

Dissertation
zur Erlangung des Grades des
Doktors der Naturwissenschaften
der Naturwissenschaftlich-Technischen Fakultät der
Universität des Saarlandes

End-to-end Quantum Computation
of the Response Function of Elastic Structures

Saarbrücken, 2025

Sven Danz



UNIVERSITÄT
DES
SAARLANDES

Tag des Kolloquiums: 28.11.2025

Dekan: Prof. Dr.-Ing. Dirk Bähre

Berichterstatter: Prof. Dr. Frank Wilhelm-Mauch
Prof. Dr. Sevag Gharibian
Prof. Dr. Peter P. Orth

Vorsitz: Prof. Dr. Martin Müser

Akad. Mitglied: Dr. Philipp Hövel

Abstract

Quantum computing is at a pivotal stage where hardware advancements are steady, yet most quantum algorithms lack guaranteed advantages in real-world applications. This thesis explores the application of quantum computing to elastic structure problems. In our case, this is related to the broader class of finite element problems.

We propose an end-to-end response function calculator for coupled oscillators based on quantum phase estimation, addressing its typical input-output bottlenecks. Specifically, we achieve eigenstate preparation in a single gate, encode a sparse $N \times N$ matrix using at most $O(\log_2^2 N)$ gates, and minimize the sampling overhead by reducing the output. These innovations result in an overall runtime that is polynomial in N , but outperforms classical alternatives, while reducing memory requirements exponentially compared to classical computation.

The core strategy lies in simplifying the problem complexity and focusing on minimalistic outputs while exploiting the full N -dimensional Hilbert space only during the intermediate quantum steps. This work demonstrates that quantum computing can not only reduce memory demands, but also achieve meaningful speed-ups when real-world problems are reformulated into humanly manageable subproblems.

Zusammenfassung

Quantencomputer entwickeln sich rasant weiter, doch die meisten Quantenalgorithmen verlieren ihren Vorteil bei angewandten Problemen. Diese Arbeit untersucht den Einsatz von Quantencomputern zur Analyse elastischer Strukturen, welcher auf die größere Familie der Finite-Elemente-Probleme übertragbar ist.

Wir präsentieren einen vollständigen Algorithmus zur Berechnung von Antwortfunktionen gekoppelter Oszillatoren, der auf der Quantenphasenschätzung basiert und typische Input-Output-Herausforderungen löst. Konkret umfasst dies die Erzeugung von Eigenzuständen in einem einzigen Gatterschritt, die Implementierung einer dünnbesetzten $N \times N$ -Matrix mit maximal $O(\log_2^2 N)$ Gatterschritten sowie die Reduktion des Messaufwands auf das notwendige Minimum. Dies führt zu einer Laufzeit, die polynomiell in N ist und klassische Alternativen übertrifft, während der Speicherbedarf im Vergleich zu klassischen Algorithmen exponentiell geringer ist.

Der Schlüssel zu diesem Fortschritt liegt in der Vereinfachung der Problemkomplexität und der Fokussierung auf minimale Ausgaben, während der volle N -dimensionale Hilbertraum ausschließlich in den quantenmechanischen Zwischenschritten genutzt wird. Diese Arbeit zeigt, dass Quantencomputer nicht nur den Speicherbedarf reduzieren, sondern auch schwer erreichbare Geschwindigkeitsvorteile erzielen können – vorausgesetzt, das Problem kann in handhabbare Teilprobleme umformuliert werden.

List of Publications

Stefan Schröder et al. “An optimization approach for a milling dynamics simulation based on Quantum Computing”. In: *Procedia CIRP* 121C (2024), pp. 13–18. doi: 10.1016/j.procir.2023.09.223

Preprints

Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405.08694

Stefan Schröder et al. “A methodical approach to evaluate the potential of Quantum Computing for Manufacturing Simulations”. In: *arXiv e-prints* (2024). arXiv: 2408.08730

Sven Danz, Tobias Stollenwerk, and Alessandro Ciani. “Quantum oracles for the finite element method”. In: *arXiv e-prints* (2025). arXiv: 2504.19827

Project reports

Wolfgang Maass et al. “QUASIM: Quantum Computing Enhanced Service Ecosystem for Simulation in Manufacturing”. In: *KI - Künstliche Intelligenz* (2024). issn: 1610-1987. doi: 10.1007/s13218-024-00860-x

Acknowledgements

The following document is a very technical summary of the last three years of my working life. However, besides circuits and equations, I spend a lot of my time with great people. They have not only influenced this work, but also how I have developed and made the whole process a joy. I would like to express my gratitude at this point for all the advice, help and simply happy moments they have given me.

A special thanks goes to Frank Wilhelm-Mauch, a mentor who shines by giving you the freedom you need to grow as an independent young scientist, while always finding time and good advice when you need it. Most of my work has been done in collaboration with Alessandro Ciani as an unofficial supervisor, with whom I have shared countless hours of sometimes louder scientific discussions, that have essentially led to what you find in this thesis. I would like to thank Tobias Stollenwerk for his professional guidance, his open ear for administrative complaints, and our nerding out about software.

In the process of writing this thesis, I relied on our *Writers club* for feedback, technical details, and motivation. A special thanks goes to Jurek Frey, who proofread several chapters. I would also like to thank the entire QUASIM team for making this work possible. Finally, a big thank you to the PGI-12, which is an incredible institute of diverse and great scientists.

Contents

1	Introduction	1
1.1	Related work	1
1.2	Structure	3
1.3	Notation	4
2	Quantum Bits, Gates and Circuits	5
2.1	Quantum systems as memory unit	6
2.2	Quantum operations as logic gates	8
2.3	Quantum Fourier transform	10
2.4	Quantum phase estimation	11
2.5	Matrix encoding	12
2.5.1	Block encoding and pseudo Hamiltonian simulation	13
2.5.2	Compilation	14
2.6	Amplitude amplification	19
3	Elastic Structures	21
3.1	Response of coupled oscillators	22
3.1.1	Unperturbed dynamics	23
3.1.2	Adding an external force	24
3.2	Finite element method for elastic structures	25
3.2.1	Discretization of elastic structures	26
3.2.2	Limitations of the test functions	28
3.2.3	Boundary conditions	29
	Appendices	30
3.A	Analytical form of the response functions	30
4	Quantum Arithmetic	31
4.1	Adders	31
4.2	From adder to polynomial	34
4.2.1	Multiplier	36
4.2.2	Horner's scheme based polynomial	38
4.3	Square root with Newton-Raphson method	39
4.3.1	(Reciprocal) square root	40
	Appendices	43
4.A	Fixed-point exponentiation	43
4.A.1	Exponentiator	43
4.A.2	Operation for signomial functions	45
4.B	In-place arithmetic	47
4.B.1	Square (inSQ)	47

4.B.2	Multiplication (inMUL)	48
4.C	Newton-Raphson reciprocal	49
4.D	Quantum logic operators	50
4.D.1	Comparisons	51
4.D.2	Boolean logical operations	52
5	Quantum Phase Estimation as Eigenpair Problem Solver	55
5.1	The algorithm	56
5.2	Oracle implementation	58
5.2.1	Simplification of stiffness and mass matrix	60
5.2.1.1	Mass lumping	60
5.2.1.2	Stiffness matrix entries in one dimension	62
5.2.2	Oracle implementation based on quantum arithmetic	63
5.2.2.1	Contour of the geometry	64
5.2.2.2	Required arithmetic operations	65
5.3	Non-local response	67
5.4	Data availability	69
	Appendices	70
5.A	Advanced contours	70
6	Complexity Analysis	71
6.1	Memory requirements	73
6.1.1	Phase register size for given eigenvalue tolerances	74
6.1.2	Phase register size for given weight tolerances	75
6.2	Runtime of one iteration	79
6.3	Upper limit for the sample size	79
6.4	Critical assessment	81
6.4.1	Limitations of the algorithm	81
6.4.2	Possible improvements	82
7	Partial Eigenpair Solver	83
7.1	Efficient sub-problem	83
7.2	Eigenvalue based amplitude amplification	84
7.3	Comparison with classical algorithms	86
8	Reduced Eigenvalue Solver on NISQ Hardware	89
8.1	The NISQ-friendly hybrid algorithm	89
8.2	Benchmarking with minimal geometries	90
8.3	Benchmarking results	91
9	Conclusion	93
	Bibliography	95

List of Figures

2.1	Bloch sphere	7
2.2	Circuit: quantum Fourier transform	11
2.3	Circuit: quantum phase estimation	11
2.4	Circuit: State transformation U_T	17
2.5	Circuit: walk operator V	19
2.6	Amplitude amplification	20
3.1	Single blade demonstrator	22
3.2	Coupled oscillators	23
4.1.1	Fixed-point representations	32
4.1.2	Circuit: quantum adder	32
4.1.3	Circuit: bit-shifted quantum adder	35
4.2.1	Circuit: generic quantum multiplier	37
4.2.2	Circuit: Horner's scheme based polynomials	39
4.3.1	Circuit: Newton-Raphson iteration step for the reciprocal square root	41
4.3.2	Circuit: Newton-Raphson method	41
4.A.1	Circuit: quantum exponentiator	43
4.A.2	Circuit: quantum algorithm for signomials	46
4.B.1	Circuit: in-place square	48
4.B.2	Circuit: in-place multiplier	48
4.C.1	Circuit: Newton-Raphson iteration step for the reciprocal	50
4.D.1	Circuit: Kronecker delta	51
4.D.2	Circuit: greater than	52
4.D.3	Circuit: conjunction and disjunction	53
5.1.1	Circuit: core of the local response function calculator	56
5.2.1	Test functions	61
5.2.2	Simplified geometry	62
5.3.1	Circuit: core of the non-local response function calculator	68
7.1.1	Circuit: eigenpair solver	84
7.2.1	Circuit: tester for $\lambda \in [\lambda_l, \lambda_r)$	85
7.3.1	Runtime comparison	87
8.1.1	Circuit: NISQ-friendly eigenvalue solver	90
8.2.1	Simple blade geometries	91
8.2.2	Error scaling of the NISQ-friendly eigenvalue solver	92

List of Tables

2.1	Quantum gate primitives	9
4.1.1	Complexity of quantum adders	33
4.A.1	Complexity of quantum-classical exponentiators	45
6.0.1	Parameters used in the complexity analysis.	72

List of Algorithms

1	Local response function calculation	59
2	Oracle O_g implementation	66

List of Acronyms

FEM	finite element method
NISQ	noisy intermediate scaled quantum
QFT	quantum Fourier transform
QPE	quantum phase estimation
AA	amplitude amplification
CES	complete eigenpair solver
PES	partial eigenpair solver
msb.	most significant bit
IPW	in-process-workpiece

Introduction

It is part of the experience of scientists and engineers that many relevant problems appearing in nature can be formulated as the problem of finding the eigenvalues and eigenvectors of a matrix. Calculating them is tedious and can push even supercomputers to their limits. Consequently, a natural question to ask is whether a new way of computing could push these limits. More specifically, we ask whether quantum algorithms, which exploit phenomena such as state superposition and entanglement, can provide some advantages over standard classical algorithms in solving eigenproblems. The answer to this question is the subject of active research, but it is highly non-trivial and depends on the fine details of the problem itself.

In this thesis, we focus on a particular category of eigenproblems, namely those associated with a system of N coupled harmonic oscillators. Needless to say, systems of coupled harmonic oscillators are ubiquitous in nature, as they can be used to effectively model the low energy dynamics of very different physical systems, from electrical circuits to the vibration of molecules. In essence, our goal is to perform a modal analysis of the system, focusing on the calculation of response functions in frequency (or Laplace) domain. Response functions describe how much an external perturbation applied to a certain oscillator, influences the dynamics of another oscillator in the system. These functions have an analytical functional form that is related to an eigenproblem defined on the system of coupled oscillators. We thus focus on the question of how these response functions can be extracted using a quantum algorithm and in which cases we can expect significant quantum speedups.

1.1 Related work

The idea of using a quantum computer for eigenproblems dates back to the early days of quantum algorithm development [6]. In the quantum chemistry setting, where one is typically interested in characterizing the low-lying eigenvalues and eigenstates of a molecular Hamiltonian, it is in fact considered one of the most promising applications of quantum computers [7–9].

Parts of this chapter have been published in the preprint “Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405.08694”. Other parts has been shared in the preprint “Sven Danz, Tobias Stollenwerk, and Alessandro Ciani. “Quantum oracles for the finite element method”. In: *arXiv e-prints* (2025). arXiv: 2504.19827”.

Broadly speaking, one can distinguish between two categories of quantum algorithms for eigenproblems, namely algorithms based on the variational quantum eigensolver [10, 11], which lack rigorous scaling guarantees, and those based on the quantum phase estimation (QPE) subroutine [12–14]. In this work, we focus on the latter, and in particular on a formulation of standard QPE for eigenvalue estimation that relies on the block-encoding of the matrix of interest in a (larger) unitary [9, 15–21]. A key feature of the problem that we exploit is to estimate the response functions, we do not need to invoke a state preparation subroutine that prepares a desired eigenvector as a quantum state, for instance via controlled rotations [22], reject sampling method [23], eigenvalue transform [24] or adiabatic state preparation (see discussion in Ref. [9]). These kinds of state preparation subroutines hide additional complexity and can ruin potential quantum speedups when properly taken into account. Instead, as we will show, in our case the omnipresent state preparation problem in QPE reduces to the preparation of a product state, from which we need to determine the relevant eigenvalues and associated weights via importance sampling. The simultaneous estimation of multiple eigenvalues using either QPE or other quantum subroutines has been studied in the literature using different approaches [25–28].

The block-encoding, that we suggest, relies on matrix based oracles¹. The use of oracles in quantum computing can be traced back to the Deutsch-Jozsa algorithm [29, 30], in which the oracle computes a potentially balanced function. Grover’s algorithm [31] for database search also assumes access to an oracle that marks computational base states of interest. In fact, Grover’s algorithm achieves a quadratic quantum advantage in the query sense compared to classical algorithms. In recent years, several quantum algorithms have been re-formulated in the unified framework of quantum signal processing and quantum singular-value transformation [9, 15, 17, 18, 32]. The fundamental primer of this framework is the concept of block-encoding of a generic matrix H into a block of a unitary U . One way to construct such block-encoding requires the availability of matrix oracles that encode information about the position and the value of the non-zero matrix elements (see Refs. [9, 15, 17, 33, 34] for examples). This constitutes the so-called sparse-access model to the matrix information, which assumes that the matrix has bounded sparsity s and each matrix element can be computed efficiently with a reversible quantum circuit. We show that this is the case for finite element method (FEM) based oscillation problems.

The works that are most related to ours on resolving coupled harmonic oscillators on a quantum computers are Ref. [35] and Ref. [36]. Ref. [35] analyzes the problem of obtaining the classical time-evolution of a system of coupled harmonic oscillators. To do this, the authors employ a particular encoding of the positions and momenta of the oscillators in the amplitudes of an n -qubit quantum system. On this system, a Hamiltonian can be defined that mimics the time-evolution of the system of coupled oscillators at any time. With this approach, while one cannot efficiently access the full solution, it is possible to estimate efficiently relevant quantities such as the kinetic energy of a subset of oscillators. An interesting open question is whether response functions can also be estimated using the time-evolution approach of Ref. [35].

Ref. [36] instead focuses on a normal mode analysis of systems of coupled oscillators, in the

¹Oracles are black-box quantum operations that are assumed to be implementable without proof.

same spirit as our work, and essentially using the same kind of embedding of the problem in quantum systems. However, Ref. [36] does not discuss the application of estimating response functions and assumes that a good approximation of the eigenvectors can be prepared in a quantum system. While this would also allow estimating response functions in principle, we show that it is not necessary.

Finally, our work is also connected to quantum approaches to solving the wave equation [37] or more generally for finite element simulations of solid structures [38, 39]. In fact, as we detail in Chapter 3, once discretized, these problems can be mapped to a system of coupled harmonic oscillators.

1.2 Structure

The remaining thesis is structured as follows. In Chapter 2, we give a review of the basics of quantum computing. Starting by the introduction of quantum systems as memory unit and quantum operations as logic gates. The last sections of this chapter explain more advanced but established quantum algorithms, namely the Fourier transform, phase estimation, block encoding, and amplitude amplification, which are all necessary for the response function calculator presented in this work. In Chapter 3, we give a review of the theory required for the analysis of elastic structures. We introduce the concept of linear response in coupled oscillators, a suitable model for oscillations of all kinds, and show that its calculation can be reduced to solving an eigenproblem. Further, we introduce the finite element method (FEM), that enables us to map elastic structures onto coupled oscillators. Before we go into the main algorithm, we review the existing literature about quantum arithmetic in Chapter 4. Furthermore, we extend it by a detailed description of how to compute polynomial, signomial, and square root (and reciprocal square root) functions based on quantum adders. All those operations are necessary for the implementation of oracles in the main algorithm. This chapter contains a complete discussion of the corresponding computational complexities. In Chapter 5, the core of this thesis, we propose a QPE based algorithm for the calculation of eigenpairs, that allow to find the local response function of coupled oscillators. QPE, typically, struggles with bottleneck issues for the encoding of the initial state and the matrix. We describe how to mitigate them using a trivial state preparation and an oracle based matrix block encoding. Further, we describe in detailed the implementation of those oracles based on FEM, necessary simplifications, and their computational complexity, showing that their implementation is no additional bottleneck. As last part of this chapter, we extend the eigenpair solver for non-local response. The corresponding complexity analysis is gathered in Chapter 6 and contains the memory requirements, the worst-case runtime of one iteration of the proposed quantum algorithm, and an upper limit for the total sample size necessary for the eigenpair extraction. We finish this chapter with a critical assessment of the limitations and possible improvements of our quantum approach. In Chapter 7 we propose the addition of amplitude amplification limiting the outcome to a small subset of eigenpairs. We call this a partial eigenpair solver (PES). This chapter ends with a comparison with classical algorithms, that focus on small subsets as well. Up to this point, the majority of the algorithms presented are only suitable for fault-

tolerant quantum computers. This is why, in Chapter 8, we discuss a reduced hybrid eigenvalue solver that runs on noisy intermediate scaled quantum (NISQ) hardware. Due to its runtime intensive classical component, it will never surpass state-of-the-art classical eigenvalue solvers. However, we could show on existing quantum hardware that it is already possible to estimate the eigenvalues of small but industry-related problems. We conclude by summarizing our results and describe possible next steps in Chapter 9.

1.3 Notation

In this work, $i^2 = -1$ denotes the imaginary unit. Further, we define the natural numbers $\mathbb{N} = \{1, 2, \dots, \infty\}$, the integer numbers $\mathbb{Z} = \{-\infty, \dots, 1, 0\} \cup \mathbb{N}$, the real numbers \mathbb{R} , the complex numbers \mathbb{C} , $\mathbb{Z}_n = \{0, \dots, n-1\}$ and $[n] = \{1, \dots, n\}$. We use I_n for the $n \times n$ identity matrix. Empty blocks in matrices represent zero blocks. U^\dagger is the conjugate transpose of the matrix U and equals its inverse $U^\dagger = U^{-1}$ for unitaries. We use \otimes for the Kronecker product. The Heaviside function is defined by $\Theta(x > 0) = 1$, $\Theta(x \leq 0) = 0$, and we introduce the binary sign defined by $\text{sgn}_b(x) = \neg\Theta(x)$. We further use the term signomial functions for $\text{sig}(b) = \sum_{k=0}^K c_k b^{a_k}$ where the coefficients c_k and the exponents a_k are real numbers. In this work, we are only interested in rational $a_k = k/\chi$, where $\chi \in \mathbb{N}$ is a constant. We use the term eigenpair slightly different from its usual definition, calling the set of one eigenvalue λ_j and the absolute product of two, not necessarily different, entries W_{uj} of the corresponding eigenvector $\{\lambda_j, |W_{uj}W_{vj}|\}$ eigenpair.

Quantum Bits, Gates and Circuits

The interest in computers governed by the laws of quantum mechanics started in the late 1960s [40]. Back then it was mainly motivated by the interest in communication [41], basics of quantum information theory [42, 43] in the 1970s, and in quantum simulation [44] in the 1980s. The search for quantum algorithms surpassing the performance of classical counterparts succeeded in the 1990s. David Deutsch and Richard Josza proposed 1992 the first problem that can be solved on a quantum computer faster than on any classical machine [30]. Finding a use-case for their algorithm is still an open question. Two years later, 1994, Peter Shor proposed the first algorithm of wide interest. His algorithm, if implemented would endanger many cryptographic systems by solving the factorization problem of large integers [45, 46]. Another two years later, Grover described a quantum database search algorithm with quadratic speedup [31].

None of these algorithms are yet in use because, firstly, they require the appropriate quantum hardware and, secondly, they depend on many conditions that are not always met. Analogous to classical computing, a quantum device has to satisfy two requirements. It demands a memory unit that stores a state long enough to execute computations on it, and it requires control processes that allow you to initialize, transform and measure states in that memory unit. The pursuit of creating a machine that meets these requirements is an ongoing process, with progress being made every year. This thesis aims to contribute to the second, non-hardware related, of those two challenges. We study the implementation of a promising quantum algorithm for a specific use case with focus on the mitigation of potential bottleneck conditions.

This chapter shall give an overview of already known basics of quantum computing necessary to understand the remainder of this thesis. It starts with a description of the most fundamental computational unit of every quantum computer, the qubit, in Section 2.1. It follows the introduction of computational operations tailored for quantum systems and necessary for the algorithms in Section 2.2. Further, a diagrammatic technique, used in quantum computing to represent complex algorithms, is introduced there. After this, advanced quantum algorithms are described starting with the quantum equivalent of the discrete Fourier transform in Section 2.3. Based on this, the QPE is discussed in Section 2.4. This routine may appear as made for artificial problems at first, but it is the core of our main algorithm. An efficient method to encode a sparse Hermitian matrix into a unitary quantum operations based partially on block encoding

Section 2.5 of this chapter has been published in the preprint “Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405.08694”.

is described in Section 2.5. Efficient matrix encoding is crucial for the use of QPE in this manuscript. At last, in Section 2.6, an overview of amplitude amplification (AA) is given, which is a generalization of Grover's data search algorithm.

Of course, we can only give a limited introduction to the now vast world of quantum computing. For a more detailed description of quantum computing, information and algorithms, we recommend starting with Refs. [12, 47].

2.1 Quantum systems as memory unit

The first of the two requirements for every computational system is a memory unit. Breaking this down to its smallest component, we start with the quantum version of a bit, the *qubit*. Analog to classical computing, we are looking for systems with distinct states to which we can allocate values. Here, we are satisfied with two-state systems, even though multi-state systems would be possible. The name already tells, that in contrast to classical computing, we are looking for physical systems that are described by the laws of quantum mechanics. We find this in the polarization of photons, the spin states of electrons or ions, or the flux of superconducting circuits to name only a few examples. The standard notation to describe the states of those quantum systems is the Dirac or Braket notation, in which we write $|0\rangle = (1, 0)^T$ and $|1\rangle = (0, 1)^T$ for our two distinct states. The important contrast to classical systems is, that quantum systems are not limited to be in either of those state but can be in any superposition of them

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad (2.1)$$

where α and β are complex numbers that have to satisfy normalization (i.e. $|\alpha|^2 + |\beta|^2 = 1$). This increases the number of possible states to infinite states living in the Hilbert space. To name only a few, that form common bases next to $\{|0\rangle, |1\rangle\}$, we introduce

$$|\pm\rangle = \frac{1}{\sqrt{2}} (|0\rangle \pm |1\rangle), \quad (2.2a)$$

$$|\pm i\rangle = \frac{1}{\sqrt{2}} (|0\rangle \pm i |1\rangle). \quad (2.2b)$$

An alternative representation of Eq. (2.1) is

$$|\Psi\rangle = \cos \frac{\vartheta}{2} |0\rangle + e^{i\varphi} \sin \frac{\vartheta}{2} |1\rangle, \quad (2.3)$$

which is normalized by default. The two parameters ϑ and φ can be interpreted as angles in a sphere with every state living on its surface (see Fig. 2.1). We call this the Bloch sphere.

The superposition of different basis states is one of the key differences to classical bits that enables a completely new family of algorithms. It is possible to compute two different calculations in parallel on only one qubit. However, it is tricky extracting the results as the superposition collapses if measured. A measurement can be seen as a projection onto one of the two basis

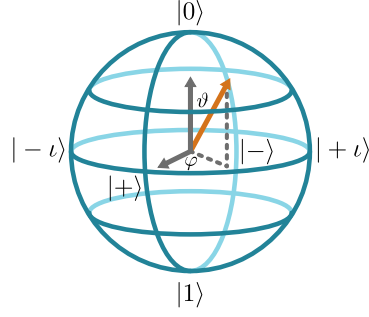


Figure 2.1: Bloch sphere: graphical representation of a two-level quantum state. It is fully described by two parameters (ϑ, φ) .

states. The basis here depends on the way we measure. For example if we measure in the so-called *Z*- or *computational basis*, the system collapses into either $|0\rangle$ or $|1\rangle$. The probability P to end on state $|i\rangle \in \{|0\rangle, |1\rangle\}$ is given by

$$P(i) = \langle \Psi | i \rangle \langle i | \Psi \rangle. \quad (2.4)$$

Analog, we can measure in the *X*- and *Y*-basis for $|\pm\rangle$ and $|\pm i\rangle$ respectively.

The system can be scaled up by introducing more than one qubit forming larger states

$$|\Phi\rangle = |\Psi_a\rangle \otimes |\Psi_b\rangle = \sum_{i,j=0}^1 a_{ij} |i\rangle \otimes |j\rangle. \quad (2.5)$$

Here, \otimes is the Kronecker product, but it is common to write multi state systems without it or even combine them into one $|\Psi_a\rangle \otimes |\Psi_b\rangle = |\Psi_a\rangle |\Psi_b\rangle = |\Psi_a, \Psi_b\rangle$. The new coefficients a_{ij} are products of the coefficients in $|\Psi_a\rangle$ and $|\Psi_b\rangle$. Multi-qubit states can also be measured leading to a collapse into one of the measurement dependent basis states just like for a single qubit. However, this gets more interesting for entangled states like the Bell states¹

$$|\Psi_{\text{Bell}}\rangle = \frac{1}{\sqrt{2}} (|0, 0\rangle + |1, 1\rangle). \quad (2.6)$$

If we measure only the first qubit in the *Z*-basis, the state will collapse into either $|0, 0\rangle$ or $|1, 1\rangle$ telling us the state of the second qubit without measurement. This is the second property of quantum systems allowing for interesting algorithms. Measuring one qubit always effects the whole network of qubits that are entangled with it. One has to see entangled qubits as one quantum system that can not be treated partially (i.e. $|\Psi_{\text{Bell}}\rangle \neq |\Psi_a\rangle \otimes |\Psi_b\rangle$).

In the circumstance of this work we will always assume access to qubits that maintain their information over the runtime of the full computation. However, this is not usually given for deep algorithms running on state-of-the-art NISQ devices and is therefore subject to current research. Only time will tell if we will achieve this in the future.

¹There are four different Bell states. We will focus only on one for simplicity.

2.2 Quantum operations as logic gates

Now that we introduced qubits as our smallest memory unit we need a way to control them during the computation. For this, we will study the time evolution of any quantum state that is described by Schrödinger's equation²

$$i\partial_t |\Psi(t)\rangle = H(t) |\Psi(t)\rangle, \quad (2.7)$$

where ∂_t is the partial derivative with respect to the time t and $H(t)$ is the time-dependent Hamiltonian describing the evolution of the qubit. The solution of the latter defines the time evolution operator $U(t)$

$$|\Psi(t)\rangle = \mathcal{T} e^{-i \int H(t) dt} |\Psi(0)\rangle \equiv U(t) |\Psi(0)\rangle, \quad (2.8)$$

where \mathcal{T} is the time-ordering operator necessary if $H(t_1)H(t_2) \neq H(t_2)H(t_1)$ for two arbitrary time stamps $t_1 \neq t_2$. By controlling $H(t)$ with meticulously selected perturbations, one can achieve arbitrary rotations (e.g. $U = e^{i\frac{\vartheta}{2}\sigma_z} = R_z(\vartheta)$,³) on the Bloch sphere. However, those transformations have to be unitary (i.e. $U^\dagger = U^{-1}$) due to the Hermitian nature of Hamiltonians. This further means that all transformations on a qubit are reversible. Only measurement are of non-reversible character.

A serial combination of just three orthogonal rotations (e.g. $R_y(\vartheta_0)R_z(\varphi)R_y(\vartheta_1)$) can achieve any rotation on the Bloch sphere. A few of those rotations are very common, which is why we gathered them in Table 2.1. We will refer to those rotations as gates from now on. Almost all of them are used in this manuscript. However, at this point we will describe only a few of them in detail starting with the Hadamard gate \bar{H} . This rotation is crucial for the construction of equal distributed superpositions

$$\bar{H} |0\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) = |+\rangle, \quad (2.9a)$$

$$\bar{H} |+\rangle = \frac{1}{\sqrt{2}} (|+\rangle + |-\rangle) = |0\rangle, \quad (2.9b)$$

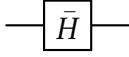
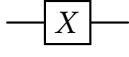
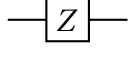
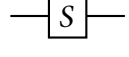
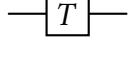
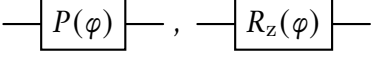
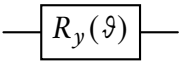
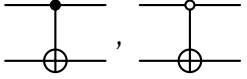
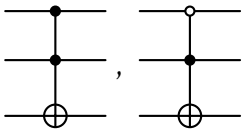

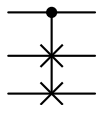
and therefore appears in almost all quantum algorithms. Multiple Hadamard gates applied in parallel to different qubits are called Walsh-Hadamard gate and are denoted with $\bar{H}^{\otimes n}$ for n qubits. The NOT gate simply flips the state in the Z -basis (i.e. $|0\rangle \leftrightarrow |1\rangle$). The Z -gate is doing the equivalent in the X -basis. It gets more interesting with the CNOT gate which is our first two-qubit gate. This gate reads the state of one qubit and applies a NOT gate to a second qubit only if the first is in the state $|1\rangle$. An alternative option that triggers only for $|0\rangle$ is also possible. This is usually called *controlled* and *0-controlled* respectively. This gate is crucial for entangling two qubits and therefore also part of almost every quantum algorithm.

Long algorithms acting on many qubits can become confusing fast. This is why the circuit notation is common (see Fig. 2.2 and Fig. 2.3). It represents qubits propagating in time as wires

²We use the $\hbar = 1$ convention here.

³Here, σ_z is the third Pauli-matrix. The rotations in x - and y -direction can be constructed analog.

Table 2.1: Elementary quantum gates. Empty blocks represent zero blocks and I_n is the $n \times n$ identity matrix. The shown gate symbols and matrices for the multi-qubit gates are not complete.

Name	Matrix	Gate symbol
Hadamard	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	
NOT	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	
Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	
T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$	
Phase	$\begin{pmatrix} e^{-i\varphi/2} & 0 \\ 0 & e^{i\varphi/2} \end{pmatrix}$	
y-Rotation	$\begin{pmatrix} \cos \vartheta/2 & \sin \vartheta/2 \\ -\sin \vartheta/2 & \cos \vartheta/2 \end{pmatrix}$	
CNOT	$\begin{pmatrix} I_2 & \\ & X \end{pmatrix}, \begin{pmatrix} X & \\ & I_2 \end{pmatrix}$	
Toffoli	$\begin{pmatrix} I_4 & \\ & \text{CNOT} \end{pmatrix}, \begin{pmatrix} \text{CNOT} & \\ & I_4 \end{pmatrix}$	
SWAP	$\begin{pmatrix} 1 & & \\ & X & \\ & & 1 \end{pmatrix}$	
CSWAP	$\begin{pmatrix} I_4 & \\ & \text{SWAP} \end{pmatrix}$	

from left to right starting in the state written to the left of it. Bundles of qubits called *registers* can be represented with three wires. The rotations acting on the qubits are illustrated as logic gates (boxes) sitting on the wire. This is why we refer to them as gates. Multi-qubit gates are represented either as larger boxes or as single-qubit gates connected with vertical wires to a second qubit. The latter illustrates a controlled gate (cf. CNOT gate). Most circuits end with one or more measurements illustrated as boxes with a scale. They measure in the Z -basis if not otherwise declared.

In the next sections we will combine multiple qubits and quantum gates into full algorithms, starting with some basic routines crucial for the work presented in this manuscript.

2.3 Quantum Fourier transform

It may sound random at first, but one of the key routines used in many algorithms is the quantum Fourier transform (QFT). This is the quantum analog of the discrete Fourier transform

$$X_x = \sum_{y=0}^{N-1} e^{\frac{2\pi i}{N} x y} y. \quad (2.10)$$

The quantum version is a routine or operation QFT applied to an n -qubit state $|x\rangle$

$$|X_x\rangle = \text{QFT} |x\rangle = \sum_{y=0}^{N-1} e^{\frac{2\pi i}{N} x y} |y\rangle. \quad (2.11)$$

Here, $N = 2^n$ is the size of the n -qubit Hilbert space. We further introduce a new notation for multi-qubit states $|x\rangle = |x_n, \dots, x_1\rangle$, with $x = \sum_{i=1}^n 2^{i-1} x_i$ and $x_i \in \{0, 1\} \equiv \mathbb{Z}_2$.

The QFT is taking one state and transforms it into a superposition of other states with specific coefficients. While doing this, it reads the information x stored in the basis of the original state and moves it into the phases of the complex coefficients in the final state. In contrast to classical computations, the inverse of this allows computations to be performed in the phase before the results are measured in the basis. This is extending the possibilities of quantum computing further. The most famous algorithms that is using this is the quantum phase estimation (QPE) described in Section 2.4.

The precise compilation of the QFT for a 4-qubit version is shown in Fig. 2.2. We recommend studying it backwards (i.e. from right to left which equals QPE[†]). The SWAP gates are not mandatory. Their sole purpose is to contain the qubit order of significance. Then we start with a Hadamard gate decoding the least significant digit x_1 . Now that we have access to x_1 , we can remove its contribution to the phases with a special phase gates $P_k = P(2\pi 2^{-k})$. This allows us to decode x_2 with a Hadamard gate. We continue until all digits are decoded. For a proper full-detail description of QFT, we recommend reading Refs. [12, 47].

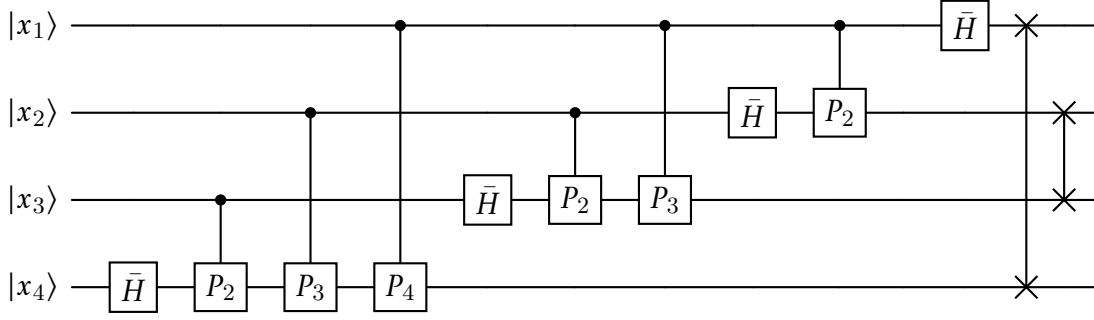


Figure 2.2: Circuit for the 4-qubit version of the QFT. The qubits are ordered by significance with the most significant at the bottom. The phase gate here denotes $P_k = P(2\pi 2^{-k})$.

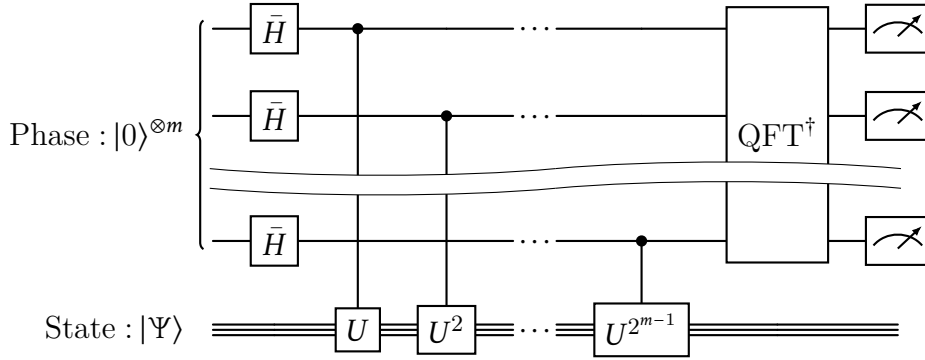


Figure 2.3: The circuit for the QPE can be split into two registers. The upper one for the phase that is measured at the end with high probability and the lower for the state $|\Psi\rangle$ on which the unitary U is applied. This is also the unitary for which the phase is estimated. The last step before the measurement is the inverse quantum Fourier transform (QFT^\dagger) (cf. Section 2.3 or [12, 47]).

