UNIVERSITY OF CALIFORNIA SAN DIEGO

An introduction to quantum computing for high energy physics

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Physics

by

Davide Provasoli

2020

DEDICATION

To my parents, who have always supported me and inspired me through their hard work and dedication.

EPIGRAPH

Explore the world.

Nearly everything is really interesting

if you go into it deeply enough.

- Richard Feynman

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Christian Bauer for being a dedicated mentor over the past few years, I have learned and grown a lot while working with him. Thank you also to Benjamin Nachman, Wibe A. de Jong and the rest of our collaborators at LBNL, with whom I worked on the research which is the foundation for this thesis. Thank you to Tongyan Lin and John McGreevy for the help while writing the present thesis, your suggestions and feedback were greatly appreciated. Thank you to my family and friends for the love and support. Finally, a special thank you to Francesca, a bright light in my life.

ABSTRACT OF THE THESIS

An introduction to quantum computing for high energy physics

by

Davide Provasoli

Master of Science in Physics

University of California San Diego, 2020

Tongyan Lin, Chair

Applications of quantum computing to high energy physics (HEP) is a relatively new field of research. As such, the relevant literature is not well organized in one place and a clear road-map for people approaching the field is not currently available. Addressing this issue was the main motivation for this article, which is intended a pedagogical introduction to the field and specifically to our research direction, i.e. application of quantum computing to parton shower event generators, with the hope of more people becoming interested in exploring this path. Our paper on the subject, *"A quantum algorithm for high energy physics simulations"* [8], is the subject of chapter 4. I hope this article can provide a direct path for people familiar with quantum mechanics and quantum field theory to start reading papers or conducting research in this area

of quantum computing applications in HEP. In this regard, the quantum algorithm we present in Chapter 4, can serve as an example of development of a novel efficient quantum algorithm to solve a problem in a particle physics model.

Chapter 1

Introduction

1.1 Historical introduction

Quantum mechanics was born at the beginning of the twentieth century in order to resolve problems which didn't seem reconcilable with physical theories available at the time, i.e. what we now call classical physics. In his 1900 solution of the black-body problem, Max Planck had to assume the existence of electromagnetic oscillators with quantized energy states in order to derive his radiation spectrum formula which matched observations and avoided the so-called ultraviolet catastrophe. In 1905, inspired by Planck's work, Albert Einstein explained the photoelectric effect by postulating that the energy of light is emitted and absorbed in discrete packets, called quanta. This energy would be proportional to the frequency of the radiation, while higher intensity would correspond to a higher number of these quanta of light.

It took until the late 20's and a generation of talented physicists, among which Erwin Schrödinger, Werner Heisenberg and Max Born, for the formalism of the new theory to be developed and for quantum mechanics to be established as a novel description of the atomic world. Meanwhile, the mathematical formalism continued to be advanced through the works of Paul Dirac, David Hilbert, John von Neumann and others through the first half of the century.

Quantum mechanics, now a well tested and accepted physical theory, has rules and principles quite different from the more intuitive ones of classical physics. It offers a probabilistic view of reality instead of a deterministic one, where systems have a probability distribution which describes how likely it is to evolve into the accessible states. Any time we make a measurement, we interact with the system and in doing so we force it to ”collapse” into one of the possible states. If we prepared many identical systems and repeated the same measurement over and over,we wouldn’t obtain the same result each time and the distribution of outcomes would approach the probability distribution mentioned above. What’s interesting is that a quantum system can evolve into multiple accessible states simultaneously (we say the system is in a superposition of these states) until a measurement is made and one state is thus probabilistically selected. Once the measurement is made, further measurements will yield the same result with unit probability. The fact that a system can be in two different states at the same time, though unintuitive, is an indispensable feature of the theory and, as we shall see, one of the keys for efficient quantum computation. Furthermore, in the quantum theory quantities such as energy and momentum of bound systems can only take on discrete values, and there are fundamental limits on how accurately we can tell the value of certain physical quantities (this is known as the uncertainty principle). Quantum mechanics was later combined with special relativity to correctly model free relativistic particles, and then expanded into quantum field theory, the framework used to describe systems with a variable number of particles, which are now characterized by excitations of underlying fields. In particle physics, quantum field theory provides models for relativistic sub-atomic particles and their interactions.

