

PROF. DR. MARIO BERTA

QUANTUM ALGORITHMS

RWTH AACHEN UNIVERSITY

Copyright © 2026 Prof. Dr. Mario Berta

Lecture notes, work in progress.

Version: October 13, 2026

<https://marioberta.info/teaching/>

Feedback about this version is appreciated and should go to berta@physik.rwth-aachen.de

Acknowledgements:

Feedback from Prof. Dr. Maarten Wegewijs is gratefully acknowledged.

Assistants:

Hazar Çakir (2027), Nikolaos Louloudis (2026, 2027),

Gereon Kossmann (2024), Tobias Rippchen (2023)

Contents

Overview	5
0.1 Introduction	5
0.2 Literature	7
1 Classical circuit model	9
1.1 Deterministic algorithms	9
1.2 Probabilistic algorithms	11
1.3 Classical computational complexity	13
2 Quantum circuit model	17
2.1 Quantum systems	17
2.2 Quantum bits and quantum states	18
2.3 Quantum gates	19
2.4 Quantum measurements	22
2.5 Quantum computational complexity	23
2.6 Remarks on quantum error correction	24
3 Quantum query complexity	27
3.1 Setting	27
3.2 Deutsch's problem	28
3.3 Deutsch–Jozsa problem	28
3.4 Simon's problem	30
3.5 Other oracle-based quantum algorithms	32
4 Quantum Fourier transform	35
4.1 Discrete Fourier transform	35
4.2 Quantum circuit	35
4.3 Period finding	38
4.4 Order finding and Shor's algorithm	38

5	Quantum phase estimation	41
5.1	Task	41
5.2	Deterministic n -bit phase “estimation”	42
5.3	Phase estimation with approximate eigenvector	43
5.4	Phase estimation with accuracy vs. failure trade-off	43
5.5	Caveats and state of the art	43
6	Hamiltonian simulation	45
6.1	Task	45
6.2	Commuting case	47
6.3	Trotter based methods	48
6.4	Linear combination of unitary based methods	51
6.5	State-of-the-art methods and caveats	54
7	Ground state energy estimation	57
7.1	Task	57
7.2	Mapping to qubit form	59
7.3	Quantum phase estimation	60
7.4	Quantum state preparation and other bottlenecks	61
8	Quantum linear system solver (QLSS)	65
8.1	Task and classical landscape	65
8.2	Quantum task	67
8.3	Quantum data access	67
8.4	Basic quantum linear system solver	68
8.5	State-of-the-art methods and caveats	73
9	Quantum random access memory (QRAM)	75
9.1	Motivation	75
9.2	Quantum state preparation: Basic ideas	75
9.3	Quantum state preparation: Circuits	78
9.4	Quantum read only memory (QROM)	80
9.5	Fanout QRAM	81
9.6	Bucket-brigade QRAM	83
9.7	Extensions and caveats	85
10	Quantum singular value transform (QSVT)	87
10.1	Motivation	87
10.2	Block encoding data access	87
10.3	Transformation of block encodings	88
10.4	Example polynomials	90
	Bibliography	93

Overview

0.1 Introduction

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

Our considerations will be purely mathematical and therefore hardware agnostic (independent of the exact physics of the underlying devices). In particular, we will assume that we have noise-free qubits and quantum gates at hand (error-corrected and fault-tolerant), albeit potentially only a rather limited number thereof. This regime is often termed *early fault tolerance (EFT)*, where “early” emphasizes 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, preventing us from running very deep quantum circuits.

We then ask which, if any, problems from scientific computing can potentially be solved significantly more efficiently by using EFT quantum technologies. Our criterion is 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 that is 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 scale similarly to the quantum methods. In fact, proposed quantum methods can sometimes be *dequantized* into well-performing classical methods called *quantum-inspired* algorithms.

We will mostly focus on algorithms with provable worst-case performance guarantees, but heuristics for typical performance on relevant instances will also be discussed from time to time. Although classical algorithms that perform well in practice are often not exactly the same as those with the best provable worst-case performance guarantees, it is important to realize that such findings are typically based on extensive practical experiments and corre-

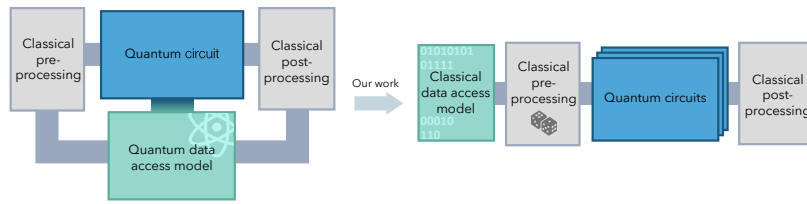


Figure 1: Motivation of recent work of our group, minimizing overall quantum resource costs by employing classical instead of quantum random memory data access structures.

sponding fine-tuning. In contrast, we take the viewpoint that, in the quantum setting, the current nascent state of quantum hardware largely allows only mathematical analyses to make statements about sufficiently large problem-instance sizes. Of course, these worst-case performance guarantees may overlook further quantum advantages that might become visible once better hardware allows for improvements adapted to specific problem instances of interest.

Our quantum algorithms will also extensively use classical sub-routines for pre- and post-processing, and we will demand a holistic viewpoint by asking to explicitly quantify all the classical and quantum resources employed, the so-called *end-to-end* computational complexity. 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

In principle, no previous physics or computer science knowledge is required (although it might of course help). We will work in an 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 the mathematics of 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 theoretical work and applied implementations. It is an ongoing research question ² how to improve on the textbook circuits covered in this course to build more noise-robust circuits, in particular by *co-designing* the high-level algorithmic structure and the lower-level fault-tolerant structure.

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

² Madelyn Cain, Qian Xu, Robbie King, Lewis R. B. Picard, Harry Levine, Manuel Endres, John Preskill, Hsin-Yuan Huang, and Dolev Bluvstein. Shor's algorithm is possible with as few as 10,000 reconfigurable atomic qubits. 2026. arXiv:2603.28627

0.2 Literature

A short literature list is as follows:

- 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 ³
- Complementary lecture notes:
Quantum Algorithms, Andrew M. Childs ⁴
- Mathematical lecture notes:
Quantum Computation, Ashley Montanaro ⁵
- Research level lecture notes:
Quantum Algorithms For Scientific Computation, Lin Lin 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 — can be consulted for specific topics:

- Quantum algorithms: A survey of applications and end-to-end complexities, available online.⁷

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. arXiv:2310.03011

To be clear, the following topics around quantum technologies will *not* be covered in the lecture:

- Quantum hardware and device physics
- Quantum error correction and fault tolerance⁸
- Quantum information theory⁹
- Quantum cryptography
- Post-quantum cryptography ¹⁰
- Quantum programming
- Analog quantum simulators
- Noisy, intermediate-scale quantum (NISQ) devices¹¹

⁸ *Quantum Error Correction* (M. Müller), RWTH Physics

⁹ *Quantum Information* (M. Berta), RWTH Physics

¹⁰ *Post-Quantum Cryptography* (D. Unruh), RWTH Computer Science

¹¹ John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018

1

Classical circuit model

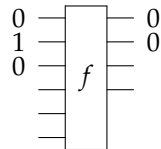
Before we introduce the general mathematical framework of quantum algorithms in Chapter 2, we first briefly discuss classical algorithms and complexity theory through a certain lens that is adapted for generalization to the quantum setting.¹

1.1 Deterministic algorithms

One way to think about deterministic classical algorithms — which will be important for moving to the quantum setting in Chapter 2 — is in terms of *classical circuits*. Namely, in the circuit model, for an n -bit-input-to- m -bit-output function

$$f : \{0,1\}^n \rightarrow \{0,1\}^m \quad \text{with } n, m \in \mathbb{N}, \quad (1.1)$$

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* specifies how inputs map to outputs.

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

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

where $\oplus$ 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

¹ This Chapter is inspired and partly adapted from the lecture notes *Quantum Information Theory*, Matthias Christandl, WS 2009/2010 LMU.

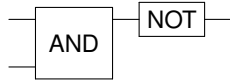
- NOT gate

$$x \text{ --- } \boxed{\text{NOT}} \text{ --- } \bar{x} = x \oplus 1$$

- AND gate

$$\begin{array}{c} x \text{ ---} \\ y \text{ ---} \end{array} \boxed{\text{AND}} \text{ --- } x \wedge y = x \cdot y$$

- NAND gate



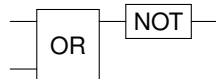
- OR gate

$$\begin{array}{c} x \text{ ---} \\ y \text{ ---} \end{array} \boxed{\text{OR}} \text{ --- } x \vee y$$

- XOR gate

$$\begin{array}{c} x \text{ ---} \\ y \text{ ---} \end{array} \boxed{\text{XOR}} \text{ --- } x \oplus y$$

- NOR gate



- FANOUT gate



The analogue of the NOT gate will be important for quantum circuits (Chapter 2), in addition to the following 2- and 3-bit gates:

- CNOT gate

$$\begin{array}{c} x \text{ ---} \\ y \text{ ---} \end{array} \boxed{\text{CNOT}} \text{ --- } \begin{array}{c} x \\ x \oplus y \end{array} = \begin{array}{c} x \text{ ---} \bullet \text{ ---} x \\ y \text{ ---} \oplus \text{ ---} x \oplus y \end{array}$$

The right drawing convention more precisely indicates the directionality: NOT is applied to *target* bit 2 *controlled* by bit 1.

- SWAP gate

$$\begin{array}{c} x \text{ ---} \\ y \text{ ---} \end{array} \boxed{\text{SWAP}} \text{ --- } \begin{array}{c} y \\ x \end{array} = \begin{array}{c} x \text{ ---} \times \text{ ---} y \\ y \text{ ---} \times \text{ ---} x \end{array}$$

- TOF gate

$$\begin{array}{c} x \text{ ---} \\ y \text{ ---} \\ z \text{ ---} \end{array} \boxed{\text{TOF}} \text{ --- } \begin{array}{c} x \\ y \\ z \oplus x \cdot y \end{array} = \begin{array}{c} x \text{ ---} \bullet \text{ ---} x \\ y \text{ ---} \bullet \text{ ---} y \\ z \text{ ---} \oplus \text{ ---} z \oplus x \cdot y \end{array}$$

The 3-bit Toffoli gate can in fact be seen as a controlled-controlled NOT gate. The convention on the right is again more precise.

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

Importantly, every function can indeed be written in terms of a few types of elementary gates and one might for example use the following *universal gate set*:

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. □

It is often stated that NOT, AND, OR are universal for classical circuits but the proof then implicitly assumes that inputs can be copied by FANOUT gates, which therefore should be added to the universal set.² Also, one often uses that the AND and OR gates can be applied to *any pair* of (non-)adjacent input wires. Here one implicitly assumes that inputs can be permuted into any desired order for the application of these gates to *adjacent* inputs. This can be achieved using SWAP gates on adjacent inputs but to prove universality one must show how these SWAPs can also be generated from the above 4 gates.³

Universality means that any function on n bits can be decomposed into a finite number of elementary gates, potentially scaling exponentially as 2^n . This can be shown to be necessary for “most” functions (in some precise sense) posing a challenge for algorithmic design where we are interested in functions that can be evaluated *efficiently*, i.e., by circuits of polynomial size in n . We will revisit this point soon in Section 1.3 on complexity classes.

Other choices of gate set are possible. As an interesting one, it is possible to instead use a *single, reversible gate* as elementary gate:⁴

Proposition 4 (Universality II). *By using the elementary TOF gate, every function can be computed provided ancilla bits can be fixed at the input and left unused at the output.*

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 erasing information during a computation always increases the entropy of the system and therefore leads to heat dissipation.⁵ The result shows that in principle this can always be avoided. Additionally, we will see in Chapter 2 that reversible circuits are conceptually close to quantum circuits.

1.2 Probabilistic algorithms

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 exponentially large in n as for quantum circuits.

² FANOUT can clearly not be generated by these 3 gates: the number of bits at their output is never larger than at their input.

³ Thus, although SWAP need not be added to the universal gate set, the number of gates required to synthesize it should be included in complexity counts. Whereas in classical circuits this is often not done, in quantum circuits one certainly *does* count them (3 CNOT gates for one SWAP). This can be confusing.

⁴ Note that this result entails that with TOF alone one can generate not only NOT, AND, OR but also FANOUT (to copy wires) and from these SWAP (to permute non-adjacent inputs to adjacent input wires of AND, OR.)

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

In this setting, we assume random number generators are available, for example, a *random* bit. To describe this, we identify the deterministic bit values with 2-dimensional real vectors $0 \leftrightarrow (1,0)^T \equiv |0\rangle$, $1 \leftrightarrow (0,1)^T \equiv |1\rangle$, i.e., we label these basis vectors by the bit values.⁶ The state of a random bit is just a vector filled with the probabilities of the bit values

$$\vec{p} = (p_0, p_1)^T \in \mathbb{R}^2 \text{ with } p_0, p_1 \geq 0, p_0 + p_1 = 1. \quad (1.3)$$

Mathematically, this is a linear combination of deterministic state vectors with probability coefficients. Using bra-ket notation

$$|p\rangle = p_0|0\rangle + p_1|1\rangle \quad (1.4)$$

where we distinguish *classical* round kets $| \rangle$ from quantum kets $| \rangle$.

In probabilistic circuits reversible gates are described by matrices representing the permutation of bit strings that label the deterministic basis vectors. For one bit, the NOT gate flips the bit value and thus permutes the two basis vectors

$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (1.5)$$

The corresponding output distribution is computed as

$$\vec{p}_{\text{out}} = \text{NOT} \cdot \vec{p} = (p_1, p_0)^T \in (\mathbb{R}^2). \quad (1.6)$$

For 2 bits we identify the 4 possible input bit strings

$$\{0,1\}^2 = \{00, 01, 10, 11\} \quad (1.7)$$

with the 1st basis vector $00 \leftrightarrow (1,0,0,0)^T \equiv |00\rangle$, the 2nd one $01 \leftrightarrow (0,1,0,0)^T \equiv |01\rangle$, etc. Now any input distribution over these 4 bit strings can be considered as a 4-component vector in the tensor product of the two vector spaces of the two bits:

$$\vec{p} = (p_{00}, p_{01}, p_{10}, p_{11})^T \in (\mathbb{R}^2)^{\otimes 2} \text{ with } p_{i,j} \geq 0, \sum_{ij} p_{ij} = 1, \quad (1.8)$$

It is constrained by positivity and normalization of the probabilities. The action of the controlled-not gate, for example, can be written in matrix form with respect to this computational basis as

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (1.9)$$

Its nontrivial action is to swap the last two basis vectors corresponding to bit strings 10 and 11 as required for a CNOT gate. Observe that reversible gates like NOT and CNOT are permutations of (the bit strings labeling) the basis vectors. Therefore their matrices obviously preserve the positivity of the probability vector and its normalization by the 1-norm $\|\vec{p}\|_1 = \sum_i |p_i|$. These matrices have the additional property of being orthogonal, i.e., $U^{-1} = U^T$.

