Chapter 1

Mathematical Preliminaries

Several topics in this thesis require a background that not all researcher will have experience in. Especially as this thesis is multi-disciplinary, I would like to include at least some basic introduction to various physical and computer science topics. Additionally, several lemmas used in this manuscript might be of independent interest, as their applicability is not restricted to the various models studied in this thesis.

1.1 Mathematical notation

Perhaps the most simple point that I would like to raise before the thesis begins in ernest is the notation that I will use throughout the paper. Much of the paper uses notation not necessarily standard in every area of physics or computer science, and I want to make sure that no confusion occurs. I will assume that various notations that are common do not need to be described, such at $\mathcal{H}$ describing a Hilbert space, or that $\mathbb{I}$ describes a the identity operator on a particular Hilbert space.

The first such notation will be for the shorthand definition of sets of particular size. Namely,

$$[k] := \{0, 1, \cdots, k-1\}. \tag{1.1}$$

This is a set of size k, with the elements ordered and labeled by the integers from 0 to $k-1$. We will often think of these as elements from $\mathbb{Z}_k$, with addition and multiplication defined over the integers modulo k.

Often this paper will want to investigate systems with many particles, and we will want an operator to only act nontrivially on one particle. In particular, if we have a Hilbert space $\mathcal{H}_{\text{total}} = \mathcal{H}_{\text{single}}^{\otimes N}$ that consists of N copies of some single Hilbert space, and if we have an operator M that acts on $\mathcal{H}_{\text{single}}$, we can define an operator $M^{(w)}$ that acts nontrivially only on the w-th copy of $\mathcal{H}_{\text{single}}$, namely

$$M^{(w)} = \mathbb{I}^{\otimes w-1} \otimes M \otimes \mathbb{I}^{N-w}. \tag{1.2}$$

In this manner, only the w-th copy of $\mathcal{H}_{\text{single}}$ is effected.

As we will also be working with graphs, we will want to note that the letter G usually denotes a particular graph. Further, $A(G)$ describes the adjacency matrix of the graph G.

$V(G)$ then describes the vertex set of G, and $E(G)$ describes the edge set of G. Note that this thesis will always deal with undirected graphs, with at most a single edge between vertices. As such, the adjacency matrix $A(G)$ will be a symmetric 0-1 matrix. We will at times want to work with a simple graph, in which self-loops do not occur, but unless otherwise specified a graph G might contain self-loops.

Much of the work in this thesis, especially when describing the ground energy of particular Hamiltonian, deals only with positive semi-definite operators. As such, if A is a positive semi-definite matrix, then $\gamma(A)$ is the smallest non-zero eigenvalue. Note that if A has a 0-eigenvalue, then this corresponds to the energy gap between the ground state and the first excited state, but if A does not have a 0-eigenvalue then this is simply the smallest eigenvalue of A.

Finally, let us assume that A acts on a Hilbert space $\mathcal{H}$, and that $\mathcal{S}$ is a subspace of $\mathcal{H}$. We will then write the restriction of A to the subspace $\mathcal{S}$ as $A\big|_{\mathcal{S}}$.

Big O notation.

1.2 Quantum information

As this thesis is about quantum information, I will assume a familiarity with the basics

1.3 Indistinguishable particles

A foundational aspect of this thesis is understanding how defining the small scale interactions between multiple particles effects the large scale behavior of the system. As we want to define the large scale systems in terms of these small systems, the Hamiltonians that we will be interested in will have symmetries when we permute the different particles.

More concretely, when we restrict our Hamiltonians to the case with n particles, each particle will have the same (finite) Hilbert space to work with, and the Hamiltonian will be symmetric under a relabelling of the particles. Explicitly, we will have that each particle will have a Hilbert space $\mathcal{H}_{\text{ind}} = \mathbb{C}^d$ for some dimension d, so that the total Hilbert space of n particles is $\mathcal{H}_{\text{tot}} = \mathcal{H}_{\text{ind}}^{\otimes n} = \mathbb{C}^{dn}$, and the Hamiltonian is an operator on $\mathcal{H}_{\text{tot}}$ that commutes with the operators that swap the individual particle Hilbert spaces.

