PROF. DR. MARIO BERTA

QUANTUM ALGORITHMS

RWTH AACHEN UNIVERSITY

Lecture notes, work in progress, feedback appreciated. Assistants: Gereon Kossmann (2025, 2024), Tobias Rippchen (2023)

https://marioberta.info/teaching/

Latest update May 6, 2025

Contents

1
Overview

1.1 Introduction

In this lecture, we will discuss what computational problems can potentially be solved in more efficient ways using specialized computing devices governed by the laws of quantum mechanics — compared to solely using computing devices that are based on the principles of classical physics. We will focus in particular on problems in scientific computing, such as from quantum many body physics and computational quantum chemistry.

Our considerations will be purely mathematical and as such also hardware agnostic (independent of the exacts physics of the underlying devices). In particular, we will assume that we have noise free (error corrected) qubits and quantum gates at hand, albeit potentially only a rather limited number thereof. This regime is often termed *early fault-tolerance (EFT)*, where the early part is emphasizing that the number of qubits available might be limited (or only available in a distributed fashion) and that the quantum clock speed might be slow, not allowing to run too deep quantum circuits.

We are then asking the question what, if any, problems from scientific computing can potentially be solved significantly more efficient by using EFT quantum technologies. Our threshold is thereby that any quantum algorithm should be at least super-quadratically faster than the corresponding state-of-the-art classical methods for the same end-to-end solution of a well-defined problem of (at least somewhat) broad scientific interest. Unfortunately, quantum algorithms are often compared to textbook, deterministic classical algorithms, which might make quantum methods look favorable. However, in many cases modern classical methods from randomized linear algebra — also termed *quantum inspired methods* — scale similarly as the quantum methods.

We will mostly focus on algorithms with provable worst case performance guarantees, but heuristics for typical performances on relevant instances will also be commented on from time to time. Whereas, it is true that classical algorithms that perform well in practice are often not exactly the same as the ones with the best

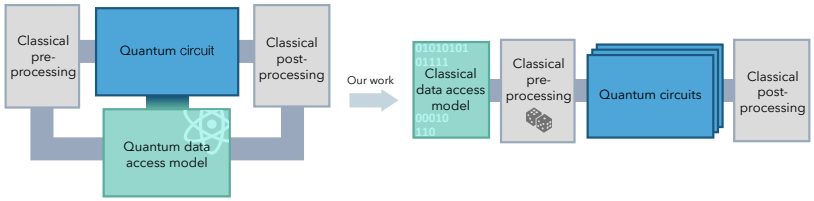


Figure 1.1: Motivation of recent work of our group, minimizing overall quantum resource costs by employing classical instead of quantum random memory data access structures.

provable worst case performance guarantees, it is important to realize that such findings are typically based on extensive practical experiments and corresponding fine tuning. In contrast, we take the viewpoint that in the quantum setting, the current nascent state of quantum hardware largely only allows mathematical analyses to make statements about sufficiently large instance sizes. Of course, these worst-case performance guarantees may overlook further advantages of quantum advantages that might becomes visible once better hardware allows for improvements adapted to specific problem instances of interest.

Our quantum algorithms will also extensively use classical subroutines for pre- and post-processing, and we will demand a holistic viewpoint by asking to explicitly quantify all the classical and quantum resources employed. That is, we will quantify the overall resource costs not only in terms of algorithmic qubits and quantum gates, but also in terms of quantum memory usage, classical algorithmic costs, classical memory usage, etc.[1]

[1] Samson Wang, Sam McArdle, and Mario Berta. Qubit-efficient randomized quantum algorithms for linear algebra. *PRX Quantum*, 5:020324, 2024

1.2 *Organization*

Lecture at RWTH Aachen: The lecture takes place every Tuesday from 2:30pm to 4:00pm in room MBP2 116 and will also be live streamed via Zoom (but not recorded). Physical attendance is strongly recommended. The course is open to Physics, Computer Science, Mathematics, Electrical Engineering, and Chemistry students (with corresponding ECTS credit points). If fewer than 30 students will take the course, we will have oral exams; otherwise a written exam.

Exercise lessons at RWTH Aachen: The weekly tutorials with exercise sheets are run by Gereon Kossmann, starting in the second week on Wednesday April 16 from 2:30pm to 3:15pm in room MBP2 116. Besides discussing the sample solutions that will always come online one week after the exercise sheet, the tutorial will also feature new content (that is in principle part of the examinable content as well).

In principle, no previous physics or computer science knowledge is required (although it might of course help). We will work in abstract and self-contained mathematical framework of computing rules. As such, only a good working knowledge in linear algebra is a prerequisite. Nevertheless, as the language of quantum theory is mathematics — or more precisely linear algebra — this is a challenging applied mathematics lecture with interdisciplinary content.

Ultimately, the goal is to put you in a position to start thesis work on quantum algorithms by diving deeper into some of the presented subjects, both in terms of theory work and applied implementations. The course will largely be taught in black board style, with concise lecture notes that are to be completed by the students during the lectures (e.g., figures, etc.). The lecture notes might still be updated during the term and any feedback would be highly appreciated.

All teaching material is available online on RWTHmoodle (work in progress).[2] In addition, a short literature list is as follows.

[2] https://moodle.rwth-aachen.de/

- Introductory book: Quantum Computation and Quantum Information, Michael A. Nielsen and Isaac L. Chuang, Cambridge University Press.
- More extensive lecture notes: Quantum Computing, Ronald de Wolf, available online.[3]
- Complementary lecture notes: Quantum Algorithms, Andrew M. Childs, available online.[4]
- Mathematical lecture notes: Quantum Computation, Ashley Montanaro, available online.[5]
- Research level lecture notes: Quantum Algorithms For Scientific Computation, Lin Lin, available online.[6]

[3] http://arxiv.org/abs/1907.09415

[4] http://www.cs.umd.edu/~amchilds/qa/

[5] https://people.maths.bris.ac.uk/~csxam/teaching/qc2020/

[6] https://math.berkeley.edu/~linlin/qasc/

A wiki like review article on end-to-end complexities of state-of-the-art quantum algorithm design — including a critical assessment — is also available online.[7] As mentioned in the introduction, we will exclusively treat the theory of quantum algorithm development, with a focus on problems from scientific computing. In particular, the following topics around quantum technologies will not be covered in the lecture:

[7] Alexander M. Dalzell, Sam McArdle, Mario Berta, Przemyslaw Bienias, Chi-Fang Chen, András Gilyén, Connor T. Hann, Michael J. Kastoryano, Emil T. Khabiboulline, Aleksander Kubica, Grant Salton, Samson Wang, and Fernando Brandão. Quantum algorithms: A survey of applications and end-to-end complexities. 2023b. URL http://arxiv.org/abs/2310.03011

- Quantum hardware and device physics
- Analogue quantum simulators
- Algorithms for noisy and intermediate-scale (NISQ) quantum processing units[8]
- Quantum error correction
- Quantum programming

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

- Quantum information theory
- Quantum cryptography
- Post-quantum cryptography

Around these other topics, some related lectures at RWTH Aachen are:[9]

- Introduction to Quantum Computing (Computer Science)
- Near-term Quantum Computation (Computer Science)
- Quantum Algorithms Seminar (Computer Science)
- Quantum Information (Physics)
- Quantum Information Seminar (Physics)
- Superconducting Qubit Circuits (Physics)
- Experimental Quantum Computing with Superconducting Qubits (Physics)
- Spin Qubits (Physics)
- Quantum optics (Physics)
- Building a Quantum Computer (Physics)
- Quantum mechanics for electrical engineers (Electrical Engineering)
- Any more?

[9] Check out https://qc.rwth-aachen.de/ regarding quantum computing theory efforts at RWTH Aachen.

Next, and before we introduce the general mathematical framework of quantum algorithms in Chapter 2, we first briefly discuss classical algorithms and complexity theory through a certain lens that is adapted for generalization to the quantum setting.[10]

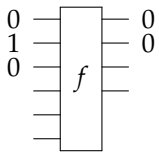
[10] The remaining part of Chapter 1 is inspired and partly adapted from the lecture notes *Quantum Information Theory, Matthias Christandl, WS 2009/2010 LMU*.

1.3 *Classical circuit model*

One way to think about classical algorithms (that will be our way in to generalize to the quantum setting) is in terms of *classical circuits* or *circuits*. Namely, in the circuit model, for an n-bit input m-bit output function

$$f : \{0,1\}^n \to \{0,1\}^m \quad \text{with } n, m \in \mathbb{N}, \tag{1.1}$$

we draw a graphical representation in terms of a box with n input *wires* and m output wires. Concrete inputs and/or outputs can also be labeled:



Moreover, the *truth table* identifies output with inputs.

Example 1. *Consider the parity function, which takes n input bits x_i and maps them to the one bit output*

$$f(x_1 x_2 \cdots x_n) = \bigoplus_{i=1}^{n} x_i \,, \tag{1.2}$$

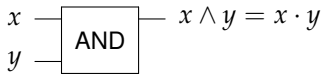
where $\bigoplus$ denotes addition modulo 2. For $n = 2$, its truth table is given by

input	*output*
00	*0*
01	*1*
10	*1*
11	*0*

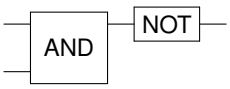
For computing general functions in the circuit model, one might then decompose the function of interest in terms of *elementary gates*. Common choices of elementary gates include the

- NOT gate

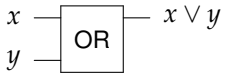
- AND gate



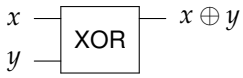
- NAND gate



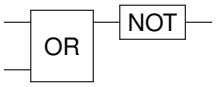
- OR gate



- XOR gate

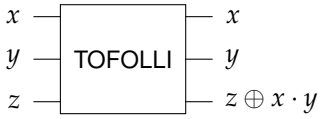


- NOR gate



- FANOUT gate

- TOFOLLI gate



Exercise 2. *Write down the truth tables for all the elementary gates above.*

Importantly, every function can be written in terms of elementary gates and one might for example use the following *universal set* (other choices are possible).

Proposition 3 (Universality I)**.** *By using the elementary gates* NOT, AND, OR, *and* FANOUT *every function can be computed.*

Proof. The proof is part of Exercise Sheet 1. □

Now, on the one hand, this means that any function on n bits can be decomposed into 2^n elementary gates. On the other hand, for functions that can be evaluated *efficiently* we are aiming for decompositions into polynomial many elementary gates in the *input length* n. We will revisit this point soon in Section 1.4 on complexity classes.

As an interesting variation, it is possible to instead only use *reversible gates* as elementary gates.

Proposition 4 (Universality II)**.** *By using the elementary* TOFFOLI *gate, every function can be computed.*

Proof. The proof is part of Exercise Sheet 1. □

One motivation for this is as follows: For non-reversible circuits, it can be argued that when computing, erasing information always increases the entropy of the system and therefore leads to heat dissipation.[11] Additionally, we will see in Chapter 2 that reversible circuits are conceptually close to quantum circuits.

[11] Rolf Landauer. Irreversibility and heat generation in the computational process. *IBM Journal of Research and Development*, 5(3):183, 1961

As an extension of deterministic algorithms, for probabilistic algorithms, the circuits act on n-bit valued probability distributions as inputs, with corresponding m-bit valued probability distributions as outputs. That is, the wires then do not have definite values but only take certain values with certain probabilities. Note that the number of different n-bit strings in $\{0,1\}^n$ is 2^n — and as such the space of all possible input (or output) values is exponential in n.

1.4 Computational complexity theory

For any given function, the central question of computational complexity theory is how many elementary gates are needed to compute it. In principle, one is thereby equally interested in upper bounds, i.e., concrete circuit decompositions of the function into elementary gates, as well as lower bounds, showing that a certain number of elementary gates is needed.[12]

[12] Boaz Barak and Sanjeev Arora. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2007

Instead of looking at functions of fixed input-output size and counting the concrete number of elementary gates needed, one typically studies families of different sized instances, labeled by a parameter $n \in \mathbb{N}$, and is then interested in the scaling for $n \to \infty$. To quantify corresponding complexity upper and lower bounds, there is the *big O notation* in computer science:

1. $f(n) = \mathcal{O}(g(n))$ if and only if there exist $c > 0$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, we have $f(n) \leq c \cdot g(n)$
2. $f(n) = \Omega(g(n))$ if and only if there exist $c > 0$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, we have $f(n) \geq c \cdot g(n)$
3. $f(n) = \Theta(g(n))$ if and only if we have $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$

A simple example is as follows.

Example 5. *The computational complexity of the parity function is $\mathcal{O}(n)$.*

Often, the exact computational complexity of functions is not known.

Example 6. *For the multiplication of two n-bit numbers the textbook algorithm gives $\mathcal{O}(n^2)$. However, we actually have $\mathcal{O}(n \log n)$.*[13] *Only the trivial lower bound $\Omega(n)$ is known and it is a fundamental open problem what is the complexity of multiplication.*

[13] David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193 (2):563, 2021

We can equally well talk about matrix functions, a prominent example is as follows.

Example 7. *For matrix multiplication of two $n \times n$ matrices A and B, the textbook algorithm*

$$(A \cdot B)_{i,j} = \sum_{k=1}^{n} (A)_{i,k} (B)_{k,j} \tag{1.3}$$

gives $\mathcal{O}(n^3)$, whereas there is also the trivial lower bound $\Omega(n^2)$. Can you come up with a matrix multiplication algorithm with complexity better than $\mathcal{O}(n^3)$? The current record for matrix multiplication is $\mathcal{O}(n^{2.371339})$.[14] *In Exercise Sheet 1, you will work through the Strassen algorithm, which has complexity $\mathcal{O}(n^{\log_2 7})$.*

[14] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication, 2024. URL http://arxiv.org/abs/2404.16349

The big O-notation hides any (potentially large) constants and as such asymptotically optimal algorithms do not necessarily need to be practical for any finite size instance. Nevertheless, by coarse graining even further, we typically think of circuits that are polynomial in input length of the problem as being *efficient*. One can then categorize into (asymptotic) *complexity classes*. Very informally speaking, some prominent examples which focus on time complexity include:

- All problems that can be computed with (classical) circuits that are polynomial in input length (=number of bits) make up the complexity class *polynomial-time* P.
- All problems that can be computed with error probability at most $1/3$ and probabilistic (classical) circuits that are polynomial in input length make up the complexity class *bounded-error probabilistic polynomial time* BPP.
- All problems for which answers can be checked with (classical) circuits that are polynomial in input length are in the complexity class *non-deterministic polynomial-time* NP.

A famous problem that is known to be in NP, but not in P or BPP is as follows.[15]

[15] It is a fundamental open question in theoretical computer science if P = NP or P ≠ NP.

Example 8. *Integer factorization: Given an n-bit number, find its product decomposition into prime numbers. The general number field sieve algorithm roughly gives*

$$\mathcal{O}\left(\exp\left(1.9 \cdot n^{1/3}(\log n)^{2/3}\right)\right) . \tag{1.4}$$

Integer factorization is in NP *as multiplication is in* P*, but is integer factorization in* BPP*?*[16]

[16] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303, 1999

As the landmark result of quantum computing, Shor showed that there is a quantum algorithm for integer factorization with quantum gate complexity

$$\mathcal{O}\left(n^2 \log n\right) , \tag{1.5}$$

a super-polynomial (nearly exponential) quantum speed-up compared to the complexity of the best known classical algorithm.

In this course, we are mostly interested in the theory of quantum algorithm development, and as such will only comment on complexity lower bounds from time to time. In the next chapter, we introduce the *quantum circuit model* to formalize what exactly are quantum algorithms and quantum computational complexity theory.

2
Quantum circuit model

2.1 Quantum systems

We know that the laws of physics qualitatively change on different length scales and energy levels. For example, whereas objects on an everyday length scale behave according to the mathematical laws of classical mechanics, objects at the length scale of single molecules or atoms behave according to the mathematical laws of quantum mechanics. Those two theories are fundamentally different in mathematical structure and consequently physical predictions.

Now, when working with smaller and smaller fundamental components of processing units, quantum effects are at first unwanted effects which cause noise and as such great efforts are made in chip development to suppress any quantum mechanical behavior. However, turning things around, one might ask if these quantum mechanical effects can be made use of computationally? Fundamentally, this corresponds to the exciting scientific question what nature can compute efficiently. Intuitively, we might expect that such quantum based computing devices are more efficient to computationally resolve physical and chemical systems described by quantum mechanics. While we will not concern us with how, if, and when quantum computing devices can be built, our interest lies in in the theoretical computer science, algorithmic aspect of the proposal.

Now, computer scientists generally believe(d) in something called the extended Church-Turing thesis. It states that a probabilistic Turing machine can efficiently simulate any realistic model of computation, where the word efficiently here means up to polynomial-time reductions. So following this (and without formally defining Turing machines), it can be said that quantum circuits should not allow for large computational speed-up compared to classical circuits.

Shor's integer factorization algorithm shows that either the extended Church-Turing is wrong or that integer factorization is in BPP. Both came as huge surprises at the time, but one must be true (and we do not know which one).

In this lecture, we will treat the mathematical framework of devices behaving as so-called *closed quantum systems*. This means that we assume perfect shielding of the devices from the environment, as well as perfect control to exactly carry out any operation allowed by the mathematical laws of quantum mechanics. In particular, there is no noise and there are no failures allowed in this model.

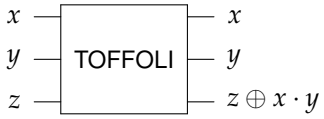
A small disclaimer is in order here. As with other non-standard notions of analogue computation, one has to make sure that there are no hidden (and potentially exponentially large) *physical costs* to actually implement the assumed model.[1] For quantum circuits, these concerns can be averted by a careful analysis of quantum error correction codes, which shows that at least in principle faulty device component can be operated with arbitrarily low noise rates; under the cost of only a low overhead. We briefly touch upon that subject in Section 2.4.

[1] Here is an example: Relativity computing. Namely, just start your computer working on an otherwise intractable problem, board a spaceship, accelerate exponentially close to the speed of light, and come back to earth after a constant time. According to the theory of special relativity time will have elapsed exponentially slower in your reference frame compared to the computer's reference frame on earth and the computation will have finished upon your return. What is the catch?

2.2 *Quantum bits and quantum gates*

To emphasize again: We are neither concerned with the actual physical laws of quantum theory nor based on what concrete technology things are implemented. Rather, we now introduce a purely abstract mathematical model for quantum computation, the *quantum circuit model*.

Our starting point is the classical circuit model and in particular its probabilistic version. As an example, recall the TOFFOLI gate acting on 3 bits as



Writing the possible input bit strings

$$\{0,1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\} \tag{2.1}$$

in vector form $000 \equiv (0,0,0,0,0,0,0,0)^T$, $001 \equiv (0,0,0,0,0,0,0,0)^T$, etc.,[2] the action of the Toffoli gate with respect to the computational basis can be written in matrix form as

$$U_{\text{TOFOLLI}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.2}$$

That is, for any input distribution described by a vector

$$\vec{p}_{\text{in}} \in \mathbb{R}^{2^3}_{\geq 0} = \bigtimes_{i=1}^{3} \mathbb{R}^2_{\geq 0} \text{ with norm } \sum_i |p_i| = 1, \tag{2.3}$$

[2] The transpose is given as

$$(1,0,0,0,0,0,0,0)^T = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

the corresponding output distribution is computed as the normalized vector

$$\vec{p}_{\text{out}} = U_{\text{TOFOLLI}} \cdot \vec{p}_{\text{in}} \in \mathbb{R}^{2^3}_{\geq 0} \,. \tag{2.4}$$

Note further that since TOFOLLI is a reversible gate, its matrix representation U_{TOFOLLI} is orthogonal, i.e., $U^T_{\text{TOFOLLI}} = U^{-1}_{\text{TOFOLLI}}$.

Now, the axioms of quantum computing, or more precisely the quantum circuit model are:

- A single quantum bit — termed *qubit* — is mathematically described by a complex vector $\vec{\psi} = (\alpha_0, \alpha_1)^T \in \mathbb{C}^2$ that is normalized $\|\vec{\psi}\|_2 = \sqrt{|\alpha_0|^2 + |\alpha_1|^2} = 1$ (with $|\alpha|^2 = \alpha\overline{\alpha}$). The coefficients are called *quantum amplitudes*.
- Multiple qubits are described by normalized complex vectors $\vec{\psi}_n$ in the tensor product of the individual $\mathbb{C}^2$ spaces. That is, for n qubits $\vec{\psi}_n \in \otimes_{i=1}^{n} \mathbb{C}^2 = \mathbb{C}^{2^n}$.
- *Quantum gates* described by complex unitary matrices U, that is, $\overline{U}^T = U^{-1}$, taking input complex vectors to output complex vectors. For n-qubit gates the dimension of U is $2^n \times 2^n$.

Complex vectors that describe qubits are termed *quantum states* and instead of the vector notation $\vec{\psi}$, we will also use the physicist's *Dirac bra-ket notation*. For one qubit it takes the form

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{2.5}$$

and a general one qubit quantum state is then written as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \vec{\psi}\,, \tag{2.6}$$

with the normalization $\sqrt{|\alpha_0|^2 + |\alpha_1|^2} = 1$. Consequently, a two qubit state can be written as

$$|\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.7}$$

with the normalization $\sqrt{\sum_{k\in\{0,1\}^2} |\alpha_k|^2} = 1$ and for

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.8}$$

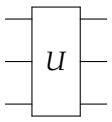
$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{2.9}$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{2.10}$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} . \tag{2.11}$$

Two qubit states that cannot be written as a product of single qubit states are correlated between the qubits and are called *entangled*. You will explore the consequences of these quantum correlations in Exercise Sheet 2.

The graphical notation for gates stays the same as in the classical case,



just with the implicit understanding we are now operating with potentially complex entries. In particular, all reversible classical gates are also quantum gates. Prominent examples of one qubit gates are the

- $\mathsf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ gate

- $\mathsf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ gate

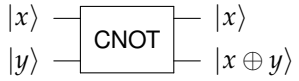
- $\mathsf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ gate

- Hadamard $\mathsf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ gate

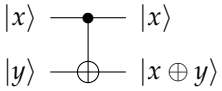
- phase $\mathsf{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ gate

- $\mathsf{T} = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix}$ gate

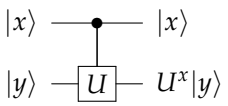
An important two qubit gate is the $\mathsf{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ gate



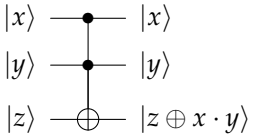
often also written as



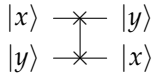
denoting that it corresponds to a *controlled* NOT gate. One can also implement general controlled U-gates



The 3 qubit Toffoli gate can in fact be seen as a controlled-controlled NOT gate as

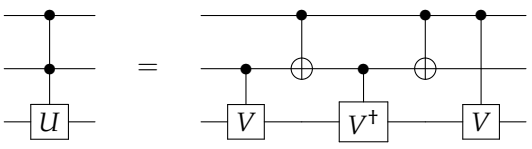


Another two qubit gate of interest is the $\mathsf{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ gate denoted as



Exercise 9. *Check that the matrix representations given above and the Toffoli gate indeed act as claimed in the circuit diagrams.*

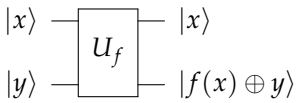
In order to build up to the understanding of more advanced quantum algorithms, it is crucial that you get a good hands on experience how to work with quantum states and how quantum gates transform them. Here is an example of a circuit identity



Exercise 10. *What quantum gate / unitary matrix is V in the above circuit identity?*

You will explore some more examples in Exercise Sheet 2. In particular, there is the so called *no-cloning theorem*, which states that there does not exist a universal cloning quantum gate that exactly copies arbitrary quantum states. You will explore and prove this theorem in Exercise Sheet 2.

