for other real molecules, both in terms of still giving correct results and ensuring that other molecules do still observe block diagonality and not just sparsity.

Finally, more work is required on investigating the effect of different Duschinsky matrices on computation time and reliable results. If we continue using assumptions such as those described previously but find they break down in edge cases the spectra computed will be incorrect, leading to them becoming worthless. If, however, we can further understand how changing a Duschinsky matrix changes the computation of a spectra, results will be greatly improved. This data is most important with regards to physical systems, so future work should focus on true systems and not just those which are randomly generated.

Bibliography

[1] Scott Aaronson and Alex Arkhipov. The Computational Complexity of Linear Optics. *Theory of Computing*, 9(1):143–252, February 2013.

[2] William Bell Allan. *Fibre optics: theory and practice*. Springer Science & Business Media, 2012.

[3] Juan Miguel Arrazola and Thomas R. Bromley. Using Gaussian Boson Sampling to Find Dense Subgraphs. *Phys. Rev. Lett.*, 121(3):030503, July 2018.

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, October 2019.

[5] Olivier Beyssac. New Trends in Raman Spectroscopy: From High-Resolution Geochemistry to Planetary Exploration. *Elements*, 16(2):117–122, April 2020.

[6] Jacques Carolan, Jasmin D. A. Meinecke, Peter J. Shadbolt, Nicholas J. Russell, Nur Ismail, Kerstin Wörhoff, Terry Rudolph, Mark G. Thompson, Jeremy L. O'Brien, Jonathan C. F. Matthews, and Anthony Laing. On the experimental verification of quantum complexity in linear optics. *Nat. Photonics*, 8:621–626, August 2014.

[7] William R. Clements, Jelmer J. Renema, Andreas Eckstein, Antonio A. Valido, Adriana Lita, Thomas Gerrits, Sae Woo Nam, W. Steven Kolthammer, Joonsuk

Huh, and Ian A. Walmsley. Approximating vibronic spectroscopy with imperfect quantum optics. *J. Phys. B: At. Mol. Opt. Phys.*, 51(24):245503, November 2018.

[8] Christophe Couteau. Spontaneous parametric down-conversion. *Contemp. Phys.*, 59(3):291–304, July 2018.

[9] Andrew J. Daley, Immanuel Bloch, Christian Kokail, Stuart Flannigan, Natalie Pearson, Matthias Troyer, and Peter Zoller. Practical quantum advantage in quantum simulation. *Nature*, 607:667–676, July 2022.

[10] Marc Dierksen and Stefan Grimme. An efficient approach for the calculation of Franck–Condon integrals of large molecules. *J. Chem. Phys.*, 122(24):244101, June 2005.

[11] E.V. Doktorov, I.A. Malkin, and V.I. Man'ko. Dynamical symmetry of vibronic transitions in polyatomic molecules and the franck-condon principle. *Journal of Molecular Spectroscopy*, 56(1):1–20, 1975.

[12] Andrew Downes and Alistair Elfick. Raman Spectroscopy and Related Techniques in Biomedicine. *Sensors*, 10(3):1871–1889, March 2010.

[13] F. Duschinsky. The importance of the electron spectrum in multi atomic molecules. concerning the franck-condon principle. *Acta Physicochim. USSR*, 7:551, 1937.

[14] John R Ferraro. *Introductory raman spectroscopy*. Elsevier, 2003.

[15] Fujitsu. Supercomputer Fugaku, October 2022. [Online; accessed 19. Oct. 2022].

[16] A. Furusawa, J. L. Sørensen, S. L. Braunstein, C. A. Fuchs, H. J. Kimble, and E. S. Polzik. Unconditional quantum teleportation. *Science*, 282(5389):706–709, 1998.

[17] Brajesh Gupt, Josh Izaac, and Nicolás Quesada. The Walrus: a library for the calculation of hafnians, Hermite polynomials and Gaussian boson sampling. *Journal of Open Source Software*, 4(44):1705, December 2019.

[18] Craig S. Hamilton, Regina Kruse, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, and Igor Jex. Gaussian boson sampling. *Phys. Rev. Lett.*, 119:170501, Oct 2017.