2.4 Quantum phase estimation

The most important quantum routine for the work presented in this manuscript is the quantum phase estimation (QPE) [48]. It is one of the pioneering quantum primitives and used in other prominent algorithms such as Shor's algorithm for prime factorization [46] and the quantum linear equation solver by Harrow, Hassidim and Lloyd [49]. As the name implies, it can be used to estimate the effective phase φ a quantum operation U inflicts on one of its eigenstates $|\Psi\rangle$

$$U |\Psi\rangle = e^{i\varphi} |\Psi\rangle. \quad (2.12)$$

Quantum systems allow computing φ by compiling Eq. (2.12) in a circuit. The remaining problem is to extract the phase. As already mentioned in Section 2.3 QFT can be used to transform information stored in the basis of a quantum state into its phases and vice-versa, which is shown in Fig. 2.3. We start by preparing a superposition of all possible states in the Z -basis with Hadamard gates \bar{H} in a secondary register. After this, a series of controlled

versions of U prepare $e^{ik\varphi}$ for every basis state $|k\rangle$ in the secondary register. It remains to decode the series of phases with the inverse of the QFT yielding a binary representation of the phase in the secondary register that can be measured. Again, we recommend reading Refs. [12, 47] for a more detailed discussion of the QPE.

While being efficient for the described use case, this routine suffers from severe bottleneck issues when used for broader application. In the form described, it requires that the eigenstate of U has to be encoded on our quantum device. This usually requires knowledge of the eigenstate combined with an efficient method of state preparation. Both requirements are generally not satisfied. Further, for the application of U we assumed that we have access to it. In other applications we may need to compile U at first, which can get expensive for larger operations. At last, we need to mention the probabilistic nature of QPE. It is only exact if a precise binary representation of φ is possible. If not, the probability to measure a false result decreases with distance from φ but stays finite. This is why it is called an estimation rather than a computation.

A common application of QPE is to use it as eigenvalue solver (cf. [49]). Given a Matrix A with one of its eigenvectors ν , one can compute the corresponding eigenvalue λ with the right mapping. The eigenvector entries need to be encoded in the amplitudes of a quantum state $|\nu\rangle$. This is commonly called *amplitude encoding*. The matrix A requires a compilation in the form $U_A = e^{iA}$ to enter the quantum device. Here, A has to be Hermitian⁴. Now, QPE, with $|\nu\rangle$ as state and U_A as operation, estimates λ .

2.5 Matrix encoding

In order to use QPE as an eigenvalue solver for arbitrary matrices H we require a reliable method encoding the matrix into a time evolution of the form e^{iH} , often referred to as Hamiltonian simulation. Here, we describe a workaround that is inspired by quantum walks and origins from Refs. [33, 34]. Our description is slightly different from the original version as we reordered the subroutines. We start in Section 2.5.1 by introducing the concept of block encoding[9, 15, 17–19, 50] followed by a reflection that turns it into a pseudo Hamiltonian simulation⁵. This contains the same routines as described in the original version but makes the separation visible. Furthermore, that order has the advantage that the new quantum step operator can be applied to states of the original eigenbasis of H . In Section 2.5.2, we describe the compilation of the block encoding based on two oracles. This is also described in [33, 34], but we filled in gaps and added circuit illustrations of the used subroutines for a better understanding.

⁴A non-Hermitian A can be substituted by $|0\rangle\langle 1| \otimes A + |1\rangle\langle 0| \otimes A^\dagger$ by adding one qubit. This is always Hermitian.

One should keep in mind that this changes the eigenstate mapping due to the increase of the Hilbert space.

⁵Pseudo because its eigenvalues are $e^{\pm i \arccos(\lambda)}$, where $|\lambda| < 1$, instead of $e^{i\lambda}$.

2.5.1 Block encoding and pseudo Hamiltonian simulation

Let us assume that we have a unitary U_H acting on $a + n$ qubits that realizes an $(\alpha, a, 0)$ -block-encoding of our n -qubit Hamiltonian H . Mathematically, this translates into the requirement

$$\langle 0|^{\otimes a} U_H |0\rangle^{\otimes a} = \alpha H, \quad (2.13)$$

where a is the number of additional ancilla qubits used for the block encoding. The eigenstates of H with eigenvalue λ will be denoted by $|\lambda\rangle$ in bracket notation. Without loss of generality, we assume that U_H is Hermitian, i.e., $U_H = U_H^\dagger$, which implies $U_H^2 = I_{a+n}$.⁶ We define the operator Π as the projector onto the subspace with all ancillas in the zero state

$$\Pi = |0\rangle\langle 0|^{\otimes a} \otimes I_n. \quad (2.14)$$

The action of a Hermitian block encoding U_H on the state $|0\rangle^{\otimes a} |\lambda\rangle$ can be written as (see Section 10.4 in Ref. [9])

$$U_H |0\rangle^{\otimes a} |\lambda\rangle = \alpha \lambda |0\rangle^{\otimes a} |\lambda\rangle + \sqrt{1 - \alpha^2 \lambda^2} |0\lambda^\perp\rangle, \quad (2.15)$$

where $|0\lambda^\perp\rangle$ is a state such that $(\langle 0|^{\otimes a} \langle \psi|) |0\lambda^\perp\rangle = 0 \forall n$ -qubit state $|\psi\rangle$ and $\langle 0\lambda'^\perp | 0\lambda^\perp\rangle = \delta_{\lambda\lambda'}$. Additionally,

$$U_H |0\lambda^\perp\rangle = \sqrt{1 - \alpha^2 \lambda^2} |0\rangle^{\otimes a} |\lambda\rangle - \alpha \lambda |0\lambda^\perp\rangle. \quad (2.16)$$

Eq. (2.15) and Eq. (2.16) imply that $\forall \lambda$ we can identify a qubit-like subspace

$$\mathcal{H}^{(\lambda)} = \text{span}\{|0\rangle^{\otimes a} |\lambda\rangle, |0\lambda^\perp\rangle\}, \quad (2.17)$$

so that the action of U_H onto this subspace can be represented by a 2×2 matrix $U_H^{(\lambda)}$ given by

$$U_H^{(\lambda)} = \begin{pmatrix} \alpha \lambda & \sqrt{1 - \alpha^2 \lambda^2} \\ \sqrt{1 - \alpha^2 \lambda^2} & -\alpha \lambda \end{pmatrix}. \quad (2.18)$$

However, the eigenvalues of this matrix are ± 1 , since U_H is unitary and Hermitian and thus, they do not carry information about the eigenvalues λ of H . To remedy this issue we define the *walk* operator V , that we use instead of U in the QPE, as

$$V = U_H(2\Pi - I_{a+n}). \quad (2.19)$$

In fact, the operator $2\Pi - I_{a+n}$ also preserves the subspaces $\mathcal{H}^{(\lambda)}$ and its action on $\mathcal{H}^{(\lambda)}$ is simply represented by a Pauli Z matrix

$$(2\Pi - I_{a+n})^{(\lambda)} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (2.20)$$

⁶If U_H is not Hermitian, one can add one ancilla qubit and define the Hermitian unitary $U_H^{(\text{new})} = |+\rangle\langle +| \otimes U_H + |-\rangle\langle +| \otimes U_H^\dagger$, which is a $(\alpha, a + 1, 0)$ -block-encoding of H .

Consequently, the action of V on $\mathcal{H}^{(\lambda)}$ is given by

$$V^{(\lambda)} = U_H^{(\lambda)} (2\Pi - I_{a+n})^{(\lambda)} = \begin{pmatrix} \alpha\lambda & -\sqrt{1 - \alpha^2\lambda^2} \\ \sqrt{1 - \alpha^2\lambda^2} & \alpha\lambda \end{pmatrix}. \quad (2.21)$$

The matrix $V^{(\lambda)}$ has eigenvalues

$$\mu_{\pm} = \alpha\lambda \pm i\sqrt{1 - \alpha^2\lambda^2} = e^{\pm i \arccos(\alpha\lambda)}, \quad (2.22)$$

with corresponding eigenvectors

$$|\mu_{\pm}\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes a} |\lambda\rangle \mp i|0\lambda^{\perp}\rangle). \quad (2.23)$$

Thus, if we start from a state $|\psi\rangle = \sum_{\lambda} c_{\lambda} |0\rangle^{\otimes a} |\lambda\rangle$ and perform *ideal* phase estimation using the operator V , the system will collapse in either $|\mu_{+}\rangle$ or $|\mu_{-}\rangle$ and the value $+\arccos(\alpha\lambda)$ or $-\arccos(\alpha\lambda)$, respectively, can be read out in the phase register. In either case, a subsequent measurement of the ancilla register in the computational basis would give all zeros with probability $1/2$ and thus, prepare the eigenstate $|0\rangle^{\otimes a} |\lambda\rangle$.

2.5.2 Compilation

For the compilation of the pseudo Hamiltonian simulation, we assume that we have oracle access to the matrix elements of H and that H is s -sparse, i.e., that each row has at most s non-zero elements. In particular, we assume that we have access to two matrix oracles. The first one is the position oracle O_P , that acts on the $2n$ -qubit state $|u\rangle |0\rangle^{\otimes n}$ as

$$O_P |u\rangle |0\rangle^{\otimes n} = \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_{\mathcal{G}}[u]} |u\rangle |v\rangle. \quad (2.24)$$

O_P stores a linear superposition of all the column indices of H in $\tilde{N}_{\mathcal{G}}[u]$, which correspond to non-zero matrix elements at row u and additional dummy indices if the number of non-zero elements is less than s , in the second register. A more detailed introduction of the extended neighborhood $\tilde{N}_{\mathcal{G}}[u]$ is given in Section 3.1.

The second oracle is the angle oracle $O_{\mathcal{J}}$ which contains information about the matrix elements of H . It encodes the angle

$$\mathcal{J}_{uv} = \arccos \sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}}, \quad (2.25)$$

and the sign of H_{uv} in an $(r + 1)$ -qubit ancilla register

$$O_{\mathcal{J}} |u, v\rangle |0\rangle^{\otimes r+1} = |u, v\rangle |\mathcal{J}_{uv}, \text{sgn}_b(H_{uv})\rangle, \quad (2.26)$$

with $\text{sgn}_b(\cdot)$ the binary sign and r the number of bits used to represent ϑ_{uv} . We renormalize H_{uv} in Eq. (2.25) with $\|H\|_{\max} = \max_{u,v} |H_{uv}|$. Specifically, since $0 \leq \vartheta_{uv} \leq \pi/2$, we assume the following r -bit representation of ϑ_{uv}

$$\vartheta_{uv} = \frac{\pi}{2} \sum_{l=1}^r \vartheta_{uv}^{(l)} 2^{-l} + \varepsilon_p, \quad (2.27)$$

with $\vartheta_{uv}^{(l)} \in \mathbb{Z}_2$ and ε_p the precision error. Thus, when we write $|\vartheta_{uv}\rangle$ in an r -qubit register, this means

$$|\vartheta_{uv}\rangle = \left| \vartheta_{uv}^{(1)}, \vartheta_{uv}^{(2)}, \dots, \vartheta_{uv}^{(r)} \right\rangle. \quad (2.28)$$

Finally, we assume that we have a quantum adder ADD available, which adds the information in one quantum register to another, defined as

$$\text{ADD } |a\rangle |b\rangle = |a\rangle |b + a \bmod 2^n\rangle, \quad (2.29)$$

where n is the number of qubits in the second register⁷. The inverse of this operation is a quantum subtractor $\text{ADD}^\dagger = \text{SUB}$. Circuit realizations of the quantum adder are well known in the literature [51–56] and further discussed in Chapter 4.

Now, we detail how a Hermitian $(\alpha, a, 0)$ -block-encoding U_H of a Hamiltonian H can be constructed using the matrix oracles O_P and O_ϑ defined in Eq. (2.24) and Eq. (2.26), respectively. In particular, we limit ourselves to the case when H is real, i.e., symmetric, and all its diagonal elements are positive, which is the case for the type of matrices we consider in this work. We start by considering a $2(n+1)$ -qubit register, where the first $a = n+2$ qubits will be the ancilla qubits needed for the block encoding. We take the block encoding unitary U_H to be of the following form

$$U_H = U_T^\dagger \text{SWAP}_{n+1} U_T, \quad (2.30)$$

where SWAP_{n+1} swaps two $(n+1)$ -qubit registers, i.e., $\text{SWAP}_{n+1} |\psi\rangle |\varphi\rangle = |\varphi\rangle |\psi\rangle$ for any $(n+1)$ -qubit states $|\psi\rangle, |\varphi\rangle$. Thus, Eq. (2.13) and Eq. (2.30) imply that

$$\alpha H = \langle 0|^{n+2} U_T^\dagger \text{SWAP}_{n+1} U_T |0\rangle^{\otimes n+2}. \quad (2.31)$$

We further assume that the map $U_T |0\rangle^{\otimes n+2}$, which maps an n -qubit quantum state to a $2(n+1)$ -qubit quantum state can be written as ($N = 2^n$)

$$U_T |0\rangle^{\otimes n+2} = \sum_{u=1}^N |\psi_u, 0, u\rangle \langle u|, \quad (2.32)$$

where $|\psi_u\rangle$ denotes an $(n+1)$ -qubit state to be determined, $|0\rangle$ the single-qubit zero state and $|u\rangle$ an n -qubit computational basis state. Plugging Eq. (2.32) into Eq. (2.31), we obtain that the states $|\psi_u\rangle$ need to satisfy

$$\alpha H_{uv} = \langle \psi_u, 0, u | 0, v, \psi_v \rangle. \quad (2.33)$$

⁷The number of qubits in the first register does not need to be equal to the one in the second register.

A possible solution is to set $\alpha^{-1} = s\|H\|_{\max}$ and take

$$|\psi_u\rangle = \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_G[u]} (\text{sgn}(u-v))^{\text{sgn}_b(H_{uv})} \left(\sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}} |0\rangle + \sqrt{1 - \frac{|H_{uv}|}{\|H\|_{\max}}} |1\rangle \right) |v\rangle, \quad (2.34)$$

where $\text{sgn}(\cdot)$ and $\text{sgn}_b(\cdot)$ are the normal and binary sign function⁸. Note that Eq. (2.34) requires the diagonal elements to be non-negative in order to give a valid block encoding. If this is not the case, one can always enforce this property by adding a suited matrix proportional to the identity to H .

Given the form of the block encoding in Eq. (2.30), all we need to show is how to construct a unitary U_T that satisfies Eq. (2.32) with states $|\psi_u\rangle$ given in Eq. (2.34). The implementation of U_T consists of three parts. First preparing a superposition of all states $|v\rangle$ with non-zero matrix entries H_{uv} . Second, encoding the matrix entries in the amplitudes, and last, preparing the corresponding signs. All these steps are shown in Fig. 2.4 as a quantum circuit that involves the matrix oracles O_P and $O_{\mathcal{J}}$.

We now describe in detail each transformation. As we see from Fig. 2.4 we have a total of six different registers. The three register highlighted in blue are additional ancilla registers needed for the implementation of the oracles, namely:

- an r -qubit ancilla register to store the value of the angles \mathcal{J}_{uv} given in Eq. (2.25);
- an ancilla qubit to store the value of the sign of the matrix elements H_{uv} ;
- an ancilla qubit (bottom one) to store the value of the sign of $u - v$.

The remaining registers form a $(2n + 1)$ -qubit register that together with an additional qubit (not shown in the figure) will be used to implement the $2(n + 1)$ -qubit block encoding we described at the beginning of this section.

The system starts in the state $|0\rangle^{\otimes r+2+n} |u\rangle |0\rangle$. After, the application of the position oracle O_P defined in Eq. (2.24) the state becomes

$$\xrightarrow{O_P} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_G[u]} |0\rangle^{\otimes r+2} |v\rangle |u\rangle |0\rangle, \quad (2.35)$$

and after the oracle $O_{\mathcal{J}}$ given in Eq. (2.26)

$$\xrightarrow{O_{\mathcal{J}}} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_G[u]} |0\rangle |\mathcal{J}_{uv}, \text{sgn}_b(H_{uv})\rangle |v, u\rangle |0\rangle. \quad (2.36)$$

We continue by reading the angle \mathcal{J}_{uv} in the ancilla register with a series of controlled y -rotations⁹ $cR_y = \prod_{k=1}^r c^k R_y(2^{-k}\pi)$ with target the ancilla qubit on the top of Fig. 2.4 and control the k -th qubit in the binary representation of $|\mathcal{J}_{uv}\rangle$ in Eq. (2.28). This gives

$$\xrightarrow{cR_y} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_G[u]} \left(\sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}} |0\rangle + \sqrt{1 - \frac{|H_{uv}|}{\|H\|_{\max}}} |1\rangle \right) |\mathcal{J}_{uv}, \text{sgn}_b(H_{uv})\rangle |v, u\rangle |0\rangle. \quad (2.37)$$

⁸We set $\text{sgn}(0) = 1$ and $\text{sgn}_b(0) = 0$.

⁹Here, $c^k R_y(\mathcal{J})$ is a controlled version of the y -rotation introduced in Table 2.1.

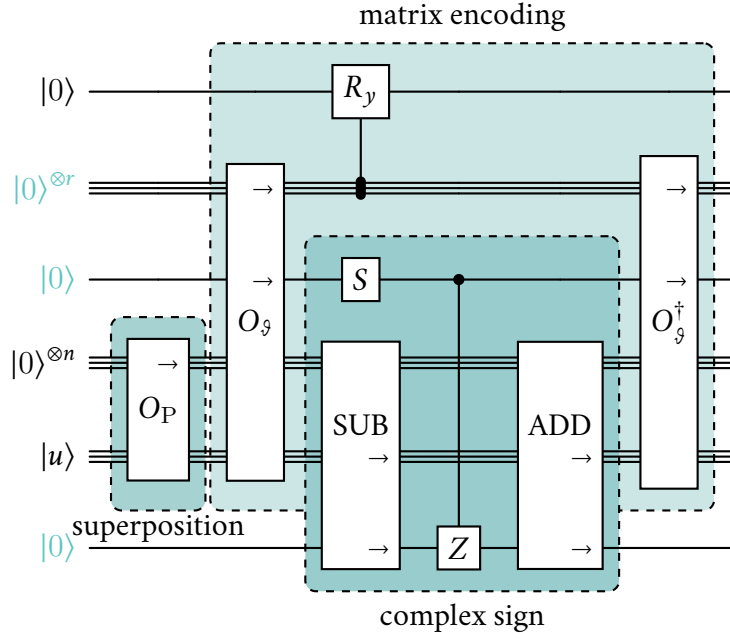


Figure 2.4: Circuit for the state transformation U_T . It consists of three parts. The preparation of the superposition with O_P , the matrix encoding in the amplitude and the multiplication with the complex sign. The order of the registers here is chosen for a compact circuit layout. The first and fourth register combine into $|\psi_u\rangle$ of Eq. (2.32). The second last register is $|u\rangle$ in Eq. (2.32). The second, third and last registers (in blue) are of auxiliary nature and will be reinitialized after the full block encoding of H . In the second register we store the intermediate angle (Eq. (2.25)). The third register is used for the binary sign $\text{sgn}_b(H_{uv})$ and the last for $\Theta(\text{sgn}(u - v))$.

It remains to implement the coefficient $\iota \operatorname{sgn}(u - v)$ for negative H_{uv} . The phase $\iota^{\operatorname{sgn}_b(H_{uv})}$ can be realized with an S -gate to the sign qubit $|\operatorname{sgn}_b(H_{uv})\rangle$:

$$\xrightarrow{S} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_{\mathcal{G}}[u]} \iota^{\operatorname{sgn}_b(H_{uv})} \left(\sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}} |0\rangle + \sqrt{1 - \frac{|H_{uv}|}{\|H\|_{\max}}} |1\rangle \right) |\mathcal{J}_{uv, \operatorname{sgn}_b(H_{uv})}\rangle |v, u\rangle |0\rangle. \quad (2.38)$$

For the relative sign, we subtract the $|v\rangle$ register from the $|u\rangle$ for which we need to increase the target register by one qubit to store the sign

$$|v\rangle |u, 0\rangle \xrightarrow{\text{SUB}} |v\rangle |u - v \bmod 2^{n+1}\rangle. \quad (2.39)$$

Notice that the most significant bit of $u - v \bmod 2^{n+1}$, which is a $(n + 1)$ -bit number, encodes the sign of the regular subtraction $u - v$, since both u and v are n -bit numbers. Thus, we can write

$$|u - v \bmod 2^{n+1}\rangle = |u - v \bmod 2^n\rangle |\operatorname{sgn}_b(u - v)\rangle, \quad (2.40)$$

where the register associated with $|u - v \bmod 2^n\rangle$ is an n -qubit register. The subtraction gives the state

$$\begin{aligned} \xrightarrow{\text{SUB}} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_{\mathcal{G}}[u]} \iota^{\operatorname{sgn}_b(H_{uv})} \left(\sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}} |0\rangle + \sqrt{1 - \frac{|H_{uv}|}{\|H\|_{\max}}} |1\rangle \right) \\ \otimes |\mathcal{J}_{uv, \operatorname{sgn}_b(H_{uv})}\rangle |v, u - v \bmod 2^n\rangle |\operatorname{sgn}_b(u - v)\rangle, \end{aligned} \quad (2.41)$$

A Z -gate, controlled by the qubit in the state $|\operatorname{sgn}_b(H_{uv})\rangle$, applied to the qubit in the state $|\operatorname{sgn}_b(u - v)\rangle$ (represented by the qubit at the bottom in Fig. 2.4) yields the desired coefficient

$$\begin{aligned} \xrightarrow{cZ} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_{\mathcal{G}}[u]} (\iota \operatorname{sgn}(u - v))^{\operatorname{sgn}_b(H_{uv})} \left(\sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}} |0\rangle + \sqrt{1 - \frac{|H_{uv}|}{\|H\|_{\max}}} |1\rangle \right) \\ \otimes |\mathcal{J}_{uv, \operatorname{sgn}_b(H_{uv})}\rangle |v, u - v \bmod 2^n\rangle |\operatorname{sgn}_b(u - v)\rangle, \end{aligned} \quad (2.42)$$

At last we invert the subtraction SUB using the adder ADD

$$\begin{aligned} \xrightarrow{\text{ADD}} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_{\mathcal{G}}[u]} (\iota \operatorname{sgn}(u - v))^{\operatorname{sgn}_b(H_{uv})} \left(\sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}} |0\rangle + \sqrt{1 - \frac{|H_{uv}|}{\|H\|_{\max}}} |1\rangle \right) \\ \otimes |\mathcal{J}_{uv, \operatorname{sgn}_b(H_{uv})}\rangle |v, u\rangle |0\rangle, \end{aligned} \quad (2.43)$$

and the oracle $O_{\mathcal{J}}$ using $O_{\mathcal{J}}^\dagger$

$$\begin{aligned} \xrightarrow{O_{\mathcal{J}}^\dagger} \frac{1}{\sqrt{s}} \sum_{v \in \tilde{N}_{\mathcal{G}}[u]} (\iota \operatorname{sgn}(u - v))^{\operatorname{sgn}_b(H_{uv})} \left(\sqrt{\frac{|H_{uv}|}{\|H\|_{\max}}} |0\rangle + \sqrt{1 - \frac{|H_{uv}|}{\|H\|_{\max}}} |1\rangle \right) \\ \otimes |0\rangle^{\otimes r+1} |v, u\rangle |0\rangle, \end{aligned} \quad (2.44)$$

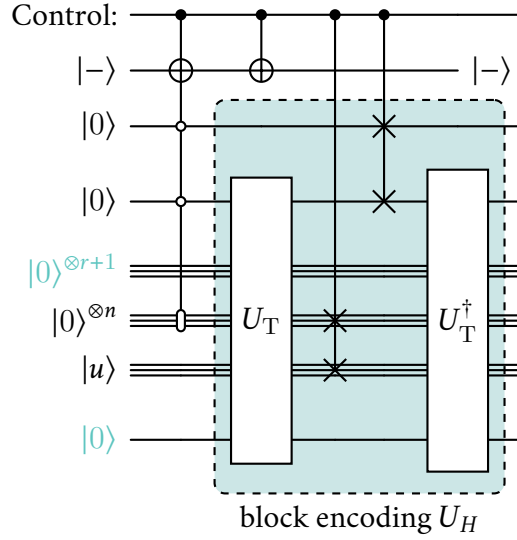


Figure 2.5: Controlled V operator with V defined in Eq. (2.19). The circuit starts with a controlled reflection around $|0\rangle^{\otimes n+2}$ that is implemented by a multi-controlled and a normal CNOT gate to an ancilla qubit initialized in $|-\rangle$. After this we see the controlled version of the block encoding $U_H = U_T^\dagger \text{SWAP}_{n+1} U_T$.

and thus, we have effectively prepared the state $|\psi_u\rangle$ in Eq. (2.34). Notice that in this procedure the $r + 1$ ancilla qubits needed for the implementation of O_g and the ancilla qubit that stores the sign of $u - v$ return to the zero state, and this is also the case for the full block encoding in Eq. (2.13). This is why we should not count them among the a ancilla qubits of the block encoding, which are thus $a = n + 2$.

The circuit implementation of U_H in the block encoding Eq. (2.30) is shown in Fig. 2.5. Note that the total number of qubits was increased by one. This is the last one of the $a = n + 2$ ancilla qubits necessary for the Hilbert-space extension. The operator in front is reflecting around $|0\rangle^{\otimes n+2}$ and transforms U_H into the quantum walk operator defined in Eq. (2.19). It consists of a multi-controlled NOT gate and a regular NOT gate applied to an ancilla qubit initialized in $|-\rangle$. The QPE demands a controlled version which is why we have yet another qubit which controls the CNOT and NOT gates in the reflection, and the SWAP gates in U_H .

2.6 Amplitude amplification

Many quantum algorithms share, that their results are hidden in a superposition of states. Accessing them is a problem solved rather early with Grover's algorithm [31]. Based on the latter, we describe the amplitude amplification (AA) in this section [57]. As the name implies, the effect of this method is to amplify the amplitude of a chosen subset of quantum states. Assuming we have a state $|\Psi_G\rangle$, generated by the operation G , that consists of a desired or *good*

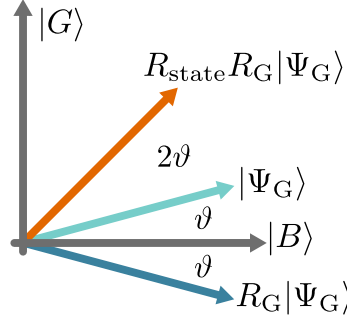


Figure 2.6: Geographical proof of the amplitude amplification (AA). Starting in $|\Psi_G\rangle$, we get closer to the *good* state $|G\rangle$ with each iteration in $AA = (R_{\text{state}}R_G)^t$.

part $|G\rangle$ and a not-desired or *bad* part $|B\rangle$

$$G|0\rangle = |\Psi_G\rangle = \cos \vartheta |B\rangle + \sin \vartheta |G\rangle. \quad (2.45)$$

In order to maximize the amplitude and with it the probability of the *good* contribution we need access to two reflection gates. The first identifies the *good* states and marks them with a negative amplitude (i.e. reflecting around $|B\rangle$) and the second reflects around the original state $|\Psi_G\rangle$

$$R_G = -(2|G\rangle\langle G| - I), \quad (2.46a)$$

$$R_{\text{state}} = 2|\Psi_G\rangle\langle\Psi_G| - I = G(2|0\rangle\langle 0| - I)G^\dagger. \quad (2.46b)$$

Alternating those two reflections multiple times yields in the desired AA

$$AA = (R_{\text{state}}R_G)^t. \quad (2.47)$$

Applied to $|\Psi_G\rangle$ we achieve an amplification of the amplitude of

$$\sin \vartheta \rightarrow g^{(t)} = \sin((2t + 1)\vartheta). \quad (2.48)$$

The proof of the latter is best done geometrical as shown in Fig. 2.6, where the effect of the reflection is shown most natural. After one iteration of $R_{\text{state}}R_G$ we have $\vartheta \rightarrow 3\vartheta$ and if repeated t times we have Eq. (2.48). Alternatively one can derive it in a few lines and with a handful of trigonometric relations. This can overshoot destroying the amplification effect if repeated too often. Hence, the optimum number of iterations for which $g^{(t_{\text{opt}})} = 1$ has to be known or at least estimated in advance

$$t_{\text{opt}} = \frac{\pi}{4\vartheta} - \frac{1}{2}. \quad (2.49)$$

The alternative to AA would be to start sampling and reject every *bad* outcome. On average this requires to sample $O(\sin^{-2} \vartheta)$ times yielding a sampling overhead of

$$t_{G,s} = \frac{1}{\sin^2 \vartheta} \geq \frac{1}{\vartheta^2} \propto t_{\text{opt}}^2, \quad (2.50)$$

which is quadratically slower than AA.

Elastic Structures

The solution of ordinary and partial differential or integral equations describing technical and physical processes, such as the deformation of components under given loads or heat conduction in solids, is rarely successful analytically and must be solved numerically. The necessary transfer to a finite-dimensional equivalent problem requires a discretization process using methods such as the finite element or finite volume method [58]. In our work, we are particularly interested in problems for which this discretization step leads to an effective model of coupled harmonic oscillators. A prominent example of these problems are those that emerge in the manufacturing industry as we detail below.

Due to their time efficiency, finite element models are often used in practice to simulate machining processes, such as milling. Milling is a metal cutting manufacturing process that uses the circular cutting motion of a tool, usually with multiple cutting edges, to produce a vast variety of surfaces on a workpiece (see Fig. 3.1). In all milling processes, in contrast to other processes, such as turning and drilling for instance, the cutting edges are not constantly engaged. Still, at least one cutting interruption per cutting edge occurs with each revolution of the tool [59]. Due to the constant cutting interruptions, depending on the milling cutter speed, a dynamic excitation of the workpiece and the tool can occur, which can have a negative effect on the surface quality in the form of vibration marks. Dynamic process stability simulations are carried out to analyze the vibrations and to improve the process design for the milling of thin-walled components [60–64]. The ultimate goal of these simulations is to obtain response functions over a certain frequency range, i.e., the typical frequency range of the cutting tool, that provide information about the effect that the cutting tool at a certain location has on the workpiece at other, possibly different, locations. For this particular case, the response function is a so-called compliance (with units m N^{-1}) as a function of the frequency of the cutting edge, as shown in Fig. 3.1, which effectively quantifies the vibration of the workpiece caused by the cutting tool.

Ref. [1] analyzed a workflow for milling dynamics simulation, examining computational time and identifying numerical problems that can be suited for quantum algorithms. The milling

The Intro and Section 3.1 of this chapter are published in the preprint “Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405.08694”. The introduction of this chapter was written by Stefan Schröder. Section 3.2 has been shared in the preprint “Sven Danz, Tobias Stollenwerk, and Alessandro Ciani. “Quantum oracles for the finite element method”. In: *arXiv e-prints* (2025). arXiv: 2504.19827”.

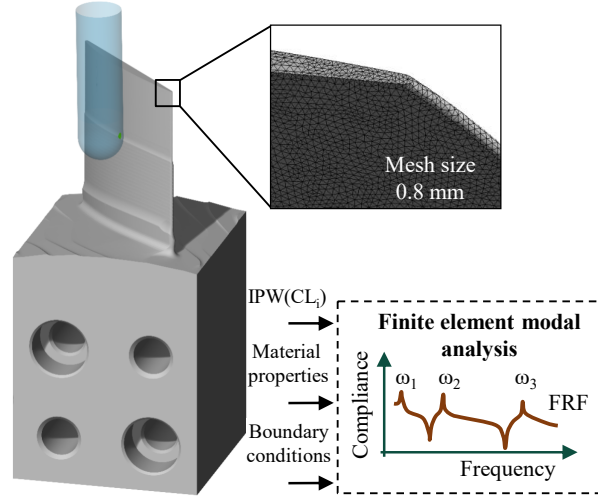


Figure 3.1: IPW of a single blade demonstrator (left) and the detailed representation of the finite element mesh. The cutting tool is shown in blue. With the cutter location-dependent IPW, material properties (density, Young's modulus, Poisson's ratio), and boundary conditions as input, the finite element modal analysis can be conducted and the frequency response function (FRF) calculated by solving the corresponding eigenproblem.

dynamics workflow considers the dynamic behavior of the workpiece, which continuously varies due to changes in the stiffness and mass of the workpiece caused by the material removal and the position-dependent force excitation by the milling tool [65]. This analysis highlighted the fact that finite element modal analysis is typically a computationally intensive simulation application in milling of thin-walled aerospace components, and as such could benefit from possible advantages that quantum algorithms could offer for this task.

The rest of this chapter is structured as follows. In Section 3.1, we draw the connection between the computation of the response function of coupled oscillators and the solution of an eigenproblem. Section 3.2 discusses the computation of the mass and stiffness matrix entries, characteristic for FEM. We derive the matrices based on the Lagrangian formalism giving an alternative to the non-trivial force-based derivation typical in literature.

3.1 Response of coupled oscillators

Let us consider a system of masses coupled by spring constants as shown in Fig. 3.2 and let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with \mathcal{V} the set of vertices and \mathcal{E} the set of edges, that describes the network of oscillators. With each vertex $u \in \mathcal{V}$ we associate a harmonic oscillator with mass $m_u > 0$ and a coupling to a common wall with spring constant $\kappa_u > 0$. We denote by $N = |\mathcal{V}|$ the number of oscillators. With each edge $(u, v) \in \mathcal{E}$ in the graph, instead, we associate a coupling spring constant $\kappa_{uv} > 0$. In what follows, we implicitly assume an ordering of the vertices, i.e., we associate with each $u \in \mathcal{V}$ a number between 1 and N . For

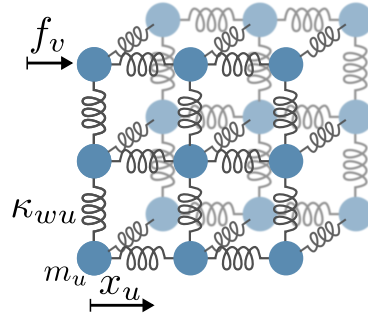


Figure 3.2: A representation of coupled oscillators. Point-like masses m_u in blue are connected by springs with spring constant κ_{wu} . The force f_v applied to mass v yields a displacement x_u at mass u .

notational simplicity, we also denote this number by u , since its meaning is clear from the context. Additionally, we denote by $N_{\mathcal{G}}[u]$ the closed neighborhood of $u \in \mathcal{V}$, i.e., the set of vertices connected to u by an edge plus u itself. We call $s_u = |N_{\mathcal{G}}[u]|$ and define the sparsity s as $s = \max_{u \in \mathcal{V}} s_u$. For later convenience, we also define the set of vertices $\tilde{N}_{\mathcal{G}}[u]$ such that $N_{\mathcal{G}}[u] \subset \tilde{N}_{\mathcal{G}}[u]$ and $|\tilde{N}_{\mathcal{G}}[u]| = s$. Basically, $\tilde{N}_{\mathcal{G}}[u]$ is the neighborhood of u enlarged with dummy vertices to ensure that the total number of elements in the set is s .

3.1.1 Unperturbed dynamics

We can compactly write the dynamical equations that govern the motion of the coupled oscillator systems, by defining two $N \times N$ matrices, namely a diagonal mass matrix \mathbf{M} with matrix elements

$$M_{uv} = m_u \delta_{uv}, \quad (3.1)$$

and a stiffness matrix \mathbf{K} with matrix elements

$$K_{uv} = \begin{cases} \kappa_u + \sum_{w \in N_{\mathcal{G}}[u] \setminus \{u\}} \kappa_{wu} & \text{if } u = v, \\ -\kappa_{uv} & \text{if } (u, v) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \quad (3.2)$$

Additionally, we denote by \mathbf{x} the N -dimensional column vector whose elements are the oscillator positions x_u with respect to the equilibrium state.