⁶ The transposed vector is given as $(1,0)^T = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Finally, a new nontrivial aspect of probabilistic circuits is that one must consider how to *sample* the output to extract algorithmic information, i.e., how to get the answers one is interested in with high probability. We will see examples of this later on, both in the classical and the quantum setting.

The above vector-matrix formulation of probabilistic classical theory provides a useful second point of entry to the mathematical definition of quantum theory in Chapter 2, complementing the analogy with classical deterministic circuits (or Boolean logic). Both should be kept in mind since in quantum theory deterministic and probabilistic classical aspects mesh together in a non-trivial way.

1.3 Classical computational complexity

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 for some worst-case functions.⁷ Clearly, in the present course where we focus algorithms, we are mostly interested in complexity upper bounds, but it is useful to comment on complexity lower bounds from time to time.

Instead of looking at functions of fixed input and output sizes and counting the concrete number of elementary gates needed, one typically studies families of differently sized instances, labeled by a parameter $n \in \mathbb{N}$ (often the bit precision), and is then interested in the scaling of the required number of gates for $n \rightarrow \infty$. To quantify corresponding complexity upper and lower bounds, there is the *big O notation* from computer science:⁸

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 $\mathcal{O}(n)$.

However, often the exact computational complexity of functions is not known, even for elementary algorithms:

Example 6. For the multiplication of two n -bit numbers, the textbook algorithm gives $\mathcal{O}(n^2)$. Harvey and van der Hoeven gave an algorithm with complexity $\mathcal{O}(n \log n)$.⁹ More recently, OpenAI gave an exact deterministic algorithm with worst-case time $\mathcal{O}(n(\log n)^{1-\kappa})$, where $\kappa = 2^{-182}$, on a fixed finite-alphabet multitape Turing machine.¹⁰ This disproves the

⁷ Boaz Barak and Sanjeev Arora. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2007

⁸ The big $\mathcal{O}$ and Ω stand for upper and lower bound "order" and the big Θ for "tight" is used when these bounds coincide.

⁹ David Harvey and Joris van der Hoeven. Integer multiplication in time $\mathcal{O}(n \log n)$. *Annals of Mathematics*, 193(2):563, 2021

¹⁰ OpenAI. Integer multiplication below $n \log n$. OpenAI Math Release preprint, 2026a. OpenAI Math Release OAI:Integer-multiplication-below-n-log-n-September-23-2026, 23 September 2026

conjectured $n \log n$ optimality in that model. The gap to the trivial lower bound $\Omega(n)$ remains open.

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 $(A \cdot B)_{i,j} = \sum_{k=1}^n (A)_{i,k} (B)_{k,j}$ gives $\mathcal{O}(n^3)$. For matrices over $\mathbb{C}$, the current arithmetic exponent satisfies $\omega \leq 9/4 = 2.25$.¹¹ Equivalently, for every $\delta > 0$, matrix multiplication can be performed in

$$\mathcal{O}(n^{9/4+\delta}) \quad (1.10)$$

arithmetic operations, whereas a trivial lower bound is $\Omega(n^2)$, set by loading the matrix. In Exercise Sheet 1, you will work through the Strassen algorithm, which has complexity $\mathcal{O}(n^{\log_2 7})$.

The big O-notation hides potentially large constants (c) and as such algorithms with optimal asymptotic scaling do not necessarily need to be practical. Nevertheless, we coarse grain the concept of algorithmic performance even further: we typically think of circuits as being *efficient* if their size is polynomial in input length of the problem. This is useful since it allows us to categorize problems that algorithms solve into asymptotic *complexity classes*. Some prominent examples which focus on time complexity include, very informally speaking:

- 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. Here time is counted as the number of sequential layers of gates executed in parallel by the circuit, called the *circuit depth*, the longest path through the circuit from any input to an output.¹²
- 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, for which it is *not known* whether it is in P or BPP is as follows.¹³

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

$$\mathcal{O} \left(\exp \left(1.9 \cdot n^{1/3} (\log n)^{2/3} \right) \right). \quad (1.11)$$

¹¹ OpenAI. An upper bound of $9/4$ for the matrix multiplication exponent. OpenAI Math Release preprint, 2026b. OpenAI Math Release OAI:Matrix-Multiplication-Nine-Fourths-October-2-2026, 2 October 2026

¹² For non-parallelizable algorithms this is just the *horizontal* circuit size. For parallelizable algorithms this can be a *diagonal* path, and there is typically a trade-off between space cost (high parallelization) and time complexity (low parallelization).

¹³ It is a fundamental open question in theoretical computer science whether $P = NP$ or $P \neq NP$.

Integer factorization is in NP as multiplication is in P, but is integer factorization in BPP or even P? It is still possible –but considered to be unlikely– that we have simply not found the efficient algorithm yet! Against this background, the landmark result of quantum computing by Shor¹⁴ shows that

There is a *quantum* phase-estimation algorithm for integer factorization with quantum gate complexity

$$\mathcal{O}\left(n^2 \log n\right), \quad (1.12)$$

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

¹⁴ Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303, 1999

In the next chapter, we introduce the *quantum circuit model* to formalize what quantum algorithms are and use quantum computational complexity to quantify their performance and meaningfully compare them with classical algorithms.

2

Quantum circuit model

2.1 Quantum systems

We know that the laws of physics qualitatively change when going to different length and energy scales. 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. These two theories are fundamentally different in mathematical structure and consequently in 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 when they are controlled (instead of suppressed)?¹ Fundamentally, this corresponds to the exciting scientific question *what nature can compute efficiently*. Intuitively, we might expect that such quantum computing devices are more efficient at computationally resolving physical and chemical systems since they are *also* described by quantum mechanics. While we will not concern ourselves with how, if, and when quantum computing devices can be built, our interest lies in the theoretical-computer-science and algorithmic aspects of this 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 physical model of computation, where the word efficiently here means up to polynomial-time overhead. So following this (and without formally defining Turing machines), this conjecture about physics implies that quantum circuits should not allow for large computational speed-up compared to classical circuits.

¹ The same question was asked about probabilistic classical circuits in the 1960s and 1970s and led to new, better algorithms. In contrast to the quantum case, it is believed that randomized algorithms are *not* fundamentally more powerful but instead are just convenient, clever ways to somehow exploit “underlying” deterministic algorithms which are hard to find. When occasionally one is found, this is called *de-randomization* of the probabilistic algorithm to a deterministic one. In terms of complexity classes, it is a widely believed conjecture that $P = BPP$.

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

In this lecture, we will adopt the mathematical framework describing devices behaving as so-called *closed quantum systems*. This means that we assume perfect shielding of the devices from their 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 allowed in this model.

An immediate disclaimer is in order here. As with other non-standard notions of analog computation, one has to make sure that there are no hidden (and potentially exponentially large) *physical costs* to actually implement the assumed model.² For quantum circuits, these concerns have been addressed by a careful analysis of quantum error-correcting codes, which shows that, at least in principle, faulty device components can be operated with arbitrarily low logical error rates at the cost of only a low overhead. We briefly touch upon this subject in Section 2.6.

² 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 states

To emphasize again: We are neither concerned with the actual physical laws of quantum theory nor with concrete technology by which things are implemented. Rather, we now introduce a purely abstract mathematical model for quantum computation, the *quantum circuit model*, just as one does for classical computation.

Our starting point is the classical circuit model, in particular its *probabilistic* version in Chapter 1. As an example, recall how the CNOT gate acts on two probabilistic bits as a matrix (Eq. (1.9)). Now, the first axioms of the circuit model of quantum computing 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 as $\|\vec{\psi}\|_2 = \sqrt{|\alpha_0|^2 + |\alpha_1|^2} = 1$ (with $|\alpha|^2 = \alpha\alpha^*$). 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* are described by complex unitary matrices U , that is, $(U^T)^* = U^{-1}$, taking complex input vectors to complex output 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*, just as in Chapter 1. For one qubit we have

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad (2.1)$$

and a general 1-qubit quantum state is then written as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \vec{\psi}, \quad (2.2)$$

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

$$|\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \quad (2.3)$$

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

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad (2.4)$$

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad (2.5)$$

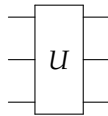
$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad (2.6)$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \quad (2.7)$$

Two-qubit states that cannot be written as a product of single-qubit states are correlated between the qubits and are called *entangled* in quantum theory.

2.3 Quantum gates

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 of Chapter 1 have quantum *analogues* with the same matrix. In addition to these, there are many more non-classical gates.

Prominent examples of 1-qubit gates are the following:

- $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ gate, the quantum analogue of NOT



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



- $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ gate, or π -phase gate



- $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, called “Hadamard gate”



- $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ gate, or $(\pi/2)$ -phase gate, called “S-gate”



- $T = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix}$ gate, $\pi/4$ -phase gate, called “T-gate”



Important quantum analogues of classical gates on 2 qubits are

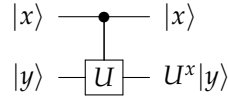
$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (2.8)$$

and on 3 qubits

$$\text{TOF} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.9)$$

Exercise 9. Check that the matrix representations given above for the gates NOT = X, CNOT, SWAP, TOF indeed act as claimed in the circuit diagrams of Chapter 1.

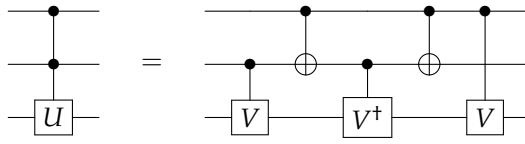
One can also implement general controlled U -gates



where U is a 1-qubit unitary.

Exercise 10. Give the matrix representation of this gate.

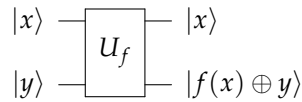
In order to build up to an understanding of more advanced quantum algorithms, it is crucial that you get a good hands-on experience how to work with quantum states and how quantum gates transform them. Here is an example of a circuit identity



Exercise 11. What quantum gate / unitary matrix is V in the above circuit identity?

You will explore more circuit identities in Exercise Sheet 2.

As we allow³ only reversible gates, i.e., unitary matrices, in the quantum circuit model, how should we implement a general, non-reversible function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$? One way is to use the $(n + m)$ -qubit quantum gate U_f defined as



where $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$, and $\oplus$ denotes bitwise addition modulo 2: $f(x) \oplus y = (f_1(x) \oplus y_1) \dots (f_m(x) \oplus y_m)$.

Exercise 12. 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 of unitaries in the quantum circuit model. In the quantum case, there are two issues, considering one- and two-qubit gates.

First, we mention that it can be shown that any unitary $2^n \times 2^n$ matrix acting on n qubits can be *exactly* decomposed into one-qubit gates and *one type* of two-qubit gate, in particular the CNOT gate (other choices are possible), with complexity $\mathcal{O}(n^2 \cdot 4^n)$. That an exponential number of gates is indeed required to build a generic circuit — as in the classical circuit model — poses a challenge for quantum algorithm design and is an ongoing research topic.

A second problem could be that one-qubit gates, unlike classical gates, form a *continuous* set of unitaries (group $SU(2)$). However, we can slightly relax the condition to *approximate* universality. Namely,

³ The restriction to unitary gates is a convenient *choice* we make in the circuit model. It is possible to implement non-unitary, non-reversible gates *without* leaving the broader framework of closed quantum systems, i.e. without introducing “noise”. For example, $\text{FANOUT}|i\rangle := |ii\rangle = \text{CNOT}|i0\rangle$ defines a valid quantum analogue of the fanout gate used in quantum error correction. It is isometric, $\text{FANOUT}^\dagger \text{FANOUT} = \mathbb{I}$, but *not* unitary $\text{FANOUT} \text{FANOUT}^\dagger \neq \mathbb{I}$. One can show that such maps can be *simulated* by a unitary with partially fixed input ancilla qubits, in close analogy to classical reversible computing.

we seek to decompose one-qubit unitaries U into a product of some finite set of elementary one-qubit gates $\{U_1, U_2, \dots, U_t\}$ as

$$\left\| U - \prod_{i=1}^t U_i \right\|_{\infty} \leq \varepsilon \quad (2.10)$$

with some small error $\varepsilon > 0$ using the infinity norm $\|O\|_{\infty} = \sup_{|\psi\rangle} \|O|\psi\rangle\|_2$ where the supremum is over normalized complex quantum-state vectors $|\psi\rangle$. This is the worst-case notion of approximation. Remarkably, the continuous nature of quantum gates turns out *not* to be a problem: Denoting $\text{polylog}(n) = \mathcal{O}((\log n)^k)$ for some $k \in \mathbb{N}$ we have ⁴

Proposition 13 (Solovay-Kitaev). *For the gate set $\{H, T\}$, it is sufficient that $t = \text{polylog}(1/\varepsilon)$. That is, arbitrary 1-qubit quantum gates can be approximated up to exponentially small error with only polynomial overhead.*

⁴ A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52:1191, 1997

Proving this statement is not part of this course, but we note that finding such efficient decompositions with as small a polylog overhead as possible is another ongoing research topic. Taken together this implies that $\{\text{CNOT}, H, T\}$ is an (approximate) universal gate set⁵. As for the classical circuit model, it remains a challenge to find *efficiently* implementable quantum algorithms that allow decomposition into only *polynomial* in n many universal quantum gates. We will explore this in particular in Chapter 6.

⁵ As in the classical case, there are many alternative universal gate sets. In this course we focus, however, on this “standard” gate set.

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!