[**TO DO:** *Make this more coherent*]

The reason that we do this is the simple fact that particles are almost always indistinguishable. As such, any initial state for these multiparticle Hamiltonians will have the same symmetry as the underlying particles.

1.4 Complexity Theory

While this thesis is for the physics department, many of the results require some basic quantum complexity theory. In particular, the computer science idea for classification of computational problems in terms of the requisite resources gives a particularly nice interpretation of why certain physical systems don't equilibrate, and give a simple explanation on why certain systems do not have a known closed form solution.

This is a simple introduction, with a focus designed to make the rest of this thesis comprehensible to those without a background in complexity theory. For a more formal introduction to Complexity Theory, I would recommend [?], with a more in depth review found in [?]. For a focus on complexity as found in quantum information, I would recommend [?].

1.4.1 Languages and promise problems

The main foundation of computational complexity is in the classification of languages based on the requisited resources to determine whether some string is in a language. Unfortunately, this requires the definition of many of these terms.

In particular, what exactly is a string? Any person who has taken a basic programming class knows that a string is simply a word, but the mathematical definition is slightly more complicated. In particular, we first need to define an alphabet, and then define a string over a particular alphabet.

Definition 1 (Alphabet)**.** An alphabet is a finite collection of symbols.

Usually, an arbitrary alphabet is denoted by Γ, while the binary alphabet is denoted by $\Sigma = \{0, 1\}$. The chosen alphabet has no impact on a particular complexity result, as any finite alphabet can be represented via the binary alphabet with overhead that is logarithmic in the size of the original alphabet (essentially, just use a binary encoding of the new alphabet).

With this definition of an alphabet, a string is simply a finite sequence of elements from the alphabet. In particular, we define Γ^n to be all length n sequences of elements from Γ, and then define

$$\Gamma^* = \bigcup_{n=0}^{\infty} \Gamma^n. \tag{1.3}$$

With this, Γ^* is the set of all strings over Γ.

Computational complexity then deals with understanding subsets of these strings. In particular, let Π_{yes} be a subset of Γ^*. The language problem related to Π_{yes} is to determine whether a given string $x \in \Gamma^*$ is contained within Π_{yes} or not. This can be trivial, such as for the case of $\Pi_{\text{yes}} = \Gamma^*$, or it can be impossible, such as in the case of the famous Halting Problem.

Related to these language problems are promise problems, in which there are two subsets of Γ^*, namely Π_{yes} and Π_{no}, such that $\Pi_{\text{yes}} \cap \Pi_{\text{no}} = \emptyset$. We are then *promised* that the $x \in \Gamma^*$ that we need to sort is contained either $\Pi_{\text{yes}} \cup \Pi_{\text{no}}$. This generally opens up some more interesting problems, as without this restriction certain complexity classes do not make sense.

[**TO DO:** *revise and revisit*]

1.4.2 Turing machines

Up to this point, we have only discussed classifications of strings, and stated that we will want to understand the various resources required to sort a given string into one of two different sets, but we have not explained how these resources are defined. There are various ways to do this, depending on the various computational model one is interested in, but to give the most intuition we will need to define a Turing Machine. These machines are a mathematical construction that allow for the explicit definition of algorithms.

At their most basic level, a Turing machine is simply a finite program along with a (countably) infinite tape that allows the machine to store information. The input to the algorithm is initially written on the tape, and the machine starts in some initial configuration. The machine can only access it's internal memory along with a single character at a time from the infinite tape, and the program progresses by changing the internal state of the machine, changing one character on the tape, and moving along the tape. While extremely limited, these machines have so far captured our ideas of computation.