Once quantum mechanics was established from a theoretical point of view, physicists quickly began to work on its applications in the laboratory, which resulted in priceless devices such as lasers and transistors. Until the 1970s however, such applications were limited to bulk samples consisting of many quantum mechanical systems (e.g many quantum particles) which as a whole possessed quantum properties (think of super-conductors for example). Since the 70’s though, physicists started to have the first successes in controlling single quantum systems and were able for instance to isolate a single atom from the surrounding environment using an ”atom trap”, to then probe different aspects of its behavior. Besides the purely scientific interest of such pursuits, control over single quantum systems proves to be indispensable to harness the power of quantum computation.

Meanwhile, with the advent of computers and computer science, scientists started using the new machines to simulate the natural world and the question quickly arose as to what is the best computer to simulate physical models. The Church-Turing thesis states that any algorithm or model of computation can be also solved efficiently with a (probabilistic) Turing machine. Therefore the Turing machine constitutes a universal computer, well suited for all computations. Of course these computations are still limited by the available number of interconnected bits and the complexity of the problem to simulate, and the struggles of simulating quantum systems which can find themselves in (and evolve into) multiple states simultaneously, became quickly apparent. Take for instance a system of N electrons, which can be in one of two spin states (spin up and spin down). There are 2^N possible initial configurations and according to quantum mechanics the system can be in a superposition of all of these states, all of which would simultaneously evolve if we interacted with the system without making measurements (for example by shining light of given frequency onto the electrons). With a classical computer the evolution of each state must be carried out separately and we can thus see how the complexity grows exponentially and quickly becomes unmanageable for realistic quantum systems which often involve a huge number of atoms or sub-atomic particles.

In the early 1980s Richard Feynman was among the first to ponder the idea of simulating physics with an entirely different kind of machine, one whose components are inherently quantum, and the closing words to his 1982 paper [1] ”Simulating Physics with Computers” have now become a must-have for papers and seminar talks in the field of quantum computation and quantum information - they read ”nature isn’t classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy".

Since the 1980s the field of quantum computing has grown significantly and several models of quantum computation have been developed, the most widely used and studied being the digital quantum circuit model, which is based on quantum bits (or qubits) and quantum logic gates. A qubit is the quantum version of the familiar bit of classical computers, meaning it can be in a 1 state or a 0 state but also in any superposition of the two, though the result of a measurement will always yield either a 1 or a 0 state. Ideally, many qubits would be interconnected, similarly to bits in regular computers, and then manipulated through quantum operations in order to carry out computations, taking advantage of the principles of superposition and entanglement (the details of how quits and quantum operations work in practice will be explored later).

From an experimental point of view qubits have been realized in many different fashions, the two most common ones to-date being trapped ions or a small superconducting circuits, both of which can be quantum mechanical systems with only two accessible states. Scientists have encountered many challenges in the physical realization of quantum computers, especially because the quantum systems used as qubits must be well isolated from the surrounding environment, in order to preserve their state and encoded information, while at the same time being easy to manipulate with precision in order to perform quantum computations. Clearly, these two requirements don't easily coexist, making for a complex experimental problem, and today, state of the art quantum computers by Google, IBM and others have around 50 working qubits. Connecting the available qubits presents another technical challenge and limit to quantum computations, with qubits in current quantum machines being able to "talk" directly only to a few the available qubits.