As only reversible gates, i.e., unitary matrices, are allowed in the quantum circuit model, how should we implement a general function $f : \{0,1\}^n \to \{0,1\}^m$? One way is to use the two qubit quantum gate U_f defined as



Exercise 11. *Check that U_f is indeed unitary for any function f.*

In the classical circuit model we found different elementary gate sets that are universal, i.e., every function can be written in terms of these gates. We should then ask the same question about universal sets in the quantum circuit model. In the following, we work with one and two qubit gates, but slightly relax the condition to approximate universality. Namely, we are looking for a decomposition of general n qubit gates U as

$$\left\| U - \prod_{i=1}^{t} U_i \right\|_\infty \leq \varepsilon \tag{2.12}$$

in terms of some elementary gates $\{U_1, U_2, \cdots, U_t\}$, some small $\varepsilon \geq 0$, and the infinity norm $\|O\|_\infty = \sup_{|\psi\rangle} \|O|\psi\rangle\|_2$ for quantum states $|\psi\rangle$ which are normalized complex vectors.[3]

[3] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52:1191, 1997

Proposition 12 (Solovay-Kitaev). *The set $\{\mathsf{CNOT}, \mathsf{H}, \mathsf{T}\}$ is universal with $t = \text{polylog}(1/\varepsilon)$, where $\text{polylog}(n) = \mathcal{O}((\log n)^k)$ for some $k \in \mathbb{N}$. That is, arbitrary quantum gates can be approximated up to exponentially small error with only a polynomial overhead.*

Proving this statement will not be part of this course, but we notice that finding such decompositions with as efficient as possible polylog overhead terms is an ongoing and important research topic.

Similarly as in the classical case, any unitary $2^n \times 2^n$ matrix acting on n qubits can now be (approximately) decomposed into $\mathcal{O}(n \cdot 4^n)$ elementary quantum gates. However, efficiently implementable quantum algorithms allow for a decomposition into only polynomial (in n) many quantum gates. Informally, this corresponds to the quantum analogue of the complexity class BPP:

- All problems that can be computed with error probability at most $1/3$ and quantum circuits that are polynomial in input

length make up the complexity class *bounded-error quantum polynomial time* BQP.

The question about large quantum speed-up compared to classical algorithms is then how BQP is related to BPP. We have the inclusion $\mathsf{BPP} \subseteq \mathsf{BQP}$ and integer factorization is a problem that is known to be in BQP, but not known to be in BPP. However, notice that there is NO proven separation between the two complexity classes. That is, in principle it could still be that $\mathsf{BPP} = \mathsf{BQP}$ and as such there would be no quantum advantage in Shor's algorithm. This illustrates the importance of studying complexity theory.

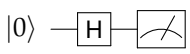
In the next section, we will discuss the last missing ingredient of the mathematical framework of quantum computing. Namely, how to actually read out results from the quantum state outputs of quantum circuits!

2.3 *Quantum measurements*

In order to read out (some of the) classical information contained in quantum states, quantum measurements are applied. For our purposes, so-called measurements in the computational basis are sufficient:

- A qubit state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ measured in the computational basis $\{0,1\}$, leads to the post-measurement binary probability distribution $\{p_0 = |\alpha_0|^2, p_1 = |\alpha_1|^2\}$, where p_0 corresponds to observing the value 0 and p_1 to the value 1. (Note the normalization $|\alpha_0|^2 + |\alpha_1|^2 = 1$.)
- Whenever a measurement is applied, only one measurement outcome will actually be observed (0 or 1). In order to resolve the post-measurement probability distribution one then needs to run the quantum circuit a few times, record the observed outcome every time, and this gives a statistical sample of the true distribution.
- In the same way n qubit states $|\psi\rangle = \sum_{k\in\{0,1\}^n} \alpha_k|k\rangle$ are measured in the computational basis $\{0,1\}^n$, leading to the post-measurement probability distribution $\{p_k = |\alpha_k|^2\}_k$ over all possible outcomes labelled by the binary strings in $\{0,1\}^n$.

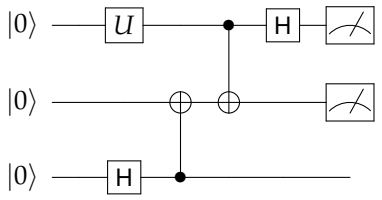
The corresponding quantum circuit symbol is a *measurement box*, e.g., as follows



In this example, the Hadamard H gate takes the quantum state $|0\rangle$ to the quantum state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and the probability of observing 0 or 1 becomes $1/2$ each.

Now, without loss of generality a quantum algorithm on n qubits starts in the all zero state $|0\cdots0\rangle = |0\rangle \otimes \cdots \otimes |0\rangle$, applies a certain

number of elementary quantum gates (from some universal set), and applies measurements on some of the qubits. The output is then a probability distribution over the computational basis of the measured qubits. Here is an example



Having put together the framework for quantum computing, what is intuitively the power of quantum circuits compared to classical circuits? The complex nature of the framework allows negative amplitudes, which can lead to *positive interference patterns*. That and only that is the mathematical reason for observed quantum speed up compared to classical circuits. Do not believe anything else said or claimed in the popular literature!

In Chapter 3, we discuss some simple toy examples of positive interference patterns in quantum circuits that will give a classical-quantum separation in query complexity.

2.4 *Remarks on quantum error correction*

Modern classical hardware is extremely reliable and the assumption of perfect control and no noise is typically reasonable. For typical quantum hardware, however, this is a priori not at all the case. This has to do with both, the very physical nature of the quantum mechanical effects that quantum computing builds on, as well as with the nascent state of quantum technology with only limited control and shielding available against the environment. As such, one needs to argue why the mathematical model of *closed quantum systems* with noise free gates and measurements still makes sense.

The answer to this question is *quantum error correction*. The basic idea of classical error correction is to build in redundancy. For example, say a bit x is flipped with probability $p \in (0, 1/2)$. First storing three copies xxx of the bit and then doing a majority vote the decode back to one bit gives the overall improved error probability

$$3p^2(1-p) + p^3 = p^2(3-2p) < p\,. \tag{2.13}$$

This is the *binary repetition code* and exemplifies the main principle behind error correction. Similar ideas work in the quantum realm, even though some adaptions are necessary due to the no-cloning theorem! Moreover, computations need to be performed on the encoded qubits and the encoding and decoding itself might be faulty etc. All these problems are addressed by the threshold theorem for quantum error correction.[4]

[4] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error rate. *SIAM Journal on Computing*, 38:1207, 2008

Theorem 13 (Quantum Fault-Tolerance). *Under reasonable (physical) assumptions on the noise model, any n qubit quantum circuit with $g(n)$ quantum gates and measurements can be simulated with probability $1 - \varepsilon$ using*

$$\mathcal{O}(g(n) \cdot \mathit{polylog}(g(n)/\varepsilon)) \tag{2.14}$$

many noisy gates and measurements, provided that every component fails with probability $p < p_{tr}$, where p_{tr} denotes a threshold that is in the range of 10^{-3} to 10^{-4}.

Hence, even with faulty components, arbitrarily long computations can be performed with arbitrarily small error (for a small overhead cost). Proving this theorem is beyond the scope of this course (which focuses on quantum algorithms).[5] The development of state-of-the-art quantum error correcting codes is a very active research field both in academia and industry.

To conclude, based on our current understanding of physics, we do not know of any reasons that quantum computing is not possible to implement and scale. Even though it is in principle thinkable that quantum computing is prohibited by some yet undiscovered new physics, many physicists would argue that this would actually be a most exciting outcome of the efforts of realizing quantum technologies. Check out[6] for a more in-depth critical discussion around the feasibility of quantum computing.

[5] It is part of the concurrent RWTH lectures *Quantum Information* in Physics.

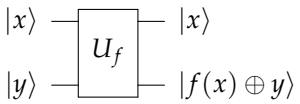
[6] https://www.scottaaronson.com/democritus/lec14.html

3
Quantum query complexity

3.1 Setting

In this chapter you are provided a first glimpse of the power of complex numbers in the quantum circuit model, allowing for positive interference effects whenever the problem of interest has enough *structure* or *symmetries*. For this we work in the framework of *query complexities*, where black box access to some *oracle* is given, and the goal is to resolve some property of the oracle using the minimal number of queries. More precisely, our way to model an oracle is as a function $f : \{0,1\}^n \to \{0,1\}^m$ and querying the oracle means inputting an n bit string, after which the corresponding output m bit string is received. For example, you might want to find an input bit string x such that the output bit string is the all zero string: $f(x) = 0 \cdots 0$.

To compare classical with quantum query complexities, we need a reversible, quantum version of the oracle to the function f. This is achieved as in Exercise 11 above with the quantum circuit



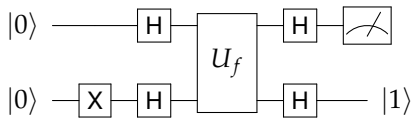
Importantly, given the ability to make a classical query to f, above quantum circuit U_f will allow to query the function f in the quantum circuit model (without knowing anything more about the inner structure of f). Or in other words, U_f is a quantum analogue for implementing the function f.

One of the advantages of query complexities model is that it is much easier to prove lower bounds. As a consequence, we will see that (large) separations between classical and quantum query complexities can be proven. On the other hand, the downside of the query complexity model is that it does not say anything about how the function f is actually implemented (as only the queries to it are counted). Consequently, it is important to realize that query complexity separations do in general not give computational complexity separations. Nevertheless, as a toy model, query complexities give a simple, first way of illustrating the power of the quantum circuit model.

3.2 *Deutsch's problem*

Given a function $f : \{0,1\} \to \{0,1\}$, decide whether f is constant or balanced. That is, what is the value of $f(0) \oplus f(1)$? Classically, and to answer definitely, one needs to query the oracle for f twice to determine this.

What can be done in the quantum setting? Let's figure out what are the measurement outcomes in the quantum circuit



The evolution of the input quantum state leading up to the quantum measurement is

$$
\begin{aligned}
|0\rangle \otimes |0\rangle &\mapsto |0\rangle \otimes |1\rangle && (3.1)\\
&\mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) && (3.2)\\
&\mapsto \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) && (3.3)\\
&\mapsto \frac{1}{2}\left((-1)^{f(0)} + (-1)^{f(1)}\right)|0\rangle \otimes |1\rangle \\
&\quad + \frac{1}{2}\left((-1)^{f(0)} - (-1)^{f(1)}\right)|1\rangle \otimes |1\rangle\,. && (3.4)
\end{aligned}
$$

Measuring the first qubit will lead to the outcome 0 with probability

$$p_0 = \left(\frac{1}{2}\left((-1)^{f(0)} + (-1)^{f(1)}\right)\right)^2 = f(0) \oplus (1)\,, \qquad (3.5)$$

which is equal to one for constant functions (and equal to zero for balanced functions). One quantum query to the function in the form of U_f turns out to be enough! In the next section, we will see that this separation of one vs two query can in fact be scaled to an exponential separation.

3.3 *Deutsch-Josza problem*

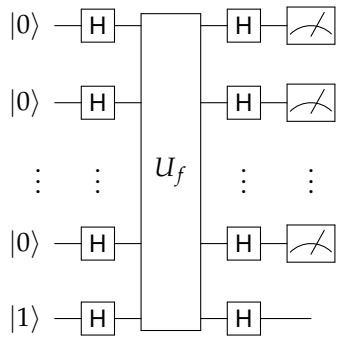
Given a function $f : \{0,1\}^n \to \{0,1\}$ with the promise that either f is constant (all 0 or all 1) on all inputs or balanced (equal number of 0's and 1's), i.e., of the form

$$|\{x : f(x) = 0\}| = |\{x : f(x) = 1\}| = \frac{1}{2} \cdot 2^n\,. \qquad (3.6)$$

Decide which one is the case. Classically, and to answer definitely, one needs to query the oracle $2^{n-1} + 1$ times to determine this.

The Deutsch-Josza quantum algorithm solves this problem with

only one quantum query to the function f in the form U_f as



The evolution of the input quantum state leading up to the quantum measurement is

$$|0\rangle^{\otimes n} \otimes |1\rangle \mapsto \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes (|0\rangle - |1\rangle) \tag{3.7}$$

$$\mapsto \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \otimes (|0\rangle - |1\rangle) \tag{3.8}$$

$$\mapsto \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \left(\frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle \right) \otimes |1\rangle \tag{3.9}$$

$$= \sum_{y\in\{0,1\}^n} \left(\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)+x\cdot y} \right) |y\rangle \otimes |1\rangle \tag{3.10}$$

with the inner product $x \cdot y = \sum_k x_k y_k$, and where we used that

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \Big(|0\rangle + (-1)^{x_1}|1\rangle\Big)(\cdots)\Big(|0\rangle + (-1)^{x_n}|1\rangle\Big) \tag{3.11}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} \prod_{y_k=1} (-1)^{x_k}|y\rangle \tag{3.12}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}. \tag{3.13}$$

After measuring all qubits, the probability of getting the all zero bit string $0\cdots0$ is

$$p_{0\cdots0} = \left(\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \right)^2. \tag{3.14}$$

Thus, if f is balanced, $p_{0\cdots0} = 0$ and otherwise $p_{0\cdots0} = 1$.

This is the promised exponential query separation separation, one quantum query vs $\Theta(2^n)$ deterministic classical queries.

However, what happens in the classical case if you are allowed to use randomness and only have to give the right answer with high probability? You will explore this in Exercise Sheet 3, but the short answer is that the large quantum query complexity advantage

collapses. In the next section, we discuss a query complexity problem that allows for an exponential separation between the classical and quantum setting, even if randomness and non-perfect success probabilities are allowed.

3.4 Simon's problem

Given a function $f : \{0,1\}^n \to \{0,1\}^n$ with the promise that for some unknown *period* $c \in \{0,1\}^n$ we have for all $x, y \in \{0,1\}^n$ that

$$f(x) = f(y) \quad \Leftrightarrow \quad x = y \oplus c\,, \tag{3.15}$$

determine c. For a deterministic classical algorithm we can just evaluate up to half of the inputs before we get a repeat, and then use that

$$x = y \oplus c \quad \Rightarrow \quad x \oplus y = y \oplus c \oplus y = c\,. \tag{3.16}$$

If we still cannot find a match then $c = 0 \cdots 0$. As such, in the worst case we need $2^{n-1} + 1$ queries to f.

Crucially, it can now also be proven that even probabilistic classical algorithms succeeding with high probability, need at least $\Omega(\sqrt{2^n})$ queries to f. This lower bound can also be achieved by a classical scheme based on the so-called *birthday paradox*. Namely, how many *random queries* Q does one has to make in order to observe a collision — i.e., getting the same entry twice — with *high probability*? We give a proof sketch that for $Q = \sqrt{2^{n+1}}$ the expected value for the number of collisions is roughly one.[1] Namely, for queries $j_1, j_2, \cdots, j_Q$, how large is the probability that we do not witness a collision (in the case $c \neq 0 \cdots 0$)? There are $\binom{Q}{2}$ pairs that could be a collision and the collision probability is $\frac{1}{2^n - 1}$. Hence, the *expected number* of collisions is

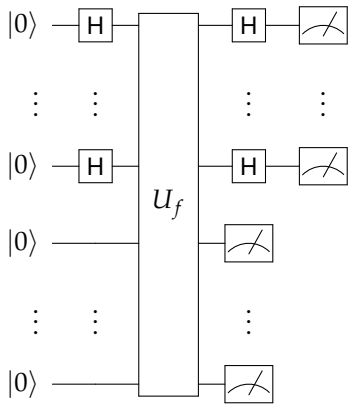
$$\frac{\binom{Q}{2}}{2^n - 1} \approx \frac{Q^2}{2^{n+1}}\,, \tag{3.17}$$

and $Q = \sqrt{2^{n+1}}$ suffices.

[1] An extended argument can also be made for the claimed worst case high probability of success.

In the quantum setting — together with classical post-processing — it turns out that $\mathcal{O}(n)$ quantum queries suffice.[2] The quantum circuit uses $2n$ qubits as

[2] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26:1474, 1997



The evolution of the input quantum state leading up to the first round of quantum measurements is

$$|0\rangle^{\otimes n} \otimes |0\rangle^{\otimes n} \mapsto \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |0\rangle^{\otimes n} \tag{3.18}$$

$$\mapsto \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |f(x)\rangle \,. \tag{3.19}$$

Upon measuring the second half of the qubits and conditioned on observing as the measurement outcome some n-bit string $f(x)$, the state of the first half of the qubits is

$$\frac{1}{\sqrt{2}} \left(|x\rangle + |x+c\rangle\right) , \tag{3.20}$$

and applying the n-fold Hadamard gate leads with the identity from Eq. (3.13) to

$$\frac{1}{\sqrt{2\cdot 2^n}} \left(\sum_{z\in\{0,1\}^n} (-1)^{x\cdot z} |z\rangle + \sum_{z\in\{0,1\}^n} (-1)^{(x\oplus c)\cdot z} |z\rangle \right)$$

$$= \frac{1}{\sqrt{2\cdot 2^n}} \cdot \sum_{z\in\{0,1\}^n} (-1)^{x\cdot z} \left(1 + (-1)^{c\cdot z}\right) |z\rangle \,. \tag{3.21}$$

Now, the final measurements will give uniformly random outcomes from the set $\{z : c\cdot z = 0\}$, with the sought-after c. Hence, upon every run of the circuit, this generates a (binary) linear equation for the unknown n-bit string c. Collecting $n-1$ independent linear equations would then be sufficient to solve for c, as the other known solution is the all zero bit string $0\cdots 0$. It can be argued that $\mathcal{O}(n)$ many runs will lead with high probability to $n-1$ independent equations and then standard Gaussian elimination of classical complexity $\mathcal{O}(n^3)$ allows to solve for c.[3]

[3] Other gates (classical and quantum) are neglected in this analysis, as we are only interested in query complexity. However, the number of such gates is upper bounded by $\mathcal{O}(n^3)$, so there is no hidden exponential complexity blow-up.

We find the promised exponential separation between randomized classical $\Theta(\sqrt{2^n})$ and quantum $\mathcal{O}(n)$ query complexity. This was the first proven exponential separation between randomized classical and quantum!

Note that this is really the setting we want to find separations for, randomness is cheap and having a success probability close to one is sufficient in practice (as one can then always repeat the algorithm a couple of times and boost the success probability arbitrarily close to one).

Of course, we are still in the query complexity setting, but Simon's quantum algorithm as above can actually be seen as the predecessor or the motivation for Shor's quantum integer factorization algorithm —- which the gives quantum computational complexity super-polynomial better than the most efficient classical algorithm known. In the next chapter, we start by treating the main quantum sub-routine of Shor's algorithm, the *quantum Fourier transform*.

3.5 *Other oracle based quantum algorithms*

Before we leave the quantum query complexity setting, we should note that there is a plethora of other query complexity based algorithms with quantum advantages to be found in the literature. Most prominently, there is a class of query algorithms that go under the name of, *Grover search*, *quantum search*, or *amplitude amplification / estimation*, that all go back to Grover's seminal work,[4] who analyzed the following problem. Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, find a $c \in \{0,1\}^n$ such that $f(c) = 1$; or output no solutions if such a c does not exist.

[4] Lov K. Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 53–62, 1998

This can be seen as a version of (unstructured) database search and classically a randomized algorithm will use $\Theta(n)$ classical queries to f. Grover's quantum algorithm uses $\Theta\left(\sqrt{n}\right)$ quantum queries U_f, a quadratic quantum speed-up. The Grover core routine can also be used outside of the query complexity framework, e.g., for quadratically speeding up textbook classical algorithms for NP problems. You will explore this in Exercise Sheet 3.

Whereas these Grover type algorithms are widely applicable, academically pleasing, and are typically treated in textbooks, from our perspective, they suffer from the following two weaknesses:

- It gives at most a quadratic classical-quantum speed-up, which will only be helpful for very advanced quantum hardware — for more primitive hardware the quantum advantage needs to be at the very least super-quadratic to have any chance to be practically witnessed. To put it bluntly, quadratic speed-ups are not the reason people are excited to build quantum computers.
- It operates in the oracle setting and when one thinks about how to implement the Grover oracle for applications, not even the quadratic quantum advantage remains generic. One always has to carefully compare to state-of-the-art specialized classical methods for the problems at hand.

Given that we are mostly interested in large quantum computational complexity advantages for scientific computing, we thus refrain from further treating Grover's algorithm in this course.[5]

[5] It is part of the concurrent RWTH lectures *Quantum Information* in Physics.

4
Quantum Fourier transform

4.1 Discrete Fourier transform

In classical signal processing, the *Fourier transform* is ubiquitous. Its discrete version, $F_N : \mathbb{C}^N \to \mathbb{C}^N$ the discrete Fourier transform (DFT), takes a complex vector $(x_0, \cdots, x_N)$ to a complex vector $(y_0, \cdots, y_N)$ defined element wise as

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \exp\left(\frac{2\pi i j k}{N}\right). \tag{4.1}$$

With the abbreviation $\omega_N = \exp(2\pi i/N)$ as the *N-th root of unity*, the DFT has the matrix representation

$$(F_N)_{j,k} = \frac{\omega_N^{jk}}{\sqrt{N}}, \tag{4.2}$$

with which it can be checked that it is unitary. The DFT can be computed in quasi-linear time with only $\mathcal{O}(N \log N)$ steps by means of the fast Fourier transform (FFT)[1] — instead of the $\mathcal{O}(N^2)$ that one might expect. This is highly useful for various applications and, e.g., leads to a multiplication algorithm of two n bit numbers with $\mathcal{O}(n \log n \log\log n)$ classical gates.[2] As the DFT is already unitary, is there a quantum algorithm to implement the DFT that uses even fewer quantum gates?

[1] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297, 1965

[2] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281, 1971

4.2 Quantum circuit

In order to easily map the problem to n qubits, we set without loss of generality $N = 2^n$. The *quantum Fourier transform (QFT)* takes n-qubit quantum states $|x\rangle$ to the n-qubit quantum state $F_N|x\rangle$. The dimension of the matrix F_N is $2^n \times 2^n$, but what is the minimal number of quantum gates to implement it in the quantum circuit model?

We use the *binary representation* of integers $k \in \{0, \cdots, 2^{n-1})$ as

$$k = k_1 2^{n-1} + \cdots + k_n 2^0 = (k_1, \cdots, k_n) \tag{4.3}$$

with the *binary fraction* notation

$$0.k_l \cdots k_m = \frac{k_l}{2} + \cdots + \frac{k_m}{2^{m-l+1}}. \tag{4.4}$$

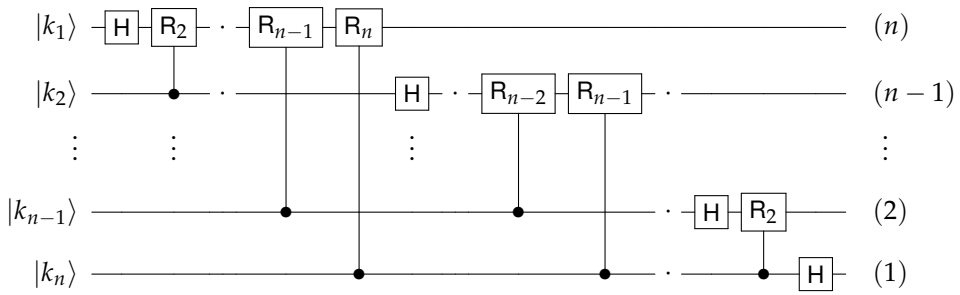
Consider the following decomposition. For a computational basis state $|k\rangle = |k_1 \cdots k_n\rangle$, we have

$$
\begin{aligned}
& F_N|k\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \exp\left(\frac{2\pi i j k}{2^n}\right)|j\rangle && (4.5)\\
&= \frac{1}{\sqrt{2^n}} \sum_{j_1=0}^{1} \cdots \sum_{j_n=0}^{1} \exp\left(2\pi i k \left(\sum_{l=1}^{n} j_l 2^{-l}\right)\right)|j_1\rangle \otimes \cdots |j_n\rangle && (4.6)\\
&= \frac{1}{\sqrt{2^n}} \sum_{j\in\{0,1\}^n} \bigotimes_{l=1}^{n} \exp\left(2\pi i k j_l 2^{-l}\right)|j_l\rangle && (4.7)\\
&= \bigotimes_{l=1}^{n} \frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i k 2^{-l}\right)|1\rangle\right) && (4.8)\\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i 0.k_n\right)|1\rangle\right)(\cdots)\frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i 0.k_1 \cdots k_n\right)|1\rangle\right). && (4.9)
\end{aligned}
$$

In terms of controlled versions of the quantum gates

$$
\mathsf{R}_k = \begin{pmatrix} 1 & 0 \\ 0 & \exp\left(2\pi i 2^{-k}\right) \end{pmatrix} \tag{4.10}
$$

this immediately suggest a quantum circuit as



with the corresponding output quantum states

$$
\begin{aligned}
(n) &= \frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i 0.k_1 \cdots k_n\right)|1\rangle\right) && (4.11)\\
(n-1) &= \frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i 0.k_2 \cdots k_n\right)|1\rangle\right) && (4.12)\\
&\vdots \\
(2) &= \frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i 0.k_{n-1}k_n\right)|1\rangle\right) && (4.13)\\
(1) &= \frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i 0.k_n\right)|1\rangle\right). && (4.14)
\end{aligned}
$$

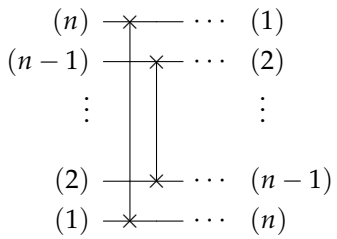
Namely, the evolution of the input quantum state for the actions on the first qubit is

$$
\begin{aligned}
|k\rangle = |k_1 \cdots k_n\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|0\rangle + \exp(2\pi i 0.k_1|1\rangle)\right) \otimes |k_2 \cdots k_n\rangle && (4.15)\\
&\mapsto \frac{1}{\sqrt{2}}\left(|0\rangle + \exp(2\pi i 0.k_1 \cdots k_n|1\rangle)\right) \otimes |k_2 \cdots k_n\rangle, && (4.16)
\end{aligned}
$$

and then going through all the other qubits as

$$\frac{1}{\sqrt{2^n}}\Big(|0\rangle + \exp\left(2\pi i 0.k_n\right)|1\rangle\Big) \otimes \Big(\cdots\Big) \otimes \Big(|0\rangle + \exp\left(2\pi i 0.k_1 \cdots k_n\right)|1\rangle\Big). \tag{4.17}$$

Finally, the additional swap gate circuit



will put the qubit states in the right order.