[19] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[20] Joonsuk Huh, Gian Giacomo Guerreschi, Borja Peropadre, Jarrod R. McClean, and Alán Aspuru-Guzik. Boson sampling for molecular vibronic spectra. *Nat. Photonics*, 9:615–620, September 2015.

[21] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[22] Soran Jahangiri, Juan Miguel Arrazola, Nicolás Quesada, and Alain Delgado. Quantum Algorithm for Simulating Molecular Vibrational Excitations. *arXiv*, June 2020.

[23] Nathan Killoran, Josh Izaac, Nicol'as Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. Strawberry Fields: A software platform for photonic quantum computing. *Quantum*, 3:129, 2019.

[24] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, January 2001.

[25] Lars S. Madsen, Fabian Laudenbach, Mohsen Falamarzi. Askarani, Fabien Rortais, Trevor Vincent, Jacob F. F. Bulmer, Filippo M. Miatto, Leonhard Neuhaus, Lukas G. Helt, Matthew J. Collins, Adriana E. Lita, Thomas Gerrits, Sae Woo Nam, Varun D. Vaidya, Matteo Menotti, Ish Dhand, Zachary Vernon, Nicolás Quesada, and Jonathan Lavoie. Quantum computational advantage with a programmable photonic processor. *Nature*, 606:75–81, June 2022.

[26] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[27] Rawad Mezher and Shane Mansfield. Assessing the quality of near-term photonic quantum devices, 2022.

[28] Thomas Monz, Daniel Nigg, Esteban A. Martinez, Matthias F. Brandl, Philipp Schindler, Richard Rines, Shannon X. Wang, Isaac L. Chuang, and Rainer Blatt. Realization of a scalable shor algorithm. *Science*, 351(6277):1068–1070, 2016.

[29] John Nash. Vibrational Modes of Formic Acid, January 2014. [Online; accessed 15. Mar. 2023].

[30] E. Neil. Lecture Notes on Simple harmonic oscillators. [Online; accessed 15. Mar. 2023], December 2021.

[31] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[32] Nicolás Quesada, Rachel S. Chadwick, Bryn A. Bell, Juan Miguel Arrazola, Trevor Vincent, Haoyu Qi, and Raúl García-Patrón. Quadratic speedup for simulating Gaussian boson sampling. *arXiv*, October 2020.

[33] Raúl GARCIA-PATRON SANCHEZ. *Quantum Information with Optical Continuous Variables: from Bell Tests to Key Distribution*. PhD thesis, Université Libre de Bruxelles, 2007.

[34] Fabrizio Santoro, Roberto Improta, Alessandro Lami, Julien Bloino, and Vincenzo Barone. Effective method to compute Franck-Condon integrals for optical spectra of large molecules in solution. *J. Chem. Phys.*, 126(8):084509, February 2007.

[35] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[36] Matthew Sparkes. Google's quantum supremacy challenged by ordinary computers, for now. *NewScientist*, August 2022.

[37] G. W. Stewart. The Efficient Generation of Random Orthogonal Matrices with an Application to Condition Estimators. *SIAM J. Numer. Anal.*, July 2006.

[38] Simon Titmuss and Stewart McWilliams. Lecture notes on physics of matter. 2021.

[39] Christopher S. Wang, Jacob C. Curtis, Brian J. Lester, Yaxing Zhang, Yvonne Y. Gao, Jessica Freeze, Victor S. Batista, Patrick H. Vaccaro, Isaac L. Chuang, Luigi Frunzio, Liang Jiang, S. M. Girvin, and Robert J. Schoelkopf. Efficient Multiphoton Sampling of Molecular Vibronic Spectra on a Superconducting Bosonic Processor. *Phys. Rev. X*, 10(2):021060, June 2020.

[40] Yibin Wang, Fangming Liu, Q. Liang, B.-J He, and J.-L Miao. Low-temperature degradation mechanism analysis of petroleum hydrocarbon-degrading antarctic psychrophilic strains. *Journal of Pure and Applied Microbiology*, 8:47–53, 02 2014.