2.4 Quantum measurements

In order to read out (some of the) classical information contained in quantum states, quantum measurements are applied. Unlike classical deterministic computations, this is a crucial *algorithmic* step which can make or break a quantum computation. For our purposes, so-called computational-basis measurements are sufficient:

- A qubit state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ measured in the computational basis $\{|0\rangle, |1\rangle\}$, leads to a binary post-measurement 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 such a measurement is applied, only one measurement outcome will actually be observed (0 or 1). To resolve the post-measurement probability distribution one needs to run the quantum circuit a few times and record the different outcomes.

- 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 $\{|0\rangle, |1\rangle\}^{\otimes n}$, leading to the post-measurement probability distribution $\{p_k = |\alpha_k|^2\}_k$ over all possible outcomes labeled by the binary strings $k \in \{0,1\}^n$.

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

$$|0\rangle \xrightarrow{\text{H}} \text{---} \boxed{\text{H}} \text{---} \boxed{\text{---}} \quad (2.11)$$

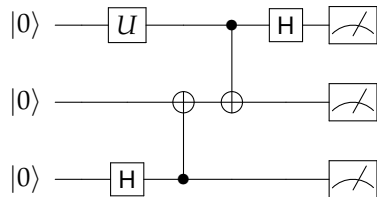
Here, the Hadamard H gate takes $|0\rangle$ to the quantum state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and the probabilities of observing 0 and 1 are each 1/2. Note that this probabilistic behavior under measurement is generated by a *reversible* gate H: Removing it from the circuit makes the measurement statistics deterministic, giving 0 with certainty, while 1 does not occur. Likewise, inserting an *additional* reversible H gate into the circuit removes the probabilistic behavior ($H^2 = \mathbb{I}$), leading back to the same deterministic statistics. This non-classical behavior hinges on the possibility of having negative coefficients in the H matrix.

Exercise 14. Explain why $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ requires H to have negative coefficients.

This is not possible with a probabilistic classical bit [Eq. (1.6)] even though it allows for linear combination, but only with positive coefficients: it stays probabilistic under any reversible classical gate.

2.5 Quantum computational complexity

Having put together the mathematical framework for quantum computing circuits we can now discuss algorithms. Without loss of generality an end-to-end quantum algorithm on n qubits starts in the all-zero state $|0 \cdots 0\rangle = |0\rangle \otimes \cdots \otimes |0\rangle$, 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:



We now want to quantify performance and compare it with classical computing. As in the latter case, we base complexity on the efficient implementability of an algorithm using at most a polynomial number in n many gates. Informally, this corresponds to the quantum analogue of the classical complexity class BPP:

- All problems that can be computed with error probability at most 1/3 using *quantum* circuits that are polynomial in qubit input length n make up the complexity class *bounded-error quantum polynomial time* BQP.

The question about large quantum speed-ups relative to classical algorithms is then how BQP is related to BPP. We have the inclusion $\text{BPP} \subseteq \text{BQP}$ since quantum circuits can generate probability distributions and simulate all classical gates using the classical universality of the *reversible* Toffoli gate. Crucially, integer factorization is a problem that is known to be in BQP by the existence of Shor's algorithm, but it is not known whether it is in BPP: There is NO proven separation between the two complexity classes, i.e., not a single example is known.⁶ This and other open problems illustrate the importance of studying complexity theory.

What is now intuitively the power of quantum circuits compared to classical circuits? The complex-valued nature is the only mathematical reason for a possible quantum speed up compared to classical circuits. Do not believe anything else said or claimed in the popular literature! Pin-pointing further quantum-classical distinctions responsible for an advantage is still an active field of research in which there seem to be no easy answers.⁷ Still, the complex values of amplitudes are at the basis of two key ideas in quantum algorithmic design which we already encountered in the circuit (2.11):

- (i) In quantum circuits we can by choice of gates internally involve⁸ all possible computational states at once in the circuit.
- (ii) Conversely, by gates we can also select only some of these computational states to show up in the final measurement outcomes (the desired end-to-end output of an algorithm) while suppressing others (undesired output). This is called constructive / destructive interference. Indeed, we will see the Hadamard gate of circuit (2.11) —and the quantum Fourier gate that generalizes it— time and again, starting from Chapter 3. There we discuss some simple toy example algorithms based on constructive interference patterns in quantum circuits that will give an exponential classical-quantum separation in query complexity, i.e., only under the strong assumption of black-box oracles.

These examples also reveal a bottleneck of getting results of an algorithm *out* of a quantum computer. Quantum algorithm design entails finding problems for which this can be done efficiently. As expected, there is a converse bottleneck of getting initial data *into* a quantum computer which is addressed in detail in Chapter 9: Loading the classical input data in general requires either an exponential number of gates or one has to face a nontrivial space-time trade-off. An additional challenge is thus to find problems for which data can be loaded efficiently or computed on the fly. Even with all of this in place there seems to be no simple useful set of general ideas that guarantees quantum speed up.

2.6 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

⁶ That is, in principle it could still be that $\text{BPP} = \text{BQP}$ and that we have just *not found yet* a polynomial classical algorithm for integer factorization. As such there could be no quantum advantage in Shor's algorithm, but it is considered to be unlikely.

⁷ The following are *not* general reasons to expect quantum speed up:

- Exponentially large state space This is also the case for classical probabilistic circuits. The dimension $4^n = (2^n)^2$ for n qubits is only quadratically larger than 2^n for n classical bits. Note that 4^n is the general quantum dimension for *open* systems.
- Superposition as linear combination of states. This is mathematically also possible in classical probabilistic circuits *except not with complex coefficients*.
- Entanglement as linear combination of product states. This is just superposition applied to compositions of n qubits so the previous point applies. Classically this corresponds to correlations.

If one brings up any of these reasons one has to explain why classically the corresponding feature does *not* allow to exponentially speed up algorithms. The one difference that is clear is the *complex values* of coefficients.

⁸ Quantum systems are *not* "in multiple states at the same time". In example (2.11) $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is a single state, not two.

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 current nascent state of quantum technology with only limited control and shielding against the environment. As such, one needs to argue why the mathematical model of *closed quantum systems* with noise-free gates and measurements is of any relevance.

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

$$3p^2(1-p) + p^3 = p^2(3-2p) < p \quad \text{if } p < 1/2. \quad (2.12)$$

This is the *3-bit repetition code* and exemplifies the main principle behind error correction. Similar ideas work in the quantum realm! Moreover, computations need to be performed on the encoded qubits, and the encoding and decoding themselves might be faulty. All these problems are addressed by the threshold theorem for quantum error correction⁹, which we state here informally:

Theorem 15 (Quantum Fault-Tolerance). *Under reasonable physical assumptions on the noise model, any n -qubit circuit with $g(n)$ gates and measurements can be simulated with success probability $1 - \varepsilon$ using*

$$\mathcal{O}(g(n) \cdot \text{polylog}(g(n)/\varepsilon)) \quad (2.13)$$

many noisy gates and noisy measurements, provided every component fails with probability $p < p_{tr}$ below a threshold $p_{tr} \approx 10^{-3} - 10^{-4}$.

Hence, even with faulty components, arbitrarily long quantum 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) and is part of other courses¹⁰. The development of state-of-the-art quantum error correcting codes is a very active research field both in academia and industry.

Finally, we note that if one could build fault-tolerant quantum computers, one could also build random-access memories with the same hardware. However, we will see in Chapter 9 that for typical applications the space requirements of this QRAM are exponentially larger than for the algorithmic units, but the error-correction overheads are smaller (e.g. since memory does not require universal circuits). There are thus opportunities in designing specialized quantum hardware for QRAM circuits.

To conclude, based on our current understanding of physics, we do not know of any reasons why quantum computing cannot be implemented and scaled. 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

⁹ Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error rate. *SIAM Journal on Computing*, 38:1207, 2008

¹⁰ *Quantum Information* (M. Berta), RWTH Physics; and *Quantum Error Correction* (M. Müller), RWTH Physics

be the most exciting outcome of the efforts of realizing quantum technologies. Check out Ref. ¹¹ for a more in-depth critical discussion around the feasibility of quantum computing.

¹¹ <https://www.scottaaronson.com/democritus/lec14.html>

3

Quantum query complexity

3.1 Setting

In this chapter you are given 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*, or, as in the next three chapters, *periodicities*. 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 a minimal number of queries.

More precisely, our way to model an oracle is as a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ 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 for the function f . with the quantum circuit

$$\begin{array}{ccc} |x\rangle & \text{---} & |x\rangle \\ |y\rangle & \text{---} & |f(x) \oplus y\rangle \end{array} \quad \begin{array}{c} \boxed{U_f} \end{array}$$

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

We say there is a separation between quantum and classical complexities for a problem if a quantum upper bound (obtained by constructing a concrete quantum algorithm) is below a classical lower bound (proved for *any classical* algorithm that solves the problem). One of the advantages of the query-complexity model is that it is much easier to prove such difficult 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 take into account how the function f is actually implemented¹ (as only the queries to it are counted). Consequently, it is important to realize that query complexity separations do in general not give

¹ Remember from Chapter 1-2 that a generic function f has an exponentially in n large circuit both in the classical and quantum setting.

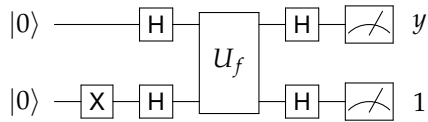
computational complexity separations. Nevertheless, as a toy model query complexities give a simple, first way of illustrating the power of the quantum circuit model and in some cases also serve as a high-level design step for more serious end-to-end algorithms².

² Famously, Shor's *end-to-end* algorithm was inspired by Simon's *query* algorithm which we discuss below.

3.2 Deutsch's problem

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

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

$$|0\rangle \otimes |0\rangle \mapsto |0\rangle \otimes |1\rangle \quad (3.1)$$

$$\mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (3.2)$$

$$\mapsto \frac{1}{\sqrt{2}} \left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle \right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (3.3)$$

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

Thus, measuring the second qubit *always* gives outcome 1 but the first qubit will lead to the outcome $y = 0$ with a probability

$$p_{01} = \left(\frac{1}{2} \left((-1)^{f(0)} + (-1)^{f(1)} \right) \right)^2 = \overline{f(0) \oplus f(1)}, \quad (3.5)$$

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

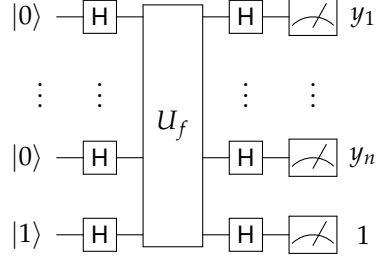
3.3 Deutsch–Jozsa problem

Given a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 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

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

decide which one is the case. Classically one needs to query the oracle $2^{n-1} + 1$ times to determine this deterministically, i.e., we have a tight lower bound $\Theta(2^{n-1})$.

The Deutsch-Josza quantum algorithm solves this problem with only one quantum query to the function f in the form of U_f as



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

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

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

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

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

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

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

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

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

After measuring all qubits, we obtain outcome 1 with certainty on the last qubit. On the other qubits, the probability of getting the all-zero bit string $0 \cdots 0$ is

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

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

This is the promised exponential query separation:
1 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 classical probabilistic complexity lower

bound improves to $\Theta(1)$ and therefore the large quantum query complexity advantage collapses. In the next section, we discuss an important query complexity problem that *does* exhibit 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 $f : \{0,1\}^n \rightarrow \{0,1\}^n$ with the promise that for some unknown *non-zero* $c \in \{0,1\}^n$ we have for all $x, y \in \{0,1\}^n$ that

$$f(x) = f(y) \iff x = y \text{ or } x = y \oplus c, \quad c \neq 0 \dots 0, \quad (3.15)$$

determine the nontrivial *period* c . Thus we know that f is a 2-to-1 map, i.e., for pairs of inputs, the function values “collide”.

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

$$x = y \oplus c \implies x \oplus y = y \oplus c \oplus y = c. \quad (3.16)$$

By the promise that $c \neq 0 \dots 0$ we must find a match. As such, in the worst case we need $2^{n-1} + 1$ queries to f so the query complexity is at least $\Omega(2^n)$.

Crucially, it can also be proven that even probabilistic classical algorithms succeeding with high probability still need at least an exponential number $\Omega(\sqrt{2^n})$ of 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 have to make in order to observe a collision of function values — i.e., getting the same entry twice — with *high probability*? We give a proof sketch that for $Q = \sqrt{2^{n+1}}$ the expected number of collisions is roughly one.³ Namely, for queries $j_1, j_2, \dots, j_Q$, how large is the probability that we do not witness a collision? There are $\binom{Q}{2}$ pairs that could collide and the collision probability is $\frac{1}{2^n - 1}$. Hence, the *expected number* of collisions is

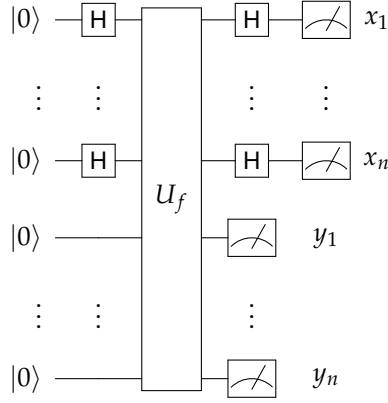
$$\frac{\binom{Q}{2}}{2^n - 1} \approx \frac{Q^2}{2^{n+1}}, \quad (3.17)$$

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

In the quantum setting, it turns out that $\mathcal{O}(n)$ quantum queries together with some clever classical post-processing suffice⁴. The quantum circuit uses $2n$ qubits:

³ An extended argument can also be made for the claimed worst case high probability of success.