Now, what is the total number of elementary quantum gates in this quantum circuit? In the first step, one Hadamard gate and $n-1$ conditional rotations are needed, in the second step one Hadamard gate and $n-2$ conditional rotations are needed etc., and hence this gives

$$n + (n-1) + \cdots + 1 = \frac{n(n+1)}{2} \tag{4.18}$$

many conditional rotations. Additionally, we have the $n/2$ swap gates in the end (assuming n even). Each swap gate can be implemented with three CNOT gates (cf. Exercise Sheet 3), each conditional rotation with a CNOT gate and a constant number of elementary single qubit gates (cf. Exercise Sheet 4).

The quantum computational complexity of the QFT becomes $\mathcal{O}(n^2)$. Compare this to the $\mathcal{O}(n2^n)$ classical gates of the FFT, an exponential improvement!

Can the QFT be used to process classical signals exponentially faster? Unfortunately, the answer is no, as the QFT does not solve the same end-to-end problem as the DFT does. Namely, the DFT takes as the input the classical description of a vector $(x_0, \cdots, x_N)$ and outputs the classical of the Fourier transformed vector $(y_0, \cdots, y_N)$. In contrast, the QFT starts with a quantum state, whose amplitudes are given by some coefficients, that are subsequently Fourier transformed. As such, there are at least two problems:

- How are the amplitudes of the original quantum state loaded?
- How are the Fourier transformed amplitudes of the final state read out?

In fact, the complexity of the QFT to perform the DFT task is not better than the classical FFT. However, the qualitatively different task of the QFT can be successfully employed in different contexts. For example, in the next section, we sketch how QFT is the quantum core routine of Shor's integer factorization algorithm.

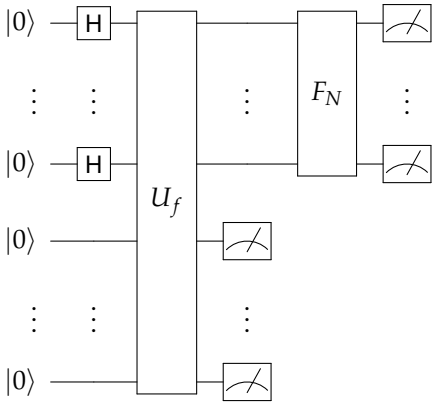
4.3 *Remarks on period finding and Shor's algorithm*

Can the QFT be used to solve a classical end-to-end problem faster than with known classical methods? The answer is yes, and one core problem for which the QFT can be applied is *period finding*.

Given a function $f : \mathbb{N} \to \{0, \cdots, N-1\}$ with the promise that for some unknown period $c \in \{0, \cdots, N-1\}$ we have for all x, y that

$$f(x) = f(y) \quad \Leftrightarrow \quad x = y \mod c\,, \tag{4.19}$$

determine c. Notice the similarity with Simon's problem from Section 3.4! For period finding, one can show the classical probabilistic query complexity lower bound $\Omega(N^{1/3}\,(\log N)^{-1/2})$, while in the quantum setting the period c can be determined with $\mathcal{O}\,(\log\log N)$ queries with the quantum circuit



which features the QFT for all the $\mathcal{O}\,(\log\log N)$ runs. While similar to the quantum circuit to resolve Simon's problem from Section 3.4, you will explore in Exercise Sheet 4 how above quantum circuit works in detail and why it solves the period finding problem.

Now, while for period finding, this is an exponential classical-quantum separation, it is (at first) again only in a query complexity setting and in order to say something about computational complexity one would also need to implement the corresponding quantum oracle U_f. This of course heavily depends on the exact function f at hand, but a series of efficient classical computational number-theoretic reductions show that integer factorization can be solved (with high probability) if the period of the *modular exponentiation* function

$$f_a(x) = a^x \mod N \tag{4.20}$$

can be found for any fixed $a \in \{0, \cdots, N-1\}$. Luckily, there is already an efficient classical algorithm to implement $f_a(x)$ for $a \in \{0, \cdots, N-1\}$ with complexity $\mathcal{O}((\log N)^2 \log\log N \log\log\log N)$. Consequently, the reversible quantum oracle U_{f_a} can be implemented in this complexity as well, and carefully putting together all the steps one finds the overall complexity $\mathcal{O}\left(n^2 \log n \log\log n\right)$

for integer factorization of n-bit integers.[3] This is improved by using the state-of-the-art multiplication scheme from[4] to the claimed $\mathcal{O}\left(n^2 \log n\right)$.

This then gives an end-to-end problem with quantum computational complexity super-polynomial better than the best known classical algorithm. This finding is really exciting and kick started the field of quantum computing as a broad research area. There are also varies related problems to integer factorization — aka the *hidden subgroup problem* — for which one can employ similar ideas to find super-polynomial classical-quantum speed-ups. We note that for all these problems, *symmetries* are of utmost importance to allow for quantum algorithms to provide large speed-ups.

Besides challenging the extended Church-Turing, Shor's efficient quantum algorithm for integer factorization has also gained popularity as the security of a widespread public key cryptosystem, the Rivest–Shamir–Adleman (RSA) scheme, relies on the hardness of the integer factorization problem.

However, as integer factorization and more generally computational number theory are otherwise not necessarily of broad interest for scientific computing, we will skip any details on the proof of Shor's algorithm[5] and rather focus on other more generic applications of quantum algorithms. In the next chapter, we explore another use of the QFT, for the task of *phase estimation*, which can indeed be more broadly employed for scientific computing.

[3] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303, 1999

[4] David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193 (2):563, 2021

[5] These proof details will be part of the concurrent RWTH lecture *Quanten-Computing Quantum Information* in Physics.

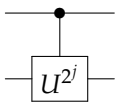
5
Quantum phase estimation

5.1 Problem setting

We are given a unitary matrix U of size $2^m \times 2^m$ and a accompanying normalized eigenvector $\vec{u}$ with eigenvalue parametrized as

$$U \cdot \vec{u} = \exp(2\pi i\phi) \cdot \vec{u} \text{ with } \phi \in [0,1], \tag{5.1}$$

due to the unitarity of U. We assume that we can prepare the corresponding quantum state $|u\rangle = V(u)|0^{\otimes m}\rangle =$ by means of an m-qubit quantum circuit $V(u)$, and can run controlled versions of the corresponding quantum gates U^{2^j} with $j = 1, \cdots, n$ for some $n \in \mathbb{N}$ as



The goal of *quantum phase estimation* is to determine an $n-$bit number that is, up to a failure probability $\varepsilon \in (0,1]$ equal to the best $n-$bit approximation $\phi(n)$ of the phase ϕ of the eigenvalue $\exp(2\pi i\phi)$.

At first sight, it is noteworthy that there are a lot of assumptions here: Not only do we have some m-qubit quantum gate $V(u)|0^{\otimes m}\rangle = |u\rangle$ for preparing the relevant eigenvector, but we also have access to a black-box quantum gate implementation of the controlled version of U and its higher powers. For the latter, it is possible to, e.g., implement via $U^4 = U \cdot U \cdot U \cdot U$, but this might in general become prohibitively expensive once we are not only interested in the number of queries to U, but also in the computational cost of implementing U and its higher powers. Notwithstanding these caveats (that we will examine in more detail in Section 5.3), we will see in Chapter 7 that the task of phase estimation as above is a basic subroutine often used in quantum algorithms for scientific computing.

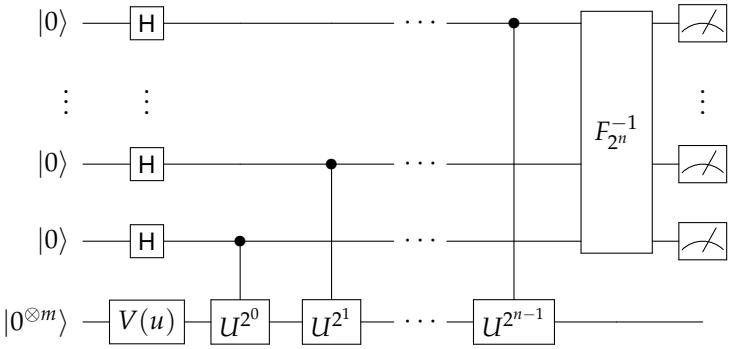
5.2 *Quantum circuit*

We first treat the special case that the phase ϕ can be exactly written as an $n-$bit string, i.e.,

$$\phi = \phi(n) = 0.\phi_1 \cdots \phi_n = \frac{\phi_1}{2} + \cdots + \frac{\phi_n}{2^n}\,, \tag{5.2}$$

recalling the the binary fraction notation. Then, the following $n+m$ qubit quantum circuit featuring the QFT can be employed[1]

[1] Apart from the query complexity to U, we have seen in the previous Chapter 4 that the QFT can be implemented in complexity $O(n^2)$.



Namely, the evolution of the input quantum state for all the steps before the QFT is

$$\begin{aligned}
&|0^{\otimes(n+m)}\rangle \\
&\mapsto \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \otimes |u\rangle && (5.3)\\
&\mapsto \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \otimes U^j|u\rangle && (5.4)\\
&= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \exp(2\pi i j\phi)|j\rangle \otimes |u\rangle && (5.5)\\
&= \frac{1}{\sqrt{2^n}} \left(|0\rangle + \exp(2\pi i 2^{n-1}\phi)|1\rangle\right) (\cdots) \left(|0\rangle + \exp(2\pi i 2^0\phi)|1\rangle\right) \otimes |u\rangle\,, && (5.6)
\end{aligned}$$

where we repeatedly applied the controlled-U^{2^j} quantum gates for $j = 1, \cdots, 2^n - 1$ in the second step, and the last inequality is via the binary representation $j = j_1 2^{n-1} + \cdots + j_n 2^0$.

Now, for the assumed exact $\phi = 0.\phi_1 \cdots \phi_n$, the quantum state on the first n-qubits just becomes

$$\frac{1}{\sqrt{2^n}} \left(|0\rangle + \exp(2\pi i 0.\phi_n)|1\rangle\right) (\cdots) \left(|0\rangle + \exp(2\pi i 0.\phi_1 \cdots \phi_n)|1\rangle\right)\,. \tag{5.7}$$

Subsequently applying the inverse QFT on these first n-qubits, and measuring then leads — by the QFT identity in Eq. (4.9) — to the n-bit string $(\phi_1, \cdots, \phi_n)$. The corresponding binary fraction

$$0.\phi_1 \cdots \phi_n = \frac{\phi_1}{2^1} + \cdots + \frac{\phi_n}{2^n} \tag{5.8}$$

is exactly equal to the sought after n-bit approximation $\phi(n)$! The complexity stays efficient as the QFT is implemented with only $O(n^2)$ quantum gates.

For the general case when $\phi \neq \phi(n)$, and a failure probability upper bounded by $\varepsilon \in (0,1]$, we can choose

$$n' = n + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right) \right\rceil > n \tag{5.9}$$

many qubits instead of the n-qubits on the QFT registers. After the corresponding measurements, the consecutive binary fraction

$$0.\phi_1 \cdots \phi_{n'} = \frac{\phi_1}{2^1} + \cdots + \frac{\phi_{n'}}{2^{n'}} \equiv \overline{\phi} \tag{5.10}$$

can then be shown to determine the best $n-$bit approximation $\phi(n)$ of the phase ϕ with probability $1-\varepsilon$. The argument gets a bit subtle for technical reasons and you will explore a simplified analysis in Exercise Sheet 5.

5.3 *Variations and caveats*

Quantum phase estimation on its own is still a query complexity algorithm, as the implementation of U, or better the controlled version of its powers, is not specified in terms of complexity. Nonetheless, will be useful for resolving the energy levels of Hamiltonians of quantum many body systems, as we will explore in some details in Chapters 6 and 7.

In the following, we address and relax some of the assumptions that went into the quantum phase estimation algorithm:

- In order to achieve an n-bit accurate approximation of the relevant phase, one needs to implement the controlled U operations up to $U^{2^{n-1}}$. A priori this is then exponentially expensive in n if one only has a black box implementation of U at hand. However, if more about the quantum circuit implementation of U is known, this can be avoided for certain structures.

- What happens if one does not have to ability to exactly prepare the relevant eigenstate $|u\rangle$, but rather can only prepare some other state $|\psi\rangle$? Writing $|\psi\rangle = \sum_v \alpha_v |v\rangle$ as a superposition in terms of the eigenstates of U (they constitute an orthonormal basis of the whole space), the quantum phase estimation algorithm will output an estimation of the corresponding phases ϕ_v with probability $|\alpha_v|^2$ each. If we have some a priori estimation on U, this might still be useful. You will explore this setting more in Exercise Sheet 5.

- The number of *ancilla qubits* on top of the n-qubits needed to represent the unitary matrix U, scales linearly with the number of bits of precision required. In practice, this can be prohibitively expensive. Moreover, the QFT is in practice also relatively complex in terms of quantum gate costs and somewhat lacking noise

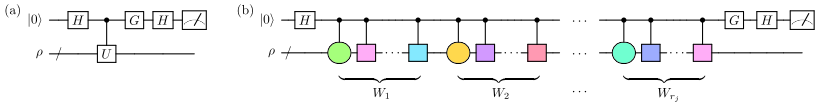


Figure 5.1: Schematic depiction of work in our group: Randomized phase estimation algorithm that avoids the use of the QFT, and only employs one ancilla qubit — thereby minimizing the overall quantum resource costs (at the cost of some classical pre- and post-processing).

resilience. Luckily, there are other, modern versions of quantum phase estimation, that avoid the use of the QFT,[2] and end-to-end only require one ancilla qubit.[3] Some versions also make extensive use of classical signal processing methods.[4]

In the next chapter, we discuss specific unitary matrices that are of the form $U = \exp(iHt)$, with H Hermitian given by a physical Hamiltonian, and how to implement them efficiently.

[2] Lin Lin and Yu Tong. Heisenberg-limited ground-state energy estimation for early fault-tolerant quantum computers. *PRX Quantum*, 3:010318, 2022

[3] Kianna Wan, Mario Berta, and Earl T. Campbell. Randomized quantum algorithm for statistical phase estimation. *Physical Review Letters*, 129:030503, 2022

[4] Thomas E. O'Brien, Brian Tarasinsk, and Barbara M. Terhal. Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments. *New Journal of Physics*, 21: 023022, 2019

6
Hamiltonian simulation

6.1 Task

Intuitively, we might expect that quantum computers are particularly good (efficient) at simulating physical systems described by quantum mechanics, and indeed that was one of the earliest proposals for quantum computers.[1] This could concern static properties such as resolving energy levels of quantum mechanical systems (e.g., of molecules) or dynamic properties, i.e., how the systems evolves in time. The latter is the task of Hamiltonian simulation. Without going into any of the details on the physics, the mathematical description is as follows.

[1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467, 1981

Given an initial n-qubit state $|\psi\rangle$ together with an n-qubit Hermitian matrix $H = H^\dagger$, that describes a quantum mechanical system, the task of *Hamiltonian simulation* is to (approximately) create the *time evolved* quantum state

$$|\psi(t)\rangle = \exp(-iHt)|\psi\rangle \text{ for } t > 0. \tag{6.1}$$

The goal is thereby to achieve this with as few quantum gates as possible.

Here, the *matrix exponential* is defined either via the spectral decomposition of the Hermitian matrix H, or even more directly via the Taylor series

$$\exp(-iHt) = \sum_{k=0}^{\infty} \frac{(-iHt)^k}{k!} = 1 - iHt + \frac{(Ht)^2}{2} + \cdots . \tag{6.2}$$

Note that $U(t) = \exp(-iHt)$ indeed becomes a unitary matrix

$$U(t)U^\dagger(t) = \exp(-iHt)(\exp(-iHt))^\dagger = \mathbb{1}, \tag{6.3}$$

and as such can be implemented in the quantum circuit model. In general, this will require $O(n \cdot 4^n)$ many elementary gates, but whenever the Hamiltonian H has some particular structure — which comes from physics or chemistry — the ambition is to find quantum circuits with only $\text{poly}(n)$ many quantum gates.

In order to achieve this, we have to resort to some *approximate notion* of Hamiltonian simulation, where we implement some unitary matrix $\overline{U}(t)$ with uniformly, for all t

$$\|\overline{U}(t) - U(t)\|_\infty \leq \varepsilon \text{ for } \varepsilon \in [0,1]. \tag{6.4}$$

Recall from around Eq. (2.12) that the infinity norm is defined as

$$\|O\|_\infty = \sup_{|\psi\rangle} \|O|\psi\rangle\|_2 \text{ for quantum states } |\psi\rangle, \tag{6.5}$$

and hence gives the strongest possible notion of approximation. In Exercise Sheet 6, you will prove that for sequences of unitary matrices $\{U_i\}_{i\in I}$ and $\{\overline{U}_i\}_{i\in I}$, we have:

$$\|\overline{U}_i - U_i\|_\infty \leq \varepsilon \; \forall i \in I \quad \Rightarrow \quad \left\| \prod_{i\in I} \overline{U}_i - \prod_{i\in I} U_i \right\|_\infty \leq |I|\varepsilon\,. \tag{6.6}$$

That is, the *error propagation* with respect to the infinity norm is well-behaved as well. Now, the goal is for a given triple (H, t, ε), to find the shortest quantum gate implementation that approximates the unitary $U(t) = \exp(-iHt)$. As remarked above, the complexity of this crucially depends on the (physical) structure of the Hamiltonian H.[2]

[2] The following content is inspired and partly adapted from the lecture notes *Quantum Computation, Ashley Montanaro* (linked in Section 1).

In general, quantum physical systems with Hamiltonians H first have to be mapped to qubits, i.e., to a complex matrix of dimension $2^n \times 2^n$, and we will discuss such mappings in the forthcoming Chapter 7. However, some Hamiltonians already have the correct form by default. To state some examples, it is convenient to work in the *Pauli basis* of the n-qubit space $\mathbb{C}^{2^n} \times \mathbb{C}^{2^n}$, given as

$$\{\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \mathbb{1}\}^{\otimes n}\,. \tag{6.7}$$

So, an example element is of the form

$$\mathsf{Z}_1 \otimes \mathsf{X}_2 \otimes \cdots \otimes \mathbb{1}_{n-1} \otimes \mathsf{Y}_n\,, \tag{6.8}$$

and you will explore this more in Exercise Sheet 6. Examples of Hamiltonians of interest in qubit form are then as follows.

Example 14. *The Heisenberg chain on a one-dimensional line is described by the Hamiltonian*

$$H_{Heisenberg} = \sum_{j\in J} \left(C_x \cdot \mathsf{X}_j \otimes \mathsf{X}_{j+1} + C_y \cdot \mathsf{Y}_j \otimes \mathsf{Y}_{j+1} + C_z \cdot \mathsf{Z}_j \otimes \mathsf{Z}_{j+1} \right), \tag{6.9}$$

where C_x, C_y, C_z denote constants and we used the shorthand notation

$$\mathsf{X}_j \otimes \mathsf{X}_{j+1} \equiv \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{j-1} \otimes \mathsf{X}_j \otimes \mathsf{X}_{j+1} \otimes \mathbb{1}_{j+2} \otimes \cdots \otimes \mathbb{1}_n \tag{6.10}$$

and similar. Crucially, $|J| = \mathrm{poly}(n)$, i.e., the number of non-zero Pauli terms only scales polynomial — whereas the Hamiltonian matrix H is of size exponential in n.

Example 15. *The Ising model on a two-dimensional square lattice is described by the Hamiltonian*

$$H_{Ising} = C \cdot \left(\sum_{j \in J} \mathsf{Z}_{i,j} \otimes \mathsf{Z}_{i+1,j} + \mathsf{Z}_{i,j} \otimes \mathsf{Z}_{i,j+1} \right) , \tag{6.11}$$

where again $|J| = \text{poly}(n)$*, and we use the same notation as in the previous example.*

There are two important take-away messages here: First, physical n-qubit Hamiltonian matrices H can typically be written with only $\text{poly}(n)$ many Pauli terms. Second, all these Pauli terms only act non-trivial in a *local neighborhood of size k* each, e.g, in the above examples only on next neighbors $k = 2$.[3] These two factors taken together reflect the fact that interactions in physics are local. As we will see, this is exactly what makes quantum physical systems amenable to simulation by the quantum circuit model. As such, this then gives a first mathematical justification for the intuition that quantum computers are particularly good at simulating physics.

[3] Note that sufficient locality also automatically implies some sparsity.

6.2 Commuting case

Let's now make things mathematically precise. How can one actually implement the sought-after unitary matrix $\exp(-iHt)$ for

$$H = \sum_{j \in J} \beta_j P_j \text{ for } P_j \in \{\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \mathbb{1}\}^{\otimes n} \text{ and } \beta_j \in \mathbb{R}, \beta = \max_{j \in J} |\beta_j| \tag{6.12}$$

in terms of elementary quantum gates? Unfortunately, on the matrix level, in general for decompositions into $H = H_1 + H_2$, we have

$$\exp(H) \neq \exp(H_1) \exp(H_2) , \tag{6.13}$$

unless $[H_1, H_2] = H_1 H_2 - H_2 H_1 = 0$. This prevents a direct simplification of $\exp(-it \sum_{j \in J} \beta_j P_j)$ in terms of its Pauli components, and you will verify this in Exercise Sheet 6.

Nevertheless, let's first analyze the simple special case that all terms $\{P_j\}_{j \in J}$ pairwise commute.[4] Then, we have

[4] These are also called *commuting* Hamiltonians.

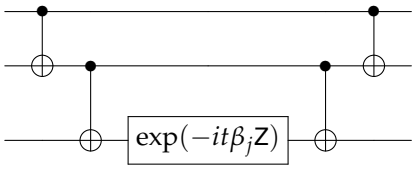
$$\exp\left(-it \sum_{j \in J} \beta_j P_j \right) = \prod_{j \in J} \exp\left(-it \beta_j P_j \right) \tag{6.14}$$

and the quantum circuit representation problem reduces to implementing one n-qubit term $\exp(-it\beta_j P_j)$ with $P_j \in \{\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \mathbb{1}\}^{\otimes n}$. Diagonalizing the Pauli matrices on each qubit with single qubit unitaries U_l and using that the eigenvalues are all ± 1, we can write

$$\exp\left(-it \beta_j P_j \right) = \left(\bigotimes_{l=1}^{k_j} U_l \exp\left(-it \beta_j \mathsf{Z}^{\otimes k_j} \right) \bigotimes_{l=1}^{k_j} U_l^\dagger \right) \otimes \mathbb{1}_{n-k_j} , \tag{6.15}$$

which only acts non-trivially on a number $k_j \leq n$ of the qubits. The remaining unitary matrix $\exp\left(-it \beta_j \mathsf{Z}^{\otimes k_j} \right)$ adds a global phase

$\exp(\pm it\beta_j)$, depending on the input state $|x_1 \cdots x_{k_j}\rangle$ having even or odd parity $\sum_{l=1}^{k_j} x_l$. To implement this, we can use the following quantum circuit, depicted for the simple case $k_j = 3$:



The straightforward generalization to larger $k_j \in \mathbb{N}$ just has more layers of CNOT gates. That is, overall the one n-qubit matrix $\exp\left(-it\beta_j P_j\right)$ is implemented with at most $O(n)$ elementary gates. Together with Eq. (6.14) this leads to a total of $O(|J|n)$ elementary gates for the commuting case, which is polynomial in n, as long as the number of terms is $|J| = \mathrm{poly}(n)$.

