CHRISTOPHER ZACHOW

THE QLS ALGORITHM FOR RADAR CROSS SECTION CALCULATION - A PARAMETER ESTIMATION FOR THE MATRIX ORACLE

DLR Deutsches Zentrum für Luft- und Raumfahrt

Universität zu Köln

THE QLS ALGORITHM FOR RADAR CROSS SECTION CALCULATION

–

A PARAMETER ESTIMATION FOR THE MATRIX ORACLE

MASTER THESIS

submitted in partial fulfillment of the requirements of

MASTER OF SCIENCE IN PHYSICS

by

CHRISTOPHER ZACHOW

born on February 20, 1993 in Grevenbroich, Germany

on

January 31, 2019 in Cologne

German Aerospace Center (DLR), SC-HPC and
University of Cologne, Institute for Theoretical Physics

Deutsches Zentrum
für Luft- und Raumfahrt

TITLE:
The QLS Algorithm for Radar Cross Section Calculation - A Parameter Estimation for the Matrix Oracle

AUTHOR:
Christopher Zachow

SUPERVISOR:
Prof. Dr. David Gross
Dr. Tobias Stollenwerk

LOCATION:
German Aerospace Center (DLR), SC-HPC and
University of Cologne, Institute for Theoretical Physics

SUBMISSION DATE:
January 31, 2019

ABSTRACT

Many problems cannot be solved analytically, thus numerical methods are required. It turns out that many problems are too complex and therefore beyond the reach of current algorithms and computers. *Quantum computation* might offer fundamental improvements in solving these problems. However, it is a difficult task to estimate the computational power of quantum algorithms. The aim of this thesis is to find applications related to aerospace especially well suited for quantum computation and estimate their possible quantum advantage. In particular we focus on the calculation of *radar cross sections*, which has been suggested as a promising candidate for *quantum speedup*. It turns out that the main problem lies in the input of classical data. Hence, the reimplementation of classical *finite element method* is needed, requiring a substantial amount of quantum resources. Asymptotically a quantum speedup is achievable but it is far beyond the reach of today's capabilities to successfully implement the quantum algorithm for this application. Additionally, a serious prediction regarding the timeframe is not foreseeable yet. Substantial breakthroughs are necessary to make quantum computation compete against classical computation in calculating radar cross sections using the finite element method.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

HHL Harrow, Hassidim, and Lloyd

FP floating point

EM electromagnetic

QC quantum computation

LUT lookup table

RCS radar cross section

FEM finite element method

QLS quantum linear system

QEC quantum error correction

QAE quantum amplitude estimation

QFT quantum Fourier transformation

QAA quantum amplitude amplification

VQE variational quantum eigensolver

QAO quantum approximate optimization

NISQ noisy intermediate state quantum

SPMV sparse matrix vector multiplication

QUBO quadratic binary unconstrained optimization

LHRS LUT-based hierarchical reversible logic synthesis

NOTATION

$\boldsymbol{x}$	vector
$\boldsymbol{X}$	matrix or vector field
$\hat{\boldsymbol{x}}$	unit vector in direction of $\boldsymbol{x}$
Ω	domain
$\partial\Omega$	domain boundary

NOMENCLATUR

λ	wavelength
ω	frequency
E_0	field strength
$\boldsymbol{k}$	wave vector
$\boldsymbol{n}$	surface normal
N	matrix dimension
τ	time

1

INTRODUCTION

In the last decades electronic data processing, and information theory in general, have become the game changers in industry and society. All areas of the human life have been influenced by these new technologies, leading to a new historic period, the information age.
The processing of huge amounts of data (Big Data) and the tackling of computational intensive problems (HPC) has led to the development of ever more powerful computers. Since classical computer architectures show specific limitations, additional components have been added. These so called *boosters* use architectures differing from their classical counterparts and are designed for a specific task (GPUs, FPGAs, etc.). However, they are still based on the same technology as classical computers.
Only recently, fundamentally new architectures have been realized. So far, most of them are only available in the laboratory, but given the current pace of scientific development, some of them might have the potential to become interesting alternatives to classical architectures. One of these new computing architectures is *quantum computation*. It was introduced a few decades ago but only recently gained traction by new hardware developments. It may offer a new way to tackle computational intensive problems. In general, quantum computation can be described as the processing of data using the concepts of quantum mechanics.
A typical research area in which computationally intensive problems arise is the area of aerospace. The problems range from aircraft design which involves *virtual prototyping*, satellite scheduling to space debris analysis. Exploiting the possible potential of quantum computation might allow for a better understanding, more complex simulations, and even more accurate solutions of these problems.

OUTLINE In this thesis we will do an exemplary study of a quantum algorithm that has been proposed to be applicable to a typical aerospace problem. We will derive an informed estimate of the required quantum resources which are necessary to make the quantum algorithm comparable to classical methods.
In particular we will study the *quantum linear system (QLS)* algorithm for solving systems of linear equations. Theoretically this algorithm scales logarithmic in the matrix dimension, thus showing exponential speedup compared to the best known classical methods. Additionally, a promising application based on this algorithm has been proposed, namely the calculation of *radar cross section (RCS)* using the *method*

of finite elements (FEM). It turns out that one of the main difficulties lies in the input of classical data. From a theoretical point of view this is solved by the *quantum oracle* formalism. Though, this is not satisfactory from a practical point of view, i.e. having the implementation in mind. A detailed analysis shows the requirement for a reimplementation of classical finite element method on the quantum computer. Therefore classical operations have to be emulated, consuming a substantial amount of quantum resources.
Even tough the quantum linear system algorithm shows exponential speedup in the matrix dimension, implementing it for the calculation of radar cross sections is far beyond the reach of today's capabilities, e.g. in terms of the maturity and size of today's quantum devices. Furthermore, substantial breakthroughs are needed, thus it is not foreseeable when such an implementation might become possible.

RELATED WORK The basis for the quantum linear system algorithm has been laid out by Harrow, Hassidim, and Lloyd in the year 2008 [1]. Originally this algorithm was named by its founders and therefore called HHL algorithm. Since the original algorithm has been altered by different authors, the class of new algorithms has been collected under the name *quantum linear system (QLS)* algorithms.
There are numerous suggested applications for this class of algorithms ranging from *differential equations* [2] to *support vector machines* [3]. Recently, the variations which have been tailored for applications in *machine learning* and *artificial intelligence* have gained high attention. However, the application to radar cross section as introduced by Clader et al. [4] is one of the few applications that have been considered in full detail. Most importantly, they took care of the input of classical data as required by this algorithm.
There is already one work comparable to this thesis, estimating the cost for reimplementing the finite element method on a quantum computer [5]. It has been published by Scherer et al. and uses the quantum programming language `quipper`. An important difference to this thesis is the employed method for the resource estimation. Supplementary, the additional cost arising from the target's shape is not considered in this work.
There is also comparable work for other quantum algorithms. As an example we want to mention the work by Roetteler et al. regarding the resource requirements for *Shor's* algorithm [6]. Nevertheless, there is still much room for additional research concerning the possible applications for quantum algorithms and their resource requirements.

CONTENTS This thesis is organised as follows: We start with a short introduction to *quantum computation* which is based on the textbook by Nielsen and Chuang [7]. In this chapter we will discuss basic building blocks, differences to classical computation and hardware realizations. Additionally, we lay out the conventions used in this thesis.
Further on, we will explicitly consider the quantum linear system (QLS) algorithm.

After getting to know the main algorithm, we familliarize with the different subalgorithms involved. Finally, we discuss the algorithmic complexity and challenging problems. Partially we will be able to offer solutions.
The classical implementation of a possible application for the QLS algorithm is the topic of the next chapter. More precisely, the calculation of *radar cross sections (RCSs)* using the *finite element method (FEM)* is considered. Starting with the differential formulation, the variational formulation is used to finally obtain a numerical solution. The chapter is completed by an example calculation.
The following chapter is used to estimate the potential for a successful implementation of the calculation of RCSs using the QLS algorithm. In particular, implementation requirements are discussed, leading to an estimation of the resource requirements for the input of the classical data. Consequently, an informed estimate in terms of consumed quantum resources is given.
In the last chapter we summarize our results and discuss possible future improvements.

2

QUANTUM COMPUTATION

Developed at the beginning of the 20th century, quantum mechanics is the fundamental theory for all physical interactions on small scales resulting in micro- and macroscopic phenomena. The most important concepts are *quantization*, *superposition* and *entanglement*. Described phenomena range from atomic spectra, superconductivity to magnetism in general.
Given the rise of classical computers in the 1960s, simulating quantum systems for better insight became an important part of physics. But classical computers have not been designed to handle the exponentially growing data structures which are necessary for simulating quantum mechanics [8]. Therefore, the idea of a new type of computer was born that could handle this structure by being inherently designed for them [9]. This computer would use quantum effects for computation and is therefore called *quantum computer*.
However the realization of a device that uses quantum effects for computation has taken some time. The first such machine has been developed by the Canadian company D-Wave and can be regarded as a type of *quantum annealer* [10]. It has the ability to solve *quadratic binary unconstrained optimization (QUBO)* problems using the adiabatic evolution from a known initial groundstate to a final groundstate, which is problem dependent. Since the problem class is restricted to optimization problems, we will not investigate this machine any further. A more general approach to quantum computation is based on *gate model* computations. This idea is based on operations on the quantum encoded data that are comparable to classical gates. Unfortunately only small machines of this type exist today and can be seen as technology demonstrators or experimental devices (see Section 2.2.1).

2.1 GATE MODEL

To start our discussion of quantum computation we want to introduce the fundamental building blocks. The basis for this discussion will be the textbook by Nielsen and Chuang [7], which can be used as a reference for further questions. Similar to Nielsen and Chuang we want to understand quantum computation as the processing of data using quantum mechanical systems. Mainly we want to discuss the building blocks of *gate model* quantum computation, namely: the *qubit*, the manipulation of qubits by *gates* and the *measurement* of qubits.

2.1.1 BUILDING BLOCKS

QUBITS The fundamental building block for quantum computation is the *qubit*. Its name is inherited from its classical ancestor the *bit*. Whereas the bit can only assume a distinct value or state from the set $\{0, 1\}$, the qubit can also assume *intermediate* values of its basis states. Formally we define the set of basis states as $\{|0\rangle, |1\rangle\}$. We use the so called Dirac notation $|\cdot\rangle$ to show the difference between a classical and quantum state explicitly. The obscure term intermediate refers to the property that the qubit $|q\rangle$ can be in a state which is a linear combination of the basis states. This linear combination is also called *superposition*. Mathematically, this relation is given as

$$|q\rangle = \alpha |0\rangle + \beta |1\rangle \text{ with } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1. \tag{2.1}$$

The last condition ensures normalization of the state $|q\rangle$ which is crucial for the interpretation as a physical state and will become clearer when talking about *measurements*. The space of all possible states $|q\rangle$ is called *Hilbert space* (see Reference [11, p. 128] for the definition).
Adding an additional qubit to the system results in a new Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ with $\mathcal{H}_j$ the single qubit spaces. The basis for the new Hilbert space is given as: $|0\rangle \otimes |0\rangle = |00\rangle, |01\rangle, |10\rangle, |11\rangle$. New states are emerging which are non separable and therefore called entangled states. An example is a state like $|\varphi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$. This state is non separable since there are no single qubit states $|q\rangle$ and $|q'\rangle$ with $|\varphi\rangle \neq |q\rangle \otimes |q'\rangle$. This shows the possibility to encode more information in the two qubit system in comparison to two single qubit systems.

GATES Similar to classical computation, the manipulation of qubits is called gate. A gate can act on a single qubit or a group of them. As we already know qubits can be described as an element in a Hilbert space. Therefore, their manipulation is carried out by *operators* mapping a Hilbert space to itself. Since a manipulated state must fulfill the same properties as the non manipulated state, the operators must be linear and unitary. They are often represented as matrices. We want to divide the set of gates into three subsets, gates acting on a single qubit, gates acting on two qubits and the generalization, gates acting on n qubits.
As an example for a single qubit gate we want to take a look at the Hadamard gate denoted as H. The application of a Hadamard gate transfers the qubit into a state of an equal weight superposition. Mathematically this can be described as follows

$$\mathrm{H}|q\rangle = \mathrm{H}(\alpha |0\rangle + \beta |1\rangle) = \frac{\alpha + \beta}{\sqrt{2}} |0\rangle + i\frac{\alpha - \beta}{\sqrt{2}} |1\rangle . \tag{2.2}$$

For the two qubit gates, the *Controlled Not* gate, denoted as CNOT, is a good example. Applying this gate to two qubits will flip the second depending on the state

of the first qubit. In contrast to the Hadamard gate, we will define it through its matrix representation in the usual computational basis

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.3}$$

Generally a quantum gate can act on n qubits and is then represented as an $N \times N = 2^n \times 2^n$ unitary matrix. An interesting result is that comparable to classical computation, it suffices to only consider for a small set of quantum gates, since any other gate can be approximated by them [7, p. 188ff]. A set showing this property is called universal gate set. For classical computation the terminology is *functional completeness* and an example for such a corresponding set is the NAND gate [12, p. 218ff]. For quantum computation an universal gate set, allowing for the approximation of any quantum gate, is given as H, T, P and CNOT. When talking about possible metrics for the computational complexity of quantum algorithms in Subsection 5.3.2, we will discuss further details of universal gates.
Whereas the Hadamard gate is one of the fundamental single qubit gates, the CNOT gate is the simplest one for two qubits.

MEASUREMENTS As distinct from classical computation, we cannot readout the result directly, due to the superposition of the qubits. But we can carry out a quantum mechanical measurement.
For a single qubit this measurement is like posing the question: "Which state do you assume?". But this question is incomplete because it lacks the information about the basis of measurement. Most often the computational basis is used, also in the paragraphs above. The application of a measurement collapses the quantum state into a definite state, the measured one. Therefore the application of successive measurements in different basis can yield different information. Thus further insight into the quantum system can be obtained by using multiple measurements.
If the qubit is in a superposition of the two basis states, as described in Equation 2.1, measuring it in the computational basis, yields one of the two basis states. The probability for the state $|0\rangle$ is given as $|\alpha|^2$ and for the state $|1\rangle$ as $|\beta|^2$. This shows that the absolute square of the amplitudes must be interpreted as a probability. A fundamental property of the measurement is the collapse of the quantum state.
For a group of qubits the measurement process becomes more complex. The number of basis states grows exponentially. As the number of possible outcomes does. As mentioned above, the probability for a specific outcome is connected to the amplitude of the outcome or state. Therefore, shifting the probability to a specific wanted outcome is desired. This has become possible to a certain extend using the *quantum amplitude amplification*. Still, obtaining the required and desired information is a challenging task.

2.1.2 QUANTUM VS. CLASSICAL COMPUTATION

Quantum computation is distinct from classical computation. Apart from the obvious change from bits to qubits, there are other notable differences. These shall be discussed in the following subsection.
The first difference is rather surprising, but implied by the no cloning theorem [7, p. 532]. It states that there is no quantum gate COPY to perform the copy of an arbitrary quantum state, i.e. the following operation

$$\text{COPY}\left(|\varphi\rangle \otimes |0\rangle\right) = |\varphi\rangle \otimes |\varphi\rangle \tag{2.4}$$

with $|\varphi\rangle$ being the arbitrary state. For classical computation this operation can simply be done with a XOR gate*. Since such a gate is missing, intermediate results can not be obtained by copying them out. Additionally, comparing states before and after manipulation requires special ideas.
The second difference regards the manipulation of qubits. Since the manipulations are carried out by linear unitary operators, they are reversible. This means any computation can also be run in backwards direction. In mathematical notation this results in the application of the adjoint operator. This is also a huge difference to classical computations where gates do not need to be reversible. In consequence, it is not straightforward to emulate classical computation on a quantum computer. The basic idea of translating classical to quantum calculations will be discussed later (see Subsection 5.3.1). In addition to that, another consequence of the reversibility is the requirement for *uncomputation*. A measurement collapses the quantum state and leaves the system in the measured state. To obtain the old input state, most often the zero basis state, the computation needs to be undone. Therefore swapping out the result to an ancilla qubit and undoing the computation is required. To undo a computation the inverse gate needs to be applied. Since quantum gates are unitary, the inverse equals the adjoint matrix. Therefore it suffices to apply the adjoint of quantum gates in reversed order. Many fundamental gates are self-adjoint making this application fairly easy.

2.1.3 CONVENTIONS

There are some general conventions in quantum computation which will be introduced in this short subsection.

MANY BODY QUANTUM MECHANICS The mathematical/physical framework for quantum computation is called *many body quantum mechanics*. Typically the tensor products are omitted, e.g. $|q\rangle \otimes |q'\rangle = |q\rangle\,|q'\rangle$. This also applies for the operators, therefore we will often use an additional index to denote the qubit or

*The XOR gate needs to be applied to two classical bits. It copies the first bit into the other one, if this has been initialized to zero. The action of the XOR gate is described by the symbol $\oplus$, thus the action is given as $(a, 0) \to (a, 0 \oplus a) = (a, a)$.

quantum register that is modified by the operator. In special cases tensor products might appear to specify them.

QUANTUM REGISTER To simplify notation we want to collect qubits into quantum registers. A collection of n qubits is called quantum register of length n. The possible states of the register can be relabeled with a binary encoding called number basis. The result is a compact notation. For example the possible states of a 2 qubit register are

$$|q\rangle = \sum_{j=0}^{3} \alpha_j |j\rangle \tag{2.5}$$

with the typical complex amplitudes α_j. Similarly to qubits, gates can be applied on registers.

ANCILLA QUBITS In many cases there is a need of some scratch space for quantum routines or algorithms. The qubits used as scratch space are called *ancilla* qubits or registers as well as *auxillary* qubits or registers. Since quantum algorithms are reversible they are often used to swap out results.

NUMBER REPRESENTATION The representation of numbers is a crucial part to use quantum computation for calculations. The easiest way is to use the amplitude of a quantum state. This method can be called analog.

$$\alpha \in \mathbb{C} \rightarrow |\alpha\rangle = \alpha |0\rangle + \sqrt{1 - |\alpha|^2} |1\rangle \tag{2.6}$$

Obviously this way is well suited to store a vector $\boldsymbol{\alpha}$ within the number basis in a quantum register of length $n = \log_2(N)$. If N is no power of two, we need to round the logarithm to the next higher integer number.

$$\boldsymbol{\alpha} \in \mathbb{C}^N \rightarrow |\boldsymbol{\alpha}\rangle = \frac{1}{C_{\boldsymbol{\alpha}}} \sum_{j=0}^{N-1} \alpha_j |j\rangle \tag{2.7}$$

For an arbitrary complex vector we need an additional normalization constant $C_{\boldsymbol{\alpha}}$. Another method is the digital representation. One such encoding method was already mentioned above. More specifically, the number basis, as resulting from the binary encoding, is such a digital representation for integer numbers.
The representation of decimal numbers is a little bit more difficult. For floating point numbers we can use the same representation as in classical computation with IEEE-754 [13] but it is very resource demanding. A simple approach is to use a fixed point representation like given in the following. For an angle $\phi \in [0, 1)$ with the binary representation $\phi = 0.\phi_1\phi_2\phi_3\ldots$ the representation on a quantum register of length n is given as $\phi \approx 0.\phi_1\phi_2\phi_3\ldots\phi_n$

$$\phi \in [0, 1) \rightarrow |\phi\rangle = |\phi_1\phi_2\phi_3\ldots\phi_n\rangle \tag{2.8}$$

A drawback of both digital representations is that both of them are approximate and therefore not exact for all numbers.

2.2 CHALLENGES AND EXPECTATIONS

After getting to know the building blocks of quantum computation, we want to summarize expectations and challenges of quantum computation.
This technology might offer a new approach to tackle computational intensive tasks. Since there are some algorithms with proven exponential speedup over the best classical algorithms, e.g. *Shor's* algorithm [14], there are high expectations. Most of them are based on the proven speedup for some quantum algorithms with respect to their classical counterparts. Additionally some people expect quantum computation to offer fast solutions for some problems with so called NP complexity.
Albeit there are high expectations towards quantum computation, tough challenges need to be mastered to develop quantum computation as a new framework for real-world applications. There are a lot of challenging problems that need to be solved in order to make quantum computation generally available, i.e. for scientists and businesses. The challenges spread over different areas which will be discussed shortly.

2.2.1 HARDWARE

Realizing a quantum computer as a physical device requires several ingredients. The first one is a physical realization for the qubits.

ARCHITECTURES Several approaches have been examined. Qubits based on the phenomenon of nuclear magnetic resonance (NMR) have been on of the first approaches [15]. Other approaches are superconducting qubits, ion-traps or semiconductor qubits that show better scalability. But there is no approach that has finally superseded all others [7, p. 277ff]. Recently, new approaches like Majorana qubits [16] have been introduced and might show even better results.
The other important ingredient is the physical realization of gates for a specific qubit architecture. On the theoretical level the quantum gates act instantly on the qubits. But on the physical level there is a time evolution from one quantum state to the other. This evolution should be done in minimal time and with minimal error, regarding the difference of target state and reached state. Typically the gates are realized by electromagnetic radiation.
Additionally, the qubit system needs to be in a controlled environment to minimize interactions with the surrounding and to circumvent the so called *cross talk*, i.e. interaction between originally non-interacting qubits.