⁴ Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26:1474, 1997



The evolution of the input quantum state is as follows:

$$|0\rangle^{\otimes n} \otimes |0\rangle^{\otimes n} \mapsto \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} |z\rangle \otimes |0\rangle^{\otimes n} \quad (3.18)$$

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

$$\mapsto \sum_{x \in \{0,1\}^n} \sum_{z \in \{0,1\}^n} \frac{1}{2^n} (-1)^{z \cdot x} |x\rangle \otimes |f(z)\rangle. \quad (3.20)$$

Upon measuring all qubits the probability is

$$p_{xy} = \frac{1}{2^{2n}} \left[\sum_{z \in \{0,1\}^n} (-1)^{z \cdot x} \delta_{y,f(z)} \right]^2 \quad (3.21)$$

By the promise on the problem, f is a 2-to-1 map such that $y = f(z)$ has precisely two solutions, denoted $z = z_y$ and $z = z_y \oplus c$, giving two interfering terms under the square:

$$\begin{aligned} p_{xy} &= \frac{1}{2^{2n}} \left[(-1)^{z_y \cdot x} + (-1)^{(z_y \oplus c) \cdot x} \right]^2 \\ &= \frac{1}{2^{2(n-1)}} \left[\frac{1}{2} (1 + (-1)^{c \cdot x}) \right]^2 = \frac{1}{2^{2(n-1)}} \delta_{c \cdot x, 0} \end{aligned} \quad (3.22)$$

Note that z_y drops out, and with it any connection between x and y via the function f . What matters is that each string of measured values x provides partial information about the period c of f . The y values are not useful.

The measurements will give uniformly random outcomes from the set $\{x : c \cdot x = 0\}$ of size 2^{n-1} with the sought-after period c , and, independently, samples y from the image set of f , also of size 2^{n-1} . Hence, every run of the circuit generates one (binary) linear equation for the unknown n -bit string c . Collecting $n - 1$ independent linear equations would then be sufficient to solve for c since by the promise $c \neq 0$ we know that c has at least one bit equal to 1. A key point is that only $\mathcal{O}(n)$ –and not exponentially many– runs will lead with high probability to such a set of $n - 1$ independent equations. Then standard (binary) Gaussian elimination of classical complexity $\mathcal{O}(n^3)$ allows to solve for c .

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

Note that this is really the setting we want to find separations for: randomness is cheap and a success probability close to one is sufficient in practice (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 actually was the predecessor or motivation for Shor's quantum integer factorization algorithm — which then gives an end-to-end 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

The literature contains a plethora of other query-complexity-based algorithms with quantum advantages. Most prominently, there is a class of query algorithms that go under the names *Grover search*, *quantum search*, or *amplitude amplification / estimation*. These all go back to Grover's seminal work,⁵ which analyzed the following problem. Given a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, find a $c \in \{0,1\}^n$ such that $f(c) = 1$; or output “no solution” if such a c does not exist.

This can be seen as a version of searching an unstructured database of size $N = 2^n$ and classically a randomized algorithm will use $\Theta(N)$ classical queries to f . Grover's quantum algorithm uses $\Theta(\sqrt{N})$ quantum queries to U_f , a quadratic quantum speedup. 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.

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:

- They give *at most a quadratic* classical-quantum speedup, 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 speedups are not the reason people are excited to build quantum computers.
- They operate 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 which might exploit some structure of the oracle func-

⁵ 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

tion for the problem at hand to such an extent that it compensates the merely quadratic quantum *query*-complexity advantage.

Given that we are mostly interested in large quantum computational complexity advantages for scientific computing, we thus refrain from further treating problems solved by Grover's algorithm⁶ in this course. However, the more general amplitude amplification version will be used as subroutine in Chapters 6, 7 and 8.

⁶ *Quantum Information* (M. Berta),
RWTH Physics

Note how this differs from Simon's algorithm and its generalizations discussed in Chapter 4. There one starts from an *exponential* query-complexity separation (without large hidden prefactors) which is much harder for a classical algorithm to compensate by exploiting some structure of the problem hidden in the f -oracle. One is thus interested in finding problems which have an efficient oracle implementation such that the exponential query advantage is maintained, the first and most famous of which is Shor's solution to the integer factorization problem based on Simon's algorithm.

4

Quantum Fourier transform

4.1 Discrete Fourier transform

In classical signal processing, the *Fourier transform* is ubiquitous. Its discrete version, the discrete Fourier transform $F_N : \mathbb{C}^N \rightarrow \mathbb{C}^N$ (DFT), takes a complex vector $(x_0, \dots, x_{N-1})$ to a complex vector $(y_0, \dots, y_{N-1})$ defined elementwise as

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

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

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

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 naively 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 The current best asymptotic result in the fixed finite-alphabet multitape Turing-machine model is the sub- $n \log n$ bound $\mathcal{O}(n(\log n)^{1-\kappa})$, with $\kappa = 2^{-182}$.³ As the DFT is already unitary, is there an n -qubit quantum algorithm to implement the DFT that uses even fewer than $\mathcal{O}(n2^n)$ quantum gates?

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)⁴ is a quantum gate that takes n -qubit quantum states $|x\rangle$ to n -qubit quantum states $F_N|x\rangle$. The $2^n \times 2^n$ matrix F_N is exponentially large, but what is the minimal number of elementary gates to implement it in the quantum circuit model?

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

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

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

³ OpenAI. Integer multiplication below $n \log n$. OpenAI Math Release preprint, 2026a. OpenAI Math Release OAI:Integer-multiplication-below-n-log-n-September-23-2026, 23 September 2026

⁴ D. Coppersmith. An approximate fourier transform useful in quantum factoring, 1994. IBM Research Report RC 19642, 1994

with the *binary fraction* notation

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

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

$$F_N |k\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \exp\left(\frac{2\pi i j k}{2^n}\right) |j\rangle \quad (4.5)$$

$$= \frac{1}{\sqrt{2^n}} \sum_{j_1=0}^1 \cdots \sum_{j_n=0}^1 \exp\left(2\pi i k \left(\sum_{l=1}^n j_l 2^{-l}\right)\right) |j_1\rangle \otimes \cdots |j_n\rangle \quad (4.6)$$

$$= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^n \sum_{j_l \in \{0,1\}} \exp\left(2\pi i k j_l 2^{-l}\right) |j_l\rangle \quad (4.7)$$

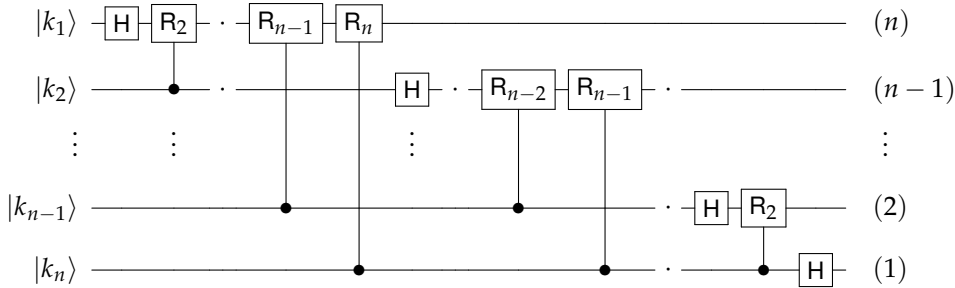
$$= \bigotimes_{l=1}^n \frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(2\pi i k 2^{-l}\right) |1\rangle \right) \quad (4.8)$$

$$= \frac{1}{\sqrt{2}} \left(|0\rangle + \exp(2\pi i 0.k_n) |1\rangle \right) (\cdots) \frac{1}{\sqrt{2}} \left(|0\rangle + \exp(2\pi i 0.k_1 \cdots k_n) |1\rangle \right). \quad (4.9)$$

In terms of controlled versions of the quantum gates

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & \exp(2\pi i 2^{-k}) \end{pmatrix}, \quad k = 2, \dots, n \quad (4.10)$$

this immediately suggests a quantum circuit as



with the corresponding product of output quantum states

$$(n) = \frac{1}{\sqrt{2}} \left(|0\rangle + \exp(2\pi i 0.k_1 \cdots k_n) |1\rangle \right) \quad (4.11)$$

$$(n-1) = \frac{1}{\sqrt{2}} \left(|0\rangle + \exp(2\pi i 0.k_2 \cdots k_n) |1\rangle \right) \quad (4.12)$$

$\vdots$

$$(2) = \frac{1}{\sqrt{2}} \left(|0\rangle + \exp(2\pi i 0.k_{n-1} k_n) |1\rangle \right) \quad (4.13)$$

$$(1) = \frac{1}{\sqrt{2}} \left(|0\rangle + \exp(2\pi i 0.k_n) |1\rangle \right). \quad (4.14)$$

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

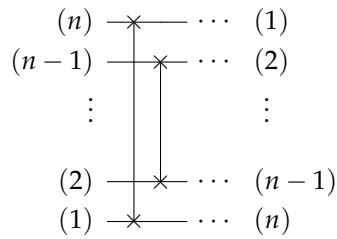
$$|k\rangle = |k_1 \cdots k_n\rangle \mapsto \frac{1}{\sqrt{2}} (|0\rangle + \exp(2\pi i 0.k_1)|1\rangle) \otimes |k_2 \cdots k_n\rangle \quad (4.15)$$

$$\mapsto \frac{1}{\sqrt{2}} (|0\rangle + \exp(2\pi i 0.k_1 \cdots k_n)|1\rangle) \otimes |k_2 \cdots k_n\rangle, \quad (4.16)$$

and then going through all the other qubits as

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

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

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

many conditional rotations. Additionally, we have the $\lfloor n/2 \rfloor$ swap gates in the end. 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 is $\mathcal{O}(n^2)$, an exponential improvement over $\mathcal{O}(n2^n)$ gates for the classical FFT!

Can the QFT be used to process classical signals exponentially faster? Unfortunately, the answer is no, as the QFT does *not* solve the *end-to-end* problem as the DFT does. Namely, the DFT takes as input the classical description of a vector $(x_0, \cdots, x_{N-1})$ and outputs the classical description of the Fourier-transformed vector $(y_0, \cdots, y_{N-1})$. 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, it can be shown that the end-to-end complexity of the QFT to perform the DFT task –i.e., when complemented by routines that solve these two problems– 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 sections we sketch how QFT is the core quantum subroutine of Shor’s integer factorization algorithm.

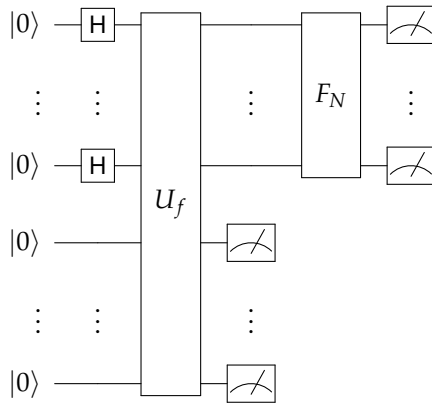
4.3 Period finding

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 key problem for which the QFT can be applied is *period finding*⁵:

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

$$f(x) = f(y) \Leftrightarrow x \equiv y \pmod{c}, \quad (4.19)$$

the task is to determine c . Notice the similarity with Simon’s problem from Section 3.4! For period finding, one can show⁶ 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 using the quantum circuit



which features one QFT for each of the $\mathcal{O}(\log \log N)$ runs. This is similar—but not equivalent, not even for $N = 2$ —to Simon’s problem from Section 3.4, and you will explore in Exercise Sheet 4 how above quantum circuit works in detail and why it solves the period finding problem.

4.4 Order finding and Shor’s algorithm

Now, while for period finding, this is an exponential classical-quantum separation, it is (at first) again only in a query complexity

⁵ This should be distinguished from *order finding* discussed below. This is not always clear in the literature.

⁶ Richard Cleve. The query complexity of order-finding. *Inform. And Comput.*, 192(2):162–171, 2004

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

$$f_a(x) = a^x \bmod N \quad (4.20)$$

can be found for any fixed $a \in \{0, \dots, N-1\}$. Luckily, there is already an efficient classical algorithm to implement $f_a(x)$ for $a \in \{0, \dots, N-1\}$ 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 $\mathcal{O}(n^2 \log n \log \log n)$ for integer factorization of n -bit integers.⁷ This can be improved by using quasi-linear multiplication⁸ to the claimed $\mathcal{O}(n^2 \log n)$.

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 various problems related to integer factorization — a.k.a. *hidden subgroup problems* — for which one can employ similar ideas to find super-polynomial end-to-end 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 thesis, Shor's efficient quantum algorithm also uproots the security of a widespread public key cryptosystem, the Rivest–Shamir–Adleman (RSA) scheme, which relies on the hardness of integer factorization.

Since integer factorization and more generally computational number theory are otherwise not necessarily of broad interest for scientific computing, we will skip the details on the proof of Shor's algorithm⁹ 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 may indeed be more broadly employed for scientific computing.

⁷ Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303, 1999

⁸ David Harvey and Joris van der Hoeven. Integer multiplication in time $\mathcal{O}(n \log n)$. *Annals of Mathematics*, 193(2):563, 2021

⁹ *Quantum Information* (M. Berta), RWTH Physics

5

Quantum phase estimation

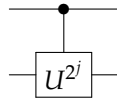
5.1 Task

We are given a $2^m \times 2^m$ unitary matrix U and an accompanying normalized eigenvector $\vec{u}$ with eigenvalue parameterized by a phase,

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

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 some m -qubit quantum gate $V(u)$ ¹ and can run controlled versions of powers of the quantum gate U^{2^j} with $j = 0, \dots, n-1$ for some $n \in \mathbb{N}$:

¹ This loading of a *known* classical vector into a quantum state is discussed in Chapter 9



The goal of *quantum phase estimation* is to determine an n -bit number that is equal to the best n -bit approximation $\phi^{(n)}$ of the phase ϕ of the eigenvalue $\exp(2\pi i \phi)$ while allowing the algorithm to fail with probability $\varepsilon \in (0, 1]$.

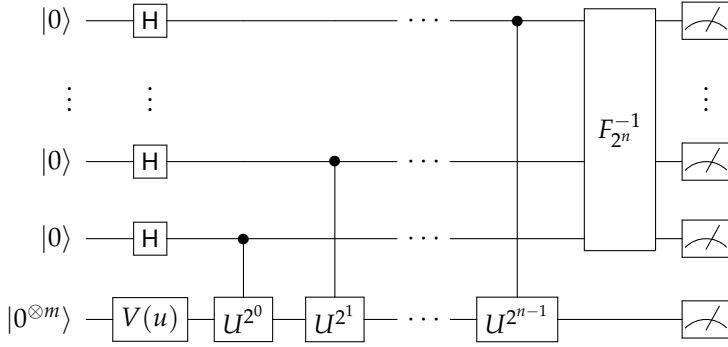
Quantum phase estimation on its own is still a query complexity algorithm. Indeed, there are a lot of assumptions here: Not only do we have some m -qubit quantum gate $V(u)|0^{\otimes m}\rangle = |u\rangle$ for preparing the relevant eigenvector, but we also have access to a black-box quantum gate implementation of the controlled version of U and its higher exponential 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 5.5), we will see in Chapters 7 and 8 that phase estimation is a useful basic subroutine in quantum algorithms for scientific computing.