Can we reproduce the same scaling for the general case with non-commuting terms $[P_{j_1}, P_{j_2}] \neq 0$ despite Eq. (6.13)?

6.3 *Trotter based methods*

The first simple idea is to anyway use the same product formula[5]

$$\prod_{j\in J} \exp\left(-it\beta_j P_j\right) \tag{6.16}$$

in the hope that it stills reasonable approximates $\exp\left(-itH\right)$ in the non-commuting case. One intuition for this is that all the P_j's only act non-trivially in a neighborhood of constant size k each ($k = 2$ in above examples), and as such the commutator norms $\|[P_{j_1}, P_{j_2}]\|_\infty$ should only scale with the constant k instead of the system size n. Consequently, we might hope that this *near-commutation* leads to a good approximation.[6]

[5] Seth Lloyd. Universal quantum simulators. *Science*, 273:1073, 1996

[6] In the following, we will not quantitatively make use of this but rather employ simplified methods.

We resort to the *Lie-Trotter expansion*

$$\exp(H_1 + H_2) = \lim_{p\to\infty} \left(\exp\left(\frac{H_1}{p}\right)\exp\left(\frac{H_2}{p}\right)\right)^p, \tag{6.17}$$

and similar multipartite versions thereof. While these are asymptotic formulae, we might take a finite $p \in \mathbb{N}$ and try to bound the difference

$$\left\| \exp(H_1 + H_2) - \underbrace{\exp\left(\frac{H_1}{p}\right)\exp\left(\frac{H_2}{p}\right)(\cdots)\exp\left(\frac{H_1}{p}\right)\exp\left(\frac{H_2}{p}\right)}_{p \text{ times}} \right\|_\infty . \tag{6.18}$$

In particular, for $p = 1$ this again reduces to the commuting Ansatz. Now, the remainder of this section is spent on making these ideas rigorous and to indeed determine the quantitative scaling.

We start with a bipartite decomposition $H = H_1 + H_2$ under the boundedness conditions $\|H_1\|_\infty, \|H_2\|_\infty \leq \gamma \in (0, 1)$. The first order

Taylor expansion

$$\exp(H) = 1 + H + \sum_{k\geq 2} \frac{H^k}{k!} \tag{6.19}$$

gives the estimate

$$\exp(H_1)\exp(H_2) = \exp(H_1 + H_2) + E_{12} \text{ with } \|E_{12}\|_\infty \leq O(\gamma^2), \tag{6.20}$$

via the choice

$$\begin{aligned} E_{12} = H_1 H_2 &+ \sum_{k\geq 2} \frac{(1+H_1)H_2^k + H_1^k(1+H_2) - (H_1+H_2)^k}{k!} \\ &+ \sum_{k,k'\geq 2} \frac{H_1^k H_2^{k'}}{k!k'!}\,. \end{aligned} \tag{6.21}$$

Iteratively applying this for the multipartite case under the boundedness conditions $\|H_j\|_\infty \leq \gamma\ \forall j \in J$ gives

$$\prod_{j\in J} \exp(H_j) = \exp\left(\sum_{j\in J} H_j\right) + E_J \text{ with } \|E_J\|_\infty \leq O\left(|J|^3\gamma^2\right). \tag{6.22}$$

This follows as we have, e.g., for $|J| = 4$, the steps

$$\begin{aligned} &\exp(H_1)\exp(H_2)\exp(H_3)\exp(H_4) \\ &= \Big(\exp(H_1+H_2) + E_{12}\Big)\exp(H_3)\exp(H_4) && (6.23)\\ &= \Big(\exp(H_1+H_2+H_3) + E_{123}\Big)\exp(H_4) \\ &\quad + E_{12}\exp(H_3)\exp(H_4) && (6.24)\\ &= \exp(H_1+H_2+H_3+H_4) \\ &\quad + \underbrace{E_{1234} + E_{123}\exp(H_4) + E_{12}\exp(H_3)\exp(H_4)}_{=E_4}\,, && (6.25) \end{aligned}$$

with the corresponding bound

$$\begin{aligned} \|E_4\|_\infty &\leq \|E_{1234}\|_\infty + \|E_{123}\exp(H_4)\|_\infty + \|E_{12}\exp(H_3)\exp(H_4)\|_\infty && (6.26)\\ &\leq (4-1)^2\cdot O(\gamma^2) + (4-2)^2\cdot O(\gamma^2)\cdot O(1) + O(\gamma^2)\cdot O(1) && (6.27)\\ &\leq 4(4-1)^2\cdot O(\gamma^2) \leq 4^3\cdot O(\gamma^2)\,. && (6.28) \end{aligned}$$

So, as long as $\|H_j\|_\infty \leq \gamma\ \forall j \in J$ for $\gamma \in (0,1)$, the first order Trotter expansion gives an approximation

$$\left\|\prod_{j\in J}\exp(H_j) - \exp\left(\sum_{j\in J} H_j\right)\right\|_\infty \leq O\left(|J|^3\gamma^2\right)\,. \tag{6.29}$$

To employ this, we first write for some $r \in \mathbb{N}$ (to be chosen later)

$$U(t) = \exp(-itH) = \left(\underbrace{\exp\left(\frac{-itH}{r}\right)}_{=U(t/r)}\right)^r = \left(\exp\Big(\sum_{j\in J}\underbrace{\frac{-it\beta_j}{r}}_{=H_j(t/r)} P_j\Big)\right)^r, \tag{6.30}$$

and then Eq. (6.29) applied for each $U(t/r)$ gives the approximation

$$\left\| U(t/r) - \underbrace{\prod_{j\in J} \exp(H_j(t/r))}_{=\overline{U}(t/r)} \right\|_\infty \leq \|E_{|J|}\|_\infty \leq O\left(|J|^3 \left(\frac{t\beta}{r}\right)^2 \right) . \tag{6.31}$$

By the error propagation bound from Eq. (6.6), this leads to

$$\left\| U(t/r)^r - \overline{U}(t/r)^r \right\|_\infty \leq O\left(|J|^3 \frac{t^2\beta^2}{r} \right) , \tag{6.32}$$

and hence there exists a constant C such that

$$\left\| \exp(-itH) - \left(\prod_{j\in J} \exp\left(\frac{-it\beta_j}{r} P_j \right) \right)^r \right\|_\infty \leq C \cdot |J|^3 \frac{t^2\beta^2}{r} . \tag{6.33}$$

Choosing

$$r = \left\lceil \frac{C|J|^3 t^2 \beta^2}{\varepsilon} \right\rceil \text{ for some } \varepsilon > 0 \tag{6.34}$$

leads to a first order Trotter scheme with total quantum gate complexity $O\left(|J|^4 \beta^2 t^2 \varepsilon^{-1}\right)$ for an ε-good approximation.

Typically, for the number of terms $|J| = \text{poly}(n)$ and for the constants $\beta = O(1)$, and as such the quantum gate complexity for implementing $\exp\left(-it\sum_{j\in J} \beta_j P_j\right)$ up to approximation $\varepsilon > 0$ in infinity norm becomes

$$O\left(\text{poly}(n) t^2 \varepsilon^{-1}\right) . \tag{6.35}$$

Using the higher p-th order version of the Lie-Trotter expansion in Eq. (6.17), one can asymptotically improve this to

$$O\left(\text{poly}(n) t^{1+1/p} \varepsilon^{-1/p}\right) . \tag{6.36}$$

Finally, much of modern research on the topic goes into specifying and optimizing the exact $\text{poly}(n)$ dependence as well as the hidden constants in the big O-notation. This heavily depends on the structure of the $\{\beta_j, P_j\}_{j\in J}$ in the Hamiltonian — or more precisely on the scaling of the commutator norms $\|[P_{j_1}, P_{j_2}]\|_\infty$ — and makes all the difference when it comes to actual implementations. Note that our argument above actually did not make use of the intuition that $\|[P_{j_1}, P_{j_2}]\|_\infty$ should only scale with the locality k instead of n, but this can be worked out.[7]

While Trotter based methods do not use any ancilla qubits (which is nice!), and in practice actually even scale much better than what can be proven in terms of worst case performance guarantees, the dependence on the approximation error ε is inverse polynomial. In the next section, we sketch some more involved, modern techniques, that improve that dependence on ε to poly-logarithmic.

[7] Andrew M. Childs, Yuan Su, Minh C. Tran, Nathan Wiebe, and Shuchen Zhu. Theory of Trotter error with commutator scaling. *Physical Review X*, 11:011020, 2021

6.4 Linear combination of unitary based methods

The basic idea is to implement with a quantum circuit an appropriately truncated Taylor series of the exponential function

$$\exp\left(-it\sum_{j\in J}\beta_j P_j\right) = \sum_{k\geq 0}\frac{(it)^k}{k!}\left(\sum_{j\in J}\beta_j P_j\right)^k \tag{6.37}$$

$$= \sum_{k\geq 0}\sum_{(j_1\cdots j_k)\in J^k}\frac{t^k\beta_{j_1}\cdots\beta_{j_k}}{k!}\cdot i^k P_{j_1}\cdots P_{j_k} \tag{6.38}$$

You will show in Exercise Sheet 7, that truncating at

$$k = O\left(t + \log\left(\varepsilon^{-1}\right)\right) \tag{6.39}$$

already leads to an ε-approximation in infinity norm. Crucially, since k only depends poly-logarithmic on ε, this then leads to the improved Hamiltonian simulation complexity in t (details to come).[8]

[8] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Physical Review Letters*, 114:090502, 2015

Consequently, it would be sufficient to implement with a quantum circuit the action of a linear combination of unitaries

$$P = \sum_{l\in L}\alpha_l P_l \text{ with } \alpha_l \in \mathbb{R} \text{ and } P_l \in \{\mathsf{X},\mathsf{Y},\mathsf{Z},\mathbb{1}\}^{\otimes n} \tag{6.40}$$

on a quantum state as $|\psi\rangle \mapsto P|\psi\rangle\cdot\||P|\psi\rangle\|_2^{-1}$. However, a linear combination of unitary matrices is in general not a unitary matrix anymore and thus it is a priori unclear how to implement that with a quantum circuit!

The so-called *linear combination of unitaries* (LCU) technique solves that problem in a probabilistic fashion, and with the introduction of $\lceil\log|L|\rceil$ many ancilla qubits. Whereas the basic idea of a truncated Taylor series as mentioned above is conceptually simple, the actual details of this method get a bit technical. Nevertheless, in the following, we sketch the main ideas in order to give you a glimpse into modern techniques for the fundamental task of Hamiltonian simulation.[9]

[9] The following content is inspired and partly adapted from the lecture notes *Quantum Computing, Ronald de Wolf* (linked in Section 1).

To implement the LCU in Eq. (6.40) on an n-qubit quantum state $|\psi\rangle$, the basic steps are:

1. Prepare $|\psi_{\text{in}}\rangle = |0^{\otimes\lceil\log|L|\rceil}\rangle\otimes|\psi\rangle$ on $\lceil\log|L|\rceil + n$ qubits.
2. Apply the $\lceil\log|L|\rceil$-qubit unitary

$$V : |0^{\otimes\lceil\log|L|\rceil}\rangle \mapsto \sum_{l\in L}\sqrt{\frac{\alpha_l}{\|\vec{\alpha}\|_1}}|l\rangle \text{ with } \vec{\alpha} = (\alpha_1,\cdots,\alpha_L), \tag{6.41}$$

   and $\|\vec{\alpha}\|_1 = \sum_{l\in L}|\alpha_l|$ on the ancilla qubits.
3. Apply the unitary

$$W = \sum_{l\in L}|l\rangle\langle l|\otimes P_l \tag{6.42}$$

   on the whole system, where the ancilla registers $|l\rangle\langle l|$ select what P_l is applied on the main register.

4. Apply the inverse unitary $V^\dagger$ on the ancilla qubits.

It is then straightforward to verify (as you will do in Exercise Sheet 7) that the final state takes the form

$$|\psi_{\text{out}}\rangle = \underbrace{\frac{\|P|\psi\rangle\|_2}{\|\vec{\alpha}\|_1}}_{=\sqrt{p_{\text{good}}}} \underbrace{|0^{\otimes\lceil\log|L|\rceil}\rangle \otimes \frac{P|\psi\rangle}{\|P|\psi\rangle\|_2}}_{=|\psi_{\text{good}}\rangle} + \underbrace{\sqrt{1-\frac{\|P|\psi\rangle\|_2^2}{\|\vec{\alpha}\|_1^2}}}_{=\sqrt{p_{\text{bad}}}} \underbrace{|\perp\rangle}_{=|\psi_{\text{bad}}\rangle}, \tag{6.43}$$

where $|\perp\rangle$ denotes some $\lceil\log|L|\rceil + n$ qubit state with the orthogonality constraint

$$\left(|0^{\otimes\lceil\log|L|\rceil}\rangle\langle 0^{\otimes\lceil\log|L|\rceil}| \otimes \mathbb{1}_n\right)|\perp\rangle = 0\,. \tag{6.44}$$

Now, measuring the first $\lceil\log|L|\rceil$ qubits, and conditioned on the all zero measurement outcome $0\cdots 0$, the post-measurement state on the other n qubits is $P|\psi\rangle \cdot \|P|\psi\rangle\|_2^{-1}$ as required!

However, unfortunately, we only have the probability

$$p_{\text{good}} = \frac{\|P|\psi\rangle\|_2^2}{\|\vec{\alpha}\|_1^2} \tag{6.45}$$

of this actually happening. Luckily, the success probability of the scheme can be boosted to close to one by employing *quantum amplitude amplification*. This additional routine repeats all four algorithmic steps above $O\big(1/\sqrt{p_{\text{good}}}\big)$ many times (together with few other elementary quantum gates), such that the resulting output $|\psi'_{\text{out}}\rangle$ has overlap close to one with $|\psi_{\text{good}}\rangle$. This can be seen as a quantum variant of Grover's algorithm, called amplitude amplification (as touched upon in Section 3.5). You will work out this step in detail in Exercise Sheet 7.

So, overall we just need to run steps 1–4 above

$$O\left(\|\vec{\alpha}\|_1 \|P|\psi\rangle\|_2^{-1}\right) \approx O\left(\|\vec{\alpha}\|_1\right) \text{ many times,} \tag{6.46}$$

where for the approximation $\|P|\psi\rangle\|_2 \approx 1$.[10]

It remains to analyze what the complexities of implementing the unitaries V, W from Eqs. (6.41)–(6.42) are:

- The latter unitary W is of the form $W = \sum_{l\in L} |l\rangle\langle l| \otimes P_l$ and corresponds to applying controlled versions of the n-qubit Pauli operators P_l, of which there are $|L|$ many.

- The former unitary V corresponds to *coherent data loading* of the real coefficients $\vec{\alpha} = (\alpha_1, \cdots, \alpha_L)$. The cost of implementing this is typically smaller than the dominating cost of implementing W and as such we neglect the contribution in our asymptotic complexity analysis. We will revisit the topic in Chapter 9 on quantum random access memory (QRAM).

[10] There are a couple of subtleties glanced over here. Firstly, the above scheme will also require to prepare the n-qubit input state $|\psi\rangle$ n-times (even though strictly speaking the task of Hamiltonian simulation only asks for one copy of $|\psi\rangle$). This can be rectified by using the more advanced *oblivious amplitude amplification*. Secondly, amplitude amplification a priori only works if P is unitary, which it is not exactly. Nevertheless, in the variant *robust amplitude amplification*, this is controlled by exploiting how close P is to unitary (via quantifying $\|P|\psi\rangle\|_2 \approx 1$).

Going back to the original Hamiltonian simulation problem from Eq. (6.37), we approximate in infinity norm

$$\exp\left(-it\sum_{j\in J}\beta_j P_j\right) \approx_\varepsilon \sum_{k=0}^{O\left(t+\log\varepsilon^{-1}\right)} \sum_{(j_1\cdots j_k)\in J^k} \underbrace{\frac{t^k\beta_{j_1}\cdots\beta_{j_k}}{k!}}_{=\alpha_{jk}} \cdot \underbrace{i^k P_{j_1}\cdots P_{j_k}}_{=P'_{j_1}\cdots P'_{j_k}}, \tag{6.47}$$

for which the number of rounds from Eq. (6.46) is upper bounded by

$$\begin{aligned}\|\vec{\alpha}\|_1 &= \sum_{k=0}^{O\left(t+\log\varepsilon^{-1}\right)} \frac{t^k}{k!} \sum_{(j_1\cdots j_k)\in J^k} \beta_{j_1}\cdots\beta_{j_k} && (6.48)\\ &\leq \sum_{k\geq 0} \frac{\left(t\|\vec{\beta}\|_1\right)^k}{k!} && (6.49)\\ &= \exp\left(t\|\vec{\beta}\|_1\right), && (6.50)\end{aligned}$$

where $\vec{\beta} = (\beta_1, \cdots, \beta_{|J|})$. Due to the exponential dependence on the time t, this is not quite good enough yet.

However, like for Trotter based methods, we can reduce to small times via the identity

$$(\exp(-isH))^p = \exp(-ispH). \tag{6.51}$$

So, instead, we run the LCU Hamiltonian simulation $p = t\|\vec{\beta}\|_1$ many times, with each

$$\text{time } s = \|\vec{\beta}\|_1^{-1} \text{ and error } \bar{\varepsilon} = \varepsilon\cdot t^{-1}\|\vec{\beta}\|_1^{-1}. \tag{6.52}$$

Now, for each of these runs, the number of iterations of the algorithmic steps 1–4 is $O\left(\exp\left(s\|\vec{\beta}\|_1\right)\right) = O(1)$ and these steps are then dominated by the cost of implementing the corresponding W from step 3. That is, one needs controlled versions of the $P'_{j_1}\cdots P'_{j_k}$ from Eq. (6.47), with k up to

$$O\left(s+\log\bar{\varepsilon}^{-1}\right) = O\left(\log\left(t\|\vec{\beta}\|_1\varepsilon^{-1}\right)\right). \tag{6.53}$$

In terms of elementary gates, this can be decomposed into

$$O\left(|J|(n+\log|J|)\log\left(t\|\vec{\beta}\|_1\varepsilon^{-1}\right)\right) \text{ steps}, \tag{6.54}$$

but we forgo the somewhat lengthy argument.[11] Finally, the total cost of doing this $p = t\|\vec{\beta}\|_1$ times becomes

$$O\left(t\|\vec{\beta}\|_1|J|(n+\log|J|)\log\left(t\|\vec{\beta}\|_1\varepsilon^{-1}\right)\right). \tag{6.55}$$

As $\|\vec{\beta}\|_1 = O(|J|)$, we conclude:

[11] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Physical Review Letters*, 114:090502, 2015

The complexity for implementing $\exp\left(-it\sum_{j\in J}\beta_j P_j\right)$ up to approximation error $\varepsilon > 0$ in infinity norm roughly becomes

$$O\Big(\underbrace{|J|^2(n+\log|J|)}_{=\mathrm{poly}(n)}\, t\log\left(|J|t\varepsilon^{-1}\right)\Big)\,, \tag{6.56}$$

i.e., quasi-linear dependence on time t and logarithmic dependence on inverse approximation error ε. In terms of asymptotic complexities this is a big improvement compared to the p-th order Trotter method with

$$O\left(\mathrm{poly}(n)t^{1+1/p}\varepsilon^{-1/p}\right)\,. \tag{6.57}$$

However, the LCU method comes with generally more demanding quantum circuits (in practice), and at the cost of roughly

$$O\left(\log\left(|J|\right)\log\left(|J|t\varepsilon^{-1}\right)\right) \tag{6.58}$$

ancilla qubits.

6.5 State-of-the-art methods and caveats

There are a multitude of conceptually different quantum algorithms for performing Hamiltonian simulation, each with their advantages and disadvantages. For early fault-tolerant quantum devices, Trotter based methods might be the most promising because they do not use any ancilla qubits and they also perform much better in typical use cases than what can be proven in terms of worst case performance guarantees. Another promising candidate for non-sparse Hamiltonians are randomized schemes, termed qDRIFT[12].

On the other end of the spectrum, are the most recent methods based on quantum signal processing, that we will touch on in the last Chapter 10. Here, the query complexity to a *block encoding* of the Hamiltonian is[13]

$$O\left(|J|t + \frac{\log\varepsilon^{-1}}{\log\left(e+\frac{\log\varepsilon^{-1}}{|J|t}\right)}\right)\,, \tag{6.59}$$

which is also known to be optimal in terms of asymptotic complexities. However, for this one does first need to create the block encoding query oracle from the Hamiltonian given in Pauli form, which leads to some additional overhead in the end-to-end gate complexity. We discuss the construction of such block encodings in Chapter 10, which sometimes also makes use of quantum random access memories (Chapter 9).

How does Hamiltonian simulation with quantum algorithms compare to classical methods? At first sight, quantum schemes are very efficient as they solely use $O(\log N)$ qubits and $O\left(\log N\right)$ gates to time evolve exponentially large n-qubit states. In comparison,

[12] Earl Campbell. Random compiler for fast Hamiltonian simulation. *Physical Review Letters*, 123:070503, 2019

[13] András Gilyén, Yuan Su, Guang Hao Low, and Nathan 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*, STOC 2019, page 193, 2019

classical (textbook) methods need $O(2^n)$ bits to keep track of the quantum state. However, this comparison should be treated cautiously:

- The classical schemes achieve something much stronger, as they give a full classical description of the evolved quantum state $|\psi\rangle$. That is, one can then read off any desired feature of the system. In contrast, quantum Hamiltonian simulation just gives a quantum description of the quantum state $|\psi\rangle$, i.e., the state sits in a quantum memory. What to further do with that? For example, reading out some observable up to precision $\delta > 0$, will require at least $\Omega(\delta^{-1})$ many copies to be measured, and as such the quantum experiment would have to be repeated many times. Let alone resolving a full classical description of the state, which would again need $O(2^n)$ copies!
- Specialized modern classical methods have nowhere near $O(2^n)$ complexity, but in practice rather also often (quasi) scale polynomial in n. This is, however, not rigorous, comes with severe scaling constraints for larger instance sizes, and much depends on the exact system in question (e.g., in computational chemistry, condensed matter physics, etc.).

In the next chapter, we combine Hamiltonian simulation with quantum phase estimation to resolve the energy spectrum of Hamiltonians.

7
Ground state energy estimation

7.1 Task

While we studied the simulation of dynamical properties of quantum mechanical systems in the last chapter, we are now interested in *static properties* of such systems. In physics, the eigenvalues of the Hamiltonian corresponds to the energy levels of the system. While the whole spectrum is of interest, most of the systems found in nature are (approximately) in their ground state, and as such the low energy spectrum and in particular the minimal eigenvalue of the Hamiltonian $\lambda_0(H)$ is of fundamental interest. The local Hamiltonian problem is then as follows.[1]

[1] Note that even though the Hamiltonian matrix is of size 2^n, for constant k the problem description itself is of size $\text{poly}(n)$.

Given an n-qubit Hamiltonian of the form

$$H = \sum_{j\in J} \beta_j P_j \text{ for } P_j \in \{\mathsf{X},\mathsf{Y},\mathsf{Z},\mathbb{1}\}^{\otimes n},\ |\beta_j| \leq 1,\ |J| \leq \text{poly}(n), \quad (7.1)$$

and every P_j acting non-trivially on at most k qubits, estimate the minimal eigenvalue $\lambda_0(H)$ up to additive error $\varepsilon \leq 1/\text{poly}(n)$.