NISQ DEVICES The first generation of currently available quantum computers is called *NISQ* devices. This abbreviation stands for *noisy intermediate state quantum* and refers to the fact that the devices are only an intermediate step towards a universal quantum computer. A drawback is their small size compared to resource requirements for quantum algorithms like the *Shor* algorithm [14]. The typical

size of today's devices is roughly in the order of 10 qubits [17] which is a small amount compared to resource estimates of about $9n$ qubits factoring a n bit number [6]. Additionally, they can only offer a small *coherence time*, i.e. the timespan with guaranteed entanglement. Therefore, the number of quantum gates running successively is limited. An interesting open question is the possible usage of NISQ devices for applications relevant to scientists and businesses.

2.2.2 ERROR CORRECTION

A challenge that is directly connected to the hardware, i.e. the physical devices, is the correction of errors. Since physical devices are often imperfect, errors are a natural consequence. For classical devices either zero, one or more bits can be flipped. These errors are easily detectable and since many data formats can handle single bit flips, they can be tolerated.
Detecting errors in quantum computation is not done easily. On the qubit level there is an additional possible error, regarding the sign in the superposition. But on the higher levels, it is also possible to have faulty gates, state preparation or measurements. To overcome this errors, *quantum error correction (QEC)* has been developed. It offers techniques to use a number of physical qubits to encode a logical qubit. A possible error in the logical qubit can then be detected by measurements on the physical qubits and corrected. A more detailed description can be seen in [18] and [7, p. 425ff].

HYBRID ALGORITHMS An alternative way of treating the errors is used by *hybrid algorithms* like the *variational quantum eigensolver (VQE)* [19] and the *quantum approximate optimization (QAO)* algorithm [20]. Instead of running completely on a quantum computer, these algorithms are designed to use both classical and quantum resources. Since the *circuit depth* of the quantum algorithm is relatively short, these algorithms might match the small *decoherence* times of NISQ devices (see Subsection 2.2.1).

2.2.3 APPLICATIONS

Since there is no proof for a general quantum advantage, applications solved by efficient quantum algorithms need to be identified. Especially applications that solve real world problems are interesting for scientists and businesses. Most promising is the cut set of the two computational complexity classes BQP and NP. To understand the theoretical limits of quantum computation and understand the importance of the mentioned cut set, we want to give a short introduction to complexity classes.

COMPLEXITY CLASSES In classical information theory several complexity classes are distinguished. The idea of complexity classes has been introduced to compare

problems in terms of their computational hardness. This involves computational time as well as computational space to store results.
The first complexity class we want to mention is the class PSPACE. This complexity class has the property to include all following complexity classes. It contains all problems that require polynomial space for the computation on any device. However there is no specific statement about the computation time of problems within this class. The next complexity class is called BPP. It consists of all problems that can be solved up to an error $\epsilon < 1/2$ on a probabilistic classical computer in polynomial time. Basically, probabilistic computation can be realized by statistical sampling. One can show that BPP $\subseteq$ PSPACE by using a small argument from complexity theory. A special case of a probabilistic computer is the non probabilistic one, which can solve fewer problems in polynomial runtime. Therefore the corresponding complexity class P is included in BPP. On the other hand there are problems that can be solved in polynomial runtime on non deterministic computers. They are collected in the complexity class NP. A fundamental question in information theory regards the relation of the complexity classes, e.g. is $\mathrm{P} \overset{?}{=} \mathrm{NP}$ or $\mathrm{NP} \overset{?}{\subseteq} \mathrm{BPP}$ [21].
To give a statement about quantum resources the complexity class BQP has been defined. Loosely speaking, it consists of all problems that can be calculated efficiently on a quantum computer. It is known that the complexity class BQP is a subclass of PSPACE but its relation to the other classical complexity classes remains mainly unknown. The article [22] by Bernstein and Vazirani contains further material concerning this question. Further research on the relation of the complexity classes is needed to show that quantum computation has a definite advantage over classical computation, e.g. NP $\subseteq$ BQP. Before such an elementary proof can be provided, examples need to be identified that are part of the cut set BQP $\cap$ NP and are not contained in P, like *Shor's* algorithm [14].

APPLICATION CANDIDATES Identifying applications for quantum algorithms is not an easy task since only a few quantum algorithms are known. Additionally the setting is upside down compared to the typical solution path. Usually we start with a known application and try to find an algorithm solving it. We can search for an algorithm that fulfills the desired needs or construct an algorithm that matches the problem/application. Finding applications to fit quantum algorithms is the other way around. There is only a small number of possible algorithms but a large number of applications. In the rest of this paragraph a candidate for a promising application will be discussed.
As already mentioned in the introduction, quantum computation is inherently well suited to solve problems related to quantum mechanics. Therefore, a promising application is the simulation of molecules that show strong quantum effects. These effects can not be handled with classical computational methods to a satisfactory extend because strongly bound electrons are involved. The quantum algorithm to tackle these type of problems is the *variational quantum eigensolver (VQE)* [19], as already mentioned in Subsection 2.2.2. Based on a classical approximation

(e.g. Hartree-Fock methods) it develops an approximation for the groundstate based on a chosen ansatz and can therefore be seen as a post Hartree-Fock method.

2.3 SUMMARY

At this point we want to review the content of the last chapter. This chapter has been divided into two major sections. The first one was used to give a short introduction to quantum computation. We mainly focused on the gate model but also mentioned alternative approaches. Most importantly, the building blocks of gate model quantum computation have been introduced. Additionally, the differences to classical computation have been mentioned, like the implications of the no cloning theorem. Furthermore, we listed the important conventions used in quantum computation to allow for an understanding of the next chapters.
The other section deals with the challenges and expectations towards quantum computation in general. On the one hand the early hardware devices suffer strongly from low qubit counts. On the other hand sophisticated quantum algorithms require perfect qubits and gate fidelities, resulting in the need for quantum error correction. Additionally, well suited applications need to be identified such that the full potential of quantum computation can be used.
Based on the knowledge obtained in this chapter, we will introduce a quantum algorithm, the quantum linear system (QLS) algorithm and discuss its properties in the next chapter.

3

QUANTUM LINEAR SYSTEM ALGORITHM

In this chapter we want to discuss the main quantum algorithm used in this thesis. The first idea of such an algorithm has been introduced in the year 2008 by Harrow, Hassidim, and Lloyd (HHL) which led to the name HHL algorithm [1]. More recently this algorithm has been improved and altered to suit specific design requirements. Algorithms, which are based on the Harrow, Hassidim, and Lloyd algorithm, are called quantum linear system (QLS) algorithms.
The main purpose of the quantum linear system algorithm is to provide the quantum equivalent of a linear system solver. Starting with an initial state representing the right hand side of a system of linear equations and some *ancilla registers*, it transforms the state to a final state proportional to the solution of the linear system.
In the following we will investigate the details of this transformation, look at the fundamental difference to classical linear system algorithms, and understand the different subroutines involved. Finally, we will be faced with three problems which need to be solved for a successful implementation.

MATHEMATICAL PROBLEM To start the investigation we need to define the mathematical problem that needs to be solved. Let $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ be a matrix and $\boldsymbol{b} \in \mathbb{C}^N$ be a vector, we search the vector $\boldsymbol{x} \in \mathbb{C}^N$ with

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$

Such an equation is called a linear system. In the definition above we have already restricted ourselves to square matrices. We want to restrict ourselves even further, such that we only consider linear systems that have a unique solution. A unique solution exists, if the rank of the matrix is the same as the rank of the expanded matrix, which is obtained by adding the right hand side vector as a new column vector to the matrix [23, p. 40]

$$\operatorname{rank}(\boldsymbol{A}) = \operatorname{rank}(\boldsymbol{A}, \boldsymbol{b}).$$

Additionally, we want to assume that the matrix is Hermitian. If that is not the case we can modify the system such that it becomes Hermitian, not changing the computational complexity, as shown in Appendix A.1.

3.1 MAIN ALGORITHM

Defining the mathematical problem properly, allows us to start our investigation of the quantum linear system algorithm. Now, we will investigate the involved subroutines, identify problems and solutions. In the following subsection we take a look at the classical HHL algorithm introduced by Harrow, Hassidim, and Lloyd. The used subroutines will be explained in detail afterwards.

STATE PREPARATION As a first step, we need to prepare two quantum registers, one of length $t = \log_2(T)$ and one of length $n = \log_2(N)$. The register of length t is used to store eigenvalues. It is prepared in an equal weight superposition, resulting in $\frac{1}{\sqrt{T}} \sum_{\tau=0}^{T-1} |\tau\rangle$. The other register will hold a state corresponding to the right hand side $\boldsymbol{b}$ of the linear system. Thus, we describe it by $\frac{1}{C_b} \sum_{j=0}^{N-1} b_j |j\rangle$ with $b_j = (\boldsymbol{b})_j$ and C_b, a proper normalization constant. The full system can be described by the following quantum state

$$\frac{1}{\sqrt{T}} \frac{1}{C_b} \sum_{\tau=0}^{T-1} \sum_{j=0}^{N-1} b_j |\tau\rangle \, |j\rangle \,. \tag{3.1}$$

HAMILTONIAN SIMULATION To let the Hermitian matrix $\boldsymbol{A}$ act on a quantum state, we need to define the time evolution operator $\mathrm{U}_\tau(A)$ by interpreting the matrix $\boldsymbol{A}$ as a Hamiltonian. This interpretation motivates the restriction to Hermitian matrices. The time evolution operator is defined as

$$\mathrm{U}_\tau(A) = \exp(-i\tau A) \,. \tag{3.2}$$

Additionally, we want to introduce a basis change. The new basis is the eigenbasis of the matrix $\boldsymbol{A}$. Since the basis is not known, this basis change is just a mathematical trick. The new basis is denoted as $\{\boldsymbol{u}_j\}$. The corresponding quantum states will be denoted as $|u_j\rangle$. This eigenbasis has the property that the action of the time evolution can be calculated easily

$$\mathrm{U}_\tau(A) |u_j\rangle = \exp(-i\tau\lambda_j) \, |u_j\rangle \tag{3.3}$$

with λ_j the eigenvalue to eigenvector u_j. Rewriting the full system in this new basis yields

$$\frac{1}{\sqrt{T}} \sum_{j=0}^{N-1} \sum_{\tau=0}^{T-1} \beta_j |\tau\rangle \, |u_j\rangle \,. \tag{3.4}$$

Now we apply the controlled Hamiltonian evolution $\sum_{\tau'=0}^{T-1} |\tau'\rangle \langle\tau'| \otimes \exp\left(-iA\tau' \frac{t_0}{T}\right)$ onto the second register, controlled by the first. The constant t_0 is needed to ensure

an evolution time that minimizes the error ϵ of the representation for the eigenvalues λ_j. Therefore, we choose it such that $t_0 \sim \frac{1}{\epsilon}\left|\frac{\lambda_{max}}{\lambda_{min}}\right|$ [1]. The resulting formula is

$$\begin{aligned}&\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1}\sum_{\tau'=0}^{T-1}\sum_{j=0}^{N-1}|\tau'\rangle\ \langle\tau'|\tau\rangle \otimes \exp\left(-iA\tau'\frac{t_0}{T}\right)\beta_j\,|u_j\rangle \\ =&\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1}\sum_{j=0}^{N-1}\beta_j \exp\left(-i\lambda_j\tau\frac{t_0}{T}\right)\ |\tau\rangle\ |u_j\rangle\,.\end{aligned} \tag{3.5}$$

Even though the definition and action of the time evolution operator is straightforward, its actual implementation for a general Hermitian matrix is not. Therefore, its implementation, which is called Hamiltonian simulation, will be subject of Subsection 3.2.3.

QUANTUM FOURIER TRANSFORM The next step is to transfer the eigenvalues from the amplitude to the first register. This transfer is done by the *quantum Fourier transformation (QFT)* (see Subsection 3.2.2). It can be described by the mapping $|\tau\rangle \to \frac{1}{\sqrt{T}}\sum_{k=0}^{T-1}\exp\left(\frac{2\pi i}{T}k\tau\right)\ |k\rangle$. In total, this results in

$$\begin{aligned}&\frac{1}{T}\sum_{\tau=0}^{T-1}\sum_{k=0}^{T-1}\sum_{j=0}^{N-1}\beta_j \exp\left(-i\lambda_j\tau\frac{t_0}{T}\right)\exp\left(\frac{2\pi i}{T}k\tau\right)\ |k\rangle\ |u_j\rangle \\ =&\frac{1}{T}\sum_{\tau=0}^{T-1}\sum_{k=0}^{T-1}\sum_{j=0}^{N-1}\beta_j \exp\left(\frac{2\pi i}{T}\tau\left(k-\lambda_j\frac{t_0}{2\pi}\right)\right)\ |k\rangle\ |u_j\rangle \\ =&\sum_{k=0}^{T-1}\sum_{j=0}^{N-1}\beta_j\alpha_{kj}\,|\tilde{\lambda}_k\rangle\ |u_j\rangle\,.\end{aligned} \tag{3.6}$$

In the last step, we defined $\alpha_{kj} = \frac{1}{T}\sum_{\tau=0}^{T-1}\exp\left(\frac{2\pi i}{T}\tau\left(k-\lambda_j\frac{t_0}{2\pi}\right)\right)$ and relabeled the first register as $\tilde{\lambda}_k = \frac{2\pi k}{t_0}$. Under the assumption of infinite precision, we can evaluate α_{kj} to δ_{kj}, which motivates us to choose T and t_0 such that the difference between α_{kj} and δ_{kj} is small enough. Thus $\tilde{\lambda}_k$ is just an approximation to the k-th eigenvalue. A detailed error discussion can be found in the book by Nielsen and Chuang on page 224 [7].

VALUE CONTROLLED ROTATION Finally, we want to multiply the amplitudes by the inverse of the eigenvalues. We add an ancilla register of length one and rotate its state controlled by the value in the first register (see Subsection 3.2.1).

$$\sum_{k=0}^{T-1}\sum_{j=0}^{N-1}\beta_j\alpha_{kj}\,|\tilde{\lambda}_k\rangle\ |u_j\rangle\left(\sqrt{1-\left(\frac{C_A}{\tilde{\lambda}_k}\right)^2}\,|0\rangle + \frac{C_A}{\tilde{\lambda}_k}\,|1\rangle\right) \tag{3.7}$$

Here, we introduced another constant C_A to ensure that all rotations are valid. The next step is to "free" the first register by *uncomputation*. The purpose of this is

to ensure that we can ignore them in the following. For infinite precision we have $\alpha_{kj} = \delta_{kj}$ which leads to

$$\sum_{j=0}^{N-1} \beta_j \left|u_j\right\rangle \left(\sqrt{1 - \left(\frac{C_A}{\lambda_j}\right)^2} \left|0\right\rangle + \frac{C_A}{\lambda_j} \left|1\right\rangle \right). \tag{3.8}$$

MEASUREMENT Measuring the value of the ancilla register yields one of the two possible states. If we obtain a $\left|1\right\rangle$, the rest of the system is in a state that is a solution to the linear system up to a normalization constant

$$\frac{1}{C_x} \sum_{j=0}^{N-1} \frac{\beta_j}{\lambda_j} \left|u_j\right\rangle. \tag{3.9}$$

In the above equation the normalization constant is called C_x.

3.2 SUB ALGORITHMS

Getting to know the involved subroutines is an essential part in understanding the full quantum linear system algorithm. Therefore, we will discuss these subroutines in more detail.

3.2.1 VALUE CONTROLLED ROTATION

The *value controlled rotation* performs a rotation on an ancilla qubit by an value $x \in [0, 1]$, encoded in the first quantum register $\left|x\right\rangle$ of length $n = log_2(N)$. Its action is given as

$$\begin{aligned} \left|x\right\rangle \left|0\right\rangle &\rightarrow \left(\sum_{k=0}^{N-1} \left|k\right\rangle \left\langle k\right| \otimes R(k) \right) \left|x\right\rangle \left|0\right\rangle \\ &\rightarrow \left|x\right\rangle R(x) \left|0\right\rangle \\ &\rightarrow \left|x\right\rangle \left(\sqrt{1 - x^2} \left|0\right\rangle + x \left|1\right\rangle \right). \end{aligned} \tag{3.10}$$

Since the sine function can be inverted on a proper domain and there are efficient classical approximation formula for this inverse, a quantum circuit exists to implement the computation $\left|x\right\rangle \rightarrow \left|\phi\right\rangle = \left|\frac{1}{2\pi} \arcsin(x)\right\rangle$. The execution of classical calculations on quantum resources is discussed in Subsection 5.3.1. Using this angle ϕ, we are able to diagonalize the rotation operator $R(x)$ with a unitary matrices that are independent of the angle

$$\begin{aligned} R(x) = R(\phi) &= \begin{pmatrix} \cos(2\pi\phi) & -\sin(2\pi\phi) \\ \sin(2\pi\phi) & \cos(2\pi\phi) \end{pmatrix} \\ &= U \begin{pmatrix} \exp(2\pi i\phi) & 0 \\ 0 & \exp(-2\pi i\phi) \end{pmatrix} U^\dagger = U \tilde{R}(\phi) U^\dagger. \end{aligned} \tag{3.11}$$

The matrix $\tilde{R}(\phi)$ can now be split into submatrices S_k, since the angle in binary encoding can be rewritten as

$$\begin{aligned}\phi &= 0.\phi_1\phi_2\phi_3\ldots\phi_n \\ &= 2^{-1}\phi_1 + 2^{-2}\phi_2 + \cdots + 2^{-n}\phi_n \\ &= \sum_{k=1}^{n} 2^{-k}\phi_k.\end{aligned} \tag{3.12}$$

We get $\tilde{R}(\phi) = \prod_{k=1}^{n}(S_k)^{\phi_k}$ with $S_k = \begin{pmatrix} \exp\left(2\pi i 2^{-k}\right) & 0 \\ 0 & \exp\left(-2\pi i 2^{-k}\right) \end{pmatrix}$, since ϕ_k can either be zero or one. The final circuit is given in Figure 3.1.

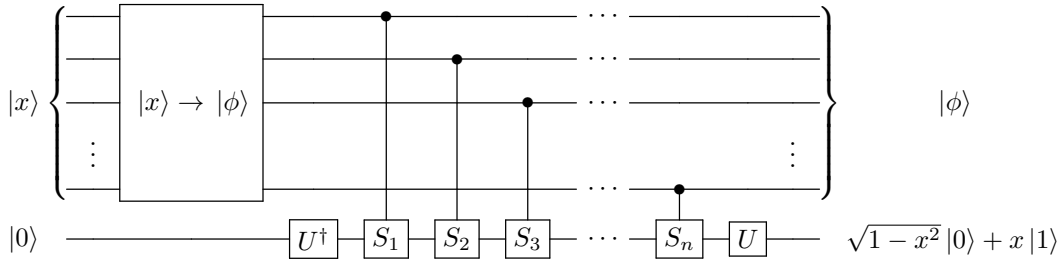

Figure 3.1: Value controlled rotation: The figure gives the quantum circuit to realize the value controlled rotation gate on n qubits. Before the rotation can be applied the angle needs to be computed using trigonometric functions. The quantum register holding the angle $\phi \in [0, 1)$ controls the rotation of the ancilla qubit.

3.2.2 QUANTUM FOURIER TRANSFORM

Similar to the classical discrete *Fourier transformation*, the *quantum Fourier transformation* maps the amplitudes of a quantum state to its Fourier transformation [7, p. 217ff]. The action of the Fourier transformation, on a quantum register of $n = \log_2(N)$ qubits, is given as

$$\sum_{j=0}^{N-1} x_j\,|j\rangle \to \sum_{k=0}^{N-1} y_k\,|k\rangle \;\text{ with }\; y_k = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} x_j \exp\left(\frac{2\pi i}{N} jk\right). \tag{3.13}$$

Even though the definition of the quantum Fourier transformation is a closed equation, it can be shown that the n-th Fourier coefficient is not influenced by the $n-1$ other coefficients and therefore we can build up a recursive scheme for the quantum Fourier transformation (see Figure 3.2).

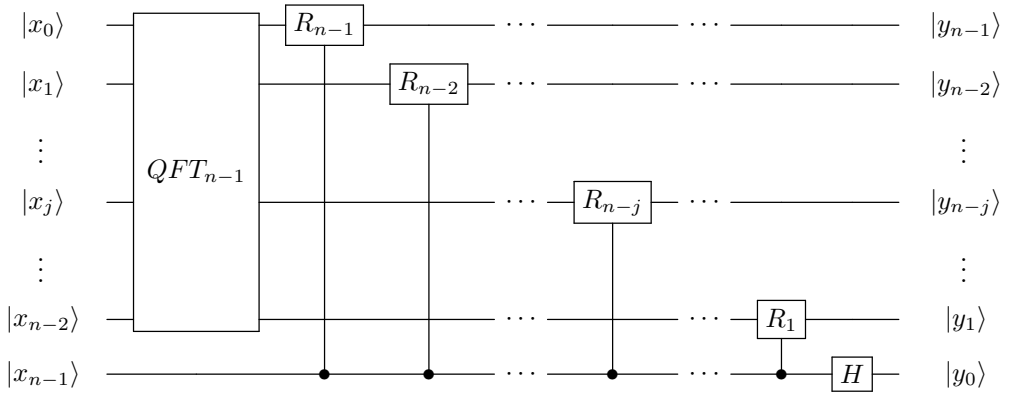

Figure 3.2: Quantum Fourier transformation: The figure gives the quantum circuit to realize the quantum Fourier transformation on n qubits. This scheme describes the recursive implementation using rotation matrices defined as $R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi 2^{-k}} \end{pmatrix}$.