From a more theoretical perspective, designing algorithms for quantum computers is also particularly challenging. In their book "Quantum Computing and Quantum Information" [2] (undoubtedly the most popular book in the field) Nielsen and Chuang list two main reasons for this. The first reason is that we as humans have intuition for the classical world but very little for the quantum one, which results in a tendency to think of algorithms that are well suited for classical, and not quantum, computers. The second reason is that according to the concept of universal computation, any algorithm can be simulated with a (probabilistic) Turing machine and therefore a quantum algorithm is truly interesting and applicable only if it is better than any existing classical algorithm, meaning that it must be able to carry out the same computation employing less resources. Some classical algorithms, though doable in principle on a Turing machine, might in practice end up taking an unreasonable amount of time, of the order of millions or even billions of years. We usually say these algorithms are unsolvable and have exponential complexity in the number of resources, meaning that if we linearly increased input resources (e.g. if we were trying to factor an integer we doubled or tripled the integer) the time required to conduct the computation would increase exponentially, making the problem quickly intractable. The question is whether there are efficient quantum algorithms that can carry out these same intractable computations in much shorter periods of time, making these problems solvable in practice. Pursuing this idea, in 1994 a mathematician from MIT, Peter Shor, demonstrated that two famous problems, namely finding prime factors for an integer and the so-called discrete algorithm problem, which are believed to have no efficient solution on a classical computer, could be solved efficiently on a quantum computer (where by efficiently we usually mean in a time that is polynomial and not exponential in the number of resources). A year later Lov Grover came up with a new quantum algorithm to speed up the process of searching a list for a specific entry. So we have a few examples of quantum algorithms outperforming all classical equivalents, but the list today is fairly short and lots of effort is being directed to the discovery of more of such efficient quantum algorithms. This presents a fascinating and novel research direction, these algorithms being searched for in every field where complex computation is carried out, from chemistry to business logistics, and the fact that there is no roadmap on how to do this, calls for a lot of creativity on the part of the investigators.

1.2 Motivation for this thesis

Going back to the Richard Feynman's quote we mentioned earlier, a natural application of quantum computing is the simulation of inherently quantum mechanical systems using quantum computers. An area which has seen promising progress in this direction is quantum chemistry, where researchers have started to successfully model simple molecules, such as H_2, to find their bound state energies with quantum algorithms. Another such area, which is the focus of this article and of the research I have been personally involved in, is the application of quantum computation to high energy physics (HEP). Quantum field theory is the basis of the Standard Model of particle physics, the framework that describes the behavior of subatomic particles and all fundamental forces, except for gravity. Experimental results from particle accelerators need to be compared with theoretical predictions, usually consisting of scattering cross sections computed with perturbation theory. I previously mentioned some of the difficulties of simulating quantum systems, and quantum field theories are especially challenging since the fundamental objects in these theories are fields that depend on spacetime, yielding infinitely many degrees of freedom, even in a finite volume. On top of being local, interactions between fields are nonlinear and therefore highly non-trivial to model and compute.

In classical computer simulations of quantum field theories it is common to discretize spacetime in a method known as lattice field theory, so that the system essentially becomes a many-body quantum system. The discretized system is usually simulated using classical computer, but Byrnes and Yamamoto, in their 2008 paper [3], show how the time evolution of such systems can be efficiently simulated on quantum computers. However, simulating the time evolution alone is not sufficient, since in order to compare with experimental results such as scattering cross sections, we must be able to simulate initial states, time evolution and measurements of physical observables. The first to propose an algorithm to efficiently simulate such process were Jordan, Preskill and Lee (JPL) in their 2011 paper "Quantum Computation of scattering in Scalar Quantum Field Theories" [4], arguably the most famous paper in the field of quantum computation for HEP. They present an algorithm for digital quantum computers to calculate scattering amplitudes for massive ϕ^4 scalar theories, which scales polynomially in the number of particles or the desired precision. Though their algorithm, which we'll explore in more detail in Section 5.1, represents a milestone in the field, its implementation would require tens of thousands of interconnected qubits with high fidelity (the degree to which quibits are isolated from the environment and maintain the desired state), plus the authors do not provide a clear blueprint of how to implement each step of the algorithm in terms of fundamental quantum operations that can be directly programmed onto a digital quantum computer. How to realize JLP's algorithm in practice is in itself an interesting research question which me and my collaborators briefly explored, and which we believe might lead to important results, which will be particularly relevant once quantum machines become bigger and more powerful.

Our research was also on applications of quantum computing to HEP but it focused on a different objective, namely developing quantum algorithms that, although they do not provide full simulation of a quantum field theory, can model inherently quantum features of processes in particle physics that classical algorithms are currently missing. Furthermore, we look for algorithms which would be implementable on NISQ (Noisy Intermediate-Scale Quantum Computers) era machines, quantum computers with between 50 and a few hundred qubits which are still subject to some noise and will be available in the near future (as mentioned above we recently passed the 50 qubit mark).