This is generally believed to be a hard problem for classical computers. One way to see this is that the Boolean satisfiability problem for $k = 3$ can be encoded in the local Hamiltonian problem, where (a decision version) of the former problem is known to be as hard as nay problem in NP.[2] Namely, for Boolean variables $\vec{x} = (x_1, \cdots, x_n)$ and a given formula $f(\vec{x})$ based on AND, OR, and NOT operators, the 3-SAT problem asks if there are values $\vec{x} = (x_1, \cdots, x_n)$ such that the formula becomes true. For example, for $n = 6$, an instance is

[2] In contrast, 2-SAT is known to be in P.

$$f(\vec{x}) = \underbrace{(x_1 \vee \overline{x}_2 \vee x_3)}_{=C_1} \wedge \underbrace{(x_2 \vee x_3 \vee x_5)}_{=C_2} \wedge \underbrace{(x_1 \vee x_4 \vee x_6)}_{=C_3}, \quad (7.2)$$

with the clauses $\{C_j\}_{j=1,2,3}$. To encode into the local Hamiltonian problem, the basic idea is as follows. Note that, e.g., the first clause has the only non-satisfying assignment $(0,1,0)$ and with that we

can associate an $n = 6$ qubit Hamiltonian term

$$H_{C_1} = \begin{pmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&\mathbf{1}&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{pmatrix} \otimes \mathbb{1}_4 \otimes \mathbb{1}_5 \otimes \mathbb{1}_6 \,, \tag{7.3}$$

where now $\langle x|H_{C_1}|x\rangle = 0$ if and only if $\vec{x}$ satisfies the clause. Encoding in the same way all clauses $\{C_j\}_{j\in J}$ of a formula $f(\vec{x})$ into local Hamiltonian terms gives the Hamiltonian

$$H = \sum_{j\in J} H_{C_j} \tag{7.4}$$

and finding the minimal eigenvalue corresponds to determining the minimal number of unsatisfied clauses, with $\lambda_{\min}(H) = 0$ if and only if the formula $f(\vec{x})$ is satisfiable. Finally, known hardness results for 3-SAT, i.e., Cook's theorem,[3] then imply the NP-hardness for (a decision version of) the local Hamiltonian problem as stated above.

[3] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings Symposium on Theory of Computing*, STOC '71, page 151, 1971

In fact, by having general non-commuting local Hamiltonian terms, one can even show — in a precise complexity-theoretic sense — that already for $k = 2$ the local Hamiltonian problem is in general believed to be hard for a quantum computer as well! However, all of these hardness results use Hamiltonians which do not actually come from *physical systems*, and there is the hope that under further, reasonable physical assumptions the problem becomes feasible on a quantum computer. In any case, what this analysis shows, is that further structure is needed in order to hope for (provably) efficient algorithms.

One possibility that we pursue in the following, is to assume that we have an *Ansatz state* $|\psi_{\text{Ansatz}}\rangle$ at hand, which is promised to have some (a priori unknown) overlap

$$\gamma_{\text{Ansatz}} = |\langle\psi_{\text{Ansatz}}|\psi_0\rangle|^2 > 0 \tag{7.5}$$

with the true ground state $|\psi_0\rangle$ of H, which corresponds to the eigenvector with minimal eigenvalue $\lambda_0(H)$. This is called the *guided local Hamiltonian problem* and a combination of quantum phase estimation (Chapter 5) together with Hamiltonian simulation (Chapter 6) is immediately applicable.

Whereas the black box availability of an appropriate Ansatz state ultimately leaves things up to heuristic methods, our ambition is now to make rigorous statement assuming that we have an Ansatz state of certain quality at hand. On a complexity-theoretic level, one can (roughly) show that classical computers can solve the guided local Hamiltonian problem up to constant precision starting from $\gamma_{\text{Ansatz}} = \text{const.}$, whereas a quantum computer can solve the guided local Hamiltonian problem up to precision inverse polynomial starting from $\gamma_{\text{Ansatz}} = 1/\text{poly}(n)$.[4] Importantly, the latter precision requirement is the relevant setting in practice — but of course still requires good Ansatz states to start with.

[4] Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: Hardness and applications to quantum chemistry and the quantum PCP conjecture. In *Proceedings Symposium on Theory of Computing*, STOC 2022, page 19, 2022

Leaving precise complexity-theoretic considerations aside going forward, the three necessary steps are: (i) mapping the quantum many body system and corresponding Hamiltonian of interest to qubits (Section 7.2), (ii) derive and prepare an appropriate Ansatz state (Section 7.4), and (iii) run quantum phase estimation on these inputs (Section 7.3).

7.2 *Mapping to qubit form*

How does one map general interacting quantum many body systems to n-qubit Hamiltonians? Condensed matter physics examples that we saw include the one-dimensional Heisenberg chain (Example 14) and the two-dimensional Ising model (Example 15). However, these models were already natively given in qubit form because they describe lattice models, where each vertex accommodates an electron, which has two spin degrees of freedom (= one qubit), and the interaction of these is directly modeled with Pauli matrices.

Another example are *fermionic systems*, such as, e.g., electrons for the description of molecules in computational quantum chemistry. Here, the electrons interact with heavy nuclei through the Coulomb interaction, and in good approximation the Born-Oppenheimer approximation treats the nuclei as static point charges. Working with the electrons in so-called *second quantization* and making an Ansatz onto n basis states, the resulting effective Hamiltonian can be written as[5]

$$H = \sum_{pq=1}^{n} h_{pq} \cdot a_p^\dagger a_q + \sum_{pqrs=1}^{n} g_{pqrs} \cdot a_p^\dagger a_q^\dagger a_r a_s \,, \tag{7.6}$$

in terms of the fermionic creation $a_p^\dagger$ and annihilation a_p operators, and coefficients $h_{pq}, q_{pqrs} \in \mathbb{R}$. Without going into the underlying physics, the relevant properties are the fermionic anti-commutation relations

$$\{a_p, a_q^\dagger\} = a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq} \cdot \mathbb{1} \tag{7.7}$$

$$\{a_p, a_q\} = \{a_p^\dagger, a_q^\dagger\} = 0 \,, \tag{7.8}$$

and one needs to map these to the Pauli matrices $\sigma_0 = \mathbb{1}$, $\sigma_1 = \mathsf{X}$, $\sigma_2 = \mathsf{Y}$, and $\sigma_3 = \mathsf{Z}$ with the relations

$$\{\sigma_j, \sigma_k\} = 2\delta_{jk} \cdot \mathbb{1} \,. \tag{7.9}$$

A *fermion to qubit mapping* then maps the relations in Eqs. (7.7)–(7.8) to the relations in Eq. (7.9), and one way to achieve this is the Jordan-Wigner transformation

$$a_p = \mathsf{Z}^{\otimes(p-1)} \otimes \underbrace{\frac{1}{2}(\mathsf{X} + i\mathsf{Y})}_{\text{on } p\text{-th qubit}} \otimes \mathbb{1}^{\otimes(n-p)} \,, \tag{7.10}$$

that you will verify in Exercise Sheet 9. The Jordan-Wigner transformation is the most simple, but note that it introduces non-locality.

[5] So far, these are all standard simplifications/approximations from chemistry — that are equally the starting point in classical methods.

Other more involved mappings that avoid this are possible as well, and you will explore that in Exercise Sheet 9.

Another closely related model of a quantum many body system is the *Fermi-Hubbard lattice model*, which was specifically introduced to understand the phenomena of high temperature superconductivity.[6] The advantage here is that the number of terms in the corresponding fermionic Hamiltonian only scales linearly in the number of lattice sites n, compared to the $O\left(n^4\right)$ worst case scaling for the molecular fermionic Hamiltonians from Eq. (7.6). We do not discuss the Fermi-Hubbard model further, but mention that a complete understanding of the phase diagram is still elusive, and hence there is room for algorithms with small quantum footprint to be impactful.

[6] Ground state energy estimation is not quite enough for this task and instead one needs to understand thermal Gibbs states for temperature $T > 0$.

Other relevant quantum many body systems of interest are within nuclear physics or models of quantum gravity. However, we refrain from treating those, partly because the quantum resources to simulate and resolve such systems seems to be considerably more costly than examples in condensed matter physics and computational quantum chemistry.

7.3 *Quantum phase estimation*

Having mapped the Hamiltonian of interest to qubits, assume that we are given an n-qubit Hamiltonian

$$H = \sum_{j\in J} \beta_j P_j \text{ for } P_j \in \{\mathsf{X},\mathsf{Y},\mathsf{Z},\mathbb{1}\}^{\otimes n} \text{ and } \beta_j \in \mathbb{R}, \tag{7.11}$$

$$\text{with } \beta = \max_{j\in J} |\beta_j|,\ |J| \leq \text{poly}(n), \tag{7.12}$$

and an Ansatz state $|\psi_{\text{Ansatz}}\rangle$ with $\gamma_{\text{Ansatz}} = |\langle\psi_{\text{Ansatz}}|\psi_0\rangle|^2$, where $|\psi_0\rangle$ denotes the eigenvector of H with minimal eigenvalue $\lambda_0(H)$. To keep things simple, we further assume that all eigenvalues $\{\lambda_i(H)\}_{i=0}^{2^n-1}$ of H have an exact d-bit binary representation, and that

$$\lambda_0(H) < \lambda_1(H) \leq \lambda_2(H) \leq \cdots, \tag{7.13}$$

with the spectral gap $\Delta(H) = \lambda_1(H) - \lambda_0(H)$ also resolvable with a d-bit binary representation.

Under these (strong) assumptions, we can then run quantum phase estimation as discussed in Chapter 5 and Exercise Sheet 5 for the unitary

$$U = \exp(-i\pi\overline{H}) \text{ with } \overline{H} = \frac{H}{\sum_{j\in J}|\beta_j|}, \tag{7.14}$$

and input state

$$|\psi_{\text{Ansatz}}\rangle = \langle\psi_{\text{Ansatz}}|\psi_0\rangle \cdot |\psi_0\rangle + \sum_{j=1}^{2^n-1} \alpha_j |\psi_j\rangle, \tag{7.15}$$

written in the eigenbasis $\{|\psi_j\rangle\}_{j=0}^{2^n-1}$ of H. Note that H, $\overline{H}$, and $\exp(-i\pi\overline{H})$ have the same eigenvectors, and that it would be sufficient to instead use the normalized Hamiltonian $H \cdot \|H\|_\infty^{-1}$ to

resolve $\lambda_0(H)$. However, as we do not have direct access to $\|H\|_\infty$, we employ the sufficient upper bound $\|H\|_\infty \leq \sum_{j\in J} |\beta_j|$.

In more detail, employing the polynomial time quantum circuit from Section 5.2 on $n+d$ qubits, the most expensive operation is to implement (a controlled version) of the unitary

$$U^{2^d-1} = \exp\left(i\pi\left(2^d-1\right)\bar{H}\right), \tag{7.16}$$

which can be achieved by means of Hamiltonian simulation as discussed in Chapter 6. Assuming that the Hamiltonian simulation can be done exactly, this outputs $\lambda_0(H)$ exactly, with probability γ_{Ansatz}, and thus

$$\text{repeating the circuit } O\left(\gamma_{\text{Ansatz}}^{-1}\right) \tag{7.17}$$

will lead to success. Using, e.g, the LCU methods — which comes at the cost of few additional ancilla qubits — the complexity of Hamiltonian simulation for Eq. (7.16) roughly scales as

$$O\left(2^d \cdot \text{poly}(n)\right) \tag{7.18}$$

for an exponentially small $\varepsilon = 2^{-n}$ Hamiltonian simulation approximation error.[7]

[7] Even though this error is exponentially small, strictly speaking one stills needs to argue about the worst case error propagation in the quantum phase estimation circuit.

Now, with more work, one can eliminate some of the assumptions made. In particular, with little additional cost, the assumption on the exact d-bit binary representation of the $\{\lambda_i(H)\}_{i=0}^{2^n-1}$ can be dropped. However, any schemes will retain a dependence on the spectral gap $\Delta(H)$, and the precision d has to be chosen such that this gap can be resolved. Besides working out slick quantum circuits for quantum phase estimation as discussed in Section 5.3, modern work goes into achieving favorable scaling with regards to the $\Delta(H)$ gap dependence. Finally, we emphasize again that the relevant quantum circuits have to be repeated $O\left(\gamma_{\text{Ansatz}}^{-1}\right)$ many times, and for that to be efficient one needs $\gamma_{\text{Ansatz}} = 1/\text{poly}(n)$. In the following Section 7.4 we critically assess when this can be achieved.

7.4 *Quantum state preparation and other bottlenecks*

The quantum algorithm presented in Section 7.3 is only efficient if one has the ability to get a good Ansatz state $|\psi_{\text{Ansatz}}\rangle$ with at least $\gamma_{\text{Ansatz}} = 1/\text{poly}(n)$. The physical intuition is that nature typically has such a state available, just because systems are usually observed in low energy states (unless otherwise stimulated). The task of a quantum computer is then to mimic nature in that aspect, but the crucial question is how to do this *efficiently*. In particular, it can be shown that generic quantum states have an exponentially small overlap 2^{-n} with the true ground state $|\psi_0\rangle$.

The first option to try to resolve this, is to use the best available guess $|\psi_{\text{Ansatz}}\rangle$ for the true ground state coming from state-of-the-art classical methods such as versions of density functional theory

(DFT) or tensor network methods (DMRG) for more strongly correlated systems. There are, however, some fundamental difficulties to be aware of:

- Classical methods are usually not rigorous — even if they come from *first principles* — and as such, end-to-end schemes become heuristic.

- Given any classical description of an Ansatz state $|\psi_{\text{Ansatz}}\rangle$, one still needs to find efficient quantum circuits to prepare it on the quantum computer.

- Classical methods typically deliver Ansatz states that are a good local approximation to the true ground state, but not globally. Note that this is sufficient to reproduce precise energy estimates within the classical methods as the Hamiltonian is build from local interaction terms. It is unfortunately not good enough for phase estimation that asks for a good global overlap $\gamma_{\text{Ansatz}} = |\langle\psi_{\text{Ansatz}}|\psi_0\rangle|^2$. So in this sense, the classical methods deliver good estimates of the energy spectrum, but not necessarily states with good global overlap to the true ground state.

The second option is to work with more physically inspired schemes via *adiabatic state preparation*. Here, the idea is to start with preparing the ground state of a simple Hamiltonian $H(0)$, and then slowly evolving this simple Hamiltonian via a *time-dependent version of Hamiltonian simulation* to the Hamiltonian of interest $H = H(1)$. The adiabatic theorem from quantum mechanics then tells us that if the evolution is slow enough with respect to the spectral gaps $\Delta(H(t))$, then the system (approximately) stays in its ground state throughout, and we end up with a good Ansatz state $|\psi_{\text{Ansatz}}\rangle$ for the final Hamiltonian $H = H(1)$. The crux here is to make this procedure precise and (provably) efficient. In particular, the adiabatic scheduling function has to be chosen such that the spectral gap $\Delta(H(t))$ never becomes exponentially small 2^{-n}. Nevertheless, adiabatic state preparation is often cited as a promising direction and you numerically investigated a small scale adiabatic scheme in Exercise Sheet 8.

The third option is to prepare finite temperature *Gibbs states*, which for sufficiently low temperature T also approximate the ground state well. This is again motivated by physical considerations, as systems typically thermalize to an equilibrium when interacting with its environment. The idea of Gibbs state preparation algorithms is to mimic this with quantum Monte Carlo methods, such as, e.g., quantum Metropolis sampling.[8] General, rigorous statements on efficiency seem out of reach, but the techniques are currently actively explored and there are some promising indications on practical run times. We will not explore Gibbs state preparation as it involves departing from the quantum circuit model and working with *open quantum systems*.[9]

[8] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete. Quantum Metropolis sampling. *Nature*, 471:87, 2011

[9] Chi-Fang Chen, Michael J. Kastoryano, and András Gilyén. An efficient and exact noncommutative quantum Gibbs sampler. 2023. URL http://arxiv.org/abs/2311.09207

Finally, even once one has a good Ansatz state, e.g., with perfect overlap $\gamma_{\text{Ansatz}} = 1$, the quantum resources needed to resolve quantum many body systems of interest might still be prohibitively high. An often cited numerical example that is classically on the verge of being challenging is the so-called FeMoco molecule that is the primary co-factor of nitrogenase, that catalyzes the conversion of atmospheric nitrogen molecules N_2 into ammonia NH_3 through nitrogen fixation. Starting from the classical description, the typical Ansatz uses 153 qubits (without counting any ancilla qubits) and in order to resolve chemical accuracy at 0.0016 Hartree, current estimates give at least an order of 10^{10} quantum gates.[10] As such, there are a plethora of modern techniques with the goal of further reducing the quantum resources needed. For example, this includes:

- Hamiltonian reduction techniques using symmetries to minimize the number of qubits.
- Low-rank decomposition techniques to reduce the number of terms in the resulting n-qubit Hamiltonian.
- Specialized algorithms that optimize phase estimation and Hamiltonian simulation all at once. This, e.g., includes randomized techniques that just scale quadratically with the Pauli weight $\lambda = \sum_{j \in J} |\beta_j|$ of the n-qubit Hamiltonian — instead of the generic $\text{poly}(n)$ scaling — and solely use one ancilla qubit.[11]

To conclude, we emphasize that there is no known rigorous and general end-to-end quantum speed-up for the complexity-theoretic hard problem of ground state energy estimation. However, there are various promising avenues to further explore strong quantum heuristics that could give good practical run times and eventually become competitive with classical techniques.

In this regard, we should recall that even though the dimension of the n-qubit Hamiltonian is 2^n, any classical methods used in practice do not keep track of the exponentially many amplitudes of the quantum state, but rather make use of the locality of the Hamiltonian to come up with locally well-performing Ansätze that scale in practice as $\text{poly}(n)$. As such, any quantum algorithm actually has to compete with these polynomial run times, and not with the generic 2^n dimension bound.

For further discussions specifically on computational quantum chemistry, we refer to the review article,[12] and a recent critical discussion on potential quantum speed-up.[13]

[10] Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. *PRX Quantum*, 2:030305, 2021

[11] Kianna Wan, Mario Berta, and Earl T. Campbell. Randomized quantum algorithm for statistical phase estimation. *Physical Review Letters*, 129:030503, 2022

[12] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92:015003, 2020

[13] Seunghoon Lee *et al.* Evaluating the evidence for exponential quantum advantage in ground-state quantum chemistry. *Nature Communications*, 14: 1952, 2023

8
Quantum linear system solver (QLSS)

8.1 Task and classical landscape

In this chapter we consider the following problem and its classical versus quantum complexity:

The *linear system of equations problem* is given by a complex $N \times N$ matrix A and complex vector $\vec{b}$ of length N, and the goal is to write down (or sample) from a vector $\vec{x}$ of length N such that $A\vec{x} = \vec{b}$.

If A has full rank, there exists a solution vector of the form

$$\vec{x} = A^{-1}\vec{b} \tag{8.1}$$

and hence the problem can be reduced to *matrix inversion*, plus matrix-vector multiplication.[1] Standard Gaussian elimination then solves the linear system of equations problem exactly, with algorithmic complexity $\mathcal{O}\left(N^3\right)$ — or more precisely with the best known matrix multiplication exponent $\mathcal{O}\left(N^{2.371339}\right)$. In addition, a memory of size $\mathcal{O}(N^2)$ is needed to have access to the input data $(A, \vec{b})$ in the first place.

[1] If A does not have full rank, the same argument can be made using the *pseudo-inverse*, which corresponds to the inverse on the support of A. NB: This even works for the more general case of $N \times M$ matrices with $N \neq M$.

The applications of this problem to scientific computing are indirect, but fundamental and manifold. This, e.g., includes solving differential equations, sub-routines in machine learning, interior point methods for optimization problems, to name a few. When the size N becomes very large, Gaussian elimination might no longer be feasible and one might resort to algorithms from *randomized linear algebra*. The characteristics of these methods are as follows:

- Typically, they do not read the whole matrix A or vector $\vec{b}$, but rather only sample certain entries with certain probabilities (for which randomness is needed).
- There are different variations of how sample access is provided, and typically information about the input data $(A, \vec{b})$ in addition to just its matrix elements in the computational basis is required (e.g., the condition number of the matrix A, discussed later).
- They only produce with a non-zero probability an approximation to the solution vector $\vec{x}$. Alternatively, even only sampling

access to an approximate version to $\vec{x}$ is achieved, such as, e.g., sampling an element x_i of the solution vector.

- The complexity of the methods depends not only on the dimension of the problem, but also on other parameters of the input data $(A, \vec{b})$. This includes the *sparsity* $s(A)$ of the matrix A in terms of row and/or column sparsity, the *condition number* $\kappa(A) = \|A\|_\infty \cdot \left\|A^{-1}\right\|_\infty$ of the matrix A, or the *approximate rank* of the matrix A measured by $\|A\|_F$.

As an example, you will explore in Exercise Sheet 9 the *randomized Kaczmarz algorithm*,[2] that, given access to the vector two-norm of the columns, as well as the total $\|A\|_F$, produces a vector $\vec{y}$ of length N with $\|\vec{x} - \vec{y}\|_2 \leq \varepsilon$ in complexity

$$\mathcal{O}\left(s(A)\kappa_F(A)^2 \log\left(\varepsilon^{-1}\right)\right) \tag{8.2}$$

for the row sparsity $s(A)$ and the *Frobenious condition number* $\kappa_F(A) = \|A\|_F \|A^{-1}\|_\infty$. Note that it is not necessary for the algorithm to a priori know $\|A^{-1}\|_\infty$, but rather the iterative algorithm will just provably terminate after a number of repetitions scaling with $\|A^{-1}\|_\infty$. Nevertheless, the scheme is only efficient when A is *well-conditioned* with $\kappa_F(A) \ll N$, and in practice this is not a given at all. Consequently, the use of randomized algorithms for linear systems of equations only makes sense in combination with linear system *pre-conditioners*, which do some pre-computations to bring the linear system into a form with well-scaling, reduced condition number. These pre-conditioning techniques are mostly heuristic and it is of course also important to keep the complexity at least comparatively efficient to the main routines.

[2] Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15:262, 2009

There are other, *quantum-inspired* or *dequantized* classical randomized linear system solvers that roughly scale as[3]

$$\widetilde{\mathcal{O}}\left(\kappa(A)^2 \kappa_F(A)^4 \varepsilon^{-2}\right), \tag{8.3}$$

where the tilde denotes up to poly-logarithmic correction terms. By inspection, the complexity is completely independent of the dimension N or the sparsity $s(A)$. However, these techniques come with a different, non-standard input model about the input matrix. They also have worse approximation error scaling that is no longer logarithmic, and they only allow to approximately query entries of the solution vector $\vec{x}$ (instead of outputting the whole vector). As such, the randomized Kaczmarz algorithm and other related stochastic gradient descent methods have their practical use cases, whereas the quantum-inspired methods have so far basically exclusively been discussed in the theoretical computer science literature.

[3] Changpeng Shao and Ashley Montanaro. Faster quantum-inspired algorithms for solving linear systems. *ACM Transactions on Quantum Computing*, 3:4, 2022

8.2 Quantum task

Given the fundamental role of the linear system of equations problem for scientific computing, it is natural to ask for quantum algorithms for the problem. For example, looking at the classical

landscape, there are no strong classical algorithms to handle matrices that are sparse $s(A) \ll N$, but still have high approximate rank as quantified by $\|A\|_F \approx N$. However, quantum computers can only process quantum states (normalized vectors) with unitary operations and consequently one first has to define a version of the linear system of equations problem that is amenable to quantum methods.

One can without loss of generality only consider Hermitian matrices $A = A^\dagger$, and for simplicity, we focus in this chapter on that case, with the additional assumptions that $N = 2^n$ and $\|A\|_\infty = 1$. (These are not restrictive).

Given a Hermitian matrix A of size $2^n \times 2^n$ with $\|A\|_\infty = 1$, a complex vector $\vec{b}$ of length 2^n, and an approximation parameter $\varepsilon \in (0,1)$, a *quantum linear system solver* (QLSS) uses n algorithmic qubits (and further ancilla qubits) towards outputting an n-qubit quantum state $|y\rangle$ such that

$$\||y\rangle - |x\rangle\|_2 \leq \varepsilon \text{ for } |x\rangle = \frac{\sum_{j=0}^{2^n-1} x_j|j\rangle}{\left\|\sum_{j=0}^{2^n-1} x_j|j\rangle\right\|_2} \text{ defined by } A\vec{x} = \vec{b} \quad (8.4)$$

with $\vec{x} = (x_0, \cdots, x_{2^n-1})$.