The system evolves according to the dynamical equation

$$\mathbf{M} \frac{d^2 \mathbf{x}}{dt^2} = -\mathbf{K} \mathbf{x}. \quad (3.3)$$

In order to transform this problem into an eigenproblem, let us first define the matrix

$$\mathbf{H} = \mathbf{M}^{-1/2} \mathbf{K} \mathbf{M}^{-1/2}, \quad (3.4)$$

which, as we will see, will play the role of a Hamiltonian in our quantum algorithm. Note that since \mathbf{M} is positive definite, also $\sqrt{\mathbf{M}}$ is positive definite and thus invertible. The matrix \mathbf{H} is Hermitian and positive-semidefinite, and we denote its eigenvalues by $\lambda_j \geq 0$ for $j \in \{1, \dots, N\}$. We now show that solving the dynamical equation Eq. (3.3) is equivalent to diagonalizing the matrix \mathbf{H} . We denote by \mathbf{W} an orthogonal matrix that diagonalizes \mathbf{H} ,

$$\mathbf{\Lambda} = \mathbf{W}^T \mathbf{H} \mathbf{W}, \quad (3.5)$$

where $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues of \mathbf{H} on the diagonal, i.e., $\Lambda_{jj} = \lambda_j$. Defining the vector of normal mode variables \mathbf{y} as

$$\mathbf{y} = \mathbf{W}^T \mathbf{M}^{1/2} \mathbf{x}, \quad (3.6)$$

we get an uncoupled system of harmonic oscillators satisfying

$$\frac{d^2 \mathbf{y}}{dt^2} = -\mathbf{\Lambda} \mathbf{y} \implies \frac{d^2 y_j}{dt^2} = -\lambda_j y_j. \quad (3.7)$$

Since $\lambda_j \geq 0$ they are usually denoted as $\lambda_j = \omega_j^2$, with ω_j the resonance frequency of the normal mode j . Note that Eq. (3.6) implies that

$$x_u = \frac{1}{\sqrt{m_u}} \sum_{j=1}^N W_{uj} y_j. \quad (3.8)$$

3.1.2 Adding an external force

Let us now add an external force to Eq. (3.3). In particular, we assume that a time-dependent force $f_v(t)$ is applied to the oscillator $v \in \mathcal{V}$, and we gather all the forces in an N -dimensional column vector $\mathbf{f}(t)$. Including the forcing term, the dynamical equation reads

$$\mathbf{M} \frac{d^2 \mathbf{x}}{dt^2} = -\mathbf{K} \mathbf{x} + \mathbf{f}. \quad (3.9)$$

Eq. (3.9) gives rise to a convolution relation between $\mathbf{f}(t)$ and $\mathbf{x}(t)$

$$\mathbf{x}(t) = \int_{-\infty}^{+\infty} d\tau \mathbf{g}(t - \tau) \mathbf{f}(\tau), \quad (3.10)$$

where $\mathbf{g}(t)$ is an $N \times N$ matrix that we call the matrix response function. The matrix elements $g_{uv}(t)$ can simply be interpreted as the response of the oscillator u when a forcing term on the oscillator v , $f_v(t)$, is a Dirac delta function (i.e. $f_v(t) = \delta(t)$) while no force is applied to the other oscillators¹. This interpretation also suggests that causality implies $\mathbf{g}(t) = 0$ if $t < 0$ (i.e we consider a retarded response function). Thus, we can simply take the upper limit of

¹The response function is also called the Green's function for the system of coupled oscillators.

the integral in Eq. (3.10) to be t rather than $+\infty$. Eq. (3.10) is the defining equation of a linear time-invariant system [66].

Now, let us introduce the bilateral Laplace transform of a generic function $\mathbf{z}(t)$ as

$$\mathbf{Z}(\nu) = \int_{-\infty}^{+\infty} dt e^{-\nu t} \mathbf{z}(t), \quad \nu = \sigma + i\omega \in \mathbb{C}. \quad (3.11)$$

The Laplace transform is defined only in the region of convergence, that is the complex domain where the integral in Eq. (3.11) converges. From the convolution theorem for the Laplace transform Eq. (3.10) in Laplace domain becomes

$$\mathbf{X}(\nu) = \mathbf{G}(\nu) \mathbf{F}(\nu), \quad (3.12)$$

with $\mathbf{G}(\nu)$ the Laplace transform of the response function in time domain $\mathbf{g}(t)$, while $\mathbf{X}(\nu)$ and $\mathbf{F}(\nu)$ the Laplace transform of the position vector $\mathbf{x}(t)$ and the vector of forces $\mathbf{f}(t)$, respectively.

In the Sections 5.1 and 5.2, we focus on a quantum algorithm to determine the diagonal elements of $\mathbf{G}(\nu)$ that take the form

$$G_{uu}(\nu) = \frac{1}{m_u} \sum_{j=1}^N \frac{W_{uj}^2}{\lambda_j + \nu^2}, \quad (3.13)$$

which we call the local response functions. However, it is also possible to obtain the off-diagonal elements of $\mathbf{G}(\nu)$ or non-local response function at the price of adding a Hadamard test, as we show in Section 5.3. We provide a derivation of Eq. (3.13), as well as the more general definition for local and non-local response, in Section 3.A.

Eq. (3.13) shows that the response function $G_{uu}(\nu)$ is completely determined once we solve the eigenproblem associated with \mathbf{H} . In fact, all we need to compute the response function is the eigenvalues λ_j and the coefficients W_{uj}^2 , with W_{uj} the matrix elements of the orthogonal matrix \mathbf{W} that diagonalizes \mathbf{H} . As discussed in the intro of this chapter, the response function is the relevant quantity to study in practical applications. State-of-the-art classical algorithms that are used in practice for the eigenproblem at hand, such as the QR algorithm, need at least $O(N^3)$ arithmetic operations. In fact, most of the fastest classical algorithms for eigenvalue computation rely on the Householder transformation which transforms a matrix into the Hessenberg form (almost triangular). This transformation needs $4N^3/3$ operations (see Chap. 11 p. 474 in [67]) and dominates the scaling of the QR algorithm. The sparsity of \mathbf{H} reduces the runtime to $O(N^2s)$.

3.2 Finite element method for elastic structures

The derivation of the mass and stiffness matrices for a continuous, solid structure using finite element method (FEM) is well understood and documented in the literature (e.g. [68–71]). The derivation of force equations in continuous systems can be intricate and less intuitive, which is why we describe an alternative more concise derivation in Section 3.2.1 based on the Lagrangian formalism. We further give an introduction to test functions and to FEM applied to classical oscillator problem. Section 3.2.2 describes the limitations for the choice of test function.

3.2.1 Discretization of elastic structures

We consider a continuous solid in d dimensions defined in a domain $\Omega \subseteq \mathbb{R}^d$, whose dynamical behavior is described by its kinetic and potential energy density. We combine both of them in the Lagrangian density

$$l(\mathbf{x}, t) = \frac{1}{2} \rho(\mathbf{x}) \mathbf{v}(\mathbf{x}, t)^T \mathbf{v}(\mathbf{x}, t) - \frac{1}{2} \sum_{\alpha, \beta, \gamma, \delta=1}^d Y^{(\alpha\beta\gamma\delta)}(\mathbf{x}) \varepsilon^{(\alpha\beta)}(\mathbf{x}, t) \varepsilon^{(\gamma\delta)}(\mathbf{x}, t). \quad (3.14)$$

The first term is the kinetic energy per unit volume with the density ρ and the velocity $\mathbf{v} = (v^{(1)}, \dots, v^{(d)})^T \in \mathbb{R}^d$ (cf. [71, 72]). The potential is reduced to its contribution from the elastic energy, neglecting others like electromagnetic potentials, thermal energy, or pressure. Hence, the second term in Eq. (3.14) is the elastic energy per unit volume, with the material-dependent elasticity tensor Y and the strain tensor ε . We recommend [68, 70, 73–79] for a detailed derivation and discussion of the symmetry properties of Y and ε . For now, we are only interested in the following property

$$Y^{(\alpha\beta\gamma\delta)} = Y^{(\alpha\beta\delta\gamma)} = Y^{(\beta\alpha\delta\gamma)}. \quad (3.15)$$

In its general form, all contributions in Eq. (3.14) depend on the position $\mathbf{x} = (x^{(1)}, \dots, x^{(d)})^T \in \mathbb{R}^d$ in space and on the time t . For the sake of compactness, we mostly omit the dependency on position \mathbf{x} and time t , unless needed for clarity. Both the strain tensor and the velocity are defined in terms of the material displacement $\mathbf{u} = (u^{(1)}, \dots, u^{(d)})^T$ from its equilibrium position

$$\varepsilon^{(\alpha\beta)} = \frac{1}{2} \left(\frac{\partial u^{(\alpha)}}{\partial x^{(\beta)}} + \frac{\partial u^{(\beta)}}{\partial x^{(\alpha)}} \right), \quad (3.16a)$$

$$\mathbf{v} = \frac{\partial \mathbf{u}}{\partial t}. \quad (3.16b)$$

The strain is the change of the displacement with respect to the position \mathbf{x} and the velocity the analogue with respect to time t . The total Lagrangian of the system reads

$$\mathcal{L} = \frac{1}{2} \int_{\mathbb{R}^d} d^d x \rho \mathbf{v}^T \mathbf{v} - \frac{1}{2} \sum_{\alpha, \beta, \gamma, \delta=1}^d \int_{\mathbb{R}^d} d^d x Y^{(\alpha\beta\gamma\delta)} \frac{\partial u^{(\alpha)}}{\partial x^{(\beta)}} \frac{\partial u^{(\gamma)}}{\partial x^{(\delta)}}, \quad (3.17)$$

where we used Eqs. (3.15) and (3.16a) to replace $\varepsilon^{(\alpha\beta)}$. The corresponding Euler-Lagrange equation describes the motion of the system. In finite element analysis, the continuous equation of motion is called the *strong form* because it is valid at every point \mathbf{x} . Solving a strong form analytically is not always possible which is why numerical approaches are common in classical simulations. For this, we discretize Eq. (3.17) and with it its Euler-Lagrange equation, obtaining a system of equation that reproduces the exact solution at a finite number of points N . On one hand, the number N should be small so that a computer can solve the equation for all those points in reasonable time. On the other hand, it should be large enough to approximate the

behavior of the full system sufficiently. Finding the sweets-spot can be difficult in practice and yields in a trade-of between accuracy and computation cost.

In order to discretize the system, we replace $\mathbf{u}(\mathbf{x})$ with an approximation $\bar{\mathbf{u}}(\mathbf{x})$

$$\bar{\mathbf{u}}(\mathbf{x}) = \sum_{i=1}^N \mathbf{u}_i \varphi_i(\mathbf{x}), \quad \mathbf{u}_i = \mathbf{u}(\mathbf{x}_i). \quad (3.18)$$

$\bar{\mathbf{u}}(\mathbf{x})$ is a linear combination of so-called *test functions* $\varphi_i(\mathbf{x})$ which are only non-zero within small parts of the full space. This approximation is exact in N arbitrary positions $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}^2$. These positions should be well distributed within the geometry $\Omega = \{\mathbf{x} : \rho(\mathbf{x}), Y^{(\alpha\beta\gamma\delta)}(\mathbf{x}) \neq 0\}$ in order to obtain a good approximation. However, they are not limited to Ω . For the purpose of a simpler mesh one can place some points external to Ω . A similar approximation $\bar{\mathbf{v}}(\mathbf{x})$ is required for the first time-derivative

$$\bar{\mathbf{v}}(\mathbf{x}) = \sum_{i=1}^N \mathbf{v}_i \phi_i(\mathbf{x}), \quad \mathbf{v}_i = \mathbf{v}(\mathbf{x}_i) = \frac{d\mathbf{u}_i}{dt}. \quad (3.19)$$

Here, $\{\phi_i(\mathbf{x}), \forall i \in [N]\}$ is another set of test functions, which are not necessarily equal to $\varphi_i(\mathbf{x})$. Using the two approximations Eqs. (3.18) and (3.19) in Eq. (3.17) yields the discretized Lagrangian

$$\bar{\mathcal{L}} = \frac{1}{2} \sum_{\alpha, \gamma=1}^d \sum_{i,j=1}^N \left[v_i^{(\alpha)} M_{ij}^{(\alpha\gamma)} v_j^{(\gamma)} - u_i^{(\alpha)} K_{ij}^{(\alpha\gamma)} u_j^{(\gamma)} \right]. \quad (3.20)$$

In Eq. (3.20) we introduced the elements of the mass matrix \mathbf{M} defined by

$$M_{ij}^{(\alpha\gamma)} = \delta^{(\alpha\gamma)} \int_{\mathbb{R}^d} d^d x \rho \phi_i \phi_j. \quad (3.21)$$

with $\delta^{(\alpha\gamma)}$ the Kronecker delta. Similar, the elements of the stiffness matrix \mathbf{K} are defined by

$$K_{ij}^{(\alpha\gamma)} = \sum_{\beta, \delta=1}^d \int_{\mathbb{R}^d} d^d x Y^{(\alpha\beta\gamma\delta)} \frac{\partial \phi_i}{\partial x^{(\beta)}} \frac{\partial \phi_j}{\partial x^{(\delta)}}. \quad (3.22)$$

Denoting by $\hat{e}_i^{(\alpha)}$ one of dN -dimensional unit, column vectors, the mass and stiffness matrices can formally be written as $dN \times dN$ matrices given by

$$\mathbf{M} = \sum_{i,j=1}^N \sum_{\alpha, \gamma=1}^d M_{ij}^{(\alpha\gamma)} \hat{e}_i^{(\alpha)} \left(\hat{e}_j^{(\gamma)} \right)^T, \quad (3.23a)$$

$$\mathbf{K} = \sum_{i,j=1}^N \sum_{\alpha, \gamma=1}^d K_{ij}^{(\alpha\gamma)} \hat{e}_i^{(\alpha)} \left(\hat{e}_j^{(\gamma)} \right)^T, \quad (3.23b)$$

²The Roman subscripts here count over the N positions and should not be confused with the Greek superscripts in parentheses that count over the d spacial dimensions.

respectively. Note that the mass and stiffness matrix are both symmetric, i.e., $M_{ij}^{(\alpha\gamma)} = M_{ji}^{(\gamma\alpha)}$ and $K_{ij}^{(\alpha\gamma)} = K_{ji}^{(\gamma\alpha)}$. We now collect all dynamical variables $u_i^{(\alpha)}$ in a single dN -dimensional column vector defined as

$$\vec{\mathbf{u}} = \sum_{i=1}^N \sum_{\alpha=1}^d u_i^{(\alpha)} \hat{\mathbf{e}}_i^{(\alpha)}. \quad (3.24)$$

With these definitions, the Lagrangian $\bar{\mathcal{L}}$ in Eq. (3.20) can be written compactly as

$$\bar{\mathcal{L}} = \frac{1}{2} \frac{d\vec{\mathbf{u}}}{dt}^T \mathbf{M} \frac{d\vec{\mathbf{u}}}{dt} - \frac{1}{2} \vec{\mathbf{u}}^T \mathbf{K} \vec{\mathbf{u}}. \quad (3.25)$$

Using the symmetry properties of \mathbf{M} and \mathbf{K} , the associated Euler-Lagrange equations can be compactly written as

$$\mathbf{M} \frac{d^2 \vec{\mathbf{u}}}{dt^2} = -\mathbf{K} \vec{\mathbf{u}}. \quad (3.26)$$

This is equivalent to Eq. (3.3).

In the remaining thesis, we refer to the \mathbf{H} matrix in Eq. (3.4) as the Hamiltonian. From now on we avoid using the bold symbols, simply denoting the Hamiltonian as H , and do the same for the other matrices.

3.2.2 Limitations of the test functions

In general an arbitrary non-zero test function can be chosen as long as Eqs. (3.21) and (3.22) are well-defined. Furthermore, the i -th test function has to satisfy

$$\varphi_i(\mathbf{x}_j) = \delta_{ij}, \quad (3.27)$$

with δ_{ij} the Kronecker delta, to guarantee $\bar{\mathbf{u}}(\mathbf{x}_i) = \mathbf{u}(\mathbf{x}_i) = \mathbf{u}_i$, $\forall i$. The same is valid for $\phi_i(\mathbf{x})$. In the remaining part of this subsection we will write only $\varphi_i(\mathbf{x})$ meaning $\phi_i(\mathbf{x})$ as well.

The approximation $\bar{\mathbf{u}}(\mathbf{x})$ does not need to be exact for positions $\mathbf{x} \notin \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. However, they are conventionally chosen close to the exact solution $\mathbf{u}(\mathbf{x})$. Hence, $\varphi_i(\mathbf{x})$ should fulfill

$$\sum_i^N \varphi_i(\mathbf{x}) \approx 1, \quad \forall \mathbf{x}. \quad (3.28)$$

The number of overlapping test functions at a point \mathbf{x} defines the sparsity of our matrices \mathbf{K} and \mathbf{M} . Sparse matrices are easier to handle in numerical calculations, which is why another convention is to set the test function to zero for large parts of the domain.

3.2.3 Boundary conditions

Up to this point we have considered a fully free system, i.e. without any boundary conditions. For structural problems it is common to fix a given number of nodes $\mathbf{x}_i \in \mathcal{X}_D$ (i.e. $u_{i\alpha} = \text{const.}$), which is called Dirichlet boundary condition. Due to this, we know some *unknown* in $\vec{\mathbf{u}}$ in advance which allows us to rearrange the original equation of motion (see Eqs. (3.4) and (3.26)), which we extend by a force term

$$\left(\mathbf{H} + \frac{d^2}{dt^2} \right) \vec{\mathbf{z}} = \vec{\mathbf{f}}, \quad (3.29)$$

with $\vec{\mathbf{z}} = \mathbf{M}^{1/2} \vec{\mathbf{u}}$. We sort the elements of $\vec{\mathbf{z}}$ into *unknown* $\vec{\mathbf{z}}_u$ and *known* $\vec{\mathbf{z}}_k$. This allows us to write

$$\begin{pmatrix} \mathbf{H}_{kk} + \frac{d^2}{dt^2} & \mathbf{H}_{ku} \\ \mathbf{H}_{uk} & \mathbf{H}_{uu} + \frac{d^2}{dt^2} \end{pmatrix} \begin{pmatrix} \vec{\mathbf{z}}_k \\ \vec{\mathbf{z}}_u \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{f}}_k \\ \vec{\mathbf{f}}_u \end{pmatrix}. \quad (3.30)$$

Applying force to a fixed node is useless (i.e. $\vec{\mathbf{f}}_k = 0$), thus the latter reduces to our new equation of motion

$$\left(\mathbf{H}_{uu} + \frac{d^2}{dt^2} \right) \vec{\mathbf{z}}_u = \vec{\mathbf{f}}_u - \mathbf{H}_{uk} \vec{\mathbf{z}}_k. \quad (3.31)$$

Hence, applying Dirichlet boundary conditions is equivalent to reducing the matrix \mathbf{H} to the remaining free nodes followed by a redefinition of the force term $\vec{\mathbf{f}}_u \rightarrow \vec{\mathbf{f}}_u - \mathbf{H}_{uk} \vec{\mathbf{z}}_k$. Other boundary conditions (e.g. Neumann boundary conditions) effect also only the force term. We recommend Refs. [69, 70] for more details. For the computation of the eigenpairs we focus on the harmonic part of Eq. (3.31) in the following.

Appendix

3.A Analytical form of the response functions

In this appendix, we show that for a system of coupled harmonic oscillators the matrix elements of the response function $G_{uv}(\nu)$ in Laplace domain can be written as

$$G_{uv}(\nu) = \frac{1}{\sqrt{m_u m_v}} \sum_{j=1}^N \frac{W_{uj} W_{vj}}{\lambda_j + \nu^2}. \quad (3.32)$$

We remark that this is a well-known fact in the study of harmonic systems, in particular in the theory of electrical circuits, where it appears in various forms [80–82]. Also, it is the base for the black-box quantization method for quantum electrical circuits [83]. Here we provide a derivation for our problem adapted from Ref. [84].

Let us consider the case in which we apply to oscillator ν an impulse $f_\nu(t) = \tilde{f}_\nu \delta(t)$, while no force is applied to the other oscillators, i.e., $f_{\nu'}(t) = 0$ for $\nu' \neq \nu$. In terms of the normal mode variables in Eq. (3.6), the dynamical equation in time domain Eq. (3.9) reads

$$\frac{d^2 y_j}{dt^2} = -\lambda_j y_j + \frac{W_{vj}}{\sqrt{m_v}} \tilde{f}_\nu \delta(t), \quad (3.33)$$

which in Laplace domain gives

$$Y_j(\nu) = \frac{1}{\sqrt{m_v}} \frac{W_{vj}}{\lambda_j + \nu^2} \tilde{f}_\nu, \quad (3.34)$$

for $j \in \{1, \dots, N\}$. From Eq. (3.8), we get that the Laplace transform of $x_u(t)$ can be written as

$$X_u(\nu) = \frac{1}{\sqrt{m_u}} \sum_{j=1}^N W_{uj} Y_j(\nu) = \frac{1}{\sqrt{m_u m_v}} \sum_{j=1}^N \frac{W_{uj} W_{vj}}{\lambda_j + \nu^2} \tilde{f}_\nu, \quad (3.35)$$

from which Eq. (3.32) follows.

Quantum Arithmetic

Before we dive any deeper into the computation of response functions, we require a basis of quantum routines for the computation of arithmetic operations. To be more precise, in this chapter, we discuss quantum routines for addition, multiplication and eventually propose some for the computation of signomial functions (cf. Eq. (5.10)). The latter are required for the implementation of the oracles used in the block-encoding of FEM based matrices. Classically one would compute the series in Eq. (5.10) in two steps: the computation of the square root followed by the computation of a polynomial. The quantum arithmetic operations, and their complexity are necessary for those two routines, are described in detail in this chapter. In order to preserve potential polynomial or exponential advantage in algorithms, that rely on this oracle, it is necessary that the oracle scales at most polylogarithmically in the matrix size N . The routines we propose are within this limitation. We begin by reviewing state-of-the-art methods for addition in Section 4.1. Both, the computation of polynomials and the square root, require the primitives addition and multiplication. Their construction is described generically in Sections 4.2 and 4.3, respectively.

We mainly summarize and combine existing literature in this chapter. Our contribution here is dominated by filling gaps (e.g. extending it for signed fixed-point numbers), fixing memory issues by adding reinitialization, presenting circuits for all routines, and a detailed analysis of the computational complexity for the computation of signomial functions.

4.1 Adders

In this thesis, we focus on the so-called fixed-point arithmetic encoded in the computational basis. To be more precise, we use the two's complement form most of the time to encode numbers $b \in [-2^{r-p}, 2^{r-p} - 2^{-p}]$ in $r + 1$ qubits. $p \in [0, \dots, r]$ is the binary point position in relation to the least significant qubit. The mapping between b and its two's complement form in qubits is summarized by the following mapping:

$$2^p b \mod 2^{r+1} = \left(\sum_{k=0}^r 2^k b_{k+1} \right) + 2^p \varepsilon_b \rightarrow |b_{r+1}, \dots, b_1\rangle \equiv |b\rangle, \quad (4.1)$$

This chapter has been published in the preprint “Sven Danz, Tobias Stollenwerk, and Alessandro Ciani. “Quantum oracles for the finite element method”. In: *arXiv e-prints* (2025). arXiv: 2504.19827”.

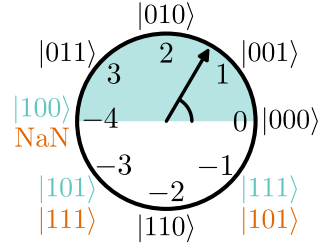


Figure 4.1.1: Comparison of the two's complement (blue, top string) and the sign-magnitude representation (orange, bottom string). Both forms share the same representation for positive numbers but are inverted for negative numbers. Here, $r = 2$ and $p = 0$.

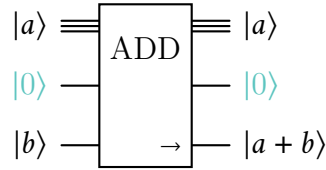


Figure 4.1.2: Generic in-place quantum adder. The \rightarrow symbol marks the target register. The center register in blue is just large enough to satisfy the ancilla qubit requirements of the used routine (cf. Table 4.1.1).

where $\{b_k \in \{0, 1\}, \forall k \in [r + 1]\}$ are the binary digits of b . This representation comes with an error $|\varepsilon_b| \leq 2^{-(p+1)}$. In what follows, $r + 1$ will be the default size of all registers used to store numbers. We further assume, that r and p are fixed and satisfy $|b| \ll 2^{r-p}$ for all numbers b . This prevents overflow due to addition, multiplication or more advanced computations. The two's complement form is mostly used in the following because it can handle the addition of negative numbers naturally. However, there are other forms like the sign-magnitude, that is beneficial for multiplications. We compare those to representations in Fig. 4.1.1

The basis of every arithmetic operation is the addition. The corresponding in-place¹ quantum routine ADD would read two quantum register with numbers encoded (e.g. a and b) and replace one of them with the sum of both

$$\text{ADD } |a\rangle |b\rangle = |a\rangle |a + b\rangle. \quad (4.2)$$

The register with altered value is called target register. To be more precise, the result in the target register would be $2^{-p}(2^p(a+b) \bmod 2^r)$ due to the cyclic nature of binary numbers. However, we omit the modulo in the assumption that we will not consider overflow in the following. This is only of relevance to subtraction which we handle by using the two's complement form. The gate symbol for the in-place adder is shown in Fig. 4.1.2 in which we added a possible ancilla register that may be necessary for some routines.

Based on the basic adder one can find modified versions. Every binary quantum adder can be easily transformed into a subtractor SUB by running it backwards (i.e. $\text{SUB} = \text{ADD}^\dagger$).

¹The alternative are *out-of-place* operations which require an additional register for the result.

Table 4.1.1: Quantum adder routines for r -digit summands. The runtimes and memory requirements are shown for representative specific implementations of each type of adder (see discussion in the main text).

Method	Runtime t_{ADD}	Memory n_{ADD} in qubits
Carry-ripple [52]	$O(r)$	$2r + 1$
Carry-select [56]	$O(\sqrt{r})$	$2r + \sqrt{r}$
Conditional sum [55]	$O(\log_2 r)$	$\approx 7r - 6$
Carry-lookahead [86]	$O(\log_2 r)$	$4r - \log_2 r - 1$
Fourier transform [51]	$O(r^2)$	$2r$

Furthermore, one can replace the quantum register $|a\rangle$ with a classical register achieving a *quantum-classical* adder qcADD_l (subtractor qcSUB_l), that adds (subtracts) the fixed number l encoded in the classical register

$$\text{qcADD}_l |b\rangle = |b + l\rangle, \quad (4.3a)$$

$$\text{qcSUB}_l |b\rangle = |b - l\rangle. \quad (4.3b)$$

Knowing the number l classically allows us to replace controlled operations connecting the classical register $|l\rangle_c$ with the quantum register $|b\rangle$ and (or) the ancilla registers with operations that act solely on the quantum registers. For instance a CNOT with control a bit in the classical register and target one in a quantum register would be replaced by the identity if the classical bit is 0 and by a NOT gate if it is 1. This makes quantum-classical routines faster compared to their *quantum-quantum* counterparts.

For the purpose of simple addition, one can use different algorithms. Classical methods that can be also implemented on quantum devices are already known in the literature. One of those methods is the carry-ripple addition, a rather slow but simple and memory efficient method that adds two numbers bit-by-bit. Two quantum versions of this algorithm are described in Refs. [54, 85]. The runtime of those algorithms scale linearly with the number of binary digits r (no separate sign qubit), but require one additional quantum register of size $O(r)$ for the carry (cf. Table 4.1.1). A more advanced version of this method is described in Ref. [52], that requires only 2 additional ancilla qubits on top of the $2r$ qubits for the two summands. Finally, Ref. [87] halved the runtime cost compared to Ref. [52] at the price of re-introducing r ancillas.

One disadvantage of the carry-ripple method is that most of the qubits are idle most of the time. This can be prevented by parallelization methods also known from classical computing. Those reduce the total runtime while increasing the memory requirements. The first of those is the carry-select adder, that reduces the total runtime to $O(\sqrt{r})$. It does so by splitting the summands bitwise into p packs of q digits. The partial q -qubit adders are executed for both possible carry input values (0 and 1). This allows us to combine them later with p so-called multiplexer that choose the correct result depending on the carry from the previous package. The total runtime scales like $O(p + q)$, which is minimal for $p = q = \sqrt{r}$. This method was first proposed in Ref. [88] and later fully described in different versions in Refs. [55, 56].

The carry-select method can be further improved by using a cascade-like instead of a linear multiplexer structure. This allows us to pre-combine number packages parallel before they get combined again. In this way, the runtime can be reduced to $\mathcal{O}(q + \log_2 p)$ which has optimal scaling $\mathcal{O}(\log_2 r)$ for $q = 1$ and $p = r$. That method is called conditional sum addition in Ref. [55].

Another classical method is the carry-lookahead addition. The idea is to split the computation of the carry and the sum into two parts executing one after the other. This yields knowledge of all carries before the bitwise addition starts, which allows us to execute them in parallel and reduces their runtime to $\mathcal{O}(1)$. The runtime of the computation of the carries scales like $\mathcal{O}(\log_2 r)$ due to a cascade-like structure. This method was first proposed for quantum devices in Ref. [86] and later further analyzed in Ref. [55].

A completely different approach, that makes use of the unique properties of quantum states, is the QFT adder originally proposed in Ref. [53] and later extended in Ref. [51]. This method uses the QFT to move one of the two summands into the phase of the quantum state, which can be increased or decreased by phase rotations. If controlled by the other summand, this yield an intuitive algorithm for addition and subtraction. An application of the inverse QFT at the end moves the sum back into the basis of the quantum state. Both, the QFT and the controlled phase rotations have a runtime which scales like $\mathcal{O}(r^2)$, which is slower than all other methods discussed in this manuscript. However, it achieves this by using the minimum of only $2r$ qubits. The QFT adder can be mapped onto a Toffoli based adder reducing the runtime to $\mathcal{O}(r)$ [89]. The runtimes and memory requirements for all those methods are gathered in Table 4.1.1.

Furthermore, starting from the basic in-place adder ADD in Fig. 4.1.2, we can construct a modified version that adds a multiple or a fraction of one of the two summands to the other as long as the factor is a power of 2:

$$\text{ADD}^{2^{\pm k}} |a\rangle |b\rangle = |a\rangle |2^{\pm k} a + b\rangle. \quad (4.4)$$

For this we shift the significance of the bits of the two numbers before adding them. In the case of the addition of a multiple of a , this means we reduce ADD to an $r - k$ qubit version and apply it to the $r - k$ least significant qubits of the first register and to the $r - k$ most significant qubits of the second register (see Fig. 4.1.3a). This process ignores the first k digits of a . Hence, those digits should be zero to prevent information loss. This matches our overflow condition for positive numbers, requiring $|a| \ll 2^{r-p}$. For the addition of the fraction $2^{-k} a$ we add the $r - k$ most significant bits of $|a\rangle$ to $|b\rangle$. Here, the k least-significant digits of a can be non-zero as their contribution vanishes due to the finite precision with r qubits. Both methods result in a shift of k digits between the two binary numbers and with it either in the sum $2^k a + b$ or $2^{-k} a + b$. The additional factor will come in handy for the multiplication. This routine works for arbitrary p as long as $a \geq 0$.

4.2 From adder to polynomial

We mentioned before that addition can be taken as the basis of every other arithmetic operation. In this section, we show this by constructing a multiplier from a generic adder, and compute a

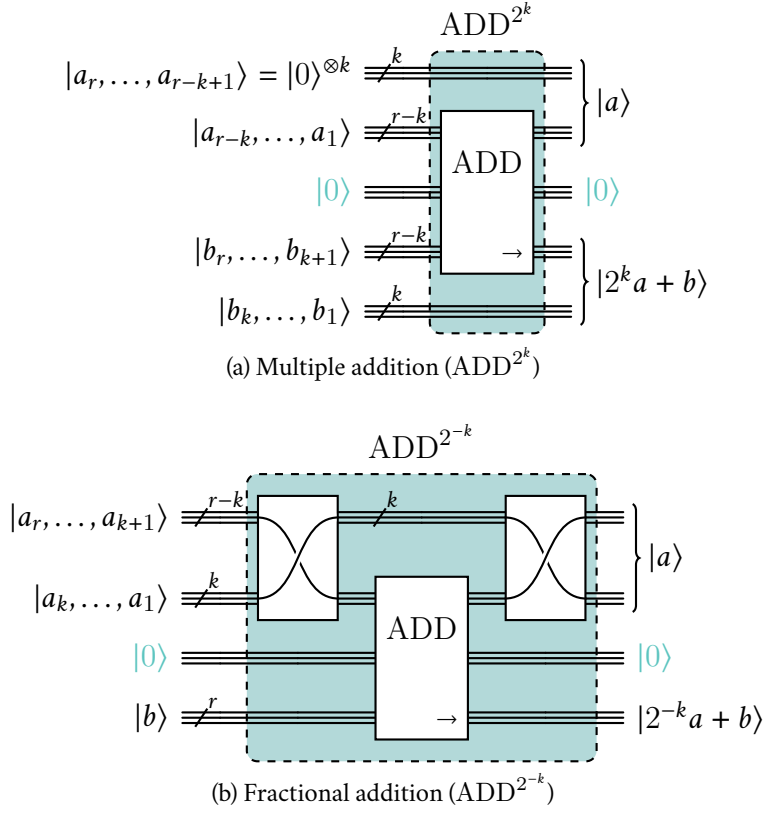


Figure 4.1.3: Modified versions of a quantum adder (ADD^{2^k}), that allow us to add multiple of a if the multiplier is a power of 2. For $k > 0$ (a), the method requires the k most significant qubits of $|a\rangle$ to be $|0\rangle$ as it neglects them. The remaining $r - k$ digits are added to the $r - k$ most significant digits of b . For $k < 0$ (b), it adds the $r - k$ most significant bits to $|b\rangle$.

signomial function (cf. Eq. (5.10)) from all other primitives. The described method is just one way of doing this and shall give a general understanding how one can achieve all this operations starting with an arbitrary adder.

4.2.1 Multiplier

The next routine we consider is the quantum multiplier MUL, which adds the product of two numbers (e.g. a and b) to a target register

$$\text{MUL } |a, b\rangle |z\rangle = |a, b\rangle |z + ab\rangle. \quad (4.5)$$

Any product can be deconstructed into a relative sign and a sum requiring only addition

$$ab = \text{sgn}(ab) \sum_{k=0}^{r-2} 2^{k-p} |a|_{k+1} |b|. \quad (4.6)$$

Here, $|a|_{k+1}$ represent the digits of the fixed-point representation of $|a|$. They control if the summand $2^{k-p}|b|$ contributes to the sum or not. Hence, we compute the product in three steps (see Fig. 4.2.1). First, we convert $|a\rangle$ (and $|b\rangle$) from their two's complement form into the sign-magnitude form $|\text{sgn}_b(a), |a|_r, \dots, |a|_1\rangle$. This allows us to access the sign and the magnitude directly. The transformation consists of two parts, a subtraction qcSUB_1 and a $\text{NOT}^{\otimes r}$ gate, both controlled by $|a_{r+1}\rangle$ and applied to $|a_r, \dots, a_1\rangle$. We provide a simple example for clarity ($r = 2$, cf. Fig. 4.1.1)

$$b = -3 \rightarrow |101\rangle \xrightarrow{\text{qcSUB}_1} |100\rangle \xrightarrow{\text{NOT}^{\otimes 2}} |111\rangle. \quad (4.7)$$

A broader discussion of converters between those two forms is given in Ref. [90]. After the transformation, the calculation of the relative sign $\text{sgn}(ab)$ is a matter of two NOT gates applied the most significant bit of the target register and controlled by $|a_{r+1}\rangle$ and $|b_{r+1}\rangle$ (cf. Fig. 4.2.1). Finally, it remains to compute the sum over $2^{k-p}|b|$ controlled by the single qubits $|a|_k$, which equals the product of the absolutes.

The sum is stored in the r least-significant bits of the target register. In the end, we need to re-transform all register into the two's complement form. It is important that we choose r and p appropriate to store any product ab , and later also $\sum_k c_k b^k$ for the polynomial, without overflow.

A qubit-shift-based modular multiplication of two positive integers, stored in qubits, was already described in Ref. [47]. We further extended the algorithm for negative fixed point numbers.