Formally, a Turing-machine is M is described by a tuple (Γ, Q, δ), where Γ is a finite set of symbol's that can be written on the infinite tape, Q is a set of possible internal states that M can store as internal memory, and δ is a function $Q \times \Gamma \to Q \times \Gamma \times \{L, S, R\}$ describing the required action of the machine M. Included in Q are two special "halting" states generally labeled `accept` and `reject`, such that the machine stops operating if it ever enters these two states, and the machine either accepts or rejects the current string. Note that we always assume that the alphabet contains a special character ␣that is not used for the input but denotes empty space along the infinite tape after the input string.

During an actual computation, a Turing Machine always starts with its internal state in a specified position, with the string used for input on the initial segment of the infinite tape and the special character ␣on every character after the input. Additionally, the pointer of the machine is located at the beginning of the tape, so that the machine is able to start reading the input (if needed). At each time step the machine then applies the transition function, updating its internal state, the character located at the current position of the tape, along with the current position of the tape until the machine reaches one of its halting states.

Note that there are several variations on these Turing Machines, such as those that have multiple infinite tapes instead of just one, and one that can move to an arbitrary position along the tape. These variations do not change the overall computational power of the model, just make it slightly more efficient. This definition is perhaps the most simple, and will suffice for now.

[**TO DO:** *get a picture of a TM here*]

One slight modification that will be useful for us is machines that compute a particular function. In particular, for a given function $f : \Gamma^* \to \Gamma^*$, we say that a Turing Machine M computes the function f if for all inputs x, the machine eventually halts and after it halts the tape will have $f(x)$ on the output tape (and nothing else).

1.4.2.1 Resources

With an explicit definition of Turing Machines, we also want to have some way to quantify the amount of resources used by a computation. Since each machine is expected to work on strings of arbitrary length, we somehow need to quantify the resources in terms of the input to a given string. So far, the important quantity in these resource problems has been the length of an input string x. Basically, the number of characters has been the interesting aspect to measure, since any machine will at least need to read the string.

With this n as the yardstick for any of our measurements, we then need to measure the length of the actual computation. In general, there are two ways to measure this length: the number of transitions that the computation used before it halted (as a measure of time), or else the number of elements of the tape that the machine visited during its computation (as a measure of space). It is important to realize that the exact value of these resources depend on the definitions used for the machine, such as the alphabet size or the number of internal states. As such, we will generally not be interested in the exact value for a given input, but will be more interested in the asymptotic scaling of the resources.

These requisite resources will generally be something of the form $\mathcal{O}(f(n))$ for some easily computable function f such as a polynomial or an exponential in n. These various scalings will give us a nice method of classifying the difficulty of computational problems. In general, we will say that a specific Turing Machine M runs in time $f(n)$ if for all inputs it halts in time $t(x)$ and $t(x) \in \mathcal{O}(f(n))$.

1.4.2.2 Uniform circuit families

While Turing Machines are sufficient for classical computation, when we want to describe some quantum complexity classes it will be useful to instead discuss quantum circuits. However, an important aspect of Turing Machines is that they are defined independently of the size of the input, while circuits need to have unique definitions for all different input sizes.

One might be tempted to simply define a computation via circuits by whether or not there exists a circuit of a given length, but this ends up giving an unreasonable amount of power to the computational model. In particular, the algorithm can hide computation in the definition of the circuit, as opposed to the actual running of the circuit itself.

To get around this, we will need to compute the circuit for the computation given the length. Namely, we will have a Turing Machine take as input the string length in unary, and the machine will output a description of the circuit. I won't go into the details here, but the

Definition 2 (Uniform family of circuits)**.** A collection $\{C_x : x \in S \subseteq \Sigma^*\}$ of circuits is a (polynomial-time) *uniform family of circuits* if there exists a deterministic Turing Machine M such that

- M runs in polynomial time.
- For all $x \in S$, M outputs a description of C_x.

Note that this definition makes no reference to the type of circuit, although we will generally assume that the circuit comes from some specific gate set.

[**TO DO:** *completely update this*]

1.4.3 Useful complexity classes

Once we have an understanding of what defines a relation, and how these are related, we can attempt to classify those languages that require different resources in order to solve.

