Refining quantum algorithms for each era of quantum computing

Richard Meister

St Peter's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2023

ABSTRACT

Many important physical and mathematical problems do not allow analytical treatment, and are difficult to solve numerically on classical computers. Quantum computing is anticipated to perform vastly better for some of these problems. Examples include simulating other quantum mechanical systems like molecules, some optimisation tasks (e.g. combinatorial optimisation and data fitting), and number-theoretical problems like prime factorisation. Being able to efficiently find answers to these questions would enable the accelerated development in other fields of science, like chemistry and engineering.

This work uses analytical and numerical methods to tackle several sub-problems in the field of quantum computing, and provides tools and algorithms that allow more efficient utilisation of the (anticipated) hardware capabilities. The considered problems span a wide range of possible quantum device capacities, from their classical simulation, via near-term intermediate scale (NISQ) and early fault-tolerant hardware, all the way to fully error corrected platforms. Covered topics include an exploration of the problem to automatically generate an efficient implementation of any arbitrary quantum algorithm using the available resources, more accurate techniques for simulating electrons in molecules, a method for extracting information about energy levels in a system from minimal data, and an effort to prepare defined states in a controlled manner. The challenge of modelling perfect or noisy quantum computers themselves using conventional computers is also addressed through the development of an easy-to-use interface to a powerful quantum emulator.

Each one of the discussed contributions represents an advance of the theoretical capabilities towards the goal of utilising quantum hardware — which is rapidly being developed alongside theoretical efforts — to its full potential.

Contents

LIST OF FIGURES

List of Tables

If you don't know where you are going, any road will get you there.
Paraphrased exchange between Alice and the Cheshire Cat.

— Lewis Carroll's *Alice in Wonderland*

1 INTRODUCTION

Contents

In physics, many interesting problems lie within the realm of quantum mechanics. Any sufficiently small system must be treated according to the its laws to make accurate predictions about its behaviour. Examples cover the description of electrons in atoms, molecules, and crystals, explaining their various properties like geometry, spectra, magnetism, etc., as well as scattering problems involving electromagnetic waves and particles, and even models for gauge theories, exploring the fundamentals of our universe.

By its very nature, quantum mechanics operates in a Hilbert space, whose dimensionality grows very quickly with the number of particles considered. Although some problems can be solved analytically, and a wide range of clever approximate techniques exists, many systems remain very challenging. One way to deal with those is to numerically store their state on a classical computer and calculate properties numerically. Unfortunately, due to the size of the Hilbert space growing exponentially with the number of particles, such numerical treatment quickly meets its limits.

One approach how to simulate large systems is to not employ classical computers, but instead simulate one quantum mechanical (QM) system with another. After all, its Hilbert space size grows just as quickly as that of the system we are trying to simulate. This idea gave rise to the field of *quantum computing*. While its first concepts were concerned mainly with the just described simulation of one QM system with another — and this is still a very highly anticipated use case for the research efforts today — more algorithms emerged that could harvest the quantum mechanical properties of such a device to solve some problems more efficiently than classical computers could. Well-known examples include prime factorisation, (approximate) optimisation problems, and even searching in a list.

In this chapter, I will give a brief overview of the history and background of quantum computing, and introduce some notation. The following chapters will then discuss the state of the art regarding some open problems in the field, and present novel contributions to the subfields of quantum compiling, Hamiltonian time evolution, state spectroscopy, and Hamiltonian eigenstate preparation. Finally, I will concisely describe a tool I developed during my doctorate, which provides easier access to classically simulated quantum computers.

1.1 A brief history

This section gives a short overview of the history of quantum computing to put the rest of the present thesis into historical context. In the interest of brevity, it will only cover the most important milestones.

The first description of using one QM system to simulate another is often credited to a keynote speech given by Richard Feynman in 1981 at MIT titled "Simulating Physics with Computers" [1]. However, several physicists had already worked on precursors to quantum computing, most notably Paul Benioff in a paper published in 1980 [2], which he later expanded on in 1982 [3].

The subsequent important milestone was David Deutsch giving a formal description of a universal quantum computer in 1985 [4], which could simulate any physical system. He speculated that, if implemented, such a machine would be able to perform calculations faster than a classical Turing machine, but no actual examples were known at the time. Several years later, in 1992, Deutsch and Jozsa [5] provided the first instances of problems which can be solved exponentially faster on a quantum computer via the Deutsch–Jozsa algorithm.[1] This was an important result, proving the existence of problems which could be solved significantly (and asymptotically) quicker on a quantum device.

Two years later, in 1994, Peter Shor presented Shor's algorithm [6] for efficient integer factorisation using quantum computers. The procedure is still very relevant today and commonly cited as one of the main prospective applications of quantum hardware. In contrast to the result by Deutsch and Jozsa, Shor's algorithm has an important practical use. Amongst others, the widely used RSA cryptosystem relies on the assumption that factoring large numbers is hard, so it could in theory be broken with a high-performance quantum computer running Shor's algorithm, making its development a very significant milestone.

Only one year later, in 1995, Shor made another important contribution [7]. He described the first error correcting code to combat decoherence on quantum devices. External disturbances to the state of a controlled system plague quantum hardware to this day, and will likely continue to do so for the foreseeable future. Therefore, error correction is a crucial subfield in quantum computing, and any large-scale implementation will be using this concept, making its first introduction an important event. Section 1.3.1 will go into some more detail about the implications of noisy hardware and the ability to correct its errors.

In 1996, one more important quantum algorithm was published, this time by Grover [8]. He showed that — given a quantum subroutine that marks a specific state with a negative sign efficiently — quantum computers can find and prepare such a state quadratically faster than any classical approach. This is due to classical methods

[1]The Deutsch–Jozsa algorithm is a generalisation of Deutsch's algorithm, which operates only on a single bit and thus did not provide an exponential speedup.

having to check each element individually, resulting in a cost linear in the number of possible elements. Conversely, the more efficient quantum variant uses a technique based on amplitude amplification that became known as Grover's algorithm.

The following year, Kitaev [9] suggested using topological features to protect quantum states against decoherence. Variations of this proposal are currently amongst the candidates for the first fully error corrected quantum computers [10].

Several other important contributions to algorithm design were made in the following years. Some examples include an algorithm to solve linear systems of equations by Harrow, Nassidim, and Lloyd (HHL) [11] in 2008, the first proposal of variational methods by Peruzzo et al. [12] in 2014, as well as mechanisms for error mitigation by Li and Benjamin [13] and Temme et al. [14] in 2016.

Of course, all of these concepts need appropriate hardware to run on, which is a major experimental challenge. The devices used for such calculations typically must be extremely cold and very well decoupled from the rest of the universe to minimise disruptions to its state. Additionally, they must be controllable with high precision to make the results accurate. After a period of many important theoretical advances in the early 1990s, experimentalists have made considerable progress in developing such hardware. Some of the most important demonstrations are the first 2-qubit NMR quantum computer solving Deutsch's problem by Jones and Mosca [15] in 1998, the earliest quantum annealing demonstration by Brooke et al. [16], the first usage of a superconducting circuit as a qubit by Nakamura et al. [17], both in 1999, the successful factorisation of $15 = 3 \times 5$ using Shor's algorithm by Vandersypen et al. [18] in 2001, and the implementation of the Deutsch–Jozsa algorithm on an ion trap quantum computer by Gulde et al. [19] in 2003.

Since then, countless improvements have been made to all of these platforms. A significant announcement was made by Arute et al. [20] at Google in 2019, claiming to have reached quantum supremacy on a superconducting quantum processor executing random circuits. The term *quantum supremacy* was popularised by John Preskill [21] to characterise quantum devices which can solve problems that are not computable in a reasonable amount of time by classical processors. Though the research team of Ref. [20] claimed that executing the circuit they used would take about 10 000 years on today's most powerful supercomputers, this assertion was disputed by Pednault et al. [22], who argue it would be simulable in only around 2.5 days. In either case, the demonstration was solely intended as a proof-of-concept, and the executed circuit has no real practical application besides showcasing the increasing capabilities of quantum hardware.

Another important investigation in this context was published by Kim et al. [23], who argue to have found evidence for the utility of quantum computing before fault tolerance by simulating a two-dimensional Ising model. Despite not directly claiming to have demonstrated quantum supremacy, the publication received considerable pushback from researchers arguing that the system in question can be quite quickly solved using advanced classical simulation techniques [24, 25]. It seems likely that this back-and-forth between authors reporting solutions to complicated tasks on quantum hardware, and the response from other community members arguing that a solution is feasible on classical hardware, will continue for some time, until an indisputable and clear quantum supremacy result is reported. When this might happen depends on both experimental and theoretical advances in the field.

1.2 Digital quantum computers and formalism

Historically, there have been two fundamentally different approaches to quantum computing. The idea described by Feynman [1] relates to the concept of *analogue quantum computing*, more precisely *analogue quantum simulation*, while later developments introduced the concept of *digital quantum computing*. In this section, I want to briefly motivate analogue quantum computing, before describing the digital version in more detail, as the rest of this thesis will only be concerned with the latter.

In some cases, a direct mapping of a problem to a quantum mechanical system can be found. This also includes instances where the problem itself is the simulation of another quantum mechanical system. In analogue quantum computers, such a mapping is exploited by carefully manipulating the stand-in setup realised in a laboratory, which represents the system in question. This, in turn, allows the investigation of properties of the original problem. Besides analogue simulation, quantum annealing and adiabatic quantum computation also fall into this category of analogue quantum computing.

In contrast to the direct mapping onto a controllable quantum system in the analogue case, digital quantum computers encode information about the system of interest in a more abstract form. This type of quantum computer is closer to the description by Deutsch [4] in 1985. He proposed to consider a collection of N two-level quantum systems, today called *qubits*, to perform calculations. These qubits are usually called $|0\rangle$ and $|1\rangle$, using the well-established *bra–ket notation* [26] for quantum states. Following the standard formalism in quantum mechanics, a single qubit can exist in a superposition of $|0\rangle$ and $|1\rangle$, as in $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. The Hilbert space of N such qubits is then 2^N-dimensional, and consists of all possible combinations of zeros and ones on every qubit. A collection of qubits is often referred to as a *quantum register*.

Changes to the state in a register are induced by unitary operators U. Deutsch showed that with this setup of qubits and unitary transformations, a quantum computer is universal, meaning it can simulate every finite physical system with arbitrarily high precision, which automatically makes it Turing complete. Furthermore, he argues that because U acts on all N qubits at once, it is to be expected that quantum computers could exceed the speed of Turing-type machines.

Today, U is usually not described as a single, large matrix acting on all qubits at once, but instead is prescribed as a series of *gates*, analogous to logic gates like NOT and XOR in classical computing, each acting on only one or a few qubits at once. A collection of such gates, for which it makes sense to group them together logically, is usually called a *quantum circuit*. Graphically, they are often depicted as lines (representing the qubits) with boxes on them (meaning the application of a gate). Examples of such circuit diagrams are plentiful throughout this thesis.

When working with circuits and gates, it is important to keep in mind that they are already an abstraction of the physics of any specific quantum hardware, which implements them as a series of actual physical operations, like laser pulses or the generation of electric fields. The consequences of this are discussed in more detail in the Literature Review.

There are several established standard sets of gates used in the field of quantum computing, and the decision for one or the other often depends on the context. This topic is more closely inspected in Section 2.1. Irrespective of what specific set is used, each gate can always be described as a particular unitary matrix of dimension 2^N, where N is the number of qubits it acts on simultaneously. Some typical, often-used examples are the Pauli gates

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

arbitrary single-qubit rotations

$$R^x(\theta) = e^{-i\theta\sigma^x/2} \qquad R^y(\theta) = e^{-i\theta\sigma^y/2} \qquad R^z(\theta) = e^{-i\theta\sigma^z/2},$$

and Hadamard (H), phase (S), and T-gates

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

Common two-qubit gates are the controlled NOT (CNOT), which acts on the second qubit like a Pauli-x gate if and only if the first qubit is in the $|1\rangle$-state, and the SWAP gate, which exchanges the states of the two qubits. Their matrix representations are

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

There is a wide range of other possible gates used in various contexts, including controlled and multi-controlled versions of the ones listed above. The given examples are solely to give some intuition about what quantum gates typically look like.

With this notion of gates making up a circuit to produce a unitary operator U, an important question to ask which set of gates is sufficient to create *any* U, either exactly, or up to some error ε. A gate set capable of this is then called *universal.* Section 2.1 lists some popular choices and discusses the context in which they are most likely to appear. Universal gate sets can contain parameterised gates, like the Pauli rotations, or have only a finite number of fixed gates in them. It would be reasonable to expect that parameterised gates span a larger space than their non-parameterised counterparts, but the Solovay-Kitaev theorem [27] guarantees that even with a discrete set, any unitary U can be approximated to within an error of ε with at most $\mathcal{O}(\log^c(1/\varepsilon))$ gates. The constant c of an optimal algorithm is as of yet unknown, but can be bounded by $1 \le c \le \frac{\log 5}{\log 3/2} \approx 4$, and may be in the region of $c \approx 2$ [26, App. 3].

1.3 Eras of quantum computing

As the title of this thesis suggests, it is anticipated that the development of quantum computing can be roughly divided into eras, each of which is characterised by the hardware capabilities of available quantum devices. Naturally, these will overlap and influence one another. The following briefly describes the categories considered in this thesis.

Simulation This age started long before the first hardware implementations of quantum devices, and will likely overlap with all of the other eras discussed here. Due to the highly sophisticated technology of classical computing, surprisingly large quantum systems can simply be simulated on increasingly productive machines. As discussed above, there seems to be a tug-of-war taking place right now, where classical simulations and quantum hardware devices try to out-compete one another. One might speculate that the time where simulation is the most capable quantum platform is about to come to an end.

NISQ The phrase of *noisy intermediate-scale quantum* (NISQ) devices was popularised by John Preskill [28]. It refers to hardware that, in contrast to the technologies discussed below, uses some physical two-level system as its logical qubits. Because, despite best efforts, these systems are not perfectly decoupled from the environment and have finite temperature, various sources can introduce errors and lead to decoherence of the quantum state. As of writing of this thesis, these devices are on the brink of overtaking classical computers in their ability to solve certain problems. Algorithms for this age focus on being resistant to the inherent noise of the system, and giving usable results despite considerable error rates.

Early fault-tolerant Once quantum hardware contains enough qubits and appropriate couplings between them, it becomes possible to implement schemes that (partly) suppress the noise of the system. This is achieved by cleverly grouping several *physical* qubits together to form a single *logical* qubit. Through sophisticated algorithms using repeated measurements of some of the qubits, it is possible to detect and even correct various errors that may occur. Such a scheme is usually called an *error correcting code*. Importantly, these code's resistance to errors increases with number of available qubits. In the early fault-tolerant era, codes are anticipated to provide *some* tolerance to decoherence events of the physical qubits, but errors may still occasionally make it through to the logical qubits. Any algorithm designed for these devices should still consider the possibility of such errors occurring.

Fully error-corrected The ultimate goal of quantum hardware development is to provide a platform where the logical qubits are protected so well from physical errors, that their influence is more or less negligible. The line between early fault-tolerance and full error correction is somewhat blurry and can be drawn at an appropriate logical error rate that is deemed negligible. From today's point of view, the implementation of such a device will likely take many more years of development time.

1.3.1 Error models

As discussed above, finite temperature and interactions with the environment can cause disruptions to a quantum state. Since the state is where all the information about a quantum computation is stored, this becomes a serious problem if it happens too frequently, because it means that a computation can only take a certain amount of time or apply a predefined number of gates, before the information is destroyed. In the fully error-corrected regime, these errors are abstracted away by the underlying code and (almost) arbitrarily long calculations may be performed. However, on other, less protected platforms, the quantum algorithms themselves that run on the hardware must be resistant against errors to some degree. In order to incorporate it into the development of algorithms, the noise must be modelled in some way.

One way for such a model is the following. At various stages during the execution of a circuit, e.g. after a certain time has elapsed, or after the application of a some particular gate, an unwanted operator (the error) may be applied to the state with some small probability. Popular choices for errors are the application of one of the Pauli operators with equal probability, which is called *depolarising noise*, or the application of an operator that takes the $|1\rangle$-state to the $|0\rangle$-state, often called an *amplitude damping error*. Especially depolarising noise is often considered, because via techniques like *twirling* [29, 30] and *randomised compiling* [31], any noise channel can be turned into depolarising noise. However, in the NISQ era, this often does not accurately capture the characteristics the various devices without such advanced methods. Reference [32] presents an effort to provide a more accurate characterisation of the platform-dependent noise that a quantum computation may encounter. The specific relevant error models to the work in this thesis are discussed in more detail in the appropriate research chapters.

Because errors only occur with some probability, the previous description of ket-vectors is not sufficient anymore to capture the full state, including the uncertainty about whether an error occurred or not. To completely characterise such a situation, a different formalism is required. One way is to represent the state not as a single ket-vector, but instead use the so-called *density operator*, which, to reflect a pure state $|\psi\rangle$ is simply

$$\rho = |\psi\rangle\langle\psi|\,.$$

However, a system that is in state $|\psi_1\rangle$ with a probability of p_1, and in $|\psi_2\rangle$ with probability p_2, can be described by

$$\rho = p_1\,|\psi_1\rangle\langle\psi_1| + p_2\,|\psi_2\rangle\langle\psi_2|\,,$$

or, more generally

$$\rho = \sum_j p_j\,|\psi_j\rangle\langle\psi_j|\,.$$

Expectation values of observables O are then calculated via

$$\mathrm{Tr}(\rho O) = \sum_j p_j O |\psi_j\rangle\langle\psi_j| = \sum_j p_j \langle\psi_j|O|\psi_j\rangle \,,$$

which is just the probability-weighted sum of the expectation values of the individual states the system may be in. Note how ρ does not actually describe the physical state of the system (which is always in some definite state), but rather respects our incomplete knowledge about the precise physical realisation in the device.

Working with density operators, state changes are induced by *quantum channels*. Applying a unitary to a state simply means multiplying ρ from the left by U and from the right by $U^\dagger$. However, as is the case for error channels, a weighted sum of (not necessarily unitary) operators may also be employed to implement an operation that *may* happen. Using the example of the damping channel, in which an error occurs with probability p, a single-qubit density operator would transform like

$$\rho \mapsto K_0 \rho K_0{}^\dagger + K_1 \rho K_1{}^\dagger$$

with the operators

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix} \quad \text{and} \quad K_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix},$$

which are usually called *Kraus operators*. They do not need to be unitary (as is obvious from this example), but they must fulfil the property

$$\sum_n K_n^\dagger K_n = \mathbb{1}$$

to ensure the channel is a completely positive trace-preserving (CPTP) map. This is necessary for ρ to remain a valid density operator.

These are just some of the most important aspects and properties of this formalism. A more complete and rigorous discussion of density operators is available in the literature [26].

If I have seen further it is by standing on the shoulders of giants.

— Sir Isaac Newton

2 LITERATURE REVIEW

Contents

Because the present thesis covers many topics in quantum simulation, quantum algorithms, and classical algorithms, this literature review will focus relatively closely on the papers and published works relevant[2] to each of the research chapters, instead of giving a sweeping overview of the whole field of quantum computing and quantum algorithms.

2.1 Quantum circuit synthesis

As mentioned in the Introduction, instructions for quantum computers are usually described in a generic way of applying certain, often standardised *gates* to a quantum state [33–36], which, when collectively performing a specific task, are then often

[2]Relevant in this case means they (try to) accomplish the same or a similar goal as the methods presented in the research chapters, but through other means.

grouped together in a *quantum circuit*. In general, this formalism is very flexible in how it can be used to describe reversible logic functions, quantum operations, and algorithms.

When developing or describing quantum algorithms, often a generic universal set is used [37], depending on what is convenient and fits the properties of the operation. Examples are the sets of single-qubit rotation gates & CNOT [38] (often used in NISQ applications), Toffoli & Hadamard [39], and {CNOT, H, S, T} [26, Sec. 4.5]. The latter is popular in the context of algorithms for error corrected platforms, as these gates can be relatively straightforward to embed into an error correcting code [40, 41].

On the other hand, gate sets can also be used to characterise a particular quantum hardware platform by way of specifying a group of unitaries that are *native* to the device, i.e. which have the most straightforward implementation in terms of actual physical operations like laser pulses or applied voltages. Naturally, these sets will be constrained in the *type* of gates (rotations, SWAPs, CNOTs, etc.), as well as the *connectivity*, i.e. which two qubits a given two-qubit gate can operate on. Some examples are devices based on trapped ions [42–46], superconducting qubits [47–50], and silicon-based hardware [51–55].

This mismatch in expressibility necessitates methods for *circuit compilation*,[3] where the description in some gate set is translated to an equivalent circuit using a different set. A somewhat related but more general task is that of *circuit synthesis*, where a circuit is constructed from a generic description of an operator's unitary action. This compilation process is analogous to the compilation process on classical hardware, where high-level programming language instructions must be translated to assembly and ultimately to byte code compatible with the specific hardware it will be executed on. In this analogy, the compilation process described in the quantum context is an intermediate step to ensure gate compatibility, before further conversion into specific physical processes like a pulse of a given duration.

If the unitary quantum operation U is given in a matrix representation (or an equivalent form from which its matrix elements can be efficiently calculated), methods

[3]Transforming circuits from one gate set and/or qubit connectivity to another is often also called *transpiling*. This thesis will consistently use the term *compiling* for this task.

collectively known as *exact unitary decompositions* may be employed. For the generic case of a completely unstructured U, it is known that no algorithm can scale sub-exponentially when working with a finite number of gates. Nielsen and Chuang [26] provide a simple, intuitive argument for this fact, which might be a useful addition to this review. Consider a unitary U on n qubits, a set of g different gates, with each gate affecting at most f qubits. A circuit containing m gates can then, starting from some initial state, produce at most $\mathcal{O}(n^{fgm})$ output states. If U is generic, it can create any output state from the input. Therefore, if the circuit was to approximate the unitary within a small error ε, it must densely cover the space of all possible states with ε-disks. From this argument, Nielsen and Chuang arrive at the conclusion that the required number of gates is of order

$$m = \Omega\left(\frac{2^n \log(1/\varepsilon)}{\log(n)}\right).$$

The above argument only applies for approximating arbitrary unitaries with a discrete set of gates. However, in practice, the task is often to exactly implement a unitary using a set of parametrised gates. For this case, Ref. [56] provides a derivation based on parameter counting, that even then, at least $\Omega(4^n)$ parametrised gates are necessary to synthesise arbitrary unitaries.

Of course, this is definitely not true for *all* U, but when considering a generic decomposition, it clearly does not make sense to look for synthesis methods with polynomial scaling in the number of gates they produce. Rather, research in this regard has been focused on reducing the actual number of gates produced, i.e. lowering the prefactor of the required exponential scaling.

So far, the discussion has been concerned with the total number of gates in a circuit. However, on near-term quantum hardware, CNOT gates are the ones that introduce disproportionately many errors to the circuit, whereas in the extended-term, the required number of T-gates is expected to be the limiting factor [40, 57, 58]. The observation that not all gates are of equal difficulty and resource-intensiveness makes it clear that simply counting the number of gates is not an ideal and universal characterisation of how hard a circuit is to implement. It also raises the question of what a good metric to judge circuit optimality could be, as apparently no single cost function can describe how expensive a circuit is to implement without knowing more about the device it will run on. The cost can be determined by the total number of generic two-qubit gates, the count of CNOTs, the number of T-gates, CNOT-depth, T-gate depth, or any other quantification of whatever the limiting resource on a given platform is. In this work, because the presented ideas target greatly disparate platforms, the metrics used to discuss and judge their efficacy will also vary depending on the context, and are mainly concerned with either total gate count, two-qubit gate count, or the number of T-gates.

One of the earliest descriptions of an algorithm to explicitly decompose an n-qubit unitary was given by Barenco et al. [38] and uses $\mathcal{O}(n^3 4^n)$ two-qubit gates, where n is the number of qubits. Subsequent work lowered this bound, with notable examples by Vartiainen et al. [59] using $\mathcal{O}(4^n)$ CNOTs and Möttönen et al. [60] ($4^n - 2^{n+1}$ CNOTs), which was followed up again by Möttönen and Vartiainen [61] arriving at an explicit construction using no more than $\frac{23}{48}4^n - \frac{3}{2}2^n + \frac{1}{3}$ CNOT gates. Shende et al. [62] come to an equivalent conclusion. Comparing this to the lower bound of $(4^n - 3n - 1)/4$ derived in Ref. [56] shows that this is asymptotically within a factor of 2 of the optimal case. Various other decomposition methods are described in the literature [63–67] and implemented as software packages, which, even if they do not improve on the worst case bound, often exhibit reduced CNOT counts for practical applications.

A disadvantage of methods directly decomposing a given unitary into a circuit is their performance on many practically relevant unitaries. We know that arbitrary unitaries can in general not be decomposed efficiently. However, most operators of interest in quantum computing are highly structured and do allow a much more efficient circuit representation than what is expected of generic unitaries. For these cases, circuit decomposition methods alone often yield suboptimal results. Therefore, some post-optimisation is often necessary to reduce the appropriate cost metric. Furthermore, since the unitary decompositions described in the literature almost exclusively try to minimise the CNOT-count, T-count optimisation will be an important tool for future applications.

Unfortunately, this task of optimising an already found (inefficient) circuit representation turns out to also be quite difficult. As Refs. [68, 69] show, deciding whether two unitaries act equivalently on a given subspace is Quantum Merlin-Arthur [70] (QMA)-complete, i.e. it cannot be evaluated efficiently, even on a quantum computer. This implies that global optimisation of quantum circuits is also QMA-complete. Therefore, optimisers for sufficiently large circuits have to rely on heuristics.

Many of the aforementioned works on unitary decomposition already include such an optimisation as a second step after the initial synthesis [63, 64, 66, 67]. Because global optimisation is hard, most of them use some local heuristic rules for circuit re-writing and gate replacement. Another approach for reducing the number of CNOT gates is to first partition the structure into smaller sub-problems. Because these sub-circuits can be chosen to be suitably small, they may then be compiled individually using any suitable algorithm that might perform better than local circuit re-writing rules. This idea is explored in Ref. [71] using exact decompositions of the sub-circuits, while [72] reports on a similar idea, but with each sub-circuit only approximated to within some unitary distance.

In some cases, an appropriately efficient circuit for the desired quantum algorithm is known, but may be incompatible with the connectivity constraints of the hardware. This specific task is often called *qubit routing* or *circuit transformation*, and several methods have been applied to efficiently find the necessary SWAP operations between gates, including simulated annealing [73], tabu search [74], artificial neural networks [75, 76], and specialised heuristics [77–80]. Furthermore, sometimes SWAPs can be avoided altogether [81].

The methods discussed so far all start from an already known (inefficient or incompatible) circuit representation of the desired operator — or construct such a circuit through decomposition — and then try to transform it in such a way that certain gates are eliminated or other constraints are fulfilled. This "top-down" approach can be contrasted with "bottom-up" methods that start with a trivial circuit, whose action is far from the desired unitary, and then try to stepwise add elements to it in an attempt to implement the target operator ever more closely.

Small problems acting on only a few qubits can even be solved using exhaustive search of all possible gate sequences [82] in order to find the optimal circuit implementing the desired unitary. Where rigorous search would be prohibitively costly, a tree-like structure may be used to represent circuit layouts, which can then be traversed more efficiently using an approximate A* search algorithm [83] to find near-optimal solutions for problems on up to four qubits [84]. Extending this idea using various techniques to reduce the size of the considered search space — which may impact the solution optimality — can be shown to work on up to six qubits [85].

A different approach is to dynamically construct a circuit through machine learning, heuristics, or metaheuristics. Several techniques have been proposed, including pseudorandom walks [86], genetic algorithms [87], temporal planning [88], and deep learning [89]. Such methods can sometimes also be used to synthesise approximations to a desired unitary, rather than a perfect re-expression, resulting in shorter circuits than exact compilation. Of particular relevance to this thesis are approaches which randomly propose circuit structure changes to try to minimise some cost function. Examples of this are the References [90–92]. Each of these works proposes the use of different cost functions, but all of them employ Metropolis-Hastings random sampling to minimise the cost.

One additional family of methods worth mentioning in this context is that of variational quantum eigensolvers (VQEs), first introduced by Peruzzo et al. [12]. In its original form, it can be seen as a highly specialised quantum circuit synthesis method; it generates a circuit that prepares the ground state of a given Hamiltonian.

There are numerous variations of VQE-based approaches. The fundamental idea, which they all have in common, is that some parametrised (shallow) circuit $U(\underline{\theta})$ is used in combination with an easy-to-prepare initial state $|\psi_0\rangle$ to generate a parametrised state $|\psi(\underline{\theta})\rangle := U(\underline{\theta})\,|\psi_0\rangle$ in a quantum register. Then, by varying the parameter vector $\underline{\theta}$ and guided by some algorithm through repeated monitoring of the expectation value of the Hamiltonian, the energy of the state is minimised.

Differences in the various incarnations of VQEs are mainly the concrete method of driving the parameters towards an energy minimum, and the specific layout of the parametrised circuit (or how it is generated). Optimisation algorithms used in this context largely fall into two distinct categories: gradient-free and gradient-based methods. Some examples of gradient-free optimisers used in the literature are the Nelder-Mead algorithm [93] in the original report on VQEs [12] and COBYLA [94] as used in Ref. [95], and CoVaR [96]. Instances of the highly popular gradient-based approaches include conjugate gradient [97], ADAM [98], quantum natural gradient descent [99] and imaginary time evolution (ITE) [100].

Although seemingly well-suited for NISQ hardware, VQEs often encounter significant problems. These are mainly related to limited ansatz expressibility [101, 102], barren plateaus [102, 103], and false minima [104–106]. *Limited ansatz expressibility* arises from the fact that in order to classically optimise the parameters, their count should scale at most polynomially with the system size. This means that only a small subsection of the full Hilbert space is within reach of the output state. Therefore, care must be taken that the fixed (or automatically generated) ansatz circuit structure can produce the desired target state within its parameter space. Sometimes, but not always, enough is known about the Hamiltonian to ensure this property. *Barren plateau* is the name given to the phenomenon that with increasing degrees of freedom, the energy landscape with respect to the parameters becomes exponentially flat in large regions of the parameter space, which makes it difficult for the optimisation routine, especially gradient-based ones, to find the ideal values of $\underline{\theta}$. *False minima* are a well-known problem in many (classical) optimisation tasks, where it is easy to find a local minimum, but difficult to find (and verify) that a given local minimum is also a global one. Despite these limitations, VQEs have been demonstrated to work well and produce accurate results in some settings [100, 107].

As will be elaborated later, even though VQEs are usually used to find the ground state of some physically relevant Hamiltonian, the problem of synthesising a circuit representation of a given unitary can be rephrased into such a ground state finding problem. Therefore, in principle, any sufficiently sophisticated VQE method may be employed to synthesise circuits implementing a given unitary. The most relevant algorithms in this context are those that generate gate sequences dynamically, as opposed to those working with a fixed ansatz and only optimising its parameters. Examples of such generative algorithms are ADAPT-VQE [108, 109] and the Evolutionary Variational Quantum Eigensolver (EVQE) [110].

Chapter 3 investigates a method in the same spirit of dynamically creating a circuit from scratch, combines many existing ideas with new techniques, and tests its suitability to synthesise small circuits, as well as its scaling properties to larger ones. Results are reported for up to 10 qubits, which required classical simulation of a 20-qubit device.

2.2 Time evolution for quantum chemistry

As already covered in the Introduction, one of the most promising applications of quantum computers is the efficient simulation of the time evolution of quantum systems. A particularly important case for potential future applications is that of interacting electrons in molecules, i.e. quantum chemistry. This task may also be seen as a highly specific circuit synthesis task, where the target unitary has the form $U(t) = e^{-iHt}$, with H the Hamiltonian of the system in question. The available methods to implement such a time evolution on quantum hardware are usually grouped into two main categories: Trotter-Suzuki-type decompositions, and post-Trotter methods.

2.2.1 Trotter-Suzuki decomposition

Initially proposed by Trotter [111], and expanded on by Suzuki [112], Trotter-Suzuki formulas — often also called *product formulas* — are a relatively simple and straightforward, yet powerful tool for constructing circuits. They are applicable if the Hamiltonian can be written as a sum of operators

$$H = \sum_{j=1}^{L} H_j,$$

where the time evolution governed by each individual H_j, i.e. the unitaries $e^{-iH_j t}$, can be implemented efficiently as a series of quantum gates. Then the total time evolution dictated by the full Hamiltonian H can be approximated by applying [26]

$$e^{-iHt} = \exp\left(\sum_{j=1}^{L} -iH_j t\right) = \prod_{j=1}^{L} \left[\exp\left(-iH_j t\right)\right] + \mathcal{O}(t^2).$$

Since the error term is proportional to t^2, this method is suitable only for small times t, where $t^2 \ll t$. To simulate longer times, the longer time t is usually divided into many small segments $\Delta t = t/r$, and the approximation is applied r times, each with time step Δt. Apart from this first order formula (i.e. it is exact up to linear order in t), higher order formulas exist [113], where the nth order decomposition is exact up to t^n, and the error term is of order t^{n+1}. Curiously, the odd-numbered orders exhibit the same error as the next-lower even-order formula, and are therefore usually neglected [114].

Product formulas have been ubiquitous in the literature as a method to simulate the dynamics of a quantum system since the first concepts of such calculations through to more recent works [115–120]. Because of their significance, considerable effort has gone into studying their properties and improving their performance by tweaking parts of the method.

In 2014, after investigating the required gate count for quantum chemistry problems, Wecker et al. [121] came to the conclusion that straightforward application of product formulas would lead to impractically deep circuits, i.e. the required coherence time exceeded those that seemed realistic at the time on quantum hardware by many orders of magnitude, leading the authors to the conclusion that drastic algorithmic improvements would be necessary to make such calculations feasible. The authors of Reference [122] argue that one important reason why Ref. [121] arrived at such infeasibly high resource estimations is that the chemical systems considered were not real-world molecules, but artificially created ones. In their study, which considers real molecules, but also includes other improvements to the algorithm, the authors conclude that much better scaling than previously reported is possible. Indeed, when resorting to empirical studies of molecules for error scaling, the most convincing results are those drawn from real compounds.

Many further variants of product formulas have been proposed and analysed since then. Hastings et al. [123], for example, showed that — among several other optimisations — rearranging the order of the individual terms before the decomposition can significantly reduce the error due to averaging effects. Somewhat similar to the idea of rearranging the terms of the Hamiltonian in Ref. [123] is the study by Childs et al. [124], where the order of the terms is not fixed in an optimal way beforehand, but rather rearranged randomly for every small time step Δt. This can lower the error bounds significantly due to cancellation and averaging effects and is surprisingly efficient, given its simplicity.

Campbell [125] takes the approach of randomisation even further by not including every term of the Hamiltonian in every time step, but rather randomly choosing them according to their magnitude and adjusting their weights such that on average the correct operator is applied. For electronic structure Hamiltonians, this random method named *qDRIFT* can lead to considerably lower gate counts in the chemically relevant time- and error regime [126]. An extension of this scheme [127] even allows fine-tuning of the performance to specific calculations and hardware availabilities.

Another important aspect to consider is the tightness of the known error bounds. Reference [128] shows that — again for electronic structure Hamiltonians — the bounds given in Ref. [121] can be loose by up to sixteen orders of magnitude. This is problematic, since for calculations not accessible to classical computers, the bound is the most direct accuracy metric to determine necessary gate counts. Reference [129] provides a more rigorous analysis of Trotter error than earlier works, arriving at much tighter bounds. The expressions for these bounds contain nested commutators of the terms in H, which emphasises one important feature of Trotter-Suzuki product formulas: they automatically take advantage of commuting and near-commuting terms in the Hamiltonian. This benefit is absent in many of the other methods described in this chapter, and can reduce the errors significantly for some systems. However, even though the required gate count was substantially reduced over time through the just discussed investigations, the scaling of the inverse simulation error of all product formulas remains polynomial in the circuit gate count.

In the spirit of probing product formulas for their applicability to quantum chemistry, Section 4.2 in Chapter 4 explores a particular version of Trotterised time evolution in this specific context and gives some realistic resource estimates for potentially interesting problems.

2.2.2 Post-Trotter methods

Because of the significance of Trotter-Suzuki-type product formulas, later approaches are often grouped together as "post-Trotter" methods. One important member of this group is the linear combination of unitaries (LCU) method. First introduced in Ref. [130], it was originally used to apply sums of different product formulas, so-called multi-product formulas known from classical computations [131]. In this initial iteration, the method is probabilistic and relies on post-selecting the ancilla for a specific state with non-unity probability.

Berry et al. [132] then expanded the principle to directly apply a truncated Taylor series expansion

$$e^{-iHt} \approx \mathbb{1} + \sum_{k=1}^{n} \frac{(-it)^k}{k!} \left[\sum_{\ell_1 \ldots \ell_k = 0}^{L-1} H_{\ell_1} \ldots H_{\ell_k} \right],$$

since this is also a sum of unitaries. One important improvement in that work is the use of so-called oblivious amplitude amplification [133]. This variant of amplitude amplification does not need knowledge about the input state — something that usually makes the use of such an amplification impractical — hence the name *oblivious*. This improves the method from probabilistic to a near-deterministic one. The authors also describe a very efficient way to construct an appropriate gate sequence, taking advantage of the particular structure of the terms in the series. Because the factorial in the denominator suppresses the magnitude of the terms in the sum very quickly, this method has exponentially better scaling of the gate count with the desired inverse error than previous methods. Introducing a correction step in the process can further decrease the complexity of the circuit [134].

The LCU method is also used a number of further implementations for Hamiltonian simulation. Low et al. [135] use linear combinations of product formulas, very similar to the original work on LCU in Ref. [130]. The key improvement is that the specific multi-product formulas proposed there take advantage of commuting terms in the Hamiltonian — like pure product formulas — while improving the complexity scaling with the inverse error to be only poly-logarithmic.

Another technique that benefited from the use of LCU is that of Hamiltonian simulation by quantum walks, also called *Szegedy walks*. These walks seek to obtain information about the Hamiltonian not by directly implementing a system's time evolution, but rather by applying a unitary operator of the form $\mathcal{W} = e^{i\arccos(H)}$, the so-called *walk operator*. The spectrum of this operator is isomorphic to the spectrum of H, and therefore allows inference of the Hamiltonian's eigenvalues from the eigenvalues of $\mathcal{W}$. Crucially, the operator $\mathcal{W}$ can be implemented exactly on a quantum computer, while this is usually not the case for H.

In Ref. [136], LCU is applied to enhance the cost scaling with the error of Szegedy walks while retaining their advantage for sparse Hamiltonians. For the specific case of quantum chemistry, and also using LCU combined with Szegedy walks, the authors of Ref. [137] arrive at the conclusion that the number of T-gates when using their approach scales like

$$\mathcal{O}\left(\frac{N^3 + N^2\log(1/\varepsilon)}{\varepsilon}\right),$$

which is very similar to the truncated Taylor series method in the allowed error ε of the simulation, but better in the number of atomic orbitals N. The authors furthermore estimate that using their method, some quantum chemistry problems that are currently inaccessible for today's classical hardware could be solved within a few hours using only about a million physical superconducting qubits, assuming an error rate of 10^{-3}, which corresponds to a few hundred logical qubits.

Further evolutions of the quantum walk approach are *quantum signal processing* [138, 139] and *qubitization* [140], which seek to linearise the Szegedy operator $\mathcal{W}$ such that the operator e^{-iHt} is recovered. In addition, qubitization has variants specifically for quantum chemistry [141].

The methods that make more elaborate use of LCU than the truncated Taylor approach, i.e. multi-product formulas, qubitization, and quantum signal processing, have been shown to exhibit even better scaling for many types of practically relevant Hamiltonians [137, 140–142]. However, there are instances where they are less suited; one prominent example being the simulation of time-dependent Hamiltonians. Even for intrinsically time-independent cases, introducing a time dependence by transforming to a rotating frame can be beneficial if the Hamiltonian is diagonally dominant. In contrast to qubitization and quantum signal processing, the approach of the truncated Taylor series in [132] can be applied to such time-dependent cases with reasonable overhead, as shown in Refs. [143, 144], making it very relevant for such instances.

Because of this continued relevance of the truncated Taylor series method, Section 4.1 in Chapter 4 investigates a modification that increases its performance when used for quantum chemistry problems, and finds significant improvements over the canonical version.

2.3 Hamiltonian energy spectra

The preceding sections already stated that the time evolution of Hamiltonian systems is often seen as a very central and important task in regards to potential uses of quantum computers. One of the reasons for this is the fact that once an efficient time evolution algorithm has been implemented, this evolution can be used to determine energy levels. This, in turn, reveals important information about the system, like equilibrium positions, excitation energies, reaction rates, or emission and absorption spectra of molecules. Perhaps the most well-known way to do this is the so-called *quantum phase estimation* (QPE) algorithm [145]. Repeatedly performing the QPE algorithm with an initial state that contains mostly the eigenstates of interest reveals (part of) the Hamiltonian spectrum.

Perhaps the most significant downside to using this technique is that it requires quite deep circuits, and the time evolution to be controlled on varying ancillary qubits, followed by a quantum Fourier transform, further increasing the circuit depth. This limits its usefulness in the near- to mid-term eras of NISQ and early fault tolerant devices. Several resource-friendlier techniques have been proposed for determining (a part of) the spectrum of a given Hamiltonian; one approach is to keep using ideas from QPE, but reduce its quantum resource requirements through clever measurement and post-processing techniques.

A first simplification is to use only a single qubit as ancilla, and sequentially estimate each bit of the phase separately, starting from the least significant bit. Using already gained information about bits of lower significance, the higher significance bits can then be determined in an iterative manner, which considerably reduces the hardware requirements. This procedure was first discussed by Dobšíček et al. [146] under the name *iterative phase estimation algorithm* (IPEA).

Another approach was proposed by Svore et al. [147], referred to as *fast phase estimation*, where the quantum system evolution times are not neatly aligned with the bits in the binary representation of the phase, but rather chosen randomly. This allows inference of information about multiple bits at a time, ultimately lowering the required circuit depth and number of qubits further. The *Bayesian phase estimation* (BPE) [148] approach takes this randomised idea even further, by first assuming a prior distribution of what the phase might be, collecting data throughout the iteration, and continuously updating this prior in order to infer the (likely) ideal next measurement. This procedure is shown to yield even quicker convergence in some cases, with the additional benefit of being able to handle noise in the system better than previous techniques. Within the BPE framework, it is possible to eliminate the need for controlled time evolution (which is usually a big challenge), in exchange for controlled state preparation, as Sugisaki et al. [149] show. Variants of the latter approach specialised for the calculation of spin excitations [150] and vertical ionisation energies [151] also exist.

Even better suited for NISQ-type hardware might be the approach in Ref. [152], which prepares some initial state populated with the eigenstates of interest, and then directly measures the expectation value of the time evolution operator via a single ancilla. Fourier transforming this time signal on classical hardware then reveals the spectrum. The authors of Ref. [153] explore this idea further, and provide some numerical evidence for simulated quantum computers, but highlight some limitations on noisy hardware, in this case superconducting qubits. In Ref. [154], a similar idea is investigated, but rather than exactly extracting eigenvalues, this work presents efficient methods of determining in which one of several bins an eigenvalue might lie. Clinton et al. [155] build on this approach and, using the practical example of the Fermi-Hubbard model and building on gate synthesis methods presented in Ref. [156], show that the combination of these techniques moves the resource requirements for an accurate phase estimation quite close to what is currently available on noisy hardware.

While discovering the relative phase change of a quantum state via Fourier transform is a relatively well-explored route, it is worth noting that maximum-likelihood approaches based on classical and quantum Fisher information have been shown to yield the same scaling as Fourier-based methods [157, 158]. It is possible that as methods become more sophisticated, despite both methods having equivalent asymptotic scaling, one of them will exhibit significantly smaller constant factors.

Now briefly moving away from QPE approaches, an alternative way that completely eliminates the need for time evolution altogether is to use variational quantum eigensolvers (VQEs), whose basic principles have already been discussed in Section 2.1. Because the ground state energy of a system often takes the place of the most significant and important quantity, many methods address this state and its energy exclusively. While some of these techniques show promising performance in regards to finding the *ground state* energy of a system, one drawback of VQEs is that although such a ground state preparation is conceptually relatively straightforward, some more elaborate techniques are required to prepare *excited states*.

One solution is to search for the minimum of the shifted and squared Hamiltonian $(H-\lambda)^2$, which will find the eigenstate closest in energy to λ. This method is commonly referred to as the *folded spectrum method* and has been described as early as 1934 by MacDonald [159], has since been used e.g. in classical computations for silicon quantum dots, and was recently also shown to work on ideal simulated quantum hardware in a method called *folded spectrum VQE* [160]. Although it comes with a quadratic penalty in the number of terms in the Hamiltonian that needs to be simulated, this method might be useful in some cases on NISQ hardware.

The spectrum of a Hamiltonian can also be built up one state at a time using variational methods, as Ref. [161] shows. Once the ground state $|\varphi_0\rangle$ of a Hamiltonian H is found, a modified operator $H_1 := H + \beta_0 |\varphi_0\rangle\langle\varphi_0|$ can be used, in which the energy of the found ground state is raised such that the ground state of H_1 is now the first excited state of H. Iterating this procedure can, in principle, discover the whole spectrum of H and its corresponding eigenstates, although highly excited states require significant effort. The authors of [162] present a similar scheme based on imaginary time evolution.

If the goal is to prepare *any* (random) eigenstate and determine its energy, another approach is to minimise the variance of the energy $\mathrm{var}(E) = \langle H^2\rangle - \langle H\rangle^2 \geq 0$, which will only vanish for eigenstates. Zhang et al. [163] use this approach to sample high-energy eigenstates of a Hamiltonian. Once such an eigenstate is found, its actual energy can be straightforwardly measured. The authors of Ref. [163] demonstrate this principle using a similar approach, reaching the conclusion that targeting a specific state with a known energy and optimising for the energy and variance simultaneously can lead to faster convergence than exclusively using either one of the two.

The discussion so far has been focused around determining energies on an absolute scale, i.e. relative to a reference system that does not evolve in time. However, often this is not strictly necessary. Many interesting properties — such as the aforementioned absorption and emission spectra, reaction rates, etc. — are dictated by the magnitude of the gaps between the eigenstates, rather than their absolute energies. Several works have looked at this problem in particular, and how to implement it in a resource-friendly way on quantum hardware.

Reference [164] — inspired by [165] — proposes to approach this problem by repeatedly preparing a superposition of two eigenstates as an initial state, acting k times on that state with a unitary operator whose effect is in some deterministic way related to the energies, and then measuring the overlap with the initial state.[4] The authors demonstrate their method with a relatively simple implementation of calculating energy gaps in an H_2 molecule on IBM quantum hardware. One downside

[4]In practice, this overlap is measured by un-preparing the final state and measuring the overlap with the computational basis state from which the initial state preparation started.

of this work is the necessity of preparing the initial superposition of exactly two eigenstates, which might be non-trivial.

References [166, 167] follow essentially the same idea as [164], but with some slight modifications. In these approaches, the times for which the system is evolved are drawn probabilistically, and a superposition of more than two states is investigated. Numerically simulated and experimental results show that this method can — given enough quantum resources and a-priori knowledge about the system — yield quite accurate results.

The authors of Ref. [168] present yet another similar scheme, but use the isochronally sampled time evolution of the expectation value of an appropriate observable more directly via a Fourier transform to unveil the eigenenergies — quite similar to the method in Ref. [152], but without the need for any controlled gate. They further note that if an observable can be found that anti-commutes with the Hamiltonian, even the absolute spectrum can be calculated, not only the transition energies. In the same manner, Stroeks et al. [169] also use an isochronally sampled signal, but only consider the overlap of the time-evolved state with the initial state instead of generic observables. Using the ESPRIT algorithm [170], they show that under some conditions, equivalent information can be extracted from the imaginary-time evolution signal (as opposed to real-time evolution), and propose a Monte Carlo scheme to obtain that signal from a given Hamiltonian.

Matsuzaki et al. [171] investigate a method to find the gap of a Hamiltonian also based on the Fourier transform of a signal sampled at regular intervals, but partially circumvent the problem of preparing an appropriate initial state by employing quantum annealing (QA). Within the regime accessible to QA, this may be a viable option for future applications.

Overall, the task of efficiently finding energies, or even energy differences, of a Hamiltonian remains a topic of high importance and research interest in the realm of quantum computation. Finding absolute energies via QPE and related techniques often still requires too many quantum resources to be viable, not least because of the necessary controlled time evolution, while VQE protocols are not expected to scale well due to issues with ansatz expressibility, barren plateaus, local minima, and the required measurement effort. Approaches for determining the size of energy gaps in the Hamiltonian via time series analysis seem promising candidates for being applicable on NISQ hardware, but can also scale to early fault-tolerant and fully error corrected devices.

The drawback of these time series methods is that quite often, little is known about which observables should ideally be used for the time series in order to yield good signal. Chapter 5 proposes a method that circumvents this problem by employing the technique of classical shadows, which allows the simultaneous evaluation of exponentially many observables, mitigating this problem of selecting appropriate operators by hand *a priori*. Correlation analysis between the observables can then recover signal even when no single observable would suffice.

2.4 Preparation of Hamiltonian eigenstates

As discussed in the previous section, when modelling physical systems — such as molecules, but also other key models of interest like spin chains or the Fermi-Hubbard model — on quantum hardware, a key quantity of interest is the energy spectrum of the simulated system. It alone often contains valuable information about the system's core characteristics. However, in some cases, it can be desirable to actually prepare one of the Hamiltonian eigenstates on given quantum hardware to perform additional measurements or transformations on it. This allows the inspection of properties not directly reflected in the Hamiltonian (and therefore the energy spectrum), e.g. certain symmetries, magnetisation in the absence of an external field, etc.

One method where this feature of preparing an eigenstate is included without any extra effort is that of the previously discussed variational quantum eigensolver, as they optimise the parameters of some ansatz circuit to produce a state, whose energy is minimal within the accessible parameter space. Therefore, by their very nature, these approaches produce the full ground state in the register (provided they converged appropriately), which is then of course also available for further investigations besides measuring its energy. As elaborated on in Section 2.3, this procedure is not limited to the ground state, but can also work on excited states.

Due to the aforementioned limitations and practical problems of finding the appropriate circuit structure and ideal parameters, it may not seem likely that VQE methods will have many applications beyond the NISQ era. Therefore, I will turn focus to the early fault-tolerant and fully error-corrected hardware regimes, in which algorithms with a higher demand on quantum resources but without convergence problems seem likely to be more relevant.

Perhaps the most ubiquitous, and what might be called the "canonical" way to prepare eigenstates of a defined energy, is the previously mentioned *quantum phase estimation* (QPE) algorithm. As mentioned in Section 2.3, performing a QPE with some initial state $|\psi\rangle$ that is not an eigenstate will return an approximation to the energy of one of the eigenstates with a probability proportional to the overlap of the respective eigenstate $|\varphi_j\rangle$ with the initial state $|\langle\varphi_j|\psi\rangle|^2$. However, additionally (and crucially), after the QPE procedure, the main register (without the ancilla) is left in a state that has significantly more overlap with $|\varphi_j\rangle$ than before the procedure [26, Sec. 5.2.1]. It can therefore be used as a probabilistic state preparation method, if the initial state $|\psi\rangle$ already has considerable overlap with the target state. Additionally, when targeting a specific state, its energy must be known in advance through other means; a limitation VQEs typically do not face when trying to prepare the ground state.

A more direct approach to prepare an eigenstate is through the use of so-called *quantum nondemolition measurements* (QND) [172], originally introduced in the context of gravitational wave detectors [173]. This type of measurement refers to observables whose repeated evaluation at various points in time does not change the measured quantity. Trivially, any observable that commutes with the system Hamiltonian H is a QND observable, and thus also the energy $E = \langle H\rangle$ itself. In Ref. [174], the authors use the coupling of the system to an ancilla qubit in order to measure the full interacting many-body Hamiltonian H in a single shot, thus collapsing the system into a narrow band of eigenstates, with energies close to the measured one. To show its real-world applicability, the procedure is demonstrated experimentally on trapped ions. While QND measurements are a very interesting research direction, the method is difficult to generalise to arbitrary Hamiltonians and thus not quite as generic as most other methods described here.

Another method, which is somewhat related to the imaginary-time evolution sometimes used in VQEs, is the so-called *probabilistic imaginary-time evolution* (PITE). It uses both controlled forwards- and backwards real-time evolution, as well as measurements, to implement the (non-unitary) imaginary-time evolution probabilistically. Though conceptually intriguing, the authors show that upon implementation, in many cases the success probability of each time step remains impractically small, suppressing the chance of success of the whole procedure exponentially. Therefore, it seems that further work is required to make this approach feasible for practical applications.