The multiplication requires one additional register of size $r + 1$ giving a total of n_{MUL} qubits, which containing $n_{\text{MUL},a}$ ancillas

$$n_{\text{MUL}} = 3(r + 1) + n_{\text{ADD},a}, \quad (4.8a)$$

$$n_{\text{MUL},a} = n_{\text{ADD},a}, \quad (4.8b)$$

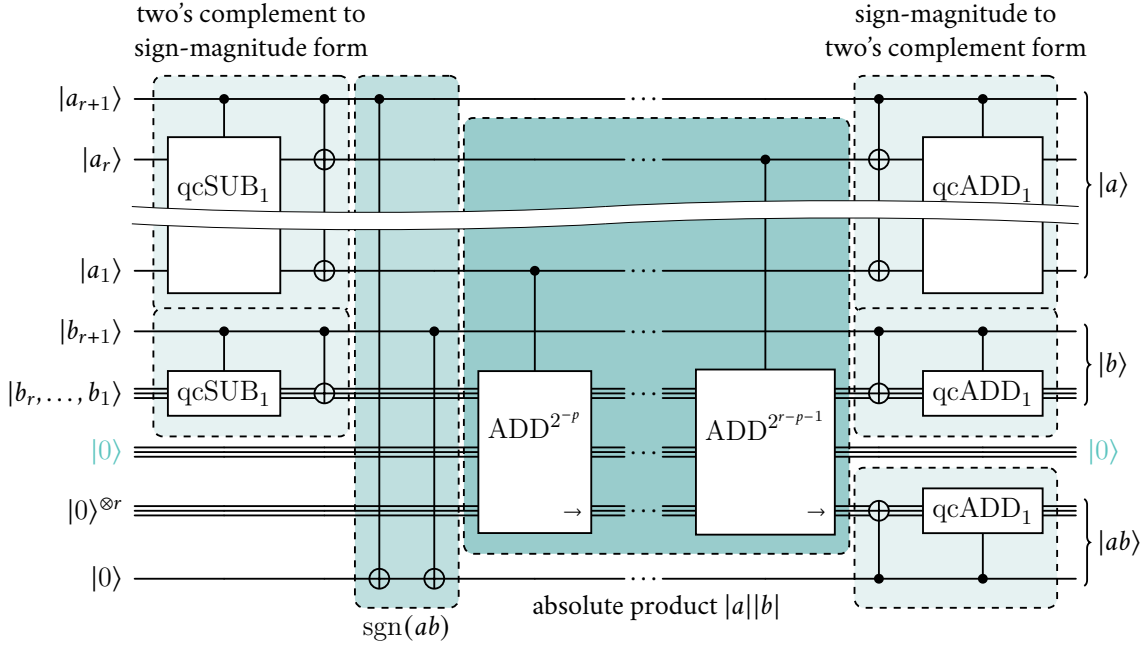


Figure 4.2.1: Quantum multiplier (MUL) based on the modified quantum adder (ADD^{2^k}) from Fig. 4.1.3. It starts with a transformation from the two's complement into the sign-magnitude form. Next we compute the relative sign, followed by the product of the magnitudes. The ancilla register in blue is just large enough to satisfy the ancilla requirements of the adders. Here, we neglect that the adders and subtractors in the form transformations may also require ancilla qubits.

Here, $n_{\text{ADD},a}$ is the adder specific ancilla requirement. The runtime of this routine is

$$t_{\text{MUL}} = 2t_{\text{qcADD}} + (2r + 2)t_{\text{CNOT}} + \sum_{k=0}^{r-1} t_{\text{ADD}^{2^{k-p}}} = \mathcal{O}(rt_{\text{ADD}}), \quad (4.9)$$

where t_{ADD} is the runtime of the chosen r qubit adder (cf. Table 4.1.1). t_{CNOT} , t_{qcADD} , and $t_{\text{ADD}^{2^{k-p}}}$ are the runtimes of one CNOT gate, qcADD_1 , and the modified adder $\text{ADD}^{2^{\pm k}}$ respectively. Here, we use that the adder dominate the runtime and all modified adders require less runtime than the basic ADD.

4.2.2 Horner's scheme based polynomial

With a working routine for the multiplication of negative fixed-point numbers we can go on and compute polynomials. Here we will follow Horner's scheme, similar to [91], which decomposes polynomials efficiently by minimizing the total number of multiplications.

$$\text{poly}(b) = \sum_{k=0}^K c_k b^k = c_0 + b(c_1 + b(c_2 + \dots + b(c_{K-1} + bc_K) \dots)). \quad (4.10)$$

This routine becomes ineffective for polynomials with only few non-zero coefficients c_k . However, for our case, the expansion of the arccosine of a square root (cf. Eq. (5.10)), only half of all coefficients are zero.

The desired quantum operation POLY shall read a number b stored in a register and return $\text{poly}(b)$ in another register

$$\text{POLY } |b\rangle |0\rangle = |b\rangle |\text{poly}(b)\rangle. \quad (4.11)$$

The full implementation of POLY is described in Fig. 4.2.2 It starts with encoding the coefficients $\{c_0, \dots, c_{K-1}\}$ in K quantum registers next to the input register

$$|b, c_{K-1}, c_{K-2}, \dots, c_0\rangle. \quad (4.12)$$

Next, it computes the most inner part of Eq. (4.10) by computing the product bc_K with the quantum-classical multiplier qcMUL and ads it to the second register. Now, we have

$$|b, c_{K-1} + bc_K, c_{K-2}, \dots, c_0\rangle. \quad (4.13)$$

It follows a series of $K - 1$ iterations in which the previous result is multiplied with b before added to the next register. This yields in

$$|b, c_{K-1} + bc_K, c_{K-2} + b(c_{K-1} + bc_K), \dots, \text{poly}(b)\rangle. \quad (4.14)$$

At last, it remains to uncompute the intermediate results with the Hermitian conjugate of all previous gates not applied to the target register. In this way, we get the final state

$$|b, 0, 0, \dots, \text{poly}(b)\rangle. \quad (4.15)$$

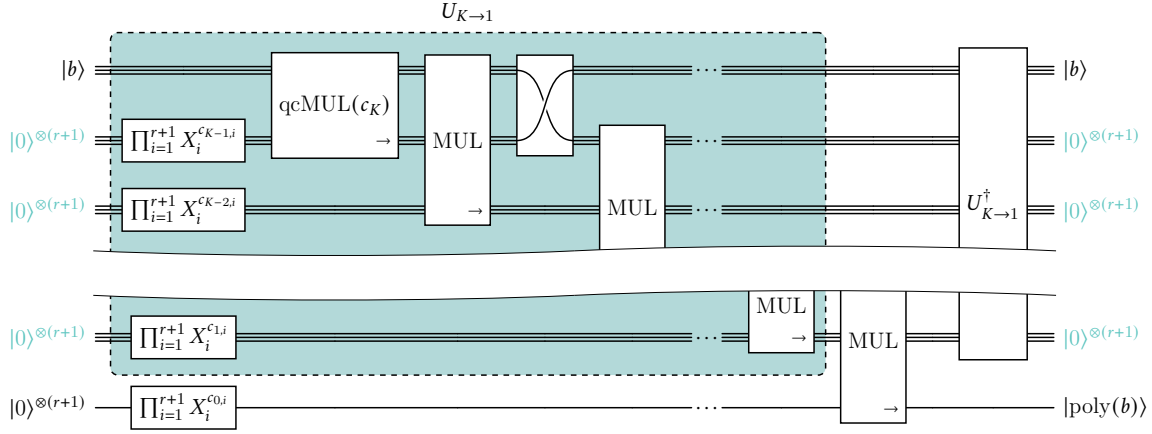


Figure 4.2.2: Quantum operation for the computation of polynomials (POLY) based on Horner's scheme. This requires both a quantum-quantum multiplier (MUL) and a quantum-classical multiplier (qcMUL). The ancilla qubits used by them are hidden here for illustrative reasons. The last gate represents the Hermitian conjugate $U_{K \rightarrow 1}^\dagger$ of all gates in the blue box. It reinitializes the ancilla registers and returns the input register to $|b\rangle$.

This routine requires a total n_{POLY} qubits with n_{POLY} ancillas

$$n_{\text{POLY}} = (K + 1)(r + 1) + n_{\text{ADD,a}}, \quad (4.16a)$$

$$n_{\text{POLY,a}} = (K - 1)(r + 1) + n_{\text{ADD,a}}. \quad (4.16b)$$

The runtime is dominated by the multipliers

$$t_{\text{POLY}} = \mathcal{O}(K t_{\text{MUL}}). \quad (4.17)$$

Assuming non-optimal but memory efficient carry-ripple adder based multiplication we get $t_{\text{POLY}} = \mathcal{O}(K r^2)$.

In Section 4.A we describe a more general routine for the computation of signomial functions that can surpass Horner's scheme if the total number of non-zero terms J is small in comparison to the maximum exponent K .

4.3 Square root with Newton-Raphson method

The computation of the square root is closely related to division methods. Here, as well, we can learn from classical algorithms. Starting with the slow division, restoring and non-restoring versions of division have been developed for quantum devices [92, 93]. On the other hand, classical fast division methods exist such as the Goldschmidt division and the Newton-Raphson division. The quantum version of Goldschmidt division is described in Ref. [93]. In this section, we focus on the Newton-Raphson division that was already described in Refs. [91, 94], and

propose an analog routine for the reciprocal and standard square root. Furthermore, we propose an alternative version of the standard Newton-Raphson division, that reduces the memory requirements by reinitializing all ancilla registers in Section 4.C.

The Newton-Raphson method allows us to find the root of a function f in an iterative process. For this we approximate f with its tangent at a given estimation x_k and determine the root x_{k+1} (i.e. $f(x_{k+1}) = 0$) of the tangent

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \quad (4.18)$$

This becomes the new estimation for the root. For most functions, the new estimation is closer to the exact root. One can repeat this process multiple times to reduce the error. The convergence rate of this method depends on f and the initial estimation x_0 .

4.3.1 (Reciprocal) square root

The Newton-Raphson method can be used to compute the reciprocal square root $S^{-1/2}$ with S being the input value. It can be further used for the computation of the regular square root $S^{1/2}$ by multiplying its result with S . We need a function for which the root fulfills $x = S^{-1/2}$

$$f_S(x) = \frac{1}{x^2} - S, \quad (4.19)$$

and accordingly

$$x_{k+1} = \frac{1}{2}x_k(3 - Sx_k^2). \quad (4.20)$$

Here, we can also define an error $\varepsilon_k = |1 - \sqrt{S}x_k|$ that changes like

$$\varepsilon_{k+1} = |1 - \sqrt{S}x_{k+1}| = \left|1 + \frac{1}{2}\sqrt{S}x_k \left(1 - \sqrt{S}x_k\right)^2\right| = \left|1 + \frac{1}{2}\sqrt{S}x_k\right| \varepsilon_k^2. \quad (4.21)$$

This converges with quadratic order, if $\varepsilon_{k+1}/\varepsilon_k < 1$, and $\varepsilon_k < 1$ which is equivalent to

$$\sqrt{S}x_k \in \left(\frac{-\sqrt{17}-1}{2}, -1\right) \cup \left(0, \frac{\sqrt{17}-1}{2}\right), \quad (4.22)$$

and

$$\sqrt{S}x_k \in (0, 2), \quad (4.23)$$

respectively. Hence, the method converges as long as $\sqrt{S}x_0 \in (0, 1.56)$. Due to the quadratic convergence L scales logarithmic with the desired number of correct digits

$$L = \mathcal{O}(\log_2 r). \quad (4.24)$$

The exact runtime relies on a good first estimation x_0 . A safe way to guarantee convergence is to always choose x_0 close to 0. Alternatively, one could use a lookup table that selects the

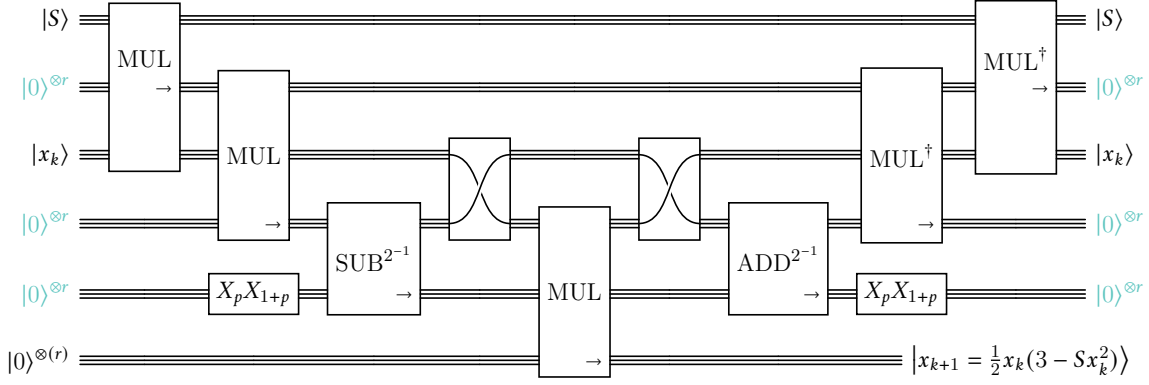


Figure 4.3.1: Circuit for one iteration step of the quantum Newton-Raphson method tailored to find $S^{-1/2}$. Ancilla qubits required by MUL, $\text{ADD}^{2^{-1}}$, and $\text{SUB}^{2^{-1}}$ are hidden. Here, $|S\rangle$ could be a superposition of states allowing parallel computing.

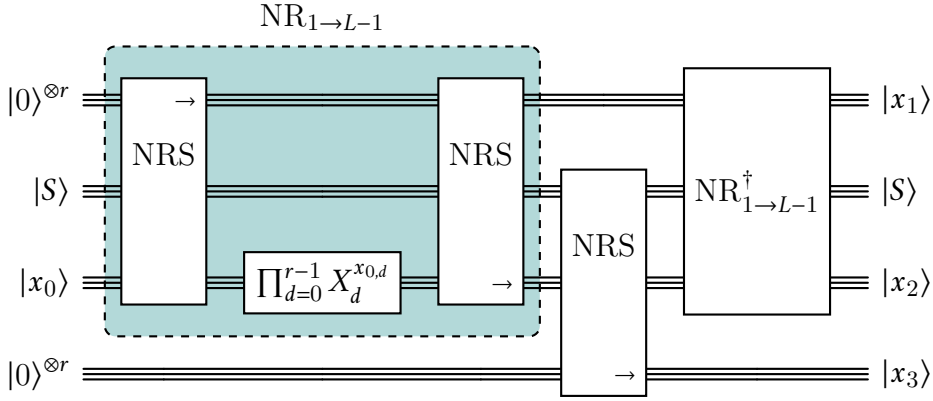


Figure 4.3.2: Full quantum Newton-Raphson method. The Newton-Raphson steps (NRS) are described in Fig. 4.C.1.

initial point depending on the range of the number S [94, 95], at the price of adding controlled operations.

A quantum implementation for this was previously briefly proposed in ref. [91]. We show a detailed implementation for one Newton-Raphson iteration illustrated in Fig. 4.3.1. It starts with computing Sx_k^2 in two iterations of MUL. After this, we subtract it from 1.5, that we encoded, using two NOT gates, in an ancilla register. Here, we use a subtractor based on the modified adder illustrated in Fig. 4.1.3b. This allows us to subtract only half of Sx_k^2 . It follows a final multiplication with x_k yielding $x_{k+1} = x_k(1.5 - 0.5Sx_k^2)$. Those steps are to repeat L times as illustrated in Fig. 4.3.2 for $L = 3$. The initial estimation x_0 is known, which allows us to uncompute and reuse its register. This reduces the total memory requirements by one register. The remaining registers with the intermediate results $\{x_1, \dots, x_{L-1}\}$ have to keep their information until the very end before we uncompute them all together. For the square root one multiplies x_L by S and stores the result in an additional register.

The memory requirements of one step are

$$n_{\text{SQRT-step}} = 6r + n_{\text{ADD,a}}, \quad (4.25a)$$

$$n_{\text{SQRT-step,a}} = 3r + n_{\text{ADD,a}}. \quad (4.25b)$$

Here, $n_{\text{SQRT-step}}$ is the total qubit number and $n_{\text{SQRT-step,a}}$ only the ancillas. We assumed r -qubit register without sign qubits to prevent complex results ($S < 0 \rightarrow \sqrt{S} \notin \mathbb{R}$). The whole Newton-Raphson method for the square root, with L iterations, requires

$$n_{\text{SQRT}} = (L + 4)r + n_{\text{ADD,a}} + r, \quad (4.26a)$$

$$n_{\text{SQRT,a}} = (L + 2)r + n_{\text{ADD,a}} + r. \quad (4.26b)$$

qubits in total and in ancillas respectively. The first and second Newton-Raphson step NRS requires $n_{\text{SQRT-step}}$ qubits, and after this we need to increase the qubit number by r for the target of each additional Newton-Raphson step. The last r qubits are only required for the product of $S^{-1/2}$ and S . The runtime of the full computation is

$$t_{\text{SQRT}} = (2L - 1)(5t_{\text{MUL}} + 2t_{\text{ADD}}) + t_{\text{MUL}} = \mathcal{O}(Lt_{\text{MUL}}). \quad (4.27)$$

Here, we omit the runtimes of the encoding of x_0 and the SWAP gates due to their small impact next to t_{MUL} . The additional t_{MUL} is required for the final product $S^{-1/2}S$, but is not changing the overall scaling. We do not replace L using Eq. (4.24), because its optimum depends on the difference between the desired and initial number of correct digits and with it usually performs better than $\log_2 r$. At last, one has to reinitialize the registers storing intermediate. This is neither changing the qubit requirements nor the scaling of the total runtime which is why we omit further details.

Appendix

4.A Fixed-point exponentiation

In this appendix, we describe how to continue from the fixed-point multiplier achieving and exponentiation allowing fractional exponents and with it how to compute signomial functions in a more general way than described in Section 4.2.

4.A.1 Exponentiator

Following the multiplier, the next operation is the quantum exponentiation EXP

$$\text{EXP } |a, b\rangle |0\rangle = |a, b\rangle |b^a\rangle. \quad (4.28)$$

Here, we assume $a, b > 0$ to prevent complex numbers if $a < 1$. Therefore, we require only r qubits for $|b\rangle$ and $|a\rangle$ each (no sign qubit needed). Similarly to Eq. (4.6), the exponentiation can be decomposed into a product of multiples of b

$$b^a = b^{\left(\sum_{k=0}^{r-1} 2^{k-p} a_{k+1}\right)} = \prod_{k=0}^{r-1} b^{(2^{k-p} a_{k+1})}. \quad (4.29)$$

The digit a_{k+1} controls if the multiplier $b^{2^{k-p}}$ is part of the product or not. In this case, we do not have a modified version of MUL that increases the input b by the desired exponent 2^{k-p} . This means that we need another quantum routine that computes the square and square root of the input value b . The full quantum exponentiator works as follows (see Fig. 4.A.1). We start by reducing b to $b^{2^{-p}}$ by applying the Hermitian conjugate of an in-place square operation $\text{inSQ } p$

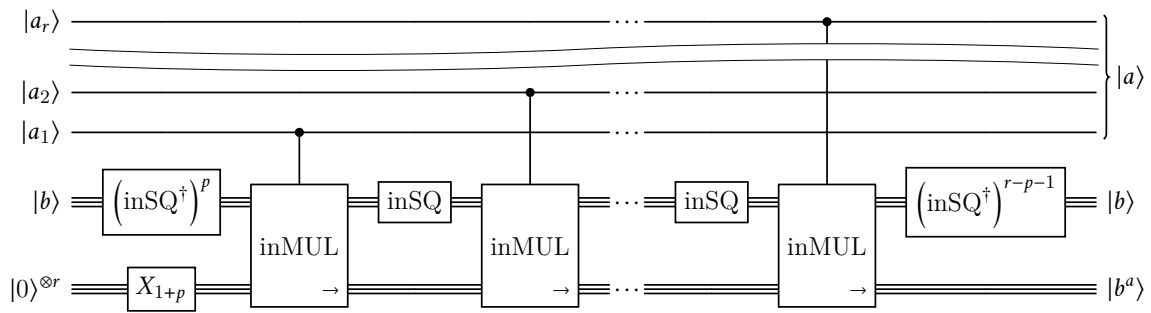


Figure 4.A.1: Quantum exponentiation (EXP) based on an arbitrary in-place quantum multiplier (inMUL) and an in-place square (inSQ). The $\text{CNOT}^{\otimes r}$ denotes a series of r parallel CNOT gates, with control on the qubits in $|b\rangle$ and applied to the qubits in the bottom register. The ancilla qubits for inMUL and inSQ are hidden.

times. Parallel, we encode a 1 in the target register by using a NOT gate X applied to its $1 + p$ -th least significant qubit. Next, the first contribution $b^{2^{-p}}$ for $k = 0$ can be multiplied to the target register with an in-place multiplier controlled by $|a_1\rangle$. After this, we square $b^{2^{-p}}$ once with inSQ achieving $|b^{2^{1-p}}\rangle$. Another in-place multiplier, controlled by $|a_2\rangle$, multiplies the $b^{2^{1-p}}$ to the value in the target register. The application of the square routine followed by a controlled multiplier will be repeated another $r - 2$ times. It is important that both the routine for the square and the multiplier are in-place routines. Otherwise, we would require $O(r^2)$ additional qubits to store intermediate products [47]. In Section 4.B we describe how to construct inSQ and inMUL. This results in the state $|a, b^{2^{r-p-1}}, b^a\rangle$. The last step is to return the second register to $|b\rangle$. This can be achieved by applying inSQ^\dagger $r - p - 1$ times.

One can modify the given exponentiator for negative b and integer a . The routine requires only the addition of one Toffoli gate, applied to the sign qubit of the target register, at the end, while controlled by the sign qubit $|b_r\rangle$ and the smallest integer qubit $|a_{p+1}\rangle$. The first p inSQs would cancel with the $(\text{inSQ}^\dagger)^p$ due to $|a_p, \dots, a_1\rangle = |0\rangle^{\otimes p}$.

Previous quantum-quantum exponentiators work similar to the routine described in this manuscript (cf. [47]). However, we replaced out-of-place multiplier with in-place ones, reducing the memory requirements. Furthermore, we describe exponentiation with fixed-point numbers allowing for non-integer exponents, which is useful for certain expansions (see Section 5.2.2).

The memory requirements are dominated by the n_{inSQ} qubits of the in-place square operation (see Sections 4.B and 4.3). In addition, EXP needs $2r$ qubits. Thus, for the total memory requirements n_{EXP} of the exponentiator and its ancilla requirements $n_{\text{EXP,a}}$ we have

$$n_{\text{EXP}} = (L + 7)r + n_{\text{ADD,a}} \quad (4.30a)$$

$$n_{\text{EXP,a}} = (L + 4)r + n_{\text{ADD,a}}. \quad (4.30b)$$

Here, L is the number of iterations required for inSQ (cf. Sections 4.B and 4.3). The runtime of the exponentiation consists of the contribution from the $2(r - 1)$ repetitions of the inSQ and inSQ^\dagger , and the r in-place multiplications. This gives a runtime t_{EXP} for the exponentiation of

$$t_{\text{EXP}} = 2(r - 1)t_{\text{inSQ}} + rt_{\text{inMUL}} = O(Lrt_{\text{MUL}}), \quad (4.31)$$

where t_{inSQ} , and t_{inMUL} are the runtimes of the in-place square inSQ, and the in-place multiplier inMUL respectively. For the second equation, we used that inSQ and inMUL contain both $O(L)$ MULs (cf. Sections 4.B and 4.3).

Quantum exponentiators are required for Shor's algorithm [46], where the base b is known and only the exponent a requires storage in a quantum register. For this specific purpose, two quantum-classical exponentiator for integers, based a carry-ripple adder, were already proposed in Refs. [54, 85]. They achieve an $O(r^3)$ runtime with $7r + 1$ and $5r + 1$ qubits respectively (cf. Table 4.A.1). Quantum-classical exponentiators, based on either the conditional sum or the carry-lookahead adder, both reach an $O(r \log_2^2 r)$ runtime with $\approx 1.9r^2$ qubits [55]. They achieve this speed-up by parallelizing the multiplier in blocks executed in a cascade-like structure. A quantum-classical exponentiator based on QFT adders achieves an $O(r^3)$ runtime in only $2r + 3$ qubits. This is described in Ref. [96]. At last, we also refer to Ref. [97],

Table 4.A.1: Quantum-classical exponentiation for r -digit exponents (required in Shor's algorithm)

Method	Runtime t_{EXP}	Memory n_{EXP} in qubits
Carry-ripple [85]	$\mathcal{O}(r^3)$	$5r + 3$
Conditional sum [55]	$\mathcal{O}(r \log_2^2 r)$	$\approx 1.9r^2$
Carry-lookahead [55]	$\mathcal{O}(r \log_2^2 r)$	$\approx 1.9r^2$
Fourier transform [96]	$\mathcal{O}(r^3)$	$2r + 3$
Fourier transform [97]	$\mathcal{O}(r^2)$	$9r + 2$

which contains a great overview of existing methods. They, further propose a QFT based method that requires $9r + 2$ qubits and has a runtime of $\mathcal{O}(r^2)$. This is possible by merging classical-controlled rotation gates. These results are summarized in Table 4.A.1.

4.A.2 Operation for signomial functions

At last, we propose a quantum routine for the computation of signomial functions of the form

$$\text{sig}(b) = \sum_{k=0}^K c_k b^{k/\kappa}, \quad (4.32)$$

with κ an integer and, by definition of a signomial, $b > 0$. Normally, signomial functions allow for arbitrary real exponents, but fixed-point arithmetic limits this to rational exponent k/κ . The corresponding quantum routine SIG is defined by

$$\text{SIG } |b\rangle |0\rangle = |b\rangle |\text{sig}(b)\rangle. \quad (4.33)$$

Here, we assume that we know K , all c_k , and κ in advance. This allows us to use a quantum-classical exponentiator qcEXP in which b remains in a quantum register but the exponent a is stored classically², i.e. only operations controlled by a classical bit in the state $|a_k\rangle_c = |1\rangle_c$ remain. This simplifies the quantum routine in width and depth. Similarly, we can use a quantum-classical multiplier qcMUL for the multiplication with the classically stored weights c_k . The full routine for the computation of the signomial starts with the initialization of c_0 in the target register after which we use a quantum-classical out-of-place multiplier qcMUL(c_1) to multiply b , stored in one quantum register, with c_1 , stored in a classical register, and add the product to the target register (see Fig. 4.A.2). After this a series of J similar iterations follows, where J is the number of remaining summands with non-zero weight $c_k, \forall k \geq 2$. They all start with increasing the power of b with the quantum-classical exponentiator qcEXP(k/κ) to $b^{k/\kappa}$, followed by a quantum-classical multiplier with the corresponding weight c_k adding their product $c_k b^{k/\kappa}$ to the target register. The last part of one iteration is to return $|b^{k/\kappa}\rangle \rightarrow |b\rangle$ with

²Not to confuse with the quantum-classical exponentiator in Table 4.A.1, where a is stored in a quantum register and b classically.

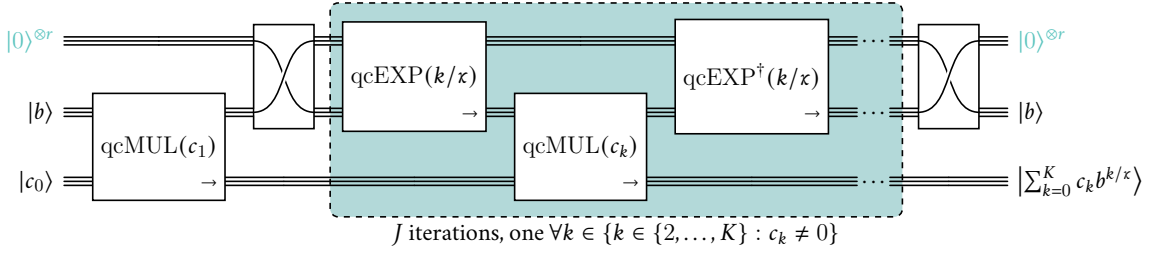


Figure 4.A.2: Quantum routine (SIG) for signomial functions based on quantum-classical versions of the multiplier (qcMUL) and exponentiation (qcEXP). The arguments in parentheses are the classical multipliers and exponents respectively. We hide the ancilla qubits required in qcMUL and qcEXP.

the inverse $\text{qcEXP}^\dagger(k/\chi)$. The reinitialization of $|b\rangle$ at the end of each step can be combined with the following exponentiation going directly to the next power (e.g. $|b^2\rangle \rightarrow |b^5\rangle$) by slightly modifying qcEXP. Even though the combined reinitialization reduces the runtime of the routine, it is not changing the overall scaling which is why we consider the previously described routine in the following.

The routine for a signomial with J non-zero weights consists of $O(J)$ quantum-classical multiplier and $O(J)$ quantum-classical exponentiator in series yielding a runtime of

$$t_{\text{SIG}} = O(J(t_{\text{qcMUL}} + t_{\text{qcEXP}})). \quad (4.34)$$

Here, t_{qcMUL} and t_{qcEXP} are the runtimes of the used multiplier and exponentiator respectively. The advantage of quantum-classical multiplication and exponentiation is that the number of internal steps is at most \bar{r} instead of r , where \bar{r} is the number of bits required to represent the maximum factor or exponent respectively. This reduces the runtime of qcMUL and qcEXP to

$$t_{\text{qcMUL}} = O(t_{\text{ADD}} \log_2(|c|_{\max})), \quad (4.35a)$$

$$t_{\text{qcEXP}} = O(t_{\text{MUL}} L \log_2(K\chi)), \quad (4.35b)$$

where $\log_2 |c|_{\max} = \max_k \log_2[\lfloor |c_k| \rfloor (|c_k| - \lfloor |c_k| \rfloor)]$ is the maximum number of digits required for the weights c_k occurring in Eq. (4.32). Analogously, $\log_2(K\chi)$ digits are required to represent the exponent K/χ . This reduces the runtime (Eq. (4.34)) to

$$t_{\text{SIG}} = O(JL \log_2(K\chi) t_{\text{MUL}}). \quad (4.36)$$

Here, we assume $\log_2 |c|_{\max} < r$ and with it $t_{\text{MUL}} > \log_2 |c|_{\max} t_{\text{ADD}}$, i.e. $t_{\text{qcMUL}} < t_{\text{qcEXP}}$. The computation of a signomial function requires a total of n_{SIG} qubits containing $n_{\text{SIG,a}}$ ancillas

$$n_{\text{SIG}} = r + n_{\text{qcEXP}} = (L + 7)r + n_{\text{ADD,a}}, \quad (4.37a)$$

$$n_{\text{SIG,a}} = (L + 5)r + n_{\text{ADD,a}}. \quad (4.37b)$$

Here, $n_{\text{qcEXP}} = n_{\text{EXP}} - r$ is the number of qubits required for the quantum-classical exponentiator that is not controlled by a second quantum register.

One can modify this routine for multi-variable signomials $\text{sig}(a, b, \dots)$ by combining quantum-classical exponentiators qcEXP that sample from different quantum registers with the various bases $(|a\rangle, |b\rangle, \dots)$. The different powers of a, b, \dots can be combined with quantum-quantum in-place multipliers inMUL before they will be added to the target register with a qcMUL . This is not affecting the runtime, but increases the memory requirements by Mr qubits, where M is the number of variables in the signomial. They consist of $M - 1$ additional input registers and 1 ancilla register to store intermediate products.

4.B In-place arithmetic

Some quantum implementations of arithmetic operations like the multiplier MUL or exponentiator EXP are naturally out-of-place. Based on numbers, encoded in the basis of two input register, they prepare another number in a third register. This yields several wasted quantum registers if one stacks multiple of those operations in series. We can prevent this by using in-place versions instead, that reuse one of the input register to store the result. The structure of in-place operations (based on their out-of-place versions) is always the same and can be split into three parts [54] as shown in the following for the computation of an invertible function $f(a)$:

1. Computation of $f(a)$ with the out-of-place arithmetic operator U_f

$$U_f |a, 0\rangle = |a, f(a)\rangle. \quad (4.38)$$

2. Swap of the target and input register

$$\text{SWAP } |a, f(a)\rangle = |f(a), a\rangle. \quad (4.39)$$

3. Annihilation of the input a with the Hermitian conjugate $U_{f^{-1}}^\dagger$ for the inverse f^{-1} of the arithmetic operation

$$U_{f^{-1}}^\dagger |f(a), a\rangle = |f(a), 0\rangle. \quad (4.40)$$

In this appendix, we describe the two in-place operations inSQ and inMUL , occurring in the exponentiator in Section 4.2.

4.B.1 Square (inSQ)

The quantum implementation for the in-place square operation inSQ is shown in Fig. 4.B.1. It starts with the computation of the square. For this it copies the input value a to an ancilla register by utilizing a column of CNOT gates, before it uses both registers as input for the out-of-place multiplication MUL . The product a^2 is stored in another ancilla register. The first ancilla register, storing $|a\rangle$ can now be reinitialized. For the second step, we swap the two register $|a\rangle$ and $|a^2\rangle$. At last, we need the inverse of the original arithmetic operation which

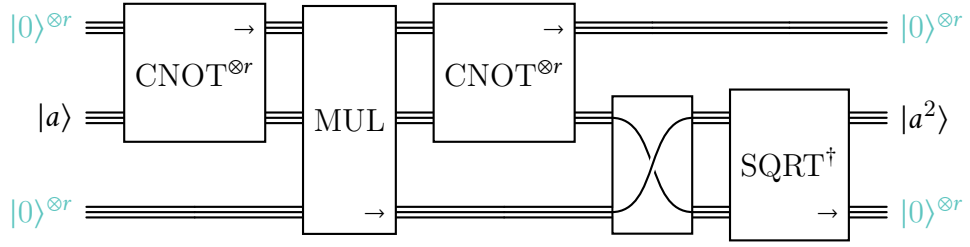


Figure 4.B.1: Quantum in-place square (inSQ) based on a quantum multiplier (MUL) and a square root (SQRT) routine. The $\text{CNOT}^{\otimes r}$ represents r parallel CNOT gates. The ancilla qubits used in MUL and SQRT are hidden.

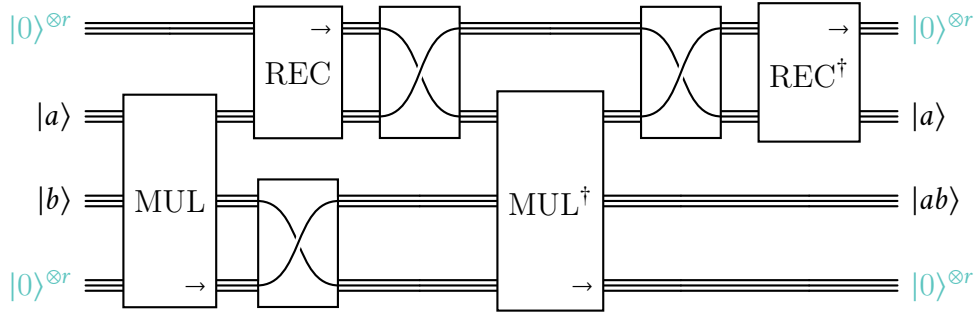


Figure 4.B.2: Quantum in-place multiplier (inMUL) based on an arbitrary quantum multiplier (MUL) and an operation (REC) that computes the reciprocal. The ancilla qubits necessary for MUL and REC are hidden.

is the square root. By using the register that stores $|a^2\rangle$, we can compute a without accessing the $|a\rangle$ register. The Hermitian conjugate of the corresponding quantum gate allows us to annihilate (instead of generate) a in the ancilla register.

The memory requirements consist of one r -qubit ancilla register next to the n_{SQRT} qubits necessary for SQRT (see Section 4.3). This adds up to

$$n_{\text{inSQ}} = (L + 3)r + n_{\text{ADD},a}, \quad (4.41a)$$

$$n_{\text{inSQ},a} = (L + 2)r + n_{\text{ADD},a}. \quad (4.41b)$$

$n_{\text{inSQ},a}$ denotes the ancilla qubits required for inSQ. The runtime t_{inSQ} is dominated by the runtime t_{SQRT} of SQRT

$$t_{\text{inSQ}} = \mathcal{O}(t_{\text{SQRT}}) = \mathcal{O}(Lt_{\text{MUL}}). \quad (4.42)$$

4.B.2 Multiplication (inMUL)

The in-place multiplication inMUL starts similar with the computation of the product ab via MUL storing it in an ancilla register (cf. Fig. 4.B.2). The second step is the swap of one of the

inputs ($|b\rangle$) with the target register storing $|ab\rangle$. The last step requires a quantum divider, which is equal to the multiplication of the reciprocal of the divisor with the dividend

$$\frac{\text{dividend}}{\text{divisor}} = \text{dividend} \times \text{divisor}^{-1}. \quad (4.43)$$

Hence, we start by computing the reciprocal of a with REC and store it in an ancilla register. Next, we multiply a^{-1} with the previous computed product ab resulting in b . The Hermitian conjugate of MUL annihilates b in the ancilla register. It remains to undo REC.

This routine requires a total memory of

$$n_{\text{inMUL}} = 2r + n_{\text{REC}} = (L + 5)r + n_{\text{ADD,a}} \quad (4.44a)$$

$$n_{\text{inMUL,a}} = (L + 3)r + n_{\text{ADD,a}}, \quad (4.44b)$$

with n_{inMUL} the total qubit number, $n_{\text{inMUL,a}}$ the ancillas. and n_{REC} is the memory requirement of REC (cf. Section 4.C). The runtime t_{inMUL} is dominated by the runtime t_{REC} of REC yielding

$$t_{\text{inMUL}} = O(t_{\text{MUL}} + t_{\text{REC}}) = O(Lt_{\text{MUL}}). \quad (4.45)$$

The in-place variants described here require additional operations (REC and SQRT). Those can be implemented using the Newton-Raphson method, that is described in Sections 4.3 and 4.C.