1.4.3.1 Classical complexity classes

Perhaps the most well known question in computational complexity is the Pvs NPproblem. However, what exactly are these classes. At a most basic level, one can think of Pas those classification problems that have an efficient classical solution, while NPare those that can be checked in an efficient manner.

Definition 3 (P)**.** A promise problem $\mathcal{A} = (\mathcal{A}_{\text{yes}}, \mathcal{A}_{\text{no}})$ is in the class Pif there exists a polynomial-time Turing Machine M such that $M(x)$ accepts x if and only if $x \in \mathcal{A}_{\text{yes}}$.

Note that the Turing Machine M is required to halt on all inputs, and thus this is exactly what we mean by a polynomial-computation. Some simple examples of languages in ¶are [**TO DO:** *find* P*languages*]

Definition 4 (NP)**.** A promise problem $\mathcal{A} = (\mathcal{A}_{\text{yes}}, \mathcal{A}_{\text{no}})$ is in the class NPif there exists a polynomial q and a polynomial-time Turing Machine M such that

- if $x \in \mathcal{A}_{\text{yes}}$, then there exists a string $y \in \Sigma^{q(|x|)}$ such that $M(x, y)$ accepts.
- if $x \in \mathcal{A}_{\text{no}}$, then for all strings $y \in \Sigma^{(q(|x|)}$, $M(x, y)$ rejects.

Essentially, a language is in NPif a given string can be proven to be in the language. This includes useful problems such as whether a given graph has a 3-coloring, whether an integer p has at least k prime factors, and all problems in ¶.

1.4.3.2 Bounded-Error Quantum Polynomial Time

With these classical problems now defined, we will want to understand what happens when we include quantum mechanics. There is a way to define a quantum Turing machine, in an analog to the classical case, but the current state of the art has instead gone toward using quantum circuits instead.

Intuitively, the idea behind Bounded-Error Quantum Polynomial Time (BQP) consists of those problems that can be solved by a quantum computer efficiently. However, we need to somehow encode the circuit

Definition 5 (BQP)**.** A promise problem $\mathcal{A} = (\mathcal{A}_{\text{yes}}, \mathcal{A}_{\text{no}})$ if there exist a uniform family of quantum circuits $Q = \{Q_n : n \in \mathbb{N}\}$ such that

- If $x \in \mathcal{A}_{\text{yes}}$, then $Q_{|x|}(|x\rangle) \geq \frac{2}{3}$.
- If $x \in \mathcal{A}_{\text{no}}$, then $Q_{|x|}(|x\rangle) \leq \frac{1}{3}$.

Note that these t

1.4.3.3 Quantum Merlin-Arthur

In addition to having an understanding of when a quantum computer can solve a particular problem, we will also want an understanding of those problems that most likely cannot be

Definition 6 (QMA)**.** A promise problem $\mathcal{A} = (\mathcal{A}_{\text{yes}}, \mathcal{A}_{\text{no}})$ if there exists a uniform family of quantum circuits $Q = \{Q_n : n \in \mathbb{N}\}$ such that

- If $x \in \mathcal{A}_{\text{yes}}$, then there exists a state $|\,\psi\,\rangle \in \mathbb{C}^{p(|x|)}$ such that $Q_{|x|}(|\,x\,\rangle, |\,\psi\,\rangle) \geq \frac{2}{3}$.
- If $x \in \mathcal{A}_{\text{no}}$, then for all states $|\,\psi\,\rangle \in \mathbb{C}^{p(|x|)}$, $Q_{|x|}(|\,x\,\rangle, |\,\psi\,\rangle) \leq \frac{1}{3}$.

Intuitively, this is like the class NPin that the circuit only accepts if there exists a proof that the input is in the language. This proof might be extremely difficult to construct, but it still exists.

1.4.3.4 Reductions and Complete Problems

While we are interested in these complexity classes, it is often difficult to work with the exact definitions used. As an example, in the definition of NP, to show something for all of the class we would somehow need to encode the entire computation of the Turing machine in our proof. One way to get around this is via reductions.