Real-time evolution as in QPE and PITE can also be used in a stochastic way to selectively keep only a specific set of states within a small energy window. This method, called the *rodeo algorithm* [175], uses a circuit construction very similar to that of QPE, but the times for which the quantum system is evolved between measurements are drawn from a probability distribution. This results in random, but rapid, convergence of the state towards the target. Similar to QPE and other related approaches, it requires knowledge of the energy of the targeted state, controlled time evolution of the system, and measurements of ancilla qubits. This approach seems very promising, but its stochastic nature leaves some room for improvement.

Chapter 6 will present a method that is quite closely related to the rodeo algorithm and quantum phase estimation, while also drawing from ideas of the PITE approach, but has some distinct advantages over each of these existing methods. Although still probabilistic in its outcome for a specific run, the method itself is deterministic, which allows rigorous quantification of the convergence and usage of prior knowledge of the state.

2.5 Simulating quantum computers on classical hardware

In the current era of quantum computing, where NISQ devices are just bordering on outperforming classical computers in some highly specific tasks, it is apparent that development of new quantum algorithms, potentially targeting hardware many years away, cannot be bound to the present hardware capabilities. In the early phase of quantum computing, the most significant contributions to the field have been made by deriving them analytically, as evident by the description in Chapter 1. The Deutsch–Josza algorithm [5], Shor's algorithm [6], and Grover's search [8] are just some of the most popular analytically derived results.

However, as the literature on quantum algorithms becomes more and more populated with increasingly sophisticated methods of achieving desired results on quantum hardware, analytic results are sometimes difficult to obtain. In these cases, a demonstration via direct implementation of a scheme can often suffice as a first proof-of-concept. Because new algorithms frequently target hardware with noise levels and coherence times beyond the currently available devices, we often resort to classical simulation of such quantum processors.

In the most basic case of wanting to simulate only a few qubits and trivial gates, a simulator can be quickly implemented using any sufficiently sophisticated linear algebra system and does not require any software packages specialised in simulating quantum systems. However, using software specifically dedicated to emulating[5] quantum hardware can be useful when collaborating with other researchers, publishing research code, requiring specialised, non-trivial quantum operations, and, of course, when trying to simulate so many qubits that it substantially stresses the classical hardware and optimisations are required to keep simulation times and memory load feasible.

As many research groups globally work concurrently on advancing the field of quantum computation, diverse use cases and demands led to the emergence of a great variety of emulator software packages. In this section, I will briefly address the most

[5]I will use *simulating* and *emulating* interchangeably in this thesis, as is common in the quantum (but not classical) computing field.

popular implementations together with some of their advantages and disadvantages, where appropriate, to offer a bit of background for the development of *pyQuEST*, a wrapper for an existing quantum simulation suite *QuEST*, presented in Chapter 7.

Many simulators discussed here are not exclusively dedicated to bringing the best possible simulation to the user, but rather provide a whole toolkit to develop and work with quantum algorithms and circuits, and run it on various *backends*, i.e. mechanisms that perform the actual calculation.

Perhaps the most widely used one of those packages is *Qiskit* [176], developed by IBM, and available for Python. It contains a wide variety of tools and functions to create and manipulate quantum circuits. The standard installation of Qiskit comes with a highly non-optimised built-in quantum simulator for very simple tasks on few qubits. For larger calculations requiring more resources, the *Aer* package is provided by the same developer team, which seamlessly integrates with an existing Qiskit installation. However, Qiskit can also deploy to other backends, some of which are classical simulators (like *Rigetti Quantum Virtual Machine*, or the graphics card (GPU) accelerated *cuStateVec*), but most address actual quantum hardware, like that provided by *IBM Quantum*, *Azure Quantum*, *Alpine Quantum Technologies*, *Quantinuum*, etc.

Another popular Python package for creating and modifying circuits is *Cirq* [177], developed predominantly by Google. Very similar to Qiskit, it focuses mainly on providing functionality for conveniently handling quantum circuits, and comes with basic, but quite inefficient, simulation capabilities. For a more optimised simulator backend, the developers point to *qsim* [178], which integrates well with Circ. Furthermore, some backends addressing various cloud-based quantum hardware architectures are also available.

The *Amazon Braket Python SDK*[6] [179] follows a very similar pattern. It provides tools to create circuits, comes with a local simulator, but can also run the user's algorithms on more powerful classical hardware provided by Amazon Web Services, as well as on cloud-based NISQ devices by various providers. The *Microsoft Azure Quantum Development Kit* [180] uses a slightly different philosophy. It comes not only

[6]SDK here means software development kit.

with support for Python, but also has its own quantum programming language Q#, as well as APIs for .NET languages. Similar to above, results of circuit executions are generated via either a local simulator, or cloud-based quantum hardware. Another project in the same vein is *ProjectQ*, initially developed by Steiger et al. [181] at ETH Zürich. It, too, provides similar capabilities of creating circuits and abstractions, can execute circuits using the builtin classical simulator, but may also dispatch them to NISQ hardware available in the cloud.

As mentioned above, all discussed options for simulators so far are first and foremost full toolkits, mainly focused on providing the user with instruments to efficiently and conveniently implement or transform their circuits and algorithms. Naturally, their simulation backends are often not highly optimised, and thus do not utilise the available hardware to its full potential. Therefore, when simulator performance is of high importance, software focusing more strictly on optimising the classical simulation aspect can often be beneficial.

One such high-performance simulator is the *Intel Quantum Simulator* [182] (Intel-QS), formerly known as *qHiPSTER* [183]. Available for C++ and Python, it takes the relatively straightforward approach of storing the full pure state vector (but importantly not density matrices) and implementing gate applications to it algorithmically (rather than via explicit vector-matrix-multiplication). The main design foci of this software are parallelisation of these operations across multiple CPUs with shared memory (i.e. a single computer with a multi-core CPU), as well as distributing the state across distinct but connected computers (often called *nodes*) in a compute cluster. With development supported by a large CPU manufacturer, it makes excellent use of advanced processor instruction sets such as the *Advanced Vector Extensions* (AVX). It cannot, however, run its calculations on graphics processing units (GPUs), which often provide a significant reduction in runtime due to their massive parallelisation capabilities.

A more versatile but also very performant simulation software is *Qulacs* [184]. It allows storing pure state vectors, but also density matrices, enabling the simulation of noisy circuits at the cost of using more memory than a pure state calculation. Also available for use through C++ and Python, it can utilise multiple CPU cores and

GPUs to carry out its calculations. Distribution across multiple compute nodes is also supported, although this comes with a more limited set of available operations.

Outside the realm of the popular languages C++ and Python, the framework *Yao* [185] is worth mentioning. It is written for the Julia programming language [186], whose combination of high-level language features combined with strong performance make it an interesting choice for applications like emulating quantum hardware. Supporting pure state vector and density matrix quantum registers, the calculations can be performed on either multiple CPU cores, or on GPUs. Though the Julia language is not the most widespread, Yao's high performance combined with Julia's features make it an intriguing alternative to other more popular simulators for C++ and Python.

The final simulator to mention here is *QuEST* [187], another emulation software package storing the full pure state vector or density matrix in memory. Its approach is very similar to the just mentioned Intel-QS and Qulacs simulators, in that it aims at high performance calculations by applying gates to quantum states in an efficient algorithmic manner, instead of straightforward matrix-vector-multiplications. QuEST's feature set, however, is appreciably more general than that of the simulators discussed above, as it supports multiple CPU cores, GPU acceleration for both Nvidia and AMD graphics cards, and can scale to compute clusters without loss of features. The main downside of QuEST is that code utilising it must be written in C or C++, which often requires more time and care than development in higher-level languages like Python. Chapter 7 presents a solution to this limitation by describing a Python interface, which can be used to interact with the QuEST backend while still using Python's rich feature set.

It should be noted that while storing the full state vector or density matrix in memory is the most general way to emulate quantum hardware, it is also the most resource-hungry approach. For specific types of problems, techniques like matrix product state [188] (MPS) representations or low-rank stabiliser decompositions [189] can save large quantities of memory and compute time. Some simulation software uses this to provide more efficient calculations for specific classes of problems. In the present context of full state simulators, however, these will not be elaborated on in detail.

In addition to the discussed simulators, there are countless other software utilities targeting different aspects of quantum computing and related topics, which are also outside the scope of this review. To the interested reader, Reference [190] provides a large collection of such projects, which is still not complete, but illustrates the great variety of available codebases in the context of quantum computing.

Still round the corner there may wait, a new road or a secret gate.

— J. R. R. Tolkien

3 GENERIC CIRCUIT SYNTHESIS

The research in this chapter was conducted jointly by Simon Benjamin, Cica Gustiani, and myself, with the core idea coming from SB, methodological details being developed by all three, and numerical results presented later mainly generated by CG (subspace compilation results) and RM (all other results). Its findings were published as Ref. [191], which this chapter will closely follow. All writing below is my own, in places using text verbatim from the published manuscript.

Contents

This first research chapter of the present thesis will focus on an exploratory effort in synthesising generic quantum circuits. For the particular approach described here, the desired operator U, which the output circuit should approximate or exactly represent, must be given as a "black box" operator on a (real or simulated) quantum device, similar to the method discussed in Ref. [92]. In simulation, this can take any form: a full matrix, an algorithmic description, a circuit in a different gate set, or any other way which appropriately describes which input states map onto which output states. On actual quantum hardware, this could be, for example, an inefficient and expensive implementation via a deep circuit, for which a more concise form should be found.

The investigated method integrates well-known ideas with some, to the best of my knowledge, new techniques. These novel contributions are the restriction of the cost function to a certain subspace, detection of redundant gates using the quantum metric tensor, and the application of adapted versions of *random search* and *tabu search* to the problem of finding a circuit structure.

To assess the efficacy of the proposed algorithm, the quality of the output circuits, as well as the scaling properties to many qubits, a diverse set of tasks is considered under various constraints on a range of numbers of qubits. To ensure the ability to investigate the influence of using different gate sets on the outcome, all calculations were performed on quantum devices with up to 20 qubits, emulated by classical hardware via pyQuEST [192] (see also Chapter 7) and QuESTlink [193].

This chapter is structured as follows. Section 3.1 explains how the presented method relates to and builds on previous work, and introduces the cost function used as part of the algorithm. In Section 3.2, the specific routines and subroutines used to generate the circuits are discussed in detail, followed by Section 3.3, in which the results of various synthesis tasks are reported. Their significance and implications are then discussed in Section 3.4.

3.1 Unitary equivalence via energy minimisation

3.1.1 Previous work

This chapter substantially builds upon earlier formalisms introduced in, for example, Refs. [92, 194]. For completeness, this section briefly recapitulates their methods and motivates the later proposed extensions to them.

When synthesising a circuit $\mathcal{C}$ for a given unitary U, some measure of how well $\mathcal{C}$ approximates U is necessary to drive an optimisation routine towards (approximate) equivalence of $\mathcal{C}$ and U. Jones and Benjamin [194] use the energy of an artificial Hamiltonian to provide such a measure for the case where equivalence is wanted only for a single input state $|\psi_0\rangle$. Khatri et al. [92], on the other hand, provide a method called *Hilbert-Schmidt test*, which uses the average fidelity $\langle\psi|\mathcal{C}^\dagger U|\psi\rangle$ over Haar-distributed random states $|\psi\rangle$ to take *all* input states into account when measuring the closeness of U and $\mathcal{C}$. In the following, the derivation starts from the formalism in [194] and extends it to arrive at a variant of the Hilbert-Schmidt test, which will then be used as the cost function in the rest of the investigation.

The technique in [194] starts by first applying the desired unitary U to the target input state $|\psi_0\rangle$. Then, the task is to find a circuit $\mathcal{C}^\dagger$ that inverts the action of U, such that at the output $|\psi_1\rangle := \mathcal{C}^\dagger U|\psi_0\rangle$ the initial state $|\psi_0\rangle$ is recovered up to a global phase.[7] If the output is proportional to the input, $|\psi_0\rangle \sim |\psi_1\rangle$, then necessarily $\mathcal{C} \sim U$ holds for the input state $|\psi_0\rangle$. This condition can be checked using an appropriately constructed (artificial) gapped Hamiltonian $\tilde{H}$, whose ground state of some known energy is $|\psi_0\rangle$. If this ground state energy is measured at the output, the original input state is necessarily recovered. The initial problem is therefore now an energy minimisation task — as depicted in Fig. 3.1 — of the form

$$\min_{\mathcal{C}} \langle\psi_0|U^\dagger \mathcal{C}\tilde{H}\mathcal{C}^\dagger U|\psi_0\rangle. \tag{3.1}$$

However, it is often desirable to synthesise a circuit $\mathcal{C}$ that completely recovers the action of a given unitary U for *all* relevant input states. This might be the full

[7]In this chapter the symbol $\sim$ between operators means equality up to a potential global phase, i.e. $A \sim B \leftrightarrow A = e^{i\theta}B$ with arbitrary real θ.

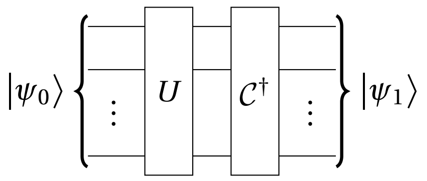


Figure 3.1: The setup used in Ref. [194] to synthesise a circuit $\mathcal{C}$ which has the same action on $|\psi_0\rangle$ as U. The target is $|\psi_1\rangle \sim |\psi_0\rangle$.

Hilbert space of $|\psi\rangle$ — I will call it $\mathcal{H}$ — or a closed subspace thereof, depending on the application. Recent research [91, 195–198] has shown that in some cases, using a subset of $k \ll \dim(\mathcal{H})$ random states $|\psi_k\rangle$ sampled from $\mathcal{H}$, and minimising the sum of their energies

$$\min_{\mathcal{C}} \sum_k \langle \psi_k | U^\dagger \mathcal{C} \tilde{H} \mathcal{C}^\dagger U | \psi_k \rangle \tag{3.2}$$

is a sufficient condition to get unitary equivalence. However, this may only hold for highly structured operators. Therefore, in this chapter, cost functions building on Ref. [92] are used to probe all states in the Hilbert space simultaneously, rather than restricting them to a random sample thereof.

Starting from Eq. (3.1) and Fig. 3.1, it is possible to achieve full unitary equivalence of U and $\mathcal{C}$ by exploiting the Choi–Jamiołkowski isomorphism [199, 200]. From an all-zero input, first the maximally entangled state $|\psi'_0\rangle = \sum_k |k\rangle \otimes |k\rangle$ is created. Then the action of $\mathcal{C}^\dagger U$ on every state in $\mathcal{H}$ can be mapped to the action of $\mathcal{C}^\dagger U \otimes \mathbb{1}$ on the single state $|\psi'_0\rangle$ in the space $\mathcal{H} \otimes \mathcal{H}'$, where $\mathcal{H}'$ is a copy of $\mathcal{H}$. Figure 3.2 shows a circuit construction of this method, which is equivalent to the Hilbert-Schmidt test introduced in Ref. [92]. The output state $|\psi'_1\rangle$ is proportional to the input state $|\psi'_0\rangle$ if and only if $\mathcal{C} \sim U$. Therefore, applying the inverse of the preparation circuit that created $|\psi'_0\rangle$ returns the state to the original computational all-zero state if indeed $\mathcal{C} \sim U$. In order to estimate how close the result is to this ideal, any Hamiltonian $\tilde{H}$ which has $|0\rangle_{\mathcal{H}} \otimes |0\rangle_{\mathcal{H}'}$ as its unique and gapped ground state can be used as an artificial Hamiltonian. Because an expectation value of $\langle \psi_1 | \tilde{H} | \psi_1 \rangle = 0$ exactly corresponds to $U \sim \mathcal{C}$ by construction, energy minimisation techniques can be used to find a circuit $\mathcal{C}$ which is equivalent to a given unitary U for all input states. Note that the choice $\tilde{H} = |0\rangle\langle 0|_{\mathcal{H}} \otimes |0\rangle\langle 0|_{\mathcal{H}'}$ would exactly recover the (global) Hilbert-Schmidt test, which has a target expectation value of $\langle \tilde{H} \rangle = 1$, as opposed to 0 which is used in all other discussion presented in this chapter.

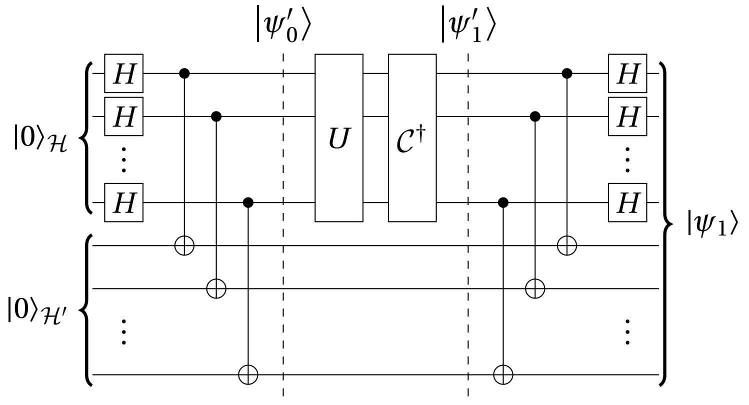


Figure 3.2: The circuit setup equivalent to a Hilbert-Schmidt test in Ref. [92], which was used to synthesise $\mathcal{C} \sim U$. The target is $|\psi_1\rangle \sim |0\rangle_{\mathcal{H}} \otimes |0\rangle_{\mathcal{H}'}$.

As shorthand notation for the cost function, $\langle \tilde{H} \rangle$ will be used to mean the expectation value $\langle \psi_1 | \tilde{H} | \psi_1 \rangle$ in the full augmented space with $|\psi_1\rangle$ as produced by the circuit in Fig. 3.2. Possible choices for $\tilde{H}$ are discussed in the next subsection.

Using $\langle \tilde{H} \rangle$ to determine unitary equivalence ignores global phase factors by which U and $\mathcal{C}$ might differ. In most scenarios, this is desirable, since such a global phase is physically irrelevant. However, if the resulting $\mathcal{C}$ is to be used in a controlled fashion within a larger circuit, a global phase mismatch between U and $\mathcal{C}$ becomes a physically relevant relative phase. In this case, an additional phase gate with an appropriate parameter must be added to $\mathcal{C}$ at the end of the synthesis process. A simple Hadamard test [201] can be used to determine the appropriate parameter of this final phase gate.

3.1.2 Synthesis Hamiltonians

When expressing the condition of unitary equivalence as an energy minimisation problem, there is some freedom in choosing an appropriate $\tilde{H}$, as any gapped Hamiltonian with the all-zero computational basis state as its ground state can be used. In this analysis, the two Hamiltonians used are

$$H_{\text{sum}} = \frac{1}{2} + \frac{1}{2N} \sum_k \sigma_k^z \tag{3.3}$$

where σ_k^z is the Pauli-z operator acting on qubit k, and N is the total number of qubits, and

$$H_{\text{proj}} = \mathbb{1} - |0\rangle\langle 0| . \tag{3.4}$$

While H_{proj} corresponds — as previously mentioned — to the *global* Hilbert-Schmidt test, H_{sum} is more closely related to the *local* Hilbert-Schmidt test [92], and the local cost function proposed in [202]. While both of these Hamiltonians have a ground state energy of 0 and maximum energy of 1, they differ in the energy structure of their excited states, and thus also in how they measure the closeness of a circuit is to a target unitary.

The projector-based H_{proj} simply measures the overlap with the desired all-zero state, while assigning every other product state the same maximum energy of 1.

On the other hand, the sum-based H_{sum} assigns each computational basis state an energy proportional to its Hamming distance [203] from the ground state, i.e. it measures how many bit flips would be necessary to get to the desired all-zero state. The state $|0\ldots01\rangle$ therefore has a lower energy than the state $|1\ldots11\rangle$ when measured with H_{sum}, while H_{proj} evaluates them as being at equal distances from the target.

The choice of Hamiltonian can have a significant impact on the difficulty of finding the optimal parameters. For instance, Refs. [92, 202], discuss that global cost functions like H_{proj} are very likely to come across barren plateaus during the parameter optimisation, while employing local costs — such as H_{sum} — is much less prone to such problems.

Even though in this work only H_{sum} and H_{proj} are considered, many other Hamiltonians may be used. Depending on the application, they can be designed to emphasise different properties of what constitutes a *good* output state, and may penalise some highly undesirable properties more harshly than others.

3.1.3 Subspace compilation

In some cases, only a small subspace of a unitary is relevant to the task at hand. One example would be the time evolution operator under a particle-number conserving Hamiltonian, when starting from a state with a specific number of particles. For these instances, it is not necessary and — from the point of view of minimising the total required resources — even undesirable to synthesise a circuit implementing the entire unitary. Such a unitary would have the form

$$U = U_1 \oplus U_2 \oplus U_3 \oplus \cdots \oplus U_n,$$

where U_j are unitaries in the respective subspaces, and $\oplus$ is their direct sum. The task is then, given U and the subspace of interest as a set of basis states, to synthesise a circuit that correctly implements one of the unitaries U_j, without regard for the rest of the operator.

This protocol will be referred to as *subspace compilation*. Provided the size of the subspace that U_j operates on is significantly smaller than the full space of the operator U, the described method can significantly decrease the number of gates in the synthesised circuit, as well as arrive at its result in much less time than synthesis of the full unitary.

For simplicity and without loss of generality, assume the unitary is only composed of two blocks, $U = U_1 \oplus U_2$, where U acts on the full Hilbert space $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2$, and U_1 and U_2 act on the subspaces $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively.

In this setup, it is sufficient to create a copy of the subspace $\mathcal{H}_1$ in order to synthesise a circuit recreating the action of U_1. This is in contrast to the full space compilation described above, where a copy of $\mathcal{H}$ was required. In the augmented space $\mathcal{H} \otimes \mathcal{H}_1'$, the creation of Bell pairs in Fig. 3.2 is then replaced by a custom preparation operator P, which has the action

$$|\Phi\rangle_{\mathcal{H}\mathcal{H}_1'} := P\,|0\rangle_{\mathcal{H}} \otimes |0\rangle_{\mathcal{H}_1'} = \frac{1}{\sqrt{d_1}} \sum_{s_j \in \mathcal{S}} |s_j\rangle_{\mathcal{H}}\,|j\rangle_{\mathcal{H}_1'}, \qquad (3.5)$$

with d_1 being the dimension of the subspace $\mathcal{H}_1$, and $\mathcal{S}$ being an orthonormal basis in $\mathcal{H}_1'$. Note that P maximally entangles $\mathcal{H}$ and $\mathcal{H}_1'$, i.e. $\mathrm{Tr}_{\mathcal{H}}\,|\Phi\rangle\langle\Phi|_{\mathcal{H}\mathcal{H}_1'} = \mathbb{1}_{\mathcal{H}_1'}$.

From here, the compilation procedure is the same as for the full-space case, where first the target U, and then the inverse of the circuit candidate $\mathcal{C}^\dagger$ are applied. Finally, before the measurement in the computational basis, the state preparation must be reverted using $P^\dagger$. Figure 3.3 shows the circuit setup of this method, which can be seen as an interpolation between Fig. 3.1 and Fig. 3.2.

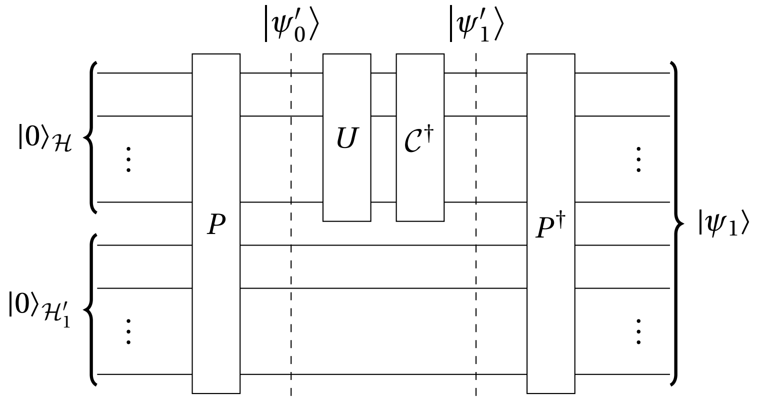


Figure 3.3: One cost evaluation in a subspace compilation. The unitary P prepares a maximally entangled state $|\Phi\rangle_{\mathcal{H}\mathcal{H}_1'}$ according to Eq. (3.5).

3.2 Ab initio circuit synthesis

In this section I discuss algorithms and subroutines used to vary the structure of ansatz-circuits and their parameters in order to minimise the expected energy under specific Hamiltonians, thereby solving various circuit synthesis and VQE problems. Notation and definitions for our synthesis protocols are introduced in the very next subsection, while Sections 3.2.2 to 3.2.5 describe procedures and subroutines, which are then used within the algorithms laid out in Section 3.2.6.

3.2.1 Formalism

The building blocks of the produced circuits are primitive gates G_k, which can be single-qubit rotations, multi-qubit rotations, SWAPs, etc., acting on various sets of qubits. In the following, it is assumed that every gate has a classical parameter, e.g. a rotation angle, associated with it. The formalism is easily extended to also include non-parametrised gates; they are simply gates whose parameter is permanently fixed. In the terminology used here, altering the parameter associated with a gate does not constitute a replacement of the gate itself.

A specific set of gates $\mathcal{L} = \{G_k\}$ is referred to as a *gate library*. Note that each G_k specifies which qubits the gate acts upon; for example, the Pauli-x rotations on different qubits 1 and 2 — R_1^x and R_2^x respectively — would be considered two separate gates G_k. This allows a gate library to include information about only locally available gates and qubit connectivity.

A *circuit structure* (or *ansatz circuit*) $\mathcal{C}$ can be represented by an ordered sequence of such primitive gates

$$\mathcal{C} := \big(\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{N-1}\big), \tag{3.6}$$

where $\mathcal{C}_k \in \mathcal{L}$.

Before the circuit can be applied to a quantum state, a parameter vector $\underline{\theta}$ containing the parameter θ_k for each gate $\mathcal{C}_k$ must be assigned. In the following discussion this is written as

$$\mathcal{C}(\underline{\theta}) := \big(\mathcal{C}_0(\theta_0), \mathcal{C}_1(\theta_1), \ldots, \mathcal{C}_{N-1}(\theta_{N-1})\big).$$

Applying a circuit $\mathcal{C}(\underline{\theta})$ to a state $|\psi\rangle$ consequently means evaluating

$$\mathcal{C}_{N-1}(\theta_{N-1}) \cdot \ldots \cdot \mathcal{C}_1(\theta_1) \cdot \mathcal{C}_0(\theta_0)\, |\psi\rangle\,.$$

A given circuit $\mathcal{C}(\underline{\theta})$ can be modified in two fundamentally different ways. One is to change the parameters $\underline{\theta}$, which will be called *parameter optimisation*. The other is to add or remove gates to or from the circuit structure $\mathcal{C}$, which will be referred to as *circuit structure modifications*. The procedures for how these modifications are performed are explained in detail in the following subsections. The routines are also given as simplified versions in high-level pseudocode in Appendix A.2, which might miss some performance enhancing tweaks for the sake of clarity and brevity.

3.2.2 Parameter optimisation

For a given circuit structure $\mathcal{C}$ containing parameterised gates — introduced above by the name *ansatz circuit* — the task is to to find the parameter vector $\underline{\theta}$ which minimises the expected energy of the artificial Hamiltonian $\tilde{H}$, i.e. the cost function. At this minimum, the circuit $\mathcal{C}(\underline{\theta})$ most closely (according to the used Hamiltonian $\tilde{H}$) approximates the desired unitary U within the scope of its parameter space. In this analysis, imaginary time evolution (ITE) [100, 204], a close relative of *quantum* *natural gradient optimisation* [99], is used, but in a slightly modified version. First, the matrix object[8]

$$A_{ij} = \mathrm{Re}\Big(\langle \partial_i \psi | \partial_j \psi \rangle - \langle \partial_i \psi | \psi \rangle \langle \psi | \partial_j \psi \rangle\Big), \tag{3.7}$$

which will be referred to as the *quantum metric tensor (QMT)* [99, 205, 206], and the *gradient vector*

$$B_i = -\langle \partial_i \psi | H | \psi \rangle \tag{3.8}$$

are computed. The time evolution of the parameter vector $\underline{\theta}$ is then given by [204]

$$\mathbf{A}\underline{\dot{\theta}} = \underline{B}. \tag{3.9}$$

Using the forward Euler method yields the update rule for the parameter vector

$$\underline{\theta}_{t+1} = \underline{\theta}_t - \lambda \mathbf{A}^{-1} \underline{B}. \tag{3.10}$$

The matrix $\mathbf{A}$ is often close to singular, so some regularisation method is required to stabilise the iteration. The results below are produced with Tikhonov regularisation, but other techniques may be used as well.

To find a suitable λ, an idea similar to the one presented in Ref. [194] is used. Each time step t starts from a small (arbitrary) initial value of $\lambda_0 = 0.05$. But, because the evaluation of the QMT and the gradient vector may be expensive operations [207] compared to the evaluation of the expected energy for a given set of parameters, it makes sense to try to find the minimum energy along the established search direction, before re-computing the QMT and gradient vector. In the proposed method, the step size is exponentially increased until a local minimum along the established step direction is found. This means repeatedly multiplying λ by some constant factor κ until the energy increases, and then accepting the immediately preceding λ-value. However, if the initial step size turns out to already increase the energy, λ is instead shrunk exponentially by a factor of $1/\kappa$ until a decrease in energy is found or a minimum step size is reached. This exponential search can relatively quickly traverse barren plateaus

[8]The shorthand notation used here means $|\partial_\mu \psi\rangle := \frac{\partial |\psi(\theta)\rangle}{\partial \theta_\mu}$.

and is almost always useful in emulators, where the gradient direction is known to high numerical precision. Therefore, in these settings, good initial parameters to avoid barren plateaus are less critical. The usefulness of this method in avoiding barren plateaus is reduced on real quantum hardware when shot noise limits how precisely the gradient direction can be determined.

To detect convergence, absolute and relative changes of the energy are used. The stopping criterion is that one of the conditions must be met a number $k_{\text{conv}} \sim 5$ of times. Algorithm 1 in Appendix A.2 shows the full procedure.

3.2.3 Introduction of new gates

Dynamic expansion of a given ansatz circuit has been explored before by VQE methods [108–110] and in more general contexts [90, 91, 195]. The formalism introduced here is very generic, but naturally shares some ideas with previous works, especially Ref. [91].

The algorithms introduced later rely on the concept of a *move*, which refers to the most basic possible modification of a circuit structure $\mathcal{C}$, i.e. the insertion of one additional gate at some position in the gate sequence. Given a library of gates $\mathcal{L}$ and a circuit with N gates, it is convenient to define a move as a tuple (G, n), with a gate $G \in \mathcal{L}$ and an index $0 \leq n \leq N$. Applying such a move to an existing circuit $\mathcal{C}$ means inserting gate G at position n.

$$\mathcal{C} \mapsto \left(\mathcal{C}_0, \ldots, \mathcal{C}_{n-1}, G, \mathcal{C}_n, \ldots, \mathcal{C}_{N-1}\right) \tag{3.11}$$

The routine ApplyMove in Algorithm 2 (Appendix A.2) performs exactly this action.

The gate and index of a specific move are denoted by subscripts G and n, respectively. For instance, if $M = (R_1^x, 4)$, then $M_G = R_1^x$ and $M_n = 4$. This definition allows for convenient passing around of moves between functions, which will be useful later.

For a circuit containing N gates, all possible moves using a library $\mathcal{L}$ are given by

$$\mathcal{M}_{\text{all}} = \left\{(G, n) \mid 0 \leq n \leq N \text{ and } G \in \mathcal{L}\right\}. \tag{3.12}$$

However, many of the moves in $\mathcal{M}_{\text{all}}$ will lead to redundancies in the circuit, because neighbouring identical gates acting on the same qubits can be merged straightforwardly. In the presented method, such modifications leading to obvious redundancies are eliminated from the set of possible moves, and only potentially useful moves are considered. The specific method used is given in Algorithm 2 (Appendix A.2) and works as follows. For a specific qubit k, only the gates in $\mathcal{C}$ acting on qubit k are considered. Potentially useful circuit modifications in this subset of the circuit are then insertions of gates from the library between consecutive pairs of gates which also act on qubit k, but are different to the previous and following gate also acting on qubit k. The set comprised of moves generated in this fashion — with k iterating over all possible qubits — contains all potentially useful moves.

In many cases, e.g. if some gates in $\mathcal{L}$ commute with one another, Algorithm 2 will still include moves that lead to redundancies. The quantum metric tensor as defined in Section 3.2.2 can be used to detect such expendable gates. The details of this operation are given in the next subsection.

3.2.4 Removal of superfluous gates

In a circuit $\mathcal{C}$, not all gates in the sequence necessarily contribute to generating the desired unitary in a useful way. In this analysis, three distinct techniques with varying computational cost and ability to detect such redundancies are used, which will be discussed in the following.

In the algorithms presented, these methods are used in the order *small parameter* $\rightarrow$ *QMT-assisted* $\rightarrow$ *trial and error*, as shown in Algorithm 4 (Appendix A.2). Each method can, in principle, also detect all redundancies of the previous methods,[9] but is more costly to perform, which makes this staged approach useful.

[9]In the generalisation of also allowing non-parametrised gates in the circuit — which is not discussed further — only *trial and error* may be used for those specific gates.

Small parameter removal

The computationally cheapest way to detect non-contributing gates is by checking the associated parameters after optimising them.[10] For every circuit parameter close to 0 modulo[11] 2π, $\theta_k \approx 0 \bmod 2\pi$, the corresponding gate $\mathcal{C}_k$ can be removed immediately.

Quantum metric tensor assisted removal

As a more sophisticated and computationally slightly more expensive method to detect further redundancies, the quantum metric tensor (QMT) as defined in Eq. (3.7) can be checked for linearly dependent rows.

The QMT contains information about how the output state changes with respect to varying the parameters. If rows i and j in this tensor are linearly dependent, the linearised actions of $\mathcal{C}_i$ and $\mathcal{C}_j$ are equivalent in the tangent space of the circuit $\mathcal{C}$ at the current position $\underline{\theta}$ in parameter space. Intuitively, this means that changes to the parameters θ_i and θ_j from their current values would move the state in the same direction inside some subspace of the full Hilbert space. While this does not guarantee that $\mathcal{C}_i$ and $\mathcal{C}_j$ have equivalent actions in the full Hilbert space $\mathcal{H}$ or at a different point $\underline{\theta}'$ in parameter space, it is a strong indication of it. Therefore, the condition

$$\left|(A_{i,\cdot}^{\mathsf{T}} \cdot A_{j,\cdot}) - \|A_{i,\cdot}\|\|A_{j,\cdot}\|\right| < \varepsilon_{\mathrm{QMT}}, \tag{3.13}$$

where $A_{i,\cdot}$ is the i^{th} row vector of $\mathbf{A}$, can be used as a heuristic for detecting potentially redundant gates with an appropriately small $\varepsilon_{\mathrm{QMT}}$.

At many points in the iteration the QMT is already known from the previously performed parameter optimisation and does not need to be explicitly re-calculated. As calculating this matrix is computationally relatively expensive, this saves valuable computing time.

The removal is performed as follows. For all pairs of rows i, j the condition in Eq. (3.13) is checked. If it is fulfilled, the parameter of the first gate is adjusted to $\theta_i \leftarrow \theta_i + \theta_j$ and the gate $\mathcal{C}_j$ is removed from the circuit $\mathcal{C}$, as well as θ_j from the

[10]It is assumed that every gate G approaches the identity for small parameters, i.e. $\lim_{\theta\to 0} G(\theta) = \mathbb{1}$.
[11]For practical reasons the slightly unusual definition of $a \bmod b = a - b\lfloor a/b + .5\rfloor$ is used, which returns values in the interval $[-b/2, b/2)$ instead of the usual $[0, b)$.

parameter vector $\underline{\theta}$. If this removal does not significantly increase the energy, the new circuit is kept, otherwise the deletion is reverted. Among other redundancies, this method allows for the relatively easy detection of identical gates separated by gate sequences they (non-trivially) commute with.

Trial and error removal

The above method is good at finding redundancies where one gate can absorb a different one into its parameter. It cannot, however, detect cases where multiple gates need to adjust their parameters in order to compensate the removal of one specific gate. For these cases a third strategy is employed, which is computationally more expensive, but can also detect much more subtle redundancies.

Given a circuit $\mathcal{C}$ and a set of gate indices $\mathcal{R}$ to be considered candidates for removal, for each $k \in \mathcal{R}$, the gate $\mathcal{C}_k$ is deleted from the circuit without replacement, and a full parameter optimisation as in Section 3.2.2 is performed with the modified circuit. If the energy after the relaxation is not significantly higher than before, the deleted gate is considered redundant and remains removed. I will refer to this method as a *hard removal*, because the circuit is abruptly taken to a different point in parameter space, potentially far away from a local minimum.

An alternative to the aforementioned *hard removal* explored but not reported here, is a method which may be called *soft removal*. The parameter θ_k, for $k \in \mathcal{R}$, is shifted towards zero by some predetermined amount, and a single imaginary time step for all parameters except θ_k is performed right afterwards, with the goal of driving the parameters towards a new minimum close by. This procedure is repeated until the parameter is close to zero, and only then is the gate completely removed. This allows the circuit to stay close to the local minimum already it is in. Therefore, this method can occasionally lead to better results.

3.2.5 Initial circuit

When the gates in the considered library $\mathcal{L}$ have limited connectivity, i.e. not every qubit can interact with every other qubit directly, finding useful circuit structure modifications by randomly adding gates can become increasingly improbable. For example, in a linear chain of qubits with nearest-neighbour connectivity, having qubit 0 interact with qubit 3 requires the correct simultaneous addition of at least three gates. The effort of finding the correct combination of three gates is further hampered by the fact that — at least for the artificial Hamiltonians $\tilde{H}$ used in the circuit synthesis tasks presented here — the energy landscape with regard to adding only a subset of those gates is flat.

To alleviate these connectivity limitations, it can be helpful to choose not to start with a completely empty circuit, but to have an initial structure of parameterised SWAP gates, where every qubit can be close to every other qubit at some point, and is able to — but does not need to — subsequently return to its original position. This can be achieved by an ansatz of the form

$$\mathcal{C}^{(0)} = \prod_{n=0}^{N} \left[\prod_{k=1}^{\lceil N/2-1 \rceil} \mathcal{S}_{2k,2k+1}(\pi/2) \prod_{k=0}^{\lfloor N/2-1 \rfloor} \mathcal{S}_{2k,2k+1}(\pi/2) \right] \tag{3.14}$$

where N is the number of qubits and

$$\mathcal{S}_{i,j}(\theta) := \exp\left(i\frac{\theta}{2}\text{SWAP}_{i,j} \right) \tag{3.15}$$

is a gate which performs no action for $\theta = 0$ and swaps the qubits i and j if $\theta = \pi$. The ansatz is visualised in Fig. 3.4, where the $\mathcal{S}$ gate is indicated by the usual SWAP symbol, but drawn with dotted lines.

The $\mathcal{S}_{i,j}$ gates are typically not part of the gate library $\mathcal{L}$ and must be compiled to it separately. If all parameters converge to either 0 or π, it is sufficient to compile the SWAP gate to the desired gate library $\mathcal{L}$. Otherwise, a generic expression for $\mathcal{S}_{i,j}(\theta)$ must be found, or each instance must be compiled separately.

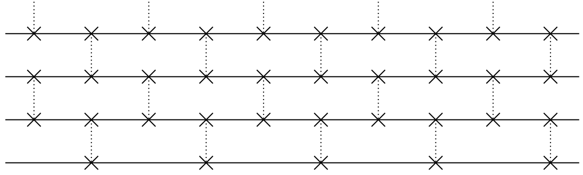


Figure 3.4: The initial circuit used for gate libraries $\mathcal{L}$ where two-qubit gates are limited to nearest neighbours.

3.2.6 Circuit structure finding

This section elaborates on the algorithms later used to synthesise quantum circuits, all of which are adaptions of well-known optimisation techniques.

Hill climbing

The simplest and most straightforward algorithm employed here is a variant of hill climbing [208], which iteratively searches some *neighbourhood* of the current circuit structure, and accepts the lowest energy solution within that neighbourhood. In the present case, the neighbourhood of a circuit structure is defined as all circuits reachable by applying N_{moves} *moves* as introduced in Section 3.2.3. For the purposes of this analysis of hill climbing, only $N_{\text{moves}} = 1$ is considered.

Starting from some initial circuit $\mathcal{C}^{(0)}$, all moves are tried, their parameters optimised according to Section 3.2.2, and the energies of the resulting circuits are recorded. The move which resulted in the lowest energy is kept, and the structure resulting from it becomes the new circuit $\mathcal{C}^{(1)}$. This procedure is repeated until the energy falls below a given threshold, or none of the possible moves decrease the energy. Removal of non-contributing gates is only performed once at the end of the iteration. Algorithm 5 (Appendix A.2) shows this procedure.

Because it only checks the immediate neighbourhood of the current circuit for improvements, this method quickly gets stuck in local minima of the cost function. This problem is typical for hill climbing algorithms. In some very limited cases, it is possible to extend the search radius to all combinations of two or more sequential steps, i.e. $N_{\text{moves}} > 1$, in order to escape such local optima. However, in most cases the search space volume grows very rapidly with the search depth, making larger search radii impractical.

Random search

To be able to escape local minima in which hill climbing gets stuck, it is necessary to make larger steps in configuration space. However, as mentioned in the previous section, the size of the neighbourhood grows too rapidly to exhaustively search it. It is therefore sensible to resort to a variant of random search [208], which in each iteration proposes a random modification to the circuit, and accepts it if it lowers the energy.

A single proposed modification — this will be called a *random step* — consists of applying a number of N_{moves} randomly chosen *moves* to the circuit, and only then optimising its parameters. This essentially means adding N_{moves} random gates to the circuit. Such a random step moves the current ansatz further in the space of circuit structures and thus has the ability to escape from local optima. Because of the randomness involved when selecting new moves, many of the added gates are potentially not contributing to the reduction in energy. Therefore, removing as many of the newly added gates as possible after each random step via the methods discussed in Section 3.2.4 is essential to keep the circuit efficient.

When drawing a random move, some gate types (e.g. controlled rotations) might be overrepresented compared to others (e.g. local rotations), simply because there are more of them. This also skews the number of gates by type in the final circuit, which seems to sometimes hinder performance. As a measure to counteract this phenomenon, the moves may be sorted into groups. To draw a random move, a group is chosen according to an appropriate probability distribution, followed by the selection of a specific move from the chosen group with uniform probability.[12] For the results produced in this work only two such groups were employed: one containing moves with single-qubit gates, and the other including those with two-qubit gates. Both were given equal probability.

Instead of simply accepting every random step that lowers the cost function, it furthermore proved beneficial to sample a small number $N_{\text{samp}} \sim 10$ of random steps and choose to keep only the step resulting in the lowest cost. Algorithm 3 (Appendix A.2) shows the full procedure.

Tabu search

As an extension to random search — which relies purely on chance to find useful modifications to the circuit — a variant of tabu search [209] was also used in an effort

[12]For clarity, this detail is omitted in Algorithm 3.

to avoid repeated application of unsuccessful circuit modifications. This section briefly outlines the idea behind the algorithm and its potential pitfalls.

The overall structure of this variant of tabu search is exactly the same as for random search in Algorithm 3, but with $N_{\text{samp}} = 1$. It has however, the additional feature of a *tabu list*, which will be called τ. Whenever a move is performed, it is recorded into this list, together with a label recording at which iteration the move was performed. For a number of iterations $T_{\text{tabu}} \sim 20$, the same move must then not be repeated. Therefore, before choosing a random move, all tabu moves are removed from the set of potential moves.

To keep the action of the moves consistent when positions in the circuit change due to the insertion or deletion of gates, the insertion indices $M_n\ \forall M \in \tau$ in the stored moves are updated accordingly whenever the circuit changes. This means decrementing all stored indices $M_n > m\ \forall M \in \tau$ by 1 if the gate with index m is deleted, and incrementing all indices $M_n \geq m\ \forall m \in \tau$ by 1 when a new gate at index m is inserted.

The additional feature of a tabu list has the potential of steering the search away from non-improving moves and increasing the likelihood of choosing useful moves. Its practical advantage proves to be limited, as is shown in the Results section.

3.3 Results

This section contains the numerical findings for different applications of the methods discussed above. Unless otherwise noted, the calculations use the default hyperparameters listed in Table A.1 in Appendix A.1.

3.3.1 Cost functions as proxy for unitary equivalence

To determine whether two operators[13] $\mathcal{C}$ and U are equivalent, an appropriate metric to use is the global-phase invariant operator norm of their difference

$$\begin{aligned}\mathcal{D}(U,\mathcal{C}) &:= \min_{\phi} \left\| U - e^{i\phi}\mathcal{C} \right\| \\ &= \min_{\phi}\left[\max_{|\psi\rangle} \left\| U|\psi\rangle - e^{i\phi}\mathcal{C}|\psi\rangle \right\|_2\right], \end{aligned} \tag{3.16}$$

which in this work will be referred to as the *operator distance*. It is the largest possible L_2-norm of the difference (and thus the Euclidean distance) between the desired output $U|\psi\rangle$ and the output of the recompiled version $\mathcal{C}|\psi\rangle$.

For the calculations only considering a subspace, it is sensible to also define

$$\mathcal{D}_{\mathcal{S}}(U,\mathcal{C}) := \mathcal{D}(\Pi_{\mathcal{S}} U \Pi_{\mathcal{S}}, \Pi_{\mathcal{S}} \mathcal{C} \Pi_{\mathcal{S}}) \tag{3.17}$$

where $\Pi_{\mathcal{S}}$ is the projector onto the relevant subspace.

Throughout the results the expected energy of H_{sum} or H_{proj} as defined in Eqs. (3.3) and (3.4) is used to assess how closely a constructed circuit $\mathcal{C}$ reproduces the desired unitary U. Note, however, that in general $\langle\tilde{H}\rangle \not\propto \mathcal{D}(\mathcal{C},U)$. This is easily demonstrated with an n-controlled Pauli-z gate $C_{0..n-1}[\sigma_n^z]$, for which the identity operator yields an energy of $\langle H_{\text{proj}}\rangle = \langle H_{\text{sum}}\rangle = 2^{-n}$, but the operator distance has a macroscopic value of $\mathcal{D}(\mathbb{1}, C_{0..n-1}[\sigma_n^z]) = \sqrt{2}$. The energy is therefore not guaranteed to correspond to a small operator distance. However, numerical evidence presented momentarily suggests that it is still a good measure in most cases.

To gain confidence in the fact that the cost functions are indeed a good stand-ins for the operator distance, a large number of results was surveyed. For the final circuits of many 5-qubit quantum Fourier transform (QFT) synthesis outcomes, the cost functions of $\langle H_{\text{sum}}\rangle$ and $\langle H_{\text{proj}}\rangle$ were calculated and compared to the respective operator distance $\mathcal{D}$. The results are plotted in Fig. 3.5 and show that close to convergence the used cost functions are both very good proxies for the actual operator distance. It is therefore reasonable to use the described cost functions to assess the quality of the recompiled circuit.

[13]From here on we will omit explicit indication of the parameter vector $\underline{\theta}$ wherever practical.

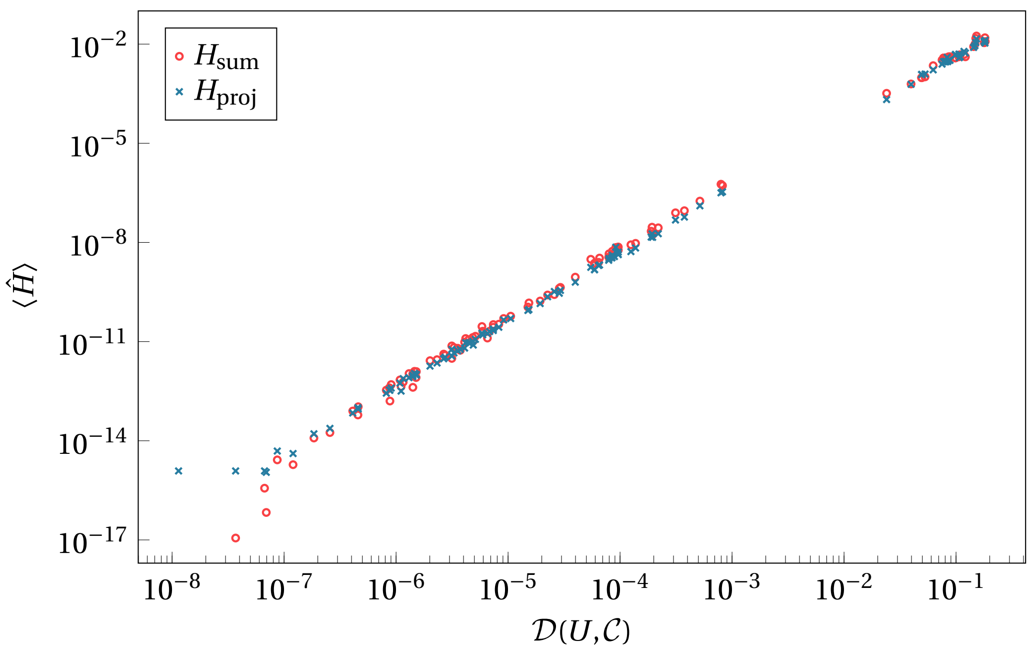


Figure 3.5: Operator distance between the target U and recompiled $\mathcal{C}$ unitaries for a set of recompiled QFT circuits on 5 qubits. The data points at very small costs are in the range of numerical noise and likely influenced by finite data type accuracy. The large gap in data around $\mathcal{D} \approx 10^{-2}$ is caused by the separation between converged (reached critical circuit expressibility) and not converged (got stuck in a local minimum) calculations, i.e. if a calculation got stuck, it was usually at a relatively high energy.

3.3.2 Random unitaries

Dense unitaries

To test and benchmark the circuit synthesis protocols, uniformly Haar-distributed random unitaries were created[14] as targets, and calculations to find circuit representations for them were carried out. For each target on N qubits, a single attempt was made to create a circuit performing the same action using the gate set of

$$\mathcal{L}_{\text{allrot}} = \left\{R_k^\sigma, C_\ell[R_k^\sigma] \,|\, \sigma \in \{x,y,z\}, k,\ell \in [N] \text{ and } k \neq \ell\right\}, \tag{3.18}$$

where $[N] \equiv \{1,\ldots,N\}$, which contains all local single-qubit Pauli rotations R_k^x, R_k^y, and R_k^z on each qubit k, as well as single-controlled versions thereof with no connectivity constraints. The results of 100 random unitaries per method and number of qubits are shown in Fig. 3.6.

[14]The unitaries were generated using the method `scipy.stats.unitary_group` in Python and the function `CircularUnitaryMatrixDistribution` in Mathematica.

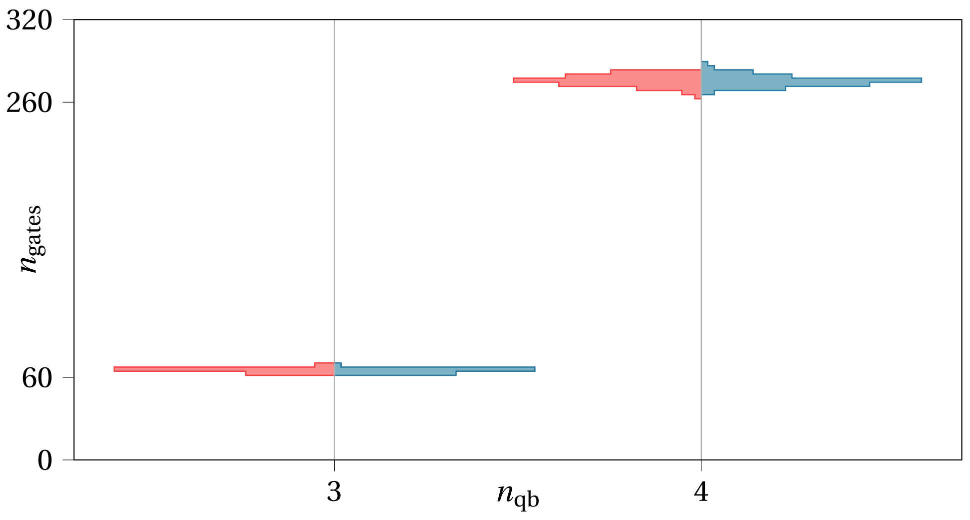


Figure 3.6: Histograms of the number of gates required to express Haar-distributed unitaries as a circuit containing only gates from $\mathcal{L}_{\text{allrot}}$ for different numbers of qubits. Left (red) data was obtained using tabu search, right (blue) data is from the random search algorithm.