4.C Newton-Raphson reciprocal

Similarly to the square root in Section 4.3 we can use the Newton-Raphson method to compute the reciprocal. For this, we require a function with a root at $x = R^{-1}$, where R is the input value. A common choice is

$$f_R(x) = \frac{1}{x} - R, \quad (4.46)$$

which, from Eq. (4.18), gives

$$x_{k+1} = x_k (2 - Rx_k). \quad (4.47)$$

This converges with quadratic order towards R^{-1} , which we can show by defining an error $\varepsilon_k = |1 - Rx_k|$ and study how it changes with increasing k

$$\varepsilon_{k+1} = |1 - Rx_{k+1}| = (1 - Rx_k)^2 = \varepsilon_k^2. \quad (4.48)$$

With this, the number of correct digits in x_k doubles with every iteration, if $Rx_0 \in (0, 2)$ (cf. Refs. [95, 98]). For the initial estimation x_0 of the reciprocal we can rely on a method, that is based on finding the leading 1 in the binary form and moving it to the other side of the binary point as described in Ref. [91]. This gives an approximation that is good enough to require only a few Newton-Raphson iterations (2-4 in their case).

The corresponding quantum circuit consists of multiplication and subtraction (cf. Fig. 4.C.1). Each iterative step starts with the multiplication of x_k and R . The result gets subtracted from

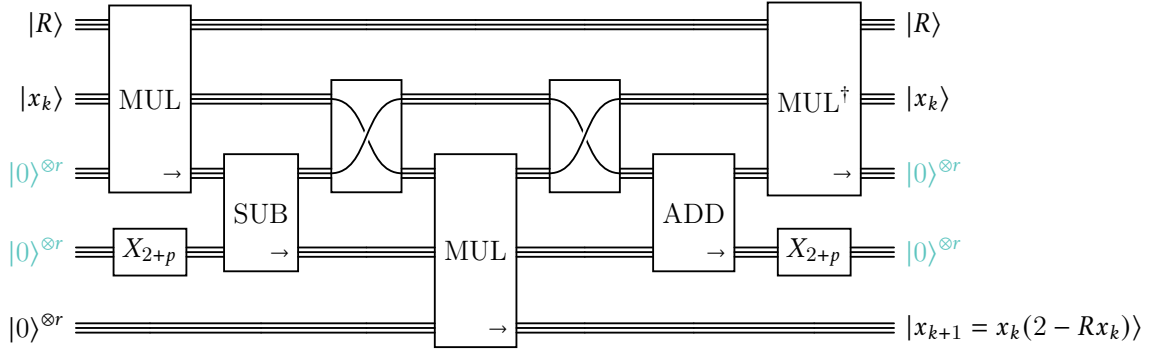


Figure 4.C.1: Circuit for one iteration step of the quantum Newton-Raphson reciprocal R^{-1} . We hide the ancilla qubits necessary for MUL, SUB and ADD.

another register in which we encoded the number 2 with a single not gate applied to the $(2 + p)$ -th least significant qubit, where p is the number of digits after the binary point. A multiplication of the difference $(2 - Rx_k)$ with the initial estimation x_k follows, giving our next estimation x_k stored in the lowest register of Fig. 4.C.1. It remains to uncompute the previous steps reinitializing the ancilla registers. This gives one step of the Newton-Raphson method. The full series is analogous to Fig. 4.3.2.

One iteration of the Newton-Raphson step for the reciprocal requires

$$n_{\text{REC-step}} = 5r + n_{\text{ADD,a}}, \quad (4.49a)$$

$$n_{\text{REC-step,a}} = 3r + n_{\text{ADD,a}}, \quad (4.49b)$$

qubits and the full circuit with L steps

$$n_{\text{REC}} = (L + 3)r + n_{\text{ADD,a}}, \quad (4.50a)$$

$$n_{\text{REC,a}} = (L + 1)r + n_{\text{ADD,a}}, \quad (4.50b)$$

qubits. The runtime of the full circuit is

$$t_{\text{REC}} = L(3t_{\text{MUL}} + 2t_{\text{ADD}} + 2t_{\text{SWAP}}) + 2t_{x_0} = \mathcal{O}(Lt_{\text{MUL}}). \quad (4.51)$$

Here, t_{SWAP} is the runtime of one (or r parallel) SWAP gates, which is negligible compared to t_{MUL} . The runtime t_{x_0} of the encoding of x_0 is also small compared with t_{MUL} .

4.D Quantum logic operators

For the oracle implementation we need to test boolean conditions (see Eq. (5.21)). In this appendix, we propose quantum implementations for two types of operations:

- Comparisons like *equal* ($=$) and *greater than* ($>$).
- Boolean logical operations like *conjunction* (\wedge) and *disjunction* (\vee).

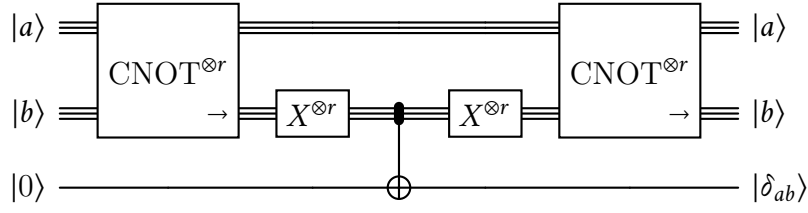


Figure 4.D.1: Quantum circuit for the Kronecker delta operation for two r -qubit registers $|a\rangle$ and $|b\rangle$. The $\text{CNOT}^{\otimes r}$ represents r parallel CNOT gates while the center gate is an r -qubit 0-controlled NOT gate with $r - 2$ hidden ancilla qubits.

4.D.1 Comparisons

In order to compare two binary numbers (a and b) stored in the computational basis of two quantum registers, we can make use of CNOT gates. For two registers of r qubits each (focusing on positive numbers, $a, b > 0$), we apply a series of r CNOT gates between qubit pairs of equal significance (see Fig. 4.D.1). If both numbers are equal this will fully annihilate any 1 in the target register. One application of NOT gates to all qubits of this register inverts all of them leaving 1s in all qubits if $a = b$. It follows an r -qubit controlled CNOT gate targeting an ancilla qubit. This triggers only if all control qubits are 1 and with it if $a = b$. Hence, we have the Kronecker delta δ_{ab} in the target register. It remains to restore the initial values a and b .

This routine requires a total of

$$n_{\delta} = 2r + 1 + n_{r\text{CNOT},a} = 3r - 1, \quad (4.52)$$

qubits. Here, $n_{r\text{CNOT},a} = r - 2$ is the ancilla number of the r -qubit controlled CNOT gate [99]. The runtime is equal to

$$t_{\delta} = 4 + t_{r\text{CNOT}} = \mathcal{O}(t_{\text{TOF}} \log_2 r) = \mathcal{O}(\log_2 r), \quad (4.53)$$

where we introduced the runtime of the r -qubit CNOT gate $t_{r\text{CNOT}}$ measured in Toffoli gates $t_{\text{TOF}} = \mathcal{O}(1)$ executed parallel in a cascade like structure..

The previous methods allow us to test if $a = b$, but what if we are interested in $a > b$ or $a < b$? For this we propose another routine based on ADD and SUB. We start by increasing the r -qubit register storing b by one qubit as the new most-significant bit, i.e., $|b\rangle \rightarrow |0, b\rangle$. Then we subtract a from b using an $r + 1$ qubit SUB applied to $|a\rangle |0, b\rangle$.

$$\text{SUB } |a\rangle |0, b\rangle = |a\rangle |b - a \bmod 2^{r+1}\rangle. \quad (4.54)$$

This is shown in Fig. 4.D.2. Here, it is important that subtraction in the two's complement form can lead to negative numbers indicated by a 1 in the most significant qubit, i.e. only if $a > b$, the most significant qubit changes from $|0\rangle \rightarrow |1\rangle$

$$|b - a \bmod 2^{r+1}\rangle = |\text{sgn}_b(b - a)\rangle |b - a \bmod 2^r\rangle. \quad (4.55)$$

At last, we need to restore b by applying an r qubit ADD to the two input registers only.

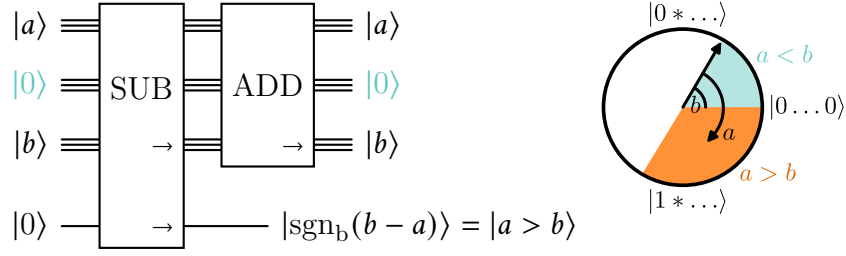


Figure 4.D.2: Quantum circuit for the *greater than* operation ($>$) on the left. On the right we illustrate how the subtraction can yield a bit flip in the leading qubit.

This routine requires

$$n_{>} = 2r + 1 + n_{\text{ADD},a} \quad (4.56)$$

qubits, where $n_{\text{ADD},a}$ is the number of qubits necessary for an $r + 1$ qubit adder. We need one qubit less because $|a\rangle$ is only an r -qubit register. The runtime is governed by the adder

$$t_{>} = O(t_{\text{ADD}}). \quad (4.57)$$

Both comparisons ($(=)$ and $(>)$) can be extended for negative numbers by adding one qubit to the registers and using the two's complement form.

4.D.2 Boolean logical operations

Eq. (5.21) requires the combination of multiple comparisons. This can be achieved with the conjunction (\wedge , *and*). The Toffoli gate is the natural quantum implementation of the conjunction as it inverts the information in the target qubit only if both control qubits $|A\rangle$ and $|B\rangle$ are $|1\rangle$ (*true*). Applied to an ancilla qubit in state $|0\rangle$, this yields in $|0\rangle$ (*false*) for $AB = 0$ and $|1\rangle$ (*true*) for $AB = 1$ (see Fig. 4.D.3a). Multiple Toffoli gates can be combined for more than two conditions (cf. r -qubit CNOT gate [99]).

Complex geometries require the combination of conditions with the disjunction (\vee , *or*). For this we use the logical not (\neg) and invert the two conditions A and B before we combine them with a Toffoli. This yields in $\neg A \wedge \neg B$, which is the opposite of what we desire. Another negation results in

$$\neg(\neg A \wedge \neg B) = A \vee B. \quad (4.58)$$

Translated in quantum operations we need to invert the input and output of a Toffoli gate with NOT gates (see Fig. 4.D.3b).

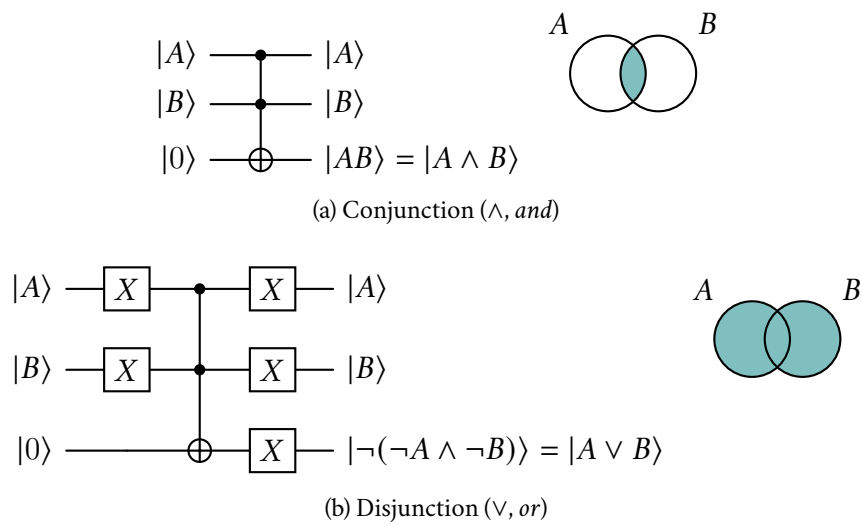


Figure 4.D.3: Quantum circuit and Venn diagrams for the conjunction (\wedge , *and*) and the disjunction (\vee , *or*) operation in (a) and (b) respectively. A and B are boolean variables being either 0 or 1 for false and true.

Quantum Phase Estimation as Eigenpair Problem Solver

The first of our key results is an end-to-end quantum algorithm that computes the local and non-local response functions of a system of coupled oscillators. We start by providing the basic intuition behind the algorithm for the case of the local response function $G_{uu}(\nu)$ in Eq. (3.13), while we refer to Section 5.1 for more technical details. In order to compute $G_{uu}(\nu)$, we need to determine the eigenvalues λ_j and the coefficients W_{uj}^2 . In our approach, each vertex $u \in \mathcal{V}$ is associated with a bitstring $|u\rangle$ of length n in the computational basis, where the number of qubits is $n = \lceil \log_2 N \rceil$. For simplicity, from now on we assume that N is a power of two, i.e., $N = 2^n$. Since, W is the orthogonal transformation that diagonalizes H , we can express the bitstring $|u\rangle$ as a linear combination of the eigenstates $|\lambda_j\rangle$ of H :

$$|u\rangle = \sum_{j=1}^N W_{uj} |\lambda_j\rangle. \quad (5.1)$$

Thus, we can interpret W_{uj} as weights in the previous decomposition. Eqs. (3.13) and (5.1) convey the core idea behind the algorithm. If we prepare the bitstring $|u\rangle$ and perform measurement in the eigenbasis $|\lambda_j\rangle$ of H we will get the eigenvalues λ_j with probability W_{uj}^2 . By repeating this procedure many times we can estimate the coefficients W_{uj}^2 via statistical averages and obtain an estimate for the response function $G_{uu}(\nu)$. Note that there is an implicit importance sampling in this procedure: the eigenstates that contribute the most to $G_{uu}(\nu)$ are also those whose eigenvalue will be measured more often. In order to implement the measurement in the eigenbasis we use QPE.

The full algorithm for the local response is described in detail in Section 5.1. In Section 5.2 we describe in detail how to implement the oracles necessary for the block encoding of H (cf. Sections 2.5.1 and 5.1). We further discuss necessary simplifications to maintain the efficiency of the oracles. A modification, necessary for the estimation of the non-local response, is proposed in Section 5.3.

Sections 5.1, 5.3 and 5.4 of this chapter have been published in the preprint “Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405.08694”. Section 5.2 has been shared in the preprint “Sven Danz, Tobias Stollenwerk, and Alessandro Ciani. “Quantum oracles for the finite element method”. In: *arXiv e-prints* (2025). arXiv: 2504.19827”.

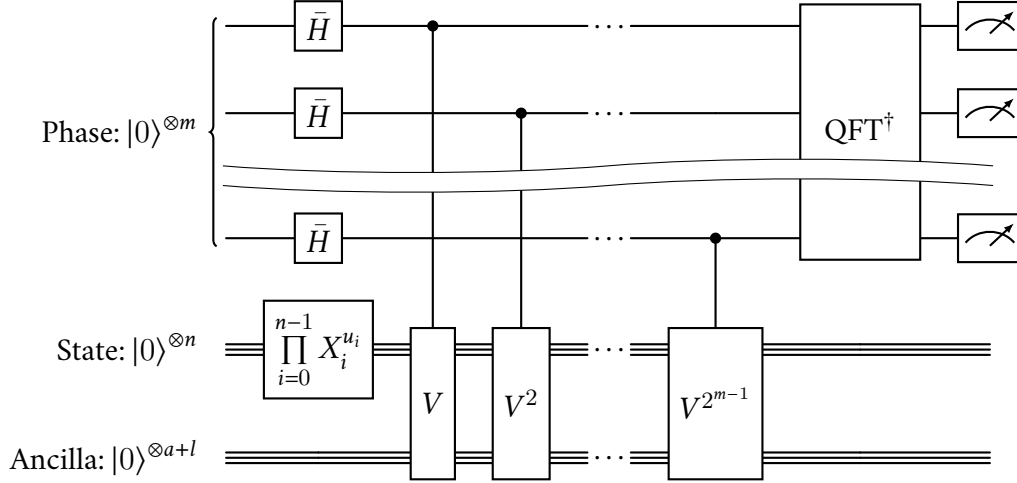


Figure 5.1.1: QPE circuit for the computation of the local response function. The state preparation is realized by a series of NOT-gates. QPE is performed using the walk operator V defined in Eq. (2.19) followed by the inverse quantum Fourier transform (QFT^\dagger) [12]. Sampling from the phase register and repeating many times allows us to estimate the relevant eigenvalues λ_j and the weights W_{uj}^2 in Eq. (3.13).

5.1 The algorithm

In this section, we detail how the local response function $G_{uu}(\nu)$ in Eq. (3.13) can be obtained using QPE. In particular, we describe the algorithm based on the standard QPE [12, 14], that makes use of the inverse quantum Fourier transform. A similar analysis can also be performed using other versions of QPE that use a single ancilla qubit for the phase register [13, 18, 48, 100] or coherent QPE with fewer qubits [101]. As we have seen in Section 3.1, in order to obtain the response function, we need to solve the eigenproblem associated with the Hermitian matrix in Eq. (3.4). The quantum eigenpair solver, we propose here, requires three main registers as shown in Fig. 5.1.1:

1. An m -qubit register that, at the end of the circuit, stores the information about the eigenphase;
2. An n -qubit register that represents the Hilbert space associated with the Hamiltonian H ;
3. An $(a + l)$ -qubit ancilla register, which includes the a ancillas for the block-encoding described in Section 2.5 as well as l ancilla qubits needed for the implementation of the oracles (see Section 2.5.2).

We start by preparing a state $|u\rangle$ associated with one of the vertices $u \in \mathcal{V}$ in the state register (cf. Fig. 5.1.1) by using a product of single-qubit NOT gates

$$\prod_{i=0}^{n-1} X_i^{u_i} |0\rangle^{\otimes n} = |u\rangle = \sum_{j=1}^N W_{uj} |\lambda_j\rangle, \quad (5.2)$$

where X_i denotes the NOT gate on qubit i . This is equivalent to an amplitude encoding of the weights W_{uj} in the basis of the eigenstates $|\lambda_j\rangle$ of H . The choice of u affects the set of weights W_{uj} and with it the probability to collapse into the corresponding eigenstate $|\lambda_j\rangle$ upon measurement in this basis.

Including the a ancilla qubits needed for the block-encoding, we can rewrite the state in Eq. (5.2) in the eigenbasis of the unitary V given in Eq. (2.23) as

$$|0\rangle^{\otimes a} |u\rangle = \frac{1}{\sqrt{2}} \sum_{j=1}^N W_{uj} \left(|\mu_+^{(j)}\rangle + |\mu_-^{(j)}\rangle \right). \quad (5.3)$$

After the execution of QPE with V as unitary operator controlled by the m qubits of the phase register [12], we get:

$$\xrightarrow{\text{QPE}} \frac{1}{\sqrt{2}} \sum_{j=1}^N W_{uj} \sum_{x=0}^{M-1} \left(a_{x+}^{(j)} |\mu_+^{(j)}\rangle + a_{x-}^{(j)} |\mu_-^{(j)}\rangle \right) |x\rangle, \quad (5.4)$$

where $M = 2^m$ and $a_{x\pm}^{(j)}$ is the amplitude applied by the QPE subroutine

$$a_{x\pm}^{(j)} = \frac{1}{M} \sum_{z=0}^{M-1} e^{\frac{2\pi iz}{M} (\pm\varphi_j - x)}, \quad (5.5)$$

with

$$\varphi_j = \frac{M}{2\pi} \arccos \left(\frac{\lambda_j}{s \|H\|_{\max}} \right). \quad (5.6)$$

The coefficients $a_{x\pm}^{(j)}$ are peaked around $x = \pm\varphi_j \bmod M$, respectively. This tells us that the probability $P(x)$ to measure x in the phase register will peak around $\pm\varphi_j \bmod M$ for all $j \in \{1, \dots, N\}$, and it is given by

$$P(x) = \sum_{j=1}^N \frac{W_{uj}^2}{2} \left(|a_{x+}^{(j)}|^2 + |a_{x-}^{(j)}|^2 \right), \quad (5.7)$$

where we used the orthonormality of $|\mu_{\pm}^{(j)}\rangle$.

After sampling multiple times from the phase register, we can obtain an estimate of the probabilities Eq. (5.7), from which we need to extract the eigenvalues λ_j and the weights W_{uj}^2 . Here, we assume that we have sufficient resolution to determine the peaks and their centers. Possible practical methods to determine the eigenvalues are discussed in Refs. [25, 100, 102], and could be tested in practice for our problem. The center $\tilde{\varphi}_j$ of the peak can be translated into the eigenvalues by inverting Eq. (5.6). The probability to sample from a certain eigenstate is proportional to the weight W_{uj}^2 of interest. The last can be estimated by adding the probabilities of the $2Q$ values closest to $\tilde{\varphi}_j$

$$W_{uj}^2 \approx 2 \sum_{q=-Q}^{Q-1} P(\lceil \varphi_j \rceil + q), \quad (5.8)$$

where, we assumed that $\tilde{\varphi}_j \in [\lfloor \varphi_j \rfloor, \lceil \varphi_j \rceil]$. The additional factor of 2 is due to the $1/2$ in Eq. (5.7). The necessary size of Q is further discussed in Section 6.1.2. The full routine is gathered in Algorithm 1.

In Section 5.3, we discuss a more general method which uses a modified Hadamard test to estimate the non-local response function $G_{uv}(\nu)$.

5.2 Oracle implementation

Many promising quantum algorithms which provide a quantum advantage over classical methods assume the efficient implementation of quantum oracle. In fact, the runtimes of these algorithms are often quantified in terms of the number of oracle queries, which establishes the *query complexity* of the algorithm. In contrast, the *gate complexity* of an algorithm quantifies the runtime in terms of the number of elementary gates either quantum or classical. The implicit assumption behind the query model is that an oracle call has unit cost independently of whether the oracle is classical or quantum. Therefore, whether a potential quantum advantage in the query model carries over to the gate model is contingent upon the efficient implementation of such a quantum oracle in terms of elementary quantum gates. Whether this is the case or not depends on the particular use case at hand. Thus, it is crucial to identify the use cases for which an efficient implementation of the quantum oracles is possible in order for quantum algorithms to be successful in real world applications.

In general, it is possible to achieve reversible versions of every oracle that is possible classically. This is, because we can implement the NAND (NOT-AND), a universal gate, with a Toffoli and a NOT gate. However, this usually requires ancilla qubits potentially endangering quantum memory reductions. Further, aiming for quantum algorithms with speed-up over their classical equivalents, tightens the definition of *efficient*, meaning, an oracle that is efficient for the use in a classical algorithm could be a bottleneck in a faster quantum algorithm.

While it is natural to look at possible applications of quantum computing to FEM simulations [38, 39], it is first necessary to show how FEM simulations can be efficiently embedded in a quantum computer in the oracle model. A previous analysis of the computations, necessary for implementation of quantum oracles, was done, with less detail, in Ref. [91]. Also, we highlight the work of Ref. [103], which, in the same spirit of our work, provides an explicit construction for the block-encoding of certain structured sparse matrices, but without a connection to FEM. For concreteness, we focus on the construction of the oracles in the context of normal mode analysis of solid structures. Once discretized, the dynamic behavior of the problem is completely defined by the so-called mass \mathbf{M} and stiffness \mathbf{K} matrices (cf. Eqs. (3.4) and (3.26))

$$\frac{d^2}{dt^2} \vec{\mathbf{z}} = - \underbrace{\mathbf{M}^{-1/2} \mathbf{K} \mathbf{M}^{-1/2}}_{\equiv \mathbf{H}} \vec{\mathbf{z}}, \quad (5.9)$$

with $\vec{\mathbf{z}} = \mathbf{M}^{1/2} \vec{\mathbf{u}}$. This problem can be mapped to a quantum computer using two oracles for the sparse matrix \mathbf{H} (see Section 2.5). The first is for the determination of the non-zero elements in

Algorithm 1: Calculation of the local response function $G_{uu}(\nu)$ on a quantum computer. The first part estimates classically the register and sample size required by the quantum circuit following in the loop over t . The classical post-processing in the end is required to extract the response function from the measurements.

Data:

- Oscillator u
- Mass m_u
- Renormalization $s\|H\|_{\max}$
- Minimum eigenvalue gap $\Delta_\lambda^{(u)}$
- Tolerances $\varepsilon, \delta, \zeta$
- Controlled quantum walk operator cV which contains the oracles O_g and O_P
- Number of oscillators N
- Number of distinguishable eigenvalues N_u

Result: Local response function $G_{uu}(\nu)$

begin

$m, n, a + l \leftarrow$ compute the required register sizes

$N_S \leftarrow$ compute the required sample size

for $t \in \{1, \dots, N_S\}$ **do**

$\text{reg}_{\text{phase}} \leftarrow \sum_{i=0}^{2^m-1} |i\rangle$

$\text{reg}_{\text{state}} \leftarrow |u\rangle$

$\text{reg}_{\text{ancilla}} \leftarrow |0\rangle^{\otimes a+l}$

for $i \in \{0, \dots, m-1\}$ **do**

cV^{2^i} to $[\text{reg}_{\text{phase}}^{(i)}, \text{reg}_{\text{state}}, \text{reg}_{\text{ancilla}}]$

QFT † to $\text{reg}_{\text{phase}}$

$\varphi_t \leftarrow$ measure $\text{reg}_{\text{phase}}$

define $P(\varphi) = \sum_{t=1}^{N_S} \delta(\varphi - \varphi_t) / N_S$

$\varphi \leftarrow$ identify N_u peaks in $P(\varphi)$

for $j \in \{j : \varphi_j \in \varphi\}$ **do**

$\lambda_j \leftarrow s\|H\|_{\max} \cos(2^{1-m}\pi\varphi_j)$

$W_{uj}^2 \leftarrow 2 \sum_{q=-Q}^{Q-1} P(\lceil \varphi_j \rceil + q)$

$G_{uu}(\nu) \leftarrow m_u^{-1} \sum_{j=1}^{N_u} W_{uj}^2 / (\lambda_j + \nu^2)$

a given row u of \mathbf{H} . In structural problems, the non-zero positions are not random, but follow strict rules. Assuming that we have a system with high symmetry (a regular mesh), the number of rules, necessary to describe all those positions, stays minimal. Hence, a small superposition, generated by a Walsh-Hadamard gate, followed by an u -dependent shift (addition) is usually sufficient.

The second oracle $O_{\mathcal{J}}$ computes the binary sign $\text{sgn}_b(a \geq 0) = 0, \text{sgn}_b(a < 0) = 1$ of H_{uv} and a related angle \mathcal{J}_{uv} defined by Eq. (2.25)

$$\mathcal{J}_{uv} = \arccos \sqrt{\frac{|H_{uv}|}{\|\mathbf{H}\|_{\max}}},$$

with $\|\mathbf{H}\|_{\max} = \max_{uv} |H_{uv}|$, and stores them in $r + 1$ qubits (see Eq. (2.26))

$$O_{\mathcal{J}} |u, v\rangle |0\rangle^{\otimes(r+1)} = |u, v\rangle |\text{sgn}_b(H_{uv}), \mathcal{J}_{uv}\rangle.$$

We further simplify the arccosine in Eq. (2.25). A Taylor expansions transforms it into a signomial function. Note that this is formally a signomial function, which is similar to polynomial functions but allow non-integer exponents for positive variables.

$$\arccos \sqrt{x} = \frac{\pi}{2} - \sqrt{x} - \frac{3}{40}\sqrt{x}^3 - \frac{5}{112}\sqrt{x}^5 + \dots, \quad (5.10)$$

that we truncate after at order K assuming that all x of interest satisfy $x \ll 1$. This method is not accurate for $x \approx 1$. However, it is sometimes possible to renormalize x in trade for a lower accuracy or one can replace $\arccos \sqrt{x}$ with $2 \arcsin \sqrt[4]{(1-x)/2}$, which is equal, but its arguments are limited to $[0, 1/\sqrt[4]{2}]$ (cf. Ref. [91]). The Taylor expansion of the arcsine is similar to the arccosine which is why we stick with the requirements of Eq. (5.10) for simplicity.

The purpose of this section is to demonstrate how the quantum computation of \mathcal{J}_{uv} scales with respect to the matrix size N , and other relevant parameters, in typical scenarios. This brings us to our second key result, demonstrating how to implement the required oracles in polylogarithmic time and with it not endangering potential exponential speed-ups.

5.2.1 Simplification of stiffness and mass matrix

We have not talked so far about the computational routines required to compute the integrals appearing in the mass and stiffness matrix (see Eqs. (3.21) and (3.22)). The best scenario would be that we do not have to compute them at all. In this section, we will describe how to utilize highly symmetrical meshes to achieve a piecewise constant function for the matrix elements.

5.2.1.1 Mass lumping

From Eq. (3.21), it is clear that one can always choose the test functions $\phi_i(\mathbf{x})$ so that the mass matrix is positive and diagonal (a beneficial form, cf. Section 5.2.2). One way of achieving

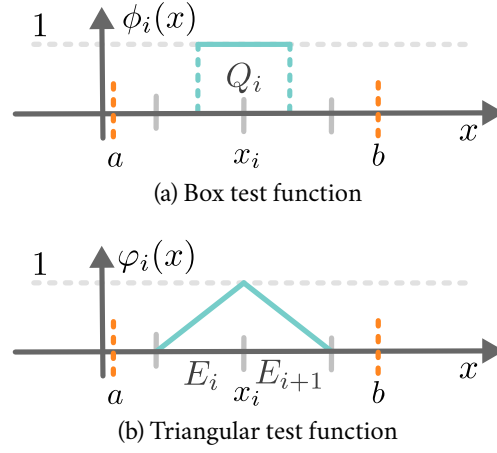


Figure 5.2.1: (a) One-dimensional test function $\phi_i(x)$ (in blue) for the mass matrix in an equidistant mesh. In orange, we see the limits a and b of the geometry Ω . (b) One-dimensional test function $\varphi_i(x)$ (blue) for the stiffness matrix in an equidistant mesh. In orange, we see the limits a and b of the geometry Ω .

this is by *mass lumping*, which is a well established technique [71, 104]. We will describe here one version that relies on box-functions for $\phi_i(\mathbf{x})$. Mathematically, this is achieved by first introducing a set $Q \subset \mathbb{R}^d$ such that $\Omega \subseteq Q$. We partition Q into N subdomains Q_i such that $\mathbf{x}_i \in Q_i$, $Q = \cup_{i=1}^N Q_i$, and $Q_i \cap Q_j = \emptyset, \forall i \neq j$. Examples for one-dimensional and two-dimensional geometries are shown in Fig. 5.2.1a and Fig. 5.2.2 respectively. We can then take

$$\phi_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in Q_i, \\ 0 & \text{if } \mathbf{x} \notin Q_i. \end{cases} \quad (5.11)$$

With this choice the elements of the mass matrix become

$$M_{ij}^{(\alpha\gamma)} = \delta^{(\alpha\gamma)} \delta_{ij} \int_{Q_i} d^d x \rho \quad (5.12)$$

In order to gain further insights into the calculations that need to be performed, let us calculate explicitly the matrix elements $M_{ij}^{(\alpha\gamma)}$ for a d -dimensional hypercuboid mesh C^d with constant grid size Δ (see Fig. 5.2.2 for an example in two dimensions). For homogeneous systems with constant density ρ , and extending ρ to all Q_i for which $\mathbf{x}_i \in \Omega$ (keeping $\rho = 0$ in Q_i for which $\mathbf{x}_i \notin \Omega$), the elements of the mass matrix are

$$M_{ij}^{(\alpha\gamma)} = \begin{cases} \delta^{(\alpha\gamma)} \delta_{ij} \rho \Delta^d & \text{if } \mathbf{x}_i \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \quad (5.13)$$

The integration is more involved for non-constant densities, which are either polynomial or can be approximated by polynomials in \mathbf{x} . They lead to matrix elements polynomial in i . Similar, non-uniform mesh elements destroy the simplicity of Eq. (5.13).

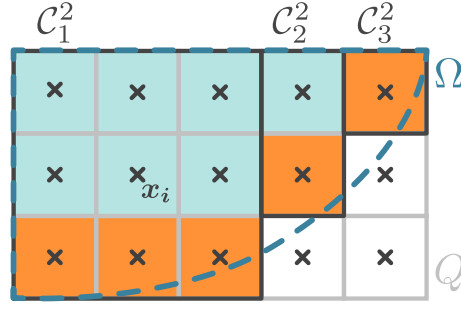


Figure 5.2.2: Alignment between the geometry Ω within the dark blue dashed line and the elements $Q_i \subset Q$ shown as a grid. The elements in light blue are fully part of Ω and with it fulfil $\rho \neq 0$. In orange ($\mathbf{x}_i \in \Omega$) we extend ρ to the whole element to sustain the symmetry. White elements have $\rho = 0$. A simplified geometry $\Omega' = \bigcup_{i=1}^3 C_i^2$ is shown in black contours

5.2.1.2 Stiffness matrix entries in one dimension

In this section, we simplify the elements of the stiffness matrix \mathbf{K} (see Eq. (3.22)). For illustrative reasons, we limit our analysis to the one-dimensional case, in which the domain can simply be taken as an interval $\Omega = [a, b]$ with $a < b$. Non-connected geometries can be considered as separate problems. Setting $Y_{1111} \rightarrow Y$ and assuming a uniform structure with constant Y within Ω , we have

$$K_{ij}^{(1D)} = Y \int_{\Omega} dx \frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x}. \quad (5.14)$$

To discretize the problem we introduce N points $\{x_1, x_2, \dots, x_N\}$ such that $[x_1, x_N] \subseteq [a, b]$ with $x_1 < x_2 < \dots < x_N$. These points align with those used in the mass matrix. In one dimension, we can always choose a set of points that is uniformly distributed with gap size $\Delta = x_{i+1} - x_i, \forall i \in [N-1]$. Here we chose one for which the lower (higher) border of the one dimensional elements $Q_1^{(1D)}$ ($Q_N^{(1D)}$) aligns with a (b), i.e. $x_i = a + (i-1/2)\Delta$ with $\Delta = (b-a)/N$ (see Figs. 5.2.1a and 5.2.1b). We introduce N linear test functions defined as

$$\varphi_i(x) = \begin{cases} 1 + \frac{x-x_i}{\Delta} & \text{if } x \in E_i, \\ 1 - \frac{x-x_i}{\Delta} & \text{if } x \in E_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \quad (5.15)$$

with $i \in [N]$ and $E_i = (x_i - \Delta, x_i]$. E_i and E_{i+1} are the two sub-intervals on the left and on the right of node x_i . Note that the test functions are overlapping as they are non-zero in two sub-intervals. Eq. (5.15) is illustrated in Fig. 5.2.1b.

The stiffness matrix (Eq. (5.14)) requires the first derivatives of the test function

$$\frac{\partial \varphi_i}{\partial x} = \begin{cases} \frac{1}{\Delta} & \text{if } x \in E_i, \\ -\frac{1}{\Delta} & \text{if } x \in E_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \quad (5.16)$$

and finally the product of two of them

$$\frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x} = \begin{cases} \delta_{ij} \frac{1}{\Delta^2} - \delta_{i-1,j} \frac{1}{\Delta^2} & \text{if } (x \in E_i), \\ \delta_{ij} \frac{1}{\Delta^2} - \delta_{i+1,j} \frac{1}{\Delta^2} & \text{if } (x \in E_{i+1}), \\ 0 & \text{otherwise.} \end{cases} \quad (5.17)$$

The matrix elements of the stiffness matrix read

$$K_{ij}^{(1D)} = (\delta_{ij} - \delta_{i-1,j}) \frac{Y}{\Delta^2} \int_{E_i \cap \Omega} dx + (\delta_{ij} - \delta_{i+1,j}) \frac{Y}{\Delta^2} \int_{E_{i+1} \cap \Omega} dx. \quad (5.18)$$

The symmetry is broken only by the contour of Ω . Similar to Section 5.2.1.1 we reduce the domain Ω , in which Y is non-zero, to every $E_i \subset \Omega$, i.e. set $E_1 = E_{N+1} = \emptyset$. The stiffness matrix for this special case splits into two parts

$$K_{ij}^{(1D)} = \begin{cases} 2\delta_{ij} \frac{Y}{\Delta} - \delta_{i\pm 1,j} \frac{Y}{\Delta} & \text{if } i \in [N] \setminus \{1, N\}, \\ \delta_{ij} \frac{Y}{\Delta} - \delta_{i\pm 1,j} \frac{Y}{\Delta} & \text{if } i \in \{1, N\}, \end{cases} \quad (5.19)$$

where $[N] \setminus \{1, N\}$ is the bulk of the geometry and $\{1, N\}$ denotes the edge or boundary.