Notice that this features a twofold normalization in the problem formulation: We chose to focus on the case $\|A\|_\infty = 1$ and the solution quantum state $|x\rangle$ is equal to the solution vector $\vec{x}$ normalized by $\|\vec{x}\|_2$. Notice further that in the end one just has the quantum state $|x\rangle$ sitting in the quantum computer. By measuring this quantum state, this then allows to sample certain properties of $\vec{x}$, such as, e.g., overlaps $\langle\phi|x\rangle$ with n-qubit quantum states $|\phi\rangle$ (at the cost of some additional sample complexity). However, QLSSs do not give efficient access to the full solution vector $\vec{x}$, as this would require an additional sample complexity scaling with the dimension N.[4] Nevertheless, the above suggests that quantum computer can sample from the solution of linear systems of equations of dimension $N = 2^n$ with only n qubits space cost!

[4] Scott Aaronson. Read the fine print. *Nature Physics*, 11:291, 2015

8.3 *Quantum data access*

To quantify the actual complexity and overall end-to-end implementation cost of QLSSs we need to specify how the matrix A and the vector $\vec{b}$ are accessed and loaded into the quantum computer. Concerning the vector $\vec{b}$, the standard model is to assume access to a preparation unitary (and its inverse)

$$U_b|0^{\otimes n}\rangle = |b\rangle = \frac{\sum_{j=0}^{2^n-1} b_j|j\rangle}{\left\|\sum_{j=0}^{2^n-1} b_j|j\rangle\right\|_2} \text{ with } \vec{b} = (b_0, \cdots, b_{2^n-1}), \quad (8.5)$$

and the complexity of the QLSS is then quantified by the number of times U_b and its inverse are invoked (among other contributing

factors). Note, however, that U_b will also have some complexity to be implemented, and unless the elements are computed *on-the-fly*, this implementation scales at the very least with the sparsity $s(b)$ of $\vec{b}$. Similarly as the coherent data loading of the relevant coefficients for the LCU Hamiltonian simulation method (Section 6.4), the detailed construction of corresponding quantum circuits is related to quantum random access memories, which we discuss in Chapter 9.

Concerning the Hermitian matrix A, there are different *data loading models* and motivated by the complexity bottleneck in the landscape of classical linear system solvers, we mention in particular the *sparse access model*: First, one assumes access to a matrix entry query oracle unitary (and its inverse)

$$U_{A,\text{entry}} : |i,j\rangle \otimes |0\rangle \mapsto |i,j\rangle \otimes |(A)_{i,j}\rangle \,, \tag{8.6}$$

where the last register uses some d many qubits to load the matrix entries $(A)_{i,j}$ with d-bits of precision. Importantly, and in contrast to what is needed for classical solvers, this gives *coherent access* to the matrix elements by loading them into quantum amplitudes. Second, assuming that the exponentially large matrix A is both exactly row and column $s(A)$ sparse, one assumes access to these *non-zero positions* through a query oracle unitary (and its inverse)

$$U_{A,\text{pos}} : |k,l\rangle \mapsto |k,\text{pos}(k,l)\rangle \,, \tag{8.7}$$

where the $s(A)$ non-zero entries in column k are at positions

$$\text{pos}(k,0), \cdots, \text{pos}(k,s(A)-1) \,. \tag{8.8}$$

The complexity of the QLSS is then quantified by the number of times $U_{A,\text{entry}}$, $U_{A,\text{pos}}$, and its inverses are invoked (among other contributing factors). Like for U_b, and again unless the elements are computed *on-the-fly*, the detailed construction and cost of these unitaries $U_{A,\text{entry}}$ and $U_{A,\text{pos}}$ is related to quantum random access memories, which we discuss in Chapter 9. Here, we just note that this construction will scale at the very least with the sparsity $s(A)$.

8.4 Basic quantum linear system solver

Having introduced the oracle unitaries $(U_b, U_{A,\text{entry}}, U_{A,\text{pos}})$ that give coherent access to the entries of A and $\vec{b}$, we are now ready to describe the main routines of QLSSs. The first QLSS was introduced in[5] and is based on the quantum phase estimation routine (Chapter 5).[6]

It is sufficient to consider Hermitian matrices, but we simplify the problem further and assume that A only has positive eigenvalues, i.e., that it is positive definite. (This is not restrictive). Then, there is an eigendecomposition

$$A|v_j\rangle = \lambda_j |v_j\rangle \tag{8.9}$$

[5] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103: 150502, 2009

[6] In the following, we discuss a simplified version of this algorithm, following some of the presentation in *Quantum Algorithms For Scientific Computation, Lin Lin* (linked in Section 1).

with the (normalized) eigenvectors v_j written as quantum states $|v_j\rangle$ and increasingly ordered eigenvalues

$$0 < \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{2^n-1} = 1\,. \tag{8.10}$$

Going forward, we further assume for simplicity that all λ_j have an exact d-bit binary representation.

The basic subroutine is to run quantum phase estimation of the unitary $U = \exp(i2\pi A)$ on the input state $|b\rangle$, leading to a quantum state of the form

$$U_{\text{QPE}}\left(|0^{\otimes d}\rangle \otimes |b\rangle\right) = \sum_{j=0}^{2^n-1} \beta_j |\lambda_j\rangle \otimes |v_j\rangle\,. \tag{8.11}$$

Here, U_{QPE} denotes the quantum phase estimation circuit from Section 5.2 (excluding the measurements in the end), and

$$|b\rangle = \sum_{j=0}^{2^n-1} \beta_j |v_j\rangle \tag{8.12}$$

is the expansion of $|b\rangle$ in the eigenbasis of A. Note that running U_{QPE} requires implementing controlled Hamiltonian simulation $\exp(i2\pi tA)$ for variable t, via the query access unitaries $U_{A,\text{entry}}$ and $U_{A,\text{pos}}$. For now, we continue by just quantifying the complexity of the QLSS by counting the needed calls to $U_A = \exp(i2\pi A)$ and then revisit the point about the implementation of controlled Hamiltonian simulation later.

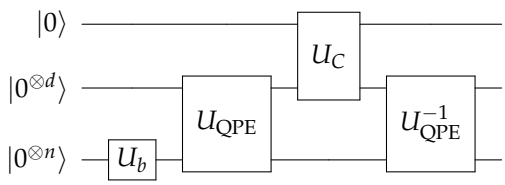
In order to solve the linear system of equations, we would like to create a normalized version of the expression

$$A^{-1}|b\rangle = \sum_{j=0}^{2^n-1} \frac{\beta_j}{\lambda_j} |v_j\rangle \tag{8.13}$$

and towards that, we employ the unitary U_C defined via

$$U_C\Big(|0\rangle \otimes |\lambda_j\rangle\Big) = \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} \cdot |0\rangle + \frac{C}{\tilde{\lambda}_j} \cdot |1\rangle\right) \otimes |\lambda_j\rangle\,, \tag{8.14}$$

controlled on an additional ancilla qubit, and where $\tilde{\lambda}_j$ approximates λ_j, and C denotes some (for now) unspecified normalization constant of choice. Assuming implementations of U_C and U_{QPE} (we get back to these later), the quantum circuit U_{QLSS} of the QLSS then takes the form



and the overall output state becomes

$$U_{\mathrm{QLSS}}\Big(|0\rangle \otimes |0^{\otimes d}\rangle \otimes |b\rangle\Big) = \sum_{j=0}^{2^n-1} \beta_j \left(\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}}\cdot|0\rangle + \frac{C}{\tilde{\lambda}_j}\cdot|1\rangle\right) \otimes |0^{\otimes d}\rangle \otimes |v_j\rangle\,. \tag{8.15}$$

Discarding the d ancilla qubits and measuring the one remaining ancilla qubit gives — conditioned on measurement outcome one — the quantum state

$$|y\rangle = \|\vec{y}\|_2^{-1} \sum_{j=0}^{2^n-1} \frac{C\beta_j}{\tilde{\lambda}_j}|v_j\rangle \text{ with } \vec{y} = \sum_{j=0}^{2^n-1} \frac{C\beta_j}{\tilde{\lambda}_j}\vec{v}_j \tag{8.16}$$

on the remaining main n qubits. This is exactly the sought-after quantum state with $\||y\rangle - |x\rangle\|_2 \leq \varepsilon$ as required by Eq. (8.4).

The success probability of measuring one is

$$p(1) = \|\vec{y}\|_2^2 = \left\| \sum_{j=0}^{2^n-1} \frac{C\beta_j}{\tilde{\lambda}_j}|v_j\rangle \right\|_2^2 \tag{8.17}$$

$$\approx C^2 \cdot \left\| A^{-1}|b\rangle \right\|_2^2 \tag{8.18}$$

and the choice $C = \lambda_0$ optimizes this. We then find

$$p(1) \approx \lambda_0^2 \cdot \left\| A^{-1}|b\rangle \right\|_2^2 \geq \lambda_0^2 \cdot \frac{\||b\rangle\|_2^2}{\|A\|_\infty^2} = \lambda_0^2\,, \tag{8.19}$$

and as we assumed $\|A\|_\infty = \lambda_{2^n-1} = 1$, we roughly get

$$p(1) \geq \Omega\left(\kappa(A)^{-2}\right)\,. \tag{8.20}$$

Thus, we need to run above quantum circuit and measurement about $\kappa(A)^2$ many times until success. If wanted, this can be improved to $\mathcal{O}(\kappa(A))$ many runs by means of *amplitude amplification*, similarly as for the LCU Hamiltonian simulation scheme presented in Section 6.4. You will explore this in Exercise Sheet 10.

Next, we estimate the dependence of the complexity on the approximation error ε. You will show in Exercise Sheet 10, that in order to achieve the overall approximation error

$$\||y\rangle - |x\rangle\|_2 = \left\| \frac{\vec{y}}{\|\vec{y}\|_2} - \frac{\vec{x}}{\|\vec{x}\|_2} \right\|_2 \leq \varepsilon \tag{8.21}$$

one needs to provide the *multiplicative* approximation

$$\tilde{\lambda}_j = \lambda_j(1+\varepsilon/4) \quad \forall j\,. \tag{8.22}$$

This then requires to run the quantum phase estimation routine U_{QPE} to additive precision $\varepsilon\lambda_0 = \varepsilon\kappa(A)^{-1}$. Hence, to implement U_{QPE} the needed query complexity to $U_A = \exp(i2\pi A)$ becomes $\mathcal{O}\left(\kappa(A)\varepsilon^{-1}\right)$.

To subsume, the complexity to achieve an ε-approximation becomes:

- In each round one query to U_b and U_C
- In each round $\mathcal{O}\left(\kappa(A)\varepsilon^{-1}\right)$ many queries to $U_A = \exp(i2\pi A)$
- Employ $\mathcal{O}(\kappa(A)^2)$ many runs.

To break this up further, it remains to construct U_C and $U_A = \exp(i2\pi A)$ from the query access oracles $U_{A,\text{entry}}$, $U_{A,\text{pos}}$ and its inverses. For the implementation of U_C, we first rewrite the relevant coefficients from Eq. (8.14) as

$$\theta_j = \frac{1}{\pi}\arcsin\left(\frac{C}{\lambda_j}\right) \tag{8.23}$$

with $\tilde{\theta}_j$ its corresponding d-bit representation, leading to

$$U_C\Big(|0\rangle \otimes |\lambda_j\rangle\Big) = \Big(\cos(\pi\tilde{\theta}_j)|0\rangle + \sin(\pi\tilde{\theta}_j)|1\rangle\Big) \otimes |\lambda_j\rangle\,. \tag{8.24}$$

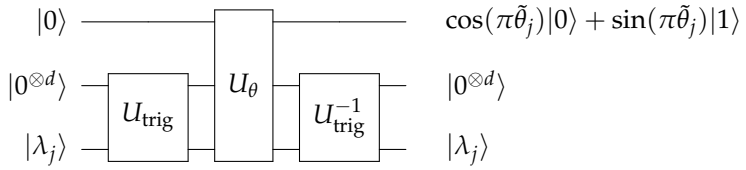
Then, given a d-bit representation of an angle θ, you will show in Exercise Sheet 10, how to implement the $(d+1)$-qubit unitary U_θ acting as

$$U_\theta : |0\rangle \otimes |\theta\rangle \mapsto \Big(\cos(\pi\theta)|0\rangle + \sin(\pi\theta)|1\rangle\Big) \otimes |\theta\rangle \tag{8.25}$$

Further, in Exercise Sheet 10, you will also explore *classical arithmetic circuits* employing d ancilla qubits and $\text{poly}(d)$ many elementary quantum gates to implement Eq. (8.23). That is, implementing the unitary

$$U_{\text{trig}}\Big(|0^{\otimes d}\rangle \otimes |\lambda_j\rangle\Big) = |\tilde{\theta}_j\rangle\,. \tag{8.26}$$

Overall, the quantum circuit for U_C then takes the form



Going forward, we assume that d is chosen large enough such that we can ignore the resulting finite arithmetic approximation error from $\tilde{\theta}_j$ versus θ_j.[7]

The implementation of powers of $U_A = \exp(i2\pi A)$ corresponds to Hamiltonian simulation of the Hamiltonian A, with variable times t up to $t = \mathcal{O}\left(\kappa(A)\varepsilon^{-1}\right)$. However, contrary to the ground state energy estimation problem where the Hamiltonian was given in terms of its sparse Pauli decomposition (Chapter 7), the matrix A is now given in terms of the sparse oracle access in the computation basis, via $U_{A,\text{entry}}$, $U_{A,\text{pos}}$ and its inverses. As such, one can no longer just import the Hamiltonian simulation methods from Chapter 6.

Luckily, for example, a version of the LCU method from Section 6.4 can be worked out for the sparse input model as well, scaling

[7] Strictly speaking one would have to carefully monitor the error propagation.

for an ε-approximate Hamiltonian simulation with

$$\mathcal{O}\left(t \cdot \text{poly}(n, s(A)) \log\left(\varepsilon^{-1}\right)\right) \tag{8.27}$$

in terms of the query complexity to $U_{A,\text{entry}}$, $U_{A,\text{pos}}$ and its inverses — plus some additional constant quantum gate overhead. Going forward, we ignore the scaling of the approximation error from Hamiltonian simulation, as it can be made exponentially small.[8]

[8] Strictly speaking one would have to carefully monitor the error propagation.

Having sketched the quantum circuit of all the necessary components, one arrives at the complexity of

- $\mathcal{O}\left(\kappa(A)\varepsilon^{-1} \cdot \text{poly}(n, s(A))\right)$ many queries to $U_{A,\text{entry}}$, $U_{A,\text{pos}}$ and its inverses
- one query to U_b
- some constant quantum gate overhead,

for each of the required $\mathcal{O}(\kappa(A))$ many rounds (using the version with amplitude amplification).

Assuming that the matrix A is sparse with $s(A) = \text{poly}(n)$ equally for both rows and columns, and that the vector $\vec{b}$ is sparse as well with $s(b) = \text{poly}(n)$, this leads to the complexity

$$\widetilde{\mathcal{O}}\left(\kappa(A)^2 \varepsilon^{-1}\right), \tag{8.28}$$

where the tilde denotes up to poly-logarithmic correction terms. If the matrix A is well-conditioned with scaling $\kappa(A) = \mathcal{O}(\text{poly}(n))$, and the oracles U_b, $U_{A,\text{entry}}$, and $U_{A,\text{pos}}$ can all be implemented in complexity $\text{poly}(n)$, the overall complexity of QLSSs stays $\text{poly}(n)$ for input matrices of dimension 2^n.

The implementation of the oracles might be achieved with the help of quantum random access memory (see Chapter 9). However, note that while the algorithmic complexity of the quantum random access memory might be $\text{poly}(n)$, it will still require at least as many qubits as non-zero entries in the input data $(A, \vec{b})$. (And the number of memory qubits might then be much larger than the $n + d + 1$ algorithmic qubits used.)

8.5 State-of-the-art methods and caveats

There are various technical assumptions we made that can be dropped:[9]

[9] Pedro C.S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babbush, and Dominic W. Berry. Optimal scaling quantum linear-systems solver via discrete adiabatic theorem. *PRX Quantum*, 3:040303, 2022

- The normalization $\|A\|_\infty = 1$ and the positivity $A > 0$, e.g., extending to Hermitian $A = A^\dagger$.
- The exact d-bit binary representation of the eigenvalues $\{\lambda_j\}$ and the d-bit binary representation of the angles $\{\theta_j\}$.

- The QLSS outputs a quantum state of the normalized solution vector, but if the normalization is also needed, it can be determined at comparable cost as described above.

Based on the most recent *adiabatic methods* (cf. your explorations from Exercise Sheet 8), the complexity of QLSSs can be brought down to

$$\mathcal{O}\left(\kappa(A)\log\left(\varepsilon^{-1}\right)\right) \tag{8.29}$$

many queries to a *block encoding* data access to A. This access can be constructed from the oracles $U_{A,\text{entry}}$, $U_{A,\text{pos}}$ and its inverse with only small overhead costs (see the last Chapter 10). The complexity in Eq. (8.29) is then also optimal.

However, for estimating the norm of the solution vector, there is a lower bound of $\Omega(\kappa(A)\varepsilon^{-1})$, showing that poly-logarithmic scaling in the approximation error is not possible for this task.[10] There are various further caveats that stay with QLSSs and any potential application must take these into account:

[10] Alexander M. Dalzell. A shortcut to an optimal quantum linear system solver. 2024. URL http://arxiv.org/abs/2406.12086

- In order to create the oracles U_b, $U_{A,\text{entry}}$, and $U_{A,\text{pos}}$ based on quantum random access memory, one needs additional memory qubits that scale at least with the sparsity. The cost of this can be significant and all dominating.
- The quantum state $|y\rangle$ outputted by the QLSS does not directly reveal any classical information about the solution of the linear system of equations. If one is for example to measure an overlap $\langle\phi|x\rangle$ with some fixed n-qubit state $|\phi\rangle$, an additional sampling complexity of $\mathcal{O}(\varepsilon^{-2})$ is needed, bringing the end-to-end cost of the state-of-the-art QLSS from Eq. (8.29) to

$$\widetilde{\mathcal{O}}\left(\kappa(A)\varepsilon^{-2}\right), \tag{8.30}$$

  ignoring poly-logarithmic terms. If one does want full information about the (normalized) classical solution vector $\vec{x}$, one can show that the overall scheme scales as (see, e.g.,[11])

$$\widetilde{\mathcal{O}}\left(N\kappa(A)\varepsilon^{-2}\right), \tag{8.31}$$

  which, unfortunately, again picks up the exponential dependence N. Note that this is then also (nearly) matched by randomized classical solvers.
- Generic matrices are not well conditioned and as such $\kappa(A)$ can become very large, rendering QLSSs useless. So, similarly as in the classical case, one should therefore heavily use linear system pre-conditioners.

[11] Alexander M. Dalzell, B. David Clader, Grant Salton, Mario Berta, Cedric Yen-Yu Lin, David A. Bader, Nikitas Stamatopoulos, Martin J. A. Schuetz, Fernando Brandão, Helmut G. Katzgraber, and William J. Zeng. End-to-end resource analysis for quantum interior-point methods and portfolio optimization. *PRX Quantum*, 4:040325, 2023a

Given all of the above, full end-to-end complexities of QLSS for applications remain largely unclear, and have to be explored

further. Regarding this, we recently proposed an alternative hybrid *randomized* QLSS that only uses a classical data structure and completely avoids the use of quantum random access memory, whenever the matrix A is given in the Pauli basis (which is, e.g., useful for Hamiltonians and computing corresponding Green's functions).[12]

[12] Samson Wang, Sam McArdle, and Mario Berta. Qubit-efficient randomized quantum algorithms for linear algebra. *PRX Quantum*, 5:020324, 2024

9
Quantum random access memory (QRAM)

9.1 Motivation

Quantum algorithms operating on classical data, such as QLSSs discussed in Chapter 8, typically assume that the data is readily available by loading it into the quantum processing unit (QPU). The complexity of this step is then often quantified in terms of the number of queries to such a classical data loading routine. However, for true end-to-end complexities, explicitly constructing data loading oracles for unstructured classical data sets, comes at a cost — both in terms of quantum gate complexity, and space cost in terms of the qubit count. In the literature, the term *quantum random access memory* (QRAM) is used with various meanings, but in the following, we understand it generally as a construction that enables *coherent access to classical data*, such that multiple different elements in a classical database can be read in superposition.[1]

[1] Similar constructions are available for loading and operating on quantum data, but this is beyond the scope of the lecture.

The main example tasks we are interested in are *quantum state preparation* from data vectors (Sections 9.2 – 9.3) and *loading matrix elements into quantum states* (Sections 9.4 – 9.6). For a given classical data set of certain size and structure, we are then interested in the qubit cost, the quantum gate cost, and the quantum gate *depth* required to perform these data loading operations. The gate depth thereby corresponds to how many *layers* of quantum gates — executed in parallel — it takes to complete the task. It is defined as the *longest path* in the quantum circuit, and for architectures that easily allow to run quantum gates on different qubits in parallel, depth corresponds to *time complexity*. Note that there is typically a trade-off between space cost and time complexity.

9.2 Quantum state preparation: Basic ideas

Here, we consider the task of generating an n-qubit state

$$|b\rangle = \frac{1}{\left\| \sum_{i=0}^{2^n-1} b_i |i\rangle \right\|_2} \cdot \sum_{i=0}^{2^n-1} b_i |i\rangle \tag{9.1}$$

from a given classical data vector $\vec{b} = (b_0, \cdots, b_{2^n-1})$ of length 2^n. This is a necessary sub-routine for LCU Hamiltonian simulation

as introduced in Section 6.4, as well as for the QLSSs discussed in Chapter 8. Note the exponential compression of the classical memory space 2^n to only n qubits.

Now, to create $|b\rangle$ one needs by definition at least n-qubits, but one might also employ (plentiful) ancilla qubits. In Exercise Sheet 11, you will explore a *hard-coded* n-qubit unitary U_b that directly prepares $|b\rangle$ with $\mathcal{O}(n)$ ancilla qubits and $\mathcal{O}(n \cdot 2^n)$ many gates.[2] This is already sufficient for the required LCU Hamiltonian simulation unitary V from Eq. (6.41), but for QLSS applications one would like to stay within poly(n) complexity to have the potential for large quantum speed-up.

[2] The scheme can be improved to no ancilla qubits and the optimal depth $\mathcal{O}(2^n/n)$ with more advanced methods.

Unfortunately, there are information-theoretic lower bounds dictating that any quantum circuit preparing the 2^n different amplitudes from Eq. (9.1) requires at least $\Omega(2^n)$ many gates.

While this is bad news, one can resort to quantum depth, which corresponds to the time it takes to prepare the quantum state. While aforementioned $\mathcal{O}(n \cdot 2^n)$ gates method also has $\mathcal{O}(n \cdot 2^n)$ depth, it is possible to create $|b\rangle$ with only $\mathcal{O}(n)$ depth — albeit at the cost of $\mathcal{O}(2^n)$ many ancilla qubits and still with $\mathcal{O}(2^n)$ total gate count.[3] This is then more similar to classical data structures, that can load entries fast, i.e., in logarithmic depth, at the price of a large linear space cost. In the remainder of this section, we discuss the main ideas of the $O(n)$ depth construction following the presentation in.[4]

[3] If the state $|b\rangle$ is s-sparse, meaning that only s of the coefficients $\{b_i\}$ are non-zero, then advanced methods lead to the greatly improved $\mathcal{O}(\log(ns))$ depth with only $\mathcal{O}(ns \log s)$ ancilla qubit space cost.

[4] B. David Clader, Alexander M. Dalzell, Nikitas Stamatopoulos, Grant Salton, Mario Berta, and William J. Zeng. Quantum resources required to block-encode a matrix of classical data. *IEEE Transactions on Quantum Engineering*, 3:1, 2022

For simplicity, we assume that the coefficients $\{b_i\}$ are real and positive (this is not essential), and use the notation

$$\vec{b} = [b_0, \cdots, b_{2^n-1}] \text{ and } b_u^v = [b_u, \cdots, b_{v-1}] \text{ for } v > u. \tag{9.2}$$

The first step is to arrange the data into a binary tree structure of depth n, in a fashion where the amplitudes b_i^2 are stored in the leaf notes of the tree. This is depicted in Figure 9.1.