3.2.3 HAMILTONIAN SIMULATION

The algorithm *Hamiltonian simulation* describes the application of the time evolution operator of an arbitrary Hermitian matrix. Let $\boldsymbol{A}$ be a complex, Hermitian $N \times N$ matrix, $|\psi\rangle$ a quantum register of length $n = log_2(N)$ and $\tau \in \mathbb{R}^+$ a time. The action of the Hamiltonian simulation is described as follows

$$|\psi\rangle \to \mathrm{U}_\tau(A)\ |\psi\rangle = \exp(-i\tau A)\ |\psi\rangle\,. \tag{3.14}$$

Although, applying such an operator to a quantum register is straightforward, building an efficient quantum circuit to do so, requires further work. The idea is to divide the operator on different levels, such that easier applicable operators with efficient gate realizations arise [4, 24]. The first step is to divide the relatively large time into r smaller times, such that $\tau/r \ll 1$. The emerged operator can be applied r-times

$$\mathrm{U}_\tau(A) = \left[\mathrm{U}_{\tau/r}(A)\right]^r. \tag{3.15}$$

For simplicity, we relabel the times $\tau/r \to \tau$. The next decomposition step will take effect on the matrix level. We want to decompose the matrix into one-sparse Hermitian matrices $\boldsymbol{A} = \sum_{c=1}^{m} \boldsymbol{A}_c$. One-sparse means that there is not more than one non zero element per row or column. In general such a decomposition is not efficiently computable. If the matrix $\boldsymbol{A}$ is sparse, we can use the methods of graph colouring to define it [24]. For the special case of a banded matrix, the decomposition is trivial. Using the *Suzuki higher order integrators* which are based on the *Suzuki Trotter* formula, we can approximate $\exp(-i\tau A)$ by applying the unitaries $\exp(-i\tau A_c)$ according to the following scheme.

SUZUKI HIGHER ORDER INTEGRATORS In general $[A, B] \neq 0$ causes

$$\exp(-i\tau(A+B)) \neq \exp(-i\tau A)\exp(-i\tau B)\,.$$

Therefore, we want to use a recursive operator $\mathrm{S}_{2k}(-i\tau)$ to approximate $\exp(-i\tau A)$ up to approximation error $\mathcal{O}\left(|\tau|^{2k+1}\right)$. The operator is defined as

$$\begin{aligned}
\mathrm{S}_{2k}(-i\tau) &= \left[\mathrm{S}_{2k-2}(-ip_k\tau)\right]^2 \mathrm{S}_{2k-2}(-i(1-4p_k)\tau)\left[\mathrm{S}_{2k-2}(-ip_k\tau)\right]^2, \\
\mathrm{S}_2(-i\tau) &= \Pi_{c=1}^{m} \exp(-iA_c\tau/2)\, \Pi_{c'=m}^{1} \exp(-iA_{c'}\tau/2), \\
p_k &= \left(4 - 4^{1/(2k-1)}\right)^{-1}.
\end{aligned} \tag{3.16}$$

Often a small value for k is sufficient for the desired accuracy. As a simple example, we want to choose $m = 2$ and $k = 1$, which results in

$$\begin{aligned}
\mathrm{U}_\tau(A) &= \exp(-i\tau(A_1 + A_2)) \\
&\approx S_2(-i\tau) \\
&= \exp(-i\tau A_1/2)\exp(-i\tau A_2)\exp(-i\tau A_1/2).
\end{aligned} \tag{3.17}$$

A detailed convergence and error analysis can be found in References [25] and [24]. In consequence an implementation for operators of the type $\exp(-i\tau A_c)$ for $\boldsymbol{A}_c$, a one-sparse matrix need to be found.

HAMILTONIAN CIRCUIT We want to define a set of unitary, self adjoint operators for each matrix $\boldsymbol{A}_c$. One operator for the argument U_c^p and one for the amplitude U_c^m of the matrix elements shall be defined. They are given as follows

$$\mathrm{U}_c^m |a\rangle\ |b\rangle\ |y\rangle\ |z\rangle = |a\rangle\ |b \oplus \mathrm{col}_c(a)\rangle\ |y \oplus \mathrm{amp}_c(a)\rangle\ |z\rangle \tag{3.18}$$

$$\mathrm{U}_c^p |a\rangle\ |b\rangle\ |y\rangle\ |z\rangle = |a\rangle\ |b \oplus \mathrm{col}_c(a)\rangle\ |y\rangle\ |z \oplus \mathrm{arg}_c(a)\rangle. \tag{3.19}$$

These operators act on four registers, the first one holds the row index a with index c defining the matrix. The column index, being hold in the second register, is $\mathrm{col}_c(a)$ and since A_c is one-sparse, it is unique. The amplitude of the matrix element* is given as $\mathrm{amp}_c(a) = \mathrm{amp}(\langle a| A_c |\mathrm{col}_c(a)\rangle)$ and its argument as $\mathrm{arg}_c(a) = \arg(\langle a| A_c |\mathrm{col}_c(a)\rangle)$. They are hold in the third and fourth register respectively. Since A_c is Hermitian by definition, we know that $\mathrm{col}_c(\mathrm{col}_c(a)) = a$, $\mathrm{amp}_c(\mathrm{col}_c(a)) = \mathrm{amp}_c(a)$ and $\mathrm{arg}_c(\mathrm{col}_c(a)) = -\mathrm{arg}_c(a)$. The set of operators constitutes a *matrix oracle* which gets its name from the resemblance to an ancient oracle.
Additionally, we define an operator called *quantum random walk* operator which is explained in detail in Subsection 3.2.5. Its action is defined by swapping two registers and keeping a third in place.

$$\mathrm{H}^{\mathrm{rw}} |a\rangle\ |b\rangle\ |y\rangle = y\,|b\rangle\ |a\rangle\ |y\rangle. \tag{3.20}$$

As a last ingredient, we need the *value controlled phase* gate. For $\phi \in [0, 1)$ its action is given as follows

$$\mathrm{P}_\tau |\phi\rangle\ |0\rangle = \exp(2\pi i\tau\phi)\ |\phi\rangle\ |0\rangle. \tag{3.21}$$

*For $z \in \mathbb{C}$, we get $z = \mathrm{amp}(z)\exp(i\arg(z))$

Its implementation is comparable to the implementation of the *value controlled rotation* gate 3.2.1 and can be seen in Subsection 3.2.4. With these four operators, the matrix $\boldsymbol{A}$ can formally be written as

$$A = \sum_{c=1}^{m} A_c = \sum_{c=1}^{m} \mathrm{U}_c^p P_{1/4\pi}^{\dagger} \mathrm{U}_c^p \mathrm{U}_c^m \mathrm{H}^{\mathrm{rw}} \mathrm{U}_c^m \mathrm{U}_c^p \mathrm{P}_{1/4\pi} \mathrm{U}_c^p. \tag{3.22}$$

The action of the operators A_c is given as

$$\begin{aligned} A_c \left|a\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle &= \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^{\dagger} \mathrm{U}_c^p \mathrm{U}_c^m \mathrm{H}^{\mathrm{rw}} \mathrm{U}_c^m \mathrm{U}_c^p \mathrm{P}_{1/4\pi} \mathrm{U}_c^p \left|a\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle \\ &= \mathrm{amp}_c\left(a\right) \exp\left(i \mathrm{arg}_c\left(a\right)\right) \left|\mathrm{col}_c\left(a\right)\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle. \end{aligned} \tag{3.23}$$

For the sake of brevity, we skip this proof and refer the interested reader to the Appendix A.4. An important point to keep in mind is that, this equation is just descriptive. Since $\boldsymbol{A}_c$ is not unitary, it is not directly applicable to qubits. Therefore the equation is not rigorous in a mathematical or physical sense.
Now we want to implement the unitary time evolution operator for a single submatrix $\exp\left(-i\tau A_c\right)$. Since $\mathrm{U}_c^m \mathrm{U}_c^m = \mathrm{U}_c^p \mathrm{U}_c^p = 1$ and $\mathrm{P}_{1/4\pi}^{\dagger} \mathrm{P}_{1/4\pi} = 1$, we can simplify the operator $\exp\left(-i\tau A_c\right)$ as

$$\exp\left(-i\tau A_c\right) = \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^{\dagger} \mathrm{U}_c^p \mathrm{U}_c^m \exp\left(-i\tau \mathrm{H}^{\mathrm{rw}}\right) \mathrm{U}_c^m \mathrm{U}_c^p \mathrm{P}_{1/4\pi} \mathrm{U}_c^p \tag{3.24}$$

Thus, we only need a circuit for $\exp\left(-i\tau \mathrm{H}^{\mathrm{rw}}\right)$. Whereas this circuit will also be explained in Subsection 3.2.5, the action is simply given by

$$\begin{aligned} &\exp\left(-i\tau \mathrm{H}^{\mathrm{rw}}\right) \left|a\right\rangle \left|\mathrm{col}_c\left(a\right)\right\rangle \left|\mathrm{amp}_c\left(a\right)\right\rangle \left|\mathrm{arg}_c\left(a\right)\right\rangle \\ &\quad = \cos\left(\mathrm{amp}_c\left(a\right)\tau\right) \left|a\right\rangle \left|\mathrm{col}_c\left(a\right)\right\rangle \left|\mathrm{amp}_c\left(a\right)\right\rangle \left|\mathrm{arg}_c\left(a\right)\right\rangle \\ &\quad -i \sin\left(\mathrm{amp}_c\left(a\right)\tau\right) \left|\mathrm{col}_c\left(a\right)\right\rangle \left|a\right\rangle \left|\mathrm{amp}_c\left(a\right)\right\rangle \left|\mathrm{arg}_c\left(a\right)\right\rangle. \end{aligned} \tag{3.25}$$

Combining all parts we get a unitary operator for each of the matrices, given as

$$\begin{aligned} &\exp\left(-i\tau A_c\right) \left|a\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle \\ &\quad = \cos\left(\mathrm{amp}_c\left(a\right)\tau\right) \left|a\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle \\ &\quad -i \sin\left(\mathrm{amp}_c\left(a\right)\tau\right) \exp\left(i \mathrm{arg}_c\left(a\right)\right) \left|\mathrm{col}_c\left(a\right)\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle. \end{aligned} \tag{3.26}$$

3.2.4 VALUE CONTROLLED PHASE SHIFT

The *value controlled phase shift* gate is very similar to the value controlled rotation gate which has been described in Subsection 3.2.1. For a normalized phase shift $\phi \in [0, 1)$, encoded in a quantum register of length n and an ancilla qubit, it is defined as

$$\mathrm{P}_{\tau} : \left|\phi\right\rangle \left|0\right\rangle \rightarrow \exp\left(2\pi i \tau \phi\right) \left|\phi\right\rangle \left|0\right\rangle. \tag{3.27}$$

For simplicity, we set $\tau = 1$. The implementation of this operator can be realized as follows. The operator acting on the ancilla qubit, in dependence on the phase ϕ, is given as

$$\mathrm{P}(\phi) = \begin{pmatrix} \exp(2\pi i\phi) & 0 \\ 0 & \exp(-2\pi i\phi) \end{pmatrix}. \tag{3.28}$$

Using the binary encoding of the normalized phase shift $\phi = 0.\phi_1\phi_2\ldots\phi_n = \sum_{k=1}^{n} 2^{-k}\phi_k$, we can define consecutive shift operators S_k as follows

$$\mathrm{S}_k = \begin{pmatrix} \exp\left(2\pi i 2^{-k}\right) & 0 \\ 0 & \exp\left(-2\pi i 2^{-k}\right) \end{pmatrix}. \tag{3.29}$$

This allows us to rewrite the phase shift operator $\mathrm{P}(\phi)$ as $\mathrm{P}(\phi) = \prod_{k=1}^{n} (\mathrm{S}_k)^{\phi_k}$. The full quantum circuit can be seen in Figure 3.3.
Sometimes an additional qubit in the first register is used to represent the sign of the angle ϕ. If such an encoding is used, we need to add two CNOT gates to the quantum circuit. These gates are depending on the sign qubit and flip the ancilla qubit depending on it. They have to be distributed to the beginning and the end of the circuit. The idea behind the flip of the ancilla qubit is that P acting on the $|1\rangle$ state results in a negative phase shift.

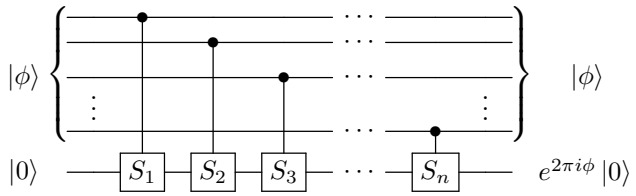

Figure 3.3: Value controlled phase shift: The figure gives the quantum circuit to realize the value controlled phase shift on n qubits. A quantum register holding the phase $\phi \in [0, 1)$ controls the shift of the ancilla qubit.

3.2.5 QUANTUM RANDOM WALK

The *quantum random walk* operator H^{rw} is a central part for the Hamiltonian simulation subroutine. The name is derived from the *quantum walk* operator as described in [26]. The operator acts on three registers, the first two registers need to have the same length n and the other one is of arbitrary length. Its action is defined as swapping the first two registers and being diagonal in the third register

$$\mathrm{H}^{\mathrm{rw}} |a\rangle\ |b\rangle\ |y\rangle = y\,|b\rangle\ |a\rangle\ |y\rangle\,. \tag{3.30}$$

This operator can be rewritten in terms of the single registers as

$$\mathrm{H}^{\mathrm{rw}} = \left(\bigotimes_{j=1}^{n} \mathrm{S}_{j,j} \right) \otimes \sum_{y} y\,|y\rangle\ \langle y| \tag{3.31}$$

with *swap* operator $\mathrm{S}_{j,l}$ that swaps the qubits j of the first and l of the second register. For the Hamiltonian simulation subroutine (see Subsection 3.2.3) the quantum random walk time evolution $\exp\left(-i\tau \mathrm{H}^{\mathrm{rw}}\right)$ is needed. The action is given as

$$\exp\left(-i\tau \mathrm{H}^{\mathrm{rw}}\right) |a\rangle\ |b\rangle\ |y\rangle = \cos\left(y\tau\right) |a\rangle\ |b\rangle\ |y\rangle - i \sin\left(y\tau\right) |b\rangle\ |a\rangle\ |y\rangle\,. \tag{3.32}$$

The quantum circuit to accomplish this action and a short example can be found in the Appendix A.5. Further information can be found in References [4, 26].

3.2.6 STATE PREPARATION

The first subroutine invoked, is the *state preparation*. Whereas preparing a state like $\frac{1}{\sqrt{T}} \sum_{\tau=0}^{T-1} |\tau\rangle$ is a simple task, preparing one proportional to the right hand side is challenging.
The first task can be solved easily, since it only requires applying the Hadamard gate H separately to all qubits in the first register, like in Equation 2.1. In comparison further work is needed to prepare the second register. The task is to prepare a quantum register proportional to a complex vector $\boldsymbol{b}$ with entries $\left(\boldsymbol{b}\right)_j = b_j$ as

$$\frac{1}{C_b} \sum_{j=0}^{N-1} b_j |j\rangle \tag{3.33}$$

with a normalization constant C_b. Using the procedure as proposed in article [27] by Grover et al., which was originally referenced by Harrow et al. in article [1], is generally not possible. It requires the possibility to efficiently compute partial sums of the vector entries, like $\sum_{j=J_1}^{J_2} |b_j|^2$. A more general preparation scheme has been proposed by Clader et al. in Reference [4]. This preparation scheme is used here, since it is based on the same oracle formalism as the matrix oracle.
Instead of preparing the state above, a more elaborate construction, using a *garbage state* and an ancilla qubit, is used. The state to prepare is given as

$$\cos\left(\phi_b\right) |\tilde{b}\rangle\ |0\rangle + \sin\left(\phi_b\right) |b\rangle\ |1\rangle\,. \tag{3.34}$$

The garbage state is denoted as $|\tilde{b}\rangle$ and the angle ϕ_b is needed for normalization. Initially we prepare a state given as

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle\ |0\rangle\ |0\rangle\ |0\rangle \tag{3.35}$$

with register two and three of length t, able to hold the amplitude $\alpha_j = \mathrm{abs}\left(b_j\right)$ and argument $\phi_j = \frac{1}{2\pi}\arg\left(b_j\right)$ of the vector entries. The vector oracle for the right hand side $\boldsymbol{b}$ is controlled by the first register which encodes the row index. Querying it encodes the amplitude and argument in the second and third register, resulting in the following state

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle\ |\alpha_j\rangle\ |\phi_j\rangle\ |0\rangle\,. \tag{3.36}$$

In the next step, the value controlled phase shift gate, as described in Subsection 3.2.4 is used to apply a phase shift to the ancilla qubit in the fourth register, proportional to the argument encoded in register three. The resulting intermediate state is given as

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{i\phi_j} \left|j\right\rangle \left|\alpha_j\right\rangle \left|\phi_j\right\rangle \left|0\right\rangle . \tag{3.37}$$

As a last step, we use the value controlled rotation gate as described in Subsection 3.2.1. This rotates the ancilla qubit proportional to the amplitude encoded in the second quantum register. To ensure normalization, a constant needs to be used

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \left|j\right\rangle \left|\alpha_j\right\rangle \left|\phi_j\right\rangle \left(\sqrt{1 - C_b^2 \alpha_j^2} \left|0\right\rangle + C_b \alpha_j \left|1\right\rangle\right) . \tag{3.38}$$

To bring this equation into the proposed form in Equation 3.34, we need to define the angle ϕ_b as $\sin^2(\phi_b) = \frac{C_b^2}{N} \sum_{j=0}^{N-1} \alpha_j^2$. For the right hand side state and the garbage state, we get the following final result

$$\left|b\right\rangle = \frac{1}{\sqrt{N} \sin(\phi_b)} \sum_{j=0}^{N-1} C_b \alpha_j e^{i\phi_j} \left|j\right\rangle , \tag{3.39}$$

$$\left|\tilde{b}\right\rangle = \frac{1}{\sqrt{N} \cos(\phi_b)} \sum_{j=0}^{N-1} \sqrt{1 - C_b^2 \alpha_j^2} e^{i\phi_j} \left|j\right\rangle . \tag{3.40}$$

Since the Value Controlled Phase Shift gate and the Value Controlled Rotation gate only consume $n = \log_2(N)$ operations, an efficient state preparation is possible. But it requires the usage of a black box oracle to encode the amplitude and argument of the vectors values into the quantum registers. The properties of such an oracle will be discussed in one of the next paragraphs.
By changing the input state, the output state of the algorithm has also been altered. The new output or final state of the quantum linear system algorithm will be given as

$$\cos(\phi_x) \left|\tilde{x}\right\rangle \left|0\right\rangle + \sin(\phi_x) \left|x\right\rangle \left|1\right\rangle \tag{3.41}$$

with ϕ_x a normalization angle and $\left|\tilde{x}\right\rangle$ a garbage state.

3.3 DISCUSSION

In this last section three different topics will be discussed, beginning with the algorithmic complexity. Especially, we will compare the complexity of classical algorithms to the complexity of the QLS algorithm. The next topic deals with different challenges on an algorithmic level. Most of them will be solved by tricks or additional subroutines. This chapter will be sum up at the end.

3.3.1 COMPLEXITY

Before discussing the computational complexity of the quantum linear system algorithm, we need to consider the computational complexity of solving linear systems in general.

CLASSICAL COMPLEXITY The simplest classical algorithm to solve a general system of linear equations, is the *Gaussian elimination* algorithm. In the worst case, it performs $\mathcal{O}(N^3)$ operations on the matrix and right hand side vector [28, p. 11]. Thus, it uses only polynomial runtime. Thereby, we can conclude that solving a system of linear equations is a problem in the complexity class P.
Instead of solving the system of linear equations directly, like the Gaussian elimination algorithm does, it is possible to refine an approximate solution iteratively. The collection of *Krylow subspace methods* is such a class of recent iterative algorithms. The most important operation in sparse iterative solvers is *sparse matrix vector multiplication (SpMV)*. For a sparse matrix with at most s entries per row and N rows we have a runtime cost for this operation of $\mathcal{O}(sN)$. This shows that the runtime cost of a sparse iterative solver is at least linear in the number of rows. Another important measure in the context of Krylow subspace methods is the *condition number* of the matrix $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$, since it affects the convergence of the algorithm [23, p. 41ff, p. 215ff]. As an example, the conjugate gradient method has a runtime complexity of $\mathcal{O}(\sqrt{\kappa}\log(\epsilon)\, sN)$ with ϵ being the approximation error.

QLS COMPLEXITY The computational complexity of the QLS algorithm can be split in terms of runtime and space complexity. Both quantities will be discussed below but we start with the space complexity.
Directly related to the size of the quantum registers is the space complexity. To store a right hand side vector of length N, we need a quantum register of length $\log_2(N)$. Additionally, we need a quantum register to store the matrix eigenvalues. The length of this register is determined by the accuracy ϵ_T and the number of bits required to encode the eigenvalues. Therefore, it can be estimated as $\kappa\log(1/\epsilon_T)$ [7, p. 224]. The full space complexity is given as

$$\mathcal{O}_{QLSA-space} = \mathcal{O}(\log(N) + \kappa\log(1/\epsilon_T)). \tag{3.42}$$

The runtime complexity of the quantum linear system algorithm is dominated by the Hamiltonian simulation subroutine since all other subroutines have lower complexity. Its complexity is given as $\mathcal{O}(\log(N)m^2\kappa^2/\epsilon)$ with m the number of submatrices which is roughly equal to the number of entries per row s. The computational complexity will be explained using a small argument. As can be seen from Figure A.1 the scaling is linear in the number of qubits and therefore logarithmic in the matrix size. Since the quantum register to hold the eigenvalues and the simulation time t_0 are proportional to the condition number the scaling is quadratic. Additionally, we have

a quadratic scaling in the number of submatrices caused by the Suzuki higher order integrators. This results in a total runtime complexity for the QLS algorithm of

$$\mathcal{O}_{QLS-time} = \mathcal{O}\left(\log(N)m^2\kappa^2/\epsilon\right). \tag{3.43}$$

Compared to the best classical methods, this is an exponential speedup in the matrix dimension. But in terms of the condition number, the scaling is worse. In most classical algorithms the scaling is proportional to the square-root of the condition number. This shows that the exponential speedup over classical algorithms is only given for problems with $\kappa \sim \mathcal{O}(\log(N))$.