The integral can still be simplified for non-homogenous systems ($Y(x) \neq \text{const.}$) if the elasticity is a polynomial in x . However, this replaces the piecewise values in $K_{ij}^{(1D)}$ with polynomials in i and j . Non-polynomial elasticities can be approximated by polynomials. For problems, living in d dimensions, the list of conditions in Eq. (5.19) extends with the number of element types (bulk, face, edge or corner elements). Also, the number of Kronecker-delta terms increases with the number of ways two test functions can overlap. Furthermore, in d dimensions it is not generally given that one can choose a mesh for which $Q = \Omega$. For symmetry reasons the mesh and with it the elements will always fill a hypercuboidal space, while the given geometries Ω are usually more complex in their form. Hence, one has to extend the list of conditions by $\mathbf{x}_i, \mathbf{x}_j \in \Omega$ to test if a given element is contained in Ω and return 0 otherwise (cf. Eq. (5.13)).

The computation of the stiffness matrix requires logic operations to distinguish between the different cases. This is described in Section 4.D. In the case of non-homogeneous elasticity, it needs a routine for the calculation of polynomials. The memory and runtime optimized computation of polynomials on quantum hardware is described in Section 4.2.

5.2.2 Oracle implementation based on quantum arithmetic

The core of our proposed algorithm requires an oracle for the encoding of H . However, due to the possible zero-rows in the mass and stiffness matrices the equation of motion Eq. (3.26) contains trivial rows ($0 = 0$) not contributing to the overall solutions, but making M non-invertible. We prevent this by marking those rows with flags ($\mathcal{F}_M, \mathcal{F}_K \neq 0$) on the diagonal elements. This way, they stay disconnected from the remaining matrix and with it contain their

eigenvalues, besides adding the degenerate value \mathcal{F}_M or \mathcal{F}_K respectively. For the combined matrix H (see Eq. (3.4)) with the elements $H_{uv} = H_{ij}^{(\alpha\gamma)}$ this makes

$$H_{ij}^{(\alpha\gamma)} = \begin{cases} \frac{K_{ij}^{(\alpha\gamma)}}{\sqrt{M_{ii}^{(\alpha\alpha)} M_{jj}^{(\gamma\gamma)}}} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \Omega \setminus \mathcal{X}_D, \\ \delta_{ij} \delta^{(\alpha\gamma)} \mathcal{F} & \text{otherwise.} \end{cases} \quad (5.20)$$

Here, we used the structure of Eqs. (5.13) and (5.19), and combined the two flags in $\mathcal{F} = \mathcal{F}_K \mathcal{F}_M^{-1}$ marking the degenerate eigenvalue, which is not contributing to the solution. At this step, we apply the Dirichlet boundary condition based matrix reduction (see Section 3.2.3) by marking the corresponding nodes $\mathbf{x}_i \in \mathcal{X}_D$ with the flag \mathcal{F} . For the use in quantum computing it is commonly required to have $N = 2^n$ for some integer qubit number n . The flag \mathcal{F} can also be used to fill up empty rows to achieve the latter.

For a homogeneous one-dimensional system the mass and stiffness matrix are described by Eqs. (5.13) and (5.19) thus we have

$$H_{ij}^{(1D)} = \begin{cases} 2\delta_{ij} \frac{Y}{\rho \Delta^2} - \delta_{i\pm 1, j} \frac{Y}{\rho \Delta^2} & \text{if } i \in [N] \setminus \{1, N\} \text{ and } x_i, x_j \notin \mathcal{X}_D, \\ \delta_{ij} \frac{Y}{\rho \Delta^2} - \delta_{i\pm 1, j} \frac{Y}{\rho \Delta^2} & \text{if } i \in \{1, N\} \text{ and } x_i, x_j \notin \mathcal{X}_D, \\ \delta_{ij} \mathcal{F} & \text{otherwise,} \end{cases} \quad (5.21)$$

with a max norm of $\|H\|_{\max} = \max(2Y\rho^{-1}\Delta^{-2}, \mathcal{F})$. We can separate \mathcal{F} from the eigenvalues by setting it larger than the largest eigenvalue λ_{\max}

$$\mathcal{F} > \|H\|_1 \geq \lambda_{\max}. \quad (5.22)$$

In our example, the 1-norm is given by

$$\|H\|_1 = \max_i \sum_{j=1}^N |H_{ij}| = \frac{3Y}{\rho \Delta^2}. \quad (5.23)$$

5.2.2.1 Contour of the geometry

From Eq. (5.20), we need to test if $\mathbf{x}_i \in \Omega \setminus \mathcal{X}_D$. In practices, the boundary conditions are usually defined as a subdomain $\mathcal{X}_D \subset \Omega$ instead of a list of nodes, which is why we treat it as such in the following. This means testing a list of conditions that need to be fulfilled before we assign any contribution to H_{uv} . In this section we concentrate on the implementation of those conditions.

Many geometries can be approximated sufficiently by a combination of hypercuboids (see Fig. 5.2.2). However, this method can be adjusted to arbitrary contours that are described by a function (see Section 5.A). Hypercuboids are defined by

$$C^d = \{\mathbf{x} : x^{(\alpha)} \in [a^{(\alpha)}, b^{(\alpha)}], \forall \alpha \in [d]\}, \quad (5.24)$$

where α counts over all or a subset of the d dimensions which are limited by the upper and lower boundaries $a^{(\alpha)}$ and $b^{(\beta)}$ respectively. The corresponding conditions for $\mathbf{x}_i \in C^d$ are

$$a_\alpha \leq x_{i_\alpha, \alpha} \leq b_\alpha, \quad \forall \alpha \in [d]. \quad (5.25)$$

Note that we remapped the index $i \rightarrow \mathbf{i} = (i^{(1)}, \dots, i^{(d)})^T$. This yields in a total of $2d$ conditions per $\mathbf{x}_i \in C^d$ that requires computation and combination. The quantum implementation of the *greater than* comparison is described in Section 4.D.1, while the combination of multiple conditions via *logical and* is described in Section 4.D.2. For larger geometries one can extend the approximated geometry

$$\Omega' = C_1^d \cup C_2^d \cup \dots \cup C_{N_{\text{geo}}}^d, \quad (5.26)$$

where C_i^d is the i -th hypercuboid with unique limits and N_{geo} is the number of combined primitive geometries. Effectively, this extends the number of conditions for $\mathbf{x}_i \in \Omega'$ to $2dN_{\text{geo}}$. The conditions for $\mathbf{x}_i \notin \mathcal{X}_D$ can be constructed analogously using N_D hypercuboids. In one dimension, no approximation is necessary. Ω can be described precisely with $N_{\text{geo}} = 1$ interval C^1 .

5.2.2.2 Required arithmetic operations

In this section, we discuss the arithmetic operations necessary for O_g introduced in Eq. (2.25). A full list of steps is shown in Algorithm 2 for a homogeneous one-dimensional problem. Extending the following for d dimensions would mostly change the number of conditions that need to be tested and due to $d = \text{const.}$, we neglect it from the beginning in this complexity analysis. We further assume that Ω consists of $N_{\text{geo}} = 1$ intervals while keeping N_{geo} to see its impact for higher dimensional problems. The set of Dirichlet boundary conditions can usually be described by simple geometries as well. Hence, we denote N_D the number of geometries describing the Dirichlet boundaries. We approximate $\arccos \sqrt{\cdot}$ as in Eq. (5.10) truncated after $k = K$.

The algorithm starts by assigning three target registers for $H'_{ij} = H_{ij} \|H\|_{\max}^{-1} \sqrt{|H'_{ij}|}$, and \mathcal{H}_{ij} requiring $3r + 1$ qubits next to the $2r$ input qubits. Next, it tests if $x_i, x_j \in \mathcal{X}_D$ and if $i \in \{1, N\}$. This requires a total of $4N_D + 2N_{\text{geo}}$ comparisons combined, which requires a maximum of

$$n_{\text{geo},a} = 8N_D + 4N_{\text{geo}} - 1 \quad (5.27)$$

ancilla qubits, including $4N_D + 2N_{\text{geo}} - 2$ qubits for a multi-controlled CNOT gate and one target qubit. The runtime scales linearly with the number of comparisons, while each comparison contains two ADDs

$$t_{\text{geo}} = \mathcal{O}((N_D + N_{\text{geo}}) t_{\text{ADD}}). \quad (5.28)$$

However, one can parallelize the comparisons for the additional cost of $\mathcal{O}((N_D + N_{\text{geo}})r)$ ancilla qubits. Combining the results of the comparisons scales logarithmically with their number if executed in a cascade like structure. With this we have

$$t_{\text{geo}} = \mathcal{O}(\log_2(N_D + N_{\text{geo}}) + t_{\text{ADD}}) = \mathcal{O}(\log_2(N_D + N_{\text{geo}}) + r), \quad (5.29)$$

Algorithm 2: Oracle O_g for H_{ij} access (see Section 5.2) in one dimension. Here, we access the reduced $H'_{ij} = H_{ij} \|H\|_{\max}^{-1}$ with flag $\mathcal{F} = 4Y\rho^{-1}\Delta^{-2}$. CSM is the transform between the two's complement and the sign magnitude representation (see Fig. 4.2.1). The If statements represent controlled operations.

Data: $|i, j\rangle$, N , and \mathcal{X}_D

Result: $|\text{sgn}_b(H_{ij}), g_{ij}\rangle$

begin

$H' \leftarrow |0\rangle^{\otimes(r+1)}$

$S \leftarrow |0\rangle^{\otimes r}$

$g \leftarrow |0\rangle^{\otimes r}$

if $x_i, x_j \in \mathcal{X}_D$ **then**

$H' \leftarrow |\delta_{ij}\rangle$ // flag

else if $i \in \{1, N\}$ **then**

$H' \leftarrow \left| \frac{1}{4}\delta_{ij} - \frac{1}{4}\delta_{i\pm 1, j} \right\rangle$ // edge

else

$H' \leftarrow \left| \frac{1}{2}\delta_{ij} - \frac{1}{4}\delta_{i\pm 1, j} \right\rangle$ // bulk

$H' \leftarrow |\text{sgn}_b(H_{ij}), |H'_{ij}\rangle = \text{CSM} |H'_{ij}\rangle$

$S \leftarrow \text{SQRT} \left| |H'_{ij}\rangle \right\rangle$

$g \leftarrow \text{POLY} |S_{ij}\rangle$

return $|\text{sgn}_b(H_{ij}), g_{ij}\rangle$

assuming $t_{\text{ADD}} = \mathcal{O}(r)$.

Based on those conditions, the renormalized contributions $H'_{ij} = H_{ij} \|H\|_{\max}^{-1}$ are encoded using addition and the Kronecker delta operation introduced in Chapter 4 and Section 4.D.1, respectively. This requires

$$n_{H',a} = r - 1 \quad (5.30)$$

additional ancilla qubits for the Kronecker-delta including the target qubit. The runtime is

$$t_{H'} = \mathcal{O}(\log_2 r + t_{\text{qcADD}}) = \mathcal{O}(r), \quad (5.31)$$

where we assumed that $t_{\text{qcADD}} = \mathcal{O}(r)$.

The transformation from two's complement into sign-magnitude form requires one controlled qcSUB_1 followed by r CNOT gates. Neither of them requires ancilla qubits. Their runtime is

$$t_{\text{Trafo}} = \mathcal{O}(r + t_{\text{qcADD}}) = \mathcal{O}(r). \quad (5.32)$$

Here, we used that the runtime of CNOT gates $t_{\text{CNOT}} = \mathcal{O}(1)$.

The second last step is the computation of the square root in $\arccos \sqrt{\cdot}$ based on the Newton-Raphson method SQRT. For this we need

$$n'_{\text{SQRT},a} = (L + 4)r + n_{\text{ADD},a} \quad (5.33)$$

ancilla qubits, the new target register included. The runtime is

$$t_{\text{SQRT}} = \mathcal{O}(Lt_{\text{MUL}}). \quad (5.34)$$

The polynomial in Eq. (5.10) can be truncated after $k = K$ leading to

$$n'_{\text{POLY,a}} = (K - 1)(r + 1) + n_{\text{ADD,a}} \quad (5.35)$$

additional ancilla qubits. The runtime is

$$t_{\text{POLY}} = \mathcal{O}(Kt_{\text{MUL}}) = \mathcal{O}(Kr^2). \quad (5.36)$$

In the last step we assumed a carry-ripple based multiplier with $t_{\text{MUL}} = \mathcal{O}(r^2)$.

In total, the oracle $O_{\mathfrak{J}}$ requires

$$n_{O_{\mathfrak{J}}} = \mathcal{O}((L + K + N_{\text{geo}} + N_{\text{D}})r) \quad (5.37)$$

qubits and has a runtime of

$$t_{O_{\mathfrak{J}}} = \mathcal{O}((K + L)r^2 + \log_2(N_{\text{D}} + N_{\text{geo}})). \quad (5.38)$$

Both, the memory requirements and the runtime are independent of N and with it efficient. One could argue that, r has to be large enough to store values in $[0, N^2]$ for $i \in [N]$ and $\Delta^{-2} = \mathcal{O}(N^2)$, which translates into $r = \mathcal{O}(\log_2 N)$. However, this makes the runtime only polylogarithmic in N which is not endangering potential exponential speed-ups.

5.3 Non-local response

Until this point, we limited our algorithm for illustrative reasons to local response functions. Here, we describe how to modify it to compute non-local entries $G_{uv}(v)$ of the response function whose analytical form is given in Eq. (3.32). For the local contributions the preparation of the computational basis state $|u\rangle$ is crucial to extract the weights W_{uj}^2 . The disadvantage of this method is that we can only extract the absolute square of those weights, which does not contain the relative sign and makes this method not suited to evaluate the terms $W_{uj}W_{vj}$ that appear in Eq. (3.32).

A way around this is to add another qubit and still prepare $|u\rangle$ in the state register as shown in Fig. 5.3.1. We will now use the modified Hadamard test to *add* $|v\rangle$ to this state. At first, we apply a Hadamard gate to the additional ancilla qubit starting in $|0\rangle$ (bottom qubit in Fig. 5.3.1). Thus, we have

$$\xrightarrow{\bar{H}} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|u\rangle. \quad (5.39)$$

We now apply a series of CNOT gates with the ancilla qubit as control and the qubits in the n -qubit register as target. The CNOT will be only applied to target qubit i for which $v_i \oplus u_i = 1$.

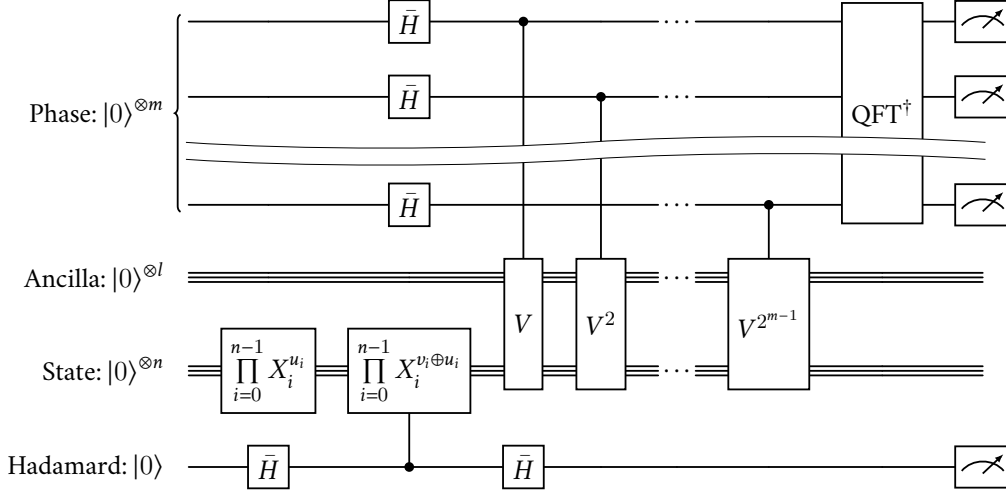


Figure 5.3.1: Modified circuit for the computation of the non-local response function. One can see two additions compared to Fig. 5.1.1. The first is an additional qubit and the second is a modified Hadamard test which we use to prepare the state $|u\rangle$ next to $|v\rangle$.

This brings the state $|u\rangle$ to $|v\rangle$ when the control qubit is in $|1\rangle$. Accordingly, after this series of CNOT gates we get

$$\xrightarrow{\text{CNOTs}} \frac{1}{\sqrt{2}} (|0\rangle |u\rangle + |1\rangle |v\rangle). \quad (5.40)$$

At last, we apply another Hadamard gate to the ancilla qubit and obtain

$$\xrightarrow{\bar{H}} \frac{1}{2} (|0\rangle (|u\rangle + |v\rangle) + |1\rangle (|u\rangle - |v\rangle)). \quad (5.41)$$

Measuring the additional qubit results in either the sum or difference of those two states

$$\frac{1}{\sqrt{2}} (|u\rangle \pm |v\rangle) = \frac{1}{\sqrt{2}} \sum_{j=1}^N (W_{uj} \pm W_{vj}) |\lambda_j\rangle. \quad (5.42)$$

If we proceed now as before with the local version of the algorithm, depending on the ancilla being in 0 or 1, we would extract the square of those amplitudes

$$\frac{1}{2} (W_{uj} \pm W_{vj})^2 = \frac{1}{2} W_{uj}^2 + \frac{1}{2} W_{vj}^2 \pm W_{uj} W_{vj}, \quad (5.43)$$

where we used that W_{uj} and W_{vj} are real. Let $P(0, x)$, $P(1, x)$ the probability of measuring the bitstring x in the phase register given that the ancilla is measured in 0 and 1, respectively. By adapting Eq. (5.8) with the new amplitudes we find

$$\frac{1}{2} (W_{vj} + W_{uj})^2 \approx 2 \sum_{q=-Q}^{Q-1} P(0, [\varphi'_j] + q), \quad (5.44a)$$

$$\frac{1}{2}(W_{vj} - W_{uj})^2 \approx 2 \sum_{q=-Q}^{Q-1} P(1, \lceil \varphi'_j \rceil + q), \quad (5.44b)$$

or, equivalently,

$$W_{vj}W_{uj} \approx \sum_{q=-Q}^{Q-1} \left(P(0, \lceil \varphi'_j \rceil + q) - P(1, \lceil \varphi'_j \rceil + q) \right). \quad (5.45)$$

5.4 Data availability

The algorithm presented in this manuscript was tested for the example of linearly coupled oscillators. For this case, it was implemented and emulated from end-to-end in Qiskit [105], including a simplified compilation of the matrix oracles. The corresponding repository containing all the required packages and a demonstration notebook can be found at <https://go.fzj.de/resp-QPE>.

Appendix

5.A Advanced contours

In Section 5.2.2.1 we already described how to test if a node is within a geometry consisting purely of hypercuboids. In this section we will extend the list of primitive geometries and give an idea of how to construct more advanced geometries. The second primitive is a d -dimensional hyperellipsoid \mathcal{E}^d limited by

$$\sum_{\alpha=1}^d \left(\frac{\Delta i^{(\alpha)} - c^{(\alpha)}}{h^{(\alpha)}} \right)^2 \leq 1, \quad (5.46)$$

where $\mathbf{c} = (c^{(1)}, \dots, c^{(d)})^T$ is the center of the hyperellipsoid and $h^{(\alpha)}$ is the length of its semi-axes in direction α . The implementation of this condition requires the computation of a polynomial with d terms (see Section 4.2) and the comparison with 1 (see Section 4.D.1).

The conditions for hypercuboids and hyperellipsoids can be combined partially into other geometries like a cylinder

$$\Delta^{-1} a^{(1)} \leq i^{(1)} \leq \Delta^{-1} b^{(1)} \wedge \sum_{\alpha=2}^3 \left(\frac{\Delta i^{(\alpha)} - c^{(\alpha)}}{h^{(\alpha)}} \right)^2 \leq 1, \quad (5.47)$$

or the geometries can be combined into bigger geometries (e.g. $\mathbf{x}_i \in \mathcal{C}^d \cup \mathcal{E}^d$). Rotated versions of those geometries can be used by rotating the position variable (e.g. $x^{(1)} \rightarrow \cos \theta x^{(1)} + \sin \theta x^{(2)}$). The list of conditions can be extended by arbitrary signomial functions describing the limits of the geometry if necessary.

Complexity Analysis

The advantage of our proposed algorithm depends on its memory requirements and its runtime, i.e. its computational complexity. We start this chapter with a summary of the main results in terms of scaling with respect to relevant parameters. Before we do this, we define the additional parameters that enter the scaling analysis. First, we define the minimum, positive gap in the eigenvalue spectrum as

$$\Delta_\lambda = \min\{|\lambda_i - \lambda_j| : i, j \in [N], |\lambda_i - \lambda_j| > 0\}. \quad (6.1)$$

Additionally, we define the minimum, positive eigenvalue gap at oscillator u as

$$\Delta_\lambda^{(u)} = \min\{|\lambda_i - \lambda_j| : i, j \in [N], |\lambda_i - \lambda_j| > 0, W_{ui}^2 > 0, W_{uj}^2 > 0\}, \quad (6.2)$$

with W_{ui} and W_{uj} being the coefficients in Eq. (5.1). This is the parameter that matters in our algorithm since, in the presence of degeneracies, weights associated with the same eigenvalue would just be gathered together in Eq. (3.13) and eigenvectors that have no support at oscillator u give zero contribution. Obviously, $\Delta_\lambda^{(u)} \geq \Delta_\lambda$. The scaling further depends on the required, additive tolerances ε and δ for the eigenvalues λ_j and weights W_{uj}^2 , respectively, which we treat as independent parameters of our choice¹. Moreover, we define the parameter N_u to be the number of eigenvectors that have support at oscillator u , that is for which $W_{uj}^2 > 0$. Note that in the worst case scenario $N_u = N$. Finally, we denote by ζ the probability that the estimate of W_{uj}^2 deviates by more than δ . All the parameters used in the complexity analysis are summarized in Table 6.0.1.

Our main result is described by the following theorem.

Theorem 1. *Let H be an $N \times N$ Hermitian matrix that describes a system of coupled oscillators as in Eq. (3.4) and let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph associated with the network of oscillators. Let H be s -sparse with maximum norm $\|H\|_{\max}$. Given an oscillator $u \in \mathcal{V}$, there exists a quantum*

This chapter has been published in the preprint “Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405.08694”.

¹The smaller the error on the eigenvalues, the smaller the error on the weights, and vice versa. However, for our analysis it is easier to think about them as independent tolerances.

Table 6.0.1: Parameters used in the complexity analysis.

Symbol	Meaning
N	Number of rows or columns of H
s	Sparsity of H
$\ H\ _{\max}$	Maximum absolute matrix entry of H
$\Delta_{\lambda}^{(u)}$	Minimum eigenvalue gap at oscillator u
ε	Additive eigenvalue tolerance
δ	Additive weight tolerance
ζ	Probability of failure
N_u	Number of nonzero W_{uj} at oscillator u

algorithm that approximates the eigenvalues λ_j of H and the weights W_{uj}^2 in Eq. (3.13), with additive error ε and δ , respectively, that requires

$$n_{\text{tot}} = O \left(\log_2 \left(N s \|H\|_{\max} \max \left(\frac{1}{\varepsilon}, \frac{1}{\delta \Delta_{\lambda}^{(u)}} \right) \right) \right) \quad (6.3)$$

qubits and

$$N_{\text{queries}}^{(\text{tot})} = O \left(\frac{s \|H\|_{\max}}{\delta^2} \ln \left(\frac{N_u}{\zeta} \right) \max \left(\frac{1}{\varepsilon}, \frac{1}{\delta \Delta_{\lambda}^{(u)}} \right) \right) \quad (6.4)$$

total queries of the matrix oracles O_P and O_g in Eq. (2.24), Eq. (2.26), respectively.

A detailed error analysis of the QPE-based algorithm behind Theorem 1 is presented in Sections 6.1 to 6.3. In what follows, we provide further comments on the complexity of our result.

First, we note that Eq. (6.3) implies that ε should be taken at least $\varepsilon < \Delta_{\lambda}^{(u)}$, since $\delta < 1$, to use the full potential of n_{tot} qubits. Moreover, in Eq. (6.3), we are not considering the ancilla qubits necessary to implement the oracles O_P , O_g , since these are problem-dependent. If we use r bits to represent the matrix elements, the required number of bits in classical algorithms is usually $O(rN^2)$, while for s -sparse matrices it can be $O(rsN)$ in the best case. The gate complexity of our quantum algorithm depends on the complexity of implementing the oracles O_P and O_g , which are, as mentioned above, problem-dependent. Generally, as long as the matrix elements of a matrix can be accessed efficiently classically, this should also hold on a quantum computer, although at the price of having several ancillas to make the computation reversible. As an example, for the case of a linear chain of oscillators, we provide a full compilation of the oracles into elementary gates in the code available in the repository associated with this manuscript (see Section 5.4). In Section 5.2, we show how to implement the oracles for more general FEM

problems efficiently. As such, we choose to present the complexity of our algorithm in terms of the number of oracle queries N_{queries} . The query complexity of a single run of QPE is (see Section 6.2)

$$N_{\text{queries}} = \mathcal{O} \left(s \|H\|_{\max} \max \left(\frac{1}{\varepsilon}, \frac{1}{\delta \Delta_{\lambda}^{(u)}} \right) \right). \quad (6.5)$$

This is mostly due to the repetition of the controlled- V operator in the core of the phase estimation (cf. Fig. 5.1.1 and Algorithm 1).

QPE is a probabilistic algorithm from which we sample multiple times to estimate the eigenvalues and the associated probabilities. As we show in Section 6.3, the number of samples N_S needed to estimate each weight W_{uj}^2 via empirical averages with error δ and failure probability ζ scales as

$$N_S = \mathcal{O} \left(\frac{1}{\delta^2} \ln \left(\frac{N_u}{\zeta} \right) \right). \quad (6.6)$$

In the worst case scenario when $N_u = N = 2^n$, the scaling would be logarithmic with the size of the matrix N , i.e., linear with the number of qubits n . We highlight the fact that this is not equivalent to requiring that the probability distribution W_{uj}^2 is estimated with error δ in total variation distance. Using the definition of total variation distance and Hoeffding's inequality similarly to Section 6.3, one can show that in this case the scaling would be linear, and not logarithmic, with N_u . Moreover, we are not requiring to find all the N_u eigenvalues that contribute in the response function at oscillator u . This would require a number of samples scaling at least as N_u , but what matters for us is the evaluation of the weights W_{uj}^2 with error δ , since the weights appear at the numerator in Eq. (3.13). This leads to the inverse quadratic scaling in δ in Eq. (6.6). Combining Eq. (6.5) and Eq. (6.6), we obtain the total number of queries as in Eq. (6.4).

The rest of this chapter is structured as follows. We start with an analysis of the worst case memory requirements for given tolerances ε and δ in Section 6.1. The runtime of our algorithm consists of two parts. First, the runtime of a single iteration, limited in Section 6.2, and second, the sample size required to guarantee an estimation within the tolerances in Section 6.3. We conclude this chapter with a critical assessment of the scenarios in which the proposed algorithm does not contain its speed-up.

6.1 Memory requirements

The full QPE estimation is applied to three registers. The m -qubit phase register, the n -qubit state register and the $(a + l)$ -ancilla register used by the controlled walk operator cV . We will focus on the $n + a + l + 1$ qubits required for the implementation of cV . For this we start with U_T described in Fig. 2.4. The total number of qubits n_{U_T} required for the state transformation is given by

$$n_{U_T} = 2n + 1 + \max(a_P, r + 1 + a_g, r + 2 + a_{\text{ADD}}), \quad (6.7)$$

where $(r + 1)$ is the number of qubits used by O_g to store the angle ϑ_{uv} and the sign of H_{uv} . Similar to m it effects the accuracy of the results. Moreover, the number of ancilla qubits necessary for the position and angle oracles, and the quantum adder are given by a_P , a_g and a_{ADD} , respectively. In Eq. (6.7), we used that the ancilla qubits used in O_P can be reused in O_g , while the ancilla qubits used there can be used again in O_{ADD} .

The state transformation U_T appears twice in the circuit for cV (cf. Fig. 2.5). The circuit is further extended by a control qubit and the last qubit necessary for the block-encoding. Hence, we have a total number of n_{cV} qubits necessary for the controlled quantum walk, which is

$$n_{cV} = 2n + 3 + \max(a_P, r + 1 + a_g, r + 2 + a_{ADD}, n + 2). \quad (6.8)$$

Here we see additional $n + 2$ ancilla qubits which can be reused. The first $n + 1$ of them are required for the Toffoli gates in the multi-controlled NOT gate in Fig. 2.5 and the last one is the target of the same. In total, we have m repetitions of cV with varying control qubit which leads to the scaling described in Eq. (6.3). There we replaced m with the scaling Eq. (6.42) derived in Sections 6.1.1 and 6.1.2.

In the following two sections, we discuss the effect of discretization and statistical errors on the estimated eigenvalues $\tilde{\lambda}_j$ and weights \tilde{W}_{uj}^2 and how one can control the maximum error by increasing the size of the phase register. This analysis proves the statement of Theorem 1.

In particular, we analyze the effect of the discretization of the phase when it is stored as binary number in the m -qubit register with focus on the eigenvalue in Section 6.1.1 and the weights in Section 6.1.2. The size of the phase register determines the number of oracle queries necessary to implement the controlled- V^{2^k} operations in Algorithm 1 shown in Fig. 5.1.1. We discuss this connection in Section 6.2. We point out that the number of oracle queries constitutes the main limitation of the algorithm in most *practical* cases (see discussion in Section 6.4). In Section 6.3, we describe the connection between the sample size and the statistical accuracy of the estimated weights. In the following, for notational simplicity, we focus on the non-degenerate systems, but the results carry over to the degenerate case as discussed in Section 6.4. A discussion of the limits and potential optimal conditions is also given in Section 6.4.

6.1.1 Phase register size for given eigenvalue tolerances

The following analysis is more easily understood by considering the case in which we perform QPE using the operator V as in Fig. 5.1.1, but taking as initial state in the n -qubit register a single eigenstate of H $|\lambda\rangle$. When we measure the m -qubit phase register at the end of QPE we get a bitstring that represents an integer number between 0 and $M - 1$ with $M = 2^m$. This gives an estimate of $\pm\varphi \bmod M \in [0, M)$ with φ (cf. Eq. (5.6)) given by

$$\varphi = \frac{M}{2\pi} \arccos \left(\frac{\lambda}{s \|H\|_{\max}} \right). \quad (6.9)$$

Accordingly, there exists a $\tilde{\varphi} \in \{0, 1, \dots, 2^{m-1} - 1\}$ such that²

$$|\varphi - \tilde{\varphi}| \leq \frac{1}{2}, \quad (6.10)$$

to which we can associate an estimated eigenvalue $\tilde{\lambda}$ by inverting Eq. (6.9). The phase stored in the ancilla register corresponds to the estimated $\tilde{\lambda}$ as follows

$$\tilde{\varphi} = \frac{M}{2\pi} \arccos \left(\frac{\tilde{\lambda}}{s\|H\|_{\max}} \right). \quad (6.11)$$

Thus, we get

$$\frac{M}{2\pi} \left| \arccos \left(\frac{\lambda}{s\|H\|_{\max}} \right) - \arccos \left(\frac{\tilde{\lambda}}{s\|H\|_{\max}} \right) \right| \leq \frac{1}{2} \quad (6.12)$$

The mean value theorem for the arccosine function states that for $x > \tilde{x}$, $\exists x_0 \in (\tilde{x}, x)$ such that

$$\frac{|\arccos x - \arccos \tilde{x}|}{|x - \tilde{x}|} = \frac{1}{\sqrt{1 - (x_0)^2}}. \quad (6.13)$$

Applying the mean value theorem to our case we get that $\exists \lambda_0 \in (\tilde{\lambda}, \lambda)$ if $\lambda > \tilde{\lambda}$ or $\exists \lambda_0 \in (\lambda, \tilde{\lambda})$ if $\lambda < \tilde{\lambda}$, such that

$$s\|H\|_{\max} \frac{\left| \arccos \left(\frac{\lambda}{s\|H\|_{\max}} \right) - \arccos \left(\frac{\tilde{\lambda}}{s\|H\|_{\max}} \right) \right|}{|\lambda - \tilde{\lambda}|} = \frac{1}{\sqrt{1 - \left(\frac{\lambda_0}{s\|H\|_{\max}} \right)^2}} \geq 1, \quad (6.14)$$

where the last inequality holds since $0 \leq \lambda_0 \leq s\|H\|_{\max}$. Combining Eq. (6.12) and Eq. (6.14), we get

$$|\lambda - \tilde{\lambda}| \leq \frac{\pi s\|H\|_{\max}}{M}. \quad (6.15)$$

This tells us that the error of the estimated eigenvalue is below ε if the number of qubits in the phase register satisfies

$$m \geq m_{\min}^{(1)} = \left\lceil \log_2 \left(\frac{\pi s\|H\|_{\max}}{\varepsilon} \right) \right\rceil. \quad (6.16)$$

6.1.2 Phase register size for given weight tolerances

We analyze the error on the estimated weights W_{uj}^2 of the local response function defined in Eq. (5.8) due to the finite size of the phase register. In our analysis we assume that our estimated peak $\tilde{\varphi}_j$ satisfies $\lceil \tilde{\varphi}_j \rceil = \lceil \varphi_j \rceil$ with φ_j associated with the true eigenvalue λ_j as in Eq. (5.6)³. Accordingly, our estimator for the weight W_{uj}^2 is

$$\tilde{W}_{uj}^2 = 2 \sum_{q=-Q}^{Q-1} P(\lceil \varphi_j \rceil + q), \quad (6.17)$$

²The first bit in the phase register just represents the sign of $\pm \varphi \bmod M$, so φ is represented with $m - 1$ bits.

³An analogous derivation is possible if we assume $\lfloor \tilde{\varphi}_j \rfloor = \lfloor \varphi_j \rfloor$.

with the range Q to be determined.

The parameter Q should be taken large enough to guarantee

$$\left| \tilde{W}_{uj}^2 - W_{uj}^2 \right| \leq \delta \quad (6.18)$$

An increase in Q prevents two things. First, it limits the probability we miss by adding only over the $2Q$ probabilities for values closest to the center of a peak. This is connected to the lower bound of $\tilde{W}_{uj}^2 - W_{uj}^2$. The second is that we need to increase the distance between two peaks to fit $2Q$ between them which prevents probability *leakage* between them. This is connected to the upper bound of $\tilde{W}_{uj}^2 - W_{uj}^2$.