The number of gates in the circuits produced by random and tabu search correlates well with the increasing degrees of freedom of the targets, which for a dense unitary on N qubits is 2^{2N}. Therefore, for 3 qubits, it is expected that on average no fewer than 64 gates are required, and a mean number of 65 gates was found in the synthesised circuits. For 4 qubits, a mean of 275 gates was observed, while ideally 256 would be expected. This indicates that, at least for dense, unstructured unitaries, the circuits produced by tabu and random search do not contain a large number of superfluous gates.

Note that the results show no significant difference between tabu and random search, which is discussed in more detail in Section 3.3.3.

Subspace compilation

In order to assess the efficacy of compiling only in a certain subspace, another random unitary operator was generated, but this time with more structure. The operator is block-diagonal in the computational basis within subspaces of equal Hamming weight,[15] i.e. the number of ones and zeros in a state is conserved upon applying this operator. Each of these blocks of equal Hamming weight was populated with a Haar-distributed random unitary. To compare the subspace compilation to compiling the

[15]The Hamming weight of a binary number (like a computational basis state) is the number of set bits, i.e. the count of ones.

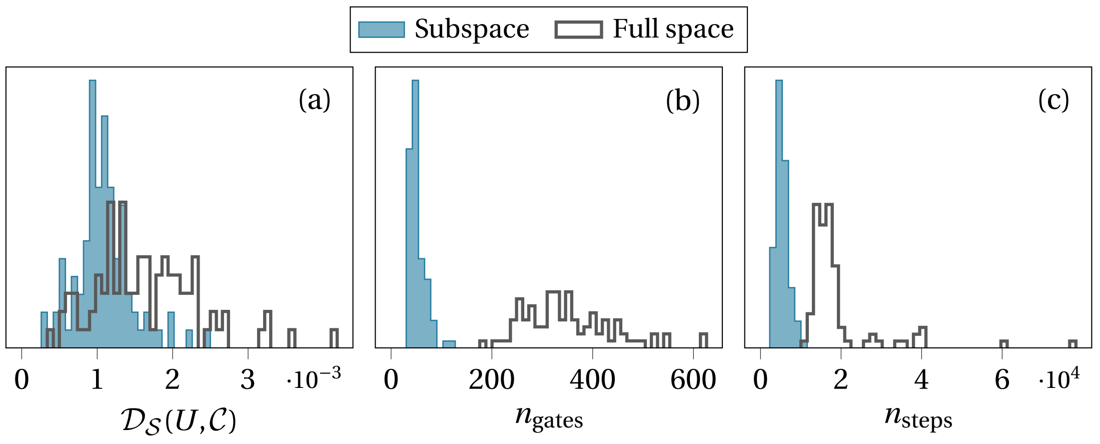


Figure 3.7: Normalised histograms showing properties of synthesised circuits $\mathcal{C}$ recovering the action of single 4-state block U_1 within a block-diagonal operator U on 4 qubits. The blocks making up U are each Haar distributed random unitaries. Empty histograms in grey ▭ show results for the full space method, filled histograms in blue ▬ show subspace results. Circuits for same unitary U were synthesised 100 times. (a) Operator distance as in Eq. (3.16) within the subspace $\mathcal{H}_1$. (b) Number of gates in the resulting circuit. (c) Number of total imaginary time evolution steps needed by the algorithm.

full unitary, both variants were employed (random search being the structure search variant), with the subspace calculations only addressing the block with Hamming weight 1, i.e. the states $\{|0001\rangle, |0010\rangle, |0100\rangle, |1000\rangle\}$. For numerical reasons — and because at this problem size barren plateaus proved not to be an issue — the cost function $\langle H_{\text{proj}}\rangle$ was used in this example. The target energy was set to $\langle H_{\text{proj}}\rangle \leq 10^{-5}$, and other hyperparameters were $N_{\text{moves}} = 30$ and $N_{\text{samp}} = 10$. Because of the stochastic nature the compilation process, a total of 100 synthesis attempts were made each in the full- and subspace, of which 97 in the subspace and 95 in the full space converged. Figure 3.7 summarises the outcomes of the successful calculations.

The results distinctly show that for comparable accuracy (a), considering only a subspace results in vastly fewer gates (b), as well as many fewer iterations of the algorithm (c). Therefore, it is reasonable to expect this method to yield much better results whenever the target unitary conserves some quantity, and the relevant subspace is known in advance. Prominently, one example for this case is time evolution operator in quantum chemistry. There, the number of electrons is a conserved and known quantity, and corresponds to the Hamming weight of the states. Reference [210] explores this application in greater detail.

3.3.3 Quantum Fourier transform

As shown in the previous subsection, when compiling random unitaries, the number of required parameters quickly makes compiling circuits with limited available resources for more than a few qubits infeasible. However, practically relevant circuits usually have much more structure than random unitaries. Therefore, as examples of unitaries closer to real-world applications, quantum Fourier transform (QFT) [26] circuits were also synthesised using various gate sets. Figure 3.8 shows the target circuit of a 4-qubit QFT as an example. It is re-expressed using the established gate set $\mathcal{L}_{\text{allrot}}$, as well as

$$\mathcal{L}_{\text{NNrot}} = \left\{R_k^\sigma, C_\ell[R_k^\sigma] \,|\, \sigma \in \{x,y,z\},\ k,\ell \in [N] \text{ and } |k-\ell| = 1\right\} \tag{3.19}$$

which contains all local single-qubit rotations and local rotations controlled by nearest neighbours in an open linear chain topology. Furthermore, to test the ability of the proposed algorithms to work with a much more restricted gate set, QFT circuits were synthesised using the library

$$\mathcal{L}_{\text{SWAP}} = \left\{R_k^\sigma, \mathcal{S}_{k,\ell} \,|\, \sigma \in \{x,y,z\},\ k,\ell \in [N] \text{ and } |k-\ell| = 1\right\}. \tag{3.20}$$

This set, in addition to local rotations, contains the parameterised SWAP gate introduced in Eq. (3.15) between neighbouring qubits as the only entangling operator. None of these gate sets contain Hadamard or controlled phase gates — which constitute the majority of the gates in the canonical circuit — making this a suitable benchmarking synthesis task.

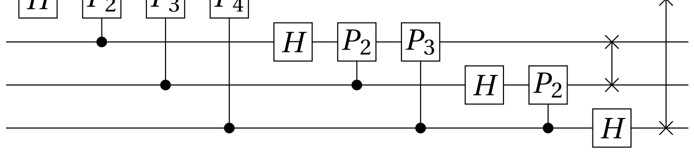


Figure 3.8: Example of a quantum Fourier transform circuit on four qubits as used in the present calculations. The controlled operators $P_n = e^{-i\pi(\sigma^z - \mathbb{1})2^{-n}}$ are phase gates with rotation angles of $2\pi/2^n$.

For demonstration purposes, QFT circuits for 3 to 6 qubits were synthesised, using all mentioned gate sets, and all discussed circuit structure generation algorithms, 100 times each. The target energy for the used cost function was $\langle H_{\text{sum}} \rangle \leq 10^{-8}$. The results are shown in Fig. 3.9. In addition to the outcomes plotted in Fig. 3.9, 10 more calculations were performed for each of $n_{\text{qb}} = 7 \dots 10$ qubits using tabu search and the $\mathcal{L}_{\text{allrot}}$ gate set. These resulted in the convergence of 8, 7, 5, and 1 calculations, respectively, and minimum gate counts of 72, 93, 114, and 138.

The results show that the *hill climbing* algorithm can perform well for some problems, especially when the available gate set has high expressibility as is the case for $\mathcal{L}_{\text{allrot}}$ and $\mathcal{L}_{\text{NNrot}}$. In these cases, hill climbing found circuits with gate counts close to the lower bound of all investigated algorithms, albeit at a higher computational cost than tabu and random search. However, more restricted gate sets like $\mathcal{L}_{\text{SWAP}}$ severely hamper its ability to find solutions at all. Being the only deterministic algorithm we presented, circuits it fails to synthesise cannot be helped by re-running the procedure.

The probabilistic schemes of *tabu search* and *random search*, on the other hand, produce different outcomes for every run. Using a fully connected gate set consistently yields a higher probability for convergence than restricting the interactions to neighbouring qubits in a linear chain, despite employing an initial circuit of SWAPs to counteract connectivity constraints. Calculations starting from an empty circuit (not plotted) show even lower convergence rates. Further decreasing the expressibility of the available gates by using $\mathcal{L}_{\text{SWAP}}$ as the gate set sees another significant drop in the relative number of converged calculations for $n_{\text{qb}} \geq 4$, indicating that all of the applied algorithms struggle to find solutions when they are greatly restricted in the choice of gates they can add.

Comparing the approaches of *tabu* and *random search*, shows — as for random unitaries — no significant difference in the number of gates in the resulting circuit or the convergence probability, despite tabu search trying to remember and avoid unsuccessful circuit modifications. It is to be expected for this mechanism not to have a noticeable impact when using libraries containing many gates, like $\mathcal{L}_{\text{allrot}}$, because the neighbourhood — i.e. the circuits which can be produced by adding a single gate from the library at any position — is much larger than the number of additions the algorithm can reasonably try during the iteration. Specifically, at every point in the

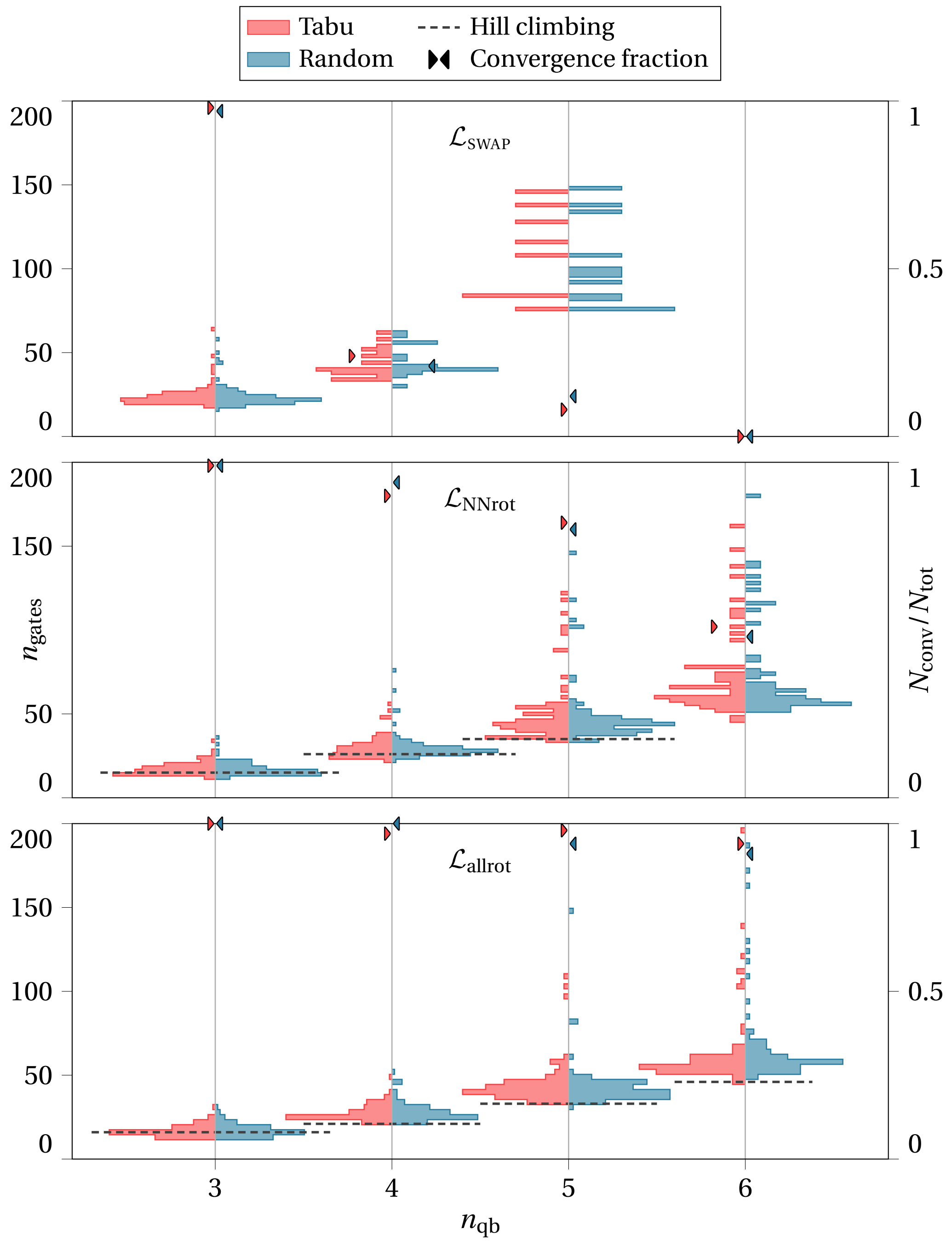


Figure 3.9: Number of gates and fraction of converged calculations for synthesis of a QFT circuit for $n_{\text{qb}} = 3\dots6$ qubits and different gate sets $\mathcal{L}_{\text{SWAP}}$, $\mathcal{L}_{\text{NNrot}}$, and $\mathcal{L}_{\text{allrot}}$. Shaded graphs are histograms of the number of gates in the resultant circuit, left, red 🟥 for tabu search, right, blue 🟦 for random search, centred around their respective number of qubits. Each calculation was run 100 times with identical parameters as listed in the text and appendix. Red and blue carets indicate the fraction of converged calculations for each instance according to the right scale. Separation from the central line is for visual purposes only. Grey dashed lines ----, if present, indicate that the hill climbing algorithm converged for a particular instance and shows the number of qubits in the final circuit it produced.

iteration, $N_{\mathcal{L}}(N_{\text{gates}}+1)$ different moves are possible, with the current circuit gate count N_{gates} and the number of gates in the library $N_{\mathcal{L}}$. This means, for example, on 5 qubits with an existing circuit containing 20 gates, the $\mathcal{L}_{\text{allrot}}$ gate set produces several hundred potential moves, of which the algorithms typically explore ~ 20. However, even when using the $\mathcal{L}_{\text{SWAP}}$ library with only very few gates in it, which thus produces a smaller neighbourhood for each circuit, the results show no significant difference between the two approaches. This observation is independent of how many iterations the *tabu* moves are remembered for.

3.3.4 Multi-qubit Toffoli

A potentially difficult operator to synthesise from only two-qubit gates is the n-qubit Toffoli gate, i.e. a Pauli-x gate with $n-1$ controls. The difficulty lies in the fact that it only acts on a very small subspace of $\mathcal{H}$, which is not straightforward to exclusively address using gates which act on much larger subspaces of $\mathcal{H}$. Additionally, while it is relatively straightforward to detect whether a given circuit can act exclusively in the given subspace, finding a measure indicating that a given circuit is *close* to having this property proves difficult. If this kind of measure were found, it could guide the compilation process in the right direction. Unfortunately, the used cost functions used in this analysis do not contain such information.

In the implementation used here, because a large portion of all possible input states must remain unchanged by the circuit, adding any small number of gates will likely result in their parameters being tuned to zero during optimisation, as this matches the correct action on most input states. The algorithm then subsequently removes the gates with vanishing parameters, leading to no progress being made.

Figure 3.10 shows the results of synthesis calculations to generate n-qubit Toffoli gates for $n = 3, 4, 5$ using the $\mathcal{L}_{\text{allrot}}$ gate library, 100 times each. For the smaller cases of $n = 3$ and 4, the algorithm can find correct circuits reliably. In these instances, the number of simultaneously added gates is large enough to overcome the previously discussed limitation. This is helped by the fact that the number of states on which the Toffoli acts like the identity operator is not overwhelmingly bigger than the number of

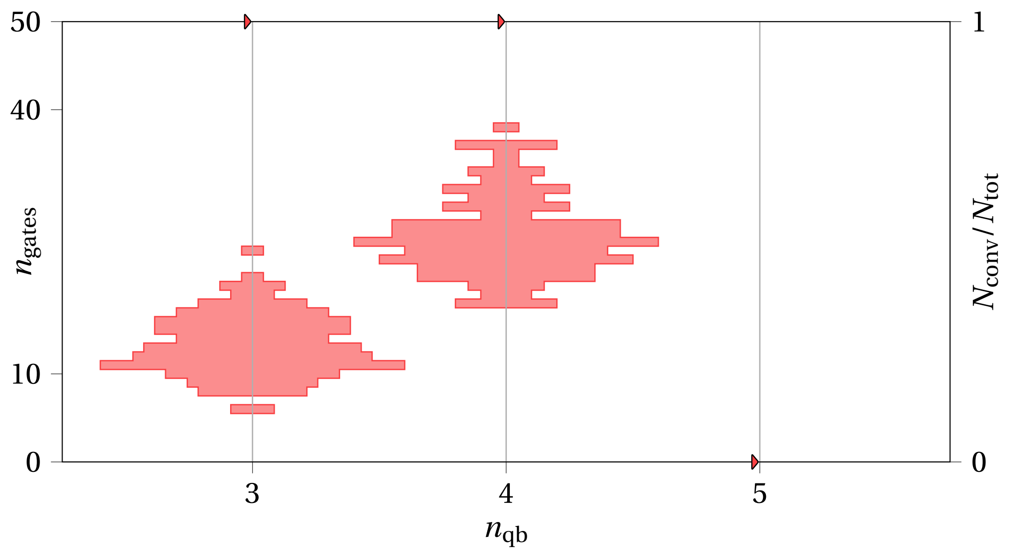


Figure 3.10: Histograms of the numbers of gates required and convergence fractions when synthesising an n_{qb}-qubit Toffoli using the $\mathcal{L}_{\text{allrot}}$ gate set and tabu search. Histograms are centred around the corresponding number of qubits, red carets indicate how many of the started calculations converged. Histogram data for 5 qubits is missing because none of the calculations succeeded.

states on which it acts nontrivially. Therefore, in these cases, an appropriate circuit can be synthesised virtually every time. For $n \geq 5$, on the other hand, the algorithm was not able to find any solutions at all.

As an attempt to inject some prior knowledge into the method, 5-qubit Toffoli gate synthesis was also performed by starting the iteration from one of the successful 4-qubit Toffoli synthesis results. With this "warm start" technique, 80 out of 100 calculations converged. While such an assisted start deviates from the strict *ab initio* framework — even more so than the initial SWAP network used in some of the calculations — and requires a specific incremental structure of the target circuit, it can still be a very useful resource for some tasks.

3.4 Discussion

In this chapter, a variant of the Hilbert-Schmidt test [92] was combined with artificial Hamiltonians H_{sum} and H_{proj} similar to Ref. [194] to construct cost functions representing the closeness of a unitary to a dynamically created circuit, where the cost can in principle be evaluated on a quantum computer. However, only emulators of such quantum hardware were employed in the generation of the results, thus circumventing some practical challenges like shot noise and barren plateaus, whose impact on the performance on the presented scheme remains to be investigated. This question of how noisy hardware would affect the results is left open as a potential future research direction. Three different algorithms which use this cost function to dynamically construct quantum circuits replicating the action of a given unitary from the ground up were presented and their performance on various synthesis tasks was demonstrated.

The obtained results suggest that the presented algorithms are able to generate circuit representations for dense random unitaries with gate counts close to what would be expected to be optimal, based on the degrees of freedom[16] in such a unitary. For block-diagonal random unitaries, the numerical evidence furthermore shows that generating a circuit whose closeness to the target is only judged within a restricted subspace greatly reduces both the synthesis resource requirements and the gate count in the resulting circuit. This can be important when time evolving Hamiltonians with such a block-diagonal structure, as is usually the case in quantum chemistry [211].

None of the numerically investigated cases showed a significant difference between *random search* and *tabu search*. This strongly suggests that the very simple variant of tabu search used here in an effort to guide the search through the circuit structure space more efficiently than random moves is not sophisticated enough to yield any practical advantage. Note that the implementation of tabu search used here only incorporates its most basic aspect of short-term memory. Its performance could potentially be improved by including more elaborate concepts such as intermediate-term and long-term memory [209].

The presented results for synthesising quantum Fourier transform circuits using various gate sets show that for small numbers of qubits and highly expressive sets of gates, the hill climbing method can consistently produce circuits with very few gates, almost always close to the minimum number found for any method in the scope of this work. However, if the method gets stuck in a local minimum or proceeds too slowly due to the increasing size of the neighbourhood, there are no provisions in the

[16]The degrees of freedom in a generic unitary grow exponentially with the number of qubits, so there is limited practical use for these cases.

presented framework to overcome these problems. Tabu and random search produced accurate results even for a very restricted gate set on a small number of qubits, but scaled unfavourably when increasing the number of qubits in these cases.

Finally, the attempts at synthesising an n-qubit Toffoli gate clearly showed the limitations of the discussed method. Due to the properties of the cost functions, as discussed in Section 3.3.4, the only successful attempts at synthesising circuits for Toffoli gates on 5 qubits was by assisting the algorithm with previously generated knowledge.

It is important to emphasise that there is significant value in the ability to synthesise even small unitaries, since such compiled functions can be used as components of larger algorithms, even if the compilation technique does not scale favourably to many qubits. For example, in grid-based chemistry [212] (see also Section 4.2), although the total number of computational qubits may be in the thousands, the QFT used to interconvert between real space and momentum space is simply the product of QFTs acting on each dimension and particle separately, often requiring the transformation to act on only around 20 qubits. A different example is the method in Ref. [71], where large circuits are first decomposed into blocks small enough for individual recompilation. The presented method may be used as a subroutine in such a scheme to synthesise efficient circuits for each of these blocks. Therefore, even the ability to compile a multi-qubit gate involving 3 or 4 qubits efficiently into a compact set of 1- and 2-qubit gates can be valuable. Consequently, the significance of the techniques described in this paper does not depend on their ability to scale directly to circuit sizes that might be considered 'post-classical' ($\gtrsim 50$ qubits). Nevertheless, it would still be desirable to scale the presented algorithms to such large scale endeavours. As the results suggest that the presented methods will require significant further development for any such task to be realistic, I now remark on a few potential directions for improvement.

Firstly, note that the algorithms used for circuit structure modifications are largely independent from the parameter optimisation routine, except for reusing information in the quantum metric tensor. Therefore, if a different method for optimising the parameters proves more suitable, it can be straightforwardly substituted for the imaginary time evolution used in the present investigation. One promising candidate is introduced in Ref. [96] named CoVaR, where an eigenstate of the system is prepared by a root-finding algorithm similar to Newton's method. By tweaking the spectrum of the synthesis Hamiltonians such that only product states are eigenstates, this method could be used to find the appropriate parameters in each iteration.

Secondly, it is not strictly necessary that the operators in the library consist only of unitaries. Instead, it would be possible to also include ancilla qubits on which intermediate measurements may be performed, and whose outcomes can become part of the cost function or even the following control flow. In this case, should imaginary time evolution remain the parameter optimisation tool of choice, it must be adapted to find the correct descent direction [205].

Thirdly, there are also possible enhancements to the Hamiltonians used to generate the cost functions. As briefly mentioned in Section 3.1.2, instead of the relatively straightforward H_{proj} and H_{sum}, other properties of the solution may be included to judge how suitable a particular outcome is, such as the desired entanglement via a witness, or the conservation of symmetries in the problem.

Fourthly and perhaps most challengingly, an enthralling direction of further research is to deviate from the random nature of adding gates to the existing structure, and explore ways for introducing circuit variants that are more guided by more of the available information. While employing more elaborate variants of the tabu search algorithm used here may enhance the performance significantly, another promising path forward might be to use information from the output state to deduce which circuit modifications are most likely to reduce the cost function. This could potentially provide much more direct and cheaper guidance than exploring the vast neighbourhood of a circuit and trying to learn from it.

I finally remark that because of the formulation as an energy minimisation problem, the presented methods to construct circuits *ab initio* can not only be used to express a desired unitary using various target gate sets, but also as a variational quantum eigensolver to prepare the ground state of some physical Hamiltonian. For this task, the cost function is straightforwardly replaced by the energy of the Hamiltonian of interest. This procedure, among further applications, is explored in Ref. [210].

As is often the case in optimisation problems, the most generic solvers pay for their generality with solving time penalties and/or suboptimal end results. Such is also the case in this chapter. If the unitary of interest has a specific form, a specialised method is likely to produce better results than the universal approach presented here. The next chapter discusses two such specialised methods, which can be used to generate circuits implementing the time evolution of a system, and perform particularly well if that system is a quantum chemical one.

Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.

— Richard Feynman

4 HAMILTONIAN TIME EVOLUTION FOR QUANTUM CHEMISTRY

This chapter describes two distinct methods for simulating chemical systems on quantum hardware. One follows Ref. [213], which was mainly my own work, with conceptual input from Earl Campbell and Simon Benjamin. The other focuses on the paper "Grid-based methods for chemistry simulations on a quantum computer" by Chan et al. [212], to which I contributed some of the simulation code, performed numerical calculations on HPC hardware yielding data, and performed parts of the analysis. For context and completeness, Section 4.2.1 gives a short summary of the framework and important aspects of the methods and findings of [212] to which I did not directly contribute. Section 4.2.2 contains my contributions to the work in detail, partly using text which I originally authored verbatim.

Contents

Chapter 3 discussed a very generic method of how to synthesise circuits for any task, as long as it can be given as a black box gadget on a (simulated) quantum computer. While the results for few qubits seem promising, scalability to larger systems appears to be a yet unsolved issue. However, in many instances, true generality is not required when creating circuits. Indeed, specialised methods which only work for particular tasks may show more favourable scaling properties.

Since a central task for which quantum computers promise to be useful is the real-time evolution of quantum states, bespoke algorithms for creating circuits that realise Hamiltonian simulation have seen significant progress over the last few years, as discussed in Section 2.2. These methods may be considered specialised quantum compilers, implementing only a particular type of unitary. In this chapter, I will discuss two instances of such approaches in the context of quantum chemistry, i.e. the system of interest is comprised of nuclei and electrons, which is one of the main research areas where it is hoped that quantum computers would be useful, as elaborated on in the Introduction.

The first is a variant of the truncated Taylor series method [132], itself derived from the linear combinations of unitaries framework [130]. As I will show, its performance can be significantly improved by taking advantage of the specific structure that Hamiltonians in quantum chemistry usually have. In Section 4.1 I describe this adaptation in detail.

The second is grid-based methods, where the wave function in first quantisation at a regular grid of points is stored in a the state of a quantum register. Again, this is an approach that seems best suited for quantum chemistry applications, and the main Reference [212] for this section examines it from this point of view.

4.1 Truncated Taylor series for quantum chemistry

This section describes, as alluded to above, a variant of the truncated Taylor series scheme [132], that strives to produce more accurate results for some quantum mechanical systems, especially those of electrons in molecules when expressed in second quantisation and mapped to qubits via the Jordan–Wigner transformation [214].[17] The method builds on the observation that the Hamiltonians of these systems often have terms that vary considerably in magnitude. Therefore, in a Taylor series, some (small) terms may be discarded at a lower expansion order, while other (large) terms should be kept to a higher order. The way this idea is applied to the truncated Taylor series method respects the efficient circuit implementation of Ref. [132] and adapts improvements to the SELECT and PREPARE subroutines introduced in Refs. [137, 216].

This section is structured as follows. Section 4.1.1 contains a detailed description of modified method adapted from [132]. Its effectiveness is tested using calculations of various molecules in their stable configurations, with the findings being reported in Section 4.1.2, while Section 4.1.3 gives further interpretations of the results and mentions potential further work.

4.1.1 Method

As mentioned at the beginning of this chapter, the method presented here is closely related to the approach introduced by Berry et al. [132]. I will give a detailed description of the modified method, which at the same time serves as a summary of [132].

Linear combination of unitaries

The protocol is based on a method of adding unitaries using ancilla qubits [130]. The starting point is a Hamiltonian of the form

$$H = \sum_{\ell=0}^{L-1} \alpha_\ell h_\ell, \tag{4.1}$$

where α_ℓ are real positive scalars[18] and h_ℓ are unitaries for which implementations on a quantum computer exist. Without loss of generality, assume the terms are sorted by magnitude, i.e. $\alpha_{\ell+1} \leq \alpha_\ell$. The approach also used in [132] is to implement an approximation to the corresponding time evolution operator

$$U(t) = e^{-iHt} \tag{4.2}$$

[17]Other mappings like Bravyi—Kitaev [215] are possible, but not explicitly addressed in this work.
[18]Complex phases can always be absorbed into the operators h_ℓ.

with a Taylor series. Taking t to be sufficiently small, the series representation of $U(t)$ can be approximated by the sum

$$U_{\underline{L}}(t) := \mathbb{1} + \sum_{k=1}^{\infty} \frac{(-it)^k}{k!} \prod_{j=1}^{k} \left(\sum_{\ell_j=0}^{L_j-1} \alpha_{\ell_j} h_{\ell_j} \right) \tag{4.3}$$

where $\underline{L}$ is a vector of L_k with $k \in \mathbb{N}^+$ and elements $0 \leq L_k \leq L$, meaning the individual sums in the product only contain the L_k largest terms of H. To illustrate this more clearly, consider the following example where $\underline{L} = (4, 2, 1, 0, \ldots)$, which leads to

$$\begin{aligned} U_{\underline{L}}(t) = \mathbb{1} &- it(\alpha_0 h_0 + \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3) \\ &- \frac{t^2}{2}(\alpha_0 h_0 + \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3) \cdot (\alpha_0 h_0 + \alpha_1 h_1) \\ &+ \frac{it^3}{6}(\alpha_0 h_0 + \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3) \cdot (\alpha_0 h_0 + \alpha_1 h_1) \cdot (\alpha_0 h_0). \end{aligned}$$

This limitation of the number of terms in each sum is the main difference to [132], where the series is truncated at some appropriate order n, which yields

$$U_n(t) := \mathbb{1} + \sum_{k=1}^{n} \frac{(-it)^k}{k!} \prod_{j=1}^{k} \left(\sum_{\ell_j=0}^{L-1} \alpha_{\ell_j} h_{\ell_j} \right). \tag{4.4}$$

Equation (4.4) is a special case of Eq. (4.3), where all orders up to n are included in full.[19] The modified version of the sum includes some orders only partially, giving greater control over the total gate count and allowing for quicker convergence of the error bounds by selectively adding mostly terms with larger weight.

The magnitude of the time step t will turn out to be a fixed value restricted by the method. Longer times $\tau = rt$, can be simulated by applying $U_{\underline{L}}^r$. However, most of the following description will focus on the implementation of a single time step.

To keep the notation simple, the products of the coefficients α_ℓ with $t^k/k!$ are gathered into new variables β_j, and all products of the unitaries h_ℓ together with $(-i)^k$ are collected into operators V_j, with a newly introduced label j numbering all terms in the sum. Note that even if different products of h_ℓ yield identical operators, they

[19]For all quantities with an $\underline{L}$ subscript I will alternatively replace it with n to mean an $\underline{L}$ where $L_k = L$ for $k \leq n$ and $L_k = 0$ for $k > n$.

are treated as separate V_j, each with a corresponding weight β_j. By construction, all β_j are also real and positive. Thus, Eq. (4.3) becomes

$$U_{\underline{L}} = \sum_{j=0}^{m-1} \beta_j V_j \tag{4.5}$$

where the time-dependence of $U_{\underline{L}}$ and β_j is not explicitly denoted, and the total number of terms m implicitly depends on $\underline{L}$.

In order to apply $U_{\underline{L}}$ to a state $|\psi\rangle$ in the system Hilbert space $\mathcal{H}$, it is convenient to define the unitary operators $\mathcal{P}(t)$ and $\mathcal{S}$ (PREPARE and SELECT) in accordance with [132]. The PREPARE operator $\mathcal{P}$, whose time dependence will be implicit from here on, maps the $|0\rangle$ state of a set of ancilla qubits (forming the Hilbert space $\mathcal{H}_\mathrm{A}$) to the weighted superposition

$$\mathcal{P}\,|0\rangle := \frac{1}{\sqrt{s_{\underline{L}}}} \sum_{j=0}^{m-1} \sqrt{\beta_j}\,|j\rangle \quad \in \mathcal{H}_\mathrm{A} \tag{4.6}$$

with the implicitly t-dependent normalisation constant

$$s_{\underline{L}} := \sum_{j=0}^{m-1} \beta_j . \tag{4.7}$$

The SELECT operator $\mathcal{S}$ acts on a state $|\psi\rangle \in \mathcal{H}$ with the unitary V_j, where j is given by the state of the ancilla introduced above. So its action on a tensor state of $|j\rangle\,|\psi\rangle \in \mathcal{H}_\mathrm{A} \otimes \mathcal{H}$ is

$$\mathcal{S}\,|j\rangle\,|\psi\rangle := |j\rangle\,V_j\,|\psi\rangle . \tag{4.8}$$

Analogously to [132], these two operators $\mathcal{P}$ and $\mathcal{S}$ can be combined to introduce a new operator

$$\mathcal{W} := (\mathcal{P}^\dagger \otimes \mathbb{1})\,\mathcal{S}\,(\mathcal{P} \otimes \mathbb{1}) \tag{4.9}$$

which has the effect

$$\mathcal{W}\,|0\rangle\,|\psi\rangle = \frac{1}{s_{\underline{L}}}\,|0\rangle\,U_{\underline{L}}\,|\psi\rangle + N\,|0^\perp, \Phi\rangle \tag{4.10}$$

where N is the appropriate constant for the state to be normalised, and $|0^\perp, \Phi\rangle$ is a garbage state whose ancilla part has no overlap with the ancillary $|0\rangle$ state.

Oblivious amplitude amplification

The naïve way to obtain $U_{\underline{L}}|\psi\rangle$ would be to measure the ancilla of $\mathcal{W}|0\rangle\,|\psi\rangle$, see Eq. (4.10), and post-select for the ancilla $|0\rangle$ state. However, since $s_{\underline{L}}$ increases with t, the success probability of this straightforward approach diminishes for large t. Additionally, t is always subject to convergence of Eq. (4.3). Due to the postselection, dividing the total t into smaller segments and repeating the process multiple times also suppresses the total success probability.

One way around this problem also used in [132] is the so-called oblivious amplitude amplification. As detailed in Lemma 1 in Appendix B, and references therein, if $U_{\underline{L}}$ were unitary and $s_{\underline{L}} = 2$, the amplification operator

$$\mathcal{Q} := -\mathcal{W}R\mathcal{W}^\dagger R, \tag{4.11}$$

with $R := 2\Pi - \mathbb{1}$ being the reflection operator about the $|0\rangle$ state of the ancilla and $\Pi := |0\rangle\langle 0| \otimes \mathbb{1}$ meaning the projector onto the ancilla $|0\rangle$, would have the effect [133]

$$\mathcal{Q}\mathcal{W}|0\rangle\,|\psi\rangle = |0\rangle\,U_{\underline{L}}|\psi\rangle\,. \tag{4.12}$$

Thus, this amplified operator warrants the definition

$$\mathcal{A} := \mathcal{Q}\mathcal{W} = -\mathcal{W}R\mathcal{W}^\dagger R\mathcal{W}. \tag{4.13}$$

The requirement of $s_{\underline{L}} = 2$ can be satisfied as follows. The form of the modified Taylor expansion leads to $s_{\underline{L}}$ being of the form

$$s_{\underline{L}}(t) := \sum_{k=1}^{\infty} \frac{t^k}{k!} \prod_{j=1}^{k} \underbrace{\left[\sum_{\ell_j=0}^{L_j-1} \alpha_{\ell_j}\right]}_{:=\Lambda_j} = \sum_{k=0}^{\infty} \frac{t^k}{k!} \prod_{j=1}^{k} \Lambda_j. \tag{4.14}$$

The restriction $s_{\underline{L}} = 2$ therefore forces the simulation time t to be the only real root of

$$\sum_{k=0}^{\infty} \frac{t^k}{k!} \prod_{j=1}^{k} \Lambda_j - 2 = 0 \tag{4.15}$$

which I will call $t_{\underline{L}}$. If all orders were included in full, i.e. $L_k = L$, $\forall\, k$, all Λ_j would be equal and the infinite sum on the left becomes the series of the exponential function. I will refer to the time step for this case as $t_\infty = \log(2)/\Lambda$, with the definition $\Lambda := \sum_{j=0}^{L} \alpha_j$.

Shorter times can be accomplished by using an extra qubit, as described in Reference [133]. Since the only requirement for oblivious amplitude amplification to work is $s_{\underline{L}} = 2$, and shorter times mean $s_{\underline{L}} < 2$ — i.e. the amplitude of the ancilla $|0\rangle$ is too large — its amplitude can be reduced by introducing an additional qubit to the ancilla and preparing its $|1\rangle$ state with enough weight such that the overall ancilla $|0\rangle$ reduces to amplitude $1/2$. These shorter times are only relevant in the last time step of a simulation and have almost the same cost as a full step, so the rest of the discussion will be limited to multiples of $t_{\underline{L}}$.

Equation (4.12) only strictly holds for unitary $U_{\underline{L}}$, but the series truncation means that $U_{\underline{L}}$ is only close to unitary. Again following [132], the action of $\mathcal{A}$ for a general $U_{\underline{L}}$ can be derived by applying $\mathcal{A}$ to a state $|0\rangle\left|\psi\right\rangle$ and projecting onto the ancilla $|0\rangle$, which yields

$$\Pi\mathcal{A}|0\rangle\,|\psi\rangle = |0\rangle\left(\frac{3}{s_{\underline{L}}}U_{\underline{L}} - \frac{4}{s_{\underline{L}}^3}U_{\underline{L}}U_{\underline{L}}^\dagger U_{\underline{L}}\right)|\psi\rangle\,, \tag{4.16}$$

(derivation in Appendix B, Lemma 2) and the operator that is actually applied in the $|\psi\rangle$ subspace is

$$\tilde{\mathcal{A}}_{\underline{L}} := \frac{3}{s_{\underline{L}}}U_{\underline{L}} - \frac{4}{s_{\underline{L}}^3}U_{\underline{L}}U_{\underline{L}}^\dagger U_{\underline{L}}. \tag{4.17}$$

Gate construction

In this subsection, I want to elaborate on the specific gate construction to implement $\mathcal{A}$ efficiently, adapted from [132]. First, the ancilla is divided into $\kappa+1$ registers, where $\kappa := \|\underline{L}\|_0$ is the number of non-zero elements in the vector $\underline{L}$. The first of these registers is named q and contains κ qubits, while the others are given labels $c_1 \dots c_\kappa$, with c_k containing $\lceil \log_2 L_k \rceil$ qubits.

The q register's purpose is to represent different orders of the Taylor expansion, while the c_k registers are needed for the terms in each order. This makes it convenient to use a multi-index $j := (k, \ell_1, \dots, \ell_k)$. The corresponding state of the ancilla is

$$|j\rangle := |k\rangle_q\,|\ell_1\rangle_{c_1}\dots|\ell_k\rangle_{c_k}\cdots \tag{4.18}$$

where the state of the registers $c_{k'}$ with $k' > k$ is left unspecified. The coefficient associated with this index is

$$\beta_j = \beta_{(k,\ell_1,\ldots,\ell_k)} = \frac{t^k}{k!}\alpha_{\ell_1}\ldots\alpha_{\ell_k}. \tag{4.19}$$

PREPARE For this operator, it is convenient to slightly deviate from [132]. Exact implementation of $\mathcal{P}$ as defined in Eq. (4.6) would necessitate the preparation of the c_k registers to be conditioned on qubits in the q register. We can, however, implement an operator $\mathcal{P}^\star$, which acts equivalently to $\mathcal{P}$ when used in $\mathcal{W}$, i.e. $\mathcal{W} = (\mathcal{P}^{\star\dagger}\otimes\mathbb{1})\mathcal{S}(\mathcal{P}^\star\otimes\mathbb{1})$, but is performed independently on each of the c_k registers without controls on the qubits in q, making it cheaper to implement in practice.

The q register will contain the prefactor for each order k and uses unary coding, i.e. $|k\rangle_q := |1^k 0^{k-\kappa}\rangle_q$. Thus, the PREPARE operator $\mathcal{P}^\star(t)$ acts on this register proportional to

$$|0^\kappa\rangle_q \mapsto \sum_{k=0}^{\kappa}\sqrt{\frac{t^k}{k!}\prod_{j=1}^{k}\Lambda_j}\,|k\rangle_q. \tag{4.20}$$

This can be implemented by a rotation on the first qubit, and rotations controlled by the previous one on each subsequent qubit.

The c_k registers can now all be almost identically prepared to contain the coefficients of the Hamiltonian, where each index ℓ is mapped to the qubits of c_k in regular binary coding. So the action of $\mathcal{P}^\star$ on a single register c_k is proportional to

$$|0\rangle_{c_k} \mapsto \sum_{\ell=0}^{L_k-1}\sqrt{\alpha_\ell}\,|\ell\rangle_{c_k}. \tag{4.21}$$

For this, any efficient method for arbitrary state preparation can be used, whose cost is discussed shortly.

Combining these constituents into a single unitary $\mathcal{P}^\star$ and applying it to the whole ancilla yields the desired operator equivalent to Eq. (4.6) if used in $\mathcal{W}$, which is shown in more detail in Lemma 3 in Appendix B.

SELECT Using the established structure of the ancilla, the $\mathcal{S}$ operator must have the action

$$\mathcal{S}\,|k\rangle_q\,|\ell_1\rangle_{c_1}\dots|\ell_k\rangle_{c_k}\dots|\ell_\kappa\rangle_{c_\kappa}\,|\psi\rangle = |k\rangle_q\,|\ell_1\rangle_{c_1}\dots|\ell_k\rangle_{c_k}\dots|\ell_\kappa\rangle_{c_\kappa}\,\tilde{h}_{\ell_1}\dots\tilde{h}_{\ell_k}\,|\psi\rangle \tag{4.22}$$

with $\tilde{h}_\ell := -i h_\ell$. This can be accomplished by having a sequence of groups of unitaries in the circuit.[20] Each of the groups indexed by $m = 1\dots\kappa$ contains the unitaries $\tilde{h}_{\ell_m}$, with $\ell_m = 0\dots L_m - 1$, acting on the target state $|\psi\rangle$.

The register c_m is used as the addressing register for group m, i.e. the state $|\ell_m\rangle_{c_m}$ determines which unitary in group m is applied. To achieve this, the fact that the c registers are in binary coding can be used, so ℓ_m is represented as a binary number with the $\lceil\log_2 L_m\rceil$ qubits in c_m as digits. By controlling $\tilde{h}_{\ell_m}$ on the c_m register in a way that matches the binary representation of ℓ_m, only the unitary with the correct index is applied. For example, $\tilde{h}_5$ would be controlled by the last and antepenultimate qubit in c_m and anti-controlled by all other qubits in c_m (since 5 corresponds to the state $|0\dots0101\rangle$ in binary coding).

Additionally, the q register specifies how many of the groups are applied. If q is in the state $|k\rangle_q$, only the first k groups should be active. The unary coding in q makes this straightforward to implement by additionally controlling every unitary in group m with the m^{th} qubit in q. Figure 4.1 shows a sketch of the full construction.

The implementation method for $\mathcal{S}$ described here is well-suited to easily visualise the working principle of the operator. However, because the structure of the controls of $\tilde{h}_\ell$ forms a so-called unary iteration (iterating over all integers in sequence), the structure of the controls can be slightly modified to lower the number of required T-gates [137]. The consequences of this modified construction on the implementation cost is discussed in the next subsection.

Gate cost Finally, the implementation cost of the discussed procedure is determined by the gate complexity of the operator $\mathcal{A}$, which can be estimated as follows. Its constituents are two reflections R, and three instances of $\mathcal{W}$, each of which contains

[20]The groups in the circuit are numbered right-to-left to match the established numbering convention of the operators.

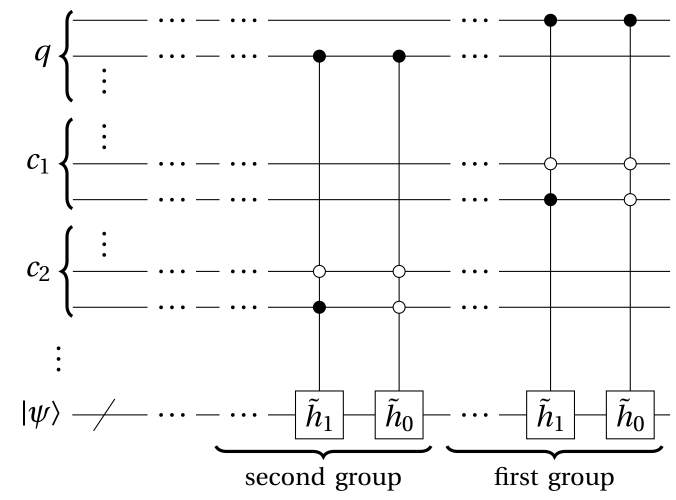


Figure 4.1: Sketch of the gate construction for $\mathcal{S}$. By taking advantage of the unary iteration structure, the T-count of the multi-controls can be significantly reduced [137]. However, this non-optimised diagram is included to visualise the working principle of the $\mathcal{S}$ operator.

one $\mathcal{S}$ and two $\mathcal{P}^\star$. For calculations using full orders as in [132], this analysis translates exactly to the gate construction given there. Considering the context and potential applicability of the procedure, the most sensible cost to consider is the the number of expensive T-gates [40, 57, 58] when using the universal gate set of Clifford + T.

Each reflection R is a single Pauli-z operator on one of the ancilla qubits (padded between two Pauli-x gates), anti-controlled on all others. This can be done with $\mathcal{O}(\sum_k \log_2 L_k)$ T-gates and a second ancilla register of size $(\kappa + \sum_k \lceil \log_2 L_k \rceil - 2)$ [26].

The PREPARE stage for the q register consists of $\kappa - 1$ controlled rotations with a total T-complexity of $\mathcal{O}(\kappa)$. Each of the κ registers c_k needs to be initialised to a specific state with $2^{\lceil \log_2 L_k \rceil} \sim L_k$ coefficients, requiring between $\mathcal{O}(\sum_k L_k)$ and $\mathcal{O}(\sum_k \sqrt{L_k} \log^2(L_k/\epsilon))$ T-gates per register, depending on the number of additionally available ancillas, where ϵ is the accuracy of the preparation [216]. In total, this yields a T-count between $\mathcal{O}(\sum_k L_k)$ and $\mathcal{O}(\sum_k \sqrt{L_k} \log^2(L_k/\epsilon))$.

As mentioned above, the fact that the controls of each h_ℓ in $\mathcal{S}$ form a so-called unary iteration can be exploited to lower the T-gate count. Each sequence of L_k operators can be implemented using $\mathcal{O}(L_k)$ T-gates [137], plus L_k times the cost of performing a single $-ih_\ell$ operator, totalling to $\sum_k L_k$ such operators. Thus, the T-complexity of $\mathcal{S}$ for generic Hamiltonians in the form of Eq. (4.1) is of $\mathcal{O}(\sum_k L_k)$, which will be used will use in this analysis. Moreover, recent work [141] has shown that a SELECT process can be yet more efficient for the special case of Jordan–Wigner-transformed N-orbital electronic structure problems, where the T-complexity is as low as $\mathcal{O}(nN)$, with $n = \max\{k : L_k \neq 0\}$.

Combining all these counts results in a total complexity of $\mathcal{O}\big(\sum_k L_k\big)$ for $\mathcal{A}$. As a proxy to use for the total gate cost in our results it is therefore reasonable to define

$$C_{\underline{L}} := \sum_{k=1}^{\infty} L_k = \|\underline{L}\|_1. \tag{4.23}$$

This definition includes the cost of a full expansion up to order n as the special case $C_n = nL$, consistent with previous notation. From this cost of a single time step, I discuss the complexity C_ϵ to reach some desired total simulation error ϵ in the next subsection.

Error bounds

The error of the method per time step can be quantified as the norm of an operator $\Delta_{\underline{L}}$ which satisfies

$$U(t_\infty) = \tilde{\mathcal{A}}_{\underline{L}}(t_\infty) + \Delta_{\underline{L}}(t_\infty) \tag{4.24}$$

where the considered time step size is now $t_\infty = \log(2)/\Lambda$. The error made after one step can be found by applying $\Pi\mathcal{A}$ once and tracing out the ancilla, which as the bound[21]

$$\delta_{\underline{L}} := \|\Delta_{\underline{L}}(t_\infty)\| \leq 2 - s_{\underline{L}}(t_\infty) =: \varepsilon_{\underline{L}} \tag{4.25}$$

up to order $\varepsilon_{\underline{L}}$ (details in Appendix B, Lemma 4). Because using $t_{\underline{L}}$ or t_∞ makes no difference in the error up to order $\varepsilon_{\underline{L}}$, it is sufficient to exclusively use t_∞ in these calculations. The error for a total simulation time $\tau = r\,t_\infty = r\log(2)/\Lambda$, $r \in \mathbb{N}$, is then

$$\|\tilde{\mathcal{A}}_{\underline{L}}(t_\infty)^r - U(t_\infty)^r\| \leq r\delta_{\underline{L}} = \frac{\Lambda\delta_{\underline{L}}}{\log 2}\tau \leq r\varepsilon_{\underline{L}}, \tag{4.26}$$

also up to order $\varepsilon_{\underline{L}}$ (see Appendix B, Lemma 5).

[21] $\|\cdot\|$ with an operator as its argument always means the operator norm in this thesis.

The bound on the total simulation error of r steps gets the new name $\epsilon := r\varepsilon$. The T-gate complexity C_ϵ of a simulation for time τ in terms of the total error bound ϵ is then in the range

$$\mathcal{O}\left(\frac{\Lambda\tau\log\frac{\Lambda\tau}{\epsilon}}{\log\log\frac{\Lambda\tau}{\epsilon}}\right) < C_\epsilon \leq \mathcal{O}\left(\frac{L\Lambda\tau\log\frac{\Lambda\tau}{\epsilon}}{\log\log\frac{\Lambda\tau}{\epsilon}}\right), \tag{4.27}$$

depending on the Hamiltonian. This is shown in detail in Appendix B, Lemma 10, which makes use of Lemmas 6 to 9.

Insertion strategy

The notion of partially included orders together with an expression for the error bound allows the implementation of a simple heuristic. Starting from any given (partial) expansion $\underline{L}$ it is straightforward to determine which of the vector elements L_k should be increased by 1 — i.e. which additional gate should be included — to give the largest decrease of the error bound, and thus the quickest convergence. Specifically, it is the k which maximises the expression

$$\sum_{\nu\geq k}\frac{t^\nu}{\nu!}\alpha_{1+L_k}\prod_{\substack{j\neq k\\1\leq j\leq\nu}}\left(\sum_{i=1}^{L_j}\alpha_i\right). \tag{4.28}$$

Starting from $\underline{L}=\underline{0}$, repeatedly adding terms that maximise (4.28) results in a greedy algorithm for decreasing the error bound, which is used to iteratively construct circuits whose accuracy is examined in the next Section.

4.1.2 Results

Molecule survey

The first observation to make is that for Hamiltonians with evenly distributed magnitudes α_ℓ, the only benefit of using the modification presented here is the finer control over the total gate count. By construction, whenever the cost introduced in Eq. (4.23) $C_{\underline{L}} = \nu L$, $\nu \in \mathbb{N}$, the revised protocol and the method used in [132] yield identical results.

It seems reasonable that the modification may be advantageous whenever the magnitudes of α_ℓ vary over several orders of magnitude, because this allows terms

Table 4.1: Molecules used for the demonstration of algorithmic performance with their molecular formula, PubChem Compound ID (CID), number of qubits (excluding ancillas), and number of terms L.