3.3.2 CHALLENGES

In the last sections we have seen that the quantum linear system algorithm is a high level algorithm. This results in a bunch of details and pitfalls that need to be considered. Some of them are challenging and some of them need improvement. The next paragraphs will give answers to these questions.

SOLUTION READOUT Similar to the state preparation problem, which describes the problem of preparing a quantum state proportional to a classical vector, is the solution readout problem. At the end of the quantum linear system algorithm, we have obtained a quantum state that is proportional to the linear systems solution. The entries of the solution vector are encoded as the amplitudes of the quantum mechanical superposition. Since measuring the quantum state collapses the superposition to a specific realization, it is not possible to obtain all amplitudes with a single measurement.
Using the method of statistical sampling, it is possible to obtain the amplitudes within a certain accuracy. But this method is strongly not desirable, since it requires at least $\mathcal{O}(N)$ samples, demolishing the exponential speedup of the quantum linear system algorithm. Additionally a sufficient number of measurements can be quite high, since vector entries with a small amplitude tend to be seldomly realized.
A more robust approach to solve the solution readout requires a special problem form. If we are not interested in the full solution but in a property of the solution, like the overlap with another complex vector, the solution readout problem can be solved.
Assuming, we are interested in the overlap of the solution vector $|x\rangle$ with another vector $|R\rangle$. Corresponding to the state preparation for the quantum linear system algorithm, a state representing the vector $|R\rangle$ is prepared

$$\cos(\phi_R)\ |\tilde{R}\rangle\ |0\rangle + \cos(\phi_R)\ |R\rangle\ |1\rangle. \tag{3.44}$$

This preparation is based on the assumption that access to a quantum oracle for the vector $\boldsymbol{R}$, in the sense of the following paragraph, is granted. A more detailed discussion about the solution readout can be found in the supplementary material of Reference [4].

QUANTUM ORACLES An important part of the state preparation and the Hamiltonian simulation subroutine is the *quantum oracle* formalism. This formalism is used to describe the input of classical data into a quantum register.
Generally we can describe a quantum oracle as a black box quantum gate. Meaning that no statement about the inner workings can be made. Therefore, no estimate about runtime and space complexity of the oracle is given. To make the quantum oracle efficient, its runtime and space complexity should be negligible compared to the full algorithm. It acts on two groups of qubits which are called input and output registers. Controlled by the data encoded in the first group, it encodes data into the other one. In the QLS algorithm quantum oracles are used to prepare registers holding the classical argument and amplitude of the vector and matrix entries.
The quantum oracle formalism may offer an explanation for the possible advantage of quantum algorithms over their classical counterparts. This is due to the fact that the quantum oracle can be queried with a superposition of all possible input states. Consequently, all possible output states are obtained with a single oracle call. Allowing for a parallel data input into the quantum registers.
Even though such an abstract definition is useful to describe the desired action, it is useless in targeting an explicit realization of such an oracle. A possible approach might be to use lookup tables (LUTs). A search is carried out on the classical data, controlled by the data in the first register. A classical algorithm to fulfill this task requires at least linear time in the number of elements. Even quantum algorithms, like *Grover's search* algorithm [29], have a computational complexity of $\mathcal{O}(\sqrt{N})$. Therefore, the approach of search in a lookup table is not desirable since it increases the runtime significantly.
The only possibility left, is to do a *on-the-fly* computation. Instead of using a lookup table the information must be computed when needed. Thus, a quantum circuit is needed to calculate the data and encode it in the output register. This requires a special problem form, since the classical data must be given analytically. For the QLS algorithm it is required to have an explicit formula for the calculation of matrix values. The whole classical calculation of the matrix values must then be emulated on the quantum computer.
It is important to keep in mind that for other quantum algorithms the lookup table approach might be well suited. If their computational complexity is much higher, the lookup table approach does not add much overhead to the quantum algorithm.

DETERMINISM The last step makes the algorithm non-deterministic, which is not desirable in the sense of a reliable tool for solving systems of linear equations. This problem is due to the measurement subroutine. The rotated ancilla qubit is measured. It must yield the $|1\rangle$ state, to leave the other registers in a state proportional to the solution of the linear system. Thus, it is possible that repetitions of the algorithms yield no successful measurement. Therefore, it is desireable to change the algorithm such that this measurement subroutine is exchanged by another more reliable technique. Clader et. al solve this problem by using the *quantum*

amplitude amplification (QAA) algorithm and the *quantum amplitude estimation (QAE)* algorithm. A detailed action of these two quantum algorithms can be seen in the supplementary material to Article [4].

3.3.3 SUMMARY

In the last chapter we discussed the quantum linear system (QLS) algorithm which enables us to solve systems of linear equations on a quantum computer. After dealing with the main algorithm, we focused on the different sub algorithms involved. Most importantly we discussed the subroutine Hamiltonian simulation. A part of this subroutine will play a significant role in the Chapter 5. Additionally, we discussed the algorithmic complexity of the QLS algorithm and identified challenges when using this algorithm for an application.
The next chapter will be used to discuss a possible application for the QLS algorithm, the calculation of *radar cross section (RCS)* using the *finite element method (FEM)*. To get a basic understanding of the application we start with the study of radar cross section. Afterwards, in Chapter 5 we will discuss, how to apply the quantum linear system algorithm to the calculation of radar cross sections.

4

RADAR CROSS SECTIONS

Developed in the beginning of the 20th century, radar has become one of the most important fundamentals for modern aviation and is nowadays mainly used for airspace monitoring. But also new applications, like obstacle detection for autonomous vehicles, are arising. Based on electromagnetic scattering, it provides a simple way to detect planes in the air. When building a plane, it is essential to consider its behavior under radar. The most important single quantity to keep in mind is the *radar cross section (RCS)*. Roughly speaking, it is the ratio of scattered to incoming intensity weighted by the square of the distance between object and radar. Therefore, it is measured as an area. In the field of civil aviation the radar cross section of planes only plays a minor role. On the other hand, military aviation tries to minimize the radar cross section to make aircraft undetectable and hidden from the enemy.
Since the radar cross section of an aircraft is such an important feature, it should be considered during the development process. Today's development processes involve *virtual prototyping*. Thus, the possibility is needed to calculate the radar cross section of a possible future aircraft. Additionally, doing a measurement is often very cost-intensive and therefore not desirable. The procedure of virtual prototyping has the advantage that features can be optimized very easily and characteristics like the radar cross section can be simply recalculated.
In this chapter we want to study the radar cross section of aircraft. We focus on its calculation using the *finite element method (FEM)*, keeping in mind its usage as an application for the quantum linear system algorithm. To get a basic understanding of the radar cross section, we will provide the theoretical background for its calculation, starting with Maxwell's equations, continuing with the variational formulation and ending with the numerical solution. This journey will be mainly based on the book "The Finite Element Method in Electromagnetics" by Jian-Ming Jin [30]. At the end of this chapter we will show an example, obtained with classical linear algebra methods. In the next chapter we discuss how to calculate radar cross sections with the quantum linear system algorithm.

4.1 DEFINITION

The starting point is the definition of the radar cross section. Before giving the explicit definition, we need to specify the geometric setup (Figure 4.1). For simplicity

we will restrict the calculations to two dimensions. We use a *mono static radar* which consist of a single sender and receiver at the same place. The sender positioned at $\boldsymbol{r}_0 = (r_0, \theta_0)$ in polar coordinates emits an electromagnetic wave that travels through the space (homogenous medium like air) to the object located around $(0, 0)$ and gets reflected. The reflected wave travels back to the receiver positioned at $\boldsymbol{r}_0$ and can be detected. We want to denote the scattered field with $\boldsymbol{E_{sc}}(\boldsymbol{r})$ and the incoming

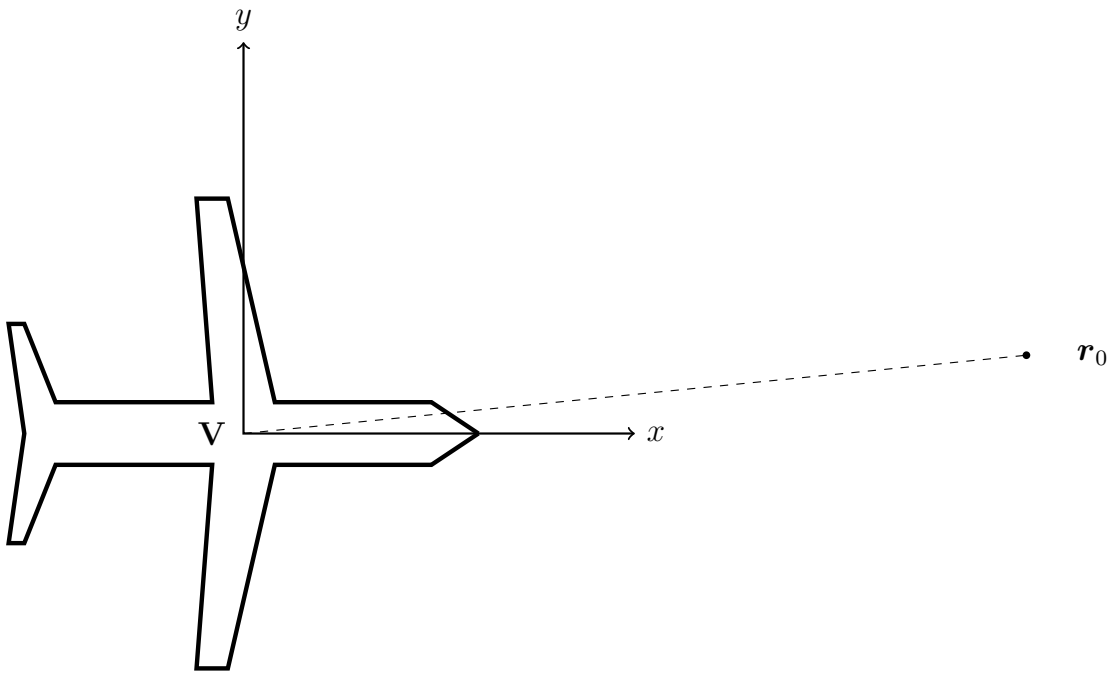

Figure 4.1: Geometric setup with object: The figure shows the object V positioned at $(0, 0)$ and the radar positioned at $\boldsymbol{r}_0$.

field as $\boldsymbol{E_{inc}}(\boldsymbol{r})$. The radar cross section [31, p.33] can now be defined as the ratio of scattered and incoming intensity in the limit of infinite distance

$$\sigma(\theta_0) = \lim_{r_0 \to \infty} 4\pi r_0^2 \frac{|\boldsymbol{E_{sc}}(r_0, \theta_0)|^2}{|\boldsymbol{E_{inc}}(r_0, \theta_0)|^2}. \tag{4.1}$$

In the calculation of the radar cross section we will omit the limiting process of the position $\boldsymbol{r}_0$ to obtain finitely quantities. This is motivated by the fact that we have to deal with a fixed observer positioned far away from the object.
As for all imaging procedures we need to consider the wavelength of the electromagnetic field as an additional parameter for the radar cross section. When considering the radar cross section as a function of the wavelength (see Figure 4.2) we can distinguish three cases [31, p. 33]. Let a denote a typical size of the object, we define the Rayleigh region as $2\pi a/\lambda \ll 1$. On the other hand we define the optical region with $2\pi a/\lambda \gg 1$ and in between we can observe the resonance or Mie region. The Rayleigh region is of high interest to radar engineers since interactions with rain and clouds are minimized. But for radar imaging the optical region is more convenient. Therefore, typical aircraft radars operate in the so called L, S, C, X, K

band which range from $0.75 \times 10^{-2}\,\mathrm{m}$ to $30 \times 10^{-2}\,\mathrm{m}$ [31, p. 8]. In the definition above (see Equation 4.1) we have left out the wavelength since we want to fix it to a finite value for the whole calculation.

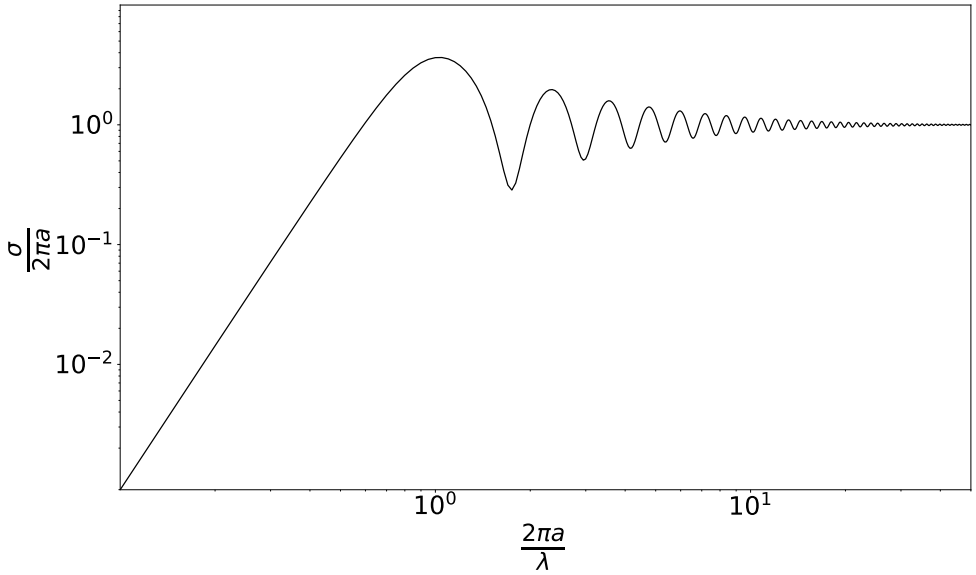

Figure 4.2: Wavelength dependence of the radar cross section: The figure shows the wavelength dependence of the radar cross section. The target is a sphere of radius a, thus the RCS can be calculated analytically. Three regions can be distinguished, the Rayleigh region for $2\pi a/\lambda \ll 1$, the Mie Region for $2\pi a/\lambda \sim 1$ and the optical region for $2\pi a/\lambda \gg 1$.

4.2 RCS CALCULATION

To calculate the radar cross section we need a description for the electromagnetic field around the object. There are many different methods to obtain such a description and in result calculate the radar cross section of a future aircraft. Two widely used methods which will not be examined any further in here are the *physical optical method* [32] and the *method of moments* [33]. We want to consider a method closely related to the method of moments, namely the finite element method since it matches the desired needs. Each of the mentioned methods has their specific advantages and disadvantages. Therefore, the insufficiencies of the chosen method have to be kept in mind.

4.2.1 DIFFERENTIAL FORMULATION

The starting point to obtain a mathematical description of the electromagnetic scattering process are the differential equations governing the behaviour of electromagnetic waves.

MAXWELL EQUATIONS Maxwell's equations are the fundamental set of equations to describe electromagnetic phenomena [34]. In a material, e.g. air, they are given as

$$\nabla \cdot \boldsymbol{B}(\boldsymbol{r},t) = 0, \tag{4.2}$$
$$\nabla \times \boldsymbol{E}(\boldsymbol{r},t) = \frac{\partial}{\partial t}\boldsymbol{B}(\boldsymbol{r},t), \tag{4.3}$$
$$\nabla \cdot \boldsymbol{D}(\boldsymbol{r},t) = \rho(\boldsymbol{r},t), \tag{4.4}$$
$$\nabla \times \boldsymbol{H}(\boldsymbol{r},t) = \boldsymbol{j}(\boldsymbol{r},t) + \frac{\partial}{\partial t}\boldsymbol{D}(\boldsymbol{r},t), \tag{4.5}$$

with the materials polarization $\boldsymbol{P}$ and magnetization $\boldsymbol{M}$.

$$\boldsymbol{P}(\boldsymbol{r},t) = \boldsymbol{D}(\boldsymbol{r},t) - \epsilon_0 \boldsymbol{E}(\boldsymbol{r},t) = (\epsilon_r - 1)\,\epsilon_0 \boldsymbol{E}(\boldsymbol{r},t) \tag{4.6}$$
$$\boldsymbol{M}(\boldsymbol{r},t) = \frac{1}{\mu_0}\boldsymbol{B}(\boldsymbol{r},t) - \boldsymbol{H}(\boldsymbol{r},t) = \frac{1}{\mu_0}\left(1 - \frac{1}{\mu_r}\right)\boldsymbol{B}(\boldsymbol{r},t)$$

The parameters ϵ_r, μ_r are the electric and magnetic permittivity of the surrounding material, e.g. air. We want to assume the absence of electromagnetic sources such that $\rho(\boldsymbol{r},t) = 0$ and $\boldsymbol{j}(\boldsymbol{r},t) = 0$.

HELMHOLTZ EQUATION To use a time-harmonic ansatz, we need to restrict ourselves to cases when the scattering does not change the frequency of the electromagnetic waves. We choose the following ansatz for the electric and magnetic field

$$\boldsymbol{E}(\boldsymbol{r},t) = \boldsymbol{E}(\boldsymbol{r})\, e^{-i\omega t}, \tag{4.7}$$
$$\boldsymbol{B}(\boldsymbol{r},t) = \boldsymbol{B}(\boldsymbol{r})\, e^{-i\omega t}.$$

Inserting this ansatz into Equations 4.3 and 4.5 we get

$$\nabla \times \boldsymbol{E}(\boldsymbol{r}) = -i\omega \boldsymbol{B}(\boldsymbol{r}), \tag{4.8}$$
$$\frac{1}{\mu_r \mu_0}\nabla \times \boldsymbol{B}(\boldsymbol{r}) = -\epsilon_r \epsilon_0 i\omega \boldsymbol{E}(\boldsymbol{r}). \tag{4.9}$$

Combining both equations results in the *Helmholtz equation* for the electric field

$$\nabla \times \nabla \times \boldsymbol{E}(\boldsymbol{r}) - \epsilon_r \mu_r k^2 \boldsymbol{E}(\boldsymbol{r}) = 0 \tag{4.10}$$

with wavevector $k = \omega\sqrt{\epsilon_0 \mu_0}$. The rotation of Equation 4.5 can then be calculated and Equation 4.3 can be inserted to obtain a Helmholtz equation for the magnetic field. We can conclude that in the absence of field sources within the time-harmonic ansatz both descriptions are equivalent.

BOUNDARY CONDITIONS The electric field $\boldsymbol{E}(\boldsymbol{r})$ is the sum of the incoming $\boldsymbol{E_{inc}}(\boldsymbol{r})$ and scattered $\boldsymbol{E_{sc}}(\boldsymbol{r})$ field. Their interaction will be induced by a boundary condition on the surface of the object V. For simplicity an object made of a perfectly

conducting material is considered. Thus, the tangential field on the surface is vanishing. In formulas we obtain

$$\boldsymbol{n} \times \boldsymbol{E}(\boldsymbol{r}) = 0 \Rightarrow \boldsymbol{n} \times \boldsymbol{E_{sc}}(\boldsymbol{r}) = -\boldsymbol{n} \times \boldsymbol{E_{inc}}(\boldsymbol{r}) \text{ at } \partial V \tag{4.11}$$

with $\boldsymbol{n}$ the normal-vector orthogonal to the surface. In principle we can now calculate the radar cross section of an object V by setting the parameters (wavelength, incoming field, etc.) and solving the Helmholtz equation to obtain the electric field at the point of observation. Then we can relate the absolute value of scattered and incoming field to obtain the radar cross section.

4.2.2 VARIATIONAL FORMULATION

Calculating the scattered field with analytical methods becomes impossible when analyzing arbitrarily shaped objects V. But it is still possible to obtain a numerical solutions for such problems. Therefore we use a variational formulation and the finite element method. The idea is to write the differential equation as an approximation problem and solve this problem on a discretised version of the geometry.

COMPUTATIONAL DOMAIN Before introducing the variational formulation, we enclose the object V with a box, the computational domain Ω. We require the computational domain to be *Lipschitz continuous* and to fully surround the object. The boundary of the computational domain shall be split into the outer boundary $\partial\Omega$ and the inner boundary between the object of interest ∂V. We need to pose conditions at the outer boundary $\partial\Omega$ since it has to be fairly distant (see [30, p. 283]) from the position of observation $\boldsymbol{r}_0$. Furthermore we impose absorbing boundary conditions at it to allow a free flow of the electromagnetic field out of this computational domain. This behaviour is approximated up to first order with the following formula

$$\boldsymbol{n} \times (\boldsymbol{\nabla} \times \boldsymbol{E}(\boldsymbol{r})) \approx -ik\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E}(\boldsymbol{r})) \text{ at } \partial\Omega. \tag{4.12}$$

The intention behind this additional geometric object is to solve the Helmholtz Equation 4.10 only within the computational domain. The motivation is the assumption that most interactions will be near the object V. Therefore, we can restrict ourselves to a domain near the object and extrapolate the electromagnetic field to the position of observation. To extrapolate the electromagnetic field we use a far field approximation, given as

$$\begin{aligned} \boldsymbol{E_{sc}}(\boldsymbol{r}_0) \approx \frac{e^{-ikr_0}}{4\pi r_0} \int_{\partial\Omega} & (ik\hat{\boldsymbol{r}}_0 \times \boldsymbol{E_{sc}}(\boldsymbol{r}) \times \boldsymbol{n} \\ & -\hat{\boldsymbol{r}}_0 \times \hat{\boldsymbol{r}}_0 \times \boldsymbol{n} \times \boldsymbol{\nabla} \times \boldsymbol{E_{sc}}(\boldsymbol{r}))\, e^{ik\hat{\boldsymbol{r}}_0 \boldsymbol{r}} dS \end{aligned} \tag{4.13}$$

with $\hat{\boldsymbol{r}}_0$ the unit vector in direction of $\boldsymbol{r}_0$ and its norm $r_0 = |\boldsymbol{r}_0|$ [30, p. 27].