For the lower limit we need to study the probability to measure x in the phase register defined in Eq. (5.7) and focus only on one contribution

$$P(x) \geq \frac{W_{uj}^2}{2} \left| a_{x+}^{(j)} \right|^2. \quad (6.19)$$

We will now further expand $|a_{x+}^{(j)}|^2$ by using its definition Eq. (5.5) and the geometric series

$$\sum_{z=0}^{M-1} x^z = \frac{1 - x^M}{1 - x}. \quad (6.20)$$

Thus, we have

$$\left| a_{x+}^{(j)} \right|^2 = \frac{1}{M^2} \left| \frac{1 - e^{2\pi i(\varphi_j - x)}}{1 - e^{2\pi i(\varphi_j - x)/M}} \right|^2 = \frac{1}{M^2} \left[\frac{\sin(\pi(\varphi_j - x))}{\sin(\pi(\varphi_j - x)/M)} \right]^2 \geq \left[\frac{\sin(\pi(\varphi_j - x))}{\pi(\varphi_j - x)} \right]^2. \quad (6.21)$$

where we used $|\sin x| \leq |x|$. Using Eq. (6.17), Eq. (6.19) and Eq. (6.21), we get

$$\tilde{W}_{uj}^2 \geq W_{uj}^2 \sum_{q=-Q}^{Q-1} \left[\frac{\sin(\pi(\varphi_j - \lceil \varphi_j \rceil - q))}{\pi(\varphi_j - \lceil \varphi_j \rceil - q)} \right]^2, \quad (6.22)$$

which is minimal for $\lceil \varphi_j \rceil - \varphi_j = 1/2$. Hence,

$$\tilde{W}_{uj}^2 \geq W_{uj}^2 \sum_{q=-Q}^{Q-1} \left[\frac{1}{\pi(1/2 + q)} \right]^2 = 2W_{uj}^2 \sum_{q=1}^Q \left[\frac{1}{\pi(q - 1/2)} \right]^2 \quad (6.23)$$

Now, we introduce the trigamma function

$$\Psi^{(1)}(z) = \sum_{q=0}^{+\infty} \frac{1}{(z + q)^2}, \quad (6.24)$$

which for half integer values takes the form

$$\Psi^{(1)}\left(Q + \frac{1}{2}\right) = \frac{\pi^2}{2} - \sum_{q=1}^Q \frac{1}{(q - 1/2)^2}, \quad (6.25)$$

with $Q \in \mathbb{N}$. $\Psi^{(1)}(z)$ satisfies

$$\Psi^{(1)}(z) \leq \frac{z+1}{z^2} \leq \frac{2}{z}, \quad \forall z > 0. \quad (6.26)$$

More details about the trigamma function and a derivation of Eq. (6.26) can be found in Ref. [106]. Combining Eqs. (6.23), (6.25) and (6.26) leads to

$$\tilde{W}_{uj}^2 \geq W_{uj}^2 \left(1 - \frac{2}{\pi^2} \Psi^{(1)} \left(Q + \frac{1}{2} \right) \right) \geq W_{uj}^2 \left(1 - \frac{4}{\pi^2 (Q + \frac{1}{2})} \right) \quad (6.27)$$

Since, $W_{uj}^2 \leq 1$, this tells us that our estimation \tilde{W}_{uj}^2 is at most δ less than W_{uj}^2 if we choose a Q such that

$$\frac{4}{\pi^2 (Q + 1/2)} \leq \delta. \quad (6.28)$$

or

$$Q \geq Q_{\min}^{(1)} = \left\lceil \frac{4}{\pi^2 \delta} - \frac{1}{2} \right\rceil. \quad (6.29)$$

For the upper limit, let us consider

$$\tilde{W}_{uj}^2 - W_{uj}^2 = \sum_{i=0}^{N-1} W_{ui}^2 \sum_{q=-Q}^{Q-1} \left(\left| a_{\lceil \varphi_j \rceil + q, +}^{(i)} \right|^2 + \left| a_{\lceil \varphi_j \rceil + q, -}^{(i)} \right|^2 \right) - W_{uj}^2. \quad (6.30)$$

Since $\sum_{q=-Q}^{Q-1} \left| a_{\lceil \varphi_j \rceil + q, +}^{(j)} \right|^2 \leq 1$, we obtain

$$\tilde{W}_{uj}^2 - W_{uj}^2 \leq \sum_{i \neq j} W_{ui}^2 \sum_{q=-Q}^{Q-1} \left(\left| a_{\lceil \varphi_j \rceil + q, +}^{(i)} \right|^2 + \left| a_{\lceil \varphi_j \rceil + q, -}^{(i)} \right|^2 \right) + W_{uj}^2 \sum_{q=-Q}^{Q-1} \left| a_{\lceil \varphi_j \rceil + q, -}^{(j)} \right|^2. \quad (6.31)$$

Let us consider the coefficients $\left| a_{\lceil \varphi_j \rceil + q, +}^{(i)} \right|^2$. We can rewrite them as

$$\left| a_{\lceil \varphi_j \rceil + q, +}^{(i)} \right|^2 = \frac{1}{M^2} \left[\frac{\sin(\pi(\gamma_i + \lceil \varphi_i \rceil - \lceil \varphi_j \rceil - q))}{\sin(\pi(\gamma_i + \lceil \varphi_i \rceil - \lceil \varphi_j \rceil - q)/M)} \right]^2 \quad (6.32)$$

with $\gamma_i = \varphi_i - \lceil \varphi_i \rceil \in (-1, 0]$. We assume that $|\lceil \varphi_i \rceil - \lceil \varphi_j \rceil| > 2Q \forall i \neq j$, i.e., that the peaks are separated by at least $2Q$. Using the fact that $\left| a_{\lceil \varphi_j \rceil + q, +}^{(i)} \right|^2$ decreases by increasing $|\lceil \varphi_i \rceil - \lceil \varphi_j \rceil - q|$ for integer values for any fixed $\gamma_i \in (-1, 0]$, we get

$$\left| a_{\lceil \varphi_j \rceil + q, +}^{(i)} \right|^2 \leq \frac{1}{M^2} \left[\frac{\sin(\pi(\gamma_i + Q + 1))}{\sin(\pi(\gamma_i + Q + 1)/M)} \right]^2 = |a_{\gamma_i + Q + 1}|^2. \quad (6.33)$$

The function $|a_{\gamma+Q+1}|^2$ has exactly one local maximum for $\gamma \in (-1, 0]$, which coincides with the global maximum in the interval. We simply denote by $\bar{\gamma} \in (-1, 0]$ the point where $|a_{\gamma+Q+1}|^2$ is maximal. The same analysis can be repeated for $|a_{\lceil\varphi_j\rceil+q,-}^{(i)}|^2$. Thus, we get

$$\tilde{W}_{uj}^2 - W_{uj}^2 \leq \sum_{i \neq j} 2W_{ui}^2 \sum_{q=-Q}^{Q-1} |a_{\bar{\gamma}+Q+1}|^2 + W_{uj}^2 \sum_{q=-Q}^{Q-1} |a_{\lceil\varphi_j\rceil+q,-}^{(j)}|^2. \quad (6.34)$$

We will further assume that $Q < |\pm\varphi_j| < M/2 - Q$, which guarantees that the two peaks φ_j and $-\varphi_j$ are also $2Q$ away from each other and Eq. (6.33) is satisfied for $i = j$. The previous condition fails only for $\lambda_j \approx s\|H\|$, where $\pm\varphi_j \approx 0$. This can be prevented by further reducing H with the renormalization factor $\cos(2\pi Q M^{-1})$.

Now, we replace the amplitude in the last term of Eq. (6.34) as well and have

$$\tilde{W}_{uj}^2 - W_{uj}^2 \leq \left(\sum_{i \neq j} 2W_{ui}^2 + W_{uj}^2 \right) \sum_{q=-Q}^{Q-1} |a_{\bar{\gamma}+Q+1}|^2. \quad (6.35)$$

At last, we replace the amplitudes using Eq. (6.21) and use the normalization of the eigenvalues $\sum_i W_{ui}^2 = 1$ to find

$$\tilde{W}_{uj}^2 - W_{uj}^2 \leq \frac{4Q}{M^2} \left| \frac{\sin(\pi(\bar{\gamma} + Q + 1))}{\sin(\pi(\bar{\gamma} + Q + 1)/M)} \right|^2. \quad (6.36)$$

Now we use $|\sin x| \leq 1$ and $|\sin x| \geq |2x/\pi|$, $\forall |x| \leq \pi/2$ and obtain that our estimation is less than δ larger than the exact weight W_{uj}^2 if

$$\tilde{W}_{uj}^2 - W_{uj}^2 \leq \frac{Q}{(\bar{\gamma} + Q + 1)^2} \leq \frac{1}{Q} \leq \delta, \quad (6.37)$$

and $2Q \leq M - 1$. This yields another lower limit for Q , which is

$$Q_{\min}^{(2)} = \left\lceil \frac{1}{\delta} \right\rceil. \quad (6.38)$$

A comparison between Eqs. (6.29) and (6.38) shows that the latter is larger and therefore sufficient to fulfill both conditions. Thus, we set

$$Q \geq \left\lceil \frac{1}{\delta} \right\rceil \quad (6.39)$$

The previous analysis depends on the fact that we have a sufficient number of qubits m in the phase register, so that we can distinguish between two neighboring phases. Mathematically, this translates into the condition

$$\min_i |\varphi_i - \varphi_j| \geq \min_i \frac{M|\lambda_i - \lambda_j|}{2\pi s\|H\|_{\max}} \geq 2Q, \quad (6.40)$$

where we used the results obtained in Section 6.1.1. We can guarantee this if we take the number of qubits in the phase register m such that

$$m \geq m_{\min}^{(2)} = \left\lceil \log_2 \frac{4\pi s \|H\|_{\max}}{\delta \Delta_{\lambda}^{(u)}} \right\rceil, \quad (6.41)$$

with $\Delta_{\lambda}^{(u)}$ is defined in Eq. (6.2). Combining the results of Eq. (6.16) and Eq. (6.41), we obtain

$$m \geq \max \left(m_{\min}^{(1)}, m_{\min}^{(2)} \right) = \left(\left\lceil \log_2 \frac{\pi s \|H\|_{\max}}{\varepsilon} \right\rceil, \left\lceil \log_2 \frac{4\pi s \|H\|_{\max}}{\delta \Delta_{\lambda}^{(u)}} \right\rceil \right), \quad (6.42)$$

which justifies Eq. (6.3).

6.2 Runtime of one iteration

The number of qubits m in the phase register determines the number of oracles queries in the algorithm in Fig. 5.1.1. As we can see from Fig. 2.5, implementing a controlled- V operator requires using U_T and U_T^{\dagger} once each. The circuit for U_T is shown in Fig. 2.4 and uses once the position oracle O_P , once the angle oracle $O_{\mathfrak{g}}$ and once its inverse $O_{\mathfrak{g}}^{\dagger}$. Thus, for each U_T we have 3 oracle calls and a controlled- V requires 6 oracle calls. In order, to perform a single run of QPE with m qubit on the phase register we would need

$$\sum_{k=0}^{m-1} 2^k = 2^m - 1, \quad (6.43)$$

controlled- V operations, which yields the number of oracle queries

$$N_{\text{queries}} = 6(2^m - 1). \quad (6.44)$$

Using the result of Eq. (6.42) yields the scaling result in Eq. (6.5).

6.3 Upper limit for the sample size

We now turn our attention to the scaling of the number of samples N_S . Our aim here is to perform a simplified analysis that should capture the asymptotic behavior of the scaling, while the details will eventually depend on the procedure used to infer eigenvalues and weights from measured data in the phase register (see Ref. [25] for instance). In particular, we work with the assumption that the number of qubits m in the phase register is taken large enough to resolve all the peaks. From Eq. (6.16) this means

$$m \gg \left\lceil \log_2 \frac{\pi s \|H\|_{\max}}{\Delta_{\lambda}} \right\rceil. \quad (6.45)$$

If this holds we can unambiguously identify the eigenvalue peaks and neglect possible errors in the assignment of the samples to the eigenvalues. In this regime, QPE de facto behaves as a measurement in the eigenbasis of H , where each eigenvalue λ_j appears with probability $p(j) = W_{uj}^2$ ⁴. By performing QPE many times, we obtain samples from which we can reconstruct the probability distribution W_{uj}^2 .

Let Λ_u be the set of eigenvalues λ_j that have *support* on the oscillator u , i.e.,

$$\Lambda_u = \{\lambda_j : W_{uj}^2 > 0\}, \quad (6.46)$$

and $N_u = |\Lambda_u|$ the number of elements of Λ_u ($N_u \leq N$). W_{uj}^2 is a probability distribution over the finite alphabet Λ_u .

Let $p(\lambda)$ be a generic probability distribution over the finite alphabet Λ_u . With each $\lambda_j \in \Lambda_u$ we can associate a Bernoulli random variable $X_{\lambda_j}(\lambda)$ defined $\forall \lambda \in \Lambda_u$ as

$$X_{\lambda_j}(\lambda) = \begin{cases} 1, & \lambda = \lambda_j, \\ 0, & \lambda \neq \lambda_j. \end{cases} \quad (6.47)$$

Notice that $E[X_{\lambda_j}] = p(\lambda_j)$. By drawing N_S samples from $p(\lambda)$, we get samples $x_{\lambda_j}^{(i)}$, $i \in [N_S]$ for X_{λ_j} . By taking the arithmetic average of these samples we can construct the empirical estimator for $p(\lambda_j)$:

$$\tilde{p}(\lambda_j) = \frac{1}{N_S} \sum_{i=1}^{N_S} x_{\lambda_j}^{(i)}. \quad (6.48)$$

Our aim is to understand how many samples N_S we have to draw from $p(\lambda)$ to have, with probability $1 - \zeta$, that each $\tilde{p}(\lambda_j)$, $\lambda_j \in \Lambda_u$ estimated $p(\lambda_j)$ with accuracy δ . Mathematically, this means

$$\text{Prob} \left(\bigcup_{\lambda_j \in \Lambda_u} \{|p(\lambda_j) - \tilde{p}(\lambda_j)| \geq \delta\} \right) \leq \zeta. \quad (6.49)$$

Thus, our aim is to obtain bounds for the left-hand side of Eq. (6.49). First, using the union bound we get

$$\text{Prob} \left(\bigcup_{\lambda_j \in \Lambda_u} \{|p(\lambda_j) - \tilde{p}(\lambda_j)| \geq \delta\} \right) \leq \sum_{\lambda_j \in \Lambda_u} \text{Prob} (|p(\lambda_j) - \tilde{p}(\lambda_j)| \geq \delta). \quad (6.50)$$

Now, an application of Hoeffding's inequality [107, 108] to each random variable X_{λ_j} yields

$$\text{Prob} (|p(\lambda_j) - \tilde{p}(\lambda_j)| \geq \delta) = \text{Prob} \left(\left| E[X_{\lambda_j}] - \frac{1}{N_S} \sum_{i=1}^{N_S} x_{\lambda_j}^{(i)} \right| \geq \delta \right) \leq 2e^{-2N_S\delta^2}. \quad (6.51)$$

⁴In our version of QPE we have two peaks associated with each eigenvalue, but we can simply sum the probability of each peak to get W_{uj}^2 .

Plugging Eq. (6.51) into Eq. (6.50) and enforcing Eq. (6.49) we get

$$\text{Prob}\left(\bigcup_{\lambda_j \in \Lambda_u} \{|p(\lambda_j) - \tilde{p}(\lambda_j)| \geq \delta\}\right) \leq 2N_u e^{-2N_S \delta^2} \leq \zeta, \quad (6.52)$$

which holds if

$$N_S \geq \frac{1}{2\delta^2} \ln\left(\frac{2N_u}{\zeta}\right). \quad (6.53)$$

Eq. (6.53) justifies Eq. (6.6).

6.4 Critical assessment

6.4.1 Limitations of the algorithm

Neglecting δ for a moment, the computational cost (cf. Eq. (6.3) and Eq. (6.4)) contains the inverse of

$$\frac{\min\left(\varepsilon, \Delta_\lambda^{(u)}\right)}{s\|H\|_{\max}}. \quad (6.54)$$

Assuming the worst-case scenario in which, $\Delta_\lambda^{(u)} = \Delta_\lambda$, and we want to guaranty a separation of all eigenvalues, the desired tolerance ε is required to satisfy $\varepsilon < \Delta_\lambda$. We further know that the 1-norm $\|H\|_1 = \max_v \sum_u |H_{uv}|$ is smaller than $s\|H\|_{\max}$. Thus, we have

$$\frac{\varepsilon}{s\|H\|_{\max}} \leq \frac{\Delta_\lambda}{s\|H\|_{\max}} \leq \frac{\Delta_\lambda}{\|H\|_1} \leq \frac{\Delta_\lambda}{\lambda_{\max}}, \quad (6.55)$$

where we used the upper limit for the maximum eigenvalue $\lambda_{\max} \leq \|H\|_1$. Hence, the number of oracle queries scales with the ratio between the smallest eigenvalue gap and the maximum eigenvalue. This, in the general case, increases polynomially with the number of eigenvalues that have to fit between 0 and λ_{\max} , and accordingly it increases polynomially with the number of oscillators N .

The kinds of problems for which we can expect a polylogarithmic scaling in N of the total query complexity necessarily need to satisfy $\lambda_{\max}/\Delta_\lambda^{(u)} = \mathcal{O}(\text{polylog}(N))$ in our rigorous worst case scaling analysis. On the one hand, the random glued-trees problem we discuss in Ref. [2] has this property, which ultimately originates from its very large degeneracy. This shows that there exists well-defined problems for which a large exponential speedup is possible. However, the task there is not the determination of response functions. On the other hand, if we simply consider a 1D periodic chain with N oscillators and unit spring constants and masses, the eigenvalues can be obtained analytically and are given by [109]

$$\lambda_j = \frac{2}{\sqrt{N}} \left(1 - \cos\left(\frac{2\pi(j-1)}{N}\right)\right),$$

for $j \in [N]$. For this problem $s\|H\|_{\max}$ is bounded, namely $s\|H\|_{\max} = 6$, but for large N the gap for this problem scales as $\Delta_\lambda = \mathcal{O}(N^{-5/2})$, yielding eventually a polynomial scaling of the number of queries with N . Notice that these considerations have no influence on the scaling with the tolerance on the weights δ , which is δ^{-3} if we take it as an independent parameter (see Eq. (6.4)).

6.4.2 Possible improvements

Possible improvements in the scaling in the δ parameter could be achieved combining our algorithm with the quantum amplitude estimation subroutine [57]. Additionally, in the previous discussion, we have not restricted the domain of interest of the eigenvalues that are in general between 0 and λ_{\max} . However, in practice we might be interested in evaluating the response function only in a certain frequency range for which only eigenvalues $\lambda_j \in [\lambda_l, \lambda_r]$ matter. In fact, we might know in advance the typical frequency spectrum of the forcing term, as it happens in the milling case discussed in Chapter 3. In this case, we can approximate the response function as

$$G_{uu}(\nu) \approx \frac{1}{m_u} \sum_{j \in \mathcal{J}} \frac{W_{uj}^2}{\lambda_j + \nu^2}, \quad (6.56)$$

for $\text{Im}(\nu) \in [\lambda_l, \lambda_r]$ and $\mathcal{J} = \{j : \lambda_j \in [\lambda_l, \lambda_r]\}$. Thus, it might be beneficial to apply a filter to the quantum state $|u\rangle$ to remove all the contributions from eigenvalues that are outside the interval of interest $[\lambda_l, \lambda_r]$, and produce an initial, filtered state $|u_f\rangle$ such that

$$|u_f\rangle \sim \sum_{j \in \mathcal{J}} W_{uj} |\lambda_j\rangle. \quad (6.57)$$

Quantum eigenvalue filtering has been already studied in the literature [18, 110, 111] and could find an application in our problem. We note that also for classical algorithms it is possible to restrict the eigenvalue search in a specific range using filter diagonalization methods [112–115]. We propose one filter method and its speed-up in Chapter 7.

Finally, we point out that our algorithm has some connection with the vector fitting algorithm [116], where the goal is, given K observations of $G_{uu}(\nu)$ at $\nu = i\omega^{(k)}$, $k \in [K]$, to output the best estimate of the weights W_{uj}^2 , usually called residues, and the eigenvalues λ_j . In our case, we want to achieve the same task, but using the data obtained from running QPE multiple times to estimate the weights and the eigenvalues.

Partial Eigenpair Solver

Extracting all eigenvalues, paired with their probabilities, requires a large sample number (see Eq. (6.6)). A total of $O\left(\delta^{-2} \ln\left(N_\lambda^{(u)} \zeta^{-1}\right)\right)$ samples are required to guarantee an estimation of $|W_{uj}|^2$ within tolerance δ and failure probability of ζ for N eigenvalues. However, the eigenvector \mathbf{W}_j is normalized which means that the average $|W_{uj}|^2$ and with it the required tolerance $\delta = \gamma |W_{uj}|^2 = O(\gamma N^{-1})$ decreases with N . γ is the relative tolerance without hidden N dependency. This gives a sampling overhead of

$$t_s = O\left(\frac{N^2}{\gamma^2} \ln\left(\frac{N_\lambda^{(u)}}{\zeta}\right)\right), \quad (7.1)$$

next to the runtime of the QPE. This is only competitive with classical algorithms if one has knowledge allowing to choose a u with large $|W_{uj}| = O(1)$. Therefore, we consider a partial eigenpair solver (PES) in order to improve on the runtime in the following. This yields in another of our key results, an algorithms that beats classical alternatives in memory and runtime.

7.1 Efficient sub-problem

The algorithm, described in Chapter 5, calculates the complete set of all eigenpairs of H , which is why refer to it as complete eigenpair solver (CES) in this chapter. For many problems, it is sufficient to focus only on a small subset of all eigenpairs (e.g. the use case described in Chapter 3 requires only the lowest eigenpairs). We call the corresponding routine partial eigenpair solver (PES). The CES is only complete due to the initial preparation of a superposition of all eigenstates $|W_j\rangle$. We suggest the addition of an amplitude amplification (AA) (cf. Fig. 7.1.1 and Refs. [31, 57]) that returns mostly eigenstates with eigenvalues within a given interval $[\lambda_l, \lambda_r)$

$$\text{AA} \sum_{j=1}^N W_{uj} |W_j\rangle = c \sum_{j \in \mathcal{J}} W_{uj} |W_j\rangle + \sqrt{1-c^2} \sum_{j \notin \mathcal{J}} W_{uj} |W_j\rangle, \quad (7.2)$$

where $\mathcal{J} = \{j : \lambda_j \in [\lambda_l, \lambda_r)\}$ and c is an AA dependent coefficient. This increases the probability of measuring a state $|W_j\rangle : j \in \mathcal{J}$ to $|cW_{uj}|^2$. Here, c should be chosen such that

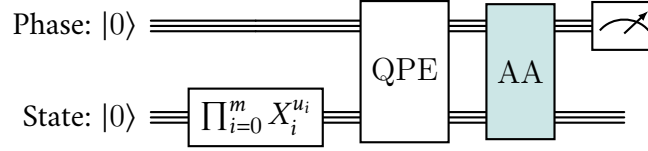


Figure 7.1.1: Eigenpair solver based on quantum phase estimation (QPE) (cf. Chapter 5). Originally, the state preparation consists of a computational basis state encoding via NOT gates. We extend it by an amplitude amplification (AA) (in blue) reducing the number of solutions.

$|cW_{uj}|^2$ and its tolerance $\delta = \gamma|cW_{uj}|^2$ is independent of N . This is given for $c = \mathcal{O}(Nk^{-1})$, where k is the number of eigenvalues within $[\lambda_l, \lambda_r)$. It remains a sampling overhead of

$$t'_s = \mathcal{O}\left(\frac{k^2}{\gamma^2} \ln\left(\frac{k}{\zeta}\right)\right). \quad (7.3)$$

Small enough intervals $[\lambda_l, \lambda_r)$ (and with it k) make this negligible in contrast to the runtime of the PES

$$t_{\text{PES}} = \mathcal{O}(t_{\text{CES}} + t_{\text{AA}}). \quad (7.4)$$

t_{CES} and t_{AA} denote the runtimes of the PES and the AA respectively.

7.2 Eigenvalue based amplitude amplification

The implementation of AA requires two reflection operators (cf. Section 2.6). One that reflects around the set of *good* states (i.e. $|W_j\rangle, \forall j \in \mathcal{J}$) and another that reflects around the original state. We identify the *good* states by their corresponding eigenvalues $\lambda_j \in [\lambda_l, \lambda_r)$ to which we get access via the phase register of the QPE. The test $\lambda_j \in [\lambda_l, \lambda_r)$ or as a substitute $\varphi_j \in [\varphi_l, \varphi_r)$ with $\varphi_{l/r} = f(\lambda_{l/r})$ (cf. Eq. (5.6)), can be implemented via a series of quantum-classical adders qcADD_l and their inverse subtractors $\text{qcSUB}_l = \text{qcADD}_l^\dagger$ as shown in Fig. 7.2.1. It reads the m -qubit phase register from Fig. 7.1.1 next to 3 additional qubits and subtracts φ_l from $|0, \varphi_j\rangle$ where the additional qubit is treated as most significant bit (msb.). This gives us

$$|\varphi_j - \varphi_l \bmod 2^{m+1}\rangle = |\text{sgn}_b(\varphi_j - \varphi_l), \varphi_j - \varphi_l \bmod 2^m\rangle. \quad (7.5)$$

The first qubit contains now the opposite of the desired $\neg \text{sgn}_b(\varphi_j - \varphi_l)$ (i.e. $\varphi_j \geq \varphi_l$). A simple NOT gate X solves this, and we have

$$|\neg \text{sgn}_b(\varphi_j - \varphi_l), \varphi_j - \varphi_l \bmod 2^m\rangle. \quad (7.6)$$

An adder qcADD_{φ_l} applied just to the main register returns its original state

$$|\neg \text{sgn}_b(\varphi_j - \varphi_l), \varphi_j \bmod 2^m\rangle. \quad (7.7)$$

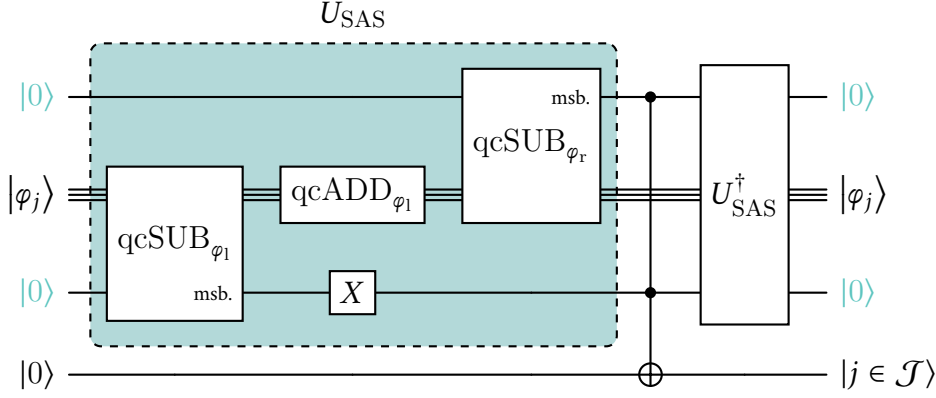


Figure 7.2.1: Quantum routine preparing the result of the boolean test $j \in \mathcal{J}$ for given φ_j in a target qubit. We use quantum adder qcADD_l and subtractor qcSUB_l to compare φ_l with the left and right limits defining $\mathcal{J} = \{j : \varphi_j \in [\varphi_l, \varphi_r]\}$ before combining the results with a Toffoli gate. At last, the top three registers are returned into their original state with the Hermitian conjugate of the first three steps.

Now, we use another ancilla qubit as msb. of the main register and subtract φ_r

$$\begin{aligned} |\neg \text{sgn}_b(\varphi_j - \varphi_l), \varphi_j - \varphi_r \bmod 2^{m+1}\rangle \\ = |\neg \text{sgn}_b(\varphi_j - \varphi_l), \text{sgn}_b(\varphi_j - \varphi_r), \varphi_j - \varphi_r \bmod 2^m\rangle. \end{aligned} \quad (7.8)$$

A Toffoli gate controlled by $|\neg \text{sgn}_b(\varphi_j - \varphi_l), \text{sgn}_b(\varphi_j - \varphi_r)\rangle$ and applied to another qubit combines the two limits into $\varphi_j \in [\varphi_l, \varphi_r]$, giving us

$$|\varphi_j \in [\varphi_l, \varphi_r], \neg \text{sgn}_b(\varphi_j - \varphi_l), \text{sgn}_b(\varphi_j - \varphi_r), \varphi_j - \varphi_r \bmod 2^m\rangle, \quad (7.9)$$

where $\varphi_j \in [\varphi_l, \varphi_r]$ is either true or false interpreted as 1 or 0 respectively. The Hermitian conjugate of the two adders, the one subtractor, and the NOT gate return the main register and the two ancilla qubits into their original state

$$|\varphi_j \in [\varphi_l, \varphi_r]\rangle |0\rangle^{\otimes 2} |\varphi_j\rangle = |j \in \mathcal{J}\rangle |0\rangle^{\otimes 2} |\varphi_j\rangle. \quad (7.10)$$

In this form, marking the *good* states (see R_G introduced in Section 2.6) is as simple as one application of a Z gate to the target qubit.

The second reflection R_{state} (around the original state, i.e. the state after the CES, cf. Fig. 7.1.1) requires the CES plus a reflection along the zero state.

$$R_{\text{state}} = \text{CES}(2|0\rangle\langle 0| - I)\text{CES}^\dagger \quad (7.11)$$

In worst case we start with a probability of $\sim k/N$ in the *good* states. The amplification requires $O(\sqrt{N/k})$ cycles [31, 57] and with it repetitions of the CES leading to

$$t_{\text{PES}} = O\left(\sqrt{\frac{N}{k}} t_{\text{CES}}\right) = O\left(\sqrt{\frac{N}{k}} s \|H\|_{\max} \max\left(\frac{1}{\varepsilon}, \frac{k}{\gamma \Delta_\lambda^{(u, \mathcal{J})}}\right)\right). \quad (7.12)$$

In the second step we replace the runtime t_{CES} with the query complexity Eq. (6.5), were, $\|H\|_{\max} = \max_{uv} |H_{uv}|$, ε denotes the eigenvalue tolerance and $\Delta_{\lambda}^{(u, \mathcal{T})}$ the minimum gap between eigenvalues $\lambda_j \in [\lambda_l, \lambda_r)$ for which $W_{uj} \neq 0$.

7.3 Comparison with classical algorithms

As we have modified the eigenpair solver proposed in Chapter 5 to be restricted to a small subset of eigenpairs, our new method is not suitable for the computation of all eigenvalues. Therefore, a comparison with the QR method ($t_{\text{QR}} = O(N^2s)$) [67, 117–119], the state-of-the-art method for complete eigenpair calculations, would be unfair. Instead, we should focus on classical partial eigenpair solvers that beat the QR method for small eigenpair subsets.

Starting with the power method, a routine based on the repetitive multiplication of the matrix H with a random vector $\mathbf{w} = \sum_{j=1}^N c_j \mathbf{W}_j$. The matrix amplifies the contribution of the largest eigenvector \mathbf{W}_1 yielding a good approximation of the same after a few iterations. This method is mainly dominated by the sparse matrix-vector multiplication requiring $O(Ns)$ basic arithmetic operations. The error of this method ε decreases with an increasing number of iterations l

$$\varepsilon \propto \left(\frac{\lambda_2}{\lambda_1} \right)^l, \quad (7.13)$$

where $\lambda_1 > \lambda_2$ are the two largest eigenvalue. This yields in a total runtime of

$$t_{\text{Power}} = O \left(Nsk \frac{\log \varepsilon}{\log \left(\frac{\lambda_2}{\lambda_1} \right)} \right). \quad (7.14)$$

Here, we find an additional factor of k as this method finds only one eigenpair at a time. In order to find the second-largest eigenvalue λ_2 one has to damp the largest eigenvalue in the matrix $H \rightarrow H - \lambda_1 \mathbf{W}_1 \mathbf{W}_1^\dagger$. This is to repeat until one has the k largest eigenvalues. We recommend Refs. [117, 118] for more details about the power method.

Compared with Eq. (7.12), our PES outperforms the classical power method quadratically in N . Again, however, we are interested in the worst-case scenario in which the eigenvalues are spread homogeneously and with it $\varepsilon = O(N^{-1})$ and $\lambda_2/\lambda_1 = O(1 - N^{-1})$. It is still difficult to compare Eqs. (7.12) and (7.14), which is why we show their scaling graphically in Fig. 7.3.1. It is clear that our PES outperforms the power method for non-trivial problem sizes. Furthermore, the non-optimal QR method is also faster than the power method due to its strong scaling with λ_2/λ_1 . We should further mention, that we can not choose an arbitrary interval $[\lambda_l, \lambda_r)$ with the power method. The power method is limited to the k largest eigenvalues if we have no fast access to H^{-1} .

The power method is beaten by multiple other classical routines nowadays. One of them is the Lanczos algorithm, which is based on the power method. It is able to find a subset of all eigenpairs fast by forming a Krylov subspace. This means we replace $H \in \mathbb{R}^{N \times N}$ with a smaller

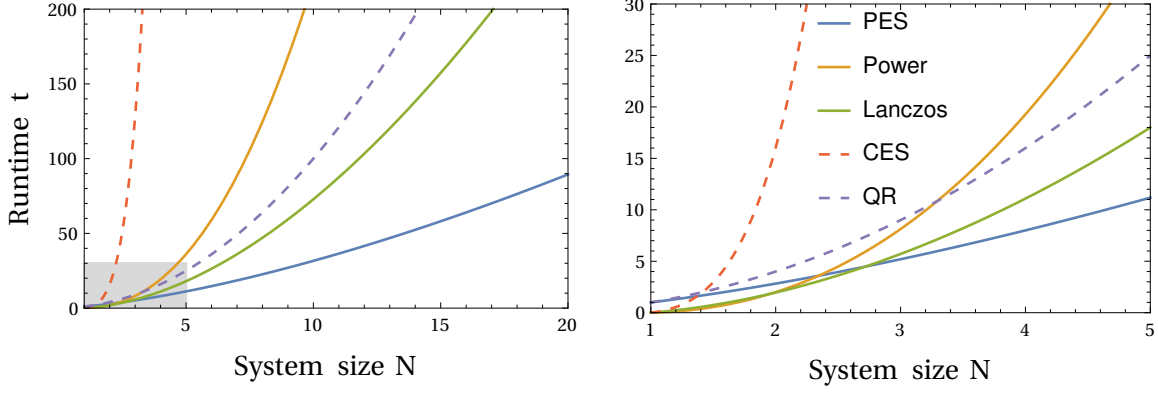


Figure 7.3.1: Runtime t comparison for the partial eigenpair solver (PES), the power method, the Lanczos method, the complete eigenpair solver (CES), and the QR method under worst conditions (i.e. $\varepsilon, \Delta_{\lambda}^{(u, \mathcal{T})}, \delta, (1 - \lambda_2/\lambda_1) \propto N^{-1}$ with N the matrix size). All other parameters are set to 1. The plot on the right shows the regime illustrated in gray in the left.

tridiagonal matrix $T \in \mathbb{R}^{l \times l}$ that shares the largest and the smallest eigenvalues with H up to an error. This is also an iterative process with an average error bounded by [120]

$$\varepsilon = \mathcal{O}\left(\frac{\log^2 N}{l^2}\right). \quad (7.15)$$

Here, the error is averaged over all possible initial vectors \mathbf{w} . Similar to the power method, this requires matrix-vector multiplication ($\mathcal{O}(Ns)$) in each iteration, leading to a runtime of

$$t_{\text{Lanczos}} = \mathcal{O}\left(Ns \frac{\log N}{\sqrt{\varepsilon}}\right). \quad (7.16)$$

We omit the additional steps necessary to find all eigenvalues of T using e.g. the QR method as its runtime $t_{\text{QR}} = \mathcal{O}(l^2)$ scales slower than the space reduction. The classical Lanczos algorithm is described in detail in Refs. [117–119].

The worst case scenario means again, that we have $\varepsilon = \mathcal{O}(N^{-1})$, which leads to runtime similar to Eq. (7.12) up to logarithmic terms. We show a comparison of both in Fig. 7.3.1 with our PES surpassing the Lanczos method. One has to keep in mind that the error bound in Eq. (7.15) is only valid for the largest eigenvalue and increases for the second or even the k -th largest eigenvalue. This brings us to the big disadvantage of the Lanczos method, which is its limitation to the extreme eigenvalues in contrast to our algorithm.

Reduced Eigenvalue Solver on NISQ Hardware

State-of-the-art of quantum computers are error-prone [121]. This limits the depth of quantum algorithms to small numbers of quantum operations before randomness dominates the system. Quantum algorithms with large numbers of quantum gates such as the HHL algorithm¹ [49] or the QPE are therefore not suitable for current devices. However, in the following, a minimal hybrid routine is described which is able to estimate the eigenvalue of a matrix H on NISQ devices. This should provide insight into the feasibility of more complex algorithms, although we can not expect any speed-up.

8.1 The NISQ-friendly hybrid algorithm

The measurement of quantum superposition leads to a collapse of the wave function into a random basis state. If one repeats the generation of this superposition followed by a measurement one can use this to estimate the expectation value $E(P)$ of the measured observable P . This makes them natural eigenvalue solver. For a quantum computer, those observables are, typically, the Pauli matrices $\sigma_i \in \{\sigma_x, \sigma_y, \sigma_z\}$ and their tensor products. An arbitrary matrix H of size $N \times N$ can be decomposed into a linear combination of those observables

$$H = \sum_{i=1}^{N^2} g_i P_i, \quad g_i = \frac{1}{N} \text{tr}(P_i H). \quad (8.1)$$

This allows us to estimate the expectation values of the single observables P_i for a given eigenvector \mathbf{v}_k of H and construct the corresponding eigenvalue λ_k of H by summing over the expectation values with the corresponding weights g_i

$$\lambda_k = E_k(H) = \sum_{i=1}^{N^2} g_i E_k(P_i). \quad (8.2)$$

This chapter has been published in the preprint “Stefan Schröder et al. “A methodical approach to evaluate the potential of Quantum Computing for Manufacturing Simulations”. In: *arXiv e-prints* (2024). arXiv: 2408.08730”

¹Named after the authors: Harrow, Hassidim and Lloyd

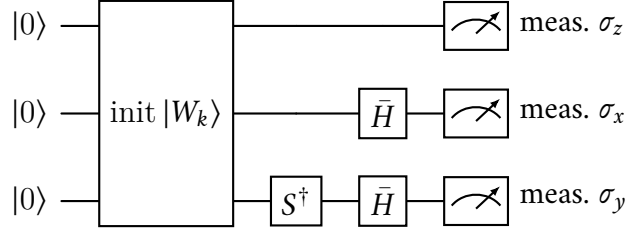


Figure 8.1.1: Exemplary three-qubit circuit for step 3 of the full algorithm. The black box on the left initializes the quantum register by amplitude encoding \mathbf{W}_k . We illustrate the transformations necessary to measure in the three different Pauli bases on the right.