Formula	CID	Qubits	L
HO	157350	12	631
HF	16211014	12	631
HN	5460607	12	631
LiH	62714	12	631
BH	6397184	12	631
BeH_2	139073	14	666
CH_2	123164	14	1086
NH_2	123329	14	1086
BH_2	139760	14	1086
H_2O	962	14	1086
BH_3	6331	16	1953
CH_3	3034819	16	1969
NH_3	222	16	2929
CH_4	297	18	6892
O_2	977	20	2239
N_2	947	20	2951
NO	145068	20	4427
CN	5359238	20	5835
BeO	14775	20	5851
LiF	224478	20	5851
CO	281	20	5851
BN	66227	20	5851
LiOH	3939	22	8734
HBO	518615	22	8758
HCN	768	22	8758
HOF	123334	22	12070
CHO	123370	22	12070
CHF	186213	22	12074
HNO	945	22	12078
H_2NO	5460582	24	9257
CH_2O	712	24	9257
NH_2F	139987	24	15673
CH_2F	138041	24	15681
CH_3F	11638	26	18600
CH_3Li	2724049	26	19548
H_3NO	787	26	22080
OCH_3	123146	26	39392
$LiBH_4$	4148881	28	27473
CH_3OH	887	28	30419
$C_4H_8O_2$	8857	76	1614647
C_8H_6	12302244	92	1897809

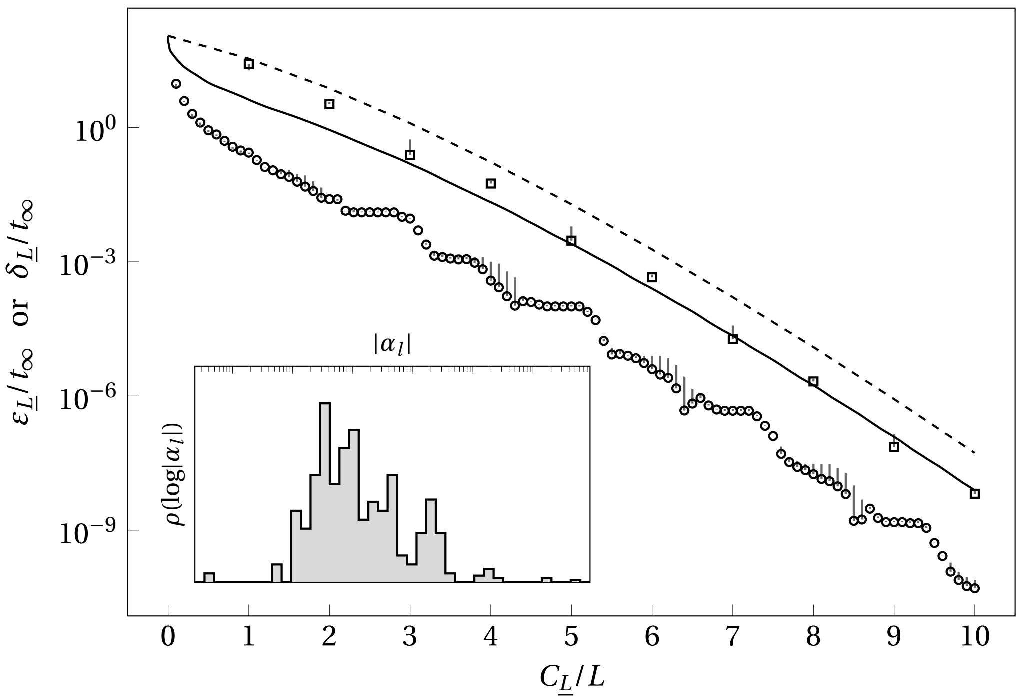


Figure 4.2: Accuracy of the Taylor expansion for the electronic Hamiltonian of hydrogen fluoride (HF), at time step size t_∞, in terms of the error per unit time vs the circuit cost $C_{\underline{L}}$ as defined in Eq. (4.23) in multiples of the cost of a full order. Lines are the error bounds $\varepsilon_{\underline{L}}$ for the unmodified - - - and modified —— circuit. Squares ▫ are the numerically obtained errors $\delta_{\underline{L}}$ for fully expanded orders, circles ∘ analogous for partial orders. The vertical grey bars point to where the error would be if we could implement $U_{\underline{L}}$ without the amplification step. The inset shows the distribution ρ of the logarithms of weights in the Hamiltonian $\log|\alpha_\ell|$.

in low orders containing small α_ℓ to be smaller than terms in higher orders containing large α_ℓ. Such magnitude distributions are often found in electronic structure Hamiltonians for molecules [217]. Because the efficiency of the new method depends critically on the specific amplitudes in the Hamiltonian, analytical results are hard to obtain. An alternative way to investigate the performance is to resort to a numerical study comparing the accuracy of the modification to the method in [132] for a group of molecules. The compounds used here are listed in Table 4.1.

The Hamiltonians for these molecules were obtained using OpenFermion [218] and PySCF [219, 220], with the basis set STO-3G [221], and geometry data retrieved from PubChem [222, 223] and the NIST Computational Chemistry Comparison and Benchmark Database [224]. Mapping from second quantisation to spin operators was done using the Jordan–Wigner transformation [214].

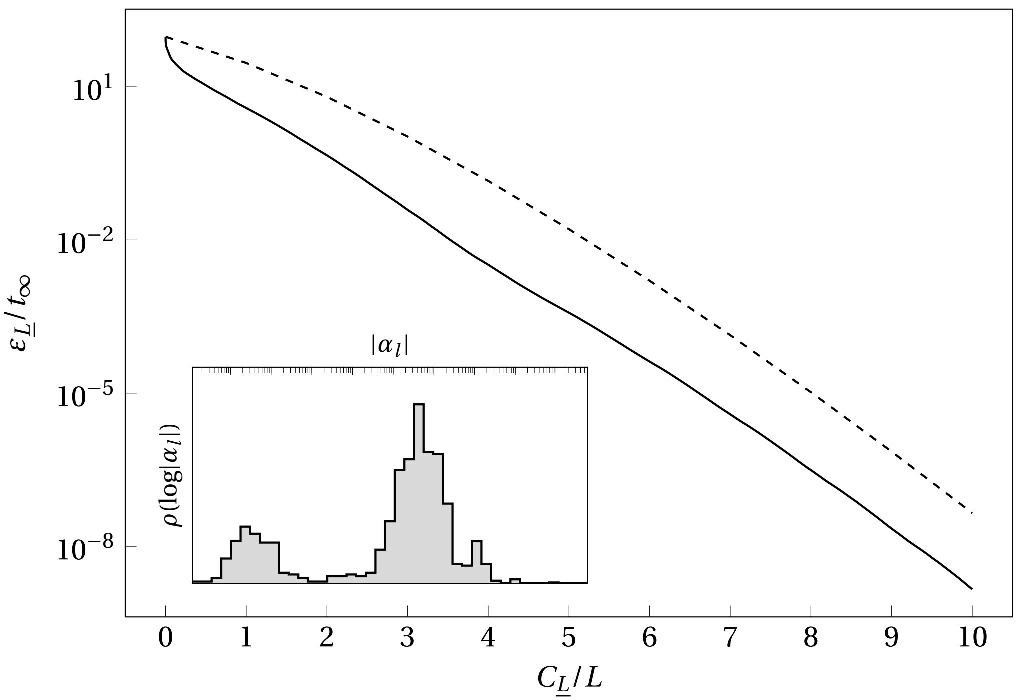


Figure 4.3: Identical plot to Fig. 4.2 showing only the convergence of error bounds for ammonia (NH_3). Notice the increasing distance between the unmodified - - - and modified —— variants of the algorithm between the orders 2 and 4, which is not present in Fig. 4.2. It is caused by the two distinct clusters in the distribution of the logarithmic weights in the Hamiltonian visible in the inset.

To showcase that the modification yields improvements regardless of the basis set used, the calculations for H_2 and LiH were repeated using cc-pVDZ, cc-pVTZ, and cc-pVQZ basis sets [225, 226]. Section 4.1.2 discusses this in more detail and shows the results.

In addition to the listed molecules, another calculation was performed, where the coefficients of the Hamiltonian for LiH were replaced with random numbers from a normal distribution with $\mu = 1$ and $\sigma = 0.1$, to show the vanishing effect of the modification whenever all weights are similar. These results are plotted alongside those for real chemical Hamiltonians and are labelled *Random*.

Figure 4.2 shows the error bounds as well as the numerically evaluated exact errors per unit time for hydrogen fluoride. Compared to the expansion to full orders, the plot shows that the modification leads to a much quicker decrease of the error bound as well as the exact error in the range $0 < C_{\underline{L}} < L$, followed by very similar convergence for

$C_{\underline{L}} > L$. This pattern is consistent with the convergence observed for most other molecules in the set.

Some compounds considered for this study — namely BH_3, CH_3, NH_3, CH_3F, CH_3Li, OCH_3, and $LiBH_4$ — show a slightly different behaviour, where the ratio of error bounds using the modified and unmodified versions increases once more later in the iteration. Figure 4.3 illustrates this using NH_3 as an example. The delayed convergence is caused by a distinct second peak in the distribution of the logarithms of weights in the Hamiltonian $\log|\alpha_\ell|$, present in the mentioned molecules. The rest of the set shows distributions rather similar to that of HF in the inset in Fig. 4.2. This disparity is also visible in the summarised results in Figs. 4.4 and 4.5, where the spread for the mentioned molecules (especially for the error in Fig. 4.4) is much greater than for the rest.

To summarise the results for all molecules, the ratio of the errors of the original and the modified version at cost values $C_{\underline{L}} = nL$ was obtained, with $n = 1\dots10$, with a time step of t_∞ for each respective molecule. The results are depicted in Fig. 4.4. Across the listed molecules, the modification consistently yields errors roughly one order of magnitude lower than the unmodified method at equivalent costs, with some ratios as low as 3 and some as high as 100.

Instead of asking how much the modified version can improve the accuracy of the simulation for a given budget, it is also interesting how much implementation cost can be saved for a desired simulation accuracy threshold. To this end, the errors δ_n of the expansions to full orders n were calculated, and the cost $C_{\underline{L}}$ of the modified version to yield the same error was recorded. The results are depicted in Fig. 4.5. Using the modified method leads to saving approximately one order in most cases, i.e. the accuracy obtained by expanding n full orders can be produced with a cost of roughly $C_{\underline{L}} \approx (n-1)L$.

The presented results show no strong correlation with neither the number of qubits in the Hamiltonian nor the number of terms L. Therefore it is reasonable to presume that these properties will also hold for other chemical Hamiltonians obtained in the same way.

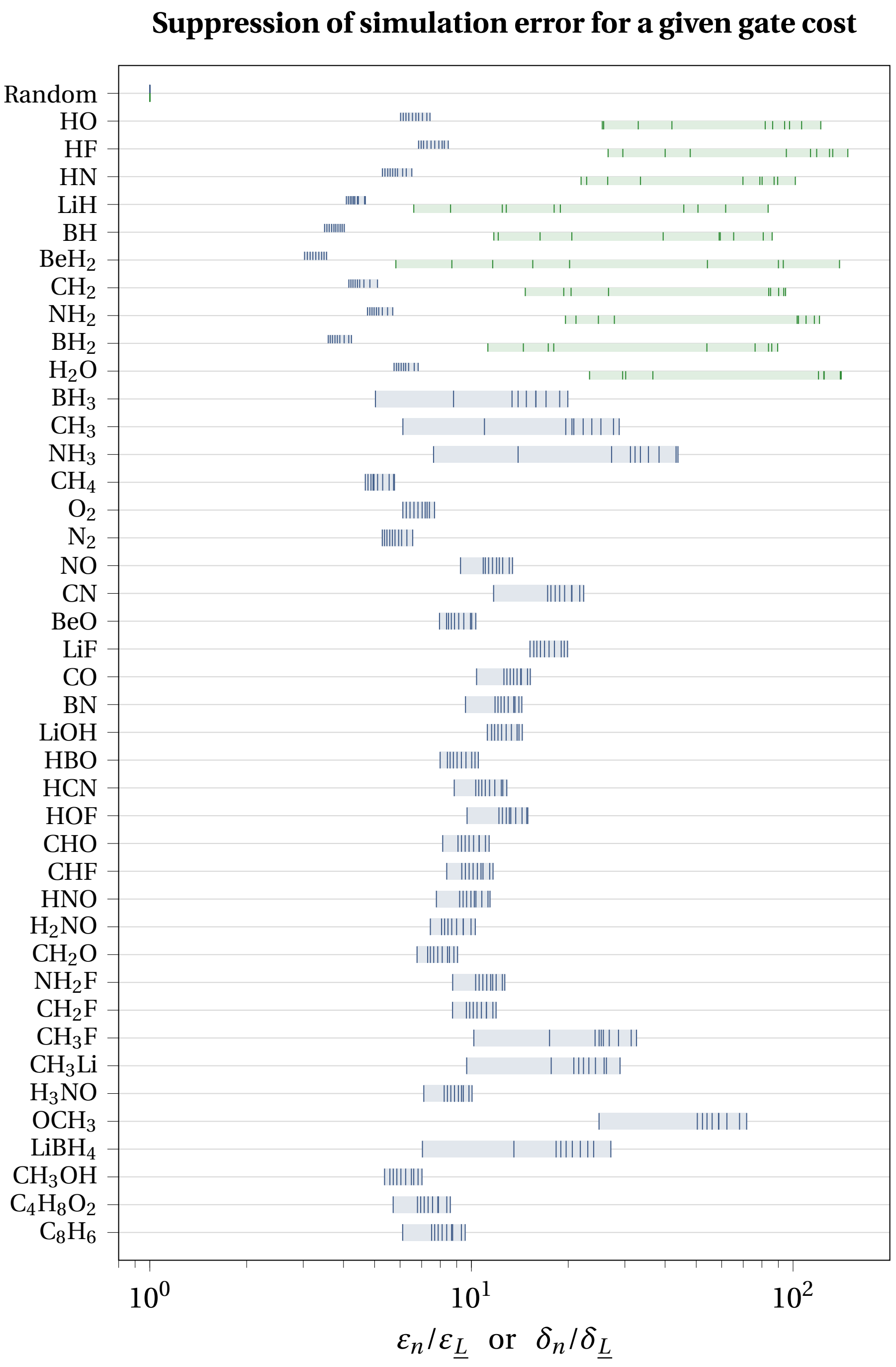


Figure 4.4: Magnitude of the errors obtained without, divided by the magnitude of the errors with the modification, for different molecules at identical implementation cost, using a time step of t_∞. Errors were evaluated at cost values $C_{\underline{L}} = (1\ldots10)L$. Each vertical line represents the ratio of errors at some cost $C_{\underline{L}}$, the shaded areas indicate the range from the smallest to the largest data point. The advantage of the modified algorithm therefore increases left-to-right. Top-bottom split data indicates ratios of error bounds $\varepsilon_n/\varepsilon_{\underline{L}}$ at the top (marked in blue) and ratios of numerically obtained errors $\delta_n/\delta_{\underline{L}}$ at the bottom (marked in green). If there is no split, the blue lines represent error bounds only.

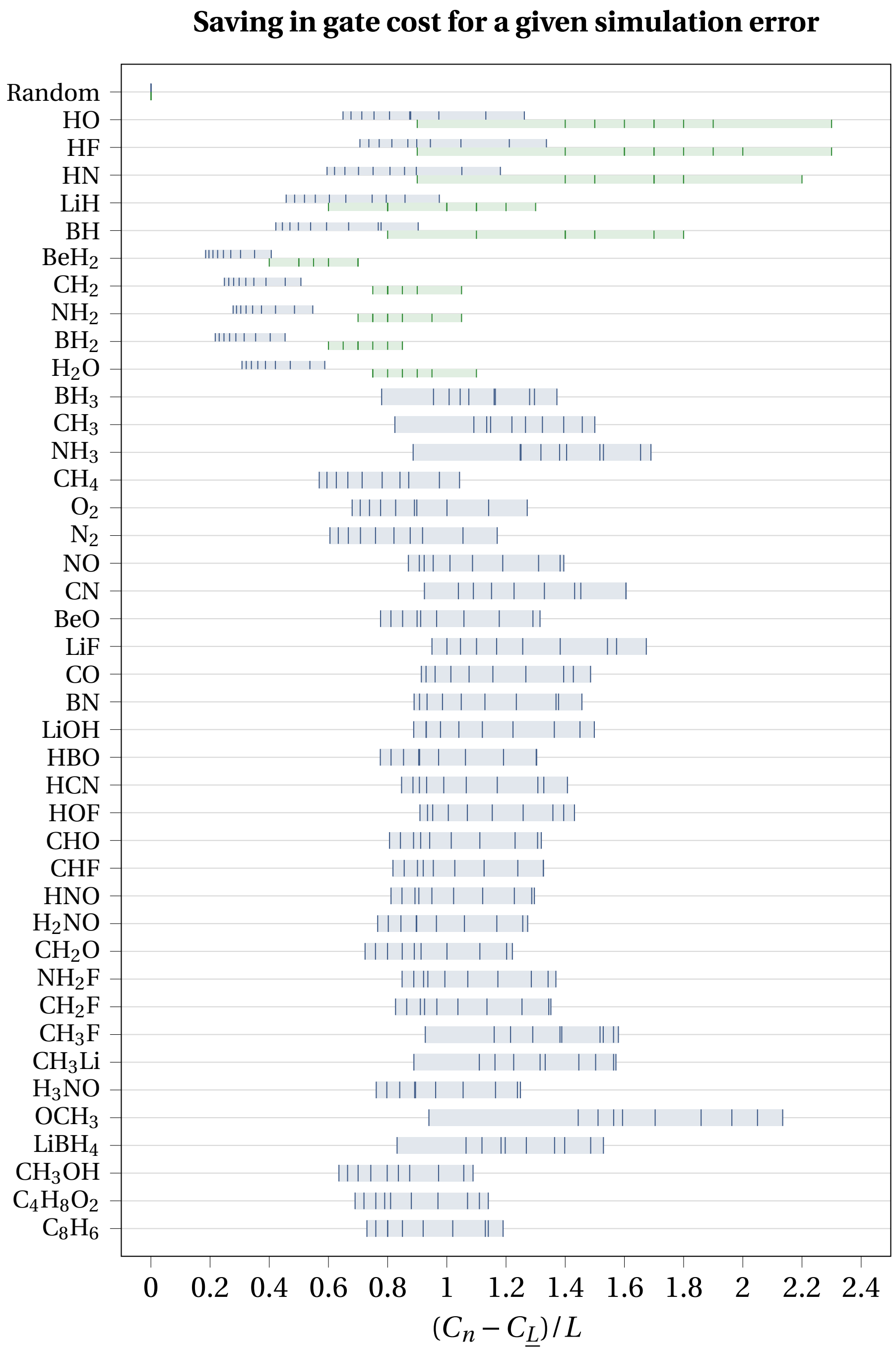
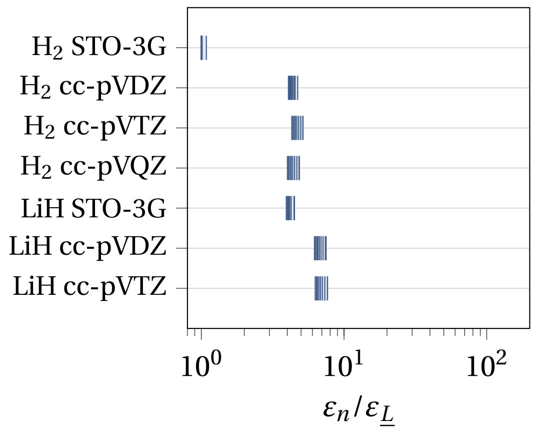


Figure 4.5: Difference between the cost of full expansions $C_n = nL$ to order $n = 1\ldots10$ and the cost of an iteratively constructed circuit $C_{\underline{L}}$ to arrive at the same error, normalised to the cost of one full order L, for each molecule. Each vertical line represents the difference at some value of n, the shaded areas indicate the range from the smallest to the largest data point. The advantage of the modified algorithm therefore increases left-to-right. The time step size is t_∞. Top-bottom split data indicates differences for error bounds at the top (marked in blue) and for numerically obtained errors on the bottom (marked in green). If there is no split, the blue lines represent error bounds only.

Effect of basis set choice

As mentioned previously, to ensure the observed effect also holds for larger basis sets, additional calculations were performed for H_2 and LiH using cc-pVDZ, cc-pVTZ, and cc-pVQZ bases.



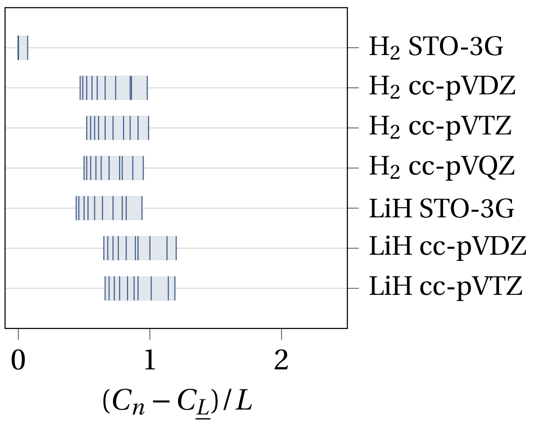


Figure 4.6: Magnitude of the error bounds obtained without, divided by the magnitude of the error with the modification, for different molecules and basis sets, using a time step of t_∞. Errors were evaluated at each cost value $C_{\underline{L}} = (1\dots10)L$. Each line represents the ratio of error bounds $\varepsilon_n/\varepsilon_{\underline{L}}$ at some cost $C_{\underline{L}}$. The advantage of the modified algorithm therefore increases left-to-right.

Figure 4.7: Difference of the cost of full expansions $C_n = nL$ to order $n = 1\dots10$ and the cost of an iteratively constructed circuit $C_{\underline{L}}$ to arrive at the same error bound, normalised to the cost of one full order L, for each molecule and basis set. The advantage of the modified algorithm increases left-to-right. The time step size is t_∞.

Figures 4.6 and 4.7 show that for the considered cases, larger basis sets seem to slightly enhance the advantage of our modification. As a special case, for H_2 our proposed method yields almost no improvement when using STO-3G, due to the very low number of only 15 terms. Apart from this outlier, the influence of the choice of basis set on the modified algorithm's performance seems small.

4.1.3 Discussion

This section illustrated that a natural extension of the method proposed in Ref. [132] can lead to noticeable improvements in the convergence of the approximation when used for electronic structure Hamiltonians of molecules. The straightforward greedy algorithm — starting from an empty expansion and iteratively inserting terms that lead to the largest decline of the error bound — yields a rapidly shrinking simulation error when starting to build up the circuit. At later stages of the circuit construction, the rate of convergence shows greater resemblance between the original and modified methods. Yet, a roughly constant factor of about an order of magnitude is maintained when measuring the simulation error vs the gate cost for the molecules tested in this section. Hence, even though the asymptotic scaling for both methods is equivalent, the modification scales down the constant prefactor, which will be beneficial for implementations on quantum hardware. When a fixed error threshold is to be achieved, this advantage translates to saving about one full order of the expansion.

The presented modification is conceptually simple: the circuit structure of the canonical truncated Taylor series method remains unchanged. A simple rearrangement of the order in which gates are added facilitates a more rapid convergence of the simulation error with the implementation cost.

Due to the lack of analytic relations between the amplitudes in the investigated Hamiltonians, only numeric results are available. However, because of the relatively large sample size of molecules considered here, it stands to reason that this behaviour will generalise to a large portion of electronic structure Hamiltonians.

As mentioned in the literature review (Section 2.2), in light of other methods like qubitization and quantum signal processing, the truncated Taylor series may be of particular relevance for diagonally dominant and time-dependent Hamiltonians. Investigating the suitability of the discussed modification to such problems would therefore be an interesting question for future work. Moreover, combining the proposed adaptations with improvements by Novo and Berry [134], who add an additional correction step to the method, could also be worth exploring.

4.2 Grid-based chemistry

The truncated Taylor series method discussed in the previous section starts from a quantum chemistry Hamiltonian in second quantisation, mapped to a sum of Pauli operators, and finds an efficient method of simulating the dynamics of that system. In essence, it expands the electronic particle density into some basis of Fock states (often comprised of hydrogen wave functions), and drives the dynamics of the coefficients of that expansion forwards in time by means of solving the Schrödinger equation for the Hamiltonian, which in this form only contains creation and annihilation operators.

However, in some cases, it can be advantageous to not use second quantisation, but instead describe and store the wave function explicitly in first quantisation, with its amplitudes discretised on a regular grid in real space.

4.2.1 Method summary

The idea of using first quantisation and a real space grid to simulate quantum dynamics has been extensively investigated before [144, 227–234]. While previous studies mostly provide theoretical resource analysis, Ref. [212] deploys significant classical computational power to directly investigate the required resources and pitfalls such a grid-based simulation might face.

Preliminaries

For the purpose of the method explored in Ref. [212], the starting point is the non-relativistic time-dependent Schrödinger equation

$$\frac{\partial}{\partial t}\Psi(\underline{r},\underline{s},t) = -iH\Psi(\underline{r},\underline{s},t),$$

which governs the evolution of the wave function $\Psi(\underline{r},\underline{s},t)$, with space ($\underline{r}$) and spin ($\underline{s}$) coordinates across time t. In systems relevant to quantum chemistry, the Hamiltonian H is usually given by[22]

$$H = H_{\text{kin}} + H_{\text{V}} + H_{\text{U}},$$

where

$$H_{\text{kin}} = -\sum_{p=1}^{P} \frac{1}{2m_p}\nabla_p^2$$

is the kinetic energy operator, which simply sums the kinetic energy of the P individual particles numbered by p. For interacting charged particles

$$H_{\text{V}} = \sum_{p\neq q} \frac{Z_p Z_q}{2|\underline{r}_p - \underline{r}_q|}$$

[22]Atomic units are used throughout this section.

is the total Coulomb interaction energy, given by the sum of all pairwise interactions. The last term H_U models interactions of the particles with an external field. Such a field can be an applied electrical potential to which a molecule is subjected, or it could represent a classical charge with a fixed position in space, which is not explicitly part of the simulation as a separate particle. Indeed, although this formalism allows for the nuclei to be included as quantum objects in the simulation, it is sometimes sensible to fix them in space and treat their fields classically, which will be the case for the rest of this discussion. For a single nucleus at the origin with charge Z, it then takes the form

$$H_\mathrm{U} = \sum_p \frac{Z Z_p}{|\underline{r}_p|},$$

where p again numbers the particles. This can be straightforwardly expanded to multiple nuclei by summing over them and inserting the appropriate coordinates in the denominator.

Reference [212] then continues to discuss appropriate choices for the basis functions that should be used to represent the wave function, arguing that when using a "simulation box" of side length L, which contains (almost all of) the wave function, an appropriate choice for the one-dimensional case in momentum space is the plane wave basis $|k\rangle$, which has the real space representation

$$\langle x|k\rangle = \frac{1}{\sqrt{L}} \exp\left(\frac{2\pi i k x}{L}\right),$$

with appropriate values for k. When representing the wave function with a quantum register containing n_r qubits, a natural choice for k is to let it take the integer values $k = -\rho, -\rho+1, \ldots, 0, \ldots, \rho-1$ with $\rho := 2^{n_r-1}$. A spatial wave function $\Psi(x)$ with the expansion

$$\Psi(x) = \langle x|\Psi\rangle = \frac{1}{\sqrt{L}} \sum_{k=-\rho}^{\rho-1} a_k \exp\left(\frac{2\pi i k x}{L}\right)$$

is then represented in the quantum register as the k-space (KS) state

$$|\Psi\rangle \leftrightarrow |\psi\rangle^\mathrm{KS} = \sum_{k=-\rho}^{\rho-1} a_k |k\rangle,$$

where $|k\rangle$ is a computational basis state whose binary representation is interpreted as an *offset binary*[23] or using *two's complement*.[24]

From this momentum space representation, the real space expression follows naturally via the discrete Fourier transform of its coefficient vector.

$$b_n = \mathcal{F}[\underline{a}] = \frac{1}{\sqrt{2\rho}} \sum_{k=-\rho}^{\rho-1} e^{\frac{\pi i n k}{\rho}} a_k$$

Because the amplitudes a_k are stored in a quantum state, this operation can be affected using an inverse[25] quantum Fourier transform.

$$|\psi\rangle^{\mathrm{RS}} = \mathrm{QFT}^{\dagger}\, |\psi\rangle^{\mathrm{KS}}$$

Note that this state is still stored in the same physical quantum register, but after the quantum Fourier transform represents the information about the simulated state using a different set of basis functions, i.e.

$$\langle x|\Psi\rangle = \sum_n b_n P(x - x_n) \tag{4.29}$$

with $x_n = \frac{nL}{2\rho}$ and

$$P(x) = e^{-\frac{i\pi x}{L}} \sqrt{\frac{2}{\rho L}} \left[\sum_{j=1}^{\rho} \cos \frac{\pi(2j-1)x}{L} \right].$$

In real space, the basis function $P(x)$ would ideally be a Dirac delta, such that the coefficients in Eq. (4.29) encode the amplitude of the wave function at a specific point in space. However, while $P(x)$ is strongly peaked around the origin, due to the finite number of plane waves in momentum space, it is not quite infinitely concentrated. Nevertheless, Ref. [212] makes the case that in the context of its exploration, P can often be treated as such. Indeed, it is argued that a suitable way to discretise a given (simulation) wave function $\Psi(x)$ into the real space basis given above is

$$\Psi(x) \approx \frac{1}{\sqrt{C}} \sum_{n=-\rho}^{\rho-1} \Psi(x_n) P_n(x), \tag{4.30}$$

23 Offset binary is a representation for signed binary numbers, where a constant offset is subtracted from the usual integer representation of every number.

24 Two's complement is a slightly more complex way to represent signed integers using binary numbers.

25 In many popular pieces of literature, the exponent signs of the classical Fourier transform and the QFT are exactly opposite.

with $P_n(x) := P(x - x_n)$ and an appropriate normalisation constant C (taking discretisation errors into account). This means the amplitudes of the real space representation in the quantum register are simply the amplitudes of the simulation wave function sampled at the peak positions x_n of the basis set, so $b_n = \Psi(x_n)$.

The discussion so far has been focused on the one-dimensional case, but generalisation to higher dimensions is straightforward. The momentum space basis functions are then products of plane waves travelling along the cartesian coordinate axes, which, after Fourier transforming each coordinate, translates to the real space basis functions also being products of their one-dimensional variants. A wave function in three dimensions is then discretised like

$$\Psi(x,y,z) \approx \frac{1}{\sqrt{C}} \sum_{n,m,l=-\rho}^{\rho-1} \Psi(x_n, y_m, z_l) P_n(x) P_m(y) P_l(z).$$

To represent this state on a quantum computer, the register is segmented into a tensor product of three individual states, each corresponding to one dimension.

$$|\psi\rangle^{\text{RS}} \propto \sum_{n,m,l} \Psi(x_n, y_m, z_l) |n\rangle |m\rangle |l\rangle$$

Analogously, representing multiple particles can be achieved by adding further such registers, where for P particles in d dimensions, a total of Pd registers are required, each containing n_r qubits.

$$|\psi\rangle^{\text{RS}} \propto \sum_{n_1, m_1, l_1, \ldots, l_P} \Psi(x_{n_1}, y_{m_1}, z_{l_1}, \ldots z_{l_P}) |n_1\rangle |m_1\rangle |l_1\rangle \ldots |l_P\rangle$$

The corresponding state in momentum space is denoted appropriately by

$$|\psi\rangle^{\text{KS}} \propto \sum_{k_1, j_1, h_1, \ldots, h_P} b_{k_1, j_1, h_1, \ldots, h_P} |k_1\rangle \left| j_1 \right\rangle |h_1\rangle \ldots |h_P\rangle .$$

For practical purposes, these individual registers are combined into a single large one, where each string of n_r neighbouring qubits represents one coordinate of one particle. For example, for two particles in two dimensions using $n_r = 4$, one of the computational basis states would be

$$|\underbrace{0100}_{|n_1\rangle}\overbrace{1110}^{|m_1\rangle}\underbrace{1101}_{|n_2\rangle}\overbrace{0010}^{|m_2\rangle}\rangle^{\text{RS}} .$$

This setup now allows for straightforward switching between momentum- and real-space representation in the quantum register by applying an (inverse) quantum Fourier transform to each of the sub-registers.

$$|\psi\rangle^{\mathrm{KS}} \quad \underset{\mathrm{QFT}}{\overset{\mathrm{QFT}^\dagger}{\rightleftarrows}} \quad |\psi\rangle^{\mathrm{RS}}$$

Split-operator time propagation

The formalism established in the previous section now allows for an efficient way of implementing the time evolution of some initial state, which is governed by the Hamiltonian H as introduced above. Using a first-order Trotter-Suzuki formula, this time evolution for small Δt can be approximated by

$$U(\Delta t) = e^{-iH\Delta t} = e^{-iH_{\mathrm{kin}}\Delta t} e^{-i(H_{\mathrm{V}}+H_{\mathrm{U}})\Delta t} + \mathcal{O}(\Delta t^2). \tag{4.31}$$

An important observation here is that for a simulation state $\Psi(x)$ stored in momentum space $|\psi\rangle^{\mathrm{KS}}$, the operator H_{kin} is diagonal with elements

$$H_{\mathrm{kin}} |k_1\rangle |j_1\rangle |h_1\rangle \ldots |h_P\rangle = \sum_p \sum_{v\in\{k,j,h\}} \frac{v^2}{2m_1} |k_1\rangle |j_1\rangle |h_1\rangle \ldots |h_P\rangle ,$$

i.e. the sum of the individual momenta of each sub-register in k-space. Even more conveniently, the kinetic part of the time evolution operator is then just the product of phases dictated by each sub-register separately.

$$e^{-iH_{\mathrm{kin}}\Delta t} = \prod_p \prod_{v\in\{k,j,h\}} \exp\left(-\frac{iv^2\Delta t}{2m_p}\right)$$

Similarly, the potential part of the Hamiltonian, $H_{\mathrm{U}} + H_{\mathrm{V}}$, is almost diagonal[26] for a state stored in position space $|\psi\rangle^{\mathrm{RS}}$ with elements

$$H_{\mathrm{U}} |n_1\rangle |m_1\rangle |l_1\rangle \ldots |l_P\rangle = \sum_p \frac{Z_p Z}{2\sqrt{x_{n_p}^2 + y_{m_p}^2 + z_{l_p}^2}} |n_1\rangle |m_1\rangle |l_1\rangle \ldots |l_P\rangle \tag{4.32}$$

[26] It would be exactly diagonal if the basis functions were actual Dirac deltas. For the purpose of the investigation in [212], they are treated as such, but note that this introduces a small error, especially for coarse-grained simulations.

and

$$H_{\mathrm{V}}|n_1\rangle|m_1\rangle|l_1\rangle\ldots|l_P\rangle$$
$$=\sum_{p\neq q}\frac{Z_pZ_q}{2\sqrt{(x_{n_p}-x_{n_q})^2+(y_{m_p}-y_{m_q})^2)+(z_{l_p}-z_{l_q})^2}}|n_1\rangle|m_1\rangle|l_1\rangle\ldots|l_P\rangle. \quad (4.33)$$

These phases do not separate quite as nicely as the kinetic part, but they are still diagonal operators. Therefore, so is the potential part of the time evolution operator $e^{-i(H_{\mathrm{U}}+H_{\mathrm{V}})\Delta t}$, which just multiplies appropriate phases given by Eqs. (4.32) and (4.33) onto the coefficients of the state.

All of these diagonal operators can be implemented using one- and two-qubit gates with cost quadratic in n_r; Reference [212] makes more detailed comments about their construction. Rather importantly in the present context, this form also excellently lends itself to being simulated on classical hardware. When these operators are applied to one of the computational basis states (given it is representing the physical state in the appropriate real- or momentum-space), the state remains the same, and its coefficient picks up a phase that is completely determined by its own bitstring. Therefore, application of all these operators is *embarrassingly parallelisable*[27] and thus comparatively fast in simulations.

These insights can now be used in the Trotterised time evolution operator in Eq. (4.31) to yield an efficient simulation scheme. However, because the kinetic and potential time evolutions are diagonal in different representations, it is necessary to insert (inverse) quantum Fourier transforms between them. This yields an evolution operator, which Ref. [212] is often referred to as the *split-operator* (SO) time evolution

$$U(\Delta t)\approx U_{\mathrm{SO}}(\Delta t):=e^{-i(H_{\mathrm{U}}+H_{\mathrm{V}})\Delta t}U_{\mathrm{QFT}}^{\dagger}e^{-iH_{\mathrm{kin}}\Delta t}U_{\mathrm{QFT}},$$

where U_{QFT} means the QFT of each individual sub-registers containing n_r qubits. Note that the operators $e^{-iH_{\mathrm{kin}}\Delta t}$ and $e^{-i(H_{\mathrm{U}}+H_{\mathrm{V}})\Delta t}$ are now diagonal.

Using this framework, Ref. [212] goes on to investigate multiple use cases and carefully examines the resource requirements, necessary temporal and spatial resolution,

[27]The term *embarrassingly parallelisable* refers to tasks that trivially decompose into a set of separate, independent tasks, lending themselves to distribute very efficiently across different processing units [235].

utility, and intricacies of simulations based on this principle in one, two, and three dimensions, noting important caveats and pitfalls one might encounter, as well as applying various techniques to increase its accuracy. The relevant part for this thesis is the simulation of a 3D helium atom, which is described in the following subsection.

4.2.2 Results

3D helium simulation

To demonstrate the suitability of the method when simulating an interacting system in three dimensions, this subsection will show results for a helium atom, i.e. two electrons interacting via Coulomb repulsion within an attractive central Coulomb potential. For the purpose of this simulation, the spin degree of freedom is ignored, as it does not play a role in the specific setup used.

At the start of the simulation, some initial state must be chosen. Eigenstates of the Hamiltonian would be one option to demonstrate numerical stability. However, the exact electron eigenstates of the helium atom cannot be solved analytically, so instead, the electronic states are initialised in an antisymmetrised combination of hydrogen wavefunctions[28]

$$\Psi_{n,l,m}(r,\theta,\varphi) = \sqrt{\left(\frac{2Z}{n}\right)^3 \frac{(n-l-1)!}{2n(n+l)!}} \left(\frac{2Zr}{n}\right)^l e^{-Zr/n} L_{n-l-1}^{2l+1}\left(\frac{2Zr}{n}\right) Y_l^m(\theta,\varphi)$$

with central charge $Z = 2$, which serves as an approximation. Without electron-electron interaction, such an initial state would indeed be an eigenstate. The two sets of quantum numbers (n,l,m) used for the initialisation were $(2,1,0)$ and $(2,1,-1)$, of which the former is the $2p_z$-orbital and the latter is the atomic orbital often denoted as $2p_{-1} = (2p_x - i\,2p_y)/\sqrt{2}$ with the usual orbitals $2p_x$ and $2p_y$. The full initial (triplet) state is then the antisymmetric wave function

$$\Psi_{\text{init}}(\underline{r}_1,\underline{r}_2) = \Psi_{2,1,0}(\underline{r}_1)\,\Psi_{2,1,-1}(\underline{r}_2) - \Psi_{2,1,-1}(\underline{r}_1)\,\Psi_{2,1,0}(\underline{r}_2). \tag{4.34}$$

The used simulation box is a cube with a side length of 25 a.u. The initial state Ψ_{init} given above is discretised at sampling points as per Eq. (4.30) using $n_r = 6$ qubits

[28] $L_k^j(x)$ is a generalised Laguerre polynomial of degree k, $Y_l^m(\theta,\varphi)$ is spherical harmonic of degree l and order m.

per subregister, which provides 64 divisions per axis and particle. The full 36-qubit state is then propagated forwards in time for 500 split-operator steps, where each step is of length 0.05 a.u., thus a total evolution of 25 a.u. is simulated. One detail worth noting for this simulation is the highly symmetric initial state. These symmetries could be exploited to make the simulation more efficient and/or precise. However, as an exploratory demonstration effort, the goal of this calculation was to show the capabilities of the method in a generic setting. Therefore, such optimisations are not included here.

Many one- and two-dimensional demonstrations in Ref. [212] use an extra ancilla qubit to perform phase estimation in order to track changes in the simulated state. However, since the classical resources required for this calculation are already considerable, and an additional ancilla qubit would double the memory requirements, the state change is tracked in a different way. At every time step, the three-dimensional probability density $\rho^{(1)}(t)$ of one of the electrons[29] is determined and recorded. In practice, this means calculating the probability associated with each of the 2^{18} position space grid points defined by the computational basis states of the three subregisters representing one of the particles. As a measure of how much the distribution of the electron deviates from its initial state, the Bhattacharyya coefficient [236]

$$\sum_n \sqrt{\rho^{(1)}(t)_n \rho^{(1)}(0)_n}$$

between the time evolved and initial probability distributions can be used. Here, the sum over n includes all grid points. This coefficient somewhat resembles a classical analogue of the usual inner product fidelity. Figure 4.8 shows the results.

As mentioned above, if the initial state were an eigenstate of the Hamiltonian, the expectation would be for the probability distribution to just stay constant. Fortunately, the Hamiltonian can be easily modified in order for this to be true. Simply removing the electron-electron interaction term reduces the problem to a hydrogenic one. Figure 4.8 also contains the Bhattacharyya coefficient results of such a calculation and confirms that this is indeed the case for this simulation. The probability distribution stays

[29]As electrons are indistinguishable, their probability distributions must be the identical.

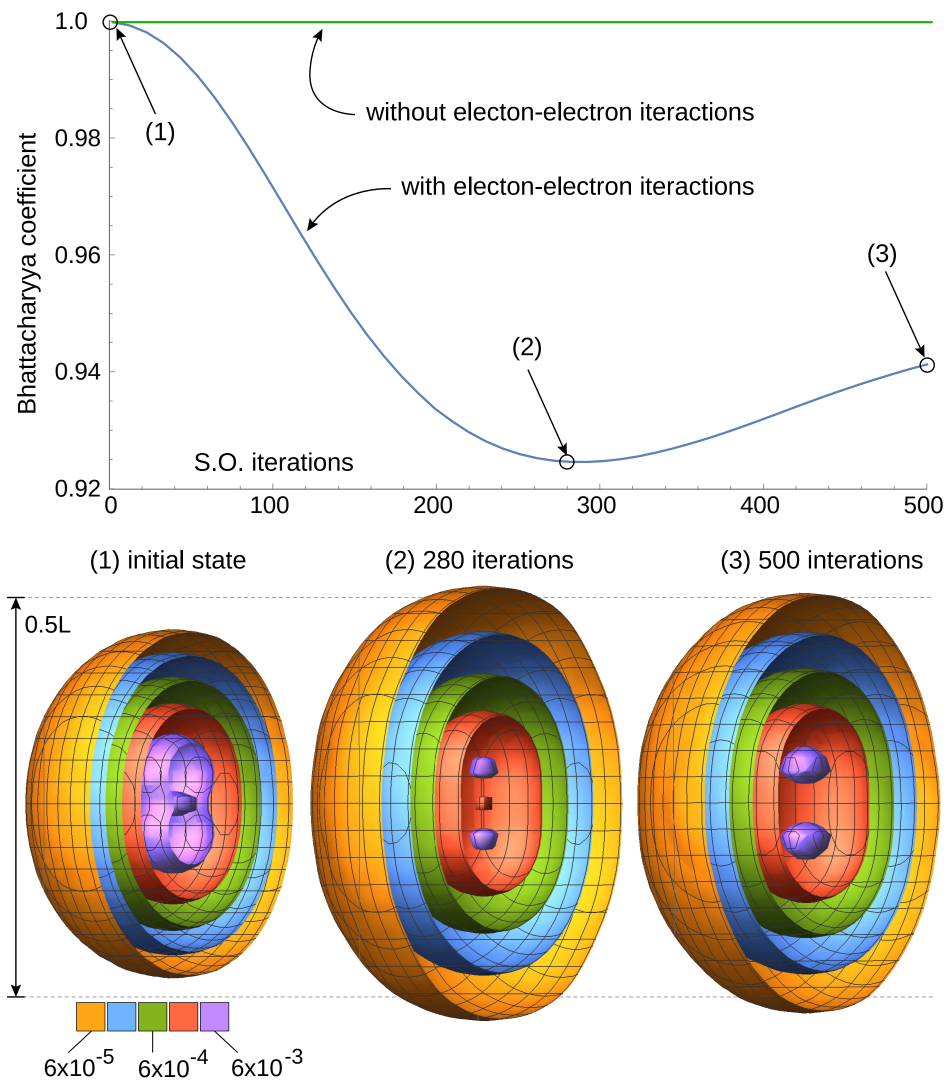


Figure 4.8: Bhattacharyya coefficients (top) — with and without *e*-*e*-interactions — and real-space electron density distribution of the helium simulation (bottom). The electron density is visualised using probability density isosurfaces at three distinct times: at the start of the simulation, when the electron cloud was maximally spread out, and at the end of the simulation. The 500 SO cycles correspond to propagation of 25 a.u. (≈ 0.6 femtoseconds). Data for this figure was generated by RM, visualisation by SB.

constant, which hints at a good choice of hyperparameters (simulation box size, grid spacing, etc.) for this calculation.

When the interaction between the electrons is included in the Hamiltonian, the initial state is not an eigenstate anymore. Thus, the probability distribution is expected to vary over time; Figure 4.8 affirms this behaviour. The charge is initially distributed rotationally symmetric about the z-axis, with some density accumulations just above and below the x–y plane. In the interacting case, the electrons partially shield each other from the core, thus the hydrogen wave functions with $Z = 2$ are concentrated too closely around the core. As time evolves, the charge initially spreads out away from the core, then returns slightly, but not to its initial distribution.

The results reported here were generated with pyQuEST (see Chapter 7), using its capability to deploy in a cluster configuration, where the numerical representation of a quantum state is partitioned between compute nodes cooperating over a network. This allows both the representation of states too large to fit into the memory of any single compute node, and their concurrent simulation — each multicore node is further able to parallelise its local simulation tasks through multithreading. Specifically, emulation of this 36-qubit quantum computer employed 32 compute nodes of the Oxford Advanced Research Computing (ARC) facility [237]. Each node contains 48 CPU cores, and took roughly 52 hours ($\approx 50\,000$ core hours) to process its 64 GiB partition of the full 1 TiB quantum state-vector. The time-limiting factor of the calculation was, as expected from the parallelisation argument made in Section 4.2.1, the quantum Fourier transform. As opposed to the phase gates, which can be calculated independently and massively parallel, the QFT requires communication between the nodes and is thus roughly an order of magnitude slower in the present case.

Space resource scaling

Using the results presented here and additional ones contained in Ref. [212], it is now possible to roughly estimate the quantum resources that would be required to simulate larger molecules of interest. Two candidates for such simulations are ammonia (NH_3), which is of profound importance in modern agriculture and for which methods of selective hydrogen atom removal are being investigated [238], and hexafluoro ethane (C_2F_6), which is a representative example of fluorocarbons [239] relevant in the chemistry of the ozone layer and in plasma etching.

To extrapolate from the knowledge gained in the simulations presented here and in Ref. [212], the following observations are useful. First, the maximum curvature of the molecular wavefunction should scale only linearly with the maximum core charge $Z_{\max}$ present in the molecule. This is reasonable when considering the solutions of the hydrogenic wavefunctions, which the core electron states (i.e. the most strongly curved) will closely approximate. Since the required resolution depends on this curvature, the number of grid points per dimension, which is 2^{n_r}, should also scale linearly with $Z_{\max}$, i.e. $e^{n_r} \propto Z_{\max}$.

Second, the side length of the simulation box needs to scale with the size of the molecule contained in it. A reasonable assumption is for the volume to scale roughly linearly with the number of contained particles P [212, 232]. For a cubic simulation box with side length L, this means $L \propto P^{1/3}$. To keep the same spatial resolution within the box, the number of qubits per sub-register must grow like $2^{n_r} \propto L$, and thus $2^{n_r} \propto P^{1/3}$.

Combining these insights leads to the conclusion that the number of qubits per particle and dimension n_r should roughly scale like

$$n_r \approx C + \log_2(Z_{\max}) + \frac{1}{3}\log_2(P), \tag{4.35}$$

with a constant C that is to be determined in this section.

Since n_r really depends on more characteristics of the problem besides $Z_{\max}$ and P, the variable C will not be an absolute constant, but rather fluctuate with other system properties, such as the geometry and electron configuration. One way forward is to proceed to estimate the required resources for the two molecules that have been identified as interesting above, consequently arriving at two example values for C. These will then give a rough idea of its range.

First, note that in the numerical modelling reported here, remarkably accurate and stable simulations can be achieved with as few as 6 qubits per x, y or z sub-register, i.e. 18 qubits per 3D particle. Methods such as iterative phase estimation employed in [212], requiring only one additional qubit, can then obtain eigenenergies with accuracy up to 6 decimal places, as shown in the Reference. The results in [212] using the so-called *augmented split-operator method* suggest that even core-peaked electronic states can be modelled with only a small increase in resolution. For the present resource analysis it can therefore be reasonable, even albeit optimistic, that $n_r = 7$ may suffice for systems with $P = Z = 1$. As argued above, for larger values of P and Z, more data points per dimension are necessary to compensate for differently sized simulation boxes and higher curvature of core states. These required changes are discussed below.

For molecules that are already well-understood, the ionisation potential can be used to estimate their long-range behaviour [240]. However, a more interesting application of quantum simulators is to explore molecular systems that have not yet been experimentally studied. Therefore, the following will provide a first-principles argument based only on the constituent atoms and their (approximate) presumed locations. The centre of the highest occupied hydrogen-like wave function of each atom is placed at the coordinates of each nucleus. Following this, its radial charge density is calculated according to Ref. [241], which also semi-empirically accounts for nucleus shielding. Summing up the contributions from all atoms gives an approximation to the total charge density if the electrons of different atoms were not interacting with each other, in what might be described as a no-chemistry approximation. The maximum of this total density on the surface of the simulation box then provides information about how strongly the electrons will interact with its boundary. For molecules of interest here, the atomic locations in their equilibrium geometry are taken from Ref. [224].

To get an idea of what an acceptable charge density at the box surface could be, the calculation of 3D helium without electron-electron interaction discussed above gives some valuable insight. From the electron configuration and box size of that simulation, its maximum surface electron density ρ_0 using the approximate method described above can be derived. Consequently, the simulation box for any other molecule might be considered sufficiently large whenever the method above yields a maximum surface electron density that does not exceed ρ_0. Continuing with relatively optimistic assumptions, this rule will be followed as a simulation box size estimator.

However, note that it is only a rough approximation to the extent of the electron cloud; specific scenarios might necessitate an increase. Interesting dynamics with substantial numbers of moving particles may require significantly larger boxes, as to not let substantial amplitude collide with the simulation boundary.

Reiterating from above, besides varying simulation box sizes, the changes in the required spatial resolution must also be taken into account. For $Z > 1$, the features of hydrogenic wave functions shrink by a factor of exactly Z^{-1}, which is usually also a good approximation for low-lying core states. To accurately resolve these electronic states, the number of grid points per unit length must therefore be increased by a factor of Z_{max}.

For Ammonia (NH_3), the more modest of the two molecules mentioned as interesting above, an optimistic estimate yields the following: The highest charge in the molecule is $Z_{\text{max}} = 7$. Therefore, the number of grid points must increase to 7 times that of the estimate for $Z = 1$. At the same time, the previously discussed method to determine the box size yields a side length of ≈ 1.1 times the length required for a single hydrogen atom, giving a total factor of $\approx 1.1 \times 7 = 7.7 < 8 = 2^3$, which means n_r must increase by 3 from its reference value of 7, i.e. $n_r = 10$ qubits per particle and dimension. Substituting these values in Eq. (4.35) and rounding where appropriate leads to a value of $C^{\text{NH}_3} \approx 6$. For 14 particles (10 electrons and 4 nuclei) in 3 dimensions, the total total number of qubits comes out to $3 \times 10 \times 14 = 420$. This might be rounded to 450, recognising that multiple ancillas may be required even in a very resource-effective implementation.

The more challenging case mentioned above was C_2F_6. The highest-charge nuclei are those of fluorine, giving $Z_{\text{max}} = 9$, thus increasing the required spatial resolution 9-fold. Surprisingly, as the electron densities of C and F drop off quite rapidly with increasing distance from the atom, the above described method suggests that the required side length of the simulation box could be as small as 0.85 times that of a single hydrogen atom. This results in a total factor of $\approx 0.85 \times 9 = 7.65 < 8 = 2^3$, again meaning that 3 qubits must be added to the reference value, yielding a total of $n_r = 10$. Using Eq. (4.35) again, for this molecule the constant term is $C^{\text{C}_2\text{F}_6} \approx 5$.

The total number of required qubits is $3 \times 10 \times 74 = 2220$, which might be rounded up to 2250 to allow for some ancilla overhead.

From these two data points, C can be cautiously estimated to roughly be on the order of ~ 10, but it should be noted that it might be lower for advantageous circumstances as in the examples given, or may also increase in unfavourable situations.

4.2.3 Discussion

This section elaborated on some of the aspects of simulating quantum chemistry using the split-operator method — a variant of Trotterisation — on a regular real-space grid, which is part of the exploration in Ref. [212]. The goal of the material presented here was two-fold. First, demonstrate that even a coarse-grained three-dimensional quantum chemistry calculation can yield numerically stable and physically reasonable results. Second, give a rough estimate of the quantum resources required for simulating quantum-chemically interesting molecules, as well as an approximate formula how these resource requirements might scale to even larger systems.

The first point was achieved via a resource-hungry 36-qubit simulation on a classical compute cluster. The relatively modest compute time can, at least in part, be attributed to the diagonal form of the kinetic and potential operators, whose elements furthermore have an algorithmic connection to their binary value and can thus be calculated on-the-fly. This enables massively parallel evaluation of their application to a state, leaving the quantum Fourier transform as the bottleneck in the gate-based classical quantum simulation software. Results of the time evolution of a 3D hydrogen atom with and without e-e-interaction show no signs of artefacts or numerical instability, suggesting that the grid spacing and time step used were sufficiently fine-grained for an accurate simulation. The physical behaviour of an expanding and then contracting electron density cloud is also in line with what might be expected from the initial state used. Therefore, these parameters can serve as a basis on which a cautious resource estimation may be performed.