GENERAL MOTIVATION Before deriving a variational formulation for the Helmholtz Equation 4.10 we want to motivate this approach. In general a differential equation can be formulated as finding a function ϕ that solves the relation $\mathcal{L}\phi = f$ with the differential operator $\mathcal{L}$ and an inhomogeneity f. Instead of finding the true solution ϕ we accept an approximate solution $\tilde{\phi}$. To find such an approximate solution we use *Galerkin's method* [35, p. 15]. A good approximate solution minimizes the weighted residual, defined as $R = \int \psi r dx$ for ψ a so called test function and $r = \mathcal{L}\tilde{\phi} - f \neq 0$ the residual of the approximate solution. The best minimization is such that $R = 0$ and thus $\int \psi \mathcal{L}\tilde{\phi} dx = \int \psi f dx$. This formulation is called weak formulation of the differential equation. The *lemma of Lax-Milgram* guarantees, under specific circumstances, that the weak solution exists and is unique [35, p. 83].

VARIATIONAL HELMHOLTZ EQUATION Now we want to obtain a weak formulation for the Helmholtz equation. Hence, we define an appropriate function space and an inner product to measure the residuals. The function space for the test functions will be given as

$$\boldsymbol{H}\left(\mathrm{curl}, \Omega\right) = \left\{ \boldsymbol{T} \in \left(L^2\left(\Omega\right)\right)^3 \text{ with } \boldsymbol{\nabla} \times \boldsymbol{T} \in \left(L^2\left(\Omega\right)\right)^3 \right\} \tag{4.14}$$

with $L^2\left(\Omega\right)$ the function space of square integrable functions [11, p. 62]. In the following, we use the standard inner product on a complex function space [11, p. 132]. It is given as

$$\begin{aligned} \langle \cdot, \cdot \rangle : \boldsymbol{H}\left(\mathrm{curl}, \Omega\right) \times \boldsymbol{H}\left(\mathrm{curl}, \Omega\right) &\to \mathbb{C} \\ \left(\boldsymbol{T}, \boldsymbol{E}\right) &\mapsto \langle \boldsymbol{T}, \boldsymbol{E} \rangle = \int_\Omega \boldsymbol{T}^\dagger \boldsymbol{E} dV. \end{aligned} \tag{4.15}$$

Using these two definitions, we can obtain a variational formulation for the Helmholtz equation

$$\int_\Omega \boldsymbol{T}^\dagger \left(\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \boldsymbol{E}\right) - \epsilon_r \mu_r k^2 \boldsymbol{T}^\dagger \boldsymbol{E} dV = \int_\Omega \boldsymbol{T}^\dagger 0 dV = 0 \tag{4.16}$$

for $\boldsymbol{T}, \boldsymbol{E} \in \boldsymbol{H}\left(\mathrm{curl}, \Omega\right)$.

BOUNDARY CONDITIONS To make this description complete, we need to add the boundary conditions (see Equation 4.11 & 4.12) to this formulation. Using Green's first Theorem A.12 we can derive the following form for the first part

$$\begin{aligned} \int_\Omega \boldsymbol{T}^\dagger \left(\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \boldsymbol{E}\right) dV = &\int_\Omega \left(\boldsymbol{\nabla} \times \boldsymbol{T}\right)^\dagger \left(\boldsymbol{\nabla} \times \boldsymbol{E}\right) dV \\ &- \int_{\partial\Omega} \left(\boldsymbol{T} \times \left(\boldsymbol{\nabla} \times \boldsymbol{E}\right)\right)^\dagger \boldsymbol{n} dS \\ &- \int_{\partial V} \left(\boldsymbol{T} \times \left(\boldsymbol{\nabla} \times \boldsymbol{E}\right)\right)^\dagger \boldsymbol{n} dS. \end{aligned} \tag{4.17}$$

On the outer boundary $\partial\Omega$ we implement the boundary conditions (see Equation 4.12) with the help of Equation A.11

$$\begin{aligned}\int_{\partial\Omega} (\boldsymbol{T} \times (\boldsymbol{\nabla} \times \boldsymbol{E}))^{\dagger} \boldsymbol{n} dS &= -\int_{\partial\Omega} (\boldsymbol{n} \times (\boldsymbol{\nabla} \times \boldsymbol{E}))^{\dagger} \boldsymbol{T} dS \\ &\approx ik \int_{\partial\Omega} (\boldsymbol{T} \times (\boldsymbol{n} \times \boldsymbol{E}))^{\dagger} \boldsymbol{n} dS \\ &= ik \int_{\partial\Omega} (\boldsymbol{n} \times \boldsymbol{T})^{\dagger} (\boldsymbol{n} \times \boldsymbol{E}) \, dS.\end{aligned} \tag{4.18}$$

The second boundary condition is incorporated into the function space. Take a look at Reference [36, p. 13] for all mathematical details. Therefore, we define a new function space as

$$\boldsymbol{H}_0 (\mathrm{curl}, \Omega) = \{\boldsymbol{T} \in \boldsymbol{H} (\mathrm{curl}, \Omega) \text{ with } \boldsymbol{n} \times \boldsymbol{T} = 0 \text{ at } \partial V\}. \tag{4.19}$$

FULL VARIATIONAL FORMULATION To obtain the full variational formulation, we define a functional F that is given as

$$\begin{aligned}F (\boldsymbol{T}, \boldsymbol{E}) = &\int_{\Omega} (\boldsymbol{\nabla} \times \boldsymbol{T})^{\dagger} (\boldsymbol{\nabla} \times \boldsymbol{E}) - \epsilon_r \mu_r k^2 \boldsymbol{T}^{\dagger} \boldsymbol{E} dV \\ &- ik \int_{\partial\Omega} (\boldsymbol{n} \times \boldsymbol{T})^{\dagger} (\boldsymbol{n} \times \boldsymbol{E}) \, dS.\end{aligned} \tag{4.20}$$

Using this new functional we can write down the full variational formulation. It is given as

$$\begin{aligned}&\text{Find } \boldsymbol{E} \in \boldsymbol{H}_0 (\mathrm{curl}, \Omega) \text{ with } \boldsymbol{E} (\boldsymbol{r}) = \boldsymbol{E}_{\boldsymbol{sc}} (\boldsymbol{r}) + \boldsymbol{E}_{\boldsymbol{inc}} (\boldsymbol{r}) \text{ such that} \\ &F (\boldsymbol{T}, \boldsymbol{E}) = 0 \text{ for all } \boldsymbol{T} \in \boldsymbol{H}_0 (\mathrm{curl}, \Omega).\end{aligned} \tag{4.21}$$

To obtain the scattered field $\boldsymbol{E}_{\boldsymbol{sc}} (\boldsymbol{r})$ from this formulation we use the relation $\boldsymbol{E} (\boldsymbol{r}) = \boldsymbol{E}_{\boldsymbol{sc}} (\boldsymbol{r}) + \boldsymbol{E}_{\boldsymbol{inc}} (\boldsymbol{r})$ which will additionally incorporate the incoming field $\boldsymbol{E}_{\boldsymbol{inc}} (\boldsymbol{r})$. This results in

$$\begin{aligned}&\text{Find } \boldsymbol{E} \in \boldsymbol{H}_0 (\mathrm{curl}, \Omega) \text{ with } \boldsymbol{E} (\boldsymbol{r}) = \boldsymbol{E}_{\boldsymbol{sc}} (\boldsymbol{r}) + \boldsymbol{E}_{\boldsymbol{inc}} (\boldsymbol{r}) \text{ such that} \\ &F (\boldsymbol{T}, \boldsymbol{E}_{\boldsymbol{sc}}) = -F (\boldsymbol{T}, \boldsymbol{E}_{\boldsymbol{inc}}) \text{ for all } \boldsymbol{T} \in \boldsymbol{H}_0 (\mathrm{curl}, \Omega).\end{aligned} \tag{4.22}$$

4.2.3 NUMERICAL SOLUTION

The next step towards solving the electromagnetic scattering problem is to find a solution to the variational formulation of the Helmholtz equation, obtained in the last section.

GENERAL MOTIVATION Gernerally speaking, solving the variational formulation of a differential equation directly with computational methods is not possible since the function space is infinite dimensional. But we can search for a solution in a

finite subspace. Therefore, we want to interpolate the approximate solution $\tilde{\phi}$ as $\tilde{\phi} = \sum_l c_l v_l$ with v_l so called basis functions. Additionally, we choose the set of test functions to be equal to the set of basis functions v_j. In total we get the following relation for every j

$$R_j = \int \left(v_j(x) \mathcal{L} \sum_l c_l v_l(x) - v_j(x) f(x) \right) dx = 0 \tag{4.23}$$

$$\Rightarrow \sum_l c_l \int v_j(x) \mathcal{L} v_l(x) dx = \int v_j(x) f(x) dx. \tag{4.24}$$

It can be written in matrix form as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with $(\boldsymbol{A})_{j,l} = \int v_j(x) \mathcal{L} v_l(x) dx$, $(\boldsymbol{x})_j = c_j$ and $(\boldsymbol{b})_j = \int v_j(x) f(x) dx$.

FINITE ELEMENT METHOD Similarly, we want to proceed for the full variational formulation of the Helmholtz Equation 4.22. But we choose a set of basis functions with specific properties. To construct these basis functions we divide the domain Ω into small subdomains or elements Ω_j. The elements do not overlap and their union is the whole domain $\Omega = \bigcup_{j \in J} \Omega_j$ with J being the element index set. Obviously we want these elements to be mathematically well-behaved like undistorted triangles or squares [35, p. 32ff]. This division of the geometry is often called geometry discretization. The boundary of two elements is called edge E_j and at each edge we define a vector valued basis function $\boldsymbol{N}_j$. The total set of basis functions is denoted as $\mathcal{N} = \bigcup_{l \in L} \boldsymbol{N}_l$ with L the index set of basis functions or edges. These basis functions have to be mathematically well-behaved as well. The collection of basis function and region is called finite element [35, p. 19f]. For the case of electromagnetic fields the class of so called Nédélec functions is a good choice [35, p. 28f]. These basis functions are highly local, e.g. for degree one we have $supp\,(\boldsymbol{N}_l) = \left\{ \bigcup_{j \in J} \Omega_j | E_l \cap \Omega_j \neq \emptyset \right\}$. This has the advantage that many integrals evaluate to zero since there is no overlap between non-neighbouring basis functions. We can rewrite the scattered electric field in terms of the basis functions as

$$\boldsymbol{E_{sc}} = \sum_{l \in L} (\boldsymbol{x})_l \boldsymbol{N}_l \text{ with } \boldsymbol{x} \in \mathbb{C}^L. \tag{4.25}$$

Additionally, we restrict the set of test functions to a discrete subset of the function space. The simplest choice for such a subspace is to use the same subspace as for the solution. Thus, we use the Nédélec functions as basis and test functions. Inserting

this interpolation and the division into subdomains into the variational formulation, yields

$$\begin{aligned}(\boldsymbol{A})_{ll'} = &\int_{\Omega} (\boldsymbol{\nabla} \times \boldsymbol{N}_l)(\boldsymbol{\nabla} \times \boldsymbol{N}_{l'}) - \epsilon_r \mu_r k^2 \boldsymbol{N}_l \boldsymbol{N}_{l'} dV \\ &- ik \int_{\partial\Omega} (\boldsymbol{n} \times \boldsymbol{N}_l)(\boldsymbol{n} \times \boldsymbol{N}_{l'})\, dS \\ = &\sum_{j \in J} \int_{\Omega_j} (\boldsymbol{\nabla} \times \boldsymbol{N}_l)(\boldsymbol{\nabla} \times \boldsymbol{N}_{l'}) - \epsilon_r \mu_r k^2 \boldsymbol{N}_l \boldsymbol{N}_{l'} dV \\ &- ik \int_{\partial\Omega \cap \partial\Omega_j} (\boldsymbol{n} \times \boldsymbol{N}_l)(\boldsymbol{n} \times \boldsymbol{N}_{l'})\, dS.\end{aligned} \tag{4.26}$$

Inside this step we have converted the variational formulation to a system of linear equations that is to find $\boldsymbol{x}$ such that

$$\begin{aligned}&\sum_{l'} (\boldsymbol{A})_{ll'} (\boldsymbol{x})_{l'} = -F(\boldsymbol{N}_l, \boldsymbol{E_{inc}}) \text{ for all } l \in L \\ &\Rightarrow \boldsymbol{Ax} = \boldsymbol{b} \text{ with } (\boldsymbol{b})_l = -F(\boldsymbol{N}_l, \boldsymbol{E_{inc}}).\end{aligned} \tag{4.27}$$

The choice of highly local basis functions was motivated by the fact that the resulting linear system is sparse, i.e. most entries in the matrix vanish. As already mentioned this is due to the fact that many integrals, part of the variational formulation, evaluate to zero. Such large scale sparse linear systems are well suited for the class iterative solvers, especially for Krylow subspace methods (see Reference [23, p. 171ff] and Page 26).

FAR FIELD APPROXIMATION The scattering problem inside the computational domain Ω is now solved. To obtain the electric field at point of observation we use a far field approximation. Inserting the Ansatz 4.25, we can rewrite the Equation 4.13 into

$$\boldsymbol{E}_{sc}(\boldsymbol{r}_0) \approx \sum_l (\boldsymbol{x})_l \boldsymbol{N'}_l(\boldsymbol{r}_0). \tag{4.28}$$

The far field extrapolation of the basis functions $\boldsymbol{N'}_l(\boldsymbol{r}_0)$ is given as

$$\begin{aligned}\boldsymbol{N'}_l(\boldsymbol{r}_0) = \frac{e^{-ikr_0}}{4\pi r_0} \int_{\partial\Omega} &(ik\hat{\boldsymbol{r}}_0 \times \boldsymbol{N}_l(\boldsymbol{r}) \times \boldsymbol{n} \\ &-\hat{\boldsymbol{r}}_0 \times \hat{\boldsymbol{r}}_0 \times \boldsymbol{n} \times \boldsymbol{\nabla} \times \boldsymbol{N}_l(\boldsymbol{r}))\, e^{ik\hat{\boldsymbol{r}}_0 \boldsymbol{r}} dS.\end{aligned} \tag{4.29}$$

The resulting radar cross section can then be calculated as

$$\sigma(\theta_0) \approx 4\pi r_0^2 \frac{\left|\sum_l (\boldsymbol{x})_l \boldsymbol{N'}_l(r_0, \theta_0)\right|^2}{\left|\boldsymbol{E_{inc}}(r_0, \theta_0)\right|^2} \tag{4.30}$$

4.3 EXAMPLE SIMULATION

To validate the theoretical concept of this section we want to take a look at a small example calculation. Hence, we have used the software package `fenics` [37–46] to implement the finite element method with first order Nédélec elements on an example geometry. The observed object is a simple representation of an aircraft (see Figure 4.3). To model the geometry of the aircraft and discretize the computational domain with triangles we use `gmsh` [47].

As an incoming electric field $\boldsymbol{E_{inc}}$ we have chosen a simple circular wave. Before we

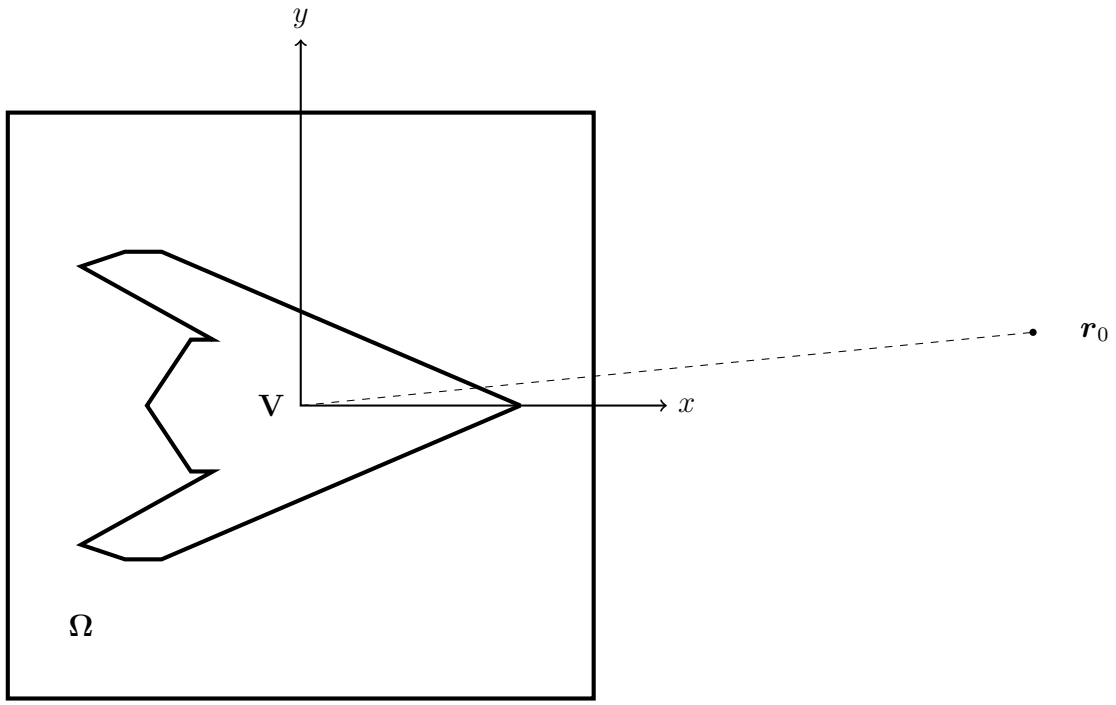

Figure 4.3: Example geometric setup: The figure shows the geometric setup with the object V positioned at $(0, 0)$ inside the computational domain Ω and the radar positioned at $\boldsymbol{r}_0$.

can actually define it, we need to take a look at the polarisation

$$\boldsymbol{p}\,(\boldsymbol{r}) = \frac{1}{|\boldsymbol{r}-\boldsymbol{r}_0|}\begin{pmatrix}(\boldsymbol{r}-\boldsymbol{r}_0)_2\\ -(\boldsymbol{r}-\boldsymbol{r}_0)_1\end{pmatrix} \text{ with } |\boldsymbol{p}(\boldsymbol{r}_0)|^2 = 1. \tag{4.31}$$

Now we can specify the incoming field as

$$\boldsymbol{E_{inc}}(\boldsymbol{r}) = E_0\boldsymbol{p}\,(\boldsymbol{r})\,e^{-i\boldsymbol{k}(\boldsymbol{r}-\boldsymbol{r}_0)} \tag{4.32}$$

with wave-vector $\boldsymbol{k} = -k\left(\widehat{\boldsymbol{r}-\boldsymbol{r}_0}\right) = -2\pi\lambda\left(\widehat{\boldsymbol{r}-\boldsymbol{r}_0}\right)$ and $\widehat{\boldsymbol{r}}$ indicating the unit vector in direction $\boldsymbol{r}$. The resulting complex linear system is solved by a *Krylow subspace method*, the *generalized minimal residual (GMRES)* solver. Choosing the typical diameter of the elements in the finite element method is a difficult task. On the one hand larger elements reduce the computational effort, on the other hand

smaller subdomains tend to reduce the approximation error. To get good results, we choose the size such that it is much smaller than the wavelength. All other chosen parameters can be read out of the following table.

h [m]	E_0 [V/m]	$\epsilon_r - 1$ [A s/(V m)]	$\mu_r - 1$ [N/A^2]	r_0 [m]
5×10^{-2}	1.94×10^4	5.89×10^{-4}	3.7×10^{-7}	1×10^4

Table 4.1: Example simulation parameters: The table shows the parameters used for the example simulation.

With the solution of the linear system we can then calculate the radar cross section using the far-field approximation as

$$\sigma(\theta_0) = \frac{4\pi r_0^2}{E_0^2} |\boldsymbol{E_{sc}}(r_0, \theta_0)|^2. \tag{4.33}$$

To obtain the full radar cross section we repeatedly calculate it with different angles θ_0 resulting in the following figures. One shows the absolute value of the scattered field inside the computational domain and the other one the calculated radar cross section for different angles of θ_0. The plot consists of 560 discrete values for θ_0 equally spaced in the interval $[0, 2\pi]$.