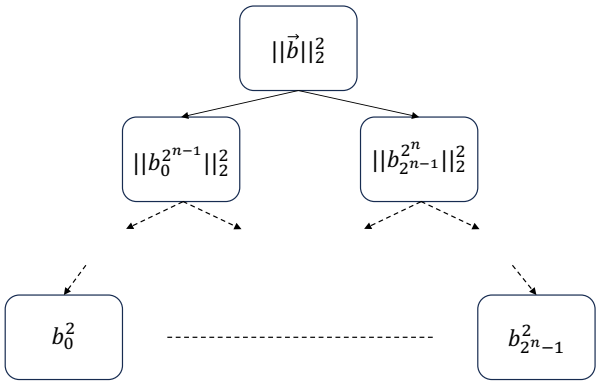


Figure 9.1: Binary tree data structure with n levels, arranging the coefficients $[b_0, \cdots, b_{2^n-1}]$ of the vector $\vec{b}$.

Based on this tree data, one can then construct the n-qubit state $|b\rangle$ in low depth with the following steps:

- The state is initialized to $|0^{\otimes n}\rangle$.
- On the first qubit, a Y-Pauli rotation by the angle

$$\theta_1 = 2\arccos\left(\frac{\left\|b_0^{2^{n-1}}\right\|_2}{\left\|\vec{b}\right\|_2}\right) \tag{9.3}$$

is applied. This creates the state

$$|b(1)\rangle = \Big(\cos(\theta_1/2)|0\rangle + \sin(\theta_1/2)|1\rangle\Big) \otimes |0^{\otimes(n-1)}\rangle\,, \tag{9.4}$$

where θ_1 can be computed from the values stored at the root of the binary tree and its children.

- On the second qubit, a Y-Pauli rotation by the angle

$$\theta_2 = 2\arccos\left(\frac{\left\|b_0^{2^{n-2}}\right\|_2}{\left\|b_0^{2^{n-1}}\right\|_2}\right) \text{ or } \theta_3 = 2\arccos\left(\frac{\left\|b_{2^{n-1}}^{3\cdot 2^{n-2}}\right\|_2}{\left\|b_{2^{n-1}}^{2^n}\right\|_2}\right) \tag{9.5}$$

is applied, where θ_2 is used conditioned on the first qubit being in state $|0\rangle$, and θ_3 is used conditioned on the first qubit being in state $|1\rangle$. This creates the state

$$\begin{aligned}|b(2)\rangle = \cos(\theta_1/2)|0\rangle\Big(\cos(\theta_2/2)|0\rangle + \sin(\theta_2/2)|1\rangle\Big) \otimes |0^{\otimes(n-2)}\rangle \\ + \sin(\theta_1/2)|1\rangle\Big(\cos(\theta_3/2)|0\rangle + \sin(\theta_3/2)|1\rangle\Big) \otimes |0^{\otimes(n-2)}\rangle\,,\end{aligned} \tag{9.6}$$

where θ_2 and θ_3 can each be computed from values stored at one level two node and its children.

- Extrapolating steps 1 and 2, the state after step k is given by

$$|b(k)\rangle = \sum_{y\in\{0,1\}^k} a_y|y\rangle \otimes |0^{\otimes(n-k)}\rangle\,, \tag{9.7}$$

where $a_y = \|b_{y\cdot 2^{n-k}}^{(y+1)\cdot 2^{n-k}}\|_2 \cdot \|\vec{b}\|_2^{-1}$ and the multiplication with y and $(y+1)$ is understood with respect to their binary representations.

- After the final n-th iteration, the state becomes

$$|b(n)\rangle = \sum_{y\in\{0,1\}^n} a_y|y\rangle = \sum_{y\in\{0,1\}^n} \frac{b_y}{\|\vec{b}\|_2}|y\rangle = \sum_{i=0}^{2^n-1} \frac{b_i}{\|\vec{b}\|_2}|i\rangle = |b\rangle\,. \tag{9.8}$$

To comprehend the last two bullet points, notice that we can rewrite the second level state as

$$|b(2)\rangle = \Big(a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle\Big) \otimes |0^{\otimes(n-2)}\rangle \tag{9.9}$$

for the values

$$a_{00} = \cos(\theta_1/2)\cos(\theta_2/2) = \left\| b_0^{2^{n-2}} \right\|_2 \cdot \left\| \vec{b} \right\|_2^{-1} \tag{9.10}$$

$$a_{01} = \sin(\theta_1/2)\sin(\theta_2/2) = \left\| b_{2^{n-2}}^{2^{n-1}} \right\|_2 \cdot \left\| \vec{b} \right\|_2^{-1} \tag{9.11}$$

$$a_{10} = \sin(\theta_1/2)\cos(\theta_3/2) = \left\| b_{2^{n-1}}^{3\cdot 2^{n-2}} \right\|_2 \cdot \left\| \vec{b} \right\|_2^{-1} \tag{9.12}$$

$$a_{11} = \sin(\theta_1/2)\sin(\theta_3/2) = \left\| b_{3\cdot 2^{n-2}}^{2^n} \right\|_2 \cdot \left\| \vec{b} \right\|_2^{-1} . \tag{9.13}$$

Consequently, the transition from $|b(k)\rangle$ to $|b(k+1\rangle$ is of the form

$$|b(k)\rangle = \sum_{y\in\{0,1\}^k} a_y |y\rangle \otimes |0^{\otimes(n-k)}\rangle \tag{9.14}$$

$$\mapsto \sum_{y\in\{0,1\}^k} a_y |y\rangle \left(\frac{a_{y0}}{a_y}|0\rangle + \frac{a_{y1}}{a_y}|1\rangle \right) \otimes |0^{n-k-1}\rangle \tag{9.15}$$

$$= \sum_{y\in\{0,1\}^{k+1}} a_y |y\rangle \otimes |0^{n-k-1}\rangle \tag{9.16}$$

$$= |b(k+1)\rangle . \tag{9.17}$$

That is, a rotation to qubit $(k+1)$ by an angle

$$\theta_{1y} = 2\arccos\left(\frac{a_{y0}}{a_y}\right) \tag{9.18}$$

is applied, where the subscript $1y$ is read in binary representation.

The scheme as presented has n layers and it remains to give explicit quantum circuits to implement it in terms of elementary quantum gates. We do so in the next section, including estimates of the complexity in terms of qubit count, depth, and total gate count.

9.3 *Quantum state preparation: Circuits*

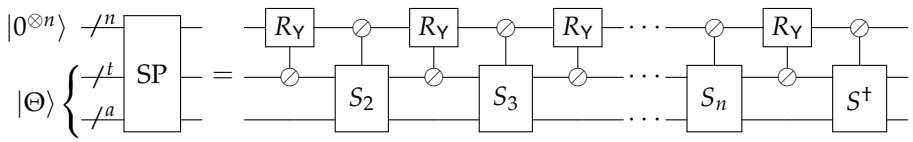
Given a binary tree data structure as depicted in Figure 9.1 with $2^{n+1}-1$ nodes, we assume that the angles θ_i for $i = 1, \cdots, 2^n - 1$ are classically pre-computed to avoid the need for any arithmetic on the quantum computer. We further assume that the $2^n - 1$ angles have an exact t-bit representation and are stored in $(2^n-1)t$ ancilla qubits. Then, the starting state on $n + (2^n-1)t$ qubits is given by

$$|0^{\otimes n}\rangle \otimes |\Theta\rangle \text{ with } |\Theta\rangle = \bigotimes_{i=1}^{(2^n-1)t} |\theta_i\rangle, \tag{9.19}$$

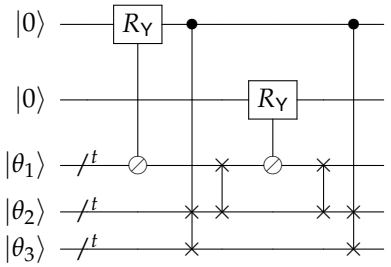
where the latter is also a computational basis state that can easily be created by a single layer of Pauli X gates on the appropriate qubits determined by the classical data angles. The goal is to implement the *state preparation* routine

$$\mathsf{SP}\Big(|0^{\otimes n}\rangle \otimes |\Theta\rangle\Big) = |b\rangle \otimes |\Theta\rangle , \tag{9.20}$$

which is accomplished by the following quantum circuit



Here, $a = (2^n - 2)t$ denotes all but t ancilla qubits, the symbol $\oslash$ indicates that a different rotation angle is performed for each of the 2^t possible settings of the register, S_p for $p = 2, \cdots, n$ swaps the t-bit description of the next angle into the first ancilla register of size t, and $S^\dagger = S_2^\dagger \cdots S_n^\dagger$ is resetting the ancilla qubits. For example, for $n = 2$ this takes the form



Note that the t-bit controlled R_Y rotation exactly corresponds to the U_θ gate from Eq. (8.25) that you constructed in Exercise Sheet 10.

Algebraically, for each basis state $|j\rangle$, a different sequence of n angles is applied, and the corresponding swap networks $S_2, \cdots, S_n$ can be written as

$$S_p = \sum_{j=0}^{2^n-1} |j\rangle\langle j| \otimes S_p^{(j)}\,, \tag{9.21}$$

where $S_p^{(j)}$ acts on the ancilla qubit registers such that the product

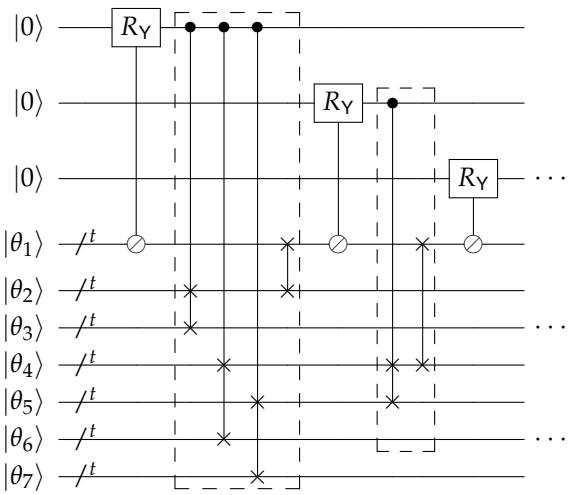
$$S_p^{(j)} S_{p-1}^{(j)} \cdots S_1^{(j)} \tag{9.22}$$

has the action of swapping the t-bit description of the angle associated with the p-th rotation for $|j\rangle$ in the first t-qubit ancilla register. Note that S_p is controlled only by the first $p-1$ bits of $|j\rangle$.

Extrapolating from the $n = 2$ case, a straightforward way to implement S_p is to first reverse the action of S_{p-1} in depth $\mathcal{O}(p)$ to swap out the $(p-i)$-th angle, and second, to swap in the correct angle in depth $\mathcal{O}(p)$. However, this would lead to the total depth

$$\sum_{p=1}^{n} \mathcal{O}(p) = \mathcal{O}(n^2)\,, \tag{9.23}$$

while we are aiming for $\mathcal{O}(n)$. The latter can be accomplished by avoiding to undo the work already accomplished by S_{p-1}, and using that S_p can be controlled on just one of the n main qubits. An example circuit for $n = 3$ is shown below, with the inverse operation $S^\dagger$ omitted



In Exercise Sheet 11, you will work through a concrete example of state preparation with constant precision for $n = 2, 3, 4$ qubits.[5]

[5] Xiaoming Sun, Guojing Tian, Shuai Yang, Pei Yuan, and Shengyu Zhang. Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis. 2021. URL http://arxiv.org/abs/2108.06150

To conclude, when the precision t is set to constant, above quantum circuits of depth $\mathcal{O}(n)$ prepare the n-qubit state $|b\rangle$, at the price of $\mathcal{O}(2^n)$ total gate count and $\mathcal{O}(2^n)$ many ancilla qubits. For this we assumed that the relevant angles $\{\theta_i\}$ are pre-computed classically, but in principle this could also be done quantumly without affecting the asymptotic complexities. Lastly, for s sparse vectors $\vec{b}$, above methods can be greatly improved to depth $\mathcal{O}(\log(ns))$ using only $\mathcal{O}(ns \log s)$ ancilla qubits.

9.4 *Quantum read only memory (QROM)*

Given a data set $\{x_0, \cdots, x_{N-1}\}$ of x_i with exact t-bit representations and $N = 2^n$, we consider the task of implementing an $(n+t)$-qubit operation acting as

$$U_N : |i\rangle \otimes |0^{\otimes t}\rangle \mapsto |i\rangle \otimes |x_i\rangle\,. \tag{9.24}$$

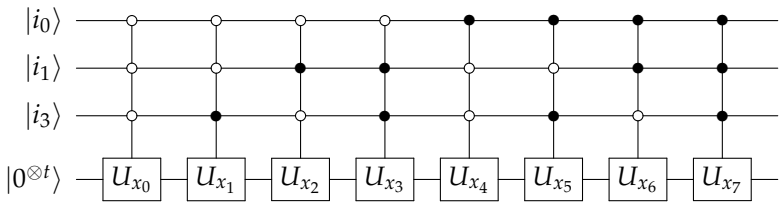
One way to achieve this in terms of a unitary matrix

$$U_N \left(\sum_{i=0}^{2^n-1} \sum_{j=0}^{2^t-1} \alpha_{ij} |i\rangle \otimes |j\rangle \right) = \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^t-1} \alpha_{ij} |i\rangle \otimes |j \oplus x_i\rangle\,. \tag{9.25}$$

We are then interested in the cost of implementing U_N in terms of depth, total gate cost, and qubit space cost. Note that when thinking of the data set $\{x_0, \cdots, x_{2^n-1}\}$ as coming from an $2^{n/2} \times 2^{n/2}$ matrix A, above task corresponds to constructing the sparse access model $U_{A,\text{entry}}$ oracle from Eq. (8.6), as used by QLSSs.[6]

A simple way to implement U_N with the use of n ancilla qubits — called the *address qubits* — is via *unary iteration*. An example for $n = 3$ with $i = i_0 i_1 i_2$ in binary representation is depicted below

[6] More precisely, this corresponds to the case of dense (non-sparse) matrices. We abstain for now from also implementing the non-zero position oracle $U_{A,\text{pos}}$ from Eq. (8.7), which is important for the case of sparse matrices.



which goes under the name *quantum read only memory* (QROM). The reason being that the circuit is hard-coded to the data set $\{x_0, \cdots, x_{2^n-1}\}$ via the fixed unitaries $\{U_{x_i}\}$ acting on the main, *bus qubits*. The idea is to iterate over all 2^n possible states of the address register, where the j-th gate transforms the bus qubit if the address register is in the state $|j\rangle$. This results in a cost of $\mathcal{O}(2^n)$ depth, $\mathcal{O}(2^n)$ gates, and $\mathcal{O}(n)$ ancilla qubits.

Unfortunately, this is not good enough yet for, e.g., QLSSs applications, where we would like to keep the depth, or in other words, time complexity, at poly(n). In the next two sections, we present two different schemes that achieve depth $\mathcal{O}(n)$, at the cost of $\mathcal{O}(2^n)$ many ancilla qubits and still with $\mathcal{O}(2^n)$ total gate count.

9.5 *Fanout QRAM*

Motivated from classical *random access memory* (RAM) structures, the simplest scheme that achieves the quantum data loading from Eq. (9.25) in depth $\mathcal{O}(n)$ is *fanout QRAM*. Its basic building blocks are *quantum routers* arranged in a binary tree, where the outputs at one level act as the inputs on the next level.

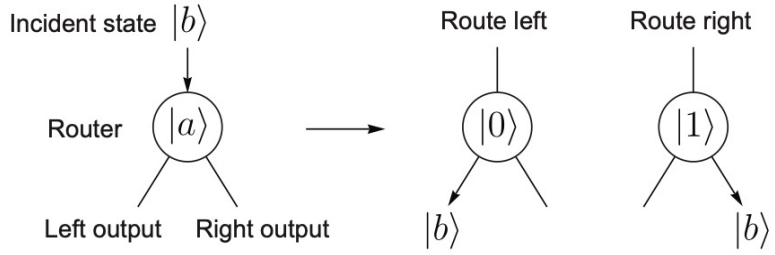
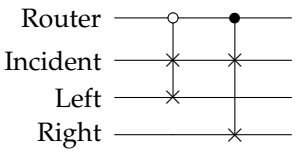


Figure 9.2: The quantum router building block of fanout QRAM.

As depicted in Figure 9.2 the quantum router takes an *incident* qubit $|b\rangle$ coming in from the top and directs it to left or right conditional on the state $|a\rangle$ of the router (the figures in this chapter are taken from[7]). The corresponding quantum circuit is



[7] Connor T. Hann, Gideon Lee, S.M. Girvin, and Liang Jiang. Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2:020311, 2021

In the following, we assume for simplicity that each element x_i is just a single bit, and follow the structure laid out in Figure 9.3 to query an x_i in the data structure:

- First, all routing qubits are initialized to zero and n address qubits are set to the desired state.
- Second, all routing qubits at level k are flipped from zero to one, depending on the corresponding k-th address qubit.
- Third, a *bus qubit* is prepared in the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and injected from the top. Following the path designated by the router states, the bus qubit arrives at the corresponding classical memory cell at the bottom.
- Fourth, the classical data bit x_i is encoded into the bus qubit by applying $\bigotimes_{i=0}^{2^n-1} \mathsf{Z}^{x_i}$ to the output modes of all routers at the bottom level of the tree. If $x_i = 1$ the bus qubit is mapped to $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, while all other qubits at the bottom are in state $|0\rangle$ and hence unaffected by Z. Note that because the location of the bus is not a priori known, classically controlled Z gates have to be applied at all bottom locations.
- Fifth, the bus qubit is routed back out of the tree, following the same path backwards, and all routers are set back to zero.

For a structure of size $n = 3$, a full example circuit to query the element x_5 at the corresponding binary address 101 is given in Figure 9.4. Crucially, and as required by Eq. (9.25), this scheme equally allows to query multiple memory elements in superposition—just as all elements work coherently.

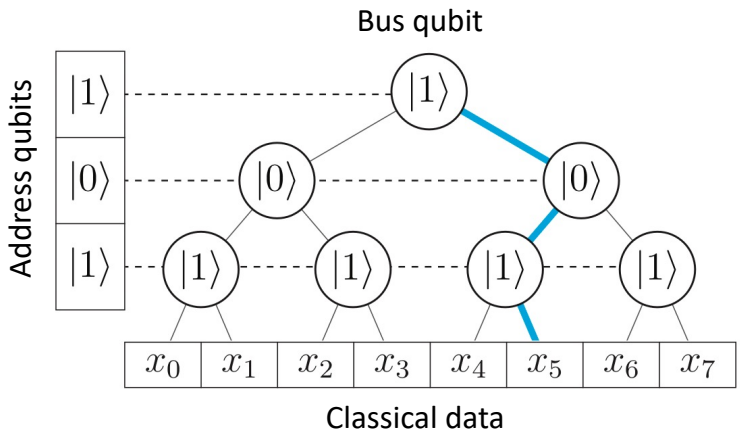
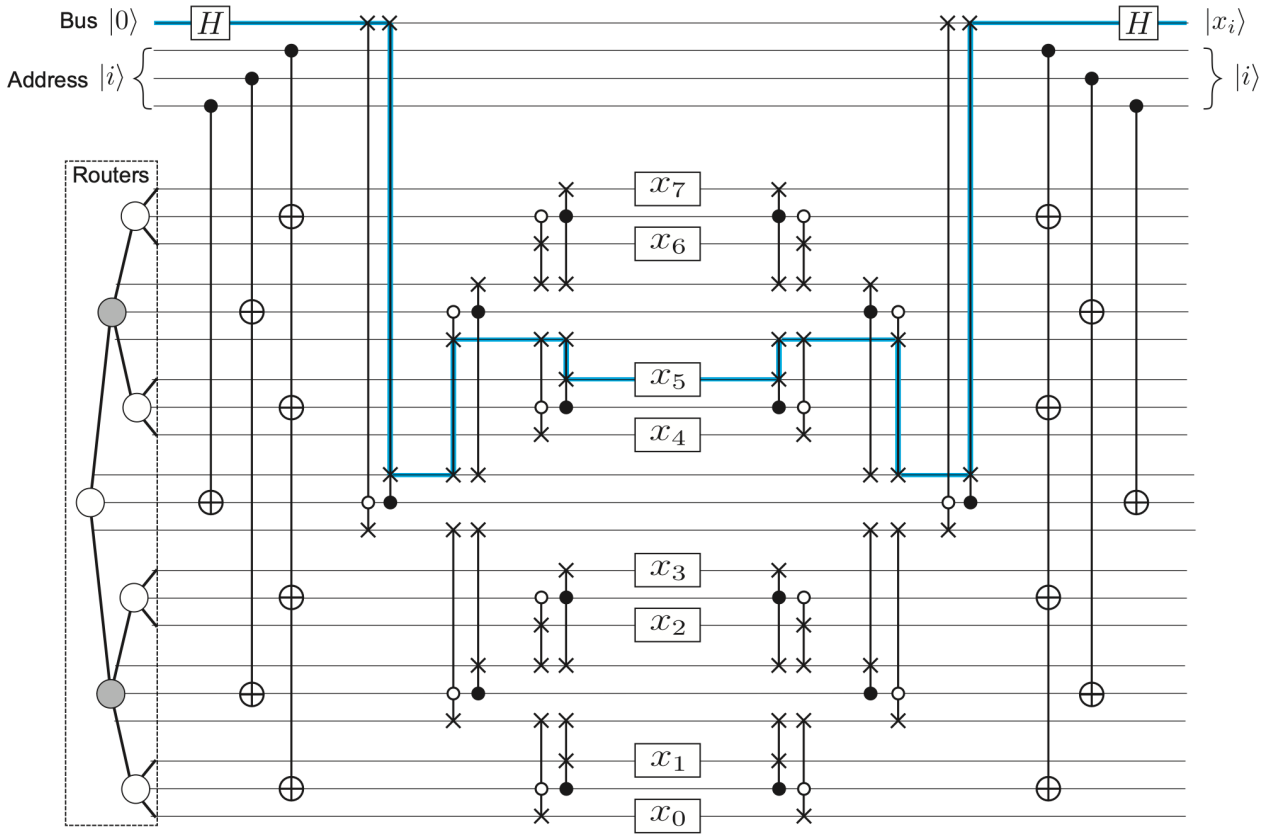


Figure 9.3: Fanout QRAM data structure with $n = 3$ for querying the element x_5 at position 101.

As the binary tree has n layers, the depth of the scheme is $\mathcal{O}(n)$, while the number of qubits and total gate count are both $\mathcal{O}(2^n)$. Overall, this fanout QRAM architecture works fine in theory, but it is somewhat unpractical, as all $\mathcal{O}(2^n)$ memory qubits have to be kept coherent during the whole query process (even though some of them do not take part in the routing).

In the next section, an alternative architecture is presented that does not suffer from this weakness.

Figure 9.4: Fanout QRAM circuit.



9.6 *Bucket-brigade QRAM*

As mentioned, the fanout QRAM structure has high susceptibility to decoherence, and since the memory qubit cost of $\mathcal{O}(2^n)$ is typically exponentially larger than the poly(n) algorithmic qubits needed by most algorithms employing QRAM, there is a strong incentive to design more noise resilient QRAM structures. That is, architectures potentially not requiring the same quantum error correction overhead as the algorithmic qubits. This is also in analogy to classical RAM data structures, which are based on highly specialized hardware, different from the architecture of the central processing unit (CPU).

Such constructions are accomplished by *bucket-brigade* QRAM. In their simplest form, the idea is to replace the routing qubits with two qubits, that feature the now three relevant states, the *active* $|0\rangle$ (route left) and $|1\rangle$ (route right), as well as the *inactive* $|w\rangle$ (wait).[8] One then initializes all routers in the $|w\rangle$ state, and assumes that the routing operation is trivial when the routing qubits are in the $|w\rangle$ state.

The main algorithmic modification compared to fanout QRAM is then that the address qubits are themselves routed into the tree during a query. Namely, for a router state $|0\rangle$ / $|1\rangle$, the address

[8] The fourth state does not matter for our considerations. Even though we do not ever make use of this fourth state, any operations are defined on the two qubits such that the overall action is unitary.