Essentially a reduction is a polynomial-time computable function from one computational problem to another. Because this reduction is easy to compute, if we can solve the second problem, then we can also solve the first problem. More concretely, let $\mathcal{A} = (\mathcal{A}_{\text{yes}}, \mathcal{A}_{\text{no}}$ and $\mathcal{B} = (\mathcal{B}_{\text{yes}}, \mathcal{B}_{\text{no}})$ be two promise problems. We say that there

1.5 Various Mathematical Lemmas

In addition to these various complexity results, it will also be useful to have a list of certain mathematical lemmas that will be used several times in the thesis. These lemmas might also be of independent interest.

1.5.1 Truncation Lemma

Perhaps the first such lemma we called the truncation lemma. The idea behind this lemma is to approximate the evolution of a state under some particular Hamiltonian with another, where the differences between the two Hamiltonians only occur far from the support of the given state. One would expect that since the state must evolve “far” in order to reach the differs between the two Hamiltonians, the evolution between the two will be close. This lemma makes this intuition precise.

Lemma 1 (Truncation Lemma)**.** *Let H be a Hamiltonian acting on a Hilbertspace $\mathcal{H}$ and let $|\,\Phi\,\rangle \in \mathcal{H}$ be a normalized state. Let $\mathcal{K}$ be a subspace of $\mathcal{H}$, let P be the projector onto $\mathcal{K}$, and let $\tilde{H} = PHP$ be the Hamiltonian within this subspace. Suppose that, for some $T > 0$, $W \in \{H, \tilde{H}\}$, $N_0 \in \natural$, and $\delta > 0$, we have, for all $0 \leq t \leq T$,*

$$e^{-iWt}|\Phi\rangle = |\gamma(t)\rangle + |\epsilon(t)\rangle \textit{ with } \||\epsilon(t)\rangle\| \leq \delta$$

and

$$(1-P)H^r|\gamma(t)\rangle = 0 \textit{ for all } r \in \{0, 1, \ldots, N_0 - 1\}.$$

Then, for all $0 \leq t \leq T$,

$$\left\| \left(e^{-iHt} - e^{-i\tilde{H}t} \right) |\Phi\rangle \right\| \leq \left(\frac{4e\|H\|t}{N_0} + 2 \right) \left(\delta + 2^{-N_0}(1+\delta) \right).$$

This lemma actually combines two different methods. The first assumes that the

Proposition 1. *Let* H *be a Hamiltonian acting on a Hilbert space* $\mathcal{H}$*, and let* $|\,\Phi\,\rangle \in \mathcal{H}$ *be a normalized state. Let* $\mathcal{K}$ *be a subspace of* $\mathcal{H}$ *such that there exists an* $N_0 \in \natural$ *so that for all* $|\,\alpha\,\rangle \in \mathcal{K}^{\perp}$ *and for all* $n \in \{0, 1, 2, \ldots, N_0 - 1\}$, $\langle\,\alpha\,|H^n|\,\Phi\,\rangle = 0$*. Let* P *be the projector onto* $\mathcal{K}$ *and let* $\tilde{H} = PHP$ *be the Hamiltonian within this subspace. Then*

$$\|e^{-it\tilde{H}}|\,\Phi\,\rangle - e^{-itH}|\,\Phi\,\rangle\| \leq 2\left(\frac{e\|H\|t}{N_0}\right)^{N_0}.$$

Proof. Define $|\,\Phi(t)\,\rangle$ and $|\,\tilde{\Phi}(t)\,\rangle$ as

$$|\,\Phi(t)\,\rangle = e^{-itH}|\,\Phi\,\rangle = \sum_{k=0}^{\infty} \frac{(-it)^k}{k!} H^k |\,\Phi\,\rangle \qquad |\,\tilde{\Phi}(t)\,\rangle = e^{-it\tilde{H}}|\,\Phi\,\rangle = \sum_{k=0}^{\infty} \frac{(-it)^k}{k!} \tilde{H}^k |\,\Phi\,\rangle.$$