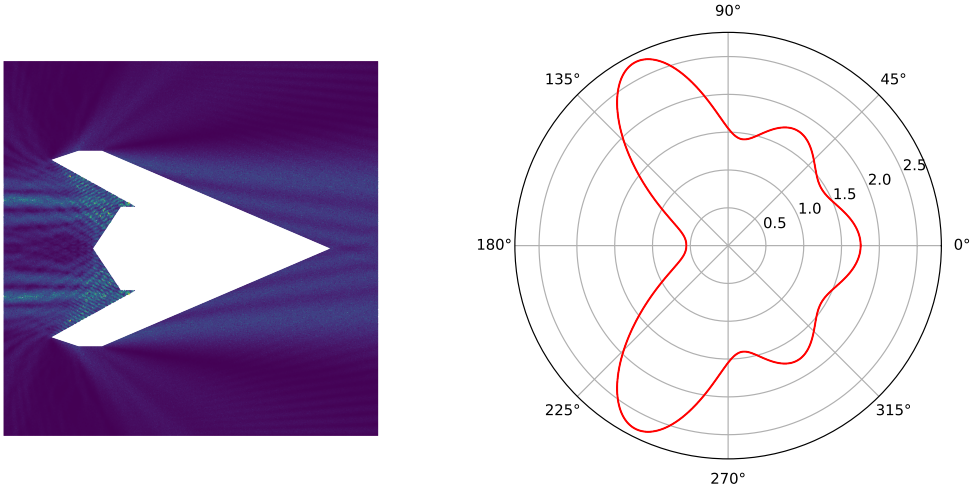

Figure 4.4: Scattered field intensity and radar cross section for a plane-like target:
Left: The left figure shows the scattered electric field intensity inside the computational domain for a plane-like target. An incidence angle of $\theta = 0$ and a wavelength of $\lambda = 7 \times 10^{-1}$ m was used.
Right: The right figure shows the radar cross section of the geometric shape on the left. An incidence angle of $\theta \in [0, 2\pi]$ and a wavelength of $\lambda = 70$ m was used. The wavelengths are distinct for a better visualization of the involved phenomena.

4.4 SUMMARY

In the last chapter the calculation of radar cross section using the method of finite elements was discussed. Starting with the definition of radar cross section and Maxwell's equations, we derived a differential formulation for the problem. Afterwards, we used a variational formulation to obtain a numerical solution. Finally, an example for such a calculation was given. The next chapter will be used to discuss the calculation of radar cross section using the quantum linear system algorithm.

5

RCS CALCULATION USING THE QLS ALGORITHM

As outlined in Subsection 2.2.3, finding applications for quantum algorithms can be a difficult task. Whereas the quantum linear system algorithm seems to solve an elementary problem, it requires a special problem setting to be applicable. A suitable application allows to circumvent or fix the challenges of the used algorithm, by generating only a small overhead in computational complexity.
The calculation of radar cross sections using the finite element method as described in the last chapter is an application that goes well with the quantum linear system algorithm. On the one hand it requires the solution of a large sparse linear system but more importantly this application does not require the full solution vector. Only the overlap of the solution with another vector as shown in Equation 4.30 needs to be calculated. This becomes possible using the method described on Page 27. However it is not straightforward to apply the quantum algorithm to this problem. This is due to the *quantum oracle* formalism used to input the classical matrix data.
In this chapter we want to work out the implementation details for the matrix oracle. To estimate the hardware requirements, we focus on the complexity cost in terms of quantum gates. We start with a short recapitulation about quantum oracles, especially the matrix oracle. It is followed by a section about its possible classical implementation, focusing on the restrictions for the finite element method. In the next step, we estimate the classical cost in terms of elementary operations. Afterwards, we define metrics to estimate the quantum circuit cost. Using these metrics, we estimate the resulting cost in terms of quantum resources for our example implementation. This estimation will be based on a classical to quantum cost translation. The chapter will be finished by discussing the results.

5.1 QUANTUM ORACLES

This chapter starts with a recapitulation about quantum oracles. The quantum oracle has been introduced in the context of Hamiltonian simulation, see Subsection 3.2.3. Furthermore, we have pointed out specific properties of quantum oracles in general (see Page 28). In the quantum linear system algorithm quantum oracles are needed to prepare an input state corresponding to the right hand side of the linear system (see Page 24), the state preparation for the state overlap, i.e. readout, (see Page 27)

and for the matrix input (see Page 21). In the following, we focus on the matrix oracle.

MATRIX ORACLE The matrix oracle is used to encode the classical matrix data into quantum registers. Since we require both argument and amplitude of the matrix values, two oracles need to be defined. Formally their action is given as follows

$$\mathrm{U}_c^m \left|a\right\rangle \left|b\right\rangle \left|y\right\rangle \left|z\right\rangle = \left|a\right\rangle \left|b \oplus \mathrm{col}_c\left(a\right)\right\rangle \left|y \oplus \mathrm{amp}_c\left(a\right)\right\rangle \left|z\right\rangle \tag{5.1}$$

$$\mathrm{U}_c^p \left|a\right\rangle \left|b\right\rangle \left|y\right\rangle \left|z\right\rangle = \left|a\right\rangle \left|b \oplus \mathrm{col}_c\left(a\right)\right\rangle \left|y\right\rangle \left|z \oplus \mathrm{arg}_c\left(a\right)\right\rangle . \tag{5.2}$$

The oracle acts on four registers, each encoding a specific value. Whereas the first register is the input register, i.e. it controls the oracle's action, the other registers are used for output. The first register holds the row a and the second holds the column $\mathrm{col}_c\left(a\right)$. The third and fourth register hold the amplitude $\mathrm{amp}_c\left(a\right)$ and argument $\mathrm{arg}_c\left(a\right)$ of the matrix values. The first two registers use an integer encoding, the last two need to hold real values. This can be done using either fixed point or floating point encoding (see Page 8). The index c indicates the submatrix. Since all submatrices are one sparse, the column associated to a row in a submatrix is unique. Consequently, the amplitude and the argument are also unique. The oracle can be queried by a superposition, allowing for a parallel input of the matrix values.

5.2 CLASSICAL RESOURCE ESTIMATION

As already discussed on Page 28, an efficient implementation of the matrix oracle for the QLS algorithm requires *on-the-fly* computation of the matrix values. Therefore, the finite element method must be run on the quantum computer. Thus an efficient implementation of the finite element method is needed, which computes matrix values when required. This is comparable to a *matrix free* version such that the full matrix is never explicitly calculated.
The following subsection will be used to describe the construction of a classical implementation for the matrix oracle. Starting the query by a specific row and submatrix, the column and argument or amplitude of the matrix value are calculated.

5.2.1 FEM IMPLEMENTATION

The implementation of the finite element method for radar cross section calculation has been discussed in Section 4.2. Most importantly, we use the same computational domain as described there but a different target, specified later on.

DISCRETIZATION As discussed in Section 4.2, the computational domain Ω needs to be discretized. This is done by dividing it into subdomains Ω_j. The structure of this discretization or mesh has a high influence on the sparsity pattern of the resulting matrix. In the case of an unstructered mesh, a lookup table (LUT) is used

to relate the region's index j to it's geometric parameters, e.g. corner points of a triangle. Using such a lookup table on the quantum computer is not efficient if it is used for the matrix oracle (see Page 44). To get an analytical relation between region index and its geometrical parameters, we need to use a structured mesh. Since we ware doing a two dimensional analysis, the simplest mesh structure is made of squares with edge length h. We use this mesh since a simple relation between region index and geometrical parameters allows an efficient computation. Thus, fewer quantum resources are needed.
The basis and test functions of choice to simulate electromagnetic phenomena are Nédélec functions (see Subsection 4.2.2). These basis functions are defined per edge. Therefore, we will not focus on the number of vertices in the mesh but on the number of edges. Assuming the number of elements Ω_j is H^2, there are $N = 2H^2 + 2H$ edges. The numbering scheme for the edges can be seen in Figure 5.1. There are nine different bands in the matrix and since each central edge is surrounded by seven other edges there are as many entries per matrix row. Given the row and the band, it is possible to calculate the corresponding edge and thus the column of this matrix entry.

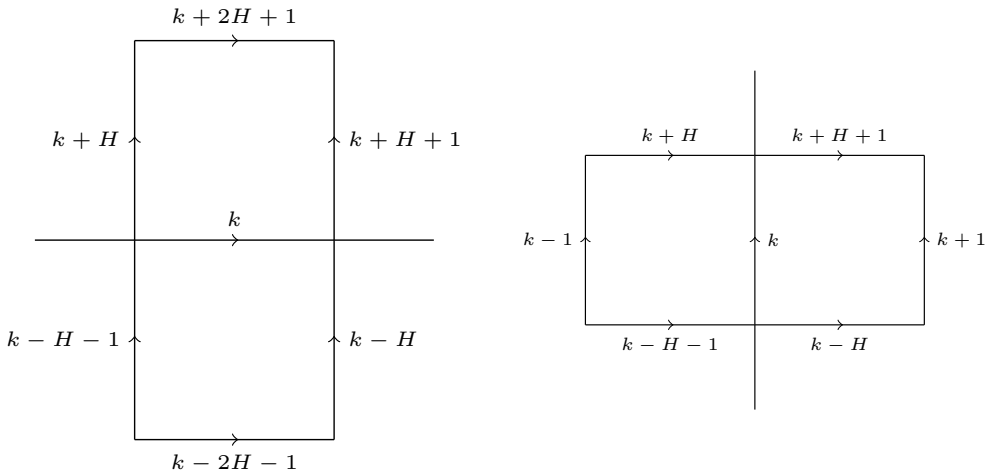

Figure 5.1: Square Mesh: The figure shows the numbering scheme for neighbouring edges. *Left*: horizontal central edge, *Right*: vertical central edge.

GEOMETRY The next step towards the calculation of the matrix entries is to calculate the type of geometric object on which the edge is situated, i.e. target, boundary or intermediate medium. The edge's number, i.e. row or column, allows for the calculation of the mathematical coordinates of the edge's mid point. These coordinates can then be used to identify the type of the geometric object. For example, if the coordinates are equal to the boundary of the computational domain, the edge should be attributed to this boundary. In this case the matrix values take a specific form which will be discussed in the following. A problem occurring at this point is to identify the object of interest. For this object a kind of *geometry oracle* must be given. This oracle takes a geometric position as an input and returns either true or

false if the edge is on the object or not. The problem of such an geometry oracle will also be discussed in Section 5.4.

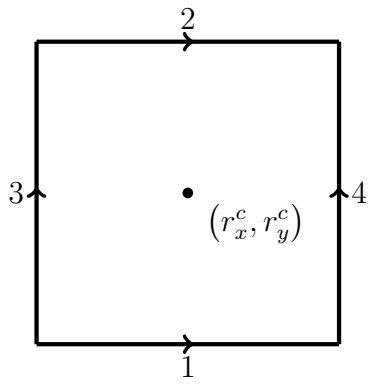

Figure 5.2: Numering scheme for edges: The figure shows the numbering scheme for edges within an element. Additionally, the edge orientation is given.

MATRIX ENTRIES Using the defined discretization and the information about the geometry type, the construction of the matrix entries can be started. Equation 4.27 describes the relation between basis functions, geometry and matrix values. It is given as

$$\begin{aligned}(\boldsymbol{A})_{ll'} = &\sum_{j\in J}\int_{\Omega_j} (\boldsymbol{\nabla}\times\boldsymbol{N}_l)^\dagger (\boldsymbol{\nabla}\times\boldsymbol{N}_{l'}) - \epsilon_r\mu_r k^2 \boldsymbol{N}_l^\dagger \boldsymbol{N}_{l'} dV \\ &- ik\int_{\partial\Omega\cap\partial\Omega_j} (\boldsymbol{n}\times\boldsymbol{N}_l)^\dagger (\boldsymbol{n}\times\boldsymbol{N}_{l'})\, dS.\end{aligned} \tag{5.3}$$

The index set J counts all subdomains Ω_j and the index set N counts all edges. Thus the matrix is of dimension $N\times N$, following the notation of Chapter 3. We define the numbering and orientation scheme for the squares with edge length h and center $\boldsymbol{r}^c = \left(r_x^c, r_y^c\right)$ as in Figure 5.2. The Nédélec basis functions of first order are given as

$$\begin{aligned}\boldsymbol{N_1}(\boldsymbol{r}) &= \frac{1}{h}\left(r_y^c + \frac{h}{2} - r_y\right)\hat{\boldsymbol{r}}_x, \qquad \boldsymbol{N_2}(\boldsymbol{r}) = \frac{1}{h}\left(r_y - r_y^c + \frac{h}{2}\right)\hat{\boldsymbol{r}}_x, \\ \boldsymbol{N_3}(\boldsymbol{r}) &= \frac{1}{h}\left(r_x^c + \frac{h}{2} - r_x\right)\hat{\boldsymbol{r}}_y, \qquad \boldsymbol{N_4}(\boldsymbol{r}) = \frac{1}{h}\left(r_x - r_x^c + \frac{h}{2}\right)\hat{\boldsymbol{r}}_y.\end{aligned} \tag{5.4}$$

Since we have chosen highly local basis and test functions, we only get contributions if edges j and j' are connected to each other. The resulting integral simplifies to the subregion that contains both edges. Thus, the sum in Equation 5.4 vanishes. Accordingly, we can evaluate the integrals depending on the four types of edges E_l

per subdomain Ω_j. For simplicity we will set $\epsilon_r = \mu_r := 1$. The volume integrals are given as

$$\int_\Omega (\boldsymbol{\nabla} \times \boldsymbol{N}_l)^\dagger (\boldsymbol{\nabla} \times \boldsymbol{N}_{l'}) - k^2 \boldsymbol{N}_l^\dagger \boldsymbol{N}_{l'} dV \tag{5.5}$$
$$= \begin{cases} +1 - \frac{k^2 h^2}{3}, & E_l \text{ and } E_{l'} \text{ equal,} \\ -1 + \frac{k^2 h^2}{12}, & E_l \text{ and } E_{l'} \text{ parallel, e.g. 1 and 2,} \\ -1, & E_l \text{ and } E_{l'} \text{ do not follow up, e.g. 1 and 3,} \\ +1, & E_l \text{ and } E_{l'} \text{ do follow up, e.g. 1 and 4.} \end{cases}$$

The surface integral over the absorbing boundary can be evaluated similarly. A contribution is given if one edge or both are on the boundary. The values are

$$ik \int_{\partial\Omega} (\boldsymbol{n} \times \boldsymbol{N}_l)^\dagger (\boldsymbol{n} \times \boldsymbol{N}_{l'}) \, dS = \begin{cases} ikh, & E_l \text{ and } E_{l'} \text{ equal,} \\ 0, & \text{otherwise.} \end{cases} \tag{5.6}$$

There is only one case left. This case occurs if one edge is on the object. Thus the boundary condition of a perfect conducting surface needs to be considered. This results in vanishing matrix values on the off diagonals and the diagonal value is set to 1.

CONCLUSION Using the described restrictions above, it is possible to implement the finite element method such that each matrix value can be calculated analytically. This only requires some fixed parameters like wavelength or edge length and the explicit parameters row and submatrix. Additionally, it is possible to implement the boundary conditions which also effects the right hand side quantum oracle.

5.2.2 CLASSICAL OPERATION COST

In classical computer science the metric for runtime is the number of floating point operations. This is due to the fact that this type of operations is very resource demanding, but they take an essential part of most algorithms. The two most fundamental operations are addition and multiplication, both have comparable runtime cost. In fact most modern computers are equipped with special units, making the two operations equivalent in terms of runtime (FMA Instructions). Furthermore, the runtime of more complex operations, e.g. division, can be estimated as a multiple of the runtime of the fundamental operations. Hence, we use the number of floating point operations as a metric to estimate the cost of the classical implementation.

TEST SYSTEM To estimate the classical cost for the matrix oracle, we have implemented the finite element method as described above in the programming language `python`. As will be discussed in Section 5.4 and was also discussed before, an oracle for the geometric features of the target is needed. As a simple example, we have chosen a

target that has a circular shape. Thus, an analytical description is available and the implementation of the geometry oracle, i.e. the computation of the edge's location, is efficient. The whole test system can be described as follows. The circular target of diameter 8 m is centered inside a quadratic computational domain with outer edge length 10 m. The computational domain is divided into square elements with edge length 1 m. The incoming electromagnetic field has a wavevector of $k = 1/2\,\mathrm{m}^{-1}$.

CLASSICAL RESULTS We have counted the number of required floating point operations and listed the most important of them in Table 5.1 and 5.2. To estimate the impact of the geometry oracle cost, there are two tables, one excluding and one including this cost. The resulting matrix is banded and has nine different bands. Since there is the possibility of implementation dependent differences, all nine different bands are given in the tables. To make them comparable, a typical number of operations per band is derived. In terms of classical computation addition and multiplication have the same cost, however in the context of quantum computation these operations have different cost. Therefore, the numbers are given for both operations. By construction this classical cost for the quantum oracle is more or less independent of the chosen parameters for the test system. This is due to the fact that all matrix values have analytical descriptions and thus there is no runtime dependence on the mesh size.

5.3 QUANTUM RESOURCE ESTIMATION

After the successful implementation of the classical oracle, we want to obtain an estimate for the quantum oracle cost or more specifically the quantum circuit cost. This estimation will be based on the emulation of classical floating point operations. Before the quantum cost can be considered, a reliable metric is needed.

5.3.1 CLASSICAL EMULATION

REVERSIBILITY The different methods for number representation have been discussed in Subsection 2.1.3. Since the finite element method requires high precision and the multiplication of very small and large numbers, we need to choose the floating point representation. Obviously, this representation has a much higher complexity than the fixed point representation. Additionally, this representation has also been used in the classical implementation.
To implement the method of finite elements, different classical operations are needed. Examples are addition and multiplication for floating point numbers. However, most classical operations are not reversible. If we take a look at addition for example, the classical circuit has two inputs but only one output. Therefore, information is lost when adding two numbers. Since quantum circuits are unitary, a direct translation

Operation	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6	Band 7	Band 8	Band 9	Typical
Addition	30	29	29	29	29	29	29	29	30	29
Subtraction	6	7	7	8	9	8	7	7	6	7
Multiplication	33	32	32	32	33	32	32	32	33	32
Division	29	27	27	26	29	26	27	27	29	27
Square	14	12	12	12	14	13	12	12	14	13
Comparator	12	12	12	12	12	12	12	12	12	12

Table 5.1: Classical operations per matrix band **excluding** target cost: The table shows the number of typical, classical floating point operations per matrix band. This estimation excludes the target cost.

Operation	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6	Band 7	Band 8	Band 9	Typical
Addition	34	33	33	34	33	34	33	33	34	33
Subtraction	6	7	7	8	9	8	7	7	6	7
Multiplication	33	32	32	33	33	33	32	32	33	32
Division	33	31	31	31	33	31	31	31	33	32
Square	26	24	24	26	26	26	24	24	26	25
Comparator	16	16	16	16	16	16	16	15	16	16

Table 5.2: Classical operations per matrix band **including** target cost: The table shows the number of typical, classical floating point operations per matrix band. This estimation includes the target cost.

from classical to quantum operations is not possible. Nevertheless, one can construct a classical reversible circuit for the desired action. Afterwards, this reversible circuit is mapped to a quantum circuit. We want to regard this as the emulation of classical operations using quantum circuits.
The theory of classical reversible circuits has a long tradition as the theory of classical non reversible circuits also has. Although seldomly used in actual hardware, it has recently drawn the attention of many computer science researchers. This is only partly related to the close connection to quantum computation but mainly by the possibility to give solutions for modern circuit design problems [48].
In the following paragraph we discuss the construction of classical reversible circuits and its translation to quantum circuits called circuit design or synthesis.

CIRCUIT SYNTHESIS The approach used in this thesis is based on automata theory, more precisely on the *LUT-based hierarchical reversible logic synthesis (LHRS)* as introduced by Soeken, Roetteler et al. in References [49–52]. Another approach worth mentioning is based on the *quantum lambda calculus* and was introduced by Selinger et al. in References [53, 54].
The starting point for our approach is a lookup table (LUT) network for the desired mathematical operation. Such LUT networks can be used to realize arbitrary *Boolean* functions. Intermediate results are stored in ancilla states and thus require additional space. Based on this LUT network, a reversible network is designed by the mapping described in the references. The reversible circuit can then be mapped to a quantum circuit. By reusing already uncomputed ancilla states, it is possible to use fewer of them. Using this synthesis procedure it is possible to derive quantum circuits emulating simple classical floating point operations. The advantage of such an automated Synthesis approach is that it becomes possible to allow for different optimization stages. Hand crafted circuit designs show better quality but they are hard to achieve, especially for more complex operations. To design circuits for advanced operations, the approach of iterative approximative methods can be used (see Reference [55]).

5.3.2 QUANTUM METRICS

IMPLEMENTATION COMPLEXITY For classical algorithms there are a handful of implicitly standardized metrics to measure the complexity of a specific algorithmic implementation. One can count the usage of memory or the absolute runtime. For quantum algorithms such metrics have not been standardized yet.
On the one hand we can observe initiatives to measure the computational capabilities of quantum hardware. On the other hand there are efforts to invent metrics to measure the quantum computational effort of quantum algorithms. Since none of the above mentioned inventions is widely accepted by now, we want to stick to more classical metrics. Two simple metrics are the gate count and the qubit count. More precisely, this is the number of qubits needed and the number of gate operations

performed to run the algorithm successfully. Additionally, both metrics have the advantage that they are hardware independent. But we have to deal with the disadvantage that hardware dependent optimizations have not been considered.
The quantum linear system algorithm will require quantum error correction. Additionally, there is no self repair mechanism in this algorithm thus errors are propagated and accumulated. Therefore, QEC must be considered as a key feature when estimating the circuit cost. Since the exact error correction scheme is unknown and might be hardware dependent, the estimation will be in terms of logical qubits and gates.