The quantum part of the algorithm is reduced to the amplitude encoding of \mathbf{W}_k in a quantum register followed by a measurement in the corresponding Pauli-basis as illustrated in Fig. 8.1.1.

The full routine for the computation of the eigenvalues λ_k consist of four steps:

1. Compute the eigenvectors \mathbf{W}_k of H on classical hardware.
2. Compute $g_i, \forall i \in [N^2]$ on classical hardware.
3. For all $P_i \in \{P_1, P_2, \dots, P_{N^2}\}$, encode the eigenvector \mathbf{W}_k in the amplitudes of a quantum register with $\lceil \log_2 N \rceil$ qubits and measure P_i multiple times.
4. Sum over the sampled expectation values $E_k(P_i)$ with the weights g_i to have λ_k .

The second step is the biggest bottleneck requiring $O(N^4)$ ($O(N^3s)$ for s -sparse matrices) classical arithmetic operations. From a quantum perspective, the most critical part is the amplitude encoding of the N entries of \mathbf{W}_k in step 3, as this requires to execute $O(N)$ rotation gates on a quantum computer in the worst case. The more gates applied, the stronger the effect of the noise. This means, that the accuracy of the eigenvalue estimations decrease with the size of the matrix H .

8.2 Benchmarking with minimal geometries

We tested the algorithm on a superconducting quantum computer from IBM². For this, a system of coupled oscillators for different geometries was considered and their resonance frequencies $\omega_k = \sqrt{\lambda_k}$ were computed. To be more precise, one-dimensional chains of coupled oscillators of various length and three different 3D toy models of compressor blades (see Figure 8.2.1). The total number of oscillators in all those systems were kept small to minimize the effect of the noise.

²IBM Quantum System One at Ehningen Germany: Number of qubits: 27; Coherence time $\sim 150 \mu s$; Single qubit gate error $\sim 0.025 \%$; Two qubit gate error $\sim 0.7 \%$; Quantum Volume: 64

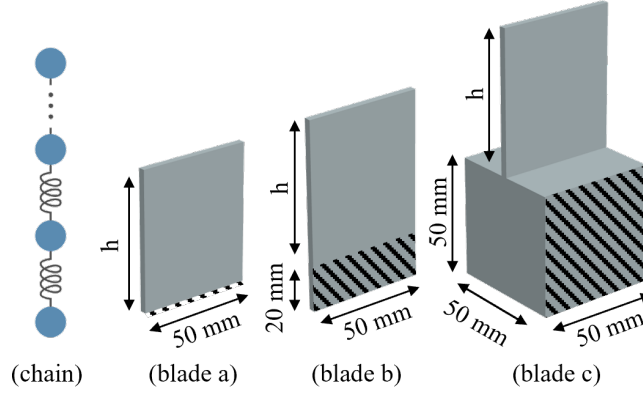


Figure 8.2.1: Investigated simplified blade geometry: A chain of 2-64 oscillators (1-6 qubits), one blade geometry with 12 oscillators (blade a, 4 qubits) and two blade geometries with 24 oscillators allocated to every node and element of an arbitrary mesh (blade b and c, 5 qubits). The shaded areas represent fixed boundary conditions which reduced the total number of oscillators. The blade height h is varied between $h = 10$ mm and $h = 60$ mm in 10 mm steps while keeping the oscillator number constant.

When computing the resonance frequencies on noisy quantum devices only an estimation $\tilde{\lambda}_k$ is achieved which lies between the exact eigenvalue λ_k and the expectation value of the fully mixed or noise dominated state λ_{mixed} . The eigenvalue of the fully mixed state is equal to g_1 which corresponds to $P_1 = I_N$. All other contributions ($E(P_i) = 0, \forall i \neq 1$) vanish for the fully mixed state. In order to compare different geometries, the focus will be on the error $\varepsilon_\lambda = \lambda_{\text{max}} - \tilde{\lambda}_{\text{max}}$ and a renormalization of it with the width $\Delta_\lambda = \lambda_{\text{max}} - \lambda_{\text{mixed}}$. Here we chose to concentrate on the maximum eigenvalue λ_{max} which allows us to estimate the maximum errors as $\lambda_{\text{max}} > \lambda_{\text{mixed}}$ for the described problems.

The relative error $\varepsilon_\lambda \Delta_\lambda^{-1}$ for the previously described geometries as a function of the number of quantum gates necessary to encode \mathbf{W}_k are shown in Figure 8.2.2. A distinction is made here between the plain computation as described in the algorithm in the left and an optimized version in the right graph. For the optimization, we used compilation and error mitigation methods³, provided by IBM's Qiskit Primitives. This shows the full potential of this device. More details about error mitigation methods are gathered in Ref. [122].

8.3 Benchmarking results

The algorithm proposed achieves the ideal resonance frequencies for minimum gate numbers. However, the effect of noise dominates fast, and the results are, after only ~ 100 gates, closer to

³ Here `optimization_level=3` and probabilistic error cancellation (`resilience_level=3`) is used. More information in <https://docs.quantum.ibm.com/run/configure-runtime-compilation> and <https://docs.quantum.ibm.com/run/configure-error-mitigation>

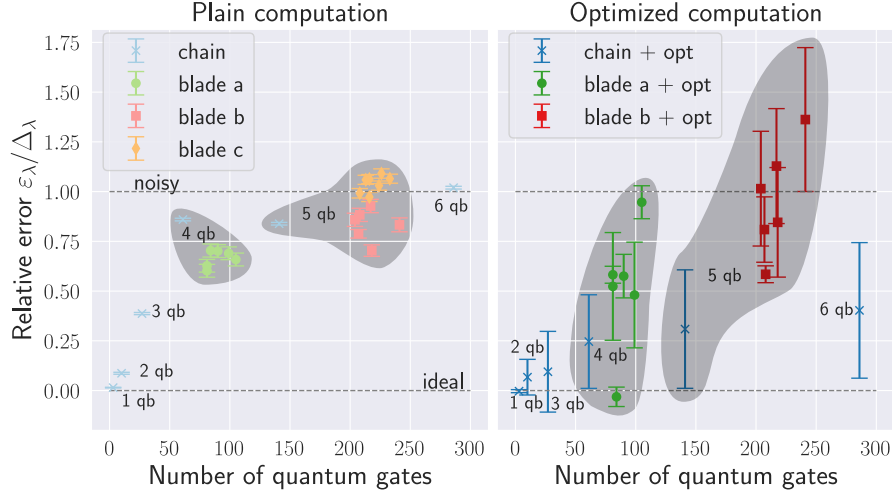


Figure 8.2.2: Relative eigenvalue error $\varepsilon_\lambda \Delta_\lambda^{-1}$ for different geometries (see Figure 8.2.1 and oscillator numbers). In the left plot, the plain computation on a noisy quantum machine is shown. On the right state-of-the-art circuit optimization is used and error mitigation methods³ provided by IBM’s Qiskit Primitives. The error bars here represent statistical errors over 100 samples for chain and 1000 samples for blade geometries. Systems with more than 6 qubits (= 6 qb) were fully noise dominated in the tests.

the random results of fully mixed states than to the desired values. The gate number increases for all geometries with the system size represented in the required qubit number. This shows that the system size is more important for the accuracy than the form of the geometry. Even state-of-the-art optimization methods³ do not manage to mitigate the effect of noise drastically. It seems like the simple geometry of the linear coupled oscillator deals slightly better with optimization methods. However, due to the increase in statistical errors from the error mitigation, this is not guaranteed, and larger sample sizes are necessary to achieve the same statistical accuracy.

Conclusion

In the core of this thesis, we have presented a quantum algorithm for obtaining response functions of systems of coupled harmonic oscillators. These kinds of problems also effectively emerge in the study of elastic structures, after discretization, as well as other areas of interest. The particular analytical structure of these functions makes them suitable for a quantum computer. The algorithm is based on the QPE subroutine, that we use as an eigenpair solver. Importantly, the state preparation at the input of the algorithm just amounts to the preparation of a product state. This allows us to analyze the scaling of the algorithm in terms of space and queries of the matrix oracles. While the algorithm achieves an exponential saving in space, the scaling of the total number of oracles queries is problem-dependent and can be polynomial in the number of qubits only when very specific conditions are met. We identify such a list of sufficient conditions leading to a rigorous worst case guarantee, but also leave open the possibility for improved practical scaling when a priori information about the structure of the problem is available. In addition, the scaling is mostly limited by statistical error coming from the estimation of the weights of the response functions.

Furthermore, we have provided a comprehensive discussion of the implementation of quantum oracles used in the block-encoding of matrices emerging in finite element analysis. Starting from the basic quantum adder, and using fixed-point arithmetic, we have constructed explicit quantum circuits for the calculation of the necessary functions, which in the specific case require the evaluation of a polynomial combined with the square root. In the appendix, we provided circuits for the implementation of logical operations such as comparison, conjunction and disjunction, that are also needed in this setting. All these subroutines might be of general interest beyond the application to the construction of quantum oracles in FEM.

In the specific case of normal mode analysis of elastic structures, the problem is completely specified by the $N \times N$ mass \mathbf{M} and stiffness \mathbf{K} matrices, which we show how to obtain using finite element analysis in arbitrary dimensions and general geometries. We also give them explicitly for the simple one-dimensional case. Our scaling analysis reveals, that under reasonable

Parts of this chapter have been published in the preprint “Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405 . 08694”. Others are shared in the preprint “Sven Danz, Tobias Stollenwerk, and Alessandro Ciani. “Quantum oracles for the finite element method”. In: *arXiv e-prints* (2025). arXiv: 2504 . 19827”. And last parts are from the preprint “Stefan Schröder et al. “A methodical approach to evaluate the potential of Quantum Computing for Manufacturing Simulations”. In: *arXiv e-prints* (2024). arXiv: 2408 . 08730”.

assumptions on the approximation of the geometry, the construction of the oracles requires $\mathcal{O}(\text{polylog}(N))$ ancilla qubits and similarly has $\mathcal{O}(\text{polylog}(N))$ runtime. This shows, that while the study of practical quantum advantages in FEM problems is still an open question, oracles should not be considered fundamental bottlenecks for potential polynomial or exponential quantum advantages from a complexity theoretic point of view.

Next to the complete eigenpair solver, we propose a partial eigenpair solver (PES) that computes only a small subset of all eigenpairs, which is sufficient for a large family of use cases. The main addition is an eigenvalue filter, based on amplitude amplification, that improves the runtime polynomial. The introduction of our PES mitigates the error dependency and allows us to not only compete with classical alternatives, but to beat them in a worst-case comparison in terms of memory requirements, runtime and versatility. Possible improvements of the algorithm could be obtained by combining the algorithm with quantum amplitude estimation instead of just amplitude amplification.

While we performed a basic scaling analysis, we believe that a deeper understanding of our algorithm and its potential can be obtained by studying its performances in practice. This could be simply understood via basic classical simulations that aim at obtaining the minimum, relevant eigenvalue gap for some class of oscillator problems. Such analysis would also shed light on the kinds of problems for which large advantages are possible. Additionally, a practical study of the algorithm performances using methods developed for the estimation of multiple eigenvalues [25, 100, 102] tailored to our problem, would be beneficial. Finally, an open question is whether estimates of the response functions can also be obtained using time-evolution based algorithms for coupled harmonic oscillator as in Ref. [35].

The majority of this thesis focuses on fault-tolerant quantum operations knowing that currently available quantum computers are still affected by noise, making long computations impossible. Nevertheless, we tested a reduced NISQ-friendly hybrid eigenvalue solver on a superconducting quantum computer from IBM to evaluate possibilities with state-of-the-art hardware. It was possible to estimate the resonance frequencies of small industry-related problems. Ongoing improvements in the accuracy of quantum hardware will increase the number of quantum gates that can be executed, and therefore the size of the problems that can be addressed. However, the presented hybrid subroutine cannot achieve a quantum advantage over classical algorithms because it computes N^2 weights classically, which is time-consuming. More advanced algorithms such as the proposed QPE based eigenpair solver are required for this.

Bibliography

- [1] Stefan Schröder et al. “An optimization approach for a milling dynamics simulation based on Quantum Computing”. In: *Procedia CIRP* 121C (2024), pp. 13–18. DOI: 10.1016/j.procir.2023.09.223.
- [2] Sven Danz et al. “Calculating response functions of coupled oscillators using quantum phase estimation”. In: *arXiv e-prints* (2024). arXiv: 2405.08694.
- [3] Stefan Schröder et al. “A methodical approach to evaluate the potential of Quantum Computing for Manufacturing Simulations”. In: *arXiv e-prints* (2024). arXiv: 2408.08730.
- [4] Sven Danz, Tobias Stollenwerk, and Alessandro Ciani. “Quantum oracles for the finite element method”. In: *arXiv e-prints* (2025). arXiv: 2504.19827.
- [5] Wolfgang Maass et al. “QUASIM: Quantum Computing Enhanced Service Ecosystem for Simulation in Manufacturing”. In: *KI - Künstliche Intelligenz* (2024). ISSN: 1610-1987. DOI: 10.1007/s13218-024-00860-x.
- [6] Daniel S. Abrams and Seth Lloyd. “Quantum Algorithm Providing Exponential Speed Increase for Finding Eigenvalues and Eigenvectors”. In: *Phys. Rev. Lett.* 83 (24 1999), pp. 5162–5165. DOI: 10.1103/PhysRevLett.83.5162.
- [7] Bela Bauer et al. “Quantum Algorithms for Quantum Chemistry and Quantum Materials Science”. In: *Chemical Reviews* 120.22 (2020), pp. 12685–12717. ISSN: 0009-2665. DOI: 10.1021/acs.chemrev.9b00829.
- [8] Sam McArdle et al. “Quantum computational chemistry”. In: *Rev. Mod. Phys.* 92 (1 2020), p. 015003. DOI: 10.1103/RevModPhys.92.015003.
- [9] Alexander M. Dalzell et al. “Quantum algorithms: A survey of applications and end-to-end complexities”. In: *arXiv e-prints* (2023). arXiv: 2310.03011.
- [10] Alberto Peruzzo et al. “A variational eigenvalue solver on a photonic quantum processor”. In: *Nature Communications* 5.1 (2014), p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213.
- [11] Jules Tilly et al. “The Variational Quantum Eigensolver: A review of methods and best practices”. In: *Physics Reports* 986 (2022), pp. 1–128. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2022.08.003.
- [12] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011. ISBN: 9781107002173. DOI: 10.1017/CB09780511976667.
- [13] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. Boston, MA, USA: American Mathematical Society, 2002. ISBN: 0821832298.

- [14] R. Cleve et al. “Quantum algorithms revisited”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 339–354. doi: 10.1098/rspa.1998.0164.
- [15] Guang Hao Low and Isaac L. Chuang. “Hamiltonian Simulation by Qubitization”. In: *Quantum* 3 (2019), p. 163. issn: 2521-327X. doi: 10.22331/q-2019-07-12-163.
- [16] Guang Hao Low and Isaac L. Chuang. “Optimal Hamiltonian Simulation by Quantum Signal Processing”. In: *Phys. Rev. Lett.* 118 (1 2017), p. 010501. doi: 10.1103/PhysRevLett.118.010501.
- [17] András Gilyén et al. “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. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 193–204. isbn: 9781450367059. doi: 10.1145/3313276.3316366.
- [18] John M. Martyn et al. “Grand Unification of Quantum Algorithms”. In: *PRX Quantum* 2 (4 2021), p. 040203. doi: 10.1103/PRXQuantum.2.040203.
- [19] Ewin Tang and Kevin Tian. “A CS guide to the quantum singular value transformation”. In: *arXiv e-prints* (2023). arXiv: 2302.14324.
- [20] David Poulin et al. “Quantum Algorithm for Spectral Measurement with a Lower Gate Count”. In: *Phys. Rev. Lett.* 121 (1 2018), p. 010501. doi: 10.1103/PhysRevLett.121.010501.
- [21] Dominic W. Berry et al. “Improved techniques for preparing eigenstates of fermionic Hamiltonians”. In: *npj Quantum Information* 4.1 (2018), p. 22. issn: 2056-6387. doi: 10.1038/s41534-018-0071-5.
- [22] Lov Grover and Terry Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions”. In: *arXiv e-prints* (2002). arXiv: quant-ph/0208112.
- [23] Maris Ozols, Martin Roetteler, and Jérémie Roland. “Quantum rejection sampling”. In: *ACM Trans. Comput. Theory* 5.3 (2013). issn: 1942-3454. doi: 10.1145/2493252.2493256.
- [24] Sam McArdle, András Gilyén, and Mario Berta. “Quantum state preparation without coherent arithmetic”. In: *arXiv e-prints* (2022). arXiv: 2210.14892.
- [25] Thomas E O’Brien, Brian Tarasinski, and Barbara M Terhal. “Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments”. In: *New Journal of Physics* 21.2 (2019), p. 023022. doi: 10.1088/1367-2630/aafb8e.
- [26] Alicja Dutkiewicz, Barbara M. Terhal, and Thomas E. O’Brien. “Heisenberg-limited quantum phase estimation of multiple eigenvalues with few control qubits”. In: *Quantum* 6 (2022), p. 830. issn: 2521-327X. doi: 10.22331/q-2022-10-06-830.
- [27] Zhiyan Ding and Lin Lin. “Simultaneous estimation of multiple eigenvalues with short-depth quantum circuit on early fault-tolerant quantum computers”. In: *Quantum* 7 (2023), p. 1136. issn: 2521-327X. doi: 10.22331/q-2023-10-11-1136.

- [28] Rolando D Somma. “Quantum eigenvalue estimation via time series analysis”. In: *New Journal of Physics* 21.12 (2019), p. 123025. doi: 10.1088/1367-2630/ab5c60.
- [29] David Deutsch and Roger Penrose. “Quantum theory, the Church–Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (1985), pp. 97–117. doi: 10.1098/rspa.1985.0070.
- [30] David Deutsch and Richard Jozsa. “Rapid solution of problems by quantum computation”. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558. doi: 10.1098/rspa.1992.0167.
- [31] Lov K. Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC ’96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. doi: 10.1145/237814.237866.
- [32] Guang Hao Low and Isaac L. Chuang. “Optimal Hamiltonian Simulation by Quantum Signal Processing”. In: *Phys. Rev. Lett.* 118 (1 2017), p. 010501. doi: 10.1103/PhysRevLett.118.010501.
- [33] Andrew M Childs. “On the Relationship Between Continuous-and Discrete-Time Quantum Walk”. In: *Commun. Math. Phys* 294 (2010), pp. 581–603. doi: 10.1007/s00220-009-0930-1.
- [34] D. W. Berry and A. M. Childs. “Black-Box Hamiltonian Simulation and Unitary Implementation”. In: *Quantum Information and Computation* 12.1–2 (2012), pp. 29–62. ISSN: 1533-7146. doi: 10.26421/QIC12.1-2-4.
- [35] Ryan Babbush et al. “Exponential Quantum Speedup in Simulating Coupled Classical Oscillators”. In: *Phys. Rev. X* 13 (4 2023), p. 041041. doi: 10.1103/PhysRevX.13.041041.
- [36] Yasunori Lee and Keita Kanno. “Modal analysis on quantum computers via qubitization”. In: *arXiv e-prints* (2023). arXiv: 2307.07478.
- [37] Pedro C. S. Costa, Stephen Jordan, and Aaron Ostrander. “Quantum algorithm for simulating the wave equation”. In: *Phys. Rev. A* 99 (1 2019), p. 012323. doi: 10.1103/PhysRevA.99.012323.
- [38] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. “Preconditioned Quantum Linear System Algorithm”. In: *Phys. Rev. Lett.* 110 (25 2013), p. 250504. doi: 10.1103/PhysRevLett.110.250504.
- [39] Ashley Montanaro and Sam Pallister. “Quantum algorithms and the finite element method”. In: *Phys. Rev. A* 93 (3 2016), p. 032324. doi: 10.1103/PhysRevA.93.032324.
- [40] Tal Mor and Renato Renner. “Conjugate Coding”. In: *Natural Computing* 13.4 (2014), pp. 447–452. ISSN: 1572-9796. doi: 10.1007/s11047-014-9464-3.
- [41] Stephen Wiesner. “Conjugate coding”. In: *SIGACT News* 15.1 (1983), pp. 78–88. ISSN: 0163-5700. doi: 10.1145/1008908.1008920.

- [42] A. S. Holevo. "Bounds for the Quantity of Information Transmitted by a Quantum Communication Channel". In: *Probl. Peredachi Inf.* 9.3 (1973), pp. 3–11.
- [43] C. H. Bennett. "Logical Reversibility of Computation". In: *IBM Journal of Research and Development* 17.6 (1973), pp. 525–532. DOI: 10.1147/rd.176.0525.
- [44] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6 (1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/BF02650179.
- [45] Peter W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.
- [46] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172.
- [47] Eleanor Rieffel and Wolfgang Polak. *Quantum Computing: A Gentle Introduction*. 1st. Scientific and Engineering Computation. MIT Press, 2011. ISBN: 9780262526678.
- [48] Alexei Y. Kitaev. "Quantum measurements and the Abelian Stabilizer Problem". In: *Electron. Colloquium Comput. Complex.* TR96 (1995).
- [49] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Phys. Rev. Lett.* 103 (2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502.
- [50] Lin Lin. "Lecture Notes on Quantum Algorithms for Scientific Computation". In: *arXiv e-prints* (2022). arXiv: 2201.08309.
- [51] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. "Quantum arithmetic with the quantum Fourier transform". In: *Quantum Information Processing* 16.6 (2017), p. 152. ISSN: 1573-1332. DOI: 10.1007/s11128-017-1603-1.
- [52] Steven A. Cuccaro et al. "A new quantum ripple-carry addition circuit". In: *arXiv e-prints* (2004). arXiv: quant-ph/0410184.
- [53] Thomas G. Draper. "Addition on a Quantum Computer". In: *arXiv e-prints* (2000). arXiv: quant-ph/0008033.
- [54] Vlatko Vedral, Adriano Barenco, and Artur Ekert. "Quantum networks for elementary arithmetic operations". In: *Phys. Rev. A* 54 (1996), pp. 147–153. DOI: 10.1103/PhysRevA.54.147.
- [55] Rodney Van Meter and Kohei M. Itoh. "Fast quantum modular exponentiation". In: *Phys. Rev. A* 71 (5 2005), p. 052320. DOI: 10.1103/PhysRevA.71.052320.
- [56] Michael Kirkedal Thomsen and Holger Bock Axelsen. "Parallel Optimization of a Reversible (Quantum) Ripple-Carry Adder". In: *Unconventional Computing*. Ed. by Cristian S. Calude et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 228–241. ISBN: 978-3-540-85194-3.

- [57] Michele Mosca Gilles Brassard Peter Høyer and Alain Tapp. “Quantum amplitude amplification and estimation”. In: *Quantum Computation and Information*. Ed. by Jr. Samuel J. Lomonaco and Howard E. Brandt. Vol. 305. Contemporary Mathematics, 2002, pp. 53–74.
- [58] Michael Jung and Ulrich Langer. *Method of finite elements for engineers. An introduction to the numerical foundations and computer simulation*. Vol. 2nd revised and extended ed. Germany: Springer Fachmedien Wiesbaden, 2013.
- [59] Fritz Klocke. *Manufacturing Processes 1: Cutting*. Germany: Springer Berlin Heidelberg, 2011.
- [60] S Maslo et al. “Improving dynamic process stability in milling of thin-walled workpieces by optimization of spindle speed based on a linear parameter-varying model”. In: *Procedia CIRP* 93 (2020), pp. 850–855. doi: 10.1016/j.procir.2020.03.092.
- [61] J Munoa et al. “Chatter suppression techniques in metal cutting”. In: *CIRP Annals* 65 (2) (2016), pp. 785–808. doi: 10.1016/j.cirp.2016.06.004.
- [62] G Quintana and J Ciurana. “Chatter in machining processes: A review”. In: *International Journal of Machine Tools and Manufacture* 51 (5) (2011), pp. 363–376. doi: 10.1016/j.ijmachtools.2011.01.001.
- [63] E Budak et al. “Prediction of workpiece dynamics and its effects on chatter stability in milling”. In: *CIRP Annals* 61 (1) (2012), pp. 339–342. doi: 10.1016/j.cirp.2012.03.144.
- [64] Y Altintas et al. “Chatter stability of milling in frequency and discrete time domain”. In: *CIRP Journal of Manufacturing Science and Technology* 1 (1) (2008), pp. 35–44. doi: 10.1016/j.cirpj.2008.06.003.
- [65] Viktor Rudel et al. “Cloudbased process design in a digital twin framework with integrated and coupled technology models for blisk milling. Front. Manuf. Technol.” In: *Frontiers in Manufacturing Technology* 2,1021029 (2813-0359 2022). DOI: 10.3389/fmtec.2022.1021029.
- [66] P. Triverio et al. “Stability, Causality, and Passivity in Electrical Interconnect Models”. In: *IEEE Transactions on Advanced Packaging* 30.4 (2007), pp. 795–808. ISSN: 1521-3323.
- [67] William H. Press et al. *Numerical Recipes in C: The Art of Scientific Computing*. 2nd. USA: Cambridge University Press, 1992. ISBN: 0521437148.
- [68] Alexandre Ern and Jean-Luc Guermond. “Theory and practice of finite elements”. In: 2004.
- [69] Singiresu S. Rao. *The Finite Element Method in Engineering*. Butterworth-Heinemann, 2005.
- [70] J. Fish and T. Belytschko. *A First Course in Finite Elements*. Wiley, 2007.
- [71] R.D. Cook, D.S. Malkus, and M.E. Plesha. *Concepts and Applications of Finite Element Analysis*. Wiley, 1989. ISBN: 9780471847885.

- [72] F. Bampi and A. Morro. “A lagrangian density for the dynamics of elastic dielectrics”. In: *International Journal of Non-Linear Mechanics* 18.6 (1983), pp. 441–447. issn: 0020-7462. doi: 10.1016/0020-7462(83)90031-8.
- [73] Gerhard A. Holzapfel. “Nonlinear Solid Mechanics: A Continuum Approach for Engineering Science”. In: *Meccanica* 37.4 (2002), pp. 489–490. issn: 1572-9648. doi: 10.1023/A:1020843529530.
- [74] J. F. Nye. *Physical Properties of Crystals - Their Representation by Tensors and Matrices*. Oxford: Clarendon Press, 1985. isbn: 978-0-198-51165-6.
- [75] Martin T. Dove. “Acoustic modes and macroscopic elasticity”. In: *Introduction to Lattice Dynamics*. Cambridge Topics in Mineral Physics and Chemistry. Cambridge University Press, 1993, pp. 95–100.
- [76] Deren Han, H. H. Dai, and Liqun Qi. “Conditions for Strong Ellipticity of Anisotropic Elastic Materials”. In: *Journal of Elasticity* 97.1 (2009), pp. 1–13. issn: 1573-2681. doi: 10.1007/s10659-009-9205-5.
- [77] Zheng-Hai Huang and Liqun Qi. “Positive definiteness of paired symmetric tensors and elasticity tensors”. In: *Journal of Computational and Applied Mathematics* 338 (2018), pp. 22–43. issn: 0377-0427. doi: 10.1016/j.cam.2018.01.025.
- [78] Martin H. Sadd. *Elasticity: Theory, applications, and numerics*. Academic Press, 2014.
- [79] Allan F. Bower. *Applied Mechanics of Solids*. CRC Press LLC, 2010.
- [80] R.W. Newcomb. *Linear Multiport Synthesis*. McGraw-Hill electronic sciences series. McGraw-Hill, 1966.
- [81] Peter Russer. *Electromagnetics, Microwave Circuit and Antenna Design for Communications Engineering*. Boston: Artech House, 2006.
- [82] R. M. Foster. “A reactance theorem”. In: *Bell System Technical Journal* 3 (2 1924), pp. 259–267.
- [83] Simon E. Nigg et al. “Black-Box Superconducting Circuit Quantization”. In: *Phys. Rev. Lett.* 108 (24 2012), p. 240502. doi: 10.1103/PhysRevLett.108.240502.
- [84] A. Ciani, D. P. DiVincenzo, and B. M. Terhal. *Lecture Notes on Quantum Electrical Circuits*. Delft: TU Delft OPEN Publishing, 2024. doi: 10.59490/tb.85.
- [85] David Beckman et al. “Efficient networks for quantum factoring”. In: *Phys. Rev. A* 54 (2 1996), pp. 1034–1063. doi: 10.1103/PhysRevA.54.1034.
- [86] Thomas G. Draper et al. “A logarithmic-depth quantum carry-lookahead adder”. In: *arXiv e-prints* (2004). arXiv: quant-ph/0406142.
- [87] Craig Gidney. “Halving the cost of quantum addition”. In: *Quantum* 2 (2018), p. 74. issn: 2521-327X. doi: 10.22331/q-2018-06-18-74.
- [88] Christof Zalka. “Fast versions of Shor’s quantum factoring algorithm”. In: *arXiv e-prints* (1998). arXiv: quant-ph/9806084.

- [89] Alexandru Paler. “Quantum Fourier addition simplified to Toffoli addition”. In: *Phys. Rev. A* 106 (4 2022), p. 042444. doi: 10.1103/PhysRevA.106.042444.
- [90] F. Orts, G. Ortega, and E. M. Garzón. “An optimized quantum circuit for converting from sign–magnitude to two’s complement”. In: *Quantum Information Processing* 18.11 (2019), p. 332. issn: 1573-1332. doi: 10.1007/s11128-019-2447-7.
- [91] Thomas Häner, Martin Roetteler, and Krysta M. Svore. “Optimizing Quantum Circuits for Arithmetic”. In: *arXiv e-prints* (2018). arXiv: 1805.12445.
- [92] Himanshu Thapliyal et al. “Quantum Circuit Designs of Integer Division Optimizing T-Count and T-Depth”. In: *2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*. 2017, pp. 123–128. doi: 10.1109/iNIS.2017.34.
- [93] S. S. Gayathri et al. “T-Count Optimized Quantum Circuit Designs for Single-Precision Floating-Point Division”. In: *Electronics* 10.6 (2021). issn: 2079-9292. doi: 10.3390/electronics10060703.
- [94] S S Gayathri, R. Kumar, and Samiappan Dhanalakshmi. “Efficient Floating-point Division Quantum Circuit using Newton-Raphson Division”. In: *Journal of Physics: Conference Series* 2335.1 (2022), p. 012058. doi: 10.1088/1742-6596/2335/1/012058.
- [95] Michael Flynn. “On Division by Functional Iteration”. In: *Computers, IEEE Transactions on C-19* (1970), pp. 702–706. doi: 10.1109/T-C.1970.223019.
- [96] Stephane Beauregard. “Circuit for Shor’s algorithm using $2n+3$ qubits”. In: *arXiv e-prints* (2003). arXiv: quant-ph/0205095.
- [97] Archimedes Pavlidis and Dimitris Gizopoulos. “Fast quantum modular exponentiation architecture for Shor’s factoring algorithm”. In: *Quantum Info. Comput.* 14.7 & 8 (2014), pp. 649–682. issn: 1533-7146.
- [98] A. Parker and J.O. Hamblen. “Optimal value for the Newton-Raphson division algorithm”. In: *Information Processing Letters* 42.3 (1992), pp. 141–144. issn: 0020-0190. doi: 10.1016/0020-0190(92)90137-K.
- [99] Adriano Barenco et al. “Elementary gates for quantum computation”. In: *Physical Review A* 52.5 (1995). Publisher: American Physical Society, pp. 3458–3467. doi: 10.1103/PhysRevA.52.3457.
- [100] K. M. Svore, M. B. Hastings, and M. Freedman. “Faster phase estimation”. In: *Quantum Info. Comput.* 14.3–4 (2014), pp. 306–328. issn: 1533-7146.
- [101] Patrick Rall. “Faster Coherent Quantum Algorithms for Phase, Energy, and Amplitude Estimation”. In: *Quantum* 5 (2021), p. 566. issn: 2521-327X. doi: 10.22331/q-2021-10-19-566.
- [102] Nathan Wiebe and Chris Granade. “Efficient Bayesian Phase Estimation”. In: *Phys. Rev. Lett.* 117 (1 2016), p. 010503. doi: 10.1103/PhysRevLett.117.010503.
- [103] Daan Camps et al. “Explicit Quantum Circuits for Block Encodings of Certain Sparse Matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 45.1 (2024), pp. 801–827. doi: 10.1137/22M1484298.

- [104] C.A. Felippa. *Introduction to Finite Element Methods*. Department of Aerospace Engineering Sciences and Center for Aerospace Structures University of Colorado, 2003.
- [105] Qiskit contributors. *Qiskit: An Open-source Framework for Quantum Computing*. 2023. doi: 10.5281/zenodo.2573505.
- [106] Feng Qi, Senlin Guo, and Bai-Ni Guo. “Complete monotonicity of some functions involving polygamma functions”. In: *Journal of Computational and Applied Mathematics* 233.9 (2010), pp. 2149–2160. ISSN: 0377-0427. doi: 10.1016/j.cam.2009.09.044.
- [107] Wassily Hoeffding. “Probability Inequalities for Sums of Bounded Random Variables”. In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30. doi: 10.1080/01621459.1963.10500830.
- [108] A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. 2nd. Applications of Mathematics. New York: Springer, 1998.
- [109] Matthias Christandl et al. “Perfect State Transfer in Quantum Spin Networks”. In: *Phys. Rev. Lett.* 92 (18 2004), p. 187902. doi: 10.1103/PhysRevLett.92.187902.
- [110] Lin Lin and Yu Tong. “Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems”. In: *Quantum* 4 (2020), p. 361. ISSN: 2521-327X. doi: 10.22331/q-2020-11-11-361.
- [111] Robert M. Parrish and Peter L. McMahon. “Quantum Filter Diagonalization: Quantum Eigendecomposition without Full Quantum Phase Estimation”. In: *arXiv e-prints* (2019). arXiv: 1909.08925.
- [112] Michael R. Wall and Daniel Neuhauser. “Extraction, through filter-diagonalization, of general quantum eigenvalues or classical normal mode frequencies from a small number of residues or a short-time segment of a signal. I. Theory and application to a quantum-dynamics model”. In: *The Journal of Chemical Physics* 102.20 (1995), pp. 8011–8022. ISSN: 0021-9606. doi: 10.1063/1.468999.
- [113] Rongqing Chen and Hua Guo. “A general and efficient filter-diagonalization method without time propagation”. In: *The Journal of Chemical Physics* 105.4 (1996), pp. 1311–1317. ISSN: 0021-9606. doi: 10.1063/1.471997.
- [114] Vladimir A. Mandelshtam and Howard S. Taylor. “A low-storage filter diagonalization method for quantum eigenenergy calculation or for spectral analysis of time signals”. In: *The Journal of Chemical Physics* 106.12 (1997), pp. 5085–5090. ISSN: 0021-9606. doi: 10.1063/1.473554.
- [115] Tsutomu Ikegami, Tetsuya Sakurai, and Umpei Nagashima. “A filter diagonalization for generalized eigenvalue problems based on the Sakurai–Sugiura projection method”. In: *Journal of Computational and Applied Mathematics* 233.8 (2010), pp. 1927–1936. ISSN: 0377-0427. doi: 10.1016/j.cam.2009.09.029.
- [116] B. Gustavsen and A. Semlyen. “Rational approximation of frequency domain responses by vector fitting”. In: *IEEE Transactions on Power Delivery* 14.3 (1999), pp. 1052–1061. doi: 10.1109/61.772353.

- [117] James W. Demmel. *Applied Numerical Linear Algebra*. USA: Society of Industrial and Applied Mathematics, 1997.
- [118] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. 4th. USA: The John Hopkins University Press, 2013.
- [119] Lloyd N. Trefethen and David Bau. *Numerical linear Algebra*. USA: Society of Industrial and Applied Mathematics, 1997.
- [120] J. Kuczyński and H. Woźniakowski. “Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start”. In: *SIAM Journal on Matrix Analysis and Applications* 13.4 (1992), pp. 1094–1122. doi: 10.1137/0613066.
- [121] R. Barends et al. “Diabatic Gates for Frequency-Tunable Superconducting Qubits”. In: *Physical Review Letters* 123.21 (2019). ISSN: 1079-7114. doi: 10.1103/physrevlett.123.210501.
- [122] Zhenyu Cai et al. “Quantum error mitigation”. In: *Rev. Mod. Phys.* 95 (4 2023), p. 045005. doi: 10.1103/RevModPhys.95.045005.