[41] Xanadu. Strawberry Fields: Introduction to quantum photonics, September 2022. [Online; accessed 19. Oct. 2022].

[42] Xanadu. Boson sampling and the permanent — Strawberry Fields, January 2023. [Online; accessed 17. Mar. 2023].

[43] Xanadu. Gaussian boson sampling and the Hafnian — Strawberry Fields, January 2023. [Online; accessed 17. Mar. 2023].

[44] Xanadu. Tutorials — Strawberry Fields, January 2023. [Online; accessed 22. Sep. 2022].

[45] Anton Zeilinger. Quantum teleportation, onwards and upwards. *Nat. Phys.*, 14:3–4, January 2018.

[46] Han-Sen Zhong, Yu-Hao Deng, Jian Qin, Hui Wang, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Dian Wu, Si-Qiu Gong, Hao Su, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Jelmer J. Renema, Chao-Yang Lu, and Jian-Wei Pan. Phase-programmable gaussian boson sampling using stimulated squeezed light. *Phys. Rev. Lett.*, 127:180502, Oct 2021.

[47] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing

You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

Appendix A

Spectra of Thymine

Below are the spectra of thymine as generated using my implementation (Figure A.1), and the simulation by Xanadu compared to experimental data (Figure A.2).

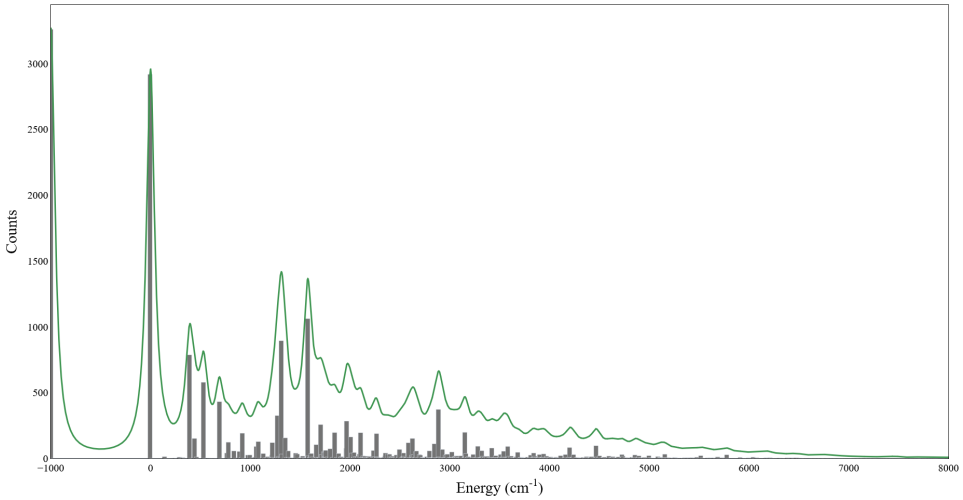


Figure A.1: Spectrum computed using my implementation for thymine up to class N=5. Note the value at -1000 represents the fact that the FCP is only 83.76% complete.