QUBIT METRICS A possible approach for a metric is to count the number of logical qubits used in the quantum circuit. If an exact quantum circuit is given, it is straightforward to count the number of qubits. But we are using a mapping from classical to quantum circuits, therefore we can only estimate the number of qubits. Since additional qubits are needed for the program logic it is hard to estimate the overall count.
Additionally, the scaling in the number of qubits of quantum architectures is unknown. Therefore, it is hard to predict the number of qubits which can be used by a quantum algorithm in the near future.

GATE COUNT METRICS In consequence we are left with gate metrics as an approach to measure the implementation complexity of the quantum circuit. As already mentioned, it is necessary to take quantum error correction into account. Just counting the number of quantum gates does not yield reliable results since only a small subgroup of gates can be implemented in an error-free manner. The idea is to choose this gate set such that all other gates can be represented or approximated by this small set. Such a subset is called universal. The full procedure is described in the following and can also be seen on page 188ff in Reference [7] and in Reference [18].
The first result regarding the universality of quantum gates is that any n qubit gate can be decomposed into a sequence of at most $2^{n-1}\left(2^n-1\right)$ one and two qubit gates [56]. Most often this number can be improved drastically. Additionally it turns out that any two qubit gate can be decomposed into a sequence of one qubit gates and CNOT gates [57]. Therefore only a single type of two qubit gates in combination with one qubit gates is needed to construct all others. Up to this point all decompositions are exact. Restricting the continuous set of single qubit gates to a discrete will require approximations. Each single qubit gate can be seen as a rotation and therefore it is possible to implement this rotation approximately. Within this thesis, we used the

universal gate set consisting of the Hadamard H, Phase P, $\pi/8$ T and CNOT gate [58]. The matrix representation of these gates is given as follows

$$\mathrm{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \qquad \mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{5.7}$$

$$\mathrm{P} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix}, \qquad \mathrm{H} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

It is important to note that the phase gate P is distinct from the value controlled phase gate as introduced in Subsection 3.2.4. Likewise important is the fact that the T gate is sometimes called $\pi/8$ gate, despite the fact that its action is related to $\pi/4$. In conclusion, counting the occurences of these four gates is an interesting metric under the consideration of quantum error correction (QEC) to estimate the quantum circuit cost.
However, we will only count the number of T gates. This is motivated by the fact that this gate is much more costly on all quantum computing architectures. To apply this gate the method of *magic state destillation* needs to be used which is very costly in terms of computational resources [59]. Therefore the number of T gates is a good estimate for the complexity of a quantum circuit.

5.3.3 QUANTUM OPERATION COST

The method described in Subsection 5.3.1 has been implemented by Soeken et al. in the software `revkit` [60]. This software allows for the synthesis of quantum circuits emulating classical operations. The generated circuits are build of the universal gate set as described in Subsection 5.3.2 and can be found in Reference [49]. There are three parameters that can be optimized, the number of qubits, the number of T gates and the LUT network depth. It turns out that there is an inverse relation between the number of qubits and T gates. Thus, optimizing for one parameter results in a poor value for the other and vice versa. For the operations, used in the classical implementation of the quantum oracle, the quantum cost are given in Tables 5.3 and 5.4. In these tables both extremes are given, the best number of qubits and the best number of T gates. Additionally, the results are given for both 16 bit floating point and 32 bit floating point representation. It turns out, that the order of magnitude for the qubits is approximately equal in both scenarios. However the number of required T gates is extremely high in case of the best qubit count. This is an additional motivation to use the best number of T gates. It might also be a good idea to use a parameter set, resulting in an nearly optimal T gate count but not fully non optimal qubit count. Another approach, not considered here is to use circuit optimization techniques to optimize the high T gate count in case of an nearly optimal qubit count.

Using the resulting cost for the emulation of the classical operations, we can estimate

Operation	Best Qubit Count		Best T Gate Count	
	Qubits	T Gates [10^3]	Qubits	T Gates [10^3]
Addition	141	723	367	12
Subtraction	141	760	368	11
Multiplication	229	6533	931	28
Division	112	8378	375	13
Square	32	196	191	5
Comparator	38	35	97	3

Table 5.3: Resources for classical operations on 16 bit floating point numbers: The table shows the required quantum resources for emulating typical classical operations like addition or multiplication. Two cases are given that correspond to the minimal number of qubits or the minimal number of T-gates.

Operation	Best Qubit Count		Best T Gate Count	
	Qubits	T Gates [10^3]	Qubits	T Gates [10^3]
Addition	350	838	819	26
Subtraction	337	564	829	26
Multiplication	778	6839	2640	89
Division	905	111 875	2007	48
Square	194	8415	894	20
Comparator	74	992	193	5

Table 5.4: Resources for classical operations on 32 bit floating point numbers: The table shows the required quantum resources for emulating typical classical operations like addition or multiplication. Two cases are given that correspond to the minimal number of qubits or the minimal number of T-gates.

the cost of a single quantum oracle call. Using the results from Tables 5.1, 5.2 for the test system and Tables 5.3, 5.4, we can calculate the total cost by multiplying the number of operations by their individual resource requirements. The resulting costs can be seen in Table 5.5. Both cases have been considered, including and excluding the oracle cost for the geometric shape of the target. Also both floating point precisions are given. In total we can estimate the typical number of operations in an oracle call to roughly a hundred. The resulting number of T gates is ~ 1.6 million for 16 bit floating point (FP) precision and ~ 5.4 million for 16 bit FP precision. From the numbers we can deduce that the additional cost for the geometry oracle is important but will not change the order of magnitude in our estimation. Thus, considering the required resources, the calculation of arbitrarily shaped objects seems possible. Furthermore, the required resources are more or less independent of parameters

employed for the test system. Therefore, it is possible to simulate a higher mesh resolution without additional computational complexity in the context of the matrix oracle.

Operation	Ex. Target T-Gates [10^3]			In. Target T-Gates [10^3]		
	Typical	16 bit FP	32 bit FP	Typical	16 bit FP	32 bit FP
Addition	29	333	755	33	379	860
Subtraction	7	80	184	7	80	184
Multiplication	32	896	2834	32	896	2834
Division	27	339	1303	32	402	1545
Square	13	69	261	25	132	502
Comparator	12	30	56	16	40	74
Total	120	1647	5392	145	1928	5997

Table 5.5: Resources per band for 16 bit and 32 bit FP numbers: The table shows the quantum resources required to compute the matrix values per band. There are two estimations, one including and one excluding the target cost. Additionally, numbers for both 16 bit and 32 bit floating point precision are shown.

5.4 OPEN PROBLEMS

CONDITION NUMBER In Subsection 3.2.3 we discussed the computational complexity of the subroutine *Hamiltonian simulation*. We estimated the time complexity as $\mathcal{O}\left(\log(N)\frac{1}{\epsilon}\kappa^2 s^2\right)$ with ϵ the error, N the number of matrix rows, s the number of entries per row and κ the matrix condition number. For the finite element method the typical scaling in the condition number is given as $\kappa(\boldsymbol{A}) \sim h^{-2}$ with h being a measure for the element diameter Ω_j [61–63]. In conclusion the time complexity grows with a factor four when doubling the mesh resolution. Since our interest lies in very fine meshes, respectively large linear systems, we are interested in small element diameters resulting in large condition numbers. A possible solution for this open question is *preconditioning*.
Instead of solving the given linear system $\boldsymbol{Ax} = \boldsymbol{b}$, a modified linear system

$$\boldsymbol{MAx} = \boldsymbol{Mb} \tag{5.8}$$

with an additional matrix $\boldsymbol{M}$, of the same size as $\boldsymbol{A}$, is solved. The idea is to choose $\boldsymbol{M}$ such that the condition number of the matrix $\boldsymbol{MA}$ is smaller than the condition number of $\boldsymbol{A}$. To make this idea useful we also require $\boldsymbol{M}$ to be efficiently computable. In the context of classical iterative solvers (see Page 26), where preconditioning is widely used, this corresponds to an efficient computation of the matrix vector product. For the quantum linear system algorithm an efficient implementation

is related to an efficient quantum oracle (see Page 28) for the matrix. Additionally, an efficient implementation of the altered right hand side in oracle form is needed.

GEOMETRY ORACLE Another open problem is the complexity of the *geometry oracle*. In out previous discussion of the implementation of the finite element method on the quantum computer, the geometry oracle has been introduced. It is used to check if a point inside the computational domain is on, in or outside the object. Similar to the matrix oracle, the geometry oracle can be seen as a black box with unknown inner workings.
For certain types of geometries an implementation is fairly easy since the geometry can be represented by an analytical formula. Thus, the computation is efficient and can be emulated on the quantum computer using the strategy described in the last chapter. A good example for such an efficiently computable geometry is the circle as used in Subsection 5.2.1. For arbitrary shaped objects like aircraft this remains unknown. A possible approach might be offered by B-splines.

5.5 SUMMARY

In the last chapter we discussed, how to implement a quantum oracle for the input of classical data. In particular, we considered an exemplary implementation of the matrix oracle for the quantum linear system algorithm and its application the calculation of radar cross sections. It turned out that the classical finite element method needs to be reimplemented on the quantum computer, leading to the emulation of classical operations. We estimated the cost for this emulation in terms of T gates for a test system. Additionally, we separated the circuit cost for the geometric shape of the target, to show that it has only a minor influence on the total cost.

6

SUMMARY

Finally, we want to sum up and discuss the results. Furthermore, possibilities for improvement are shown. We end with an outlook and some final remarks.

RECAPITULATION In the present thesis, the quantum linear system algorithm and its application to the calculation of radar cross sections has been discussed. At the beginning we focused on the basics of quantum computation. Afterwards, we took a detailed look at the QLS algorithm and the involved subroutines. We verified the computational complexity of the algorithm, especially the logarithmic speedup in the matrix dimension compared to classical methods. It turned out that the main challenge lies in the input of classical data. From a theoretical point of view this has been solved by introducing the formalism of a black box quantum oracle. This solution is not satisfying from an implementation point of view. Thus, we have described the actual realization of a quantum oracle. After deriving a possible metric for quantum circuits, we were able to perform an exemplary study of the quantum resources required to implement the matrix oracle. This estimation showed that a substantial number of quantum resources is required, namely the order of a million T gates. In the context of quantum error correction this type of gates can be regarded as the most demanding. Additionally, we were able to separate the cost for the geometric description of the target. This is important since more complex shapes require substantially more resources. Comparing the results to the capabilities of today's quantum devices shows the requirement for more advanced quantum devices. Furthermore, a serious estimate in terms of a timeframe is not possible due to the requirement of fundamental advances in quantum computation.

POSSIBLE IMPROVEMENTS Since the required resources for a successful implementation of the quantum linear system for the proposed application are quite high, there is still room for improvements. Additionally, loosening the employed restriction might allow for more reliable solutions.
At first we want to mention improvments on the algorithmic level regarding the quantum linear system algorithm. Its central part is the subroutine Hamiltonian simulation (see Subsection 3.2.3). As Hadfield et al. have pointed out in Reference [64], it might be desirable to use different levels of division for different submatrices. Thus, allowing for fewer operations, if the desired accuracy can be kept due to

properties of the submatrices. More importantly, it seems necessary to tackle the open problem regarding the condition number. One approach might be to find a new algorithm for the Hamiltonian simulation with much better dependence on the condition number.
Another area that allows for further improvement is the implementation of quantum oracles, especially the matrix oracle. More precisely, it is required to improve and optimize the emulation of the classical operations. It might be possible to use a mixed number representation, i.e. some numbers in fixed point representation, some in 16 bit floating point representation and some in 32 bit floating point representation. A possible advantage of this mixture is that only the required precision is used, thus freeing unnecessarily occupied resources.

OUTLOOK AND FINAL REMARKS Even though a serious estimate about the timeframe of a possible realization of the quantum linear system algorithm is not possible, we want to mention enhancements required to allow for a flexible tool for the calculation of radar cross section. In the calculation on Page 33 the restriction to perfectly conducting materials was made, real materials do not show such a behaviour. To model imperfect conductors it is required to employ other boundary conditions, thus enlarging the complexity of the finite element method.
Additionally, the condition of stationary scattering is rather strong. Weakening this condition will allow for more elaborate simulations, even tough a new computational approach is required. Highly interesting materials, from the perspective of a radar engineer, are so called *meta materials*. Classical methods need to altered too in order to allow a precise and fast calculation of the properties. This leaves room for powerful quantum algorithms tackling problems resulting from the simulation of these materials.
An advantage of the quantum linear system algorithm is its logarithmic speedup in the matrix dimension when compared to classical methods. Additionally, the resource requirements for the matrix oracle are more or less independent of the given parameters, especially the mesh size. Therefore, the quantum linear system algorithm might allow for the calculation of linear systems that do not match the memory limitations of to today's supercomputers. However, the condition number of the observed matrices might not allow for this scaling. Furthermore it is unclear how to describe complex target geometries analytically.
Finally, it can be concluded that quantum computation offers a high potential to tackle computational intensive problems far beyond the capabilities of today's classical computers. But it is necessary to examine algorithms in detail to come up with promising applications and consider possible pitfalls. The acquired knowledge can then be used to correct design faults or design quantum inspired classical algorithms. The future will tell us, if quantum computation has finally emerged from laboratories to data centers.

A

APPENDIX

A.1 HERMITIZATION

The quantum linear system algorithm is only suited for Hermitian matrices since the matrix is interpreted as a Hamiltonian (see Subsection 3.2.3). If the matrix resulting from the application is not Hermitian, a modification is needed. This modification shall be called hermitization.

DEFINITION For $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ a non Hermitian matrix, we define $\boldsymbol{C} \in \mathbb{C}^{2N \times 2N}$ as

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{A} \\ \boldsymbol{A}^\dagger & \boldsymbol{0} \end{pmatrix}. \tag{A.1}$$

By construction $\boldsymbol{C}$ is Hermitian. The new right hand side is given as

$$\boldsymbol{d} = \begin{pmatrix} \boldsymbol{b} \\ 0 \end{pmatrix}. \tag{A.2}$$

The unknown $\boldsymbol{x}$ from the original equation can be extracted from the new unknown $\boldsymbol{y}$ as

$$\boldsymbol{y} = \begin{pmatrix} \tilde{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix}. \tag{A.3}$$

The resulting system of linear equations is given as

$$\boldsymbol{C}\boldsymbol{y} = \boldsymbol{d}. \tag{A.4}$$

COMPUTATIONAL COMPLEXITY An important question is the change in computational complexity caused by this hermitization. By construction the number of non zero elements per row is unchanged. Therefore only the change in the number of rows and the condition number need to be considered. Doubling the number of matrix rows adds constant overhead to the computational complexity, thus it does not change the scaling. For the condition number, the analysis is a little more elaborate. The condition number of a matrix is an important mathematical property. It is a measure for the change of the solution under perturbations of the input. A full derivation can be seen in the book "Iterative Methods for Sparse Linear Systems" by Yousef Saad [23] on page 41.

Definition: (Condition Number) Corresponding to a norm $\|.\|_X$ we can define the condition number $\kappa_X(\boldsymbol{A})$ as

$$\kappa_X(\boldsymbol{A}) = \|\boldsymbol{A}\|_X \left\|\boldsymbol{A}^{-1}\right\|_X. \tag{A.5}$$

Typically, the Euclidian (or two) and the Frobenius norm are used. For the following estimation of the condition number we want to use the Frobenius norm

$$\|\boldsymbol{A}\|_F^2 = \left\|\boldsymbol{A}^\dagger\right\|_F^2 = \sum_{k,l=1}^{n} |a_{kl}|^2, \qquad \left\|\boldsymbol{A}^{-1}\right\|_F^2 = \left\|\left(\boldsymbol{A}^\dagger\right)^{-1}\right\|_F^2 = \sum_{k,l=1}^{n} \left|\left(a^{-1}\right)_{kl}\right|^2. \tag{A.6}$$

Therefore the norm of $\boldsymbol{C}$ is given as

$$\|\boldsymbol{C}\|_F^2 = \sum_{k,l=1}^{n} \left(|a_{kl}|^2 + |\bar{a}_{kl}|^2\right) = 2\,\|\boldsymbol{A}\|_F^2\,. \tag{A.7}$$

Because of its simple form, the inverse matrix for $\boldsymbol{C}$ is given as

$$\boldsymbol{C}^{-1} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{A}^{-\dagger} \\ \boldsymbol{A}^{-1} & \boldsymbol{0} \end{pmatrix}. \tag{A.8}$$

With this we get for the condition number

$$\kappa_F(\boldsymbol{C}) = \|\boldsymbol{C}\|_F \left\|\boldsymbol{C}^{-1}\right\|_F = 4\,\|\boldsymbol{A}\|_F \left\|\boldsymbol{A}^{-1}\right\|_F = 4\,\kappa_F(\boldsymbol{A})\,. \tag{A.9}$$

Since all matrix norms are equivalent, we get the following result for the condition number in any matrix norm

$$\kappa_X(\boldsymbol{C}) \propto \kappa_X(\boldsymbol{A})\,. \tag{A.10}$$

This final result shows that the condition number is not affected by the hermitization and thus there is no change in computational complexity.

A.2 VECTOR IDENTITIES

1

$$\begin{aligned} [\boldsymbol{a} \times (\boldsymbol{\nabla} \times \boldsymbol{b})]\,\boldsymbol{c} &= \quad \epsilon_{ijk} a_j \left(\nabla \times b\right)_k c_i \\ &= \quad \epsilon_{ijk} a_j \epsilon_{klm} \left(\partial_l b_m\right) c_i \\ &= \quad \epsilon_{kij}\epsilon_{klm} a_j c_i \left(\partial_l b_m\right) = \quad (\boldsymbol{c} \times \boldsymbol{a})\,(\boldsymbol{\nabla} \times \boldsymbol{b}) \\ &= -\epsilon_{jik}\epsilon_{klm} a_j c_i \left(\partial_l b_m\right) = -\boldsymbol{a}\,[\boldsymbol{c} \times (\boldsymbol{\nabla} \times \boldsymbol{b})] \end{aligned} \tag{A.11}$$

A.3 INTEGRAL IDENTITIES

FIRST VECTOR GREEN'S THEOREM Let $\boldsymbol{a}, \boldsymbol{b}$ be vector fields, $\boldsymbol{n} \in \mathbb{R}^3$ the face normal and V a *Lipschitz continuous* domain, then

$$\int_V \boldsymbol{a}\,(\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \boldsymbol{b})\; dV = \int_V (\boldsymbol{\nabla} \times \boldsymbol{a})\,(\boldsymbol{\nabla} \times \boldsymbol{b})\; dV - \int_S (\boldsymbol{a} \times \boldsymbol{\nabla} \times \boldsymbol{b})\,\boldsymbol{n}\, dS. \tag{A.12}$$

A.4 SUBMATRIX ACTION

The explicit action of each submatrix $\boldsymbol{A}_c$ can be represented as

$$
\begin{aligned}
A_c\,|a\rangle\;|0\rangle\;|0\rangle\;|0\rangle &= \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger \mathrm{U}_c^p \mathrm{U}_c^m \mathrm{H}^{\mathrm{rw}} \mathrm{U}_c^m \mathrm{U}_c^p \mathrm{P}_{1/4\pi} \mathrm{U}_c^p\,|a\rangle\;|0\rangle\;|0\rangle\;|0\rangle \\
&= \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger \mathrm{U}_c^p \mathrm{U}_c^m \mathrm{H}^{\mathrm{rw}} \mathrm{U}_c^m \mathrm{U}_c^p \mathrm{P}_{1/4\pi}\,|a\rangle\;|\mathrm{col}_c\,(a)\rangle\;|0\rangle\;|\mathrm{arg}_c\,(a)\rangle \\
&= \exp\left(i\mathrm{arg}_c\,(a)\,/2\right) \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger \mathrm{U}_c^p \mathrm{U}_c^m \mathrm{H}^{\mathrm{rw}} \mathrm{U}_c^m \mathrm{U}_c^p\,|a\rangle\;|\mathrm{col}_c\,(a)\rangle\;|0\rangle\;|\mathrm{arg}_c\,(a)\rangle \\
&= \exp\left(i\mathrm{arg}_c\,(a)\,/2\right) \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger \mathrm{U}_c^p \mathrm{U}_c^m \mathrm{H}^{\mathrm{rw}} \mathrm{U}_c^m\,|a\rangle\;|0\rangle\;|0\rangle\;|0\rangle \\
&= \exp\left(i\mathrm{arg}_c\,(a)\,/2\right) \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger \mathrm{U}_c^p \mathrm{U}_c^m \mathrm{H}^{\mathrm{rw}}\,|a\rangle\;|\mathrm{col}_c\,(a)\rangle\;|\mathrm{amp}_c\,(a)\rangle\;|0\rangle \\
&= \mathrm{amp}_c\,(a) \exp\left(i\mathrm{arg}_c\,(a)\,/2\right) \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger \mathrm{U}_c^p \mathrm{U}_c^m\,|\mathrm{col}_c\,(a)\rangle\;|a\rangle\;|\mathrm{amp}_c\,(a)\rangle\;|0\rangle \\
&= \mathrm{amp}_c\,(a) \exp\left(i\mathrm{arg}_c\,(a)\,/2\right) \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger \mathrm{U}_c^p\,|\mathrm{col}_c\,(a)\rangle\;|0\rangle\;|0\rangle\;|0\rangle \\
&= \mathrm{amp}_c\,(a) \exp\left(i\mathrm{arg}_c\,(a)\,/2\right) \mathrm{U}_c^p \mathrm{P}_{1/4\pi}^\dagger\,|\mathrm{col}_c\,(a)\rangle\;|a\rangle\;|0\rangle\;|-\mathrm{arg}_c\,(a)\rangle \\
&= \mathrm{amp}_c\,(a) \exp\left(i\mathrm{arg}_c\,(a)\right) \mathrm{U}_c^p\,|\mathrm{col}_c\,(a)\rangle\;|a\rangle\;|0\rangle\;|-\mathrm{arg}_c\,(a)\rangle \\
&= \mathrm{amp}_c\,(a) \exp\left(i\mathrm{arg}_c\,(a)\right)\,|\mathrm{col}_c\,(a)\rangle\;|0\rangle\;|0\rangle\;|0\rangle\,.
\end{aligned}
\tag{A.13}
$$

As already discussed in Subsection 3.2.3, this equation is just of descriptive character since $\boldsymbol{A}_c$ is not unitary and thus can not be applied to qubits directly.