Note that by assumption, $\tilde{H}^k|\,\Phi\,\rangle = H^k|\,\Phi\,\rangle$ for all $k < N_0$, and thus the first N_0 terms in the two above sums are equal. Looking at the difference between these two states, we have

$$\begin{aligned}
\||\,\Phi(t)\,\rangle - |\,\tilde{\Phi}(t)\,\rangle\| &= \left\| \sum_{k=0}^{\infty} \frac{(-it)^k}{k!} \left(H^k - \tilde{H}^k \right) |\,\Phi\,\rangle \right\| \\
&= \left\| \sum_{k=0}^{N_0-1} \frac{(-it)^k}{k!} \left(H^k - \tilde{H}^k \right) |\,\Phi\,\rangle - \sum_{k=N_0}^{\infty} \frac{(-it)^k}{k!} \left(H^k - \tilde{H}^k \right) |\,\Phi\,\rangle \right\| \\
&\leq \sum_{k=N_0}^{\infty} \frac{t^k}{k!} \left(\|H\|^k + \|\tilde{H}\|^k \right) \\
&\leq 2 \sum_{k=N_0}^{\infty} \frac{t^k}{k!} \|H\|^k
\end{aligned}$$

where the last step uses the fact that $\|\tilde{H}\| \leq \|P\|\|H\|\|P\| = \|H\|$. Thus for any $c \geq 1$, we have

$$\begin{aligned}
\||\,\Phi(t)\,\rangle - |\,\tilde{\Phi}(t)\,\rangle\| &\leq \frac{2}{c^{N_0}} \sum_{k=N_0}^{\infty} \frac{(ct)^k}{k!} \|H\|^k \\
&\leq \frac{2}{c^{N_0}} \exp(ct\|H\|).
\end{aligned}$$

We obtain the best bound by choosing $c = N_0/\|Ht\|$, which gives

$$\||\,\Phi(t)\,\rangle - |\,\tilde{\Phi}(t)\,\rangle\| \leq 2\left(\frac{e\|H\|t}{N_0}\right)^{N_0}$$

as claimed. (If $c < 1$ then the bound is trivial.) □

Proposition 2. *Let $U_1, \ldots, U_n$ and $V_1, \ldots, V_n$ be unitary operators. Then for any $|\,\psi\,\rangle$,*

$$\left\| \left(\prod_{i=n}^{1} U_i - \prod_{i=n}^{1} V_i \right) |\,\psi\,\rangle \right\| \leq \sum_{j=1}^{n} \left\| (U_j - V_j) \prod_{i=j-1}^{1} U_i |\,\psi\,\rangle \right\|. \tag{1.4}$$

Proof. The proof is by induction on n. The case $n = 1$ is obvious. For the induction step, we have

$$\begin{aligned} \left\| \left(\prod_{i=n}^{1} U_i - \prod_{i=n}^{1} V_i \right) |\,\psi\,\rangle \right\| &= \left\| \left(\prod_{i=n}^{1} U_i - V_n \prod_{i=n-1}^{1} U_i + V_n \prod_{i=n-1}^{1} U_i - \prod_{i=n}^{1} V_i \right) |\,\psi\,\rangle \right\| && (1.5) \\ &\leq \left\| (U_n - V_n) \prod_{i=n-1}^{1} U_i |\,\psi\,\rangle \right\| + \left\| \left(\prod_{i=n-1}^{1} U_i - \prod_{i=n-1}^{1} V_i \right) |\,\psi\,\rangle \right\| && (1.6) \\ &\leq \sum_{j=1}^{n} \left\| (U_j - V_j) \prod_{i=j-1}^{1} U_i |\,\psi\,\rangle \right\| && (1.7) \end{aligned}$$