The second point was accomplished via reasonable assumptions about the curvature of the wavefunction and what volume it would typically occupy. Using the above discussed simulation as a starting point, a simulation of the chemically interesting NH_3 was assigned a space cost of roughly 450 qubits. Although this is not too far off from quantum devices available today [242], it seems unlikely that NISQ devices would be able to perform such a calculation, keeping in mind that the simulation presented here was noise-free, while NISQ hardware is usually burdened with considerable noise and other constraints like qubit connectivity, making circuits as deep as in the simulation not feasible. Indeed, such a simulation task seems more suited for early fault-tolerant quantum hardware. Using the same technique, the practically also very relevant C_2F_6 was estimated to require about 2250 qubits, marking a considerable five-fold increase from ammonia, but still well within what is anticipated from fully error corrected devices in the future.

These observations make it seem reasonable that the techniques explored in Ref. [212] may have practical applications, once such devices are available. The manuscript [212] furthermore makes some comments about the potential runtime of such an algorithm, possible hardware architectures, and other potential uses of the split-operator method besides quantum chemistry. The interested reader is referred to the full text.

In this chapter I discussed two distinct methods of simulating the dynamics of a molecular system, from which, in principle, the energies of the Hamiltonian can be extracted via, for example, quantum phase estimation. Since the QPE algorithm is quite a costly one in terms of its implementation on a quantum device, a cheaper method of finding energies or their gaps would be very useful. The next chapter introduces such a method that can detect energy *gaps* of a Hamiltonian more efficiently.

We have to remember that what we observe is not nature in itself but nature exposed to our method of questioning.

— Werner Heisenberg

5 ALGORITHMIC SHADOW SPECTROSCOPY

This chapter presents part of a project published as Ref. [243], and closely follows parts of that manuscript. Bálint Koczor conceptualised the core idea and scope of the project. Hans Chan, Matt Goh, and myself worked on details and improvements to the algorithm, and each contributor implemented numerical demonstrations for different physical models and frameworks. My specific part was the simulation of the Fermi-Hubbard model with and without gate noise. The following explains key concepts of the work and focuses on my particular role in it, while leaving out details not relevant to, or not directly part of, my contribution. In places I use text verbatim from [243]; in those cases I was the original author of the text.

Contents

In Chapter 4 I elaborated on the importance of Hamiltonian simulation for potentially useful applications of quantum computers, and how quantum chemistry is one of the key areas where a practical benefit might be expected from such simulations.

However, solely generating a time-evolved state using these simulation techniques is not the end of the story. To infer properties like the energy level spacing of a system from its evolution, additional techniques like quantum phase estimation must be applied.

In this chapter, I will summarise and detail my contributions to a protocol that provides an efficient way to extract the magnitudes of energy gaps from the time evolution of a state using classical shadows [244] and sophisticated post-processing. As already discussed in the Literature Review, knowledge about these gaps often allows the calculation of crucial information about a system, like reaction rates, emission and absorption spectra, etc.

5.1 Method

The goal of the method presented here is to extract the energy gaps of Hamiltonian eigenstates which have significant overlap with some input state $|\psi\rangle$. Given a Hamiltonian H, whose eigenenergies will be denoted as E_k with their corresponding eigenstates being $|\varphi_k\rangle$, this statement can be formalised to finding all

$$\Delta E_{kl} = |E_k - E_l|$$

where

$$\left|\langle\psi|\varphi_k\rangle \langle\psi|\varphi_l\rangle\right| \not\ll 1.$$

Given access to *exact* expectation values of observables, this task is relatively straightforward. Starting from the Schrödinger equation for the input state $|\psi\rangle$ and expanding it into the Hamiltonian's eigenbasis yields

$$|\psi(t)\rangle = e^{-iHt}|\psi\rangle = \sum_k \underbrace{\langle\varphi_k|\psi\rangle}_{=:c_k} e^{-iE_k t},$$

which means that an observable O will oscillate like

$$\langle O(t)\rangle = \langle\psi(t)|O|\psi(t)\rangle = \sum_{k,l} \underbrace{c_k^* c_l \langle\varphi_k|O|\varphi_l\rangle}_{=:I_{kl}} e^{-i(E_k-E_l)t}. \tag{5.1}$$

The Fourier transform of the signal $\langle O(t)\rangle$ then reveals peaks at the energy differences $E_k - E_l$ with the respective intensities I_{kl}. So to observe the spectrum of a system, the time evolution $\langle O(t)\rangle$ can be sampled at discrete, equally spaced times t_m, and the discrete Fourier transform (DFT) of the obtained time signal shows the transition energies ΔE_{kl} as peaks in the spectrum.

There are two problems that arise from this method in practice. First, the intensities I_{kl} depend on the matrix elements O_{kl} of the operator O in the Hamiltonian eigenbasis. The magnitudes of these matrix elements are often not known *a priori*, and may be arbitrarily small for a chosen operator O, resulting in no signal, even when the initial state has considerable overlap with the relevant pair of Hamiltonian eigenstates. This point will be discussed momentarily. Second, the expectation values must be estimated from repeatedly sampled measurements, which means the time signal $\langle O(t)\rangle$ is typically burdened with shot noise, potentially overpowering the signals. A solution to this problem is described in Section 5.1.2.

5.1.1 Classical shadows

One possible solution to the first mentioned problem — not knowing the magnitude of matrix elements — is to sample a large number of them, in the hope that one (or several) will provide a strong signal. The technique of classical shadows allows exactly that in a resource-friendly way, i.e. it is possible to estimate many observables from very few measurements. Several variants of this technique exist [244–249], but the version most relevant to the present work is one that permits efficient estimation of expectation values of k-local Pauli strings.[30] This restriction to Pauli observables somewhat simplifies the specific procedure used to generate the results in this chapter, whose basic steps shall be reiterated here.

At every sample point in time t_m, a number N_{snap} of snapshots (often called shadows) is recorded. A single snapshot is produced as follows. For each qubit in the calculation, randomly and uniformly choose one of x, y, or z. If x is chosen, rotate the qubit by $-\pi/2$ about the y-axis; if y is chosen, rotate it by $\pi/2$ about the

[30]A k-local Pauli string is an operator which acts on at most k qubits with either a Pauli-x, -y, or -z operator.

x-axis; if z is chosen, perform no rotation. Subsequently, measure all qubits in the Pauli-z basis, which now effectively is a Pauli measurement on each qubit in either the x-, y-, or z-basis, where the direction for each qubit was chosen randomly. The information about the used measurement basis for snapshot q is stored in a vector $\underline{\mathcal{P}}_q$, with $q = 1 \ldots N_{\text{snap}}$, together with the measurement outcomes, which is a vector $\underline{b}_q$ with entries of ± 1 for each qubit.

After the measurements have been performed, the task is to estimate expectation values of k-local Pauli strings $P = \prod_{j \in \mathcal{I}} \sigma_j^{v_j}$, with $v_j \in \{x, y, z\}$, and where $\mathcal{I}$ contains $\bar{k} \leq k$ different relevant qubit indices.[31] A simple example would be $P = \sigma_2^x \sigma_4^y \sigma_5^x$, where $\mathcal{I} = \{2, 4, 5\}$, as well as $v_2 = v_5 = x$ and $v_4 = y$. To reconstruct one particular P from the collected snapshots $\{\underline{\mathcal{P}}_q, \underline{b}_q\}$, the following procedure can be used. First, for all recorded snapshots, check whether the measurement basis of that particular snapshot matches the direction v_j of the Pauli string of interest P for all indices $j \in \mathcal{I}$, i.e. $[\underline{\mathcal{P}}_q]_j = v_j \ \forall j \in \mathcal{I}$. Keep only snapshots for which this condition is fulfilled, and collect their indices into the set $\mathcal{Q}$. Then, a good estimate of the expectation value is

$$\langle P \rangle \approx \frac{3^{\bar{k}}}{N_{\text{snap}}} \sum_{q \in \mathcal{Q}} \prod_{j \in \mathcal{I}} [\underline{b}_q]_j.$$

Performing this reconstruction for every possible Pauli string which is at most k-local yields a large number of N_O observables, which can all be estimated from only a few snapshots at every time step. From here on, these operators will be called $O^{(n)}$, with $n = 1 \ldots N_O$.

Note how for this specific incarnation of classical shadows, the reconstruction has a very simple probability-theoretic interpretation. The measurement outcomes of those instances where the Pauli-string of interest was (by pure chance) actually measured are summed over, and finally divided by *expected* number of times the Pauli-string of interest would appear as a substring in all of the N_{snap} measurement bases. However, as many variants of the classical shadow technique exist, this observation does not generalise to all of them.

[31]The notation σ_k^v means, as elsewhere in this thesis, a Pauli-v operator on qubit k.

To deal with outliers, a median-of-means component is usually added to the reconstruction procedure. In this work, every collection of snapshots is divided into three equally sized batches, before the procedure described above is performed on each of them. The final expectation value is always taken as the median of these three batches. Note also here, that the method of classical shadows is more general than this, but for consistency, the described setup is used everywhere in this chapter.

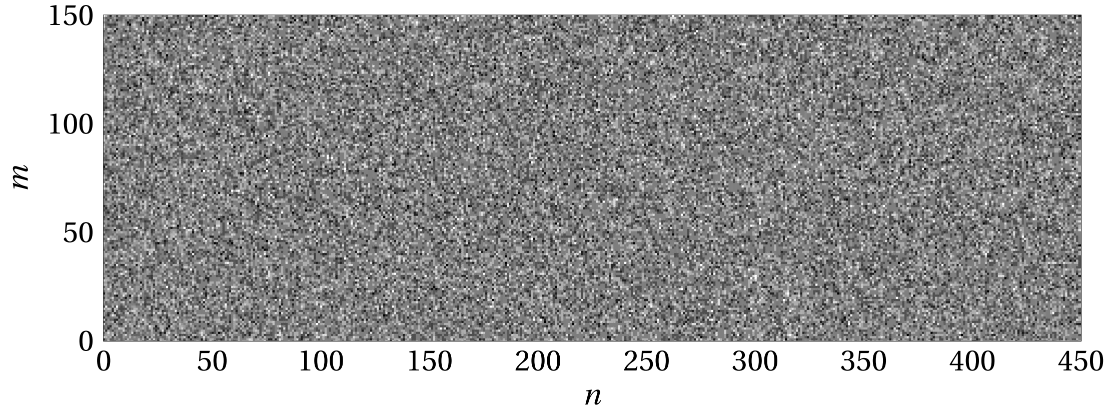
From the description above it is also obvious that the larger $\bar{k}$ is, the larger the shot noise of that observable will be, because, on average, it is measured fewer times. Therefore, the rest of this chapter will be limited to at most 3-local Pauli strings as observables, i.e. the observables are strings of Paulis that have 3 (non-identity) terms *or fewer*.

5.1.2 Post-processing

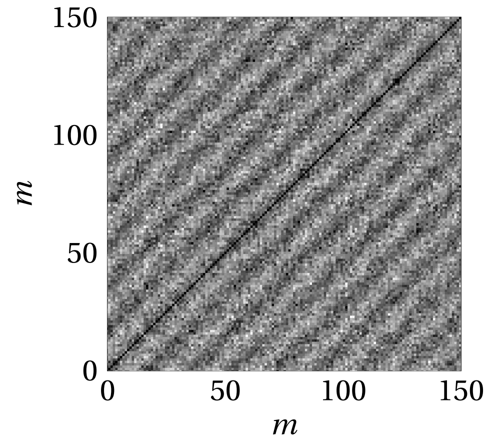
As just discussed, with the technique of classical shadows it is possible to estimate the expectation values of many observables from only few measurements. This increases the chance of including observables $O^{(n)}$ for which the matrix element $O^{(n)}_{kl}$ associated with a relevant energy gap ΔE_{kl} has a large magnitude. But, even when such observables are contained within the set of estimated Pauli strings, the signal is often still buried under shot noise, and directly Fourier transforming the time series of one observable $\langle O^{(n)}(t_m)\rangle$ yields no usable information. Figure 5.1a shows a small excerpt from the (standardised) signal matrix, where each column represents a different observable $O^{(n)}$, and each row is a different time t_m. While not a proof, it graphically illustrates that the raw signal reconstructed from classical shadows with only few measurements can look like mostly noise, even though in some columns some periodic pattern already emerges.

To continue the analysis, the signal data $\langle O^{(n)}(t_m)\rangle$ is first standardised such that the time signal of each observable has a mean of 0 and a standard deviation of 1,

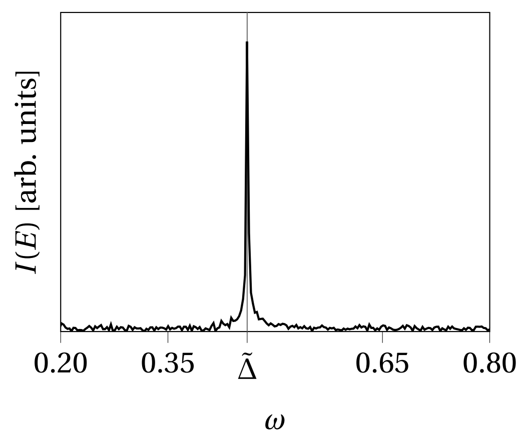
$$f_n(m) = \frac{\langle O^{(n)}(t_m)\rangle - \mu_n}{\sigma_n}$$



(a) Small section of the raw (transposed) data matrix $\mathbf{D}^{\intercal}$ of expectation values after standardising but before testing it for autocorrelations. Each pixel is the expectation value of one specific operator $O^{(n)}$ at some time index m. Time varies vertically, while horizontally the index of the (arbitrarily numbered) Pauli strings changes. Though upon close inspection some structure is visible, the matrix mostly looks like statistical noise.



(b) Bottom left section of the correlation matrix $\mathbf{C} = \mathbf{D}^{\intercal}\mathbf{D}$. A clear periodic pattern of stripes crossing the matrix diagonally from the bottom left to the top right is visible.



(c) Fourier transform of the dominant eigenvector of **C** showing a clear single peak in the spectrum with very high SNR. The exact theoretical value $\tilde{\Delta}$ of the expected peak position[32] is identical to the actual peak position.

Figure 5.1: Simple demonstration of the first steps of the algorithmic shadow spectroscopy procedure. Each Pauli string at each point in time was measured using only 50 shots. The model is a 1D 3-site Fermi-Hubbard model with the same parameters as in Section 5.2 (see there also for a description of the model), with a particularly simple initial state of $|\psi\rangle \sim |\varphi_0\rangle + |\varphi_1\rangle$. In this case, even just Fourier transforming the dominant eigenvector gives very good results. More involved systems with more peaks are better analysed using the full post processing described in the main text.

where

$$\mu_n = \frac{1}{N_O}\sum_m \langle O^{(n)}(t_m)\rangle \quad \text{and} \quad \sigma_n^2 = \frac{1}{N_O}\sum_m \left(\langle O^{(n)}(t_m)\rangle - \mu_n\right)^2 .$$

As a next step it proved beneficial to actively filter these time series $f_n(m)$ for statistically significant data, i.e. to discard signals that are indistinguishable from statistical noise. In this work, a Ljung-Box test[33] [252] — which tests each time series for autocorrelations — is used on each time series $f_n(m)$, and signals with a p-value smaller than 0.05 were discarded, i.e. only observables with a low probability of being purely noise are retained. For ease of notation, the remaining signals are collected into a data matrix $\mathbf{D}$ with the elements

$$D_{nm} = f_n(m).$$

The individual rows of this data matrix are very likely to contain some signal, but they are still burdened with a large amount of shot noise. To amplify the signal and get clean, strong peaks in the spectrum, a key step in the post-processing stage is to recognise that the time evolution of all observables follows the same underlying time dynamics according to Eq. (5.1), barring differences in the intensities. Therefore, it makes sense to calculate correlations between different time steps, across all remaining observables. For standardised data — as is the case here — the correlation is

$$C_{jk} = \frac{1}{N_O}\sum_n f_n(j) f_n(k).$$

This operation is concisely expressed as a matrix multiplication of $\mathbf{D}$ with its transpose.

$$\mathbf{C} = \frac{1}{N_O}\mathbf{D}^\intercal\mathbf{D}$$

Figure 5.1b shows a corner section of such a matrix $\mathbf{C}$ from one of the numerical calculations performed over the course of this project. There are clearly visible diagonal stripes in the plot, which have exactly the periodicity of the Rabi oscillations present in the system the data was generated from.

[32]This is the gap of the actually simulated system respecting the Trotterisation of the time evolution. It differs slightly from the gap of the exactly simulated system as elaborated in Section 5.2.1.

[33]The specific implementation used for the Ljung-Box test was the function `acorr_ljungbox` [250] from the `statsmodels` [251] Python package.

Because the matrix **C** now contains information about the temporal correlations of all expectation values deemed useful by the Ljung-Box test, its dominant eigenvectors, i.e. those whose eigenvalues have the largest magnitude, can be used to gain insight into the frequencies present in correlations, revealing the energy gaps in the Hamiltonian as discussed above.

In some cases, when the signal-to-noise ratio (SNR) is very high and only few peaks are present in the spectrum, the eigenvector corresponding to the largest magnitude eigenvalue alone yields very good results and its Fourier transform reveals clearly visible, easily detectable peaks, as shown in Fig. 5.1c.

However, to produce a cleaner and more accurate spectrum in cases with many and potentially small peaks, it can often boost the SNR to include a (small) number of c eigenvectors corresponding to the c largest-magnitude eigenvalues, which will be called v_ℓ with $\ell = 1 \dots c$. This process is essentially a principal component analysis (PCA) [253] of the signal matrix **D**. How large c should be is determined by manual inspection in this work, but heuristics for automated detection could be implemented for a more hands-off version of the method. To derive the final spectrum from the c most significant eigenvectors of **C**, a matrix function $\mathbf{X}(m)$ is generated, with the elements

$$X_{jk}(m) = \sum_n v_j(m+n)\, v_k(n).$$

It contains information about the temporal cross-correlations between the principal components v_j and v_k, after v_j has been shifted in time by m time steps. This pairwise correlation of periodic functions produces another periodic pattern, whose periodicity contains components from both functions. In this case, it is exactly the periodicity of the underlying signal data in **D**, but now with a higher SNR than simply using the dominant eigenvector alone. Fourier transforming each component of $\mathbf{X}(m)$ along its argument reveals the spectral density of the just described periodic signals in a new matrix $\mathbf{X}(\omega)$ with entries[34]

$$X_{jk}(\omega) = \mathcal{F}\left[X_{jk}(m)\right],$$

[34] I will use the frequency ω and the energy E interchangeably, depending on the conventions of the context.

which can be done efficiently using the fast Fourier transform (FFT). The largest magnitude singular value of $\mathbf{X}(\omega)$ at each frequency ω then reveals how large the amplitude of the "dominant component" is, i.e. the strength of the actual periodic signal without or with only very little noise. Therefore, at each ω, the final spectrum $I(\omega)$ is the absolute value of the largest-magnitude singular value of $\mathbf{X}(\omega)$.

Reference [243, App. C&E] goes into more detail and gives theoretical guarantees about the robustness of this method to shot noise, as well as noise caused by imperfect quantum processes. In short, the SNR is expected to scale like $\mathrm{SNR} \propto N_{\mathrm{T}} N_{\mathrm{snap}} \sqrt{N_O}$, where N_{T} is the number of different time indices t_m, N_{snap} again is the number of snapshots at each time step, and N_O is the number of considered observables that passed the autocorrelation test. Interestingly, the SNR scales with the *total* number of snapshots. The consequences of this are briefly discussed in Section 5.2.2.

5.2 Results

One era of quantum hardware in which algorithmic shadow spectroscopy may be useful is that of Hamiltonian simulation on early fault-tolerant devices.[35] These will be able to run deeper circuits than NISQ hardware, which puts simple time evolution of a quantum state via Lie-Trotter-Suzuki product formulas within reach. This is in contrast to NISQ implementations, where the time evolution of a system will probably have to be implemented using a VQE-type approach, because straightforward Trotterisation might lead to prohibitively deep circuits. However, even in early fault tolerant implementations via product formulas, some care must be taken, as expensive T-gate requirements [40, 57, 58] and some small level of noise in the system prohibit very deep circuits. Therefore, efficient methods to get the most possible information out of the simulation are crucial. This section will demonstrate in detail how shadow spectroscopy can be used to mitigate algorithmic errors, allowing coarser evolution time steps and thus saving resources, as well as how resilient it is against gate noise that may be introduced by the hardware during the simulation process.

[35]Reference [243] also analyses the utility of the method in the NISQ- and fully error corrected regimes of quantum computing. For a more complete picture the interested reader is referred to the available preprint.

The quantum system considered for this task is the Fermi-Hubbard model, which is of great potential for early quantum advantage. The Hamiltonian describing its dynamics is

$$H = -t \sum_{\langle i,j\rangle,\sigma} \left(c^\dagger_{i,\sigma} c_{j,\sigma} + c^\dagger_{j,\sigma} c_{i,\sigma} \right) + U \sum_i c^\dagger_{i,\uparrow} c_{i,\uparrow} c^\dagger_{i,\downarrow} c_{i,\downarrow}$$

where i and j number the lattice sites, $\langle\cdot,\cdot\rangle$ means pairs of neighbouring sites on the rectangular lattice, $\sigma \in \{\uparrow, \downarrow\}$ is the spin of the fermion, and $c^{(\dagger)}_{i,\sigma}$ are fermionic annihilation (creation) operators at site i with spin σ. To be treated on quantum hardware, this Hamiltonian must first be transformed into a representation of qubit Pauli operators[36] $H = \sum_{\ell=1}^{L} H_\ell$ via the Jordan–Wigner (JW) transformation. The model used in this work uses a chain with open boundary conditions and parameters $t = 1$ and $U = 2$.

For demonstration purposes, it is useful to limit the observed spectrum to a single peak, which more clearly shows how its reconstruction changes when adjusting external parameters like the Trotterisation time step or the type and severity of hardware noise. To this end, the relatively small example system of a grid of 3×2 sites, which results in a Hamiltonian on 12 qubits, is first solved numerically by exact diagonalisation, and the initial state is set to $|\psi\rangle \propto |\varphi_0\rangle + |\varphi_1\rangle$. Analytically, this initial state should give a spectrum with a single delta-peak at the energy gap $\Delta := E_1 - E_0$ between the ground- and first excited state, which is verified quite straightforwardly.

5.2.1 Algorithmic errors

The first demonstration will assume perfect gates and analyse only the influence of algorithmic errors on the peak position. As discussed above, early fault-tolerant devices are expected to be quite limited in the maximum gate depth of circuits they can apply to states. Therefore, the method of Hamiltonian time evolution considered here is the first-order Trotter-Suzuki formula [111, 112]. This means that time evolution between measurements is implemented using the approximation

$$e^{-iH\Delta t} = \prod_{k=1}^{N_{\text{Trott}}} \prod_{\ell=1}^{L} e^{-iH_\ell \delta t} + \mathcal{O}(\delta t^2),$$

[36]Hamiltonian generation and Jordan–Wigner mapping were automatically performed using OpenFermion [218].

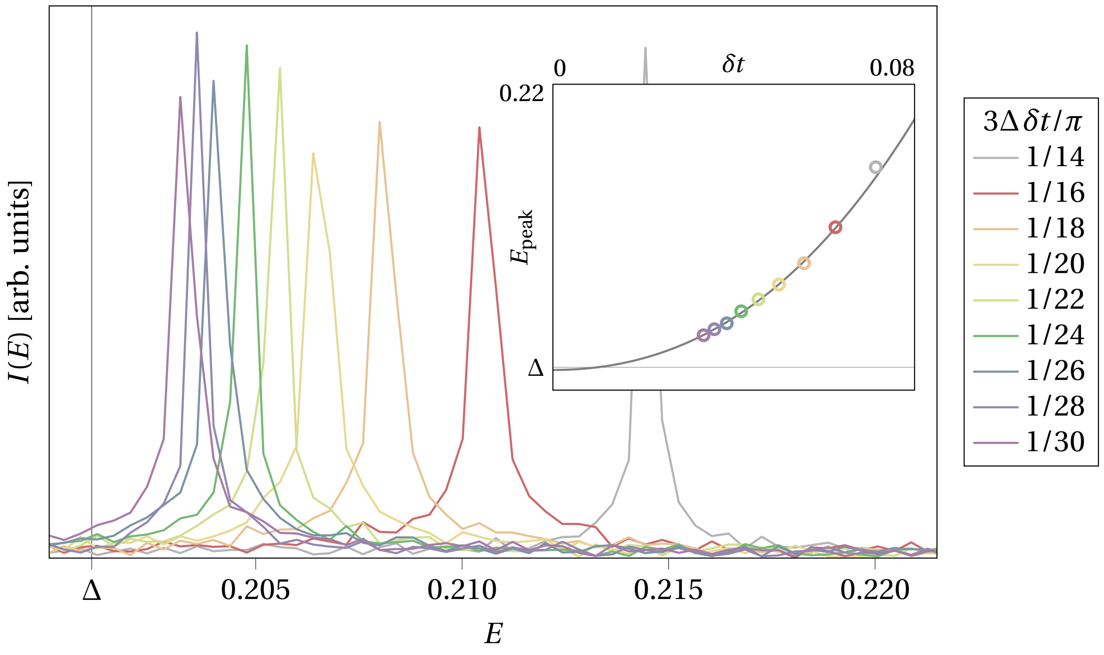


Figure 5.2: Spectra obtained using Lie-Trotter-Suzuki time evolution for a Fermi-Hubbard model with an exact spectral gap of Δ. Graphs for increasing time step sizes δt — chosen as whole-numbered fractions of $\pi/3\Delta$ — show peaks quite far away from Δ due to significant algorithmic errors. Classical shadows of only $N_{\mathrm{snap}} = 150$ snapshots at 3000 different times t_m were used to estimate expectation values of all up to 3-local Pauli observables. Inset: positions of the peaks as a function of δt used; fitting a cubic polynomial (to all but the largest step size) and extrapolating to $\delta t \to 0$ allows an accurate estimation of the exact gap Δ.

with the Hamiltonian $H = \sum_{\ell=1}^{L} H_\ell$ as obtained via the JW transformation, and an appropriately large N_{Trott}, such that $\delta t := \Delta t / N \mathrm{Trott} \ll 1$.

As the resolution of the finite grid in energy space δE is inversely proportional to the total simulation time T — which is a straightforward consequence of the Fourier transform [254] — the system must be evolved for a long time to accurately resolve the energy peaks of its spectrum. In order to implement such a long time $T = 1000\pi/\Delta$, the time step δt is chosen to be relatively large, as this keeps the number of gates feasible. This, in turn, introduces a significant algorithmic error between the system Hamiltonian and the actually implemented time evolution. Figure 5.2 shows the reconstructed spectra for different values of δt and confirms that the peaks for each δt are quite far from the exact energy gap due to these errors.

To illustrate how a much more accurate estimate of the gap can be extrapolated from this data, the inset of Fig. 5.2 plots the positions of the peaks E_{peak} as a function of the Trotter time step δt. Because the error of a first order Trotter time evolution is of $\mathcal{O}(\delta t^2)$ [129], a polynomial of the form

$$E_{\text{peak}}(\delta t) = \Delta_{\text{est}} + \alpha\,\delta t^2 + \beta\,\delta t^3$$

can be fitted to the data, with Δ_{est}, α, and β as the fit parameters.[37] The extrapolation $\delta t \to 0$ leaves only the constant term Δ_{est}, which is therefore a good estimate of the true spectral gap Δ. For the present case, this procedure gives $\Delta_{\text{est}} = 0.2009 \pm 0.0003$, which has the real gap $\Delta = 0.2010$ well within its margin of error. Reference [255] analyses this approach in more detail and gives rigorous bounds for how accurately the extrapolation of Trotter errors approximates the true energies. This type of error mitigation via extrapolation seems likely to also lend itself to other time evolution techniques, so long as the dependence of the error on some hyperparameter(s) is known to some extent — as is the case for Trotterisation and δt — for example qDRIFT [125] and stochastic time evolution [127].

The analysis in the present case was somewhat simplified by having only a single peak present in the spectrum, which is easy to track across different values of δt. For more complicated spectra with multiple peaks, this monitoring might necessitate a bit more care to reliably identify each peak in each spectrum and associate it with the corresponding one for a different time step size.

Overall, the presented approach is a powerful tool to accurately estimate the true spectral gap in the presence of algorithmic errors.

5.2.2 Shot noise

It is also interesting to explore the limits of the method in terms of how few measurements can suffice to generate a usable signal, i.e. how resistant it is to shot noise. Figure 5.3 shows results for one of the calculations presented in Section 5.2.1, but with varying numbers of measurements per time step. It is quite remarkable how few snapshots are actually required to retrieve some meaningful data, albeit with low SNR. At only 20 shots at every time step, some faint signal becomes visible in this case. As

[37] Depending on the time step sizes used in the simulation, more or fewer powers may be used, but the linear term should always be zero.

mentioned above (and Ref. [243] explains in more detail), the quality of the signal is expected to increase with the *total* number of measurements. This means that, as long as there are enough time steps involved in the simulation — which is often necessary anyway to arrive at the required combination of maximum resolvable energy and energy grid resolution — it is possible to work with very minimal data per time step.

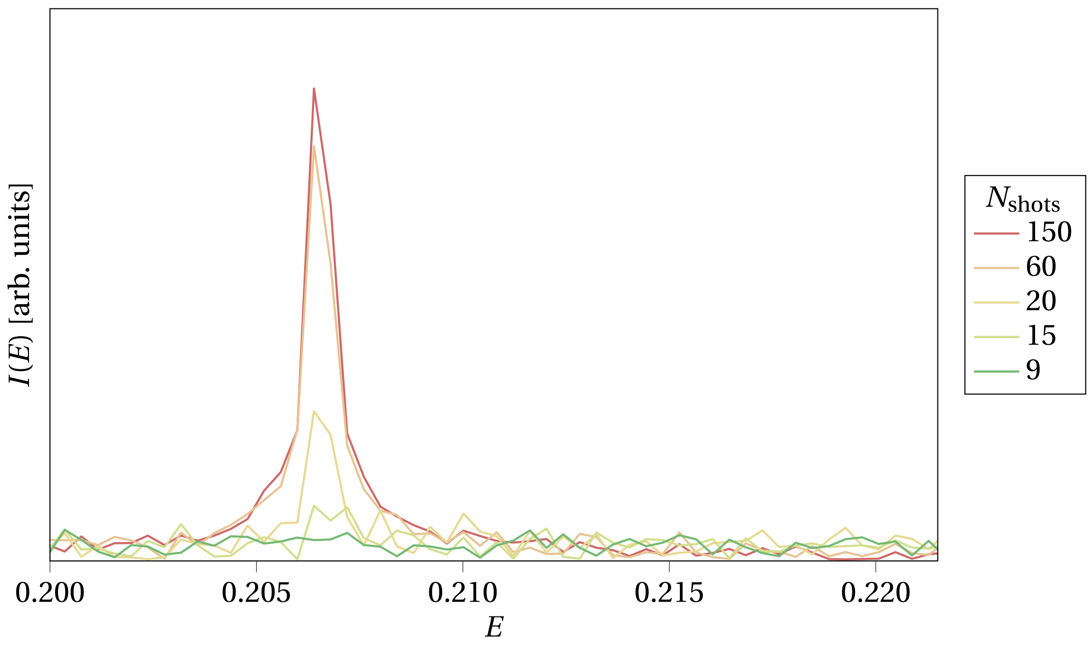


Figure 5.3: The peak of $3\Delta\delta t/\pi = 1/20$ in Fig. 5.2 reconstructed using varying numbers of snapshots at each of the 3000 different time steps. With $N_{\text{snap}} = 60$, almost no loss in peak height is observed, and even at $N_{\text{snap}} = 20$, intensity can clearly be discerned. Only at 15 and 9 shots per time step does the signal completely disappear in the noise.

5.2.3 Gate noise

In addition to errors introduced by a specific time evolution algorithm and the shot noise from a limited number of measurements, it is also worthwhile to investigate how hardware noise influences the positions and heights of the peaks in the recovered spectra. The time evolution method is again a first order Trotter product formula, but in this case the peak shapes and positions for a fixed δt will be compared to that of a noise-free simulation.

Noise models

I will demonstrate the influence of three distinct noise models which are likely to be relevant in early fault-tolerant devices. They can be described as follows.

Isotropic depolarising The first case is to assume that the dominant source of errors is a depolarising channel with equal probabilities. This is a frequently used model to investigate the robustness of error-correcting codes, because if a code has the ability to correct n-qubit depolarising noise, this means it can correct any type of n-qubit noise [26]. For early fault tolerant hardware, it seems reasonable to assume that not all such errors can be corrected, and that some will make it through to logical qubits.

In the used model, the JW transformation of the hopping terms together with the used Trotter product formula naturally results in multi-Pauli rotations of the form

$$\exp\left(-i\,\delta t\,\sigma_i^{\nu}\sigma_{i+1}^{z}\dots\sigma_{j-1}^{z}\sigma_j^{\nu}\right). \tag{5.2}$$

with $\nu \in \{x, y, z\}$. Although not directly employed in the calculations presented here, the σ^z terms sandwiched between qubits i and j can be removed by introducing a network of Fermionic SWAP (FSWAP) gates, which only consists of local gates of depth $\mathcal{O}(\sqrt{N})$ [256–259]. This motivates an error model whereby 2-qubit depolarising noise acts on qubits i and j after each of these terms with probability λ, described by the channel

$$\Phi_{i,j}^{\lambda}(\rho) = \left(1-\frac{16\lambda}{15}\right)\rho + \frac{\lambda}{15}\sum_{\nu_1,\nu_2\in\mathcal{S}}\sigma_i^{\nu_1}\sigma_j^{\nu_2}\rho\,\sigma_j^{\nu_2}\sigma_i^{\nu_1}, \tag{5.3}$$

where $\mathcal{S} = \{\mathbb{1}, x, y, z\}$, σ_i^{ν} is a Pauli-ν operator on qubit i and $\sigma_i^{\mathbb{1}}$ is the identity. Single-qubit rotations on qubit i are burdened with the equivalent depolarising channel for a single qubit at the same noise probability, given by the channel

$$\Phi_j^{\lambda}(\rho) = \left(1-\frac{4\lambda}{3}\right)\rho + \frac{\lambda}{3}\sum_{\nu\in\mathcal{S}}\sigma_j^{\nu}\rho\,\sigma_j^{\nu}. \tag{5.4}$$

Anisotropic depolarising This model is similar to the above, but every single-qubit operator is followed by the noise channel of the form

$$\overline{\Phi}_j^{\lambda}(\rho) = (1-\lambda)\rho + \frac{9\lambda}{10}\sigma_j^z\,\rho\,\sigma_j^z + \frac{\lambda}{20}\sigma_j^x\,\rho\,\sigma_j^x + \frac{\lambda}{20}\sigma_j^y\,\rho\,\sigma_j^y. \tag{5.5}$$

This means that not every Pauli error has equal probability anymore, but instead Pauli-z errors make up 90 % of all error events, and the rest is split evenly between Pauli-x and Pauli-y errors. Due to this anisotropic distribution, in contrast to above, multi-qubit Pauli gadgets as in Eq. 5.2 are followed by application of the just described single-qubit noise channel on each of the edge qubits i and j. Note that because of its implementation as two single-qubit noise events, this channel differs from the multi-qubit description used for isotropic depolarising noise by more than just the probability distribution between the x-, y-, and z-Paulis.

Anisotropic depolarising and damping A model identical to the anisotropic depolarising noise described above, but every application of $\overline{\Phi}_j^{\lambda}(\rho)$ is followed by a damping channel

$$\mathcal{N}_j^{\gamma}(\rho) = K_0^{(j)}\rho\,K_0^{(j)\dagger} + K_1^{(j)}\rho\,K_1^{(j)\dagger} \tag{5.6}$$

with the usual damping Kraus operators

$$K_0^{(j)} = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}_j \quad \text{and} \quad K_1^{(j)} = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}_j$$

on qubit j. The probability of a damping event is 1/10 that of a depolarising event, i.e. $\gamma = \lambda/10$.

Simulations

For the following discussion it is convenient to define the *circuit error rate* ξ as

$$\xi := \lambda N_{\text{gates}},$$

with the error probability λ as above, and N_{gates} meaning the number of gates in the circuit of the longest time evolution. Its usefulness relating to the final state fidelity will become apparent in the following.

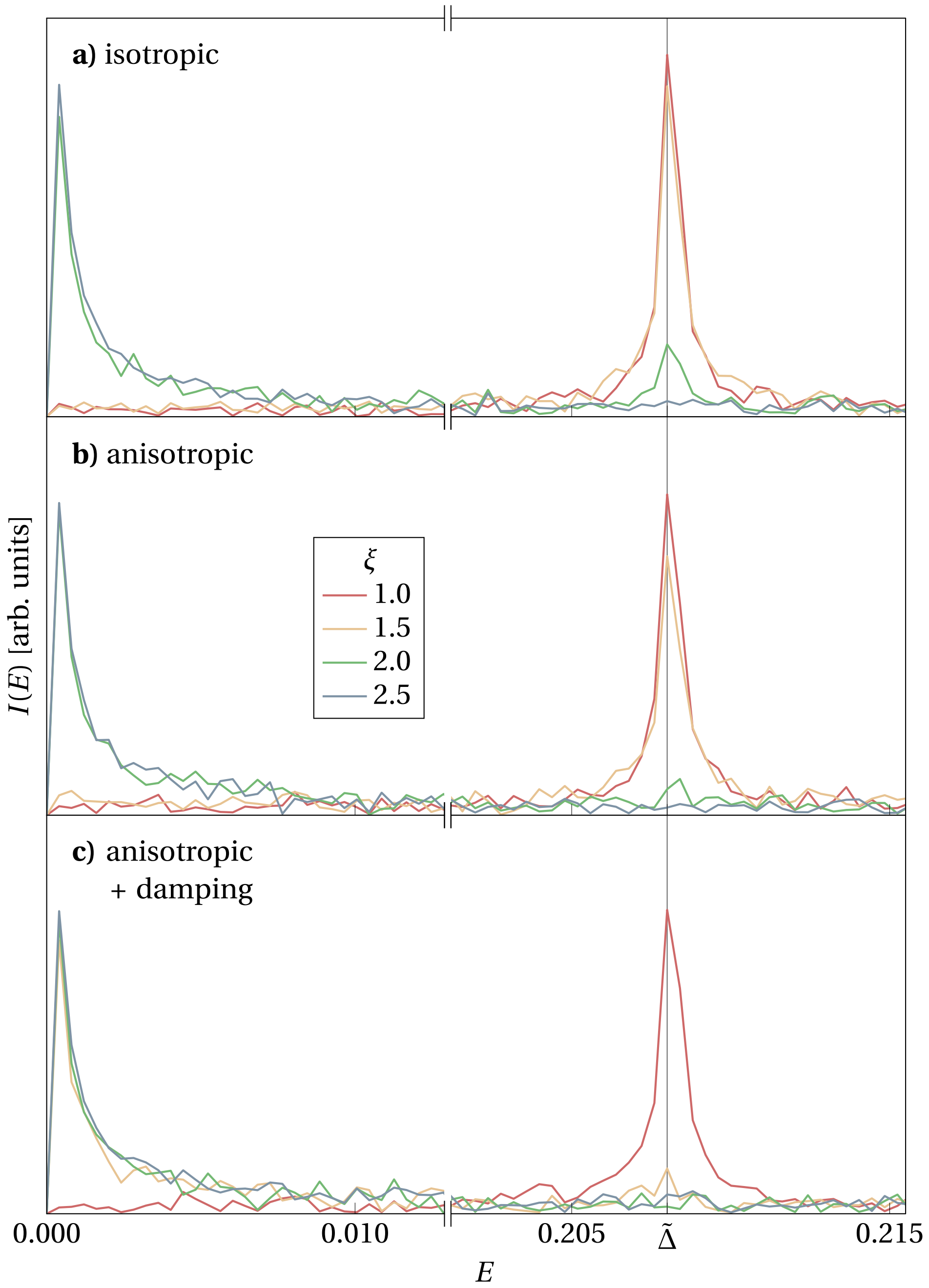


Figure 5.4: Relevant regions of the energy spectra obtained from simulations of the Hubbard model using different noise models and various circuit error rates $\xi = \lambda N_{\text{gates}}$. Each spectrum uses 150 snapshots per time step for the reconstruction of expectation values. Notice the interruption of the E-axis. The position of the peak without noise using the same Trotter time step is marked as $\tilde{\Delta}$. The results demonstrate robustness to various types of noise, as the peak *position* remains unchanged independent of noise strength and model used. However, the peak intensity vanishes for very high error rates. **a)** Isotropic depolarising noise with single gate error probability λ. **b)** Anisotropic depolarising noise with $0.9\,\lambda$ probability of Pauli-z errors, and $0.05\,\lambda$ probability each of a Pauli-x or a Pauli-y error. **c)** Anisotropic depolarising noise like in *b)*, but each depolarising channel is additionally followed by a damping channel with probability $\lambda/10$.

Figure 5.4 shows the results of simulations using each of the discussed noise models at different circuit error rates ξ. The simulation containing isotropic depolarising noise in Fig. 5.4 a) displays spectra at four distinct error rates for otherwise identical models, with ξ taking values either side of the magnitude where the signal becomes unusable. It confirms that shadow spectroscopy is indeed not greatly influenced by reasonable levels of gate noise — in this case $\xi \approx 1.0$ and 1.5 — as the relevant peaks at $E \approx 0.2$ are very pronounced and centred exactly at the location $\tilde{\Delta}$ of the noise-free peaks.[38] At $\xi \approx 2$, the peak drastically diminishes in height, and a new component close to $E \approx 0$ emerges. Finally, at $\xi = 2.5$, the signal of the system at $E \approx 0.2$ completely disappears, and all the weight apart from the random noise has shifted over to $E \approx 0$.

As briefly mentioned in Section 5.1.2 and discussed in more detail in Ref. [243], this behaviour is not unexpected. Every application of the error channel (i.e. every applied gate under the discussed error model) only retains the amplitude $(1 - 16\lambda/15)$ of the state. Restricting this analysis to two-qubit noise for simplicity, the fidelity F of the simulated density operator $\rho(T)$ at the end of the simulation with the ideal state $|\psi(T)\rangle$ is

$$F = \langle\psi(T)|\rho(T)|\psi(T)\rangle = \left(1 - \frac{16\lambda}{15}\right)^{N_{\text{gates}}} \approx e^{-\xi}$$

where the approximation holds for deep circuits with $N_{\text{gates}} \gg 1$.[39] This exponentially decaying fidelity replaces the periodic signal with statistical noise over the time of the simulation, until for sufficiently large ξ the signal is not recoverable anymore by the method described in this chapter. This reasoning applies more broadly than for the specific task discussed here. For example, in error mitigation techniques it is also often impractical to extract accurate information from the state after errors have suppressed the fidelity exponentially [260, 261]. However, in contrast to many other techniques, Fig. 5.4 a) shows that while below the threshold of ξ where the signal disappears completely, the position of the maximum remains unchanged even at relatively large error rates. Therefore, no further techniques are required to extract accurate data, as the errors only change the intensity, but not the positions of the peaks.

[38]Recall that the peak position is offset from the true gap Δ due to algorithmic errors.

[39]This follows straightforwardly from the definition $e^x = \lim_{n\to\infty}(1 + x/n)^n$, $\xi = \lambda N_{\text{gates}}$ and $16/15 \approx 1$.

One interesting property to note in this context is that in the presence of strong noise, where the peak in the reconstructed spectrum has disappeared completely, it is often still possible to retrieve useful information about the energy gaps by only retaining some portion of the data for a shorter total time $\overline{T} < T$. Of course, this lowers the spectral resolution of the result, but this is preferable to recovering no signal at all.

There is currently no automated process in the proposed protocol to detect cases like this and autonomously lower the maximum time considered, but it might be a valuable addition in the future. A simple extension of the Ljung-Box test to not only sort out observables that give no meaningful signal, but also the time cutoff after which noise dominates the signal, could be sufficient.

Shifting focus to Fig. 5.4 b), the pattern remains largely the same. The result for $\xi = 2$ shows the signal peak somewhat smaller than in plot a), but this can be attributed to the randomness of the classical shadow procedure. Because this error rate is right at the edge of a barely detectable signal, performing multiple calculations with the same parameters will sometimes produce a faint signal, and sometimes none at all. However, as in Fig. 5.4 a), for $\xi = 1$ and 1.5, the plot again shows a pronounced peak at the same position as expected.

Finally, Fig. 5.4 c) also exhibits its peak for $\xi = 1$ at the same energy $\tilde{\Delta}$, just like a) and b), indicating that shadow spectroscopy is very robust not only to isotropic and anisotropic depolarising noise — as would be expected from the arguments in Ref. [243] — but even to damping errors, to which that analysis does not strictly apply. Note, however, that the signal in Fig. 5.4 c) loses intensity much earlier than a) and b); already at $\xi = 1.5$ the peak is barely visible. This can be attributed to the fact that there is an extra contribution of $\lambda/10$ to the noise in these simulations, which ξ does not account for. But, the important feature of unchanged peak position for the case where noise is present also holds in this case.

5.3 Discussion

The work presented in this chapter introduced a method that, via a combination of the well-established technique of classical shadows and post-processing using many ideas from classical signal processing, can retrieve spectra of the energy differences of Hamiltonian eigenstates that have macroscopic overlap with an initial state in an efficient manner. By evaluating a large number of N_O observables, there is a good chance that at least some of them will contain a strong signal. Furthermore, theoretical analysis suggests that the SNR of the spectra should scale like $\propto \sqrt{N_O}$, while gate noise is suppressed as $\propto N_O^{-1}$. Crucially, gate noise only affects the height of the peaks (the SNR), but not their position. These properties of strong shot- and gate-noise resilience were confirmed numerically by recovering spectra of the Fermi-Hubbard model from minimal available data, and in the presence of noise generated by various models.

Because the quality of the recovered expectation values from classical shadows reduces exponentially with the locality of the observables, only up to 3-local Pauli strings were used in the analysis of the data. This introduces a possible limitation of the method. If the transition from one state to another requires many Pauli operators to be applied at once, the shadow spectroscopy method as presented here will not produce a peak for this transition. It is easy to construct an adversarial example, e.g. an artificial Hamiltonian with a ground state of $|00\ldots0\rangle$ and a first excited state of $|11\ldots1\rangle$. Fortunately, based on empirical observations, it seems that cases like this are the exception rather than the norm. For example, Ref. [243, App. F] details how in the context of quantum chemistry, single electron excitations mapped to qubits via the Jordan–Wigner transformation can be captured by 2-local Pauli operators.

An easily overlooked, but important, further limitation is the preparation of the initial state. I have completely circumvented this problem in this chapter by first analytically solving the system, and then using this information for state initialisation. Obviously, real-world problems do not permit this course of action. In general, preparing a good initial state remains difficult and can be attributed to the exponential hardness of finding ground- and eigenstates of a system [70]. However, as mentioned in the Literature Review, there are many methods for preparing a state which only occupies some eigenstates of interest, some relying on heuristics, while others are only applicable to problems of a particular structure. Even the method described in Chapter 6 could potentially be used in combination with shadow spectroscopy for this preparation step, if some bounds on the ground state energy are known. In any case, the efficiency of this initial step is mostly determined by the specifics of the underlying physical problem, and will thus not be discussed here any further, but it is important to keep this caveat in mind when considering potential applications.

Despite these limitations, shadow spectroscopy can be very practical in applications like quantum chemistry, high-energy physics, materials science, or any other field in which information about spectral gaps provides useful insights into the physics of a system. It may also be useful in entanglement spectroscopy [262, 263], where the transition energies in a learned "entanglement Hamiltonian" correspond to the entanglement spectrum of a subsystem. The presented method may provide a way of observing that spectrum experimentally with many fewer measurements.

Having obtained information about the energies present in the eigenspectrum of a Hamiltonian, it can be interesting to prepare one of the eigenstates to investigate its properties. The next chapter discusses one method how this might be done in a resource-friendly way.

If at first you don't succeed, try, try again.

— Edward Hickson

6 HAMILTONIAN EIGENSTATE PREPARATION

The findings in this chapter were published as Ref. [264], and the discussion here will closely follow the manuscript, using wording verbatim in places where I was the original author of the text. Most of the investigation was undertaken by myself, with occasional feedback on details and some conceptual ideas contributed by Simon Benjamin.

Contents

As discussed in the previous chapter and repeatedly throughout this thesis, often the energy spectrum of a quantum mechanical system is of key interest. But, once the energy of a state is known, it is often useful to actually prepare that specific state on a quantum register to perform further analysis on it. This analysis can include inspecting the symmetries of a state, determining other properties not directly reflected by the Hamiltonian (e.g. magnetisation of a spin system in the absence of an external magnetic field), or observing its time evolution under a different Hamiltonian, e.g. after a quench.

This chapter explores a method to efficiently prepare a state ε-close to a Hamiltonian eigenstate of known energy, using ideas closely linked to quantum phase estimation and related algorithms. The necessary components of the proposed algorithm are a single ancilla qubit, controlled real-time evolution of the simulated system, and an (ideally easy to generate) initial state that has considerable overlap with the desired target state.

The starting point of the present investigation can be considered from two different perspectives. The first is the *probabilistic imaginary-time evolution* (PITE) [232] (see also Ref. [265]), which superposes forwards and backwards real-time evolution as part of its protocol. The other is the basic circuit building block of the proposed method as a variant of the iterative quantum phase estimation algorithm [145, 146, 266], but with a shifted energy spectrum, which makes it easier to prepare the target state. Indeed, I will show that the quantum circuits arising from these two different origins are functionally equivalent in the present context.

Another closely related method is the *Rodeo algorithm* [175, 267], where repeatedly time-evolving the system for random durations results in rapid convergence to the target state. For a single iteration, the same basic circuit construction as in Ref [175] is used. However, due to its stochastic nature, rigorous bounds and guarantees for the Rodeo algorithm are difficult to obtain. This problem is circumvented by proposing deterministic choices for the durations instead, which allows more careful evaluation of the properties of the resulting state.

In the analysis of the algorithm, rigorous bounds are provided for the target state fidelity after a certain number of iterations, as well as the total success probability and the expected cost of the preparation procedure. This includes cases which capture algorithmic errors and noisy gates. These bounds are numerically verified by preparing the ground state of the electronic structure Hamiltonian of LiH in second quantisation using simulated quantum hardware.

Finally, a variant of the procedure is discussed, which uses interpolation between a trivial and the target Hamiltonian. Numerical results indicate that the total preparation cost can sometimes be lowered by first driving the state towards an eigenstate of an intermediate Hamiltonian, before converging to the target eigenstate. This procedure resembles the coarse-grained limit of a measurement-driven (i.e. Zeno effect based) adiabatic preparation (see e.g. Ref. [268]).

6.1 Method

This section contains a detailed description of the method to prepare Hamiltonian eigenstates, shows appropriate circuit implementations, and includes derivations of bounds on the quantities of interest.

6.1.1 Formal objective

First, consider a Hamiltonian H on n qubits — whose eigenstates are denoted as $|\varphi_j\rangle$ with their corresponding energies being E_j — and an arbitrary initial state $|\psi_0\rangle$. The goal is to produce a sequence of states $|\psi_k\rangle$ that converges to a desired eigenstate of the Hamiltonian $|\varphi_v\rangle$,

$$\lim_{k\to\infty} |\psi_k\rangle = |\varphi_v\rangle\,,$$

and determine an iteration index $\bar{k}$ at which it can be guaranteed that the target state infidelity falls below a given threshold

$$1 - \left|\langle\varphi_v|\psi_{\bar{k}}\rangle\right|^2 \leq \varepsilon.$$

6.1.2 Requirements

The presented method requires the following information and operators, which are assumed to be available. First, the energy E_v of the target state must already be known to some approximation. This approximate energy is denoted as $\tilde{E}$, where $E_v = \tilde{E} \pm \delta$, with some error bound $\delta \geq 0$.

Second, a lower bound Δ on the minimum difference of E_ν to any other energy E_j,

$$\Delta \leq \min_{j\in\mathcal{S}} |E_\nu - E_j|,$$

is required, where

$$\mathcal{S} := \left\{ j \mid j \neq \nu \text{ and } \left|\langle\varphi_j|\psi_0\rangle\right|^2 \not\ll \varepsilon \right\}$$

only contains indices for which the input state $|\psi_0\rangle$ has macroscopic overlap with the corresponding eigenstate.

It is also assumed that an equivalent upper bound for the largest energy difference

$$E_{\text{max}} \geq \max_{j\in\mathcal{S}} |E_\nu - E_j|$$

is available. For ground state preparation tasks, Δ is the spectral gap, and E_{max} is the largest occupied energy. In the worst case, E_{max} would equal twice the operator norm $\|H\|$ of the Hamiltonian.