5.2 Deterministic n -bit phase “estimation”

For simplicity, we first treat the special case in which the phase ϕ can be written exactly as a fraction with an n -bit string, i.e.,

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

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



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

$$\begin{aligned} & |0^{\otimes n}\rangle \otimes |0^{\otimes m}\rangle \\ \mapsto & \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \otimes |u\rangle \end{aligned} \quad (5.3)$$

$$\mapsto \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \otimes U^j |u\rangle \quad (5.4)$$

$$= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \exp(2\pi i j \phi) |j\rangle \otimes |u\rangle \quad (5.5)$$

$$= \frac{1}{\sqrt{2^n}} \left(|0\rangle + \exp(2\pi i 2^{n-1} \phi) |1\rangle \right) (\cdots) \left(|0\rangle + \exp(2\pi i 2^0 \phi) |1\rangle \right) \otimes |u\rangle \quad (5.6)$$

where we applied the controlled- U^{2^j} quantum gates for $j = 0, \dots, n-1$ in the second step, and the last equality follows from 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

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

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 $(\phi_1, \dots, \phi_n)$. The corresponding binary fraction (5.2) is exactly equal to the sought after n -bit phase ϕ . Here we are thus not really estimating the phase but are able to measure it directly with probability 1.² The query complexity stays efficient: there are

² Throughout the m -qubit ancilla register remains *independently* in the eigenstate $|u\rangle$ and the way it is measured at the end is irrelevant: one can verify that the probability distribution of ancilla and QFT register measurements factorizes.

n calls to black-box powers of U and we have seen in the previous Chapter 4 that the QFT can be implemented in complexity $O(n^2)$.

5.3 Phase estimation with approximate eigenvector

In the following, we address and relax some of the assumptions that went into the most basic quantum phase estimation algorithm, starting with the easiest one: What happens if one does not have the ability to prepare exactly the relevant eigenstate $|u\rangle$, but rather can only prepare some other state $|\psi\rangle$? Writing $|\psi\rangle = \sum_u \alpha_u |u\rangle$ as a superposition in terms of all the eigenstates of U (which constitute an orthonormal basis of the whole space), the quantum phase estimation algorithm will — just by linearity — output an estimate of the corresponding phase ϕ_u with probability $|\alpha_u|^2$ for each u , see Exercise Sheet 5. Thus, if we have some a priori approximation $|\psi\rangle$ to the eigenvector $|u\rangle$, this might still be useful. We will use this in Chapter 7 to resolve the energy levels of Hamiltonians of quantum many-body systems, for which typically only an *ansatz* for the ground state is available.

5.4 Phase estimation with accuracy vs. failure trade-off

More subtle is obtaining an accurate *estimate* of the phase in the general case in which it is not equal to an n -bit string, $\phi \neq \phi^{(n)}$, and in which estimation can fail. In this case the algorithm also becomes probabilistic. However, given a desired upper bound $\varepsilon \in (0, 1]$ on the failure probability, we can choose t instead of n qubits on the QFT register, where

$$t = n + \left\lceil \log \left(2 + \frac{1}{2\varepsilon} \right) \right\rceil > n \quad (5.8)$$

After measuring some bit string $\hat{\phi}_1 \cdots \hat{\phi}_t$, the binary fraction

$$0.\hat{\phi}_1 \cdots \hat{\phi}_t = \frac{\hat{\phi}_1}{2^1} + \cdots + \frac{\hat{\phi}_t}{2^t} \quad (5.9)$$

gives an estimate that can then be shown to approximate ϕ to n bits of accuracy,

$$|0.\hat{\phi}_1 \cdots \hat{\phi}_t - \phi| \leq \frac{1}{2^n} \quad (5.10)$$

with probability at least $1 - \varepsilon$. Roughly speaking, for fixed QFT register size t in Eq. (5.8) we can trade binary powers of failure probability (ε) for bits of estimation accuracy (n). This trade-off will be used for solving linear systems of equations in Chapter 8.

5.5 Caveats and state of the art

Besides the m qubits needed to represent the unitary matrix U , QPE is a priori exponentially expensive in the number of qubits n controlling the precision if one only has a black-box implementation

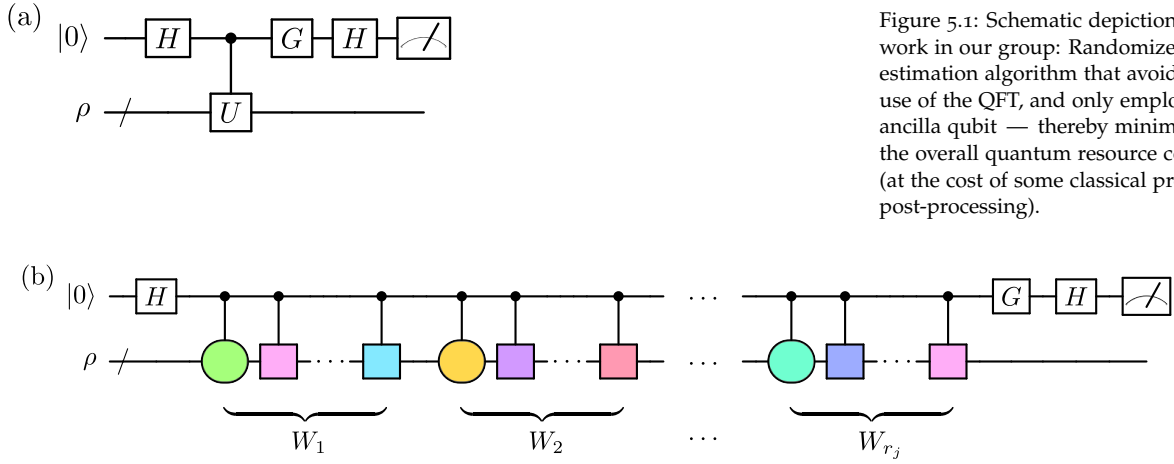


Figure 5.1: Schematic depiction of work in our group: Randomized phase estimation algorithm that avoids the use of the QFT, and only employs one ancilla qubit — thereby minimizing the overall quantum resource costs (at the cost of some classical pre- and post-processing).

of U — not its controlled version — at hand: One needs to implement the controlled U operations up to exponentially high powers $U^{2^{n-1}}$. This also means that increasing $n \rightarrow t$ [Eq. (5.8)] to suppress the failure probability of QPE only costs a few extra ancilla qubits $t - n \approx \log(1/2\epsilon)$, but these exponentially increase the required powers of U . In practice, this can be prohibitively expensive.

However, if more is known about the quantum circuit implementation of the black box U , this exponential blow-up can be avoided for problems with specific *structure*. This is the case, for example, in Shor’s algorithm for integer factorization³ and its variations. This allows to solve a whole class of so-called hidden-subgroup problems achieving an exponential separation relative to the (best known) classical end-to-end complexity of these problems.

Moreover, the QFT subroutine used in phase estimation is also relatively complex in practice in terms of quantum gate cost, and it is not clear whether it has noise resilience⁴ as, for example, some QRAM circuits have (Chapter 9). Luckily, there are other modern versions of quantum phase estimation that avoid using the QFT⁵ and require only *one* ancilla qubit end to end.⁶ Some versions also make extensive use of classical signal processing methods.⁷

In the next chapter, we discuss unitary matrices of the special form $U = \exp(-iHt)$, with Hermitian H given by a physical Hamiltonian, and how to implement them efficiently. This allows one to use phase estimation for ground-state energies in Chapter 7.

³ Quantum Information (M. Berta), RWTH Physics

⁴ 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

⁵ Lin Lin and Yu Tong. Heisenberg-limited ground-state energy estimation for early fault-tolerant quantum computers. *PRX Quantum*, 3:010318, 2022

⁶ Kianna Wan, Mario Berta, and Earl T. Campbell. Randomized quantum algorithm for statistical phase estimation. *Physical Review Letters*, 129:030503, 2022

⁷ Thomas E. O’Brien, Brian Tarasinski, 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 system 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

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

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

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

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

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

and as such can be implemented in the quantum circuit model. In general, this will require $O(n \cdot 4^n)$ 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 use an *approximate notion* of Hamiltonian simulation, where for a specified time t we implement a unitary matrix $\bar{U}(t)$ such that

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

Recall from around Eq. (2.10) that the infinity norm is defined as

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

and hence makes the approximation uniform over all input states.

In Exercise Sheet 6, you will prove that for sequences of unitary matrices $\{U_i\}_{i \in I}$ and $\{\bar{U}_i\}_{i \in I}$, we have:

$$\|\bar{U}_i - U_i\|_\infty \leq \varepsilon \forall i \in I \Rightarrow \left\| \prod_{i \in I} \bar{U}_i - \prod_{i \in I} U_i \right\|_\infty \leq |I|\varepsilon. \quad (6.6)$$

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

In general, quantum physical systems with Hamiltonians H first have to be mapped to qubits, i.e., to a complex matrix in $\mathbb{C}^{2^n \times 2^n}$, 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 $\mathbb{C}^{2^n \times 2^n}$, given as

$$\{X, Y, Z, \mathbb{1}\}^{\otimes n}. \quad (6.7)$$

So, an example element is of the form

$$Z_1 \otimes X_2 \otimes \cdots \otimes \mathbb{1}_{n-1} \otimes Y_n, \quad (6.8)$$

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

Example 16. *The Heisenberg chain on a one-dimensional line is described by the Hamiltonian*

$$H_{\text{Heisenberg}} = \sum_{j \in J} \left(C_x \cdot X_j \otimes X_{j+1} + C_y \cdot Y_j \otimes Y_{j+1} + C_z \cdot Z_j \otimes Z_{j+1} \right), \quad (6.9)$$

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

$$X_j \otimes X_{j+1} \equiv \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{j-1} \otimes X_j \otimes X_{j+1} \otimes \mathbb{1}_{j+2} \otimes \cdots \otimes \mathbb{1}_n \quad (6.10)$$

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

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

Example 17. The Ising model on a two-dimensional square lattice is described by the Hamiltonian

$$H_{\text{Ising}} = C \cdot \sum_{(i,j) \in J} (Z_{i,j} \otimes Z_{i+1,j} + Z_{i,j} \otimes Z_{i,j+1}), \quad (6.11)$$

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, each of these Pauli terms acts non-trivially only in a *local neighborhood of size k* ; in the above examples, for instance, the terms act only on nearest neighbors with $k = 2$.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

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

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

$$\exp(H) \neq \exp(H_1) \exp(H_2), \quad (6.13)$$

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

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

⁴ These are also called *commuting* Hamiltonians.

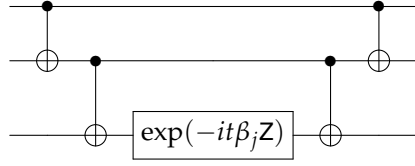
and the quantum circuit representation problem reduces to implementing one n -qubit term $\exp(-it \beta_j P_j)$ with $P_j \in \{X, Y, Z, \mathbb{I}\}^{\otimes n}$. Diagonalizing the Pauli matrices on each qubit with single qubit unitaries U_l and using that the eigenvalues are all ± 1 , we can write

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

which only acts non-trivially on a number $k_j \leq n$ of the qubits.

The remaining unitary matrix $\exp(-it \beta_j Z^{\otimes k_j})$ 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 $\exp(-it\beta_j P_j)$ 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 $|J| = \text{poly}(n)$.

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 nevertheless to use the same product formula⁵

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

in the hope that it still reasonably approximates $\exp(-itH)$ 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

We resort to the *Lie-Trotter expansion*

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

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

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

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

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.

Taylor expansion

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

gives the estimate

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

via the choice

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

Iteratively applying this to the multipartite case under the boundedness conditions $\|H_j\|_\infty \leq \gamma \ \forall j \in J$ and $|J|\gamma = O(1)$ gives

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

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

$$\begin{aligned} &\exp(H_1) \exp(H_2) \exp(H_3) \exp(H_4) \\ &= \left(\exp(H_1 + H_2) + E_{12} \right) \exp(H_3) \exp(H_4) \end{aligned} \quad (6.23)$$

$$\begin{aligned} &= \left(\exp(H_1 + H_2 + H_3) + E_{123} \right) \exp(H_4) \\ &\quad + E_{12} \exp(H_3) \exp(H_4) \end{aligned} \quad (6.24)$$

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

with the corresponding bound

$$\|E_4\|_\infty \leq \|E_{1234}\|_\infty + \|E_{123} \exp(H_4)\|_\infty + \|E_{12} \exp(H_3) \exp(H_4)\|_\infty \quad (6.26)$$

$$\leq (4-1)^2 \cdot O(\gamma^2) + (4-2)^2 \cdot O(\gamma^2) \cdot O(1) + O(\gamma^2) \cdot O(1) \quad (6.27)$$

$$\leq 4(4-1)^2 \cdot O(\gamma^2) \leq 4^3 \cdot O(\gamma^2). \quad (6.28)$$

So, in this small-step regime, the first order Trotter expansion gives an approximation

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

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

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

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

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

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

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

and hence there exists a constant C such that

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

Choosing

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

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

Typically, the number of terms is $|J| = \text{poly}(n)$ and the coefficients satisfy $\beta = O(1)$. 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

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

Using a higher-order product formula related to the Lie–Trotter expansion in Eq. (6.17), one can asymptotically improve this to

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

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 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

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

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

You will show in Exercise Sheet 7, that truncating at

$$k = O\left(t \|\vec{\beta}\|_1 + \log\left(\varepsilon^{-1}\right)\right) \quad (6.39)$$

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

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

$$P = \sum_{l \in L} \alpha_l P_l \text{ with } \alpha_l \geq 0 \text{ and unitary } P_l, \quad (6.40)$$

where signs and complex phases are absorbed into the unitaries P_l 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

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

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

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

3. Apply the unitary

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

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

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

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

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

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

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

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

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

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

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 $O(1/\sqrt{p_{\text{good}}})$ 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