where the last step uses the induction hypothesis. □

Proof of Lemma **??**. For $M \in \natural$ write

$$\begin{aligned} \|(e^{-iHt} - e^{-i\tilde{H}t})|\,\Phi\,\rangle\| &= \left\| \left(\left(e^{-iH\frac{t}{M}}\right)^M - \left(e^{-i\tilde{H}\frac{t}{M}}\right)^M \right) |\,\Phi\,\rangle \right\| \\ &\leq \sum_{j=1}^{M} \left\| \left(e^{-iH\frac{t}{M}} - e^{-i\tilde{H}\frac{t}{M}} \right) e^{-iW(j-1)\frac{t}{M}} |\,\Phi\,\rangle \right\| \\ &\leq \sum_{j=1}^{M} \left\| \left(e^{-iH\frac{t}{M}} - e^{-i\tilde{H}\frac{t}{M}} \right) \left(|\,\gamma(\tfrac{(j-1)t}{M})\,\rangle + |\,\epsilon(\tfrac{(j-1)t}{M})\,\rangle \right) \right\| \\ &\leq 2M\delta + \sum_{j=1}^{M} \left\| \left(e^{-iH\frac{t}{M}} - e^{-i\tilde{H}\frac{t}{M}} \right) \frac{|\,\gamma(\frac{(j-1)t}{M})\,\rangle}{\left\| |\,\gamma(\frac{(j-1)t}{M})\,\rangle \right\|} \right\| \left\| |\,\gamma(\tfrac{(j-1)t}{M})\,\rangle \right\| \\ &\leq 2M\delta + 2M \left(\frac{e\|H\|t}{MN_0} \right)^{N_0} (1 + \delta) \end{aligned}$$

where in the second line we have used Proposition **??** and in the last step we have used Proposition **??** and the fact that $\||\gamma(t)\rangle\| \leq 1 + \delta$. Now, for some $\eta > 1$, choose

$$M = \left\lceil \frac{\eta e \|H\| t}{N_0} \right\rceil$$

for $0 < t \leq T$ to get

$$\begin{aligned} \|(e^{-iHt} - e^{-i\tilde{H}t})|\,\Phi\,\rangle\| &\leq 2M \left(\delta + \eta^{-N_0}(1 + \delta) \right) \\ &\leq 2 \left(\frac{\eta e \|H\| t}{N_0} + 1 \right) \left(\delta + \eta^{-N_0}(1 + \delta) \right). \end{aligned}$$

The choice $\eta = 2$ gives the stated conclusion. □

Note that it would be slightly better to take a smaller value of η. However, this does not significantly improve the final result; the above bound is simpler and sufficient for our purposes.

1.5.2 Nullspace Projection Lemma

When we discuss the ground spaces and ground energies of various Hamiltonians, we will often want to know what happens to the ground spaces and ground energies when two such Hamiltonians are added together (such as adding penalties enforcing particular initial states). As such, the Nullspace Projection Lemma exactly discusses how such systems add together. As far as I am aware this lemma was initially used (implicitly) by Mizel et. al. [**TO DO:** *find correct reference*] We then used this in our proof of the QMA-completeness for the Bose-Hubbard model. We then found an additional place that used a similar lemma, with slightly better bounds. While the improvement is minor, here is a proof of the improved bound (and note that the improvement was left as a proof for the reader in the newer result).

Lemma 2 (Nullspace Projection Lemma)**.** *Let H_A and H_B be positive semi-definite matrices. Suppose that the nullspace, S, of H_A is nonempty, and that*

$$\gamma\big(H_B|_S\big) \geq c > 0 \qquad \textit{and} \qquad \gamma(H_A) \geq d > 0. \tag{1.8}$$