A.5 QUANTUM RANDOM WALK

The quantum random walk operator (see Subsection 3.2.5) is an important part of the Hamiltonian simulation subroutine (see Subsection 3.2.3). To understand the action of the exponential of this operator, we want to give the quantum circuit and discuss an example. The action of the quantum random walk operator is defined as swapping the first two registers and being diagonal in the third register, mathematically this is given as

$$
\mathrm{H}^{\mathrm{rw}}\,|a\rangle\;|b\rangle\;|y\rangle = y\,|b\rangle\;|a\rangle\;|y\rangle\,. \tag{A.14}
$$

It can be rewritten in terms of the single registers as

$$
\mathrm{H}^{\mathrm{rw}} = \left(\bigotimes_{j=1}^{n} \mathrm{S}_{j,j}\right) \otimes \sum_y y\,|y\rangle\;\langle y| \tag{A.15}
$$

with *swap* operator $\mathrm{S}_{j,l}$ that swaps the qubits j of the first and l of the second register. Often this operator is named SWAP gate and its matrix representation is given as

$$
S_{j,l} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.
$$

QUANTUM RANDOM WALK TIME EVOLUTION We are not interested in the direct application of the quantum random walk operator but in the application of its time evolution. Therefore, we need to consider the operator $\exp\left(-i\tau \mathrm{H}^{\mathrm{rw}}\right)$. To create

an efficient gate realization, we need to bring the quantum random walk operator to diagonal form. Since there is already one diagonal part, we only consider the SWAP gate $\mathrm{S}_{j,l}$. To diagonalize $\mathrm{S} = \mathrm{W}\tilde{\mathrm{S}}\mathrm{W}^{\dagger}$, we use the following unitary and self adjoint matrix

$$\mathrm{W}_{j,l} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.16}$$

The SWAP gate in the diagonal basis is given as $\tilde{\mathrm{S}}_{j,l} = \mathrm{diag}\,(1, 1, -1, 1)$. This enables us to rewrite the operator as

$$\exp\left(-i\tau \mathrm{H}^{\mathrm{rw}}\right) = \exp\left(-i\tau \left(\bigotimes_{j=1}^{n} \mathrm{W}_{j,j}\tilde{\mathrm{S}}_{j,j}\mathrm{W}_{j,j}\right) \otimes \sum_{y} y \left|y\right\rangle \left\langle y\right|\right) \tag{A.17}$$

Since the operator H^{rw} has been brought to diagonal form, it is now possible to implement the time evolution as in the following Figure A.2.

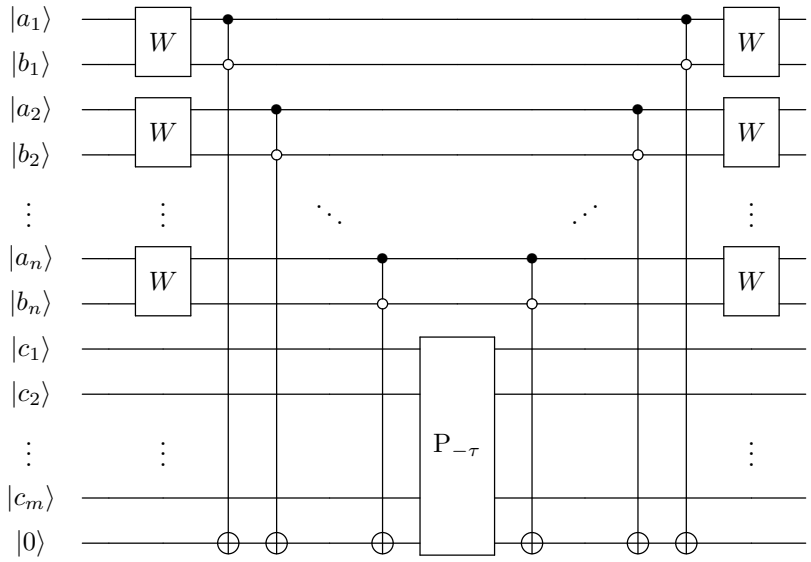

Figure A.1: Quantum random walk time evolution: The figure gives the quantum circuit to realize the quantum random walk time evolution operator on $2n + m$ qubits. Whereas the first two registers are swapped qubit wise, the ancilla qubit gets a phase shift controlled by the third register. The ancilla qubit is needed to make this calculation possible.

SMALL EXAMPLE As an example, three qubits and an additional ancilla qubit are used. For this simple case, the value controlled phase shift simplifies to a controlled rotation gate $e^{-i\tau\sigma_z}$. The quantum circuit is given in Figure A.2. Additionally, we want to write out the full calculation for the following action

$$\begin{aligned} e^{-i\tau \mathrm{H}^{\mathrm{rw}}} \left|a\right\rangle \left|b\right\rangle \left|c\right\rangle \left|0\right\rangle = &\cos\left(\tau\delta_{c,0}\right) \left|a\right\rangle \left|b\right\rangle \left|c\right\rangle \left|0\right\rangle \\ &-i \sin\left(\tau\delta_{c,0}\right) \left|b\right\rangle \left|a\right\rangle \left|c\right\rangle \left|0\right\rangle. \end{aligned} \tag{A.18}$$

To start the calculation, we need to consider the initial state, given as

$$|a\rangle\ |b\rangle\ |c\rangle\ |0\rangle = \alpha_1\beta_1\,|00\rangle\ |c\rangle\ |0\rangle + \alpha_1\beta_2\,|01\rangle\ |c\rangle\ |0\rangle + \alpha_2\beta_1\,|10\rangle\ |c\rangle\ |0\rangle + \alpha_2\beta_2\,|11\rangle\ |c\rangle\ |0\rangle\,. \tag{A.19}$$

Applying the diagonalizing gate W yields

$$\alpha_1\beta_1\,|00\rangle\ |c\rangle\ |0\rangle + \frac{\alpha_1\beta_2}{\sqrt{2}}\left(|01\rangle + |10\rangle\right)|c\rangle\ |0\rangle + \frac{\alpha_2\beta_1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right)|c\rangle\ |0\rangle + \alpha_2\beta_2\,|11\rangle\ |c\rangle\ |0\rangle\,. \tag{A.20}$$

The application of the C$\bar{\text{C}}$NOT gate will then entangle the qubits as follows

$$\alpha_1\beta_1\,|00\rangle\ |c\rangle\ |0\rangle + \frac{\alpha_1\beta_2 + \alpha_2\beta_1}{\sqrt{2}}\,|01\rangle\ |c\rangle\ |0\rangle + \frac{\alpha_1\beta_2 - \alpha_2\beta_1}{\sqrt{2}}\,|10\rangle\ |c\rangle\ |1\rangle + \alpha_2\beta_2\,|11\rangle\ |c\rangle\ |0\rangle\,. \tag{A.21}$$

Applying the off controlled rotation $\bar{\text{C}}$ROT with parameter τ, yields

$$\alpha_1\beta_1 e^{-i\tau\delta_{c,0}}\,|00\rangle\ |c\rangle\ |0\rangle + \frac{\alpha_1\beta_2 + \alpha_2\beta_1}{\sqrt{2}} e^{-i\tau\delta_{c,0}}\,|01\rangle\ |c\rangle\ |0\rangle + \alpha_2\beta_2 e^{-i\tau\delta_{c,0}}\,|11\rangle\ |c\rangle\ |0\rangle + \frac{\alpha_1\beta_2 - \alpha_2\beta_1}{\sqrt{2}} e^{+i\tau\delta_{c,0}}\,|10\rangle\ |c\rangle\ |1\rangle\,. \tag{A.22}$$

Thus, there is only a different sign for the $|10\rangle$ state of the *swap basis*. To undo the computation we need to apply the C$\bar{\text{C}}$NOT gate again, yielding

$$\alpha_1\beta_1 e^{-i\tau\delta_{c,0}}\,|00\rangle\ |c\rangle\ |0\rangle + \frac{\alpha_1\beta_2 + \alpha_2\beta_1}{\sqrt{2}} e^{-i\tau\delta_{c,0}}\,|01\rangle\ |c\rangle\ |0\rangle + \alpha_2\beta_2 e^{-i\tau\delta_{c,0}}\,|11\rangle\ |c\rangle\ |0\rangle + \frac{\alpha_1\beta_2 - \alpha_2\beta_1}{\sqrt{2}} e^{+i\tau\delta_{c,0}}\,|10\rangle\ |c\rangle\ |0\rangle\,. \tag{A.23}$$

Additionally, the inverse of the W gate needs to be applied. Since, it is self inverse this application is simply resulting in

$$\begin{aligned}&\alpha_1\beta_1 e^{-i\tau\delta_{c,0}}\,|00\rangle\ |c\rangle\ |0\rangle + \alpha_2\beta_2 e^{-i\tau\delta_{c,0}}\,|11\rangle\ |c\rangle\ |0\rangle \\ &+ \left(\alpha_1\beta_2\cos\left(\tau\delta_{c,0}\right) - \alpha_2\beta_1 i\sin\left(\tau\delta_{c,0}\right)\right)|01\rangle\ |c\rangle\ |0\rangle \\ &+ \left(\alpha_2\beta_1\cos\left(\tau\delta_{c,0}\right) - \alpha_1\beta_2 i\sin\left(\tau\delta_{c,0}\right)\right)|10\rangle\ |c\rangle\ |0\rangle\,.\end{aligned} \tag{A.24}$$

This is equal to the described action.

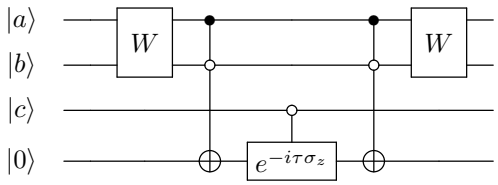

Figure A.2: Quantum random walk time evolution: The figure gives the quantum circuit to realize the quantum random walk time evolution operator on 3 qubits. Whereas the first two qubits are swapped, the ancilla qubit is rotated in dependence on the third. The ancilla qubit is needed to make this calculation possible.

BIBLIOGRAPHY

1. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum Algorithm for Solving Linear Systems of Equations. *Physical Review Letters* 103 (Oct. 2009).
2. Berry, D. W., Childs, A. M., Ostrander, A. & Wang, G. Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision. *Communications in Mathematical Physics* 356, 1057–1081 (Dec. 2017).
3. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum Support Vector Machine for Big Data Classification. *Physical Review Letters* 113 (Sept. 2014).
4. Clader, B. D., Jacobs, B. C. & Sprouse, C. R. Preconditioned Quantum Linear System Algorithm. *Physical Review Letters* 110 (June 2013).
5. Scherer, A. *et al.* Concrete Resource Analysis of the Quantum Linear System Algorithm used to compute the Electromagnetic Scattering Cross Section of a 2D Target. *Quantum Information Processing* 16 (Mar. 2017).
6. Roetteler, M., Naehrig, M., Svore, K. M. & Lauter, K. Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms. *arXiv:1706.06752 [quant-ph]* (June 2017).
7. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* 10th (Cambridge University Press, New York, NY, USA, 2011).
8. Poplavskiĭ, R. P. Thermodynamic Models of Information Processes. *Soviet Physics Uspekhi* 18, 222 (1975).
9. Feynman, R. P. Simulating Physics with Computers. *International Journal of Theoretical Physics* 21, 467–488 (June 1982).
10. Neven, H. *et al. NIPS 2009 Demonstration: Binary Classification using Hardware Implementation of Quantum Annealing* 2009.
11. Kreyszig, E. *Introductory Functional Analysis with Applications* (Wiley, 1989).
12. Peirce, C. S. *Writings of Charles S. Peirce: 1879-1884* (Indiana University Press, 1982).
13. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008,* 1–70 (Aug. 2008).
14. Shor, P. W. Polynomial Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing* 26, 1484–1509 (1997).
15. Vandersypen, L. M. K. *et al.* Experimental Realization of Shor's Quantum Factoring Algorithm using Nuclear Magnetic Resonance. *Nature* 414, 883–887 (2001).
16. Mourik, V. *et al.* Signatures of Majorana Fermions in Hybrid Superconductor Semiconductor Nanowire Devices. *Science* 336, 1003–1007 (May 2012).

17. Linke, N. M. *et al.* Experimental Comparison of Two Quantum Computing Architectures. *Proceedings of the National Academy of Sciences* 114, 3305–3310 (Mar. 2017).

18. Campbell, E. T., Terhal, B. M. & Vuillot, C. Roads Towards Fault Tolerant Universal Quantum Computation. *Nature* 549, 172–179 (Sept. 2017).

19. Peruzzo, A. *et al.* A Variational Eigenvalue Solver on a Quantum Processor. *Nature Communications* 5 (2014).

20. Farhi, E., Goldstone, J. & Gutmann, S. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]* (Nov. 2014).

21. Cook, S. A. *The Complexity of Theorem Proving Procedures* in *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (ACM, New York, NY, USA, 1971), 151–158.

22. Bernstein, E. & Vazirani, U. Quantum Complexity Theory. *SIAM Journal on Computing* (July 2006).

23. Saad, Y. *Iterative Methods for Sparse Linear Systems* 2nd (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003).

24. Berry, D. W., Ahokas, G., Cleve, R. & Sanders, B. C. Efficient Quantum Algorithm for Simulating Sparse Hamiltonians. *Communications in Mathematical Physics* 270, 359–371 (Jan. 2007).

25. Suzuki, M. Fractal Decomposition of Exponential Operators with Applications to Many-Body Theories and Monte Carlo Simulations. *Physics Letters A* 146, 319–323 (June 1990).

26. Childs, A. M. *et al.* Exponential Algorithmic Speedup by Quantum Walk. *arXiv:quant-ph/0209131,* 59 (2003).

27. Grover, L. & Rudolph, T. Creating Superpositions that Correspond to Efficiently Integrable Probability Distributions. *arXiv:quant-ph/0208112.* arXiv: quant-ph/0208112 (Aug. 2002).

28. Farebrother, R. *Linear Least Squares Computations* (Taylor & Francis, 1988).

29. Grover, L. K. A Fast Quantum Mechanical Algorithm for Database Search. *arXiv:quant-ph/9605043* (May 1996).

30. Jin, J. *The Finite Element Method in Electromagnetics* (Wiley, 2015).

31. Skolnik, M. *Introduction to Radar Systems* (McGraw-Hill, 1962).

32. Asvestas, J. The Physical Optics Method in Electromagnetic Scattering. *Journal of Mathematical Physics* 21, 290–299 (1980).

33. Gibson, W. *The Method of Moments in Electromagnetics* 2nd (Taylor & Francis, 2014).

34. Maxwell, J. C. *A Treatise on Electricity and Magnetism* (Cambridge University Press, 2010).

35. Ern, A. & Guermond, J. *Theory and Practice of Finite Elements* (Springer New York, 2004).

36. Sebelin, E. *et al.* Uniqueness and Existence Result around Lax-Milgram Lemma: Application to Electromagnetic Waves Propagation in Tokamak Plasmas. *EUR-CEA-FC. Plasma Physics and Fusion Technology (G5130)* 1609, 22 (1997).

37. Rognes, M. E., Kirby, R. C. & Logg, A. Efficient Assembly of $H(\mathrm{div})$ and $H(\mathrm{curl})$ Conforming Finite Elements. *SIAM Journal on Scientific Computing* 31, 4130–4151 (2009).

38. Alnæs, M. S., Logg, A., Mardal, K.-A., Skavhaug, O. & Langtangen, H. P. Unified Framework for Finite Element Assembly. *International Journal of Computational Science and Engineering* 4, 231–244 (2009).

39. Alnæs, M. S. *UFL: a Finite Element Form Language* (eds Logg, A., Mardal, K.-A. & Wells, G. N.) chap. 17 (Springer, 2012).

40. Alnæs, M. S. & Mardal, K.-A. *SyFi and SFC: Symbolic Finite Elements and Form Compilation* (eds Logg, A., Mardal, K.-A. & Wells, G. N.) chap. 15 (Springer, 2012).

41. Logg, A., Ølgaard, K. B., Rognes, M. & Wells, G. N. *FFC: the FEniCS Form Compiler* (eds Logg, A., Mardal, K.-A. & Wells, G.) chap. 11 (Springer, 2012).

42. Alnæs, M. S., Logg, A. & Mardal, K.-A. *UFC: a Finite Element Code Generation Interface* (eds Logg, A., Mardal, K.-A. & Wells, G. N.) chap. 16 (Springer, 2012).

43. Logg, A., Wells, G. N. & Hake, J. *DOLFIN: a C++/Python Finite Element Library* (eds Logg, A., Mardal, K.-A. & Wells, G. N.) chap. 10 (Springer, 2012).

44. Kirby, R. C. *FIAT: Numerical Construction of Finite Element Basis Functions,* (eds Logg, A., Mardal, K.-A. & Wells, G. N.) chap. 13 (Springer, 2012).

45. Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E. & Wells, G. N. Unified Form Language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software* 40 (2014).

46. Alnæs, M. S. *et al.* The FEniCS Project Version 1.5. *Archive of Numerical Software* 3 (2015).

47. Geuzaine, C. & Remacle, J.-F. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 79, 1309–1331 (2009).

48. Frank, M. P. Generalized Reversible Computing. *arXiv:1806.10183 [cs]* (June 2018).

49. Soeken, M., Roetteler, M. & Wiebe, N. *Quantum Floating Point Library* 2017.

50. Soeken, M., Roetteler, M., Wiebe, N. & De Micheli, G. Logic Synthesis for Quantum Computing. *arXiv:1706.02721 [quant-ph]* (June 2017).

51. Soeken, M., Häner, T. & Roetteler, M. Programming Quantum Computers using Design Automation. *arXiv:1803.01022 [quant-ph]* (Mar. 2018).

52. Häner, T., Soeken, M., Roetteler, M. & Svore, K. M. Quantum Circuits for Floating-Point Arithmetic. *arXiv:1807.02023 [quant-ph]* (July 2018).

53. Selinger, P. & Valiron, B. A Lambda Calculus for Quantum Computation with Classical Control. *arXiv:cs/0404056* 3461, 354–368 (2005).

54. Selinger, P. Lecture Notes on the Lambda Calculus. *arXiv:0804.3434 [cs]* (Apr. 2008).

55. Hadfield, S. Quantum Algorithms for Scientific Computing and Approximate Optimization. *arXiv:1805.03265 [quant-ph]* (May 2018).

56. DiVincenzo, D. P. Two-Bit Gates are Universal for Quantum Computation. *Physical Review A* 51, 1015–1022 (Feb. 1995).

57. Barenco, A. *et al.* Elementary Gates for Quantum Computation. *Physical Review A* 52, 3457–3467 (Nov. 1995).

58. Dawson, C. M. & Nielsen, M. A. The Solovay-Kitaev Algorithm. *arXiv:quant-ph/0505030* (May 2005).

59. Bravyi, S. & Kitaev, A. Universal Quantum Computation with ideal Clifford Gates and Noisy Ancillas. *Physical Review A* 71 (Feb. 2005).

60. Soeken, M., Frehse, S., Wille, R. & Drechsler, R. RevKit: An Open Source Toolkit for the Design of Reversible Circuits. *Lecture Notes in Computer Science* (eds De Vos, A. & Wille, R.) 64–76 (2012).

61. Bank, R. & Scott, L. On the Conditioning of Finite Element Equations with Highly Refined Meshes. *SIAM Journal on Numerical Analysis* 26, 1383–1394 (Dec. 1989).

62. Ern, A. & Guermond, J.-L. Evaluation of the Condition Number in Linear Systems arising in Finite Element Approximations. *ESAIM: Mathematical Modelling and Numerical Analysis* 40, 29–48 (2006).

63. Brenner, S. C. & Scott, R. *The Mathematical Theory of Finite Element Methods* 3rd ed. (Springer-Verlag, New York, 2008).

64. Hadfield, S. & Papageorgiou, A. Divide and Conquer Approach to Quantum Hamiltonian Simulation. en. *New Journal of Physics* 20, 043003 (Apr. 2018).

ACKNOWLEDGMENTS

My thanks goes to the department SC-HPC of the German Aerospace Center headed by Dr. Achim Basermann for the good supervision. Especially, I want to thank Dr. Tobias Stollenwerk for the help and guidance in creating this scientific report. Likewise, I want to thank Prof. Dr. David Gross for the support to write my thesis in cooperation of the University of Cologne and the German Aerospace Center and his guidance in complicated scientific questions. Finally my thanks goes to all proofreaders, especially Dr. Philipp Knechtges and Markus Zachow.

COLOPHON

This thesis was typeset by LaTeX using the typographical look-and-feel of `classicthesis` developed by André Miede.

DECLARATION

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Cologne, January 31, 2019

Christopher Zachow