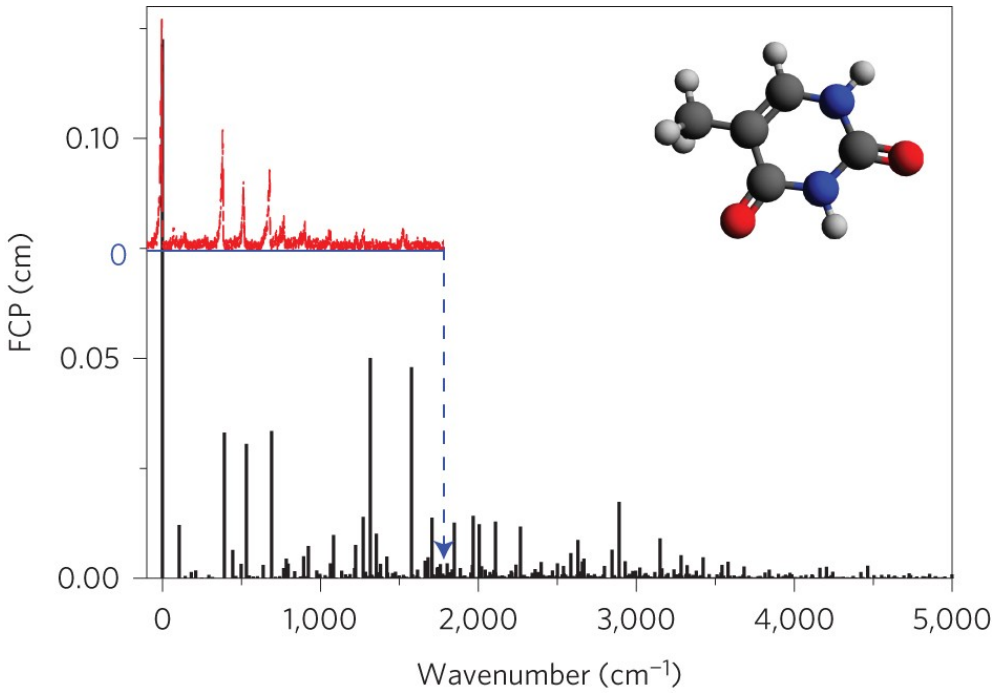


Figure A.2: Spectrum of thymine obtained from experimental data. Reproduced from [20]

Appendix B

Visualisations of Duschinsky Matrices

This chapter shows the visualisation of the Duschinsky matrix for all molecules considered. Note the previously described molecules with displacement vector $\mathbf{0}$ are not included as they have the same Duschinsky matrix as the corresponding molecules below. The Duschinsky matrix of diagonal blocks used in section 5.4 are also shown.