Then,

$$\gamma(H_A + H_B) \geq \frac{cd}{d + \|H_B\|}. \tag{1.9}$$

Proof. Let $|\psi\rangle$ be a normalized state satisfying

$$\langle\psi|H_A + H_B|\psi\rangle = \gamma(H_A + H_B). \tag{1.10}$$

Let Π_S be the projector onto the nullspace of H_A. First suppose that $\Pi_S|\psi\rangle = 0$, in which case

$$\langle\psi|H_A + H_B|\psi\rangle \geq \langle\psi|H_A|\psi\rangle \geq \gamma(H_A) \tag{1.11}$$

and the result follows. On the other hand, if $\Pi_S|\psi\rangle \neq 0$ then we can write

$$|\psi\rangle = \alpha|a\rangle + \beta|a^\perp\rangle \tag{1.12}$$

with $|\alpha|^2 + |\beta|^2 = 1$, $\alpha \neq 0$, and two normalized states $|a\rangle$ and $|a^\perp\rangle$ such that $|a\rangle \in S$ and $|a^\perp\rangle \in S^\perp$. (If $\beta = 0$ then we may choose $|a^\perp\rangle$ to be an arbitrary state in $S^\perp$ but in the following we fix one specific choice for concreteness.) Note that any state $|\phi\rangle$ in the nullspace of $H_A + H_B$ satisfies $H_A|\phi\rangle = 0$ and hence $\langle\phi|a^\perp\rangle = 0$. Since $\langle\phi|\psi\rangle = 0$ and $\alpha \neq 0$ we also see that $\langle\phi|a\rangle = 0$. Hence any state

$$|f(q,r)\rangle = q|a\rangle + r|a^\perp\rangle \tag{1.13}$$

is orthogonal to the nullspace of $H_A + H_B$, and

$$\gamma(H_A + H_B) = \min_{|q|^2+|r|^2=1} \langle f(q,r)|H_A + H_B|f(q,r)\rangle. \tag{1.14}$$

Within the subspace Q spanned by $|\,a\,\rangle$ and $|\,a^\perp\,\rangle$, note that

$$H_A|_Q = \begin{pmatrix} w & v^* \\ v & z \end{pmatrix} \qquad H_B|_Q = \begin{pmatrix} 0 & 0 \\ 0 & y \end{pmatrix} \tag{1.15}$$

where $w = \langle a|H_B|a\rangle$, $v = \langle a^\perp|H_B|a\rangle$, $y = \langle a^\perp|H_A|a^\perp\rangle$, and $z = \langle a^\perp|H_B|a^\perp\rangle$, and that we are interested in the smaller eigenvalue of

$$M = H_A|_Q + H_B|_Q = \begin{pmatrix} w & v^* \\ v & y+z \end{pmatrix}. \tag{1.16}$$

Letting ϵ_+ and ϵ_- be the two eigenvalues of M with $\epsilon_+ \geq \epsilon_-$, note that

$$\epsilon_+ = \|M\| \leq \|H_A|_Q\| + \|H_B|_Q\| \leq y + \|H_B|_Q\| \leq y + \|H_B\|, \tag{1.17}$$

where we have used the Cauchy interlacing theorem to note that $\|H_B|_Q\| \leq \|H_B\|$. Additionally, we have that

$$\epsilon_+\epsilon_- = \det(M) = w(y+z) - |v|^2 \geq wy \tag{1.18}$$

where we used the fact that $H_B|_Q$ is positive-semidefinite. Putting this together, we have that

$$\gamma(H_A + H_B) = \min_{|q|^2+|r|^2=1} \langle f(q,r)|H_A + H_B|f(q,r)\rangle = \epsilon_- \geq \frac{wy}{y + \|H_B\|}. \tag{1.19}$$

As the right hand side increased monotonically with both w and y, and as $w \geq \gamma(H_B|_S) \geq c$ and $y \geq \gamma(H_A) \geq d$, we have

$$\gamma(H_A + H_B) \geq \frac{cd}{d + \|H_B\|} \tag{1.20}$$

as required. □

References

[1] Sanjeev Arora and Boaz Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.

[2] Michael Sipser, *Introduction to the Theory of Computation*, 2 ed., Thomas Course Technology, 2006.

[3] John Watrous, *Quantum computational complexity*, Encyclopedia of Complexity and System Science, Springer, 2009, arXiv:0804.3401.