Lastly, in addition to standard Pauli-z rotations and Hadamard gates, it will be necessary to apply a *controlled real-time evolution* (RTE) $U(t) := e^{-iHt}$ to the system. This obviously implies knowledge of H in a form suitable for a circuit-based realisation.

6.1.3 Circuit implementations

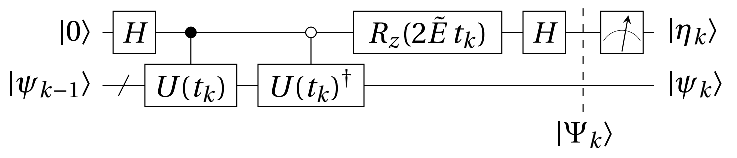
Cosine propagation

Using the established components, the proposed state preparation works as follows. First, write the initial state in terms of the Hamiltonian eigenstates

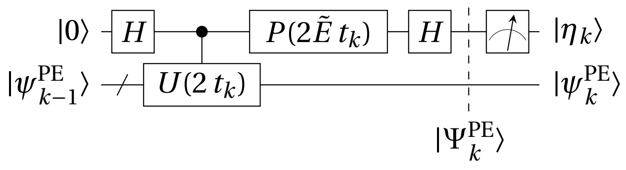
$$|\psi_0\rangle = \sum_{j=0}^{2^n-1} c_j\,|\varphi_j\rangle\,,$$

where $c_j = \langle\varphi_j|\psi_0\rangle$. After adding a single ancilla to the system and applying the circuit $\mathcal{C}$ depicted in Fig. 6.1a, right before the measurement, the state in the augmented space is

$$|\Psi_1\rangle = \sum_j c_j \left[\cos(\tilde{E}_j\,t_1)\,|\varphi_j\rangle \otimes |0\rangle + i\sin(\tilde{E}_j\,t_1)\,|\varphi_j\rangle \otimes |1\rangle\right],$$



(a) Circuit implementation $\mathcal{C}$ of the cosine-propagation using Hadamard $[H]$, Pauli-z rotation $[R_z(\theta)]$ and controlled time evolution $[U(t)]$ gates.



$|0\rangle$
$|\psi^{\mathrm{PE}}_{k-1}\rangle$
H
$U(2\,t_k)$
$P(2\tilde{E}\,t_k)$
H
$|\eta_k\rangle$
$|\psi^{\mathrm{PE}}_k\rangle$
$|\Psi^{\mathrm{PE}}_k\rangle$

(b) Single-qubit phase estimation circuit $\mathcal{C}^{\mathrm{PE}}$, using a phase shift $[P(\theta)]$ gate in addition to those mentioned above.

Figure 6.1: Functionally equivalent circuits to be used in the Hamiltonian eigenstate preparation iteration.

where the notation $\tilde{E}_j := E_j - \tilde{E}$ for the shifted energies was introduced. The probability of measuring the ancilla qubit in state $|\eta_1\rangle = |0\rangle$, is

$$P_1 = \langle\Psi_1|\Pi_0|\Psi_1\rangle = \sum_j |c_j|^2 \cos^2(\tilde{E}_j\,t_1),$$

with $\Pi_0 := \mathbb{1} \otimes |0\rangle\langle 0|$ the projector onto the ancilla-zero state. Projectively measuring the ancilla qubit and postselecting for this result, the state in the main register becomes

$$|\psi_1\rangle = \frac{1}{\sqrt{P_1}} \sum_j c_j \cos(\tilde{E}_j\,t_1)\,|\varphi_j\rangle\,.$$

After k repetitions of this procedure of applying $\mathcal{C}$ and postselecting the ancilla-zero state, using different evolution times t_ℓ, where $\ell = 1 \ldots k$, the total success probability is

$$P_k = \langle\Psi_k|\Pi_0|\Psi_k\rangle = \sum_j |c_j|^2 \prod_{\ell=1}^{k} \cos^2(\tilde{E}_j\,t_\ell) \tag{6.1}$$

and the main register state after all operations becomes

$$|\psi_k\rangle = \frac{1}{\sqrt{P_k}} \sum_j c_j \prod_{\ell=1}^{k} \cos(\tilde{E}_j\,t_\ell)\,|\varphi_j\rangle\,. \tag{6.2}$$

Single-bit quantum phase estimation

An alternative — operationally equivalent — circuit is shown in Fig. 6.1b, which is essentially a single bit quantum phase estimation circuit [145] for the eigenvalue

$E_v - \tilde{E}$. Using this circuit instead of the aforementioned cosine-propagation will yield different states

$$|\Psi_k^{\mathrm{PE}}\rangle = \frac{1}{2}\left[(\mathbb{1} + e^{-2iHt_k})\,|\psi_{k-1}\rangle \otimes |0\rangle + \left(\mathbb{1} - e^{-2iHt_k}\right)|\psi_{k-1}\rangle \otimes |1\rangle\right],$$

and, starting from $|\psi_0\rangle$, after postselecting for the ancilla-zero state k times, the main register contains

$$\begin{aligned}|\psi_k^{\mathrm{PE}}\rangle &= \frac{1}{\sqrt{P_k}}\sum_j c_j \prod_{\ell=1}^{k}\frac{1}{2}(1 + e^{-2i\tilde{E}_j t_\ell})\,|\varphi_j\rangle \\ &= \frac{1}{\sqrt{P_k}}\sum_j c_j \prod_{\ell=1}^{k} e^{-i\tilde{E}_j t_\ell}\cos(\tilde{E}_j t_\ell)\,|\varphi_j\rangle \qquad (6.3)\end{aligned}$$

with the same P_k as in Eq. (6.1). Equation (6.3) differs from the states $|\psi_k\rangle$ produced by cosine evaluation only by relative phases between states $|\varphi_j\rangle$, which should be eliminated anyway and are therefore inconsequential to the algorithm, and a physically irrelevant global phase of the target state $|\varphi_v\rangle$. The quantity of interest, $|\langle\varphi_v|\psi_k\rangle|^2$, is therefore invariant to the replacement of the circuit $\mathcal{C}$ with $\mathcal{C}^{\mathrm{PE}}$.

For consistency and simplicity, $\mathcal{C}$ will be used throughout the rest of this chapter, but note that every result and proof is either directly valid or translates straightforwardly to an equivalent result for $\mathcal{C}^{\mathrm{PE}}$.

6.1.4 Exact knowledge of E_v

If the energy of the desired state is known exactly, i.e. the uncertainty $\delta = 0$, expressions for the overlap of $|\psi_k\rangle$ with the target state and the total success probability P_k can be derived straightforwardly. In this case, $\cos(\tilde{E}_v\, t_\ell) = \cos(0) = 1$, so Eq. (6.2) becomes

$$|\psi_k\rangle = \frac{1}{\sqrt{P_k}}\left[c_v\,|\varphi_v\rangle + \sum_{j\neq v} c_j \prod_{\ell=1}^{k}\cos(\tilde{E}_j\, t_\ell)\,|\varphi_j\rangle\right]$$

with the normalisation

$$P_k = |c_v|^2 + \underbrace{\sum_{j\neq v}|c_j|^2\prod_{\ell=1}^{q}\cos^2(\tilde{E}_j\, t_\ell)}_{=:\xi_k^2} = |c_v|^2 + \xi_k^2 \qquad (6.4)$$

which is also the total success probability to measure the ancilla qubit in state $|0\rangle$ all k times. This probability is bounded from below by the overlap of the initial state with the desired target

$$P_k \geq |c_v|^2 = |\langle\varphi_v|\psi_0\rangle|^2,$$

because $\xi_k^2 \geq 0$. The overlap of the final state with the target is

$$|\langle\varphi_v|\psi_k\rangle|^2 = \frac{|c_v|^2}{P_k} = \frac{|c_v|^2}{|c_v|^2 + \xi_k^2}.$$

If the time periods t_ℓ are now resourcefully chosen such that ξ_k^2 is bounded from above, the number of iterations $\bar{k}$ needed to guarantee the desired target infidelity can be determined. The ideal sequence of t_ℓ depends on the distribution of the energies E_j and the amplitudes $|c_j|^2$ of their states within the input state. Here, a generic heuristic that suppresses every energy in the interval $[\Delta, E_{\max}]$ is described, which thus can be used even in the case of no additional information. It closely resembles a protocol the quantum phase estimation algorithm uses, albeit for slightly different reasons. However, more elaborate, tailored strategies that use additional knowledge may lead to substantially superior performance.

The longest time t_ℓ between measurements worth considering is $t = \pi/(2\Delta)$, as this will take the amplitude of the slowest oscillating state to exactly 0. The shortest reasonable time is $t = \pi/(2E_{\max})$, because this does the same for the fastest oscillating term. A universal heuristic is then to use the times

$$t_\ell = \frac{\pi}{2^{\bar{\ell}+1}\Delta} \tag{6.5}$$

where $\bar{\ell} = (\ell - 1) \bmod N$ and

$$N = \lceil \log_2(E_{\max}/\Delta) \rceil + 1. \tag{6.6}$$

To quantify the convergence to the target state, it is useful to define the maximum of the product of N cosine factors as they appear in Eq. (6.4) over all possible energies $\tilde{E}_j$ in the interval $\mathcal{I} := [\Delta, E_{\max}]$ as

$$\gamma := \max_{\tilde{E}_j \in \mathcal{I}} \prod_{\ell=1}^{N} \cos^2(\tilde{E}_j\, t_\ell) \leq 1. \tag{6.7}$$

Recall from Eq. (6.4) that ξ_k^2 is just a weighted sum of products of the same form as in Eq. (6.7) with varying $\tilde{E}_j$. Thus, after each N iterations, the magnitude of ξ_k^2 is at most a factor $\gamma < 1$ of its previous value

$$\xi_{k+N}^2 \leq \gamma \xi_k^2,$$

which, when starting from ξ_0^2, means

$$\xi_k^2 \leq \xi_0^2 \gamma^{\lfloor k/N \rfloor} = (1 - |c_v|^2)\gamma^{\lfloor k/N \rfloor}, \tag{6.8}$$

where $\xi_0^2 = 1 - |c_v|^2$, because $\sum_j |c_j|^2 = 1$.

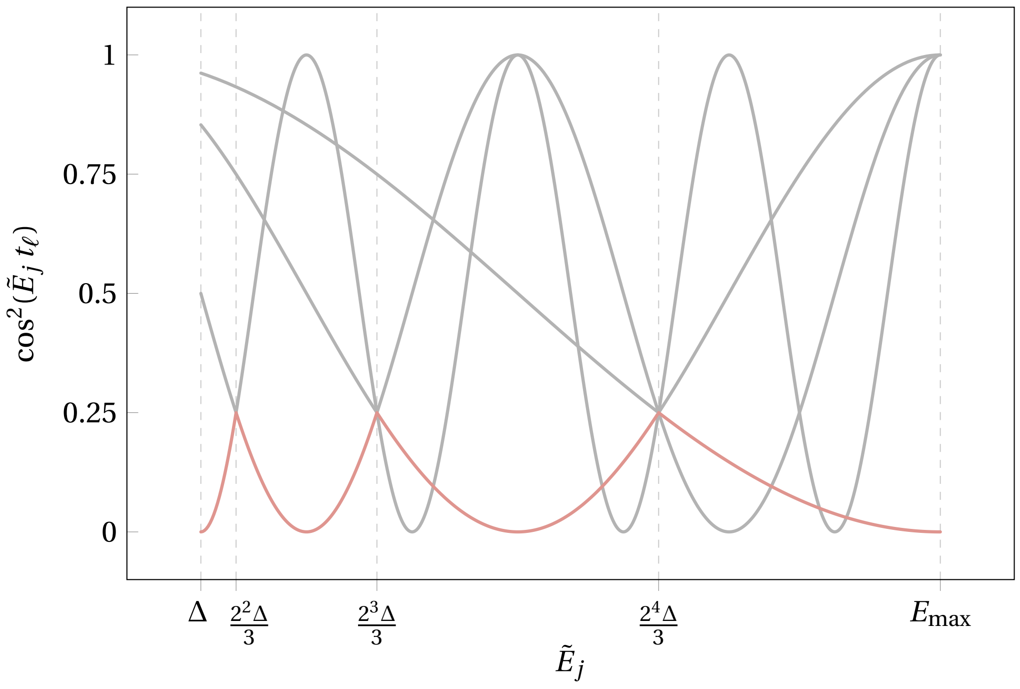


Figure 6.2: Illustration of the bound on γ, with the example value $E_{\max} = 8\Delta$. Each oscillating line plots $\cos^2(\tilde{E}_j\, t_\ell)$ for one specific t_ℓ used in the algorithm. The relevant sections of each term are coloured red. The red line stays below or at 1/4 in the entire interval, demonstrating the bound.

A quite loose but intuitive upper bound on γ can be derived as follows. Notice that when dividing $\mathcal{I}$ into sub-intervals $\mathcal{I}^{(\ell)} := [2^\ell \Delta/3, 2^{\ell+1}\Delta/3]$, with $\ell = 1 \dots N$, whichever one of these intervals $\mathcal{I}^{(\ell)}$ the energy $\tilde{E}_j$ falls into, the term $\cos^2(\tilde{E}_j\, t_\ell)$ is always smaller than or equal to 1/4,

$$\cos^2(\tilde{E}_j\, t_\ell) = \cos^2\left(\frac{\tilde{E}_j \pi}{2^\ell \Delta}\right) \leq \frac{1}{4} \quad \text{for } \tilde{E}_j \in \mathcal{I}^{(\ell)},$$

which, with a straightforward substitution of variables, is equivalent to

$$\cos^2(x) \leq \frac{1}{4} \quad \text{for } x \in \left[\frac{\pi}{3}, \frac{2\pi}{3}\right].$$

This is also illustrated in Fig. 6.2, which immediately and intuitively shows that $\gamma \leq 1/4$. The overall convergence of the algorithm is therefore

$$\begin{aligned} \left|\langle\varphi_v|\psi_k\rangle\right|^2 &\geq \frac{|c_v|^2}{|c_v|^2 + (1-|c_v|^2)\gamma^{\lfloor k/N \rfloor}} \\ &\geq \frac{|c_v|^2}{|c_v|^2 + (1-|c_v|^2)4^{-\lfloor k/N \rfloor}}. \end{aligned} \tag{6.9}$$

This means that after some initial iterations, when

$$\frac{(1-|c_v|^2)}{|c_v|^2} 4^{-\lfloor k/N \rfloor} \ll 1,$$

the fidelity converges exponentially towards 1 (i.e. the infidelity $1-|\langle\varphi_v|\psi_{\bar{k}}\rangle|^2$ decreases exponentially), because

$$\frac{1}{1+x} = 1 - x + \mathcal{O}(x^2),$$

and in this case $x \sim \exp(-k)$. The above expression for the convergence can be rearranged to calculate the maximum number of required iterations $\bar{k}$ to arrive at the desired infidelity $\varepsilon \leq 1 - \left|\langle\varphi_v|\psi_{\bar{k}}\rangle\right|^2$ as

$$\bar{k} = \left\lceil -\frac{N}{2}\log_2\left(\frac{|c_v|^2\varepsilon}{(1-\varepsilon)(1-|c_v|^2)}\right)\right\rceil \tag{6.10}$$

Implementation cost

The cost of implementing this procedure is dominated by the real-time evolution complexity of the system. Therefore, the total simulation time $\sum_\ell t_\ell$ the algorithm requires is a good proxy for the overall implementation cost. Using the sequence of t_ℓ described above, performing N iterations necessitates time-evolving the system for

$$\sum_{\ell=1}^{N} t_\ell = \sum_{\ell=1}^{N} \frac{\pi}{2^\ell \Delta} \leq \frac{\pi}{\Delta}.$$

Additionally, restarts of the preparation due to a wrong measurement outcome of the ancilla qubit must be accounted for. Fortunately, the success probability of measuring $|\eta\rangle_k = |0\rangle$ is

$$p_k := \frac{P_k}{P_{k-1}} = \frac{|c_v|^2 + \xi_k^2}{|c_v|^2 + \xi_{k-1}^2}$$

which increases quickly towards 1 — recall that ξ_k^2 is exponentially shrinking — meaning failures are most likely at the beginning of the iteration, where less cumulative simulation time has been expended.

The expected total simulation time to reach and complete the k^{th} iteration is given by sum of the simulation time to reach and pass iteration $k-1$ and the cost of one additional iteration. Additionally, this sum must be divided by p_k to account for the potentially wrong measurement outcome. This cost can be written as a recursive function

$$\mathcal{T}(k) = \begin{cases} \frac{\mathcal{T}(k-1)+t_k}{p_k} & k > 0 \\ 0 & k = 0 \end{cases}. \tag{6.11}$$

Because the probabilities p_k are often not known a priori — remember they depend on the energy distribution and the occupation of the associated states — is also useful to derive a bound for the maximum expected cost to reach the desired infidelity threshold of ε. In the most expensive case, ξ_k^2 only reduces after each N steps, and does so by the smallest possible amount of a factor of $1/4$, as given in Eq. (6.8). This minimal decrease of ξ_k^2 defers failures to later iterations in the procedure, making them more costly. The bound is then

$$\overline{\mathcal{T}}(k) = \begin{cases} \frac{\overline{\mathcal{T}}(k-N)+\frac{\pi}{\Delta}}{\bar{p}_k} & k > 0 \\ 0 & k \leq 0 \end{cases} \tag{6.12}$$

with the highest possible success probability after each N iterations

$$\bar{p}_k = \frac{|c_v|^2 + (1-|c_v|^2)4^{-\lfloor k/N \rfloor}}{|c_v|^2 + (1-|c_v|^2)4^{-\lfloor k/N \rfloor + 1}},$$

where $\overline{\mathcal{T}}(k) \geq \mathcal{T}(k)$ is the cost bound. Consequently, the total expected simulation time for preparing an eigenstate with a maximum infidelity of ε from an initial state with an overlap of $|c_v|^2$ with the target is $\overline{\mathcal{T}}(\bar{k})$, with $\bar{k}$ as in Eq. (6.10) and N as in Eq. (6.6).

6.1.5 Approximate knowledge of E_v

If the energy of the desired state is not known exactly, $\delta \neq 0$, the amplitude of the target also changes over time under cos-evolution, but much less so than all other states if $\delta \ll \Delta$, which often is a reasonable assumption. The state in the main register after k iterations therefore becomes

$$|\psi_k\rangle = \frac{1}{\sqrt{P_k}}\left[c_v \prod_{\ell=1}^{k} \cos(t_\ell\,\delta)\,|\varphi_v\rangle + \sum_{j\neq v} c_j \prod_{\ell=1}^{k} \cos(\tilde{E}_j\, t_\ell)\,|\varphi_j\rangle\right]$$

with the normalisation

$$P_k = \underbrace{|c_v|^2 \prod_{\ell=1}^{k} \cos^2(t_\ell\,\delta)}_{=:\zeta_k^2} + \xi_k^2$$

and ξ_k^2 as defined above. The assumption of $\delta \ll \Delta$ implies $t_\ell\,\delta \ll 1$ (recall that according to Eq. (6.5) $t_\ell \sim \Delta^{-1}$), which allows the expansion of ζ_k^2 into a Taylor series and truncation after the quadratic term.

$$\zeta_k^2 \approx \prod_{\ell=1}^{k}(1 - t_\ell^2\delta^2) \geq \left[\prod_{\ell=1}^{N}(1 - t_\ell^2\delta^2)\right]^{\left\lceil \frac{k}{N}\right\rceil} \approx \left[1 - \frac{\pi^2\delta^2}{\Delta^2}\underbrace{\sum_{\ell=1}^{N}\frac{1}{4^\ell}}_{\leq\frac{1}{3}}\right]^{\left\lceil \frac{k}{N}\right\rceil} \geq \left[1 - \frac{\pi^2\delta^2}{3\Delta^2}\right]^{\left\lceil \frac{k}{N}\right\rceil}$$

Similarly to before, the fidelity with the target state can be bounded as

$$\left|\langle\varphi_v|\psi_k\rangle\right|^2 = \frac{\zeta_k^2|c_v|^2}{\zeta_k^2|c_v|^2 + \xi_k^2} = \left(1 + \frac{\xi_k^2}{\zeta_k^2|c_v|^2}\right)^{-1} \tag{6.13}$$

$$= \left(1 + \frac{1-|c_v|^2}{|c_v|^2}\,4^{-\lfloor k/N\rfloor}\left(1 - \frac{\pi^2\delta^2}{3\Delta^2}\right)^{-\lceil k/N\rceil}\right)^{-1}.$$

Notice that the imprecise knowledge of the target energy only scales down the base of the exponential convergence by a factor, but does not limit the achievable fidelity. This reduced convergence rate translates to a slightly larger number of required iterations

$$\bar{k} = \left\lceil -\frac{N\log_2 \frac{|c_v|^2\varepsilon}{(1-\varepsilon)(1-|c_v|^2)}}{2 + \log_2\left(1 - \frac{\pi^2\delta^2}{3\Delta^2}\right)}\right\rceil. \tag{6.14}$$

For the bound of the expected cost, Eq. (6.12) remains valid, but $\bar{p}_k$ now takes the form

$$\bar{p}_k = \frac{|c_v|^2(1 - \frac{\pi^2\delta^2}{3\Delta^2})^{\lceil k/N\rceil} + (1-|c_v|^2)4^{-\lfloor k/N\rfloor}}{|c_v|^2(1 - \frac{\pi^2\delta^2}{3\Delta^2})^{\lceil k/N\rceil+1} + (1-|c_v|^2)4^{-\lfloor k/N\rfloor+1}},$$

which increases the total cost depending on the magnitude of δ.

6.1.6 Imperfect real-time evolution

Except in a limited number of cases, the time evolution of a system cannot be implemented exactly, but has some algorithmic error associated with it. Without loss of generality, the actually applied operator $U(t)$ can be written as

$$U(t) = e^{-iHt} + \mathcal{E}(t)e^{-i\tilde{E}t}$$

with an appropriate error operator $\mathcal{E}(t)$ to account for this finite accuracy.[40] The resulting full space state of applying the circuit $\mathcal{C}$ is then

$$|\Psi_1\rangle = \sum_j c_j \left(\left[\cos([H-\tilde{E}]\, t_1) + \mathrm{Re}(\mathcal{E}(t_1))\right] |\varphi_j\rangle \otimes |0\rangle \right. \\ \left. + i\left[\sin([H-\tilde{E}]\, t_1) + \mathrm{Im}(\mathcal{E}(t_1))\right] |\varphi_j\rangle \otimes |1\rangle\right).$$

As above, after k iterations and post-selecting for the ancilla $|\eta_k\rangle = |0\rangle$ at every step, the main register state is

$$|\psi_k\rangle = \sum_j \frac{c_j}{\sqrt{P_k}} \prod_{\ell=1}^{k} \left[\cos([H-\tilde{E}]\, t_\ell) + \mathrm{Re}\left(\mathcal{E}(t_\ell)\right)\right] |\varphi_j\rangle$$

again with the normalisation $P_k = \langle\psi_k|\psi_k\rangle$. The exact form of $\mathcal{E}$ depends on the specific RTE algorithm used; nonetheless, it is possible to establish a simple universal bound for P_k and the target state fidelity $|\langle\varphi_v|\psi_k\rangle|^2$. For this, the maximum error the RTE routine can produce[41]

$$\varepsilon_{\mathrm{RTE}} = \max_{t_\ell} \|\mathcal{E}(t_\ell)\| = \max_{t_\ell} \left\| U(t_\ell)e^{i\tilde{E}t_\ell} - e^{-i(H-\tilde{E})t_\ell} \right\|$$

is needed, with $\mathrm{Re}(\mathcal{E}) = \mathcal{E}$ being the worst case situation. Assuming $\varepsilon_{\mathrm{RTE}} \ll 1$, the state $|\psi_k\rangle$ can be expanded into powers of $\mathcal{E}$ and truncated after the first order. This yields

$$|\psi_k\rangle = \sum_j \frac{c_j}{\sqrt{P_k}} \left[\prod_{\ell=1}^{k} \cos(\tilde{E}_j\, t_\ell)\, |\varphi_j\rangle + \sum_{\ell=1}^{k} |\mathcal{E}_\ell\rangle\right] + \mathcal{O}(\mathcal{E}^2)$$

with $\langle\mathcal{E}_\ell|\mathcal{E}_\ell\rangle \le \varepsilon^2_{\mathrm{RTE}}$. The normalisation factor can be bounded by

$$P_k = \langle\psi_k|\psi_k\rangle \le |c_v|^2\zeta_k^2 + \xi_k^2 + 2k\varepsilon_{\mathrm{RTE}} + \mathcal{O}(\varepsilon^2_{\mathrm{RTE}}).$$

[40]The error operator $\mathcal{E}$ might often be time-independent, any possible dependence is included here for generality.

[41]The notation $\|\cdot\|$ means the operator norm.

Finally, the fidelity with the desired state $|\varphi_\nu\rangle$, up to order $\mathcal{O}(\varepsilon_{\mathrm{RTE}})$, has the bound

$$\begin{aligned}|\langle\varphi_\nu|\psi_k\rangle|^2 &= \frac{1}{P_k}\left|c_\nu\prod_{\ell=1}^{k}\cos(t_\ell\delta)+\sum_{\ell=1}^{k}\langle\varphi_\nu|\mathcal{E}_\ell\rangle\right|^2\\ &\geq \frac{\zeta_k^2|c_\nu|^2-2k\varepsilon_{\mathrm{RTE}}}{\zeta_k^2|c_\nu|^2+\xi_k^2+2k\varepsilon_{\mathrm{RTE}}}. \end{aligned} \tag{6.15}$$

In contrast to the case where the target energy is not exactly known, which only scaled down the convergence rate, having an imprecise real-time evolution puts a hard ceiling on the achievable fidelity. Note, however, that this bound is extremely loose, as its derivation assumed an adversarial error term. But, as is shown in the Results section, Eq. (6.15) is qualitatively accurate.

6.1.7 Gate noise

It is also worth considering the case of noisy quantum hardware. This may be a concern either because the algorithm is performed using physical qubits as the algorithmic qubits (like in the NISQ era), or because the algorithmic qubits are logical qubits, but they are of inadequate size to guarantee a negligible total error probability (a situation expected in early fault tolerant devices).

The very simple error model assumed here is that depolarising noise is applied to *every* qubit after every gate. As already discussed in Chapter 5, depolarising noise is equivalent to a certain probability of having an unwanted and undetected Pauli operator act on a qubit, i.e. for an error term on qubit k it maps the density operator ρ like

$$\rho\mapsto(1-\lambda)\rho+\sum_{\nu\in\{x,y,z\}}\frac{\lambda}{3}\sigma_k^\nu\rho\sigma_k^\nu, \tag{6.16}$$

where σ_k^ν is a Pauli-ν operator on qubit k, and λ is the error probability.

In this context, a *gate* is a Hadamard or an exponential of a Pauli string — sometimes called a *Pauli gadget* — of the form

$$\exp\left(-i\frac{\theta}{2}\bigotimes_{k=0}^{n-1}\sigma_k^{\nu_k}\right),$$

where $\nu_k\in\{\mathbb{1},x,y,z\}$ and $\sigma_k^{\nu_k}$ is again a Pauli operator on qubit k. As already mentioned in other chapters, these Pauli gadgets occur naturally when using Trotter formulas for time evolution.

No rigorous bounds are derived in the present work for this case, but the following rough first-order approximation of the achievable target state fidelity with a given error rate can nevertheless be useful in many cases. For this estimate, assume a Hamiltonian containing a large number of terms, such that the influence of all gates except the Pauli gadgets can be neglected. As discussed above, performing N iterations with the times t_ℓ results in a total simulation time of π/Δ. Assuming N_{Trott} Trotter steps per unit time are required for the desired algorithmic accuracy, carrying out N iterations requires

$$N_{\text{Pauli}} = \frac{\pi L N_{\text{Trott}}}{\Delta}$$

Pauli gadgets to implement, where L is the number of terms in the Hamiltonian. Because errors introduced in earlier iterations are largely suppressed by the measurements in later ones, the majority of the infidelity will be caused by the last N_{Pauli} gates. If each gadget introduces an error with probability λ, the total expected fidelity of the produced density operator ρ with the desired state $|\varphi_v\rangle$ can then be approximated by

$$\langle\varphi_v|\,\rho\,|\varphi_v\rangle \approx (1-\lambda)^{N_{\text{Pauli}}}. \tag{6.17}$$

Note that this estimate does not account for the use of (in general exponentially costly [269, 270]) quantum error mitigation [260] which can suppress the impact of errors, typically through the use of additional repetitions, increasing the time cost.

6.1.8 Morphing Hamiltonian

Lastly, I will explore the possibility that in some cases, if the overlap of the initial state with the target is small, the total cost of the preparation can be decreased by introducing an artificial Hamiltonian

$$H_{\text{morph}}(\alpha) = (1-\alpha) H \text{init} + \alpha H,$$

with H the target Hamiltonian as before, and an artificial Hamiltonian H_{init}, which has the initial state $|\psi_0\rangle$ as an eigenstate. Consequently, $\tilde{E}$, δ, Δ, and $E_{\max}$ all become functions of α.

This morphing Hamiltonian can then be used with a number of values $\alpha_n \in [0,1]$, where at each value α_n only N timesteps are performed before moving on to α_{n+1}. At the final value of $\alpha = 1$, the full preparation is performed to the desired accuracy.

This process somewhat resembles a combination of adiabatic evolution with the quantum Zeno effect, because the state is dragged along close to the desired state by changing the Hamiltonian, while simultaneously repeatedly measuring its phase change [268]. The numerical investigation performed for this chapter considers the coarse grained limit of this procedure with only a single intermediate value of $0 < \alpha < 1$, and demonstrates its efficacy but also limitations in the next section.

6.2 Results

To demonstrate the proposed algorithm numerically using the example of LiH, its Hamiltonian in second quantisation was generated using OpenFermion [218], and its dynamics were simulated using exact quantum emulation software [192]. The system consists of 12 qubits, has a spectral gap[42] of $\Delta \approx 0.075$ and a maximum energy of $E_{\max} \approx 9.753$, resulting in $N = 9$ different times t_ℓ. Starting from an initial state

$$|\psi_0\rangle = \frac{1}{\sqrt{5}} |\varphi_0\rangle + \frac{1}{N_\mathrm{s}} \sum_{j=1}^{2^n-1} e^{-(E_j - E_0)} |\varphi_j\rangle, \tag{6.18}$$

with an appropriate normalisation factor N_s, the routine to amplify the ground state $|\varphi_0\rangle$ was executed under each discussed limitation. The results of the numerical simulations together with the established bounds are shown in Fig. 6.3.

Exact E_v The calculations using perfect knowledge of the target state energy show the expected behaviour of overall exponential convergence after some initial iterations. The step-like structure of the numerical simulation is caused by the periodic choice of simulation times. Each of those steps corresponds to a full sequence of N different times t_ℓ. Due to the rounding in the expressions for the bounds in Eqs. (6.9), (6.13) and (6.15), these also show such step-like behaviour. However, for readability, only every Nth data point is plotted for them, i.e. the bottom left corner of each step.

[42]Hartree atomic units are used throughout this Results section.

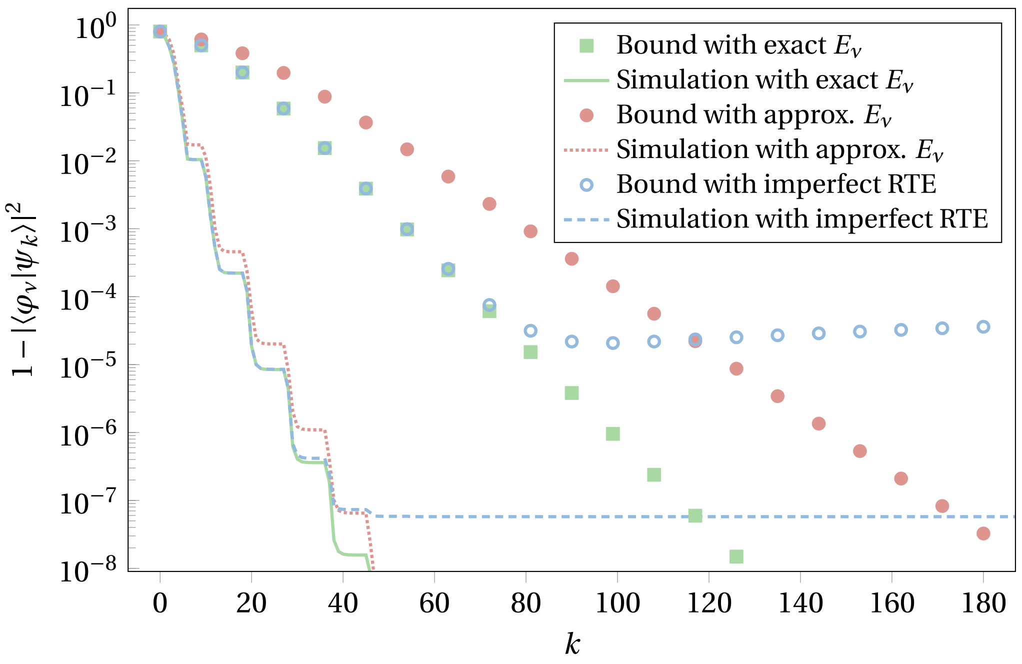


Figure 6.3: Infidelity of the produced state $|\psi_k\rangle$ in the main register with the target state $|\varphi_0\rangle$ versus the iteration number k. Lines ─── ······ - - - - are numerical results of preparing the ground state from the initial state in Eq. (6.18), markers ■●○ are the corresponding bounds derived earlier. Different colours represent different limitations; green ■ ─── for exact knowledge of the target state energy and perfect real-time evolution, red ● ······ for only approximate knowledge of the target energy but perfect RTE, and blue ○ - - - - for exact target state energy knowledge but imperfect RTE. Note that the bound and simulation of the imperfect RTE data use different errors ε_{RTE}, as described in the text.

Approximate E_v To show the effect of only knowing the energy of the target state approximately, a relatively large offset of $\delta = \Delta/3$ was used. The graph of the bound nicely illustrates the scaling down of the basis of the exponential convergence, i.e. a shallowing of the slope. The numerical simulation also shows a slightly reduced rate of convergence compared to the case of exactly known energy. Note, though, that this is not always necessarily the case. Depending on the specific distribution of energy levels and their occupation in the initial state, either one may converge faster than the other. However, the *guaranteed* convergence is always quicker the more precisely the energy of the target state is known.

Imperfect RTE I also performed a calculation of the same system with precisely known target state energy ($\delta = 0$), but using a first-order product formula [111, 112] as the RTE routine, dividing the shortest time interval into 128 slices. Importantly, in the plot the bound ∘ and the numerical simulation ---- do not use the same $\varepsilon_{\mathrm{RTE}}$, because the bound is very loose. The simulation has the numerically obtained value of $\varepsilon_{\mathrm{RTE}} \approx 5.6 \cdot 10^{-4}$, while the bound uses the much smaller $\varepsilon_{\mathrm{RTE}} = 10^{-8}$. Therefore, the simulation and the bound are only qualitatively related. The plot still shows the same pattern emerging in both cases. There is a close match between the simulation of the perfect real-time evolution and the imperfect RTE solution, until some lower threshold of the infidelity is reached, after which the imperfect RTE version becomes roughly constant,[43] and no further progress can be made. It is therefore evident that while imprecise energy knowledge only slows down the convergence of the algorithm, the presence of simulation errors puts a hard lower limit on the attainable infidelity.

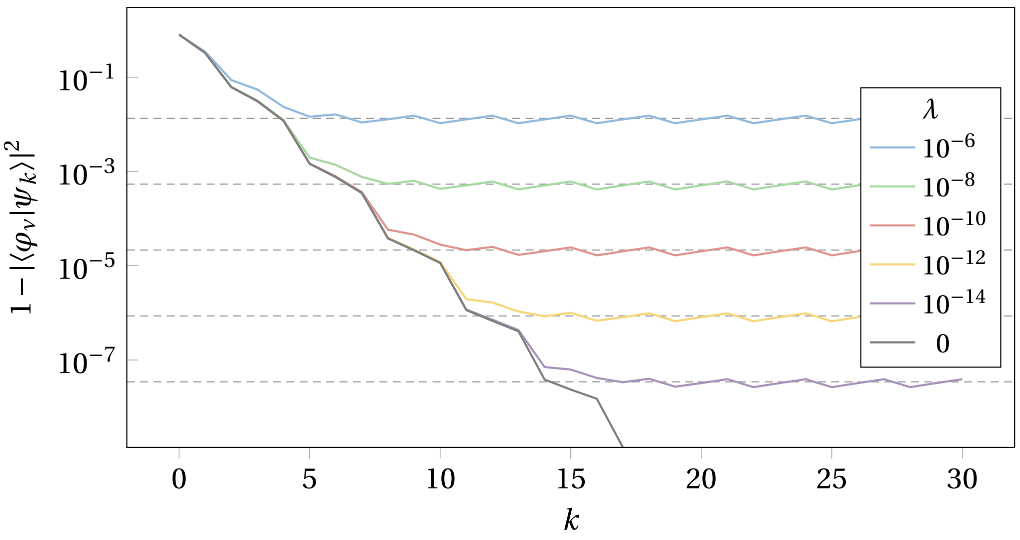


Figure 6.4: Infidelity of the prepared density operator ρ_k with the desired target state $|\varphi_v\rangle$ versus the iteration number k for different error rates λ. Coloured lines are numerical results, grey dashed lines show the expected approximate limit according to Eq. (6.17).

Gate noise In order to demonstrate the noise resilience of the discussed method, eigenstate preparation was also performed using first order Trotterisation and different noise strengths λ. Due to the increased computational demand of using the density matrix formalism to include noisy channels, Fig. 6.4 shows the results for the second quantised Hamiltonian of H_2, a much smaller system than LiH. It exhibits the same limiting pattern as in the case of algorithmic errors, where the state quickly converges

[43]The bound even slightly increases due to some of the approximations made in its derivation.

to the desired target, but then encounters a ceiling in the fidelity caused by the errors. The numerical results show good agreement with the approximate maximum fidelity derived earlier in Eq. (6.17).

Notice that while a rather small error rate λ is needed in order to obtain good results, suitable quantum error mitigation (QEM) techniques can boost performance in return for additional simulation cost. An example is symmetry verification. These options are not explored further in this investigation, because it is quite a broad topic. Additionally, which QEM is best suited for a given task will depend on several of its properties and the hardware platform used. Reference [260] gives a good overview of such techniques.

Morphing Hamiltonian To demonstrate how morphing the Hamiltonian can sometimes decrease the total cost, once again consider the LiH Hamiltonian. As initial state the computational basis state, $|\psi_0\rangle = |10000100011\rangle$ was selected,[44] which has an overlap with the ground state of $|\langle\psi_0|\varphi_0\rangle|^2 \approx 4.6 \cdot 10^{-3}$. The corresponding initial Hamiltonian used here is

$$H_{\text{init}} = \Delta \sum_{j=0}^{n-1} (2\psi_0^{(j)} - 1)\sigma_j^z$$

where $\psi_0^{(j)} \in \{0,1\}$ refers to the value of qubit j in the computational basis state $|\psi_0\rangle$. This choice guarantees that $|\psi_0\rangle$ is the gapped ground state of H_{init}, with a gap of Δ, matching that of H.

Note that the cost to implement the time evolution of $H(\alpha)$ will most likely be a function of α. The exact form of this dependence will vary with the simulation technique, though the complexity of simulating $H + H_{\text{init}}$ may be used as a cost bound for most methods. Such a dependence is not explicitly addressed in the present analysis, and only the total required simulation time in the system is reported, regardless of the value of α.

For the purposes of a first exploration, this work only considers one intermediate step between $\alpha = 0$ and $\alpha = 1$. The question of what this intermediate value should ideally be, turns out to be quite complex. Figure 6.5 shows the total required simulation

[44] Contrary to the earlier description, this is not the computational basis state with the highest possible overlap with $|\varphi_0\rangle$, but it is better suited as an instructive example.

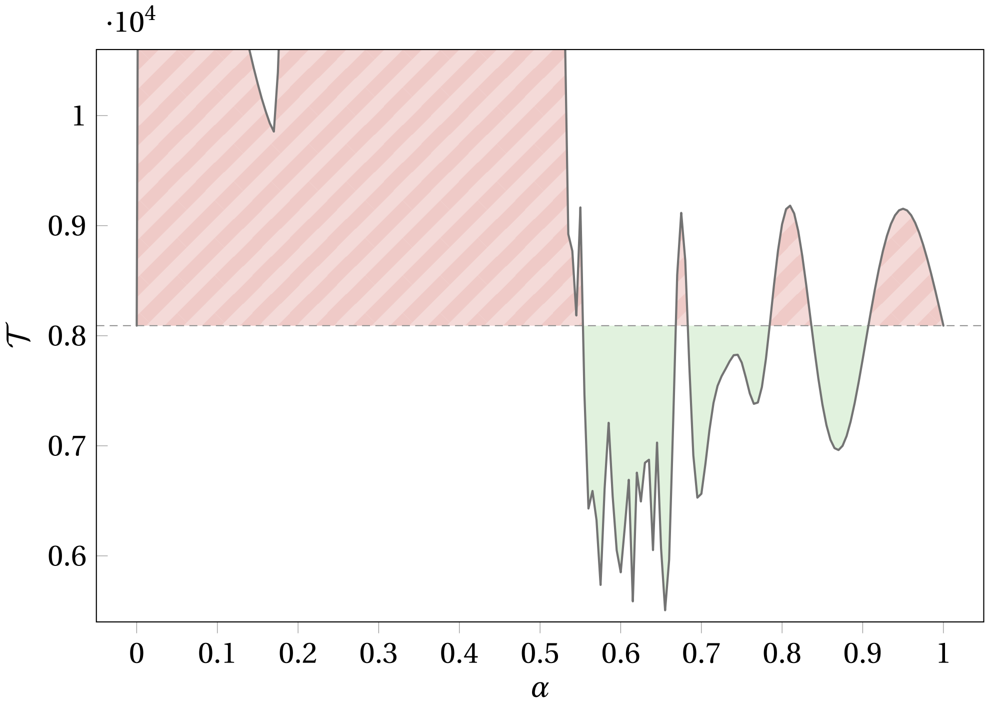


Figure 6.5: Total simulation time cost $\mathcal{T}$ when using a single intermediate value of α between H_{init} and H, depending on the choice of α. The cost is the mean total required evolution time and includes restarts after failed ancilla measurements. The solid line —— represents the morphing Hamiltonian; for reference, the grey dashed line - - - - is the cost of directly preparing the ground state of H without using H_{morph}. In the red striped region ▨, a morphing Hamiltonian as described above increases the cost, while the in the green region 🟩 the morphing method is computationally cheaper.

time to reach an infidelity of 10^{-8}, depending on where the intermediate α is placed. Green shaded regions where the graph is below the dashed line indicate values where the morphing approach is advantageous. The potentially complex behaviour of the preparation cost is exemplified in the region around $\alpha \approx 0.6$. I have identified that the rapidly oscillating character is related to the low-lying energy spectrum of $H(\alpha)$ in that area, which consists of a gap with multiple closely spaced excited states right above it.

When considering the simple case of a single α and a cheaply simulated Hamiltonian, the task of finding a near-optimal value is easily solved numerically. However, the general case of multiple intermediate values α_k remains difficult due to the rapidly increasing size of the configuration space and the non-convexity of the cost. This question is left open for further research.

6.3 Discussion

In this chapter I investigated the repeated use of a circuit closely resembling that of iterative phase estimation [146, 266] and the Rodeo algorithm [175, 267] in order to prepare eigenstates of a Hamiltonian system from arbitrary initial states. The required knowledge is the (approximate) energy of the target state, a lower bound of the energy gap of the target state to the closest lying occupied state, and an upper bound of the largest energy difference between the target and any other occupied state. The necessary tools to implement the presented algorithm are single-qubit gates on one ancilla qubit, as well as controlled real-time evolution (RTE) of the system. These elements are also typically part of related methods such as the Rodeo algorithm and various phase estimation variants.

Analytic bounds are available for the fidelity of the produced state with the target and the expected total required RTE duration for different cases. Imprecise knowledge of the target state energy results in a slower convergence rate, but does not limit the achievable fidelity. Algorithmic and gate noise, on the other hand, hardly influence the rate of convergence, but put a ceiling on how precisely the target state can be prepared. In all cases, asymptotically exponential convergence of the fidelity with the real-time evolution time is guaranteed.

The derived explicit expressions for strict bounds are, in contrast to the Rodeo algorithm, certain to be achieved. However, these bounds were anticipated to be quite loose versus a specific implementation; this indeed turned out to be the case. Still, such analytic expressions can be valuable when the method is used as a subroutine of a more substantial algorithm and its implementation cost must be bounded. They may also be useful when a certain fidelity must be guaranteed after some number of iterations without expending resources on verification.

Because of the many possible energy structures of interesting Hamiltonians, this chapter only describes a universal heuristic for the time schedule. However, using the intuitive insights about how the amplitude of unwanted states is suppressed, as described in Section 6.1, the performance may be vastly improved by using additional information about the initial state and the distribution of occupied energy levels. The highly occupied states can be targeted directly — individually or in groups — to eliminate them completely. In the extreme case of exactly knowing all N occupied energy levels, only at most N iterations need to be carried out to leave purely the target state behind. If N happens to be a tractably small number, this can be an important feature. But even in cases where N is exponentially large, if most states are concentrated in a small window of energies, they may be specifically targeted for faster convergence.

The actual gate- and/or query complexity is determined by the chosen method for the controlled time evolution. For example, Hamiltonian simulation by quantum signal processing [138] only requires $\mathcal{O}(t - \log \varepsilon_{\text{RTE}})$ gates to implement the required real-time evolution, making the actual cost of the state preparation using the generic heuristic logarithmic in the desired infidelity.

Finally, a variation of the preparation process, where the Hamiltonian is morphed from a trivial one to the Hamiltonian of the system of interest, showed some intriguing features. Numerical evidence — using the LiH system again — demonstrates that for some choices of morphing schedule, this process can decrease the cost of preparation. However, care must be taken, as unfavourable choices may easily increase the total cost. Finding a generic method to generate an efficient schedule might be an interesting topic for future research.

If in physics there's something you don't understand, you can always hide behind the uncharted depths of nature. You can always blame God. You didn't make it so complex yourself. But if your program doesn't work, there is no one to hide behind. You cannot hide behind an obstinate nature. If it doesn't work, you've messed up.

— Edsger Dijkstra

7 PYQUEST

This chapter describes an open source software package developed during my doctorate. All work below is my own.

Contents

Almost all of the numerical data on quantum algorithms in the research chapters in this thesis has not come from actual quantum hardware, but rather from software emulating a quantum computer using classical computing resources. This chapter will elaborate on *pyQuEST*, a tool making the high-performance quantum circuit simulator *QuEST* available to researchers who use Python as their main programming language. I have developed it during my doctorate for myself, my research group, and the community, in order to ease access and usability of QuEST and speed up development- and calculation times of quantum algorithms.

7.1 Introduction

Numerical methods have a long history in physics, and especially quantum physics, where they provide an option for solving or approximating solutions to problems which do not allow analytical treatment. Because of their significance, algorithms of all sorts related to the numerical solution of physics problems have received a great deal of attention, and many ingenious ideas have steadily increased their performance.

In recent years, they have become invaluable tools to the development of quantum algorithms, where they allow us to peek at what quantum hardware might soon be able to do — whether NISQ devices, early fault tolerant hardware, or fully error corrected platforms — albeit at a very limited scale.

This chapter is structured as follows. First, Section 7.2 gives more background on the motivation of the project and why it is a useful piece of software for the community. Section 7.3 elaborates on some mechanisms which are available for interfacing Python code with C libraries, and explains the reasoning behind the choice that was made. Section 7.4 focuses on the way how users install the code and the programming interface a user would interact with when using the package, while Section 7.5 contains a quite technical description of performance considerations that needed to be made during development. Finally, Section 7.6, gives a brief summary of the chapter and hints at further plans.

7.2 Necessity and utility

As mentioned in Section 2.5, there is a whole zoo of quantum emulators available for use today, in a wide variety of programming languages. One particular emulator I want to point out again is the *Quantum Exact Simulation Toolkit* (QuEST) [187], which is a high performance full state vector and density matrix emulator that can be easily deployed on CPUs, GPUs, and even distributed high performance computing (HPC) clusters. It is provided as a C/C++ library and works through a very straightforward application programming interface (API) of functions to, among other things, create and destroy quantum registers (state vectors as well as density matrices), initialise them, apply a great variety of gates, operators, and measurements to them, and read their contents if desired (the latter being strictly forbidden on real quantum hardware). The choice of language for this emulator was largely influenced by performance considerations. Using C provides a very “bare metal” way to implement efficient algorithms and create a high throughput emulator.

The downside of this choice is ease of use. QuEST itself provides very minimal convenience functions and data structures. For example, it does not have any provisions for quantum circuits, which are otherwise ubiquitous in quantum computing. With the expectation of users also writing their code in C, all other limitations of the language also naturally apply. This means no symbolic calculations, manually managing one’s own memory, and generally less convenience (and thus slower development time) than many other, more modern languages. It is still a great choice when every bit of performance matters, but often it is wise to accept some small amount of overhead for a speedy development process.

One project embracing this philosophy is named QuESTlink and has been released by Jones and Benjamin [193]. It provides a Mathematica interface to the CPU and GPU versions of QuEST, offering many convenience functions to make the development of quantum algorithms quicker and easier. Additionally, users can benefit from the vast capabilities of Mathematica for symbolic calculations and advanced analytical and numerical tools. However, Mathematica is not the most popular tool for investigating quantum computing problems, and many researchers are better versed in other programming languages like Python. In my doctorate, I have therefore decided to implement an easy to use Python interface for QuEST, named pyQuEST.

As Section 2.5 points out, there is no shortage of quantum emulators readily available for Python [176, 178, 179, 181, 183, 184]. Yet, hardly any have the performance and flexibility of QuEST, which, as mentioned above, simulates state vectors and density matrices, and easily scales from running locally on a single or multiple CPU threads, to GPUs, and even distributed instances on many compute nodes in a cluster. There even already exists a rudimentary Python interface to QuEST, named *pyQuEST-cffi*, which — as the name suggests — uses the *Foreign Function Interface for C* (CFFI) to make QuEST functions callable from Python. However, because it directly exposes the C API functions from QuEST to the user, it is quite unintuitive and atypical to use for someone familiar with (only) Python. It simply aims at *directly* working with QuEST from Python, and therefore has the bare minimum of features necessary, without any added convenience. I therefore still see it as a valuable contribution to the community to provide pyQuEST as a simple, easy to use, high performance interface to the capabilities of QuEST.

7.3 Interface choice

The core project of QuEST on top of which pyQuEST builds is focused on simulating quantum circuits as fast as possible on a given hardware. As such, any interface layer to it should adhere to the same principles, as not to waste valuable compute time at the link between user code and circuit execution at the backend. For Python, several options exist.

The most basic and straightforward way to interact with C code from Python is through *ctypes*, which is part of the Python standard library [271]. It provides a number of C-compatible data types, can dynamically load shared libraries, and call functions from them. However, it is at times quite cumbersome to use, error prone, and provides wrapping functionality only. This means that either the user needs to interact with possibly unintuitive function calls and interfaces, or there must be additional interface and convenience code written in pure Python, which might impose quite significant performance penalties.

Similarly, the aforementioned *CFFI*, upon which PyQuEST-cffi is built, provides only function-binding capabilities by directly exposing C functions to the Python user. Although it is generally somewhat easier to use, it has the same problems as ctypes, i.e. having the user call the backend functions as-is sometimes leads to unexpectedly convoluted code. As for ctypes, convenience functions on top of QuEST would have to be implemented in pure Python in this case as well.

Some more advanced solutions also exist, for example *pybind11*, *Boost.Python*, and *SWIG*. The lean pybind11 library is a promising candidate for an efficient wrapper, but all additional functionality would have to be implemented in C++, somewhat slowing down the development process. Boost.Python has — as far as is relevant to the currently discussed application — almost the same functionality as pybind11, but is in itself a large dependency. Given QuEST's no-dependency philosophy, Boost.Python does not seem like a good choice. SWIG is quite similar in functionality with the same drawbacks, but has the potential benefit of being able to also work with languages other than Python, should the demand for it arise.

In the end, the choice was made to use *Cython*, a programming language very similar to Python, specifically designed to enable rapid development of high-performance extension modules for CPython.[45] It is unique in that its code is almost fully compatible to pure Python, but has options and features, such as static typing, to vastly improve the performance. It can also directly link to external libraries (like QuEST) and call functions from them. This allows for interaction with pure C code, as well as rapid development of additional convenience features and data structures in a high level language similar Python, while keeping almost all of the performance of a low level language like C. For this project, it seems to be an ideal choice.