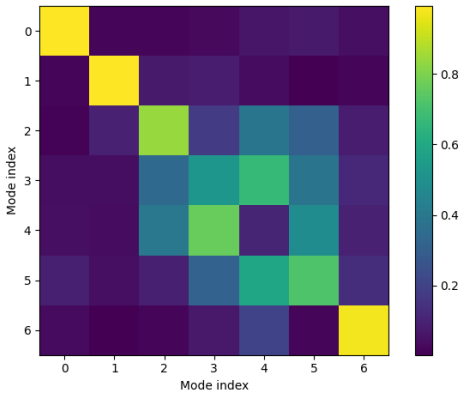


Figure B.1: Duschinsky matrix of formic acid

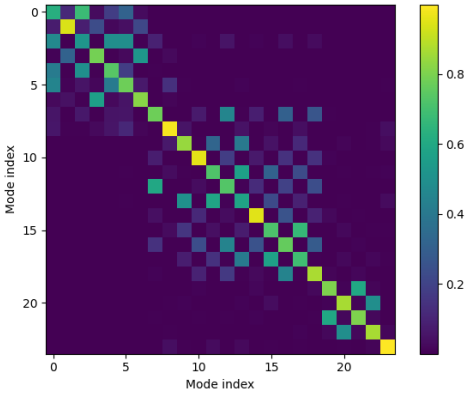


Figure B.2: Duschinsky matrix of pyrrole

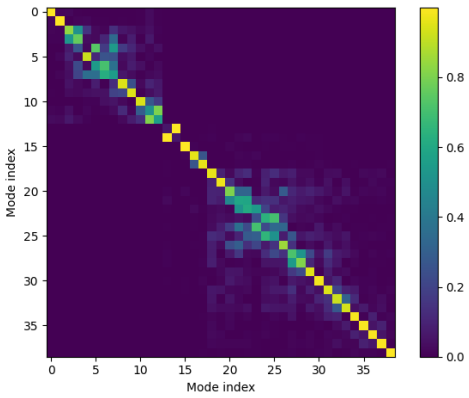


Figure B.3: Duschinsky matrix of thymine

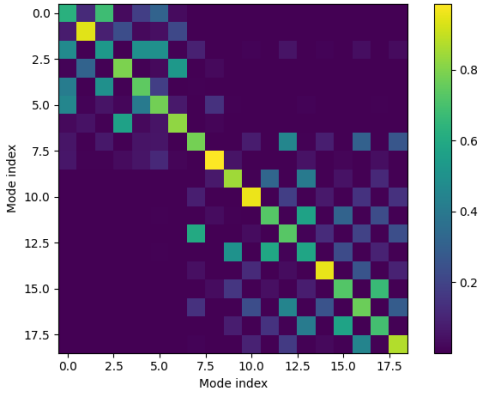


Figure B.4: Duschinsky matrix of larger diagonal block of pyrrole

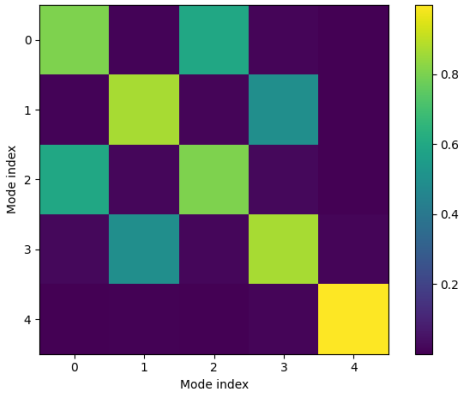


Figure B.5: Duschinsky matrix of smaller diagonal block of pyrrole

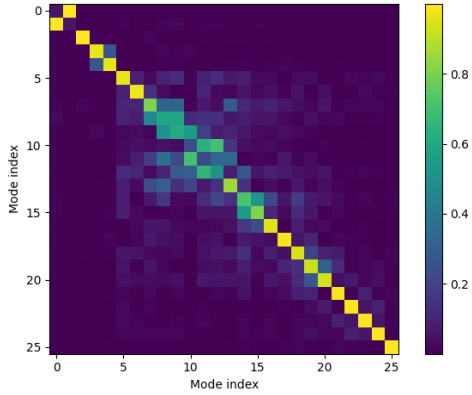


Figure B.6: Duschinsky matrix of larger diagonal block of thymine

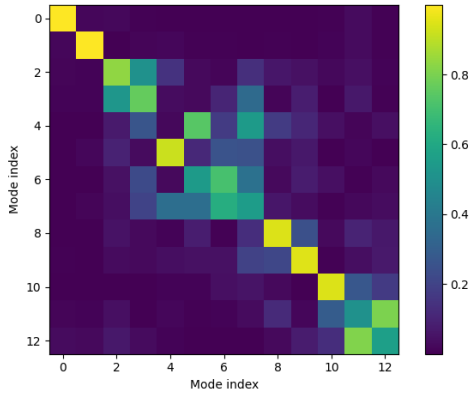


Figure B.7: Duschinsky matrix of smaller diagonal block of thymine

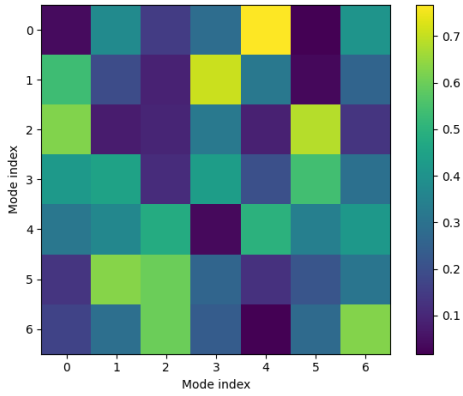