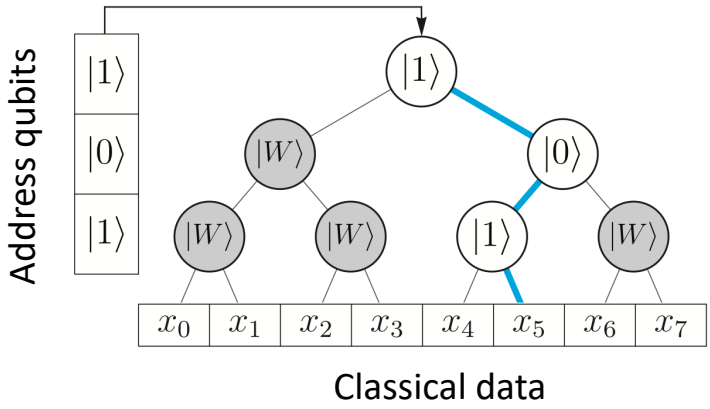


Figure 9.5: Bucket-brigage QRAM data structure with $n = 3$ for querying the element x_5 at position 101.

qubits is routed left / right as before, but for the router state $|w\rangle$, the address qubit is swapped in, such that the state of the router becomes $|0\rangle$ / $|1\rangle$. The structure of a memory query is depicted in Figure 9.5 by means of an example.

In more detail, and again assuming that each element x_i is just a single bit, the address qubits are put into the tree one by one through the root node. The first one is swapped into the first router, changing the router's state from $|w\rangle$ to $|0\rangle$ / $|1\rangle$. This state then determines the routing of the second address qubit put into the tree, after which it is swapped into a router at the second level. After all address qubits are loaded into the routers, the bus qubit is routed through the tree, and the specified element x_i is loaded from classical memory (which works slightly differently than in the fanout architecture). As the final step, the bus and address qubits are routed back in reverse.

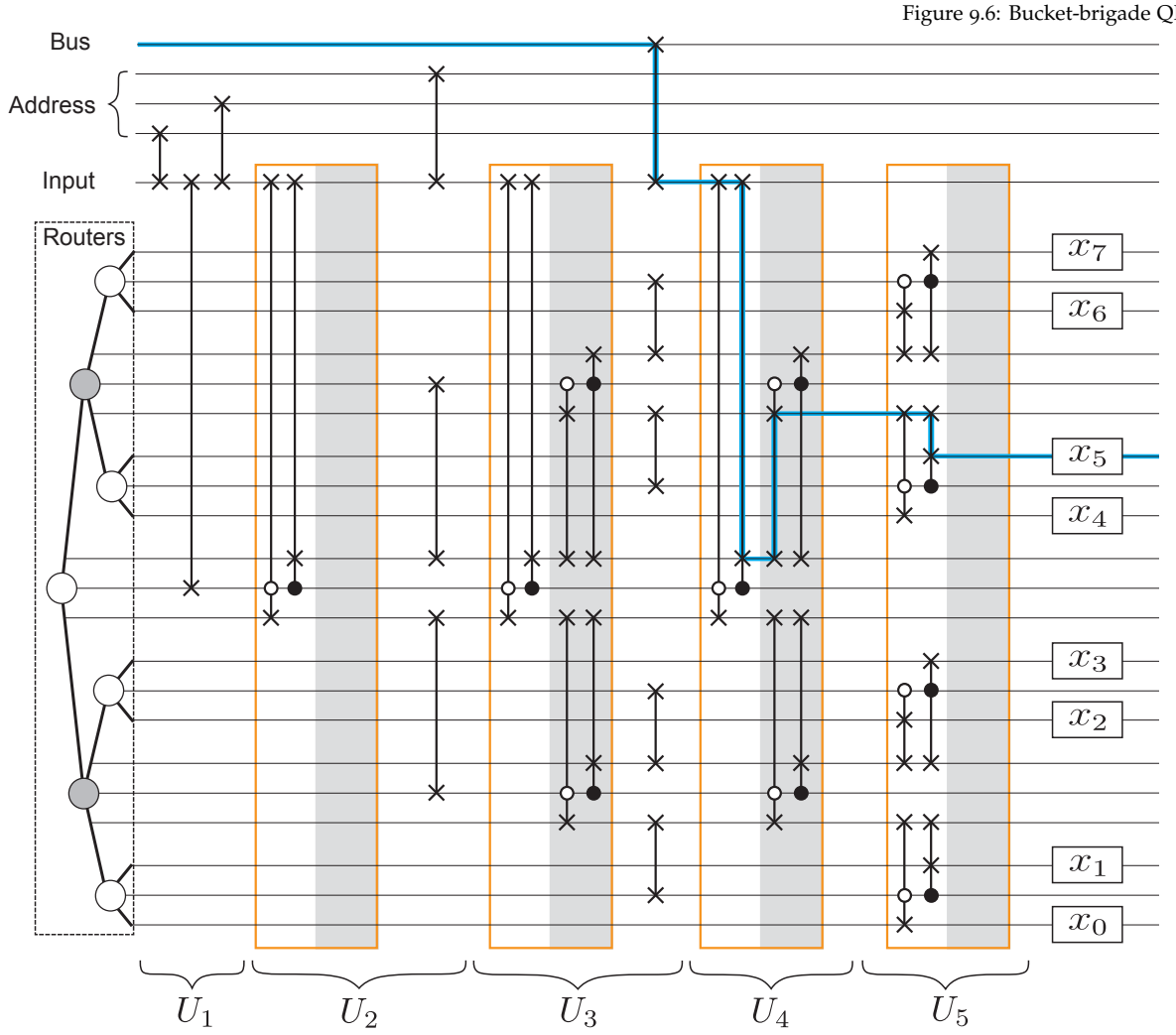
For a structure of size $n = 3$, an example circuit to query the element x_5 at the corresponding binary address 101 is given in Figure 9.6 (with the final unloading step omitted). In there, the steps U_1, U_2, U_3 route the address qubits into the tree, whereas U_4, U_5 route the bus qubit to the memory cell. You will work trough some of the details of the construction in Exercise Sheet 12.

9.7 *Extensions and caveats*

In addition to the presented QROM, fanout QRAM, and bucket-brigade QRAM architectures, there are other proposals such as *QROAM*, which allows for arbitrary space versus depth trade-off in the costs. Moreover, versions for sparse classical data sets featuring the $U_{A,\text{pos}}$ oracle can also be built at significantly lower cost scaling with the sparsity of the data set. However, this is beyond the scope of the lecture.

In general, if one could build fault-tolerant quantum computers,

Figure 9.6: Bucket-brigade QRAM.



one could also build QRAM data structures with the same hardware. However, for typical applications the space requirements of QRAM are exponentially larger than the number of algorithmic qubits, and as such there is the need to develop specialized quantum hardware for the design of QRAM circuits.[9] As QRAM circuits do not require a universal set of quantum gates — for example the bucket-brigade architectures exclusively employs controlled swap gates — there are then indications that there is some inbuilt noise resistance. This could lead to lower quantum error correction overheads than required for universal circuits, and is an important research direction to realize the power of quantum algorithms requiring QRAM.[10]

Lastly, it should also be said that given current quantum hardware proposals, the promise of full parallelization with depth as the only relevant metric for time complexity is optimistic to say the least.

[9] This is also important in comparison with classical hardware, as, e.g., poly(N) CPUs allow to do matrix multiplication in depth $\mathcal{O}(\log N)$. Note that when building universal quantum computers with N qubits, one might very well already require poly(N) classical control CPUs for managing the quantum error correction overhead.

[10] Connor T. Hann, Gideon Lee, S.M. Girvin, and Liang Jiang. Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2:020311, 2021

10
Quantum singular value transform (QSVT)

10.1 Motivation

In this lecture, we have treated various quantum algorithms, ranging from the quantum Fourier transform (Chapter 4), to quantum phase estimation (Chapters 5 and 7), to Hamiltonian simulation (Chapter 6), to linear system solvers (Chapter 8). Further, we have also seen different ways of loading the relevant input to these algorithms into a quantum computer (Chapter 9). While all these different methods are at first unrelated and developed at hoc, they have recently been shown to be part of a general framework termed *quantum singular value transformation* (QSVT), which emerged from a longer series of papers.[1]

[1] András Gilyén, Yuan Su, Guang Hao Low, and Nathan 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*, STOC 2019, page 193, 2019

The starting point is the simple realization that quantum mechanics only allows to encode data into quantum states and process them via unitary operations — while general data is given by general matrices and one wants to process these with general functions (and might even need coherent access to these non-unitary transformations). The QSVT framework allows to embed any data of non-unitary matrices into larger unitary matrices, represented by an efficient quantum circuit, and then do arithmetics such as implementing general matrix functions. Crucially, all of the aforementioned quantum algorithms can then be understood as special cases of this framework. Notably, for some tasks, such as, e.g., Hamiltonian simulation and quantum linear system solvers, this also leads to conceptually different implementations and improved complexities.

Note that we refrain in the following from giving any proofs or even full technical statements. Rather, we just briefly pitch the main results, and provide follow-up references for the interested reader.

10.2 Block encoding data access

A *block encoding* of a complex $M \times M$ matrix A is given by a unitary matrix U_A with[2]

[2] Non-square matrices can easily be treated as well.

$$U_A = \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix}, \tag{10.1}$$

where the sub-matrices labelled $(\cdot)$ are irrelevant as long as U_A is unitary, and α is a normalization factor (which is in general necessary).

More precisely, we say that the unitary U_A is a (α, a)-block encoding of the complex $M \times M$ matrix A if

$$A = \alpha \cdot \left(\langle 0|^{\otimes a} \otimes \mathbb{1}_M\right) U_A \left(|0\rangle^{\otimes a} \otimes \mathbb{1}_M\right) , \tag{10.2}$$

where $\alpha \geq \|A\|_\infty$, and a denotes the number of ancilla qubits used for the embedding.

Additionally, in practice one sometimes needs more general *approximate* (α, a, ε)-block encodings, which are defined as

$$\left\| A - \alpha \cdot \left(\langle 0|^{\otimes a} \otimes \mathbb{1}_M\right) U_A \left(|0\rangle^{\otimes a} \otimes \mathbb{1}_M\right) \right\|_\infty \leq \varepsilon , \tag{10.3}$$

where the approximation holds up to $\varepsilon \geq 0$. Given block encodings U_A and U_B, one can easily give efficient constructions of block encodings U_{A+B} of $A + B$ and U_{AB} of AB (with essentially additive costs).[3]

[3] András Gilyén, Yuan Su, Guang Hao Low, and Nathan 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*, STOC 2019, page 193, 2019

Now, how to construct block encodings U_A heavily depends on the structure of the matrix A and how access to it is given (which in general requires QRAM). Some example constructions include:

- A *Gram matrix* A with $(A)_{ij} = \langle \psi_i | \phi_j \rangle$ can be $(1, a, 0)$-block encoded with state preparation unitaries

  $$U_L\Big(|0^{\otimes a}\rangle \otimes |i\rangle\Big) = |\psi_i\rangle \quad \text{and} \quad U_R\Big(|0^{\otimes a}\rangle \otimes |j\rangle\Big) = |\phi_j\rangle \tag{10.4}$$

  via $U_A = U_R^\dagger U_L$

- An $s(A)$ row and column *sparse matrix* A of dimension $2^n \times 2^n$ can be $(s(A), n + 3, \varepsilon)$-block encoded by querying the sparse oracles from Eqs. (8.6) – (8.7) a constant number of times.[4]

[4] Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. Black-box quantum state preparation without arithmetic. *Physical Review Letters*, 122:020502, 2019

- A linear combination of unitaries $A = \sum_{l \in L} \alpha_l P_l$ as given in Eq. (6.40) can be $(\|\vec{\alpha}\|_1, \log L, 0)$-block encoded by using the unitaries V and W from Eqs. (6.41) and (6.42), respectively, via $U_A = V^\dagger W V$.

- When A has some algebraic or arithmetic structure, various refined constructions are known.[5]

[5] Daan Camps, Lin Lin, Roel Van Beeumen, and Chao Yang. Explicit quantum circuits for block encodings of certain sparse matrices. 2022. URL http://arxiv.org/abs/2203.10236; and Christoph Sunderhauf, Earl Campbell, and Joan Camps. Block-encoding structured matrices for data input in quantum computing. 2023. URL http://arxiv.org/abs/2302.10949

Note that block encodings of general (non-sparse) $M \times M$ matrices will require $\mathcal{O}(M^2)$ qubits in order to achieve $\mathcal{O}(\log M)$ depth cost (cf. Chapter 9).

10.3 *Transformation of block encodings*

The QSVT is essentially a quantum circuit that takes as an input a block encoding U_A of a matrix A and outputs a block encoding U_{A^d} of the matrix A^d for $d \in \mathbb{N}$. The cost is roughly given by $\mathcal{O}(d)$

applications of U_A and $\mathcal{O}(d)$ further elementary gates, as well as a constant number of ancilla qubits. General function transformations $f(A)$ other than monomials A^d can be achieved by implementing *polynomial approximations* of the function $f(\cdot)$ on the interval of reference.

In order to make this more precise, we first need to define what it means to apply general functions to matrices. This can be achieved with the *singular value decomposition*. For a complex $M \times M$ matrix A, it is defined as factorization of A such that

$$A = U\Sigma V^\dagger\,, \tag{10.5}$$

where U and V are unitary matrices, and Σ is a diagonal matrix composed of the non-negative real singular values $\{\sigma_i\}_{i=1}^M$ that are uniquely determined by A. A scalar function $f : \mathbb{R} \to \mathbb{C}$ is then lifted to a matrix function via

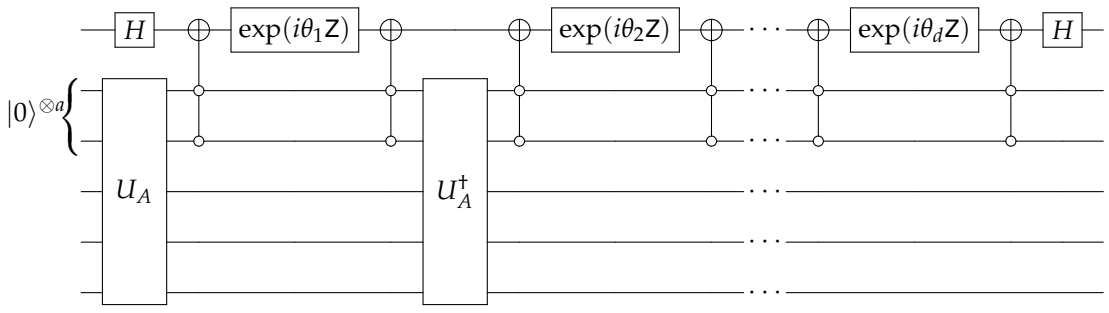
$$f(A) = Uf(\Sigma)V^\dagger\,, \tag{10.6}$$

where $f(\Sigma)$ is the diagonal matrix composed of the entries $\{f(\sigma_i)\}_{i=1}^M$. Note that if A is Hermitian, then $U = V$ and the singular value decomposition corresponds to the eigendecomposition of A. For simplicity we restrict in the following to Hermitian matrices A with $M = 2^m$, which is then also simply known as the *quantum eigenvalue transform* (QET).

Now, for a real, definite parity polynomial

$$p_d(A) : [-1,1] \to [-1,1] \text{ with } \sup_{x\in[-1,1]} |p(x)| \leq 1 \tag{10.7}$$

of degree d, the QSVT circuit takes the exact form



featuring one ancilla qubit, and where the rotation angle set $\{\theta_1, \cdots, \theta_d\}$ specific to $p_d(A)$ has to be classically pre-computed. This can be done efficiently, both in theory and in practice.[6] Moreover, general polynomials $p(A) : [-1,1] \to [-1,1]$ with $\sup_{x\in[-1,1]} |p(x)| \leq 1$ can be handled by taking linear combinations of unitaries. Overall, one finds the following theorem.

[6] Jeongwan Haah. Product decomposition of periodic functions in quantum signal processing. 2018. URL http://arxiv.org/abs/1806.10236; and Rui Chao, Dawei Ding, Andras Gilyen, Cupjin Huang, and Mario Szegedy. Finding angles for quantum signal processing with machine precision. 2020. URL http://arxiv.org/abs/2003.02831

Let U_A be a (α, a)-block encoding of a Hermitian matrix A and

$$p : [-1,1] \to \left\{ c \in \mathbb{C} : |c| \leq \frac{1}{4} \right\} \tag{10.8}$$

be a degree d polynomial. Then, QSVT gives a $(\alpha, \mathcal{O}(a))$-block encoding $U_{p(A)}$ of $p(A)$ using

- d applications of U_A and $U_A^\dagger$
- one controlled application of U_A
- $\mathcal{O}(ad)$ other elementary quantum gates.

The proof requires advanced tools from quantum computing and is based on ideas from *quantum signal processing* and *qubitization*. We refer the interested reader to the review article.[7]

[7] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2:040203, 2021

10.4 Example polynomials

In order to use the QSVT in applications, one needs to find low degree polynomial approximations of functions on relevant intervals. For this, the theory of Chebyshev polynomials is highly useful.[8]

[8] Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends® in Theoretical Computer Science*, 9:125, 2014

A most transparent example are quantum linear system solvers (Chapter 8). Here, after appropriate normalization, the relevant function is the inverse function over the interval

$$[-1, -\kappa(A)^{-1}] \cup [\kappa(A)^{-1}, 1] , \tag{10.9}$$

where $\kappa(A)$ denotes the condition number of the matrix A from the linear system $A\vec{x} = \vec{b}$. Using advanced tools from polynomial approximation theory it can be shown that a degree $d = \mathcal{O}\left(\kappa^2 \log(\kappa/\varepsilon)\right)$ is sufficient for an ε-approximation.[9] This outperforms our basic QLSS from Section 8.4, but still loses against the adiabatic solvers mentioned in Section 8.5 that have a $\mathcal{O}\left(\kappa \log(1/\varepsilon)\right)$ query complexity to U_A.[10]

[9] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46:1920, 2017

[10] Using *variable time quantum amplitude amplification* the QSVT based QLSS can be boosted to $\mathcal{O}\left(\kappa \log(\kappa/\varepsilon)\right)$.

Another prominent example is Hamiltonian simulation (Chapter 6). Here, after appropriate normalization and other technical simplifications, the complex exponential function is sought-after and one writes

$$\exp(ix) = \cos(x) + i \sin(x) , \tag{10.10}$$

where each term can now be separately approximated in terms of their truncated Jacobi-Anger expansion. This then leads to the optimal Hamiltonian simulation algorithm (up to a constant) with query complexity to the block encoding U_H of the Hamiltonian H as

$$O\left(|J|t + \frac{\log\left(\varepsilon^{-1}\right)}{\log\left(e + \frac{\log\left(\varepsilon^{-1}\right)}{|J|t}\right)} \right) \tag{10.11}$$

for simulation time t — as claimed in Section 6.5.

Note that for both of these examples, the QSVT circuits are conceptually different from what we previously discussed in the lecture. What quantum circuits perform best in the *early fault-tolerant regime* is a question of active research.

Bibliography

Scott Aaronson. Read the fine print. *Nature Physics*, 11:291, 2015.

Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error rate. *SIAM Journal on Computing*, 38:1207, 2008.

Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication, 2024. URL http://arxiv.org/abs/2404.16349.

Boaz Barak and Sanjeev Arora. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2007.

Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Physical Review Letters*, 114: 090502, 2015.

Earl Campbell. Random compiler for fast Hamiltonian simulation. *Physical Review Letters*, 123:070503, 2019.

Daan Camps, Lin Lin, Roel Van Beeumen, and Chao Yang. Explicit quantum circuits for block encodings of certain sparse matrices. 2022. URL http://arxiv.org/abs/2203.10236.

Rui Chao, Dawei Ding, Andras Gilyen, Cupjin Huang, and Mario Szegedy. Finding angles for quantum signal processing with machine precision. 2020. URL http://arxiv.org/abs/2003.02831.

Chi-Fang Chen, Michael J. Kastoryano, and András Gilyén. An efficient and exact noncommutative quantum Gibbs sampler. 2023. URL http://arxiv.org/abs/2311.09207.

Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46:1920, 2017.

Andrew M. Childs, Yuan Su, Minh C. Tran, Nathan Wiebe, and Shuchen Zhu. Theory of Trotter error with commutator scaling. *Physical Review X*, 11:011020, 2021.

B. David Clader, Alexander M. Dalzell, Nikitas Stamatopoulos, Grant Salton, Mario Berta, and William J. Zeng. Quantum resources required to block-encode a matrix of classical data. *IEEE Transactions on Quantum Engineering*, 3:1, 2022.

Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings Symposium on Theory of Computing*, STOC '71, page 151, 1971.

James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297, 1965.

Pedro C.S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babbush, and Dominic W. Berry. Optimal scaling quantum linear-systems solver via discrete adiabatic theorem. *PRX Quantum*, 3: 040303, 2022.

Alexander M. Dalzell. A shortcut to an optimal quantum linear system solver. 2024. URL http://arxiv.org/abs/2406.12086.

Alexander M. Dalzell, B. David Clader, Grant Salton, Mario Berta, Cedric Yen-Yu Lin, David A. Bader, Nikitas Stamatopoulos, Martin J. A. Schuetz, Fernando Brandão, Helmut G. Katzgraber, and William J. Zeng. End-to-end resource analysis for quantum interior-point methods and portfolio optimization. *PRX Quantum*, 4:040325, 2023a.

Alexander M. Dalzell, Sam McArdle, Mario Berta, Przemyslaw Bienias, Chi-Fang Chen, András Gilyén, Connor T. Hann, Michael J. Kastoryano, Emil T. Khabiboulline, Aleksander Kubica, Grant Salton, Samson Wang, and Fernando Brandão. Quantum algorithms: A survey of applications and end-to-end complexities. 2023b. URL http://arxiv.org/abs/2310.03011.

Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467, 1981.

Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: Hardness and applications to quantum chemistry and the quantum PCP conjecture. In *Proceedings Symposium on Theory of Computing*, STOC 2022, page 19, 2022.

András Gilyén, Yuan Su, Guang Hao Low, and Nathan 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*, STOC 2019, page 193, 2019.

Lov K. Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 53–62, 1998.

Jeongwan Haah. Product decomposition of periodic functions in quantum signal processing. 2018. URL http://arxiv.org/abs/1806.10236.

Connor T. Hann, Gideon Lee, S.M. Girvin, and Liang Jiang. Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2:020311, 2021.

Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103:150502, 2009.

David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n\log n)$. *Annals of Mathematics*, 193(2):563, 2021.

A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52:1191, 1997.

Rolf Landauer. Irreversibility and heat generation in the computational process. *IBM Journal of Research and Development*, 5(3):183, 1961.

Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. *PRX Quantum*, 2:030305, 2021.

Seunghoon Lee *et al.* Evaluating the evidence for exponential quantum advantage in ground-state quantum chemistry. *Nature Communications*, 14:1952, 2023.

Lin Lin and Yu Tong. Heisenberg-limited ground-state energy estimation for early fault-tolerant quantum computers. *PRX Quantum*, 3:010318, 2022.

Seth Lloyd. Universal quantum simulators. *Science*, 273:1073, 1996.

John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2:040203, 2021.

Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92:015003, 2020.

Thomas E. O'Brien, Brian Tarasinsk, and Barbara M. Terhal. Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments. *New Journal of Physics*, 21:023022, 2019.

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

Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends® in Theoretical Computer Science*, 9:125, 2014.

Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. Black-box quantum state preparation without arithmetic. *Physical Review Letters*, 122:020502, 2019.

A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281, 1971.

Changpeng Shao and Ashley Montanaro. Faster quantum-inspired algorithms for solving linear systems. *ACM Transactions on Quantum Computing*, 3:4, 2022.

Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303, 1999.

Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26:1474, 1997.

Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15:262, 2009.

Xiaoming Sun, Guojing Tian, Shuai Yang, Pei Yuan, and Shengyu Zhang. Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis. 2021. URL http://arxiv.org/abs/2108.06150.

Christoph Sunderhauf, Earl Campbell, and Joan Camps. Block-encoding structured matrices for data input in quantum computing. 2023. URL http://arxiv.org/abs/2302.10949.

K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete. Quantum Metropolis sampling. *Nature*, 471:87, 2011.

Kianna Wan, Mario Berta, and Earl T. Campbell. Randomized quantum algorithm for statistical phase estimation. *Physical Review Letters*, 129:030503, 2022.

Samson Wang, Sam McArdle, and Mario Berta. Qubit-efficient randomized quantum algorithms for linear algebra. *PRX Quantum*, 5:020324, 2024.