Cython code must be compiled in order to be imported into Python. This compilation works in two steps. First, the Cython source is translated down into pure C code. Then, this produced C code is compiled to a binary extension module and linked with any other libraries and dependencies. The compiled module can then straightforwardly imported into any CPython interpreter and behaves mostly like any other module.

7.4 Usage

7.4.1 Interface design

As mentioned in Section 7.2, a major point of focus for this project was the simplicity and intuitiveness of the interface the user interacts with. Without going into too much technical detail, I will describe the main points of its interface in this section.

[45]CPython is the default interpreter and "reference implementation" of the Python language. It is the most popular, but not the only available interpreter.

The package design tries to replicate that of QuEST relatively closely. First, it is a quite logical way to organise operators and data structures. Second, it makes switching from QuEST to pyQuEST and vice versa much easier, in case users are already familiar with one or the other and would like to switch. As such, it contains the modules `core` (central data structures and abstract base classes), `unitaries` (unitary gates), `gates` (non-unitary but norm preserving operators, i.e. measurements), `operators` (generic, potentially non-physical operators, and operators that are usually not considered single *gates*, like quantum Fourier transforms), `initialisations` (operators to populate a register with a pre-defined state), and `decoherence` (specific and generic quantum channels).

The pyQuEST package is centred around the primary classes `Register` and `Circuit` in the module `core`, which are also available at the root level of the module, i.e. `pyquest.Register` and `pyquest.Circuit`.

As the name suggests, a `Register` contains a quantum register of a fixed number of qubits, either representing a state vector or a density matrix, both of which must be supplied at object creation. An analogue to it exists in QuEST in the form of `Qureg`, fulfilling the same role. Once instantiated, it occupies a fixed space in memory, until it is destroyed or garbage collected, at which point the memory is automatically freed. At the backend, all memory is managed by QuEST. The contents can be accessed via regular indexing, as if the register was a 1D (for state vectors) or a 2D array (for density matrices), e.g. for a state vector register named `reg`, the indexing `reg[0:1]` returns the first 2 amplitudes belonging to the computational basis states $|0\dots00\rangle$ and $|0\dots01\rangle$. Amplitudes can also be set using this indexing, making it easy to manipulate the them manually.[46] The `Register` class also wraps many convenience functions of QuEST, like getting the total probability of a (non-physical) state, the addition of two registers, multiplication by scalars, inner products with other registers, etc. Most importantly, quantum operators can be applied to it via the `apply_operator` method. These operators can be any of those found in the various modules like `unitaries` and `operators`, or `Circuits` which are discussed momentarily.

[46]This operation is comparatively expensive; setting a full state through the operators in the `initialisations` module is generally faster.

An important concept not found in QuEST is that of a quantum circuit. Because it is such a central object in discussions of quantum algorithms, pyQuEST fills this gap with the `Cirucit` class. It behaves much like a `list` in Python, but can only contain operators supplied by pyQuEST. It is, itself, an operator, thus circuit nesting is possible straightforwardly. Applying a circuit to a state typically involves many iterations over gates, and iterating in Python is often slow, so as much of it as possible is done in pure Cython without interacting with the CPython runtime. Some more details on this are given in Section 7.5.1.

7.4.2 Deployment

The pyQuEST package has been around for some time, but, as a research tool rather than a dedicated project, is still in a relatively early stage from a software development point of view. As such, it has not yet made it to easy and straightforward availability on the Python Package Index (PyPI). At the current stage, users must clone the repository from GitHub and compile it themselves. This is usually not an issue, but does require a functioning compiler chain on the target system, as QuEST would.

To be more user-friendly, the ultimate goal is to provide compiled binaries of CPU and GPU configurations for all popular operating systems (Linux, macOS, Windows). Any niche use cases (unusual graphics cards, distributed computing, etc.) would still require manual compilation.

7.5 Performance

This section contains a concise discussion of some of the most important points to consider when implementing a wrapper such as pyQuEST. I will also show the difference in runtime when executing circuits in QuEST and in pyQuEST to demonstrate that the overhead is not always negligible, but in most cases small enough to justify the use of a higher level language with faster development times.

7.5.1 Technical considerations

Some care must be taken when using Cython for the development of (supposed) high-performance extensions. A full discussion of all pitfalls and nuances would be outside the scope of this thesis, but I will highlight some general principles, as well as a few specific implementation details that seem important for pyQuEST. Overarchingly, though, it is important to keep in mind that some discipline is required to not fall victim to excessive premature optimisation. Features should be implemented straightforwardly (keeping to the below principles), and if they turn out to be the bottleneck in some use case, only then should they be optimised.

In general almost everything statically typed is much faster than almost anything dynamically typed in Cython. This is mostly because statically typed variables will be directly translated to their C counterpart and run at native speed, while dynamic variables are represented as a pointer to a generic Python type, which must be dereferenced, type checked, reference counted, etc. This also goes for function parameters and return types. Therefore, all calls between functions within the pyQuEST module use typed function signatures. Furthermore, interactions with the CPython runtime are generally quite slow and should be kept to a minimum.

Some operators of QuEST require storage of data in dedicated structures, e.g. the generic unitary operator `Unitary` requires its matrix elements to be supplied via a QuEST-specific matrix of complex values. pyQuEST creates this matrix automatically from almost any supplied Python data structure that can be accessed with 2D indexing, but conversion from Python to QuEST might be slow. Therefore, once created, the `Unitary` object keeps the QuEST data structure stored until it is destroyed. This saves the cost of repeatedly converting the data every time the operator is applied to a register. This philosophy is adhered to throughout the project.

7.5.2 Benchmarking

As the project of pyQuEST itself is only a (deliberately) thin interface layer on top of QuEST, I will not compare the absolute performance of simulations run in pyQuEST to other quantum emulators available for Python. Even though that would be a

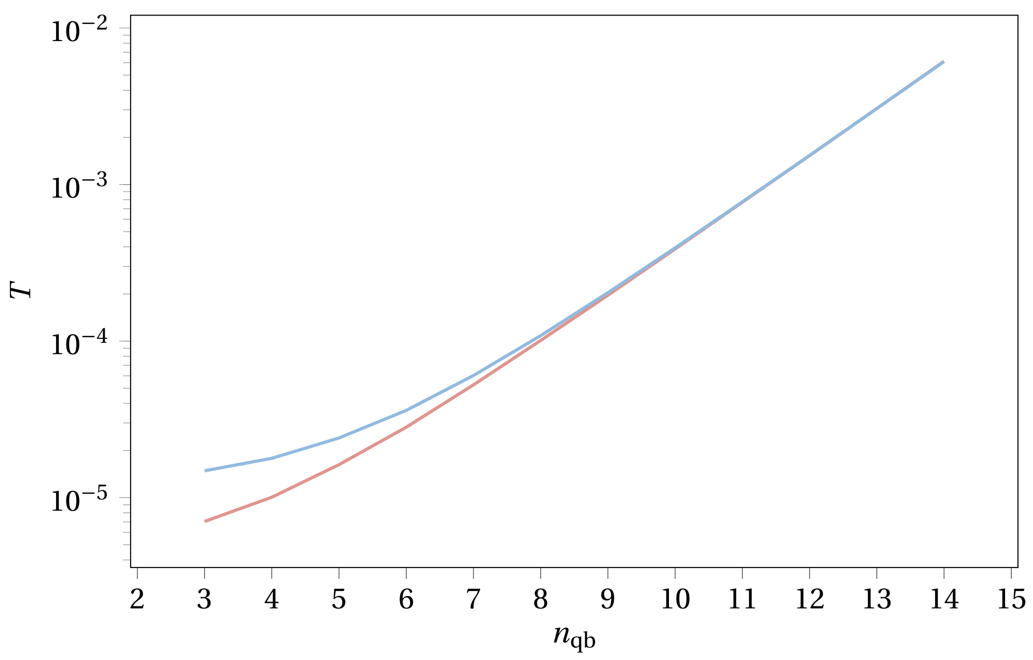


Figure 7.1: Average runtime to execute a random circuit containing 100 gates depending on the number of qubits, using QuEST (red) and pyQuEST (blue).

valuable and interesting task, it is outside the scope of this thesis, as it would put focus on the performance of the backend, distracting from the interface. Instead, I will compare timings for executing random (but identical) circuits in QuEST and pyQuEST on a single CPU core to show that the Python wrapper does in most cases not significantly impact performance, and that pyQuEST users get execution times that are often indistinguishable from QuEST itself. Since multithreaded, GPU, and HPC applications only begin to be viable when the execution time of a single gate (and thus a function call) becomes macroscopic, meaning the call overhead is negligible, these are excluded from this benchmark.

Figure 7.1 shows the obtained timings to execute a circuit containing 100 (random) gates per full circuit applied. As expected, the overhead is most noticeable for very small qubit counts, as this is when the backend calculation finishes most quickly, and the time to call the function in the first place becomes a significant part of the total execution time. At three qubits, the full 100 gate circuit takes roughly 15 µs to execute using pyQuEST, and only 7 µs using pure QuEST, marking more than double the time for pyQuEST. This phenomenon is an inherent problem of interpreted languages like

Python. Once the backend has control, it can execute as fast as its implementation permits, but handing over that control takes some (roughly constant) amount of time. Fortunately, as the execution time in the QuEST backend increases for more qubits, the relative proportion of the call overhead in the total time shrinks exponentially, which Fig. 7.1 clearly shows, until it practically disappears around the 10 qubit mark.

As this call overhead is such an innate problem of Python, there are only few ways around it. Surprisingly, Cython offers a remedy for this. If between circuit executions, some simple calculations must be performed, these can also be written in Cython and compiled together with the rest of the wrapper. Consequently, control flow never goes back to Python during the whole procedure, avoiding any overhead. Since both the user- and wrapper code are compiled to C and then machine code, they can interact without Python intervention. This presents a good alternative when performance is of high importance, but only a small section of code presents the bottleneck of the calculation.

7.6 Discussion

This chapter presented a valuable tool named pyQuEST, well suited for developing quantum algorithms in the era where capable quantum hardware is not readily available. The Python wrapper around the high-performance emulator QuEST provides some convenience features, which, alongside the natural capabilities of Python, make development quicker and easier than writing code in C.

Some of the considerations that have gone into its design and implementation were discussed, which can serve as a reference to readers who consider extending its functionality in the future. Comprehensive documentation will be available once the project has matured to its first official release.

Although many quantum simulators are already available for Python and other platforms, there is value in providing diversity to the community, because different tools prioritise different aspects of development. As already discussed in the Literature Review, many software packages specifically target deployment to actual (noisy) quantum hardware, and therefore do not put too much emphasis on fast simulation capabilities. In cases where large-scale and/or noise-free calculations are necessary, pyQuEST can therefore be an important instrument.

Throughout the work discussed in the other chapters of this thesis, I have extensively used pyQuEST myself, and hope that it will bring as much utility to the broader quantum research community as it has brought to me.

Da steh' ich nun, ich armer Tor, und bin so klug als wie zuvor!

— Johann Wolfgang von Goethe

8 Conclusion

Taking an optimistic stance, it seems possible that the field of quantum computing could stand on the brink of becoming practically useful, and may outperform classical computers in the not-too-distant future. However, in addition to the rapid development of quantum hardware that is taking place right now, further theoretical advances will be necessary to efficiently harness their power. This thesis covered a series of such theoretical advances, with each of them contributing an improvement to a different aspect of relevant calculations and applicable to distinct eras of quantum computing.

Chapter 3 explored a collection of ideas aimed at automatic generation of quantum circuits, the "assembly code" of quantum computers, sitting right above the physical instructions to laboratory equipment like lasers and coils. While for large-scale applications, clearly some more improvements are needed, the discussed method can synthesise circuits quite reliably in the small- to intermediate scale, making it most relevant in the NISQ era.

The discussion in Chapter 4 was focused around the specific application of quantum computers to solve chemistry problems, more specifically that of electronic motion around nuclei. Two distinct methods were discussed. The first is based on the elaborate technique of *linear combinations of unitaries*, and showed that, at the same cost, the accuracy of simulations can be improved by about one order of magnitude by slightly changing the well-known truncated Taylor series technique. Due to its resource requirements, this method is most applicable to fully error-corrected platforms. The second explored an approach around first quantised simulation of quantum systems, a technique that is less popular in the literature than second quantisation, but which is nonetheless expected to scale well to larger systems. Significant classical compute power was deployed to give an accurate assessment of which quantum resources would be required to solve quantum chemistry problems of substantial size. Several hundred qubits could already be enough to treat interesting problems, making it a candidate for early fault-tolerant applications.

In Chapter 5, a combination of the technique of classical shadows and sophisticated post-processing was used to recover Hamiltonian spectra from very little and noisy measurement data, making it applicable in the early fault-tolerant and even the NISQ era. The calculation of such spectra is important for characterising materials, obtaining reaction rates, and a wide variety of other system properties.

For cases where the energy levels in a system are already known — possibly from one of the methods mentioned above — Chapter 6 provides a frugal scheme to prepare one of its eigenstates. This is important if some properties of interest are not directly reflected by the Hamiltonian and thus the energy. Examples are the magnetisation in the absence of an external field, or certain spatial symmetries. After preparing the eigenstate of interest, such characteristics can be examined directly via measurements. Early fault-tolerant and fully error corrected devices seem most appropriate in this case.

Finally, Chapter 7 presented a tool to conveniently interact with a powerful classical simulator of quantum computers, which may be most useful in the era where large-scale quantum hardware is not readily available. However, as is the case in much of software development today, emulation may stay relevant for small-scale prototyping of algorithms before deploying significant quantum resources, even after such resources become available.

Together, the approaches presented in this thesis advance the field of quantum computing ever so slightly towards practically implementable algorithms that will hopefully yield invaluable insights in physics and other fields of science.

BIBLIOGRAPHY

[1] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, June 1982. DOI 10.1007/bf02650179.

[2] P. Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22(5):563–591, May 1980. DOI 10.1007/bf01011339.

[3] P. A. Benioff. Quantum mechanical Hamiltonian models of discrete processes that erase their own histories: Application to Turing machines. *International Journal of Theoretical Physics*, 21(3-4):177–201, April 1982. DOI 10.1007/bf01857725.

[4] D. Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, July 1985. DOI 10.1098/rspa.1985.0070.

[5] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, December 1992. DOI 10.1098/rspa.1992.0167.

[6] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc. Press, November 1994. DOI 10.1109/sfcs.1994.365700.

[7] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):R2493–R2496, October 1995. DOI 10.1103/physreva.52.r2493.

[8] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. ACM Press, 1996. DOI 10.1145/237814.237866.

[9] A. Y. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, January 2003. DOI 10.1016/s0003-4916(02)00018-0.

[10] E. Gibney. Inside Microsoft's quest for a topological quantum computer. *Nature*, October 2016. DOI 10.1038/nature.2016.20774.

[11] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15), October 2009. DOI 10.1103/physrevlett.103.150502.

[12] A. Peruzzo, J. McClean, P. Shadbolt, M. H. Yung, X. Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), July 2014. DOI 10.1038/ncomms5213.

[13] Y. Li and S. C. Benjamin. Efficient Variational Quantum Simulator Incorporating Active Error Minimization. *Physical Review X*, 7(2), June 2017. DOI 10.1103/physrevx.7.021050.

[14] K. Temme, S. Bravyi, and J. M. Gambetta. Error Mitigation for Short-Depth Quantum Circuits. *Physical Review Letters*, 119(18), November 2017. DOI 10.1103/physrevlett.119.180509.

[15] J. A. Jones and M. Mosca. Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer. *The Journal of Chemical Physics*, 109 (5):1648–1653, August 1998. DOI 10.1063/1.476739.

[16] J. Brooke, D. Bitko, T. F., Rosenbaum, and G. Aeppli. Quantum Annealing of a Disordered Magnet. *Science*, 284(5415):779–781, April 1999. DOI 10.1126/science.284.5415.779.

[17] Y. Nakamura, Y. A. Pashkin, and J. S. Tsai. Coherent control of macroscopic quantum states in a single-Cooper-pair box. *Nature*, 398(6730):786–788, April 1999. DOI 10.1038/19718.

[18] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–887, December 2001. DOI 10.1038/414883a.

[19] S. Gulde, M. Riebe, G. P. T. Lancaster, C. Becher, J. Eschner, H. Häffner, F. Schmidt-Kaler, I. L. Chuang, and R. Blatt. Implementation of the Deutsch–Jozsa algorithm on an ion-trap quantum computer. *Nature*, 421(6918):48–50, January 2003. DOI 10.1038/nature01336.

[20] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019. DOI 10.1038/s41586-019-1666-5.

[21] J. Preskill. Quantum computing and the entanglement frontier, 2012. arXiv 1203.5813.

[22] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff. Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits, 2019. arXiv 1910.09534.

[23] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala. Evidence for the utility of quantum computing before fault tolerance. *Nature*, 618(7965):500–505, June 2023. DOI 10.1038/s41586-023-06096-3.

[24] T. Begušić and G. K. L. Chan. Fast classical simulation of evidence for the utility of quantum computing before fault tolerance, 2023. arXiv 2306.16372.

[25] J. Tindall, M. Fishman, M. Stoudenmire, and D. Sels. Efficient tensor network simulation of IBM's Eagle kicked Ising experiment, 2023. arXiv 2306.14887.

[26] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information.* Cambridge university press, 2010. ISBN 978-1-107-00217-3.

[27] A. Y. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, December 1997. DOI 10.1070/rm1997v052n06abeh002155.

[28] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. DOI 10.22331/q-2018-08-06-79.

[29] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters. Mixed-state entanglement and quantum error correction. *Physical Review A*, 54(5):3824–3851, November 1996. DOI 10.1103/physreva.54.3824.

[30] Z. Cai and S. C. Benjamin. Constructing Smaller Pauli Twirling Sets for Arbitrary Error Channels. *Scientific Reports*, 9(1), August 2019. DOI 10.1038/s41598-019-46722-7.

[31] J. J. Wallman and J. Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Phys. Rev. A*, 94:052325, November 2016. DOI 10.1103/PhysRevA.94.052325.

[32] C. Gustiani, T. Jones, and S. C. Benjamin. The Virtual Quantum Device (VQD): A tool for detailed emulation of quantum computers, 2023. arXiv 2306.07342.

[33] T. Toffoli. Bicontinuous extensions of invertible combinatorial functions. *Mathematical Systems Theory*, 14(1):13–23, December 1981. DOI 10.1007/bf01752388.

[34] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3-4):219–253, April 1982. DOI 10.1007/bf01857727.

[35] R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6): 507–531, June 1986. DOI 10.1007/bf01886518.

[36] D. E. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, September 1989. DOI 10.1098/rspa.1989.0099.

[37] S. Jordan. Quantum Algorithm Zoo, June 2022. URL https://quantumalgorithmzoo.org.

[38] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, November 1995. DOI 10.1103/PhysRevA.52.3457.

[39] D. Aharonov. A Simple Proof that Toffoli and Hadamard are Quantum Universal, 2003. arXiv quant-ph/0301040.

[40] S. Bravyi and A. Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Phys. Rev. A*, 71:022316, February 2005. DOI 10.1103/PhysRevA.71.022316.

[41] C. Piveteau, D. Sutter, S. Bravyi, J. M. Gambetta, and K. Temme. Error Mitigation for Universal Gates on Encoded Qubits. *Physical Review Letters*, 127(20), November 2021. DOI 10.1103/physrevlett.127.200505.

[42] N. Akerman, N. Navon, S. Kotler, Y. Glickman, and R. Ozeri. Universal gate-set for trapped-ion qubits using a narrow linewidth diode laser. *New Journal of Physics*, 17(11):113060, November 2015. DOI 10.1088/1367-2630/17/11/113060.

[43] Y. Shapira, R. Shaniv, T. Manovitz, N. Akerman, and R. Ozeri. Robust Entanglement Gates for Trapped-Ion Qubits. *Physical Review Letters*, 121(18), November 2018. DOI 10.1103/physrevlett.121.180502.

[44] A. Webb, S. Webster, S. Collingbourne, D. Bretaud, A. Lawrence, S. Weidt, F. Mintert, and W. Hensinger. Resilient Entangling Gates for Trapped Ions. *Physical Review Letters*, 121(18), November 2018. DOI 10.1103/physrevlett.121.180501.

[45] T. Manovitz, Y. Shapira, L. Gazit, N. Akerman, and R. Ozeri. A trapped ion quantum computer with robust entangling gates and quantum coherent feedback, 2021. arXiv 2111.04155.

[46] S. Ma, A. P. Burgers, G. Liu, J. Wilson, B. Zhang, and J. D. Thompson. Universal Gate Operations on Nuclear Spin Qubits in an Optical Tweezer Array of ^{171}Yb Atoms. *Physical Review X*, 12(2), May 2022. DOI 10.1103/physrevx.12.021028.

[47] J. M. Chow, J. M. Gambetta, A. D. Córcoles, S. T. Merkel, J. A. Smolin, C. Rigetti, S. Poletto, G. A. Keefe, M. B. Rothwell, J. R. Rozen, M. B. Ketchen, and M. Steffen. Universal Quantum Gate Set Approaching Fault-Tolerant Thresholds with Superconducting Qubits. *Physical Review Letters*, 109(6), August 2012. DOI 10.1103/physrevlett.109.060501.

[48] D. Zhu, T. Jaako, Q. He, and P. Rabl. Quantum Computing with Superconducting Circuits in the Picosecond Regime. *Physical Review Applied*, 16(1), July 2021. DOI 10.1103/physrevapplied.16.014024.

[49] J. Long, T. Zhao, M. Bal, R. Zhao, G. S. Barron, H. sheng Ku, J. A. Howard, X. Wu, C. R. H. McRae, X. H. Deng, G. J. Ribeill, M. Singh, T. A. Ohki, E. Barnes, S. E. Economou, and D. P. Pappas. A universal quantum gate set for transmon qubits with strong ZZ interactions, 2021. arXiv 2103.12305.

[50] K. Reuer, J. C. Besse, L. Wernli, P. Magnard, P. Kurpiers, G. J. Norris, A. Wallraff, and C. Eichler. Realization of a Universal Quantum Gate Set for Itinerant Microwave Photons. *Physical Review X*, 12(1), January 2022. DOI 10.1103/physrevx.12.011008.

[51] T. Wu and J. Guo. Computational Assessment of Silicon Quantum Gate Based on Detuning Mechanism for Quantum Computing. *IEEE Transactions on Electron Devices*, 65(12):5530–5536, December 2018. DOI 10.1109/ted.2018.2876355.

[52] E. Ferraro, D. Rei, M. Paris, and M. D. Michielis. Universal set of quantum gates for the flip-flop qubit in the presence of 1/f noise. *EPJ Quantum Technology*, 9 (1), January 2022. DOI 10.1140/epjqt/s40507-022-00120-7.

[53] T. Evans, W. Huang, J. Yoneda, R. Harper, T. Tanttu, K. Chan, F. Hudson, K. Itoh, A. Saraiva, C. Yang, A. Dzurak, and S. Bartlett. Fast Bayesian Tomography of a Two-Qubit Gate Set in Silicon. *Physical Review Applied*, 17(2), February 2022. DOI 10.1103/physrevapplied.17.024068.

[54] A. Noiri, K. Takeda, T. Nakajima, T. Kobayashi, A. Sammak, G. Scappucci, and S. Tarucha. Fast universal quantum gate above the fault-tolerance threshold in silicon. *Nature*, 601(7893):338–342, January 2022. DOI 10.1038/s41586-021-04182-y.

[55] A. R. Mills, C. R. Guinn, M. J. Gullans, A. J. Sigillito, M. M. Feldman, E. Nielsen, and J. R. Petta. Two-qubit silicon quantum processor with operation fidelity exceeding 99%. *Science Advances*, 8(14), April 2022. DOI 10.1126/sciadv.abn5130.

[56] V. V. Shende, I. L. Markov, and S. S. Bullock. Minimal universal two-qubit controlled-NOT-based circuits. *Phys. Rev. A*, 69:062321, June 2004. DOI 10.1103/PhysRevA.69.062321.

[57] M. Howard, J. Wallman, V. Veitch, and J. Emerson. Contextuality supplies the 'magic' for quantum computation. *Nature*, 510(7505):351–355, June 2014. DOI 10.1038/nature13460.

[58] E. T. Campbell, B. M. Terhal, and C. Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179, September 2017. DOI 10.1038/nature23460.

[59] J. J. Vartiainen, M. Möttönen, and M. M. Salomaa. Efficient Decomposition of Quantum Gates. *Phys. Rev. Lett.*, 92:177902, April 2004. DOI 10.1103/PhysRevLett.92.177902.

[60] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa. Quantum Circuits for General Multiqubit Gates. *Phys. Rev. Lett.*, 93:130502, September 2004. DOI 10.1103/PhysRevLett.93.130502.

[61] M. Möttönen and J. J. Vartiainen. Decompositions of general quantum gates, 2005. arXiv quant-ph/0504100.

[62] V. Shende, S. Bullock, and I. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25 (6):1000–1010, June 2006. DOI 10.1109/tcad.2005.855930.

[63] R. R. Tucci. A Rudimentary Quantum Compiler, 1998. arXiv quant-ph/9805015.

[64] R. R. Tucci. An Optimization for Qubiter, 1998. arXiv quant-ph/9809055.

[65] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl. Quantum circuits for isometries. *Phys. Rev. A*, 93:032318, Mar 2016. DOI 10.1103/PhysRevA.93.032318.

[66] R. Iten, O. Reardon-Smith, E. Malvetti, L. Mondada, G. Pauvert, E. Redmond, R. S. Kohli, and R. Colbeck. Introduction to UniversalQCompiler, 2021. arXiv 1904.01072.

[67] A. M. Krol, A. Sarkar, I. Ashraf, Z. Al-Ars, and K. Bertels. Efficient Decomposition of Unitary Matrices in Quantum Circuit Compilers. *Applied Sciences*, 12(2):759, January 2022. DOI 10.3390/app12020759.

[68] D. Janzing, P. Wocjan, and T. Beth. Identity check is QMA-complete, 2003. arXiv quant-ph/0305050.

[69] B. Rosgen and J. Watrous. On the Hardness of Distinguishing Mixed-State Quantum Computations. In *20th Annual IEEE Conference on Computational Complexity (CCC'05)*. IEEE, 2005. DOI 10.1109/ccc.2005.21.

[70] A. D. Bookatz. QMA-Complete Problems. *Quantum Info. Comput.*, 14(5 & 6): 361–383, April 2014. DOI 10.26421/QIC14.5-6-1.

[71] E. Younis, K. Sen, K. Yelick, and C. Iancu. QFAST: Conflating Search and Numerical Optimization for Scalable Quantum Circuit Synthesis. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, October 2021. DOI 10.1109/qce52317.2021.00041.

[72] T. Patel, E. Younis, C. Iancu, W. de Jong, and D. Tiwari. QUEST: Systematically Approximating Quantum Circuits for Higher Output Fidelity. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, page 514–528. Association for Computing Machinery, 2022. DOI 10.1145/3503222.3507739.

[73] X. Zhou, S. Li, and Y. Feng. Quantum Circuit Transformation Based on Simulated Annealing and Heuristic Search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4683–4694, December 2020. DOI 10.1109/tcad.2020.2969647.

[74] H. Jiang, Y. Deng, and M. Xu. Quantum Circuit Transformation Based on Tabu Search, 2021. arXiv 2104.05214.

[75] A. Paler, L. Sasu, A. C. Florea, and R. Andonie. Machine Learning Optimization of Quantum Circuit Layouts. *ACM Transactions on Quantum Computing*, 4(2), February 2023. DOI 10.1145/3565271.

[76] X. Zhou, Y. Feng, and S. Li. Supervised Learning Enhanced Quantum Circuit Transformation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(2):437–447, February 2023. DOI 10.1109/tcad.2022.3179223.

[77] A. M. Childs, E. Schoute, and C. M. Unsal. Circuit Transformations for Quantum Architectures. 2019. DOI 10.4230/LIPICS.TQC.2019.3.

[78] G. Li, Y. Ding, and Y. Xie. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, April 2019. DOI 10.1145/3297858.3304023.

[79] S. Niu, A. Suau, G. Staffelbach, and A. Todri-Sanial. A Hardware-Aware Heuristic for the Qubit Mapping Problem in the NISQ Era. *IEEE Transactions on Quantum Engineering*, 1:1–14, 2020. DOI 10.1109/tqe.2020.3026544.

[80] X. Zhou, Y. Feng, and S. Li. Quantum Circuit Transformation: A Monte Carlo Tree Search Framework. *ACM Transactions on Design Automation of Electronic Systems*, 27(6):1–27, June 2022. DOI 10.1145/3514239.

[81] D. Devulapalli, E. Schoute, A. Bapat, A. M. Childs, and A. V. Gorshkov. Quantum Routing with Teleportation, 2022. arXiv 2204.04185.

[82] C. Gustiani and D. P. DiVincenzo. Blind three-qubit exact Grover search on a nitrogen-vacancy-center platform. *Phys. Rev. A*, 104:062422, December 2021. DOI 10.1103/PhysRevA.104.062422.

[83] P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. DOI 10.1109/tssc.1968.300136.

[84] M. G. Davis, E. Smith, A. Tudor, K. Sen, I. Siddiqi, and C. Iancu. Towards Optimal Topology Aware Quantum Circuit Synthesis. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, October 2020. DOI 10.1109/qce49297.2020.00036.

[85] E. Smith, M. G. Davis, J. Larson, E. Younis, L. B. Oftelie, W. Lavrijsen, and C. Iancu. LEAP: Scaling Numerical Optimization Based Synthesis Using an Incremental Approach. *ACM Transactions on Quantum Computing*, 4(1):1–23, February 2023. DOI 10.1145/3548693.

[86] O. D. Matteo and M. Mosca. Parallelizing quantum circuit synthesis. *Quantum Science and Technology*, 1(1):015003, March 2016. DOI 10.1088/2058-9565/1/1/015003.

[87] L. Arufe, M. A. González, A. Oddi, R. Rasconi, and R. Varela. Quantum circuit compilation by genetic algorithm for quantum approximate optimization algorithm applied to MaxCut problem. *Swarm and Evolutionary Computation*, 69: 101030, March 2022. DOI 10.1016/j.swevo.2022.101030.

[88] D. Venturelli, M. Do, E. Rieffel, and J. Frank. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology*, 3(2):025004, February 2018. DOI 10.1088/2058-9565/aaa331.

[89] L. Moro, M. G. A. Paris, M. Restelli, and E. Prati. Quantum compiling by deep reinforcement learning. *Communications Physics*, 4(1), August 2021. DOI 10.1038/s42005-021-00684-3.

[90] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles. Learning the quantum algorithm for state overlap. *New Journal of Physics*, 20(11):113022, November 2018. DOI 10.1088/1367-2630/aae94a.

[91] M. Bilkis, M. Cerezo, G. Verdon, P. J. Coles, and L. Cincio. A semi-agnostic ansatz with variable structure for quantum machine learning, 2023. arXiv 2103.06712.

[92] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles. Quantum-assisted quantum compiling. *Quantum*, 3:140, May 2019. DOI 10.22331/q-2019-05-13-140.

[93] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January 1965. DOI 10.1093/comjnl/7.4.308.

[94] M. J. D. Powell. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer Netherlands, 1994. DOI 10.1007/978-94-015-8330-5_4.

[95] W. Lavrijsen, A. Tudor, J. Muller, C. Iancu, and W. de Jong. Classical Optimizers for Noisy Intermediate-Scale Quantum Devices. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, October 2020. DOI 10.1109/qce49297.2020.00041.

[96] G. Boyd and B. Koczor. Training Variational Quantum Circuits with CoVaR: Covariance Root Finding with Classical Shadows. *Phys. Rev. X*, 12:041022, November 2022. DOI 10.1103/PhysRevX.12.041022.

[97] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409, December 1952. DOI 10.6028/jres.049.044.

[98] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2017. arXiv 1412.6980.

[99] J. Stokes, J. Izaac, N. Killoran, and G. Carleo. Quantum Natural Gradient. *Quantum*, 4:269, May 2020. DOI 10.22331/q-2020-05-25-269.

[100] S. McArdle, T. Jones, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, 5(1), September 2019. DOI 10.1038/s41534-019-0187-2.

[101] Y. Du, Z. Tu, X. Yuan, and D. Tao. Efficient Measure for the Expressivity of Variational Quantum Algorithms. *Physical Review Letters*, 128(8), February 2022. DOI 10.1103/physrevlett.128.080506.

[102] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles. Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus. *PRX Quantum*, 3(1), January 2022. DOI 10.1103/prxquantum.3.010313.

[103] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), November 2018. DOI 10.1038/s41467-018-07090-4.

[104] E. R. Anschuetz and B. T. Kiani. Quantum variational algorithms are swamped with traps. *Nature Communications*, 13(1), December 2022. DOI 10.1038/s41467-022-35364-5.

[105] D. Wierichs, C. Gogolin, and M. Kastoryano. Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer. *Physical Review Research*, 2(4), November 2020. DOI 10.1103/physrevresearch.2.043246.

[106] L. Bittel and M. Kliesch. Training Variational Quantum Algorithms Is NP-Hard. *Phys. Rev. Lett.*, 127:120502, September 2021. DOI 10.1103/PhysRevLett.127.120502.

[107] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson. The Variational Quantum Eigensolver: A review of methods and best practices. *Physics Reports*, 986:1–128, November 2022. DOI 10.1016/j.physrep.2022.08.003.

[108] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 10(1), July 2019. DOI 10.1038/s41467-019-10988-2.

[109] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou. Qubit-ADAPT-VQE: An Adaptive Algorithm for Constructing Hardware-Efficient Ansätze on a Quantum Processor. *PRX Quantum*, 2:020310, April 2021. DOI 10.1103/PRXQuantum.2.020310.

[110] A. G. Rattew, S. Hu, M. Pistoia, R. Chen, and S. Wood. A Domain-agnostic, Noise-resistant, Hardware-efficient Evolutionary Variational Quantum Eigensolver, 2020. arXiv 1910.09694.

[111] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959. DOI 10.1090/s0002-9939-1959-0108732-6.

[112] M. Suzuki. Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Communications in Mathematical Physics*, 51(2):183–190, June 1976. DOI 10.1007/bf01609348.

[113] M. Suzuki. Fractal decomposition of exponential operators with applications to many-body theories and Monte Carlo simulations. *Physics Letters A*, 146(6): 319–323, June 1990. DOI 10.1016/0375-9601(90)90962-n.

[114] N. Wiebe, D. Berry, P. Høyer, and B. C. Sanders. Higher order decompositions of ordered operator exponentials. *Journal of Physics A: Mathematical and Theoretical*, 43(6):065203, January 2010. DOI 10.1088/1751-8113/43/6/065203.

[115] S. Lloyd. Universal Quantum Simulators. *Science*, 273(5278):1073–1078, August 1996. DOI 10.1126/science.273.5278.1073.

[116] D. S. Abrams and S. Lloyd. Simulation of Many-Body Fermi Systems on a Universal Quantum Computer. *Phys. Rev. Lett.*, 79:2586–2589, September 1997. DOI 10.1103/PhysRevLett.79.2586.

[117] G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme. Quantum algorithms for fermionic simulations. *Phys. Rev. A*, 64:022319, July 2001. DOI 10.1103/PhysRevA.64.022319.

[118] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon. Simulated Quantum Computation of Molecular Energies. *Science*, 309(5741):1704–1707, September 2005. DOI 10.1126/science.1113479.

[119] H. Wang, S. Kais, A. Aspuru-Guzik, and M. R. Hoffmann. Quantum algorithm for obtaining the energy spectrum of molecular systems. *Physical Chemistry Chemical Physics*, 10(35):5388, 2008. DOI 10.1039/b804804e.

[120] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik. Simulation of electronic structure Hamiltonians using quantum computers. *Molecular Physics*, 109(5): 735–750, March 2011. DOI 10.1080/00268976.2011.552441.

[121] D. Wecker, B. Bauer, B. K. Clark, M. B. Hastings, and M. Troyer. Gate-count estimates for performing quantum chemistry on small quantum computers. *Phys. Rev. A*, 90:022305, August 2014. DOI 10.1103/PhysRevA.90.022305.

[122] D. Poulin, M. B. Hastings, D. Wecker, N. Wiebe, A. C. Doberty, and M. Troyer. The Trotter Step Size Required for Accurate Quantum Simulation of Quantum Chemistry. *Quantum Info. Comput.*, 15(5–6):361–384, April 2015. DOI 10.26421/QIC15.5-6-1.

[123] M. B. Hastings, D. Wecker, B. Bauer, and M. Troyer. Improving Quantum Algorithms for Quantum Chemistry. *Quantum Info. Comput.*, 15(1–2):1–21, January 2015. DOI 10.26421/QIC15.1-2-1.

[124] A. M. Childs, A. Ostrander, and Y. Su. Faster quantum simulation by randomization. *Quantum*, 3:182, September 2019. DOI 10.22331/q-2019-09-02-182.

[125] E. Campbell. Random Compiler for Fast Hamiltonian Simulation. *Phys. Rev. Lett.*, 123:070503, August 2019. DOI 10.1103/PhysRevLett.123.070503.

[126] C. F. Chen, H. Y. Huang, R. Kueng, and J. A. Tropp. Concentration for Random Product Formulas. *PRX Quantum*, 2:040305, October 2021. DOI 10.1103/PRXQuantum.2.040305.

[127] Y. Ouyang, D. R. White, and E. T. Campbell. Compilation by stochastic Hamiltonian sparsification. *Quantum*, 4:235, February 2020. DOI 10.22331/q-2020-02-27-235.

[128] R. Babbush, J. McClean, D. Wecker, A. Aspuru-Guzik, and N. Wiebe. Chemical basis of Trotter-Suzuki errors in quantum chemistry simulation. *Phys. Rev. A*, 91: 022311, February 2015. DOI 10.1103/PhysRevA.91.022311.

[129] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu. Theory of Trotter Error with Commutator Scaling. *Phys. Rev. X*, 11:011020, February 2021. DOI 10.1103/PhysRevX.11.011020.

[130] A. M. Childs and N. Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information and Computation*, 12(11&12): 901–924, November 2012. DOI 10.26421/qic12.11-12-1.

[131] S. A. Chin and J. Geiser. Multi-product operator splitting as a general method of solving autonomous and non-autonomous equations, 2010. arXiv 1005.2201.

[132] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma. Simulating Hamiltonian Dynamics with a Truncated Taylor Series. *Phys. Rev. Lett.*, 114: 090502, March 2015. DOI 10.1103/PhysRevLett.114.090502.

[133] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma. Exponential Improvement in Precision for Simulating Sparse Hamiltonians. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 283–292. Association for Computing Machinery, 2014. DOI 10.1145/2591796.2591854.

[134] L. Novo and D. Berry. Improved Hamiltonian simulation via a truncated Taylor series and corrections. *Quantum Information and Computation*, 17(7&8):623–635, May 2017. DOI 10.26421/qic17.7-8-5.

[135] G. H. Low, V. Kliuchnikov, and N. Wiebe. Well-conditioned multiproduct Hamiltonian simulation, 2019. arXiv 1907.11679.

[136] D. W. Berry, A. M. Childs, and R. Kothari. Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, October 2015. DOI 10.1109/focs.2015.54.

[137] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven. Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity. *Phys. Rev. X*, 8:041015, October 2018. DOI 10.1103/PhysRevX.8.041015.

[138] G. H. Low and I. L. Chuang. Optimal Hamiltonian Simulation by Quantum Signal Processing. *Phys. Rev. Lett.*, 118:010501, January 2017. DOI 10.1103/PhysRevLett.118.010501.

[139] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. ACM, June 2019. DOI 10.1145/3313276.3316366.

[140] G. H. Low and I. L. Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, July 2019. DOI 10.22331/q-2019-07-12-163.

[141] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush. Qubitization of Arbitrary Basis Quantum Chemistry Leveraging Sparsity and Low Rank Factorization. *Quantum*, 3:208, December 2019. DOI 10.22331/q-2019-12-02-208.

[142] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, September 2018. DOI 10.1073/pnas.1801723115.

[143] G. H. Low and N. Wiebe. Hamiltonian Simulation in the Interaction Picture, 2019. arXiv 1805.00675.

[144] R. Babbush, D. W. Berry, J. R. McClean, and H. Neven. Quantum simulation of chemistry with sublinear scaling in basis size. *npj Quantum Information*, 5(1), November 2019. DOI 10.1038/s41534-019-0199-y.

[145] A. Y. Kitaev. Quantum measurements and the Abelian Stabilizer Problem, 1995. arXiv quant-ph/9511026.

[146] M. Dobšíček, G. Johansson, V. Shumeiko, and G. Wendin. Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark. *Phys. Rev. A*, 76:030306, September 2007. DOI 10.1103/PhysRevA.76.030306.

[147] K. M. Svore, M. B. Hastings, and M. H. Freedman. Faster phase estimation. *Quantum Inf. Comput.*, 14(3-4):306–328, 2014. DOI 10.26421/QIC14.3-4-7.

[148] N. Wiebe and C. Granade. Efficient Bayesian Phase Estimation. *Phys. Rev. Lett.*, 117:010503, June 2016. DOI 10.1103/PhysRevLett.117.010503.

[149] K. Sugisaki, C. Sakai, K. Toyota, K. Sato, D. Shiomi, and T. Takui. Bayesian phase difference estimation: a general quantum algorithm for the direct calculation of energy gaps. *Physical Chemistry Chemical Physics*, 23(36):20152–20162, 2021. DOI 10.1039/d1cp03156b.

[150] K. Sugisaki, K. Toyota, K. Sato, D. Shiomi, and T. Takui. A quantum algorithm for spin chemistry: a Bayesian exchange coupling parameter calculator with broken-symmetry wave functions. *Chemical Science*, 12(6):2121–2132, 2021. DOI 10.1039/d0sc04847j.

[151] K. Sugisaki, K. Toyota, K. Sato, D. Shiomi, and T. Takui. Quantum Algorithm for the Direct Calculations of Vertical Ionization Energies. *The Journal of Physical Chemistry Letters*, 12(11):2880–2885, March 2021. DOI 10.1021/acs.jpclett.1c00283.

[152] R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme. Simulating physical phenomena by quantum networks. *Phys. Rev. A*, 65:042323, April 2002. DOI 10.1103/PhysRevA.65.042323.

[153] T. E. O'Brien, B. Tarasinski, and B. M. Terhal. Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments. *New Journal of Physics*, 21(2):023022, February 2019. DOI 10.1088/1367-2630/aafb8e.

[154] R. D. Somma. Quantum eigenvalue estimation via time series analysis. *New Journal of Physics*, 21(12):123025, December 2019. DOI 10.1088/1367-2630/ab5c60.

[155] L. Clinton, J. Bausch, J. Klassen, and T. Cubitt. Phase estimation of local Hamiltonians on NISQ hardware. *New Journal of Physics*, 25(3):033027, March 2023. DOI 10.1088/1367-2630/acc26d.

[156] L. Clinton, J. Bausch, and T. Cubitt. Hamiltonian simulation algorithms for near-term quantum hardware. *Nature Communications*, 12(1), August 2021. DOI 10.1038/s41467-021-25196-0.

[157] F. Chapeau-Blondeau and E. Belin. Quantum signal processing for quantum phase estimation: Fourier transform versus maximum likelihood approaches. *Annals of Telecommunications*, 75(11-12):641–653, September 2020. DOI 10.1007/s12243-020-00803-1.

[158] J. G. Smith, C. H. W. Barnes, and D. R. M. Arvidsson-Shukur. Iterative quantum-phase-estimation protocol for shallow circuits. *Phys. Rev. A*, 106:062615, December 2022. DOI 10.1103/PhysRevA.106.062615.

[159] J. K. L. MacDonald. On the Modified Ritz Variation Method. *Physical Review*, 46 (9):828–828, November 1934. DOI 10.1103/physrev.46.828.

[160] L. C. Tazi and A. J. W. Thom. Folded Spectrum VQE : A quantum computing method for the calculation of molecular excited states, 2023. arXiv 2305.04783.

[161] O. Higgott, D. Wang, and S. Brierley. Variational Quantum Computation of Excited States. *Quantum*, 3:156, July 2019. DOI 10.22331/q-2019-07-01-156.

[162] T. Jones, S. Endo, S. McArdle, X. Yuan, and S. C. Benjamin. Variational quantum algorithms for discovering Hamiltonian spectra. *Physical Review A*, 99(6), June 2019. DOI 10.1103/physreva.99.062304.

[163] D. B. Zhang, B. L. Chen, Z. H. Yuan, and T. Yin. Variational quantum eigensolvers by variance minimization. *Chinese Physics B*, 31(12):120301, November 2022. DOI 10.1088/1674-1056/ac8a8d.

[164] A. E. Russo, K. M. Rudinger, B. C. A. Morrison, and A. D. Baczewski. Evaluating Energy Differences on a Quantum Computer with Robust Phase Estimation. *Phys. Rev. Lett.*, 126:210501, May 2021. DOI 10.1103/PhysRevLett.126.210501.

[165] S. Kimmel, G. H. Low, and T. J. Yoder. Robust calibration of a universal single-qubit gate set via robust phase estimation. *Phys. Rev. A*, 92:062315, December 2015. DOI 10.1103/PhysRevA.92.062315.

[166] Y. Yang, Y. Li, X. Xu, and X. Yuan. A resource-efficient quantum-classical hybrid algorithm for energy gap evaluation, 2023. arXiv 2305.07382.

[167] J. Sun, L. Vilchez-Estevez, V. Vedral, A. T. Boothroyd, and M. S. Kim. Probing spectral features of quantum many-body systems with quantum simulators, 2023. arXiv 2305.07649.

[168] K. P. Gnatenko, H. P. Laba, and V. M. Tkachuk. Energy levels estimation on a quantum computer by evolution of a physical quantity. *Physics Letters A*, 424: 127843, February 2022. DOI 10.1016/j.physleta.2021.127843.

[169] M. E. Stroeks, J. Helsen, and B. M. Terhal. Spectral estimation for Hamiltonians: a comparison between classical imaginary-time evolution and quantum real-time evolution. *New Journal of Physics*, 24(10):103024, October 2022. DOI 10.1088/1367-2630/ac919c.

[170] A. Paulraj, R. Roy, and T. Kailath. Estimation Of Signal Parameters Via Rotational Invariance Techniques- Esprit. In *Nineteeth Asilomar Conference on Circuits, Systems and Computers, 1985.* IEEE, 1985. DOI 10.1109/acssc.1985.671426.

[171] Y. Matsuzaki, H. Hakoshima, K. Sugisaki, Y. Seki, and S. Kawabata. Direct estimation of the energy gap between the ground state and excited state with quantum annealing. *Japanese Journal of Applied Physics*, 60(SB):SBBI02, February 2021. DOI 10.35848/1347-4065/abdf20.

[172] C. S. Unnikrishnan. Quantum Non-Demolition Measurements: Concepts, Theory and Practice. *Current Science*, 109(11):2052, December 2015. DOI 10.18520/v109/i11/2052-2060.

[173] V. B. Braginsky, Y. I. Vorontsov, and K. S. Thorne. Quantum Nondemolition Measurements. *Science*, 209(4456):547–557, August 1980. DOI 10.1126/science.209.4456.547.

[174] D. Yang, A. Grankin, L. M. Sieberer, D. V. Vasilyev, and P. Zoller. Quantum non-demolition measurement of a many-body Hamiltonian. *Nature Communications*, 11(1), February 2020. DOI 10.1038/s41467-020-14489-5.

[175] K. Choi, D. Lee, J. Bonitati, Z. Qian, and J. Watkins. Rodeo Algorithm for Quantum Computing. *Phys. Rev. Lett.*, 127:040505, July 2021. DOI 10.1103/PhysRevLett.127.040505.

[176] Qiskit contributors. Qiskit: An Open-source Framework for Quantum Computing, 2023. DOI 10.5281/zenodo.2573505.

[177] Cirq Developers. Cirq, 2023. DOI 10.5281/zenodo.4062499.

[178] Quantum AI team and collaborators. qsim, September 2020. DOI 10.5281/zenodo.4023103.

[179] Amazon Web Services. Amazon Braket, 2020. URL https://aws.amazon.com/braket/.

[180] Microsoft Azure Quantum team. Azure Quantum, 2023. URL https://azure.microsoft.com/products/quantum/.

[181] D. S. Steiger, T. Häner, and M. Troyer. ProjectQ: an open source software framework for quantum computing. *Quantum*, 2:49, January 2018. DOI 10.22331/q-2018-01-31-49.

[182] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. D. Sawaya. Intel Quantum Simulator: a cloud-ready high-performance simulator of quantum circuits. *Quantum Science and Technology*, 5(3):034007, May 2020. DOI 10.1088/2058-9565/ab8505.

[183] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik. qHiPSTER: The Quantum High Performance Software Testing Environment, 2016. arXiv 1601.07195.

[184] Y. Suzuki, Y. Kawase, Y. Masumura, Y. Hiraga, M. Nakadai, J. Chen, K. M. Nakanishi, K. Mitarai, R. Imai, S. Tamiya, T. Yamamoto, T. Yan, T. Kawakubo, Y. O. Nakagawa, Y. Ibe, Y. Zhang, H. Yamashita, H. Yoshimura, A. Hayashi, and K. Fujii. Qulacs: a fast and versatile quantum circuit simulator for research purpose. *Quantum*, 5:559, October 2021. DOI 10.22331/q-2021-10-06-559.

[185] X. Z. Luo, J. G. Liu, P. Zhang, and L. Wang. Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design. *Quantum*, 4:341, October 2020. DOI 10.22331/q-2020-10-11-341.

[186] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. DOI 10.1137/141000671.

[187] T. Jones, A. Brown, I. Bush, and S. C. Benjamin. QuEST and High Performance Simulation of Quantum Computers. *Scientific Reports*, 9(1), July 2019. DOI 10.1038/s41598-019-47174-9.

[188] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix Product State Representations, 2007. arXiv quant-ph/0608197.

[189] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum*, 3: 181, September 2019. DOI 10.22331/q-2019-09-02-181.

[190] Quantiki Contributors. List of QC simulators, August 2023. URL https://quantiki.org/wiki/list-qc-simulators.

[191] R. Meister, C. Gustiani, and S. C. Benjamin. Exploring ab initio machine synthesis of quantum circuits. *New Journal of Physics*, 25(7):073018, July 2023. DOI 10.1088/1367-2630/ace077.

[192] R. Meister. pyQuEST - A Python interface for the Quantum Exact Simulation Toolkit, July 2022. URL https://github.com/rrmeister/pyQuEST.

[193] T. Jones and S. Benjamin. QuESTlink—Mathematica embiggened by a hardware-optimised quantum emulator. *Quantum Science and Technology*, 5(3):034012, May 2020. DOI 10.1088/2058-9565/ab8506.

[194] T. Jones and S. C. Benjamin. Robust quantum compilation and circuit optimisation via energy minimisation. *Quantum*, 6:628, January 2022. DOI 10.22331/q-2022-01-24-628.

[195] L. Cincio, K. Rudinger, M. Sarovar, and P. J. Coles. Machine Learning of Noise-Resilient Quantum Circuits. *PRX Quantum*, 2:010324, February 2021. DOI 10.1103/PRXQuantum.2.010324.

[196] M. C. Caro, H. Y. Huang, N. Ezzell, J. Gibbs, A. T. Sornborger, L. Cincio, P. J. Coles, and Z. Holmes. Out-of-distribution generalization for learning quantum dynamics. *Nature Communications*, 14(1), July 2023. DOI 10.1038/s41467-023-39381-w.

[197] M. C. Caro, H. Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. J. Coles. Generalization in quantum machine learning from few training data. *Nature Communications*, 13(1), August 2022. DOI 10.1038/s41467-022-32550-3.

[198] J. Gibbs, Z. Holmes, M. C. Caro, N. Ezzell, H. Y. Huang, L. Cincio, A. T. Sornborger, and P. J. Coles. Dynamical simulation via quantum machine learning with provable generalization, 2022. arXiv 2204.10269.

[199] A. Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Reports on Mathematical Physics*, 3(4):275–278, December 1972. DOI 10.1016/0034-4877(72)90011-0.

[200] M. D. Choi. Completely positive linear maps on complex matrices. *Linear Algebra and its Applications*, 10(3):285–290, June 1975. DOI 10.1016/0024-3795(75)90075-0.

[201] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, January 1998. DOI 10.1098/rspa.1998.0164.

[202] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1), March 2021. DOI 10.1038/s41467-021-21728-w.

[203] R.W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29(2):147–160, April 1950. DOI 10.1002/j.1538-7305.1950.tb00463.x.

[204] X. Yuan, S. Endo, Q. Zhao, Y. Li, and S.C. Benjamin. Theory of variational quantum simulation. *Quantum*, 3:191, October 2019. DOI 10.22331/q-2019-10-07-191.

[205] B. Koczor and S. C. Benjamin. Quantum natural gradient generalized to noisy and nonunitary circuits. *Phys. Rev. A*, 106:062416, December 2022. DOI 10.1103/PhysRevA.106.062416.