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

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 non-negative coefficients $\vec{\alpha} = (\alpha_l)_{l \in L}$. 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 glossed over here. First, the above scheme requires repeated coherent access to the preparation of the input state $|\psi\rangle$ (even though the task of Hamiltonian simulation is formulated for one input copy). This can be avoided 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

$$\exp\left(-it \sum_{j \in J} \beta_j P_j\right) \approx_\varepsilon \sum_{k=0}^{O(t\|\vec{\beta}\|_1 + \log \varepsilon^{-1})} \sum_{(j_1 \dots j_k) \in J^k} \underbrace{\frac{t^k |\beta_{j_1} \dots \beta_{j_k}|}{k!}}_{=\alpha_{j_k}} \cdot \underbrace{(-i)^k \operatorname{sgn}(\beta_{j_1} \dots \beta_{j_k}) P_{j_1} \dots P_{j_k}}_{=P'_{j_1 \dots j_k}}, \quad (6.47)$$

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

$$\|\vec{\alpha}\|_1 = \sum_{k=0}^{O(t\|\vec{\beta}\|_1 + \log \varepsilon^{-1})} \frac{t^k}{k!} \sum_{(j_1 \dots j_k) \in J^k} |\beta_{j_1} \dots \beta_{j_k}| \quad (6.48)$$

$$\leq \sum_{k \geq 0} \frac{(t\|\vec{\beta}\|_1)^k}{k!} \quad (6.49)$$

$$= \exp(t\|\vec{\beta}\|_1), \quad (6.50)$$

where $\vec{\beta} = (\beta_1, \dots, \beta_{|J|})$. 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

$$(\exp(-isH))^p = \exp(-ispH). \quad (6.51)$$

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

$$\text{time } s = t/p \leq \|\vec{\beta}\|_1^{-1} \text{ and error } \bar{\varepsilon} = \varepsilon/p. \quad (6.52)$$

Now, for each of these runs, the number of iterations of the algorithmic steps 1–4 is $O\left(\exp(s\|\vec{\beta}\|_1)\right) = O(1)$ 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 $P'_{j_1} \dots P'_{j_k}$ from Eq. (6.47), with k up to

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

In terms of elementary gates, this can be decomposed into

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

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

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

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

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

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

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

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

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

ancilla qubits.

6.5 State-of-the-art methods and caveats

There are many conceptually different quantum algorithms for performing Hamiltonian simulation, each with its 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 is the randomized scheme 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, for an (α, a) -block encoding of the Hamiltonian, the query complexity is¹³

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

which is known to be asymptotically optimal. For the Pauli LCU considered here, one may take $\alpha = \|\vec{\beta}\|_1 \leq |J|$. However, one first needs 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 $O(\log N)$ system qubits and $\text{poly}(\log N)$ gates to time-evolve exponentially large n -qubit states.

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

In comparison, 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, estimating an observable expectation value to additive precision $\delta > 0$ from independent measurements generally requires $\Theta(\delta^{-2})$ copies, and hence the quantum experiment must be repeated many times. Let alone resolving a full classical description of the state, which would again need $O(2^n)$ copies!
- For important structured systems, specialized modern classical methods can perform far better than the generic $O(2^n)$ representation and may scale polynomially over practically accessible regimes. Their accuracy and scaling remain strongly problem-dependent and can deteriorate for larger or more strongly correlated instances (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 correspond to the energy levels of the system. While the whole spectrum is of interest, many systems are observed in low-energy states, and hence 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

Given an n -qubit Hamiltonian of the form

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

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, while a decision version of the former problem is known to be as hard as any problem in NP.² Namely, for Boolean variables $\vec{x} = (x_1, \dots, x_n)$ and a given formula $f(\vec{x})$ based on AND, OR, and NOT operators, the 3-SAT problem asks if there are values $\vec{x} = (x_1, \dots, x_n)$ such that the formula becomes true. For example, for $n = 6$, an instance is

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

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

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

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

can associate an $n = 6$ qubit Hamiltonian term

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

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

$$H = \sum_{j \in J} H_{C_j} \quad (7.4)$$

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, the Cook–Levin theorem together with the standard reduction to 3-SAT³ implies NP-hardness for a decision version of the local Hamiltonian problem stated above.

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 QMA-complete and therefore believed to be hard even for a quantum computer. However, these worst-case 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

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

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 statements assuming that we have an Ansatz state of a 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.

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

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 16) and the two-dimensional Ising model (Example 17). However, these models were already natively given in qubit form because they describe lattice models, where a local spin degree of freedom is modeled directly as a qubit and the interactions are expressed using Pauli matrices.

Another example is provided by *fermionic systems*, such as electrons in 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 choosing an ansatz with n single-particle basis states, the resulting effective Hamiltonian can be written as⁵

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

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

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

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

$$\{a_p, a_q\} = \{a_p^\dagger, a_q^\dagger\} = 0, \quad (7.8)$$

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

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

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

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

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

Other more involved mappings that reduce this non-locality are possible as well, and you will explore them 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 phenomenon 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 $O(n^4)$ 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 arise in nuclear physics or models of quantum gravity. However, we refrain from treating those, partly because the quantum resources to simulate and resolve such systems seem 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

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

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

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 $\lambda_0(H)$. Set $B = \sum_{j \in J} |\beta_j|$, so that $\|H\|_{\infty} \leq B$. To keep things simple, we assume that the scaled phases

$$\phi_i = \frac{1}{4} \left(\frac{\lambda_i(H)}{B} + 1 \right) \in [0, 1/2] \quad (7.13)$$

have exact d -bit binary representations, and that

$$\lambda_0(H) < \lambda_1(H) \leq \lambda_2(H) \leq \dots, \quad (7.14)$$

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

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

$$U = \exp\left(\frac{i\pi}{2}(\bar{H} + \mathbb{1})\right) \text{ with } \bar{H} = H/B, \quad (7.15)$$

and input state

$$|\psi_{\text{Ansatz}}\rangle = \langle \psi_{\text{Ansatz}} | \psi_0 \rangle \cdot |\psi_0\rangle + \sum_{j=1}^{2^n-1} \alpha_j |\psi_j\rangle, \quad (7.16)$$

written in the eigenbasis $\{|\psi_j\rangle\}_{j=0}^{2^n-1}$ of H . The unitary U has the same eigenvectors and eigenphases ϕ_j , from which the energies are recovered as $\lambda_j(H) = B(4\phi_j - 1)$.

In more detail, employing the polynomial-time quantum circuit from Section 5.2 on $n + d$ qubits requires controlled powers up to $U^{2^{d-1}}$. Up to an easily implemented global phase, the total Hamiltonian evolution time across these powers is proportional to

$$\sum_{k=0}^{d-1} 2^k = 2^d - 1. \quad (7.17)$$

The controlled powers can be implemented by Hamiltonian simulation as discussed in Chapter 6. With exact simulation, measuring the phase register gives the ground-state energy with probability γ_{Ansatz} . Repeating the circuit and retaining the smallest observed energy

$$O\left(\gamma_{\text{Ansatz}}^{-1}\right) \text{ times} \quad (7.18)$$

gives constant success probability. Using, for example, the LCU methods—which come at the cost of a few additional ancilla qubits—the total Hamiltonian simulation cost roughly scales as

$$O\left(2^d \cdot \text{poly}(n)\right) \quad (7.19)$$

when the simulation error is chosen sufficiently below the target phase precision, for example of order 2^{-d} .⁷

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 phases can be dropped. However, any such scheme retains 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 5.5, modern work goes into achieving favorable scaling with respect to the 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 A full analysis must track the error propagation through the quantum phase-estimation circuit.

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 a generic quantum state has a squared overlap of order 2^{-n} with any fixed state, including 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 often do not provide certified end-to-end error bounds — even if they start from *first principles* — and the resulting schemes may therefore remain 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 built 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 path and scheduling function have to be chosen such that the spectral gap $\Delta(H(t))$ does not become exponentially small, while the Hamiltonian changes sufficiently slowly relative to this gap. 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 equilibrium when interacting with their 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 indica-

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

tions on practical run times. We will not explore Gibbs-state preparation further. Many proposals use additional primitives, including the dynamics of *open quantum systems*.⁹

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, the primary cofactor of nitrogenase, which 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 all exponentially many amplitudes of a generic quantum state. Instead, they exploit locality and additional physical structure to construct useful Ansätze. Their performance is problem dependent, so a quantum algorithm has to be compared with the strongest classical method for the same instance class, rather than only with the generic 2^n dimension bound.

For further discussions specifically on computational quantum chemistry, we refer to the review article¹² and a recent critical discussion on potential quantum speed-up.¹³

⁹ Chi-Fang Chen, Michael J. Kastoryano, and András Gilyén. An efficient and exact noncommutative quantum Gibbs sampler. 2023. arXiv:2311.09207

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

$$\vec{x} = A^{-1}\vec{b} \quad (8.1)$$

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 $\mathcal{O}(N^3)$. Fast matrix multiplication can reduce the arithmetic complexity to $\mathcal{O}(N^{\omega+o(1)})$. For matrices over $\mathbb{C}$, the current bound is $\omega \leq 9/4 = 2.25$.² 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.

The applications of this problem to scientific computing are indirect, but fundamental and numerous. They include solving differential equations, subroutines in machine learning, and interior-point methods for optimization problems. 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

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 $N \neq M$.

² OpenAI. An upper bound of $9/4$ for the matrix multiplication exponent. OpenAI Math Release preprint, 2026b. OpenAI Math Release OAI:Matrix-Multiplication-Nine-Fourths-October-2-2026, 2 October 2026

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* $\kappa(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty$ 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*,³ which, given access to the vector two-norms of the rows, 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

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

for the row sparsity $s(A)$ and the *Frobenius 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 *preconditioners*, which perform preprocessing to bring the linear system into a form with well-scaling, reduced condition number. These preconditioning techniques are problem dependent, and their construction cost must be included when comparing them with the main routines.

There are other, *quantum-inspired* or *dequantized* classical randomized linear system solvers that roughly scale as⁴

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

where the tilde denotes up to poly-logarithmic correction terms. By inspection, the displayed query complexity is independent of the dimension N and the sparsity $s(A)$. However, these techniques use a nonstandard sample-and-query input model, whose data-structure construction and preprocessing costs must be counted in an end-to-end comparison. They also have worse approximation-error scaling that is no longer logarithmic, and they only allow one 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 Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15:262, 2009

⁴ 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, high approximate rank can rule out the low-rank methods discussed above, even when $s(A) \ll N$. The performance of the remaining classical methods then depends on sparsity, conditioning, and the available data access. 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 reduce a general square matrix to a Hermitian block matrix and pad the dimension to a power of two. For simplicity, we therefore focus on Hermitian matrices $A = A^\dagger$ of dimension $N = 2^n$ and normalize them such that $\|A\|_\infty = 1$.

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

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 allows one to estimate certain properties of $\vec{x}$, such as 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 .⁵ Nevertheless, the above suggests that quantum computers can sample from the solution of linear systems of equations of dimension $N = 2^n$ with only n qubits space cost!

⁵ 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)

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

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)

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

where the last register uses d 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 a computational-basis register. 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)

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

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

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

The complexity of the QLSS is then quantified by the number of times $U_{A,\text{entry}}$, $U_{A,\text{pos}}$, and their 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 in⁶ and is based on the quantum phase estimation routine (Chapter

⁶ Aram W. Harrow, Avinandan Hasidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103: 150502, 2009

5).⁷

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. Then, there is an eigendecomposition

$$A|v_j\rangle = \lambda_j|v_j\rangle \quad (8.9)$$

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

$$0 < \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{2^n-1} = 1. \quad (8.10)$$

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 $U = \exp(i\pi A)$ on the input state $|b\rangle$. Its eigenphases are $\lambda_j/2 \in (0, 1/2]$, so there is no ambiguity between the largest eigenvalue and phase zero. After rescaling the phase register, the idealized output has the form

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

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

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

is the expansion of $|b\rangle$ in the eigenbasis of A . Note that running U_{QPE} requires implementing controlled Hamiltonian simulation $\exp(i\pi t A)$ 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 $U_A = \exp(i\pi A)$ 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

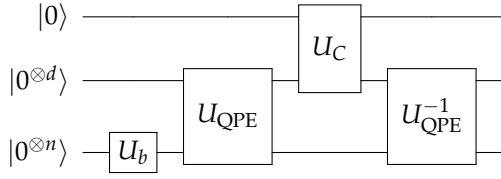
$$A^{-1}|b\rangle = \sum_{j=0}^{2^n-1} \frac{\beta_j}{\lambda_j} |v_j\rangle \quad (8.13)$$

and towards that, we employ the unitary U_C defined via

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

controlled on an additional ancilla qubit, and where $\tilde{\lambda}_j$ approximates λ_j , and C denotes some (for now) unspecified normalization constant satisfying $0 < C \leq \lambda_0$. 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

⁷ 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).



and the overall output state becomes

$$\begin{aligned}
 & U_{\text{QLSS}}(|0\rangle \otimes |0^{\otimes d}\rangle \otimes |b\rangle) \\
 &= \sum_{j=0}^{2^n-1} \beta_j \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} \cdot |0\rangle + \frac{C}{\tilde{\lambda}_j} \cdot |1\rangle \right) \otimes |0^{\otimes d}\rangle \otimes |v_j\rangle. \quad (8.15)
 \end{aligned}$$

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

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

on the remaining main n qubits. Choosing the eigenvalue precision appropriately makes this the sought-after approximation, with $\| |y\rangle - |x\rangle \|_2 \leq \varepsilon$ as required by Eq. (8.4).

The success probability of measuring one is

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

$$\approx C^2 \cdot \|A^{-1}|b\rangle\|_2^2 \quad (8.18)$$

and the ideal choice $C = \lambda_0$ optimizes this probability. In an algorithm, one instead uses a known lower bound $C \leq \lambda_0$. For the ideal choice, we find

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

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

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

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

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

one needs to provide the *multiplicative* approximation

$$|\tilde{\lambda}_j - \lambda_j| \leq (\varepsilon/4)\lambda_j \quad \forall j. \quad (8.22)$$

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 $U_A = \exp(i\pi A)$ becomes $\mathcal{O}(\kappa(A)\varepsilon^{-1})$.