Figure B.8: Duschinsky matrix of randomly generated molecule number 0

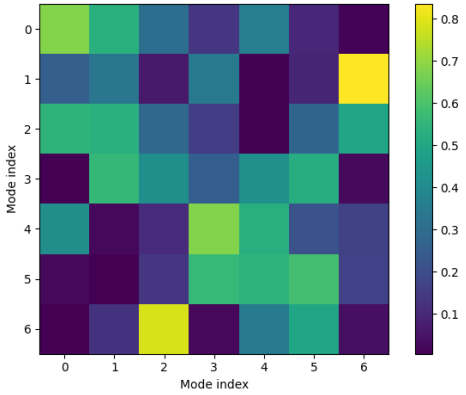


Figure B.9: Duschinsky matrix of randomly generated molecule number 1

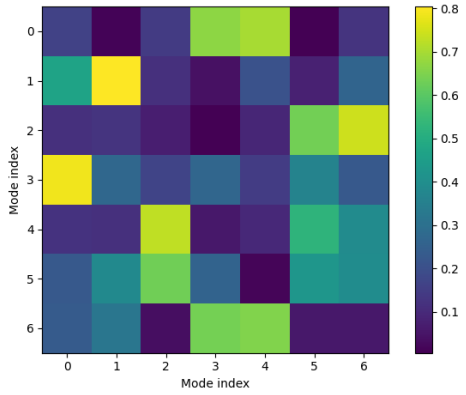


Figure B.10: Duschinsky matrix of randomly generated molecule number 2

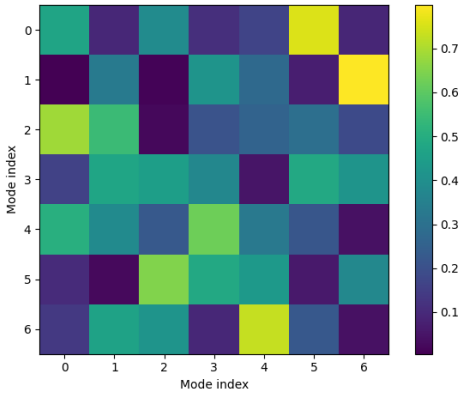


Figure B.11: Duschinsky matrix of randomly generated molecule number 3

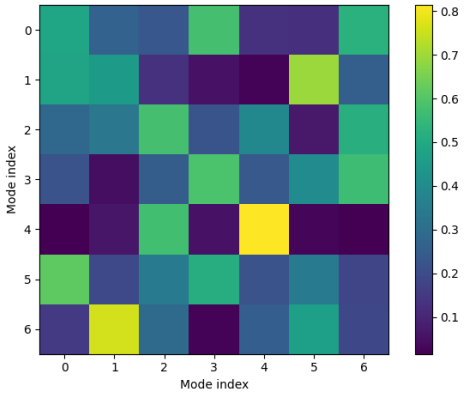


Figure B.12: Duschinsky matrix of randomly generated molecule number 4

Appendix C

Detailed results for generated molecules

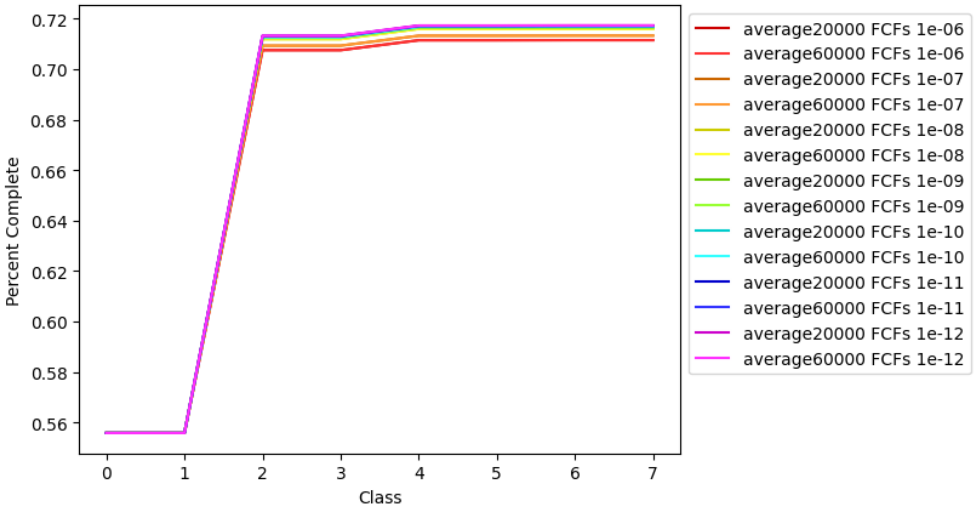


Figure C.1: Percentage contribution to the spectra at each class for varying cut-off values.