[206] N. Yamamoto. On the natural gradient for variational quantum eigensolver, 2019. arXiv 1909.05074.

[207] B. van Straaten and B. Koczor. Measurement Cost of Metric-Aware Variational Quantum Algorithms. *PRX Quantum*, 2:030324, August 2021. DOI 10.1103/PRXQuantum.2.030324.

[208] S. S. Skiena. *The Algorithm Design Manual.* Springer, third edition, 2020. ISBN 978-3-030-54256-6.

[209] F. Glover. Tabu Search: A Tutorial. *Interfaces*, 20(4):74–94, August 1990. DOI 10.1287/inte.20.4.74.

[210] C. Gustiani, R. Meister, and S. C. Benjamin. Exploiting subspace constraints and ab initio variational methods for quantum chemistry. *New Journal of Physics*, 25 (7):073019, July 2023. DOI 10.1088/1367-2630/ace182.

[211] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1), March 2020. DOI 10.1103/revmodphys.92.015003.

[212] H. H. S. Chan, R. Meister, T. Jones, D. P. Tew, and S. C. Benjamin. Grid-based methods for chemistry simulations on a quantum computer. *Science Advances*, 9(9), March 2023. DOI 10.1126/sciadv.abo7484.

[213] R. Meister, S. C. Benjamin, and E. T. Campbell. Tailoring Term Truncations for Electronic Structure Calculations Using a Linear Combination of Unitaries. *Quantum*, 6:637, February 2022. DOI 10.22331/q-2022-02-02-637.

[214] P. Jordan and E. Wigner. Über das Paulische Äquivalenzverbot. *Zeitschrift für Physik*, 47(9-10):631–651, September 1928. DOI 10.1007/bf01331938.

[215] S. B. Bravyi and A. Y. Kitaev. Fermionic Quantum Computation. *Annals of Physics*, 298(1):210–226, May 2002. DOI 10.1006/aphy.2002.6254.

[216] G. H. Low, V. Kliuchnikov, and L. Schaeffer. Trading T-gates for dirty qubits in state preparation and unitary synthesis, 2018. arXiv 1812.00954.

[217] T. Helgaker, P. Jorgensen, and J. Olsen. *Molecular electronic-structure theory*. John Wiley & Sons, Chichester, England, August 2000. ISBN 978-0-471-96755-2.

[218] J. R. McClean, K. J. Sung, I. D. Kivlichan, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, P. Gokhale, T. Häner, T. Hardikar, V. Havlíček, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle, M. Neeley, T. O'Brien, B. O'Gorman, I. Ozfidan, M. D. Radin, J. Romero, N. Rubin, N. P. D. Sawaya, K. Setia, S. Sim, D. S. Steiger, M. Steudtner, Q. Sun, W. Sun, D. Wang, F. Zhang, and R. Babbush. OpenFermion: The Electronic Structure Package for Quantum Computers, 2019. URL `https://openfermion.org`. arXiv 1710.07629.

[219] Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, S. Wouters, and G. K. L. Chan. PySCF: the Python-based simulations of chemistry framework. *WIREs Computational Molecular Science*, 8(1), September 2017. DOI 10.1002/wcms.1340.

[220] Q. Sun, X. Zhang, S. Banerjee, P. Bao, M. Barbry, N. S. Blunt, N. A. Bogdanov, G. H. Booth, J. Chen, Z. H. Cui, J. J. Eriksen, Y. Gao, S. Guo, J. Hermann, M. R. Hermes, K. Koh, P. Koval, S. Lehtola, Z. Li, J. Liu, N. Mardirossian, J. D. McClain, M. Motta, B. Mussard, H. Q. Pham, A. Pulkin, W. Purwanto, P. J. Robinson, E. Ronca, E. R. Sayfutyarova, M. Scheurer, H. F. Schurkus, J. E. T. Smith, C. Sun, S. N. Sun, S. Upadhyay, L. K. Wagner, X. Wang, A. White, J. D. Whitfield, M. J. Williamson, S. Wouters, J. Yang, J. M. Yu, T. Zhu, T. C. Berkelbach, S. Sharma, A. Y. Sokolov, and G. K. L. Chan. Recent developments in the PySCF program package. *The Journal of Chemical Physics*, 153(2), July 2020. DOI 10.1063/5.0006074.

[221] W. J. Hehre, R. F. Stewart, and J. A. Pople. Self-Consistent Molecular-Orbital Methods. I. Use of Gaussian Expansions of Slater-Type Atomic Orbitals. *The Journal of Chemical Physics*, 51(6):2657–2664, September 1969. DOI 10.1063/1.1672392.

[222] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, and S. H. Bryant. PubChem Substance and Compound databases. *Nucleic Acids Research*, 44(D1):D1202–D1213, September 2015. DOI 10.1093/nar/gkv951.

[223] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, L. Zaslavsky, J. Zhang, and E. E. Bolton. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, October 2018. DOI 10.1093/nar/gky1033.

[224] R. D. Johnson III. Computational Chemistry Comparison and Benchmark Database, NIST Standard Reference Database 101, Release 22, May 2022. URL http://cccbdb.nist.gov. DOI 10.18434/t47c7z.

[225] T. H. Dunning. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. *The Journal of Chemical Physics*, 90(2):1007–1023, January 1989. DOI 10.1063/1.456153.

[226] B. P. Prascher, D. E. Woon, K. A. Peterson, T. H. Dunning, and A. K. Wilson. Gaussian basis sets for use in correlated molecular calculations. VII. Valence, core-valence, and scalar relativistic basis sets for Li, Be, Na, and Mg. *Theoretical Chemistry Accounts*, 128(1):69–82, May 2011. DOI 10.1007/s00214-010-0764-0.

[227] S. Wiesner. Simulations of Many-Body Quantum Systems by a Quantum Computer, 1996. arXiv quant-ph/9603028.

[228] C. Zalka. Simulating quantum systems on a quantum computer. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):313–322, January 1998. DOI 10.1098/rspa.1998.0162.

[229] I. Kassal, S. P. Jordan, P. J. Love, M. Mohseni, and A. Aspuru-Guzik. Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Sciences*, 105(48):18681–18686, December 2008. DOI 10.1073/pnas.0808245105.

[230] I. D. Kivlichan, N. Wiebe, R. Babbush, and A. Aspuru-Guzik. Bounding the costs of quantum simulation of many-body physics in real space. *Journal of Physics A: Mathematical and Theoretical*, 50(30):305301, June 2017. DOI 10.1088/1751-8121/aa77b8.

[231] Y. Su, D. W. Berry, N. Wiebe, N. Rubin, and R. Babbush. Fault-Tolerant Quantum Simulations of Chemistry in First Quantization. *PRX Quantum*, 2:040332, Nov 2021. DOI 10.1103/PRXQuantum.2.040332.

[232] T. Kosugi, Y. Nishiya, H. Nishi, and Y. i. Matsushita. Imaginary-time evolution using forward and backward real-time evolution with a single ancilla: First-quantized eigensolver algorithm for quantum chemistry. *Phys. Rev. Res.*, 4: 033121, August 2022. DOI 10.1103/PhysRevResearch.4.033121.

[233] B. Poirier and J. Jerke. Full-dimensional Schrödinger wavefunction calculations using tensors and quantum computers: the Cartesian component-separated approach. *Physical Chemistry Chemical Physics*, 24(7):4437–4454, 2022. DOI 10.1039/d1cp02036f.

[234] P. J. Ollitrault, S. Jandura, A. Miessen, I. Burghardt, R. Martinazzo, F. Tacchino, and I. Tavernelli. Quantum algorithms for grid-based variational time evolution, 2023. arXiv 2203.02521.

[235] C. Moler. Matrix computation on distributed memory multiprocessors. *Hypercube Multiprocessors*, 86(181-195):31, 1986.

[236] A. Bhattacharyya. On a Measure of Divergence between Two Multinomial Populations. *Sankhyā*, 7(4):401–406, 1946. URL https://www.jstor.org/stable/25047882.

[237] A. Richards. University Of Oxford Advanced Research Computing, 2015. DOI 10.5281/zenodo.22558.

[238] M. L. Hause, Y. H. Yoon, and F. F. Crim. Vibrationally mediated photodissociation of ammonia: The influence of N–H stretching vibrations on passage through conical intersections. *The Journal of Chemical Physics*, 125(17), November 2006. DOI 10.1063/1.2363192.

[239] D. M. Lemal. Perspective on Fluorocarbon Chemistry. *The Journal of Organic Chemistry*, 69(1):1–11, January 2004. DOI 10.1021/jo0302556.

[240] H. J. Silverstone. Long-range behavior of electronic wave functions. Generalized Carlson-Keller expansion. *Phys. Rev. A*, 23:1030–1037, March 1981. DOI 10.1103/PhysRevA.23.1030.

[241] D. Ghosh and R. Biswas. Theoretical Calculation of Absolute Radii of Atoms and Ions. Part 1. The Atomic Radii. *International Journal of Molecular Sciences*, 3(2): 87–113, February 2002. DOI 10.3390/i3020087.

[242] IBM Quantum. IBM Osprey r1 (IBM_SEATTLE) providing 433 qubits, November 2022. URL `https://quantum-computing.ibm.com/services/resources`.

[243] H. H. S. Chan, R. Meister, M. L. Goh, and B. Koczor. Algorithmic Shadow Spectroscopy, 2023. arXiv 2212.11036.

[244] H. Y. Huang, R. Kueng, and J. Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, June 2020. DOI 10.1038/s41567-020-0932-7.

[245] H. Y. Huang, R. Kueng, and J. Preskill. Efficient Estimation of Pauli Observables by Derandomization. *Phys. Rev. Lett.*, 127:030503, Jul 2021. DOI 10.1103/PhysRevLett.127.030503.

[246] K. Wan, W. J. Huggins, J. Lee, and R. Babbush. Matchgate Shadows for Fermionic Quantum Simulation, 2023. arXiv 2207.13723.

[247] A. Zhao, N. C. Rubin, and A. Miyake. Fermionic Partial Tomography via Classical Shadows. *Phys. Rev. Lett.*, 127:110504, Sep 2021. DOI 10.1103/PhysRevLett.127.110504.

[248] A. A. Akhtar, H. Y. Hu, and Y. Z. You. Scalable and Flexible Classical Shadow Tomography with Tensor Networks. *Quantum*, 7:1026, June 2023. DOI 10.22331/q-2023-06-01-1026.

[249] C. Bertoni, J. Haferkamp, M. Hinsche, M. Ioannou, J. Eisert, and H. Pashayan. Shallow shadows: Expectation estimation using low-depth random Clifford circuits, 2023. arXiv 2209.12924.

[250] J. Perktold, S. Seabold, J. Taylor, and Statsmodels Developers. Ljung-Box test of autocorrelation in residuals. URL `https://www.statsmodels.org/stable/stats.html`.

[251] S. Seabold and J. Perktold. Statsmodels: Econometric and statistical modeling with Python. In *9th Python in Science Conference*, 2010. DOI 10.25080/Majora-92bf1922-011.

[252] G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, August 1978. DOI 10.1093/biomet/65.2.297.

[253] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, April 2016. DOI 10.1098/rsta.2015.0202.

[254] D. Sundararajan. *The Discrete Fourier Transform: Theory, Algorithms and Applications.* World Scientific Publishing, April 2001. ISBN 978-981-02-4521-4.

[255] G. Rendon, J. Watkins, and N. Wiebe. Improved Accuracy for Trotter Simulations Using Chebyshev Interpolation, 2023. arXiv 2212.14144.

[256] Z. Cai. Resource Estimation for Quantum Variational Simulations of the Hubbard Model. *Phys. Rev. Appl.*, 14:014059, July 2020. DOI 10.1103/PhysRevApplied.14.014059.

[257] H. Jnane, B. Undseth, Z. Cai, S. C. Benjamin, and B. Koczor. Multicore Quantum Computing. *Phys. Rev. Appl.*, 18:044064, October 2022. DOI 10.1103/PhysRevApplied.18.044064.

[258] F. Verstraete, J. I. Cirac, and J. I. Latorre. Quantum circuits for strongly correlated quantum systems. *Phys. Rev. A*, 79:032316, March 2009. DOI 10.1103/PhysRevA.79.032316.

[259] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K. L. Chan, and R. Babbush. Quantum Simulation of Electronic Structure with Linear Depth and Connectivity. *Phys. Rev. Lett.*, 120:110501, March 2018. DOI 10.1103/PhysRevLett.120.110501.

[260] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien. Quantum Error Mitigation, 2023. arXiv 2210.00921.

[261] B. Koczor. The dominant eigenvector of a noisy quantum state. *New Journal of Physics*, 23(12):123047, December 2021. DOI 10.1088/1367-2630/ac37ae.

[262] C. Kokail, R. van Bijnen, A. Elben, B. Vermersch, and P. Zoller. Entanglement Hamiltonian tomography in quantum simulation. *Nature Physics*, 17(8):936–942, June 2021. DOI 10.1038/s41567-021-01260-w.

[263] T. V. Zache, C. Kokail, B. Sundar, and P. Zoller. Entanglement Spectroscopy and probing the Li-Haldane Conjecture in Topological Quantum Matter. *Quantum*, 6:702, April 2022. DOI 10.22331/q-2022-04-27-702.

[264] R. Meister and S. C. Benjamin. Resource-frugal Hamiltonian eigenstate preparation via repeated quantum phase estimation measurements, 2022. arXiv 2212.00846.

[265] Y. Dong, L. Lin, and Y. Tong. Ground-State Preparation and Energy Estimation on Early Fault-Tolerant Quantum Computers via Quantum Eigenvalue Transformation of Unitary Matrices. *PRX Quantum*, 3(4), October 2022. DOI 10.1103/prxquantum.3.040305.

[266] A. D. Córcoles, M. Takita, K. Inoue, S. Lekuch, Z. K. Minev, J. M. Chow, and J. M. Gambetta. Exploiting Dynamic Quantum Circuits in a Quantum Algorithm with Superconducting Qubits. *Phys. Rev. Lett.*, 127:100501, August 2021. DOI 10.1103/PhysRevLett.127.100501.

[267] M. Bee-Lindgren, Z. Qian, M. DeCross, N. C. Brown, C. N. Gilbreth, J. Watkins, X. Zhang, and D. Lee. Rodeo Algorithm with Controlled Reversal Gates, 2022. arXiv 2208.13557.

[268] L. Zhao, C. A. Pérez-Delgado, S. C. Benjamin, and J. F. Fitzsimons. Measurement-driven analog of adiabatic quantum computation for frustration-free Hamiltonians. *Phys. Rev. A*, 100:032331, September 2019. DOI 10.1103/PhysRevA.100.032331.

[269] R. Takagi, H. Tajima, and M. Gu. Universal Sampling Lower Bounds for Quantum Error Mitigation. *Phys. Rev. Lett.*, 131:210602, November 2023. DOI 10.1103/PhysRevLett.131.210602.

[270] K. Tsubouchi, T. Sagawa, and N. Yoshioka. Universal cost bound of quantum error mitigation based on quantum estimation theory, 2023. arXiv 2208.09385.

[271] *The Python Standard Library*. Python Software Foundation, September 2023. URL `https://docs.python.org/3/library/`.

Appendices

A
Algorithmic details of generic circuit synthesis

A.1 Global hyperparameters

In order to make the algorithms easier to read, some hyperparameters for generic circuit synthesis are globally defined in Table A.1. They stay constant during the whole synthesis process and can be used to tweak some properties of the algorithms.

Table A.1: Hyperparameters used in the pseudocode of our subroutines, collected here for more concise descriptions of the algorithms.

Param	Default	Use
$\tilde{H}$	H_{sum}	The synthesis Hamiltonian in the augmented space. In this work either H_{sum} or H_{proj}.
δ_{abs}	10^{-5}	Threshold for the absolute energy change per iteration regarded as constant during the parameter optimisation.
δ_{rel}	10^{-3}	Limit for the relative energy change per iteration considered constant during the parameter optimisation.
$k_{\mathrm{max,opt}}$	500	Maximum number of iterations for parameter optimisation.
n_{conv}	5	Number of times the convergence criterion must be fulfilled in parameter optimisation to be considered converged.
κ	1.4	Factor by which the step size λ of the parameter optimisation is in- or decreased while searching along the gradient direction.
λ_0	0.05	Initial step size in the parameter optimisation routine.
k_{max}	10 000	Maximum number of iterations for circuit modifications.
E_{conv}	10^{-8}	Energy at which the calculation is considered converged.
N_{moves}	30	Number of gates added in a single circuit modification iteration for random and tabu search.
N_{samp}	10	Number of times a circuit is modified in a single iteration until the best result is picked for the next iteration in random search.
$\mathcal{L}$	$\mathcal{L}_{\mathrm{allrot}}$	The library to draw new gates from.
$\varepsilon_{\mathrm{QMT}}$	10^{-3}	Small quantity to detect linearly dependent rows in the QMT.
$\varepsilon_{\mathrm{param}}$	$\varepsilon_{\mathrm{remove}}$	Threshold for the magnitude of parameters below which the corresponding gate is removed.
$\varepsilon_{\mathrm{remove}}$	$\frac{E_k - E_{k-1}}{50}$	Energy increase considered acceptable for the removal of unnecessary gates.

A.2 Pseudocode

To minimise clutter in the chapter text, I collect pseudocode for most of the described routines in this appendix for completeness. In contrast to the chapter text, where the cost function is written as $\langle\tilde{H}\rangle$, in pseudocode to explicitly denote which circuit is applied, I use the notation

$$\mathcal{E}(\mathcal{C}(\underline{\theta})) = \langle\psi_1|\tilde{H}|\psi_1\rangle \tag{A.1}$$

with

$$|\psi_1\rangle = P^\dagger \mathcal{C}^\dagger(\underline{\theta}) U P(|0\rangle_{\mathcal{H}} \otimes |0\rangle_{\mathcal{H}'}) \tag{A.2}$$

where operator P prepares Bell pairs as depicted on the left side of Fig. 3.2.

```
function OPTIMISEPARAMETERS($\mathcal{C}$, $\underline{\theta}$)
    $E_0 \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta}))$,   $k_{\text{conv}} \leftarrow 0$
    for $k \leftarrow 1 .. k_{\text{max,opt}}$ do
        $A_{ij} \leftarrow \text{Re}\big(\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle\big)\ \forall\, i, j$
        $B_i \leftarrow -\langle\partial_i\psi|\hat{H}|\psi\rangle\ \forall\, i$
        $\underline{\Delta} \leftarrow$ REGULARISE($\mathbf{A}$)$^{-1}\, \underline{B}$
        $\lambda \leftarrow$ CHOOSELAMBDA($\lambda_0$,$\mathcal{C}$,$\underline{\theta}$,$\underline{\Delta}$)
        $\underline{\theta} \leftarrow \underline{\theta} - \lambda\underline{\Delta}$
        $E_k \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta}))$
        if $E_k - E_{k-1} < \delta_{\text{abs}}$ or $\frac{E_k - E_{k-1}}{E_k} < \delta_{\text{rel}}$ then
            $k_{\text{conv}} \leftarrow k_{\text{conv}} + 1$
        else
            $k_{\text{conv}} \leftarrow 0$
        if $k_{\text{conv}} = n_{\text{conv}}$ then return $\underline{\theta}$
    return fail                                    ▷ No convergence at max iterations

function CHOOSELAMBDA($\lambda$, $\mathcal{C}$, $\underline{\theta}$, $\underline{\Delta}$)
    $\lambda^- \leftarrow \lambda/\kappa$,   $\lambda^+ \leftarrow \lambda\cdot\kappa$,   $E \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta} - \lambda\underline{\Delta}))$
    $E^+ \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta} - \lambda^+\underline{\Delta}))$,   $E^- \leftarrow \mathcal{E}(\mathcal{C}(\underline{\theta} - \lambda^-\underline{\Delta}))$
    if $E^- > E < E^+$ or $\lambda \leq \lambda_{\min}$ then
        return $\lambda$                           ▷ no improvement either side
    if $E^+ < E^-$ then
        return CHOOSELAMBDA($\lambda^+$,$\mathcal{C}$,$\theta$,$\underline{\Delta}$)      ▷ grow $\lambda$
    return CHOOSELAMBDA($\lambda^-$,$\mathcal{C}$,$\theta$,$\underline{\Delta}$)          ▷ shrink $\lambda$
```

Algorithm 1: The routine used to optimise the parameters $\underline{\theta}$ for a given circuit structure $\mathcal{C}$.

```
function GenerateMoves($\mathcal{C}$)
    $\mathcal{M} \leftarrow \{\}$
    for $k \leftarrow 1 \mathinner{..} N_{\text{qubits}}$ do
        $\triangleright$ Indices of gates touching qubit $k$
        $\tilde{N} \leftarrow (p \,|\, \mathcal{C}_p$ acts on qubit $k)$
        $G_{\text{left}} \leftarrow \mathbb{1}$
        for all $m \in \tilde{N}$ do
            $\mathcal{M} \leftarrow \mathcal{M} \cup \{(G, m) \,|\, G \in \mathcal{L}$ and $G \neq G_{\text{left}}$ and $G \neq \mathcal{C}_m\}$
            $G_{\text{left}} \leftarrow \mathcal{C}_m$
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{(G, \max(\tilde{N}) + 1) \,|\, G \in \mathcal{L}$ and $G \neq G_{\text{left}}\}$
    return $\mathcal{M}$

function ApplyMove($\mathcal{C}$, $\underline{\theta}$, $M$)
    $\mathcal{C} \leftarrow (\mathcal{C}_0, \ldots, \mathcal{C}_{M_n - 1}, M_G, \mathcal{C}_{M_n}, \ldots)$
    $\underline{\theta} \leftarrow (\theta_0, \ldots \theta_{M_n - 1}, 0, \theta_{M_n}, \ldots)$
    return $\mathcal{C}, \underline{\theta}$
```

Algorithm 2: Routine GenerateMoves to generate all moves applicable to a circuit structure $\mathcal{C}$ not leading to obvious redundancies in the resulting circuit. The helper function ApplyMove applies the move M to the circuit structure $\mathcal{C}$ and parameter vector $\underline{\theta}$, and makes other routines easier to read. The routine Regularise is an arbitrary regularisation method to make the matrix inversion numerically more stable.

```
function RandomSearch($\mathcal{C}^{(0)}, \underline{\theta}^{(0)}$)
    $E_0 \leftarrow \mathcal{E}(\mathcal{C}^{(0)}(\underline{\theta}^{(0)}))$
    for $k \leftarrow 1 .. k_{\max}$ do
        ▷ $\mathcal{R}$ and $\tilde{\mathcal{R}}$ contain indices of newly added gates
        $E_k \leftarrow E_{k-1}, \quad \mathcal{R} \leftarrow \{\}$
        $\mathcal{C}^{(k)} \leftarrow \mathcal{C}^{(k-1)}, \quad \underline{\theta}^{(k)} \leftarrow \underline{\theta}^{(k-1)}$
        for $m \leftarrow 1 .. N_{\text{samp}}$ do
            $\tilde{\mathcal{C}} \leftarrow \mathcal{C}^{(k-1)}, \quad \underline{\tilde{\theta}} \leftarrow \underline{\theta}^{(k-1)}, \quad \tilde{\mathcal{R}} \leftarrow \{\}$
            for $n \leftarrow 1 .. N_{\text{moves}}$ do
                $M \leftarrow$ random element
                        of GenerateMoves($\tilde{\mathcal{C}}$)
                $\tilde{\mathcal{C}}, \underline{\tilde{\theta}} \leftarrow$ ApplyMove($\tilde{\mathcal{C}}, \underline{\tilde{\theta}}, M$)
                $\tilde{\mathcal{R}} \leftarrow \{j \mid j \in \tilde{\mathcal{R}} \text{ and } j < M_n\} \cup \{M_n\} \cup \{j+1 \mid j \in \tilde{\mathcal{R}} \text{ and } j \geq M_n\}$
            $\underline{\tilde{\theta}} \leftarrow$ OptimiseParameters($\tilde{\mathcal{C}}, \underline{\tilde{\theta}}$)
            if $\mathcal{E}(\tilde{\mathcal{C}}(\underline{\tilde{\theta}})) < E_k$ then
                $\mathcal{C}^{(k)} \leftarrow \tilde{\mathcal{C}}, \quad \underline{\theta}^{(k)} \leftarrow \underline{\tilde{\theta}}$
                $E_k \leftarrow \mathcal{E}(\tilde{\mathcal{C}}(\underline{\tilde{\theta}})), \quad \mathcal{R} \leftarrow \tilde{\mathcal{R}}$
        $\mathcal{C}^{(k)}, \underline{\theta}^{(k)} \leftarrow$ Prune($\mathcal{C}^{(k)}, \underline{\theta}^{(k)}, \mathcal{R}$)
        $E_k \leftarrow \mathcal{E}(\mathcal{C}^{(k)}(\underline{\theta}^{(k)}))$
        if $E_k < E_{\text{conv}}$ then
            return Prune($\mathcal{C}^{(k)}, \underline{\theta}^{(k)}, \{0, \ldots, N_{\text{gates}}(\mathcal{C}^{(k)})\}$)
    return fail                                    ▷ No convergence at iteration limit
```

Algorithm 3: A simplified version of the random search algorithm used to generate the results. It calls several subroutines from Algorithms 1, 2 and 4.

```
function PRUNE($\mathcal{C}$, $\underline{\theta}$, $\mathcal{R}$)
    ▷ Remove vanishing parameters
    $\mathcal{D} \leftarrow \{k \mid |\theta_k \bmod 2\pi| < \varepsilon_{\text{param}}\}$
    $\mathcal{C}, \underline{\theta}, \mathcal{R} \leftarrow$ DELETE($\mathcal{C}, \underline{\theta}, \mathcal{R}, \mathcal{D}$)
    ▷ Quantum metric tensor assisted removal
    $\mathcal{D} \leftarrow \{\}, \quad \mathcal{K} \leftarrow \{\}, \quad \underline{\theta}^{+} \leftarrow \underline{0}$
    $A_{ij} \leftarrow \mathrm{Re}\left(\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle\right) \ \forall\, i,j$
    for $k \in \mathcal{R}$ do
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{n \mid n > k$ and $|(A_{k,\cdot}^{\top} \cdot A_{n,\cdot}) - \|A_{k,\cdot}\|\,\|A_{n,\cdot}\|\,| < \varepsilon_{\text{QMT}}\}$
        for all $n \in \mathcal{D}$ do
            $\mathcal{C}', \underline{\theta}' \leftarrow$ DELETE($\mathcal{C}, \underline{\theta}, \{\}, \{n\}$)
            $\theta'_k \leftarrow \theta_k + \theta_n$
            if $\mathcal{E}(\mathcal{C}'(\underline{\theta}')) > \mathcal{E}(\mathcal{C}(\theta)) + \varepsilon_{\text{remove}}$ then
                $\mathcal{K} \leftarrow \mathcal{K} \cup \{n\}$
            else
                $\theta^{+}_k \leftarrow \theta^{+}_k + \theta_n$
    $\underline{\theta} \leftarrow \underline{\theta} + \underline{\theta}^{+}$
    $\mathcal{C}, \underline{\theta}, \mathcal{R} \leftarrow$ DELETE($\mathcal{C}, \underline{\theta}, \mathcal{R}, \mathcal{D} \setminus \mathcal{K}$)
    ▷ Trial and error removal
    while $\mathcal{R} \neq \{\}$ do
        $k \leftarrow \max(\mathcal{R})$
        $\mathcal{R} \leftarrow \mathcal{R} \setminus \{k\}$
        $\mathcal{C}' \leftarrow (\mathcal{C}_0, \ldots \mathcal{C}_{k-1}, \mathcal{C}_{k+1}, \ldots)$
        $\underline{\theta}' \leftarrow (\theta_0, \ldots \theta_{k-1}, \theta_{k+1}, \ldots)$
        $\underline{\theta}' \leftarrow$ OPTIMISEPARAMETERS($\mathcal{C}', \underline{\theta}'$)
        if $\mathcal{E}(\mathcal{C}'(\underline{\theta}')) < \mathcal{E}(\mathcal{C}(\underline{\theta})) + \varepsilon_{\text{remove}}$ then
            $\mathcal{C} \leftarrow \mathcal{C}'$
            $\underline{\theta} \leftarrow \underline{\theta}'$
    return $\mathcal{C}, \underline{\theta}$

function DELETE($\mathcal{C}$, $\underline{\theta}$, $\mathcal{R}$, $\mathcal{D}$)
    for $d \leftarrow$ INVERSESORTED($\mathcal{D}$) do
        $\mathcal{C} \leftarrow (\mathcal{C}_0, \ldots \mathcal{C}_{d-1}, \mathcal{C}_{d+1}, \ldots)$
        $\underline{\theta} \leftarrow (\theta_0, \ldots \theta_{d-1}, \theta_{d+1}, \ldots)$
        $\mathcal{R} \leftarrow \{n \mid n \in \mathcal{R}$ and $n < d\} \cup \{n-1 \mid n \in \mathcal{R}$ and $n > d\}$
    return $\mathcal{C}, \underline{\theta}, \mathcal{R}$
```

Algorithm 4: The routine to detect and remove unnecessary gates from a circuit. $\mathcal{C}$ and $\underline{\theta}$ are the circuit structure and current parameters, respectively, and $\mathcal{R}$ is a set of indices of gates that should be considered for deletion. The helper function DELETE removes the gates at the indices specified in $\mathcal{D}$ from the circuit and updates the indices in $\mathcal{R}$ accordingly.

```
function HillClimb($\mathcal{C}^{(0)}, \underline{\theta}^{(0)}$)
    $E_0 \leftarrow \mathcal{E}(\mathcal{C}^{(0)}(\underline{\theta}^{(0)}))$
    for $k \leftarrow 1 \,..\, k_{\text{max}}$ do
        $E_k \leftarrow E_{k-1}$
        for all $M \in$ GenerateMoves($\mathcal{C}^{(k-1)}$) do
            $\tilde{\mathcal{C}}, \tilde{\underline{\theta}} \leftarrow$ ApplyMove($\mathcal{C}^{(k-1)}, \underline{\theta}^{(k-1)}, M$)
            $\tilde{\underline{\theta}} \leftarrow$ OptimiseParameters($\tilde{\mathcal{C}}, \tilde{\underline{\theta}}$)
            if $\mathcal{E}(\tilde{\mathcal{C}}(\tilde{\underline{\theta}})) < E_k$ then
                $\mathcal{C}^{(k)} \leftarrow \tilde{\mathcal{C}}, \quad \underline{\theta}^{(k)} \leftarrow \tilde{\underline{\theta}}, \quad E_k \leftarrow \mathcal{E}(\tilde{\mathcal{C}}(\tilde{\underline{\theta}}))$
        if $E_k < E_{\text{conv}}$ then
            return Prune($\mathcal{C}^{(k)}, \underline{\theta}^{(k)}, \{0, \ldots, N_{\text{gates}}(\mathcal{C}^{(k)})\}$)
        if $E_k = E_{k-1}$ then
            return fail                                    ▷ No more improvement
    return fail                                ▷ No convergence at iteration limit
```

Algorithm 5: A simple variant of hill climbing for circuit synthesis.

A.3 Gate counts

This appendix collects the results exhibiting the smallest number of two-qubit gates for the QFT and Toffoli syntheses across all *random search* and *tabu search* calculations. The used gate sets are those given in the chapter text, with added parametrised SWAP ($\mathcal{S}_{i,j}$) gates for the substrate.

The mean compile time $\overline{T}$ is reported for execution on 8 cores ($n_{\text{qb}} \leq 6$) or 48 cores ($n_{\text{qb}} > 6$) of Intel Platinum 8628 CPUs within the University of Oxford Advanced Research Computing facility.

Table A.2: Details of results with the least number of two-qubit gates for the QFT and Toffoli circuits discussed in the chapter text. The number of two-qubit gates (N_{2qb}) is the lowest found in any calculation for the given circuit and gate set, N_{gates} is the corresponding number of total gates in those circuits. The mean compile time $\overline{T}$ is averaged over all calculations with the specified parameters, including ones that terminated due to a set time limit.

Circuit	n_{qb}	Gate set	N_{2qb}	N_{gates}	$\overline{T}$ [s]
QFT	3	allrot	6	14	103
QFT	3	NNrot	6	12	174
QFT	3	SWAP	6	20	829
QFT	4	allrot	12	22	342
QFT	4	NNrot	12	24	1459
QFT	4	SWAP	13	132	6228
QFT	5	allrot	20	37	9966
QFT	5	NNrot	20	35	39709
QFT	5	SWAP	30	128	116782
QFT	6	allrot	29	46	53235
QFT	6	NNrot	30	77	183580
QFT	6	SWAP	92	123	239180
QFT	7	allrot	44	72	126094
QFT	7	NNrot	46	209	248486
QFT	8	allrot	60	418	263227
QFT	8	NNrot	77	115	345283
QFT	9	allrot	82	259	317498
QFT	10	allrot	162	429	431999
Toffoli	3	allrot	5	6	26
Toffoli	4	allrot	13	20	1151
Toffoli	5	allrot	33	167	96868

B

Proofs for the truncated Taylor series method

For completeness and convenience, I collect several of the results used in Section 4.1 in this appendix, including some that may be well known.

Lemma 1. *The optimal choice for the number of amplification steps is* $\nu = 1$*, resulting in* $s_{\underline{L}} = 2$.

Proof. For unitary $U_{\underline{L}}$, the operator $\mathcal{Q}^\nu \mathcal{W}$, with $\mathcal{Q}$ as defined in Eq. (4.11), would have the effect [133]

$$\begin{aligned}\mathcal{Q}^\nu \mathcal{W} |0\rangle\, |\psi\rangle &= \sin\left[(2\nu+1)\sin^{-1}(s_{\underline{L}}^{-1})\right] |0\rangle\, U_{\underline{L}} |\psi\rangle \\ &\quad + \cos\left[(2\nu+1)\cos^{-1}(N)\right] |0^\perp, \Phi\rangle\,.\end{aligned}$$

For any given number of amplification steps ν, the amplitude of the desired state $|0\rangle\, U_{\underline{L}} |\psi\rangle$ can be tuned to 1 by setting t such that $s_{\underline{L}}$ fulfills

$$s_{\underline{L}} = \sin\left(\frac{\pi}{4\nu+2}\right)^{-1} \sim \frac{4\nu+2}{\pi}. \tag{B.1}$$

For this argument it is sufficient to analyse the full expansion to order n. To find the optimal number ν, consider the operator $\mathcal{Q}^\nu \mathcal{W}$, which contains the most expensive component $\mathcal{W}$ a total of $2\nu+1$ times. Therefore, the cost is approximately linear in ν, meaning it is also linear in s_n. However, it is known that

$$s_n = \sum_{k=0}^{n} \frac{t^k}{k!} \underbrace{\left(\sum_{\ell=0}^{L-1} \alpha_\ell\right)}_{:=\Lambda}^{k} = \sum_{k=0}^{n} \frac{t^k \Lambda^k}{k!} \approx e^{\Lambda t}, \tag{B.2}$$

where the rightmost approximation holds for sufficiently large n. Equations (B.1) and (B.2) imply that the cost is exponential in t, indicating there is no benefit in amplifying more than once. Exact numerical evaluation shows that for $n \to \infty$, one or two amplification steps ($\nu \in \{1, 2\}$) yield approximately equivalent time-per-gate, but the smaller t of $v = 1$ leads to quicker convergence in n. Consequently, it is best to choose $v = 1$. This choice forces $s_{\underline{L}}$ to satisfy

$$s_{\underline{L}} = \sin\left(\frac{\pi}{6}\right)^{-1} = 2.$$

□

Lemma 2. *The action of $\Pi\mathcal{A}$ on a product state $|0\rangle\,|\psi\rangle$ is given by [132]*

$$\Pi\mathcal{A}\,|0\rangle\,|\psi\rangle = |0\rangle\left(-\frac{4}{s_{\underline{L}}^3}U_{\underline{L}}U_{\underline{L}}^\dagger U_{\underline{L}} + \frac{3}{s_{\underline{L}}}U_{\underline{L}}\right)|\psi\rangle\,.$$

Proof. The operator R can be explicitly expanded into projectors. Using $\Pi^2 = \Pi$ as well as $\Pi\,|0\rangle\,\big|\psi\big\rangle = |0\rangle\,\big|\psi\big\rangle$ gives

$$\begin{aligned}
\Pi\mathcal{A}\,|0\rangle\,|\psi\rangle &= -\Pi\mathcal{W}R\mathcal{W}^\dagger R\mathcal{W}\,|0\rangle\,|\psi\rangle \\
&= -\Pi\mathcal{W}(2\Pi-\mathbb{1})\mathcal{W}^\dagger(2\Pi-\mathbb{1})\mathcal{W}\,|0\rangle\,|\psi\rangle \\
&= (-4\Pi\mathcal{W}\Pi\mathcal{W}^\dagger\Pi\mathcal{W} + 2\Pi\mathcal{W}\mathcal{W}^\dagger\Pi\mathcal{W} + 2\Pi\mathcal{W}\Pi\mathcal{W}^\dagger\mathcal{W} - \Pi\mathcal{W}\mathcal{W}^\dagger\mathcal{W})\,|0\rangle\,|\psi\rangle \\
&= (-4\Pi\mathcal{W}\Pi\mathcal{W}^\dagger\Pi\mathcal{W} + 3\Pi\mathcal{W})\,|0\rangle\,|\psi\rangle \\
&= (-4\underbrace{\Pi\mathcal{W}\Pi}_{\frac{1}{s_{\underline{L}}}(|0\rangle\langle 0|\otimes U_{\underline{L}})}\Pi\mathcal{W}^\dagger\Pi\Pi\mathcal{W}\Pi + 3\Pi\mathcal{W}\Pi)\,|0\rangle\,|\psi\rangle \\
&= |0\rangle\left(-\frac{4}{s_{\underline{L}}^3}U_{\underline{L}}U_{\underline{L}}^\dagger U_{\underline{L}} + \frac{3}{s_{\underline{L}}}U_{\underline{L}}\right)|\psi\rangle
\end{aligned}$$

as claimed. □

Lemma 3. *If used in $\mathcal{W}\,|0\rangle\,|\psi\rangle$, $\mathcal{P}^\star$ has the same effect as $\mathcal{P}$, explicitly $\mathcal{W}\,|0\rangle\,|\psi\rangle = (\mathcal{P}^\dagger\otimes\mathbb{1})\,\mathcal{S}\,(\mathcal{P}\otimes\mathbb{1})\,|0\rangle\,|\psi\rangle = (\mathcal{P}^{\star\dagger}\otimes\mathbb{1})\,\mathcal{S}\,(\mathcal{P}^\star\otimes\mathbb{1})\,|0\rangle\,|\psi\rangle$*

Proof. $\mathcal{P}^\star$ on any of the c_k registers has the action

$$\mathcal{P}^\star\,|0\rangle_{c_k} = \frac{1}{\sqrt{\Lambda_k}}\sum_{\ell=0}^{L_k-1}\sqrt{\alpha_\ell}\,|\ell\rangle_{c_k}$$

and on the q register

$$\mathcal{P}^\star\,|0\rangle_q = \frac{1}{\sqrt{N_q}}\sum_{k=0}^{\kappa}\sqrt{\frac{t^k}{k!}\prod_{j=1}^{k}\Lambda_j}\,|k\rangle_q$$

where $N_q = \sum_{k=0}^{\infty}\frac{t^k}{k!}\prod_{j=1}^{k}\Lambda_j$ and κ is the largest nonzero index in $\underline{L}$. Therefore

$$\mathcal{S}\,(\mathcal{P}^\star\otimes\mathbb{1})\,|0\rangle\,|\psi\rangle = \frac{1}{\sqrt{N_q\prod_{k=1}^{\kappa}\Lambda_k}}\sum_{k=0}^{\kappa}\sqrt{\left(\prod_{j=1}^{k}\Lambda_j\right)\frac{t^k}{k!}}\,|k\rangle_q\bigotimes_{j=1}^{k}\left(\sum_{\ell=0}^{L_j-1}|\ell\rangle_{c_j}\sqrt{\alpha_\ell}\tilde{h}_\ell\right)$$
$$\bigotimes_{j=k+1}^{\kappa}\left(\sum_{\ell=0}^{L_j-1}|\ell\rangle_{c_j}\sqrt{\alpha_\ell}\right)|\psi\rangle$$

Transforming back with $\mathcal{P}^{\star\dagger}$ and projecting onto the ancilla $|0\rangle$ yields

$$\begin{aligned}
\Pi(\mathcal{P}^{\star\dagger}\otimes\mathbb{1})\,\mathcal{S}\,(\mathcal{P}^{\star}\otimes\mathbb{1}) &= \frac{\Pi}{N_q\prod_{k=1}^{\kappa}\Lambda_k}\sum_{k=0}^{\kappa}\left(\prod_{j=1}^{k}\Lambda_j\right)\frac{t^k}{k!}\,|0\rangle_q\bigotimes_{j=1}^{k}\left(\sum_{\ell=0}^{L_j-1}|0\rangle_{c_j}\,\alpha_\ell\tilde{h}_\ell\right)\\
&\qquad\bigotimes_{j=k+1}^{\kappa}\underbrace{\left(\sum_{\ell=0}^{L_j-1}|0\rangle_{c_j}\,\alpha_\ell\right)}_{\Lambda_j|0\rangle_{c_j}}|\psi\rangle\\
&= |0\rangle\,\frac{1}{N_q\prod_{k=1}^{\kappa}\Lambda_k}\sum_{k=0}^{\kappa}\left(\prod_{j=1}^{\kappa}\Lambda_j\right)\frac{t^k}{k!}\prod_{j=1}^{k}\left(\sum_{\ell=0}^{L_j-1}\alpha_\ell\tilde{h}_\ell\right)|\psi\rangle\\
&= \frac{1}{N_q}\,|0\rangle\sum_{k=0}^{\kappa}\frac{t^k}{k!}\prod_{j=1}^{k}\left(\sum_{\ell=0}^{L_j-1}\alpha_\ell\tilde{h}_\ell\right)|\psi\rangle = \frac{1}{N_q}\,|0\rangle\,U_{\underline{L}}|\psi\rangle = \Pi\mathcal{W}\,|0\rangle\,|\psi\rangle
\end{aligned}$$

which is Eq. (4.10) and it is obvious that $N_q = s_{\underline{L}}$ as defined in Eq. (4.14). □

Lemma 4. *The error of a single time step $\delta_{\underline{L}}$ when using $t_\infty = \log(2)/\Lambda$ can be bounded by*

$$\delta_{\underline{L}}(t_\infty) \le 2 - s_{\underline{L}}(t_\infty) =: \varepsilon_{\underline{L}}.$$

Proof. For easier notation, first consider fully expanded orders and define

$$\tilde{\Delta}_n(t) := U_n(t) - U(t)$$

$$\varepsilon_n := s_\infty(t_\infty) - s_n(t_\infty) = 2 - s_n(t_\infty)$$

and observe that

$$\begin{aligned}
&\|\tilde{\Delta}_n(t)\|\\
&\quad= \left\|\sum_{k=1}^{n}\frac{(-it)^k}{k!}\sum_{\ell_1,\dots,\ell_k=0}^{L-1}\alpha_{\ell_1}\dots\alpha_{\ell_k}h_{\ell_1}\dots h_{\ell_k} - \sum_{k=1}^{\infty}\frac{(-it)^k}{k!}\sum_{\ell_1,\dots,\ell_k}^{L}\alpha_{\ell_1}\dots\alpha_{\ell_k}h_{\ell_1}\dots h_{\ell_k}\right\|\\
&\quad= \left\|\sum_{k=n+1}^{\infty}\frac{(-it)^k}{k!}\sum_{\ell_1,\dots,\ell_k=0}^{L-1}\alpha_{\ell_1}\dots\alpha_{\ell_k}h_{\ell_1}\dots h_{\ell_k}\right\|\\
&\quad\le \sum_{k=n+1}^{\infty}\frac{t^k}{k!}\sum_{\ell_1,\dots,\ell_k=0}^{L-1}\alpha_{\ell_1}\dots\alpha_{\ell_k}\underbrace{\|h_{\ell_1}\|\dots\|h_{\ell_k}\|}_{1}\\
&\quad= \left(1+\sum_{k=1}^{\infty}\frac{t^k}{k!}\sum_{\ell_1,\dots,\ell_k=0}^{L-1}\alpha_{\ell_1}\dots\alpha_{\ell_k}\right) - \left(1+\sum_{k=1}^{n}\frac{t^k}{k!}\sum_{\ell_1,\dots,\ell_k=0}^{L-1}\alpha_{\ell_1}\dots\alpha_{\ell_k}\right)\\
&\quad= s_\infty(t) - s_n(t)
\end{aligned}$$

which means

$$\|\tilde{\Delta}_n(t_\infty)\| \le s_\infty(t_\infty) - s_n(t_\infty) = \varepsilon_n.$$

Using these in the definition of Δ_n yields

$$\begin{aligned}
-\Delta_n(t_\infty) &= \tilde{\mathcal{A}}_n(t_\infty) - U(t_\infty) \\
&= \frac{3}{s_n(t_\infty)} U_n(t_\infty) - \frac{4}{s_n^3(t_\infty)} U_n(t_\infty) U_n^\dagger(t_\infty) U_n(t_\infty) - U(t_\infty) \\
&= \underbrace{\frac{3}{2-\varepsilon_n}}_{=\frac{3}{2}+\frac{3\varepsilon_n}{4}+\mathcal{O}(\varepsilon_n^2)} (U+\tilde{\Delta}_n) - \underbrace{\frac{4}{(2-\varepsilon_n)^3}}_{=\frac{1}{2}+\frac{3\varepsilon_n}{4}+\mathcal{O}(\varepsilon_n^2)} \overbrace{(U+\tilde{\Delta}_n)(U+\tilde{\Delta}_n)^\dagger (U+\tilde{\Delta}_n)}^{=U+2\tilde{\Delta}_n+U\tilde{\Delta}_n^\dagger U+\mathcal{O}(\tilde{\Delta}_n^2)} - U \\
&= \tilde{\Delta}_n \left(\frac{1}{2} - \frac{3\varepsilon_n}{4}\right) - U\tilde{\Delta}_n^\dagger U \left(\frac{1}{2} + \frac{3\varepsilon_n}{4}\right) + \mathcal{O}(\tilde{\Delta}_n^2) + \mathcal{O}(\varepsilon_n^2).
\end{aligned}$$

Now the error can finally be bounded to

$$\begin{aligned}
\delta_n = \|\Delta_n(t_\infty)\| &\le \left\|\tilde{\Delta}_n \left(\frac{1}{2} - \frac{3\varepsilon_n}{4}\right)\right\| + \left\|U\tilde{\Delta}_n^\dagger U \left(\frac{1}{2} + \frac{3\varepsilon_n}{4}\right)\right\| + \mathcal{O}(\varepsilon_n^2) \\
&\le \frac{\varepsilon_n}{2} + \frac{\varepsilon_n}{2} + \mathcal{O}(\varepsilon_n^2) = \varepsilon_n + \mathcal{O}(\varepsilon_n^2),
\end{aligned}$$

which straightforwardly extends to $\delta_{\underline{L}}$ with $\varepsilon_{\underline{L}}$ for partial orders. □

Lemma 5. *The error of r time steps $\left\|U^r - \tilde{\mathcal{A}}_{\underline{L}}^r\right\|$ is bounded by r times the error of a single time step $\delta_{\underline{L}} = \|\Delta_{\underline{L}}\| = \left\|U - \tilde{\mathcal{A}}_{\underline{L}}\right\|$, up to order $\delta_{\underline{L}}$.*

Proof. Use the definition of $\Delta = U - \tilde{\mathcal{A}}$ and substitute for $\tilde{\mathcal{A}}$.

$$\begin{aligned}
\left\|U^r - \tilde{\mathcal{A}}_{\underline{L}}^r\right\| &= \left\|U^r - (U-\Delta)^r\right\| = \left\|U^r - U^r + \sum_{k=1}^{r} U^{k-1}\,\Delta\, U^{r-k} + \mathcal{O}(\Delta^2)\right\| \\
&\le \sum_{k=1}^{r} \left\|U^{k-1}\,\Delta\, U^{r-k}\right\| + \left\|\mathcal{O}(\Delta^2)\right\| \le \sum_{k=1}^{r} \|\Delta\| + \mathcal{O}(\delta^2) = r\delta + \mathcal{O}(\delta^2)
\end{aligned}$$

□

Lemma 6. *The logarithmic inverse error bound of a single time step for full orders* $\log(\varepsilon_n^{-1})$ *scales like*

$$\log\frac{1}{\varepsilon_n} = \mathcal{O}(n\log n)$$

Proof. The residual of the Taylor series can be bounded by

$$\begin{aligned}
\varepsilon_n &= \sum_{k=n+1}^{\infty} \frac{(\overbrace{t_\infty\Lambda}^{\log 2})^k}{k!} \\
&= \frac{\log^n 2}{n!} \sum_{k=1}^{\infty} \frac{n!\log^k 2}{(k+n)!} \\
&\le \frac{\log^n 2}{n!} e^{\log 2} = \frac{2\log^n 2}{n!}
\end{aligned}$$

and use Stirling's approximation $n! \leq e\, n^{n+1/2}\, e^{-n}$ yields

$$\varepsilon_n \leq \frac{2e^{-n}\log^n 2}{e\, n^{n+1/2}}$$

$$\log\varepsilon_n \leq \log 2 - n - n\log\log 2 - 1 - \left(n + \frac{1}{2}\right)\log n.$$

Therefore

$$\log\frac{1}{\varepsilon_n} = \mathcal{O}(n\log n).$$

□

Corollary 7. *Because the total complexity is of the order $C_n = nL$, the complexity when using full orders depending on the error bound ε, which will be called $C_{\varepsilon,\text{full}}$, scales like*

$$C_{\varepsilon,\text{full}} = \mathcal{O}\left(\frac{L\log\frac{1}{\varepsilon}}{\log\log\frac{1}{\varepsilon}}\right)$$

Proof. Use the inequality in Lemma 6

$$n < \log\frac{1}{\varepsilon_n}$$

to replace the n in the logarithm of Lemma 6 and find

$$n = \mathcal{O}\left(\frac{\log\frac{1}{\varepsilon_n}}{\log\log\frac{1}{\varepsilon_n}}\right)$$

and therefore

$$C_{\varepsilon,\text{full}} = \mathcal{O}\left(\frac{L\log\frac{1}{\varepsilon}}{\log\log\frac{1}{\varepsilon}}\right).$$

□

Lemma 8. *The bound for the logarithmic inverse error of the modified version* $\log\varepsilon_{\underline{L}}^{-1}$ *scales like* $\mathcal{O}\left(\frac{C_L}{L}\log\frac{C_L}{L}\right) \leq \log\varepsilon_{\underline{L}}^{-1} < \mathcal{O}(C_{\underline{L}}\log C_{\underline{L}})$*, depending on the Hamiltonian.*

Proof. The left inequality follows immediately from the worst case that $\alpha_0 = \alpha_1 = \ldots$ In this case the modification is equivalent to the original method and Lemma 6 with $n = C_{\underline{L}}/L$ holds.

In the other extreme case of $\frac{\alpha_\ell}{\alpha_0} \to 0\ \forall\, \ell \in \{1\ldots L\}$, one term dominates the whole Hamiltonian, and adding h_0 in some order of the expansion equates to adding that whole order, effectively reducing L to 1. Therefore Lemma 6 with $L = 1$ defines an upper bound for the error scaling. □

Corollary 9. *The complexity of the modified version depending on the simulation error bound ε, which will be called C_ε, is bounded by*

$$\mathcal{O}\left(\frac{\log\frac{1}{\varepsilon}}{\log\log\frac{1}{\varepsilon}}\right) < C_\varepsilon \leq \mathcal{O}\left(\frac{L\log\frac{1}{\varepsilon}}{\log\log\frac{1}{\varepsilon}}\right),$$

Proof. The reasoning in Lemma 7 also applies to the bounds in Lemma 8. □

Corollary 10. *The complexity of simulating for a time τ with a total error bound ϵ is bounded by*

$$\mathcal{O}\left(\frac{\Lambda\tau\log\frac{\Lambda\tau}{\epsilon}}{\log\log\frac{\Lambda\tau}{\epsilon}}\right) < C_\epsilon \leq \mathcal{O}\left(\frac{L\Lambda\tau\log\frac{\Lambda\tau}{\epsilon}}{\log\log\frac{\Lambda\tau}{\epsilon}}\right).$$

Proof. Simulating for a time $\tau = r t_\infty = r\log(2)/\Lambda$ requires r steps. The error of a single step ε must therefore be $\varepsilon = \epsilon/r$. Substituting this in Lemma 9 and multiplying by the number of steps $r = \mathcal{O}(\tau\Lambda)$ proves the claim. □