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

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

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

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

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

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

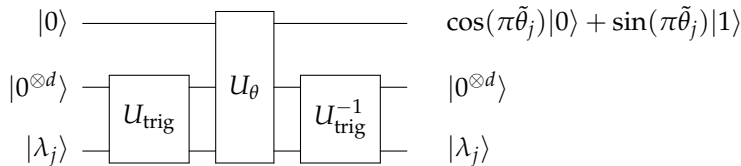
Then, given a d -bit representation of an angle θ , you will show in Exercise Sheet 10, how to implement the $(d+1)$ -qubit unitary U_θ acting as

$$U_\theta : |0\rangle \otimes |\theta\rangle \mapsto (\cos(\pi\theta)|0\rangle + \sin(\pi\theta)|1\rangle) \otimes |\theta\rangle \quad (8.25)$$

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

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

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 .⁸

The implementation of powers of $U_A = \exp(i\pi A)$ corresponds to Hamiltonian simulation of the Hamiltonian A , with variable times

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

t up to $t = \mathcal{O}(\kappa(A)\varepsilon^{-1})$. 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 their 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 for an ε -approximate Hamiltonian simulation with

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

in terms of the query complexity to $U_{A,\text{entry}}$, $U_{A,\text{pos}}$ and their 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.⁹

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

- $\mathcal{O}(\kappa(A)\varepsilon^{-1} \cdot \text{poly}(n, s(A)))$ many queries to $U_{A,\text{entry}}$, $U_{A,\text{pos}}$, and their 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 $s(A) = \text{poly}(n)$ in both rows and columns, and that U_b can be implemented in $\text{poly}(n)$ gates, this leads to the complexity

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

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.)

⁹ Strictly speaking one would have to carefully monitor the error propagation.

8.5 State-of-the-art methods and caveats

There are various technical assumptions we made that can be dropped:¹⁰

- 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.

¹⁰ 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

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

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

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 their inverses 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.¹¹ There are various further caveats that stay with QLSSs and any potential application must take these into account:

- 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 even dominant.
- The quantum state $|y\rangle$ output by the QLSS does not directly reveal any classical information about the solution of the linear system of equations. To estimate an overlap $\langle\phi|x\rangle$ with some fixed n -qubit state $|\phi\rangle$, one needs an interference procedure such as a Hadamard test and 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

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

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.,¹²)

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

¹¹ Alexander M. Dalzell. A shortcut to an optimal quantum linear system solver. 2024. arXiv:2406.12086

¹² 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

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 use suitable linear-system preconditioners.

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).¹³

¹³ Samson Wang, Sam McArdle, and Mario Berta. Qubit-efficient randomized quantum algorithms for linear algebra. *PRX Quantum*, 5:020324, 2024

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

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 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.

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

9.2 Quantum state preparation: Basic ideas

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

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

from a given classical data vector $\vec{b} = (b_0, \dots, b_{2^n-1})$ of length 2^n . This is a necessary subroutine 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 $\text{poly}(n)$ complexity to have the potential for large quantum speed-up.

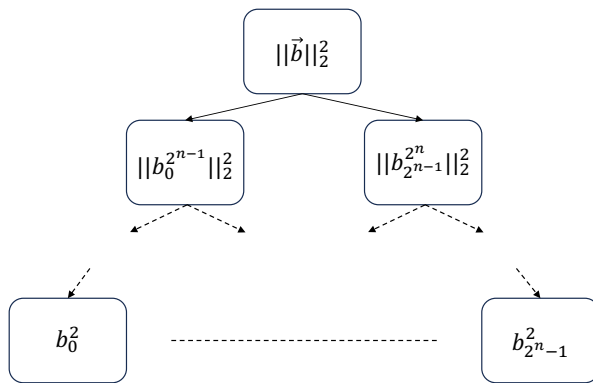
Unfortunately, information-theoretic lower bounds imply that preparing a generic state with the 2^n independent amplitudes in Eq. (9.1), to fixed precision using a fixed finite gate set, requires $\Omega(2^n)$ gates.

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 $\mathcal{O}(n)$ -depth construction, following the presentation in Ref. ⁴.

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

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

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



Based on this tree data, one can then construct the n -qubit state

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

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

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

$|b\rangle$ 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

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

is applied. This creates the state

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

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

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

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

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

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

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

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

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

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

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

for the values

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

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

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

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

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

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

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

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

$$= |b(k+1)\rangle. \quad (9.17)$$

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

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

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

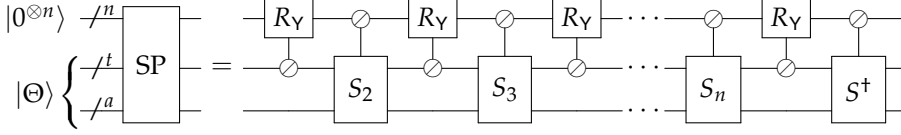
Given a binary tree data structure as depicted in Figure 9.1 with $2^{n+1} - 1$ nodes, we assume that the angles θ_i for $i = 1, \dots, 2^n - 1$ are classically precomputed to avoid the need for any arithmetic on the quantum computer. We further assume that the $2^n - 1$ angles have an exact t -bit representation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

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

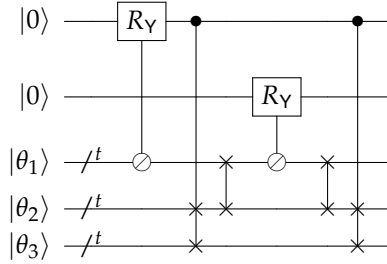
where the latter is also a computational basis state that can easily be created by a single layer of Pauli X gates on the appropriate qubits determined by the classical data angles. The goal is to implement the *state preparation* routine

$$\text{SP}(|0^{\otimes n}\rangle \otimes |\Theta\rangle) = |b\rangle \otimes |\Theta\rangle, \quad (9.20)$$

which is accomplished by the following quantum circuit



Here, $a = (2^n - 2)t$ denotes all but t ancilla qubits, the symbol $\otimes$ indicates that a different rotation angle is performed for each of the 2^t possible settings of the register, S_p for $p = 2, \dots, n$ swaps the t -bit description of the next angle into the first ancilla register of size t , and $S^+ = S_2^+ \dots S_n^+$ 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 $S_2, \dots, S_n$ can be written as

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

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

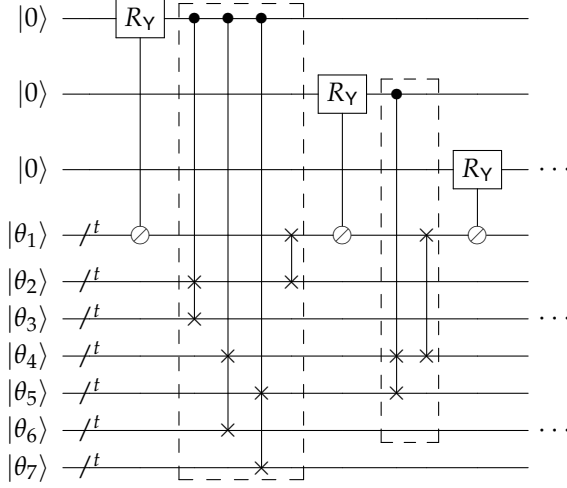
$$S_p^{(j)} S_{p-1}^{(j)} \dots S_1^{(j)} \quad (9.22)$$

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

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

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^+ 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

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 precomputed classically, but in principle this could also be done on a quantum computer 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.

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. arXiv:2108.06150

9.4 Quantum read only memory (QROM)

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

$$U_N : |i\rangle \otimes |0^{\otimes t}\rangle \mapsto |i\rangle \otimes |x_i\rangle. \quad (9.24)$$

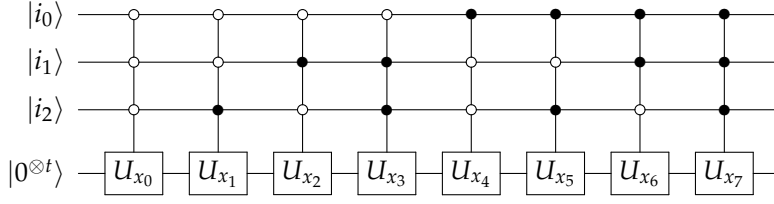
One unitary extension of this action is

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

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 $\{x_0, \dots, x_{2^n-1}\}$ 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 on the n input address qubits and t bus qubits is via *unary iteration*. Additional work qubits may be introduced when the multi-controlled gates are decomposed into elementary gates. 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 $\{x_0, \dots, x_{2^n-1}\}$ 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 $|j\rangle$. For fixed word size t , this results in $\mathcal{O}(2^n)$ logical data-gate applications and depth. The elementary gate count also depends on how the multi-controlled operations and the data gates U_{x_i} are implemented.

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 $\text{poly}(n)$. 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.

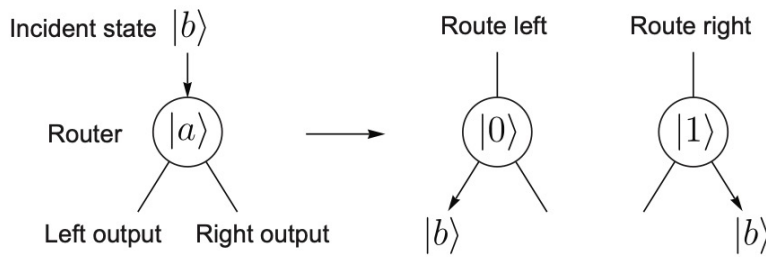
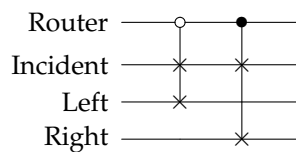


Figure 9.2: The quantum router building block of fanout QRAM.

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 from⁷). 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 $\bigotimes_{i=0}^{2^n-1} Z^{x_i}$ 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. A final Hadamard gate maps $|+\rangle$ and $|-\rangle$ to the computational-basis states $|0\rangle$ and $|1\rangle$, respectively.

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 also allows one to query multiple memory elements in superposition — just as all elements work coherently.

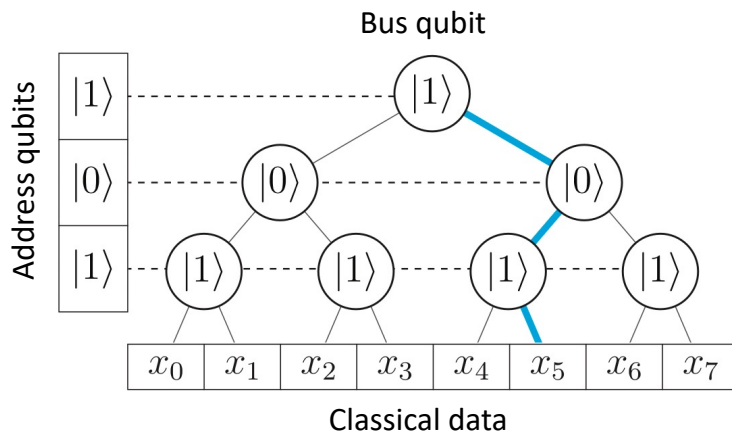
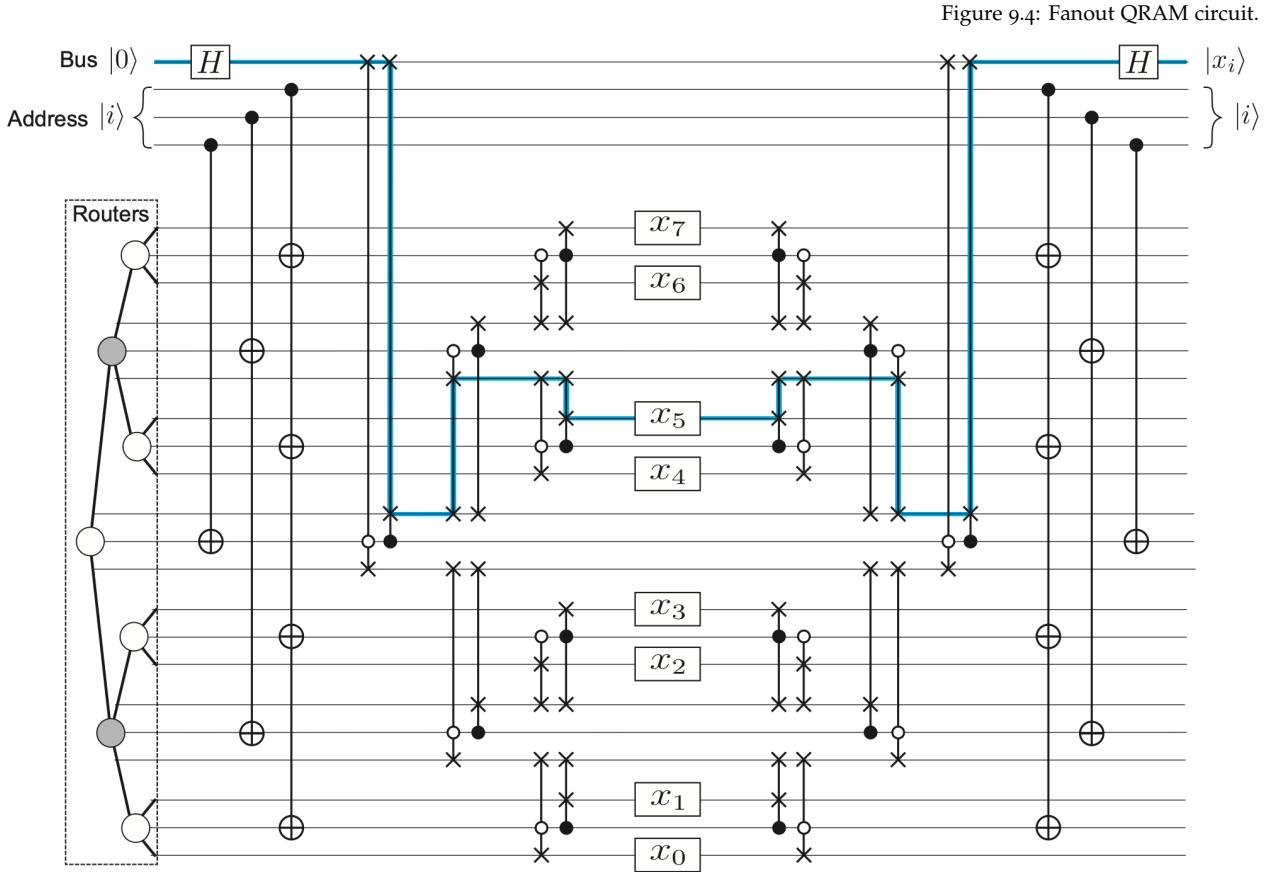


Figure 9.3: Fanout QRAM data structure with $n = 3$ for querying the element x_5 at position 101.