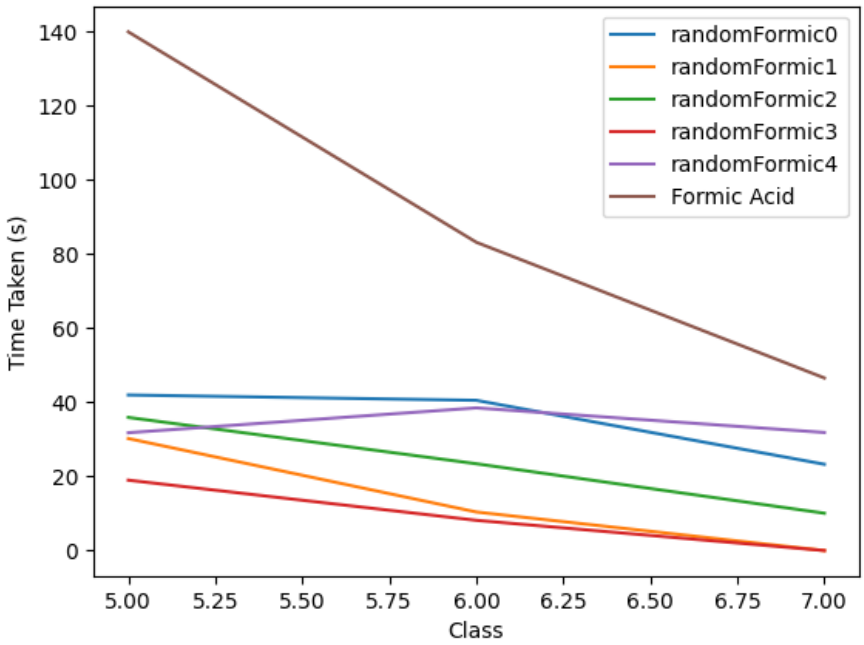


Figure C.2: Time taken for random molecules and formic acid to compute higher classes.

Appendix D

Full Spectral Data for Timing Analysis

Molecule	Total Time to Compute (s)	% Spectra Computed
Small Pyrrole Block	0.000	94.06
Formic Acid	1.047	85.05
Small Thymine Block	0.9851	83.88
Large Pyrrole Block	15.80	83.83
Pyrrole	232.9	83.77
Large Thymine Block	450.8	83.77
Thymine	53040	83.76

Table D.1: Calculation data for the timing analysis. Note the time of 0 seconds for the small pyrrole block is due to spectra being computed at a speed greater than my computer could specify.

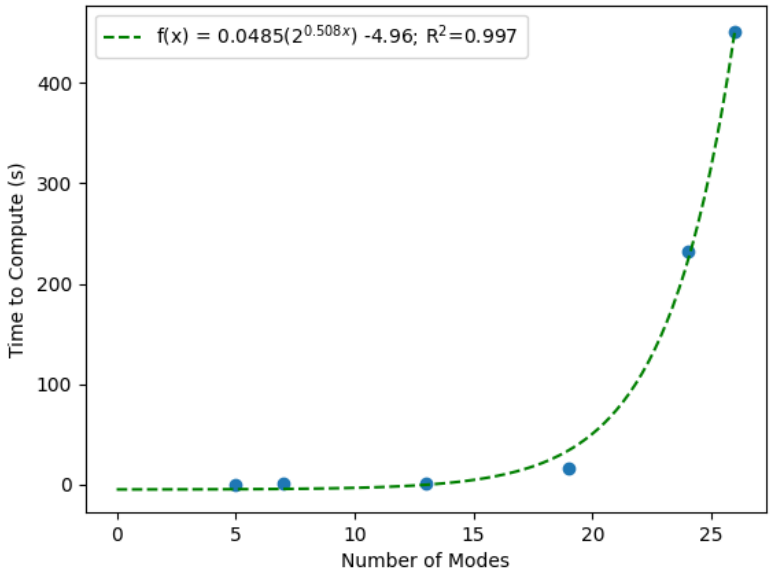


Figure D.1: Time taken to compute at least 83.76% of a spectra. The molecules looked at in size order are: the smaller diagonal block of pyrrole, formic acid, the smaller diagonal block of thymine, the larger diagonal block of pyrrole, pyrrole, and the larger diagonal block of thymine. For a discussion on what the smaller/larger blocks represent see section 5.4. The R^2 is the coefficient of determination.