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 impractical, as all $\mathcal{O}(2^n)$ routing 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.

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 $\text{poly}(n)$ algorithmic qubits needed by most algorithms employing QRAM, there is a strong incentive to design more noise-resistant 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.

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.

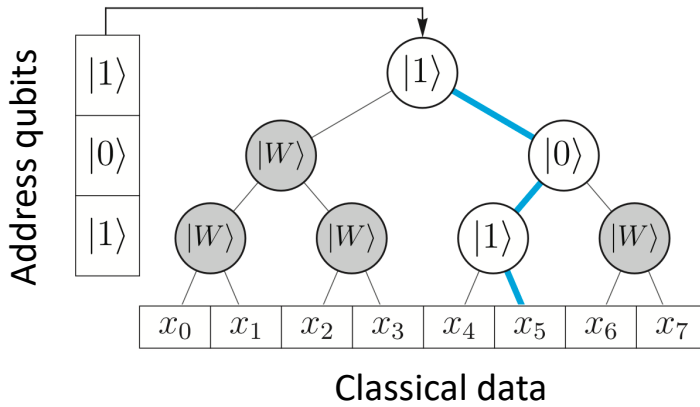
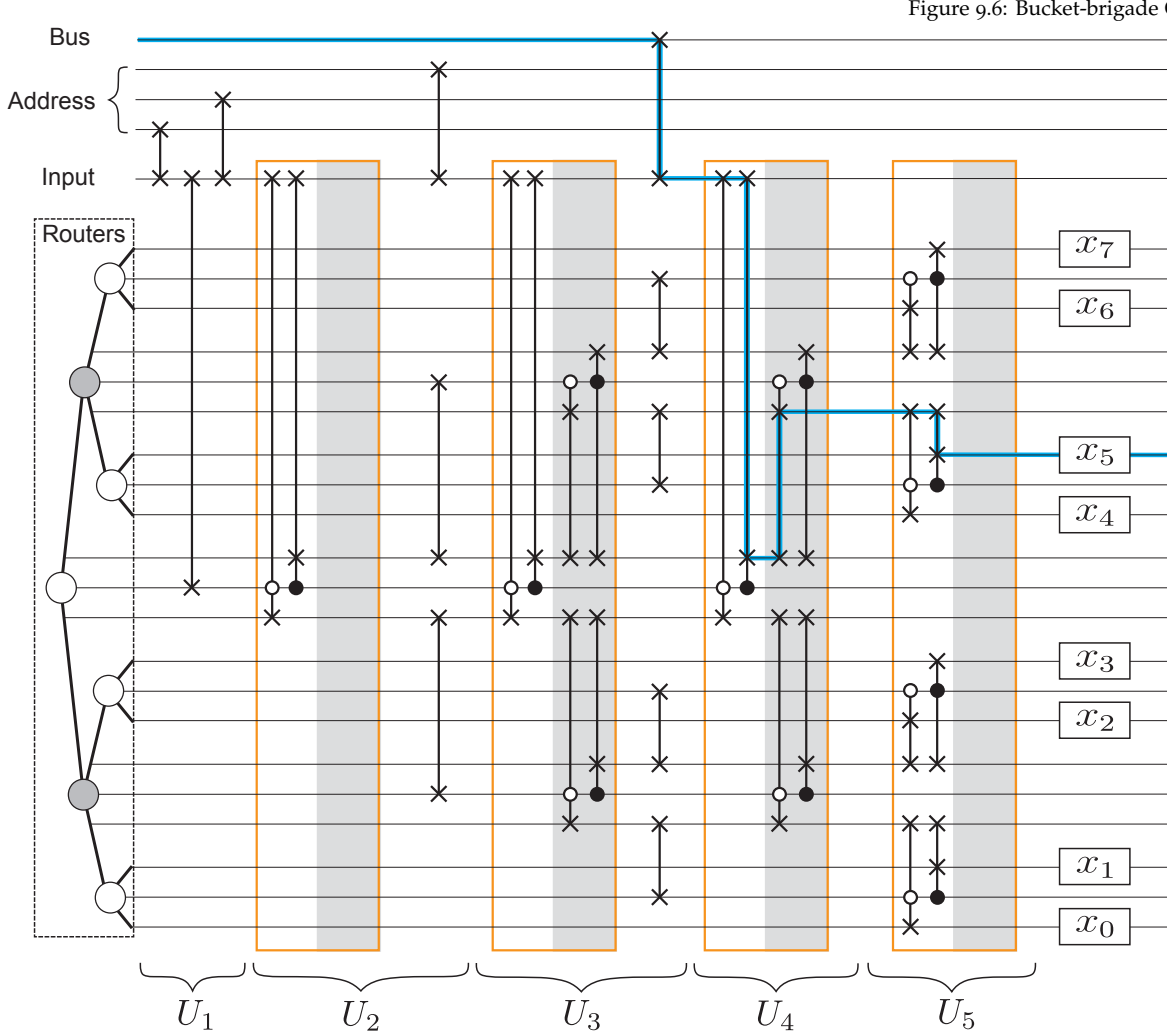


Figure 9.5: Bucket-brigade QRAM data structure with $n = 3$ for querying the element x_5 at position 101.

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 $|0\rangle / |1\rangle$, an address qubit 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 through 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 space–depth trade-offs. 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, 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 employ controlled-swap gates—there are indications that they may have some inherent noise resistance. This could lead to lower quantum error correction

⁹ This is also important in comparison with classical hardware, as, e.g., $\text{poly}(N)$ 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 $\text{poly}(N)$ classical control CPUs for managing the quantum error correction overhead.

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.

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

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) and quantum phase estimation (Chapters 5 and 7) to Hamiltonian simulation (Chapter 6) and linear-system solvers (Chapter 8). We have also seen different ways of loading the relevant input into a quantum computer (Chapter 9). Many matrix-processing primitives used in Hamiltonian simulation and linear-system solvers can be organized within a general framework termed *quantum singular value transformation* (QSVT), which emerged from a longer series of papers.¹ The quantum Fourier transform and phase estimation remain important companion subroutines rather than direct special cases of QSVT.

The starting point is the simple realization that quantum mechanics encodes data into quantum states and processes them with unitary operations, whereas classical data can be represented by general matrices and one may want coherent access to non-unitary transformations. The QSVT framework embeds non-unitary matrices into larger unitary matrices represented by efficient quantum circuits, and then performs polynomial transformations of their singular values. For some tasks, such as 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 with²

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

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).

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

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

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

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

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

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

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

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:

- An *overlap matrix* A with $(A)_{ij} = \langle \psi_i | \phi_j \rangle$ can be $(1, a, 0)$ -block encoded when state-preparation unitaries for two orthonormal families satisfy

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

via $U_A = U_L^\dagger U_R$.

- 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
- A linear combination of unitaries $A = \sum_{l \in L} \alpha_l P_l$ as given in Eq. (6.40) can be $(\|\vec{\alpha}\|_1, \lceil \log_2 |L| \rceil, 0)$ -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

For an unstructured dense $M \times M$ matrix stored in hardware, a low-depth construction may require $\mathcal{O}(M^2)$ memory cells, up to the precision used for each entry (cf. Chapter 9).

10.3 Transformation of block encodings

QSVT is a quantum circuit that takes a block encoding U_A of a matrix A and implements a polynomial transformation of its singular values. For a degree- d polynomial, the cost is roughly $\mathcal{O}(d)$ applications of U_A and $U_A^\dagger$, together with $\mathcal{O}(d)$ further elementary gates and a constant number of additional ancilla qubits. More general

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

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

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

function transformations can be achieved by finding good *polynomial approximations* on the relevant interval.

To make this more precise, recall the *singular value decomposition*. For a complex $M \times M$ matrix A , it is a factorization

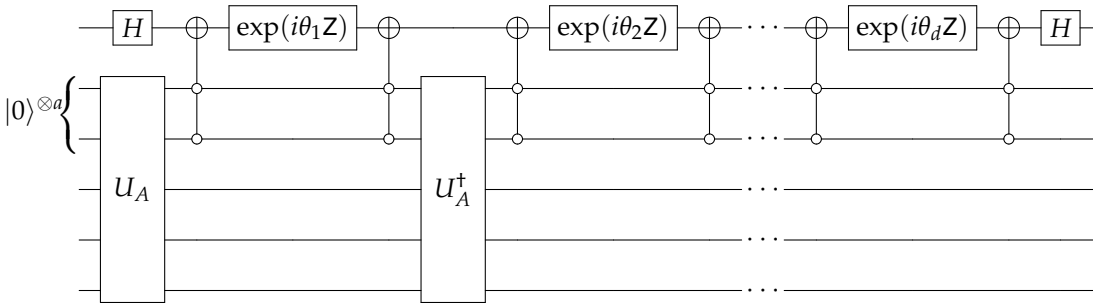
$$A = U\Sigma V^\dagger, \quad (10.5)$$

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 . QSVT replaces these singular values by polynomial values while arranging the left and right singular vectors according to the parity of the polynomial. For a Hermitian matrix, the related *quantum eigenvalue transformation* (QET) acts on the signed eigenvalues and implements the ordinary polynomial matrix function $p(A)$. For simplicity, we focus below on Hermitian matrices A with $M = 2^m$.

Now, for a real, definite parity polynomial

$$p_d : [-1, 1] \rightarrow [-1, 1] \text{ with } \sup_{x \in [-1, 1]} |p_d(x)| \leq 1 \quad (10.6)$$

of degree d , the QSVT circuit takes the exact form



featuring one ancilla qubit, and where the rotation angle set $\{\theta_1, \dots, \theta_d\}$ specific to p_d has to be classically precomputed. This can be done efficiently, both in theory and in practice.⁶ Moreover, general polynomials $p : [-1, 1] \rightarrow [-1, 1]$ 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. arXiv:1806.10236; and Rui Chao, Dawei Ding, Andras Gilyen, Cupjin Huang, and Mario Szegedy. Finding angles for quantum signal processing with machine precision. 2020. arXiv:2003.02831

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

$$p : [-1, 1] \rightarrow \left\{ c \in \mathbb{C} : |c| \leq \frac{1}{4} \right\} \quad (10.7)$$

be a degree- d polynomial. Up to arbitrary gate-synthesis precision, QSVT gives a constant-normalization block encoding of $p(A/\alpha)$ using

- d applications in total of U_A and $U_A^\dagger$
- one controlled application of U_A
- $\mathcal{O}(ad)$ other elementary quantum gates, and
- $a + \mathcal{O}(1)$ ancilla qubits.

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

10.4 Example polynomials

To use 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

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

$$[-1, -\kappa(A)^{-1}] \cup [\kappa(A)^{-1}, 1], \quad (10.8)$$

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, one can construct an odd polynomial of degree $d = \mathcal{O}(\kappa \log(\kappa/\epsilon))$ that approximates a suitably rescaled inverse on this interval.⁹ Turning the resulting block encoding into the normalized solution state also requires accounting for the success probability. Refined constructions, including variable-time amplitude amplification, retain essentially linear dependence on κ up to logarithmic factors, consistent with the state-of-the-art methods in Section 8.5.

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

$$\exp(ix) = \cos(x) + i \sin(x), \quad (10.9)$$

where each term can now be separately approximated in terms of their truncated Jacobi-Anger expansion. This then leads to the

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

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

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

optimal Hamiltonian simulation algorithm (up to a constant) with query complexity to the block encoding U_H of the Hamiltonian H as

$$O \left(\alpha t + \frac{\log(\varepsilon^{-1})}{\log \left(e + \frac{\log(\varepsilon^{-1})}{\alpha t} \right)} \right) \quad (10.10)$$

for simulation time t , where α is the block-encoding normalization. For a Pauli LCU, one may take $\alpha = \sum_{j \in J} |\beta_j|$, which is $\mathcal{O}(|J|)$ when the coefficients are bounded. This is the complexity quoted 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

Post-Quantum Cryptography (D. Unruh), RWTH Computer Science.

Quantum Error Correction (M. Müller), RWTH Physics.

Quantum Information (M. Berta), RWTH Physics.

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.

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.

Madelyn Cain, Qian Xu, Robbie King, Lewis R. B. Picard, Harry Levine, Manuel Endres, John Preskill, Hsin-Yuan Huang, and Dolev Bluvstein. Shor’s algorithm is possible with as few as 10,000 reconfigurable atomic qubits. 2026. arXiv:2603.28627.

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. arXiv:2203.10236.

Rui Chao, Dawei Ding, Andras Gilyen, Cupjin Huang, and Mario Szegedy. Finding angles for quantum signal processing with machine precision. 2020. arXiv:2003.02831.

Chi-Fang Chen, Michael J. Kastoryano, and András Gilyén. An efficient and exact noncommutative quantum Gibbs sampler. 2023. arXiv: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.
- Richard Cleve. The query complexity of order-finding. *Inform. And Comput.*, 192(2):162–171, 2004.
- 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.
- D. Coppersmith. An approximate fourier transform useful in quantum factoring, 1994. IBM Research Report RC 19642, 1994.
- Pedro C.S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babush, 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. arXiv: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. arXiv: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. arXiv: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 Tarasinski, and Barbara M. Terhal. Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments. *New Journal of Physics*, 21:023022, 2019.

OpenAI. Integer multiplication below $n \log n$. OpenAI Math Release preprint, 2026a. OpenAI Math Release OAI:Integer-multiplication-below-n-log-n-September-23-2026, 23 September 2026.

OpenAI. An upper bound of $9/4$ for the matrix multiplication exponent. OpenAI Math Release preprint, 2026b. OpenAI Math Release OAI:Matrix-Multiplication-Nine-Fourths-October-2-2026, 2 October 2026.

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. arXiv:2108.06150.

Christoph Sunderhauf, Earl Campbell, and Joan Camps. Block-encoding structured matrices for data input in quantum computing. 2023. arXiv: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.