Computing full amplitudes for high energy scatterings using perturbation theory or lattice field theory is very complicated and even impossible in certain regimes where perturbation theory fails, so physicists have developed other tools in order to make predictions to compare with experiments, one of the most successful ones being parton shower simulations, highly used in predicting results for the Large Hadron Collider. In QCD for example, after a high energy collision, emergent particles that are color charged will radiate quarks and gluons, many of which will in turn also radiate in order to create the colorless hadrons that are measured in accordance to confinement, thus creating a jet of hadrons and other particles. Parton showers simulate this process using Monte Carlo Markov Chains based on the probabilities of each particle to be radiated. However, these probabilities come from tree level calculations and are combined using a inherently classical algorithm, so that many quantum interference effects are neglected. We proposed a quantum algorithm which is able to account for some of these interference effects, in the context of a toy filed theory model. While very far from a general simulation of a quantum field theory, we were able to run the algorithm for a small number of emissions on a current quantum computer by IBM, and we hope to soon be able to run our full algorithm on upcoming NISQ machines. In order to validate the results from running our algorithm on a quantum computer, we compared them to the results of a quantum simulation which runs the same algorithm but on a classical computer. This cannot be done for the full algorithm, since on the classical computer the complexity too great too quickly, so we perform this comparison only in a regime in which the algorithm complexity is still relatively small.

The field of quantum computing applications to HEP is still a new and developing field, which attracts predominantly researchers from HEP who have a background in quantum mechanics and quantum field theory, but who know little about quantum computing. This thesis is aimed to provide a direct path for these people to start reading papers and conducting research in the field. It provides the quantum computing basics I believe appropriate for this purpose and it then looks in detail at our research effort, namely the development of a quantum version of parton showers and event generators for NISQ era quantum computers. Me and my collaborators believe this is a promising direction to obtain quantum algorithms which allow for better predictions than classical computers but that at the same time require quantum machines that are not too far in the future. On the other hand, the analysis and presentation of our work can also be viewed simply as a complete example developing a new quantum algorithm, which outperforms all known classical equivalents, in order to tackle a given problem in this case in HEP. Finally, I will conclude the article by briefly describing talking about other important works that fall under simulating physics with quantum computers. I hope readers may find in this thesis some useful and clear information, and more importantly, some inspiration to pursue their own research direction in this new and exciting field.

Chapter 2

Quantum Computing

Anybody who is not shocked by quantum theory has not understood it.
—Niels Bohr

Note that in this chapter, as well as the rest of the article, we follow the book by Nielsen and Chuang *"Quantum Computation and Quantum Information"* [2] for notation and as the main reference on general principles in quantum computing and quantum information.

2.1 Qubits and a first look at quantum information

Digital quantum computers are based on the notion of the qubit, a quantum mechanical system which has a specific mathematical structure. In fact, even though qubits are physical systems (we already mentioned that the two most common types are realized with trapped ions or small superconducting circuits) we won't be concerned on how they are realized and treat them as mathematical objects, so that we work in the framework of a general theory of quantum computations that does not depend on physical realization. A qubit is then an object with two basis states, $|0\rangle$ and $|1\rangle$, which has general state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

where α and β are complex numbers, i.e. it can be in any superposition of the basis states. If we make a measurement on the state $|\psi\rangle$ we can get $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$, such that $\alpha^2 + \beta^2 = 1$. The qubit state can alternatively be expressed in the Bloch sphere representation:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \tag{2.2}$$

where θ and ϕ are real and define a point on the three-dimensional unit sphere. Note we dropped an overall phase which has no observable effects.

Given two classical bits, there are four possible states 00, 01, 10 and 11. So if we have two qubits we have four analogous basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$, where by $|00\rangle$ we mean the tensor product of the two single-qubit states, i.e. $|0\rangle \otimes |0\rangle$. The two-qubit system can be in a superposition of the four basis states:

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle \tag{2.3}$$

where the α's are once again complex coefficients (usually called amplitudes), which determine the probability of measuring the respective states, so that for instance the probability of getting $|00\rangle$ upon measuring the state of the two qubits is $|\alpha_{00}|^2$.

Now consider the following two-qubit state, known as a Bell State or EPR pair:

$$|\psi\rangle_{bell} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}\,. \tag{2.4}$$

What's interesting about this state is that if you make a measurement on the first qubit, you'll know right away the state of the second qubit as well, unlike for instance in the two-qubit state $|\psi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$, where if you measure the first qubit to be in a 0 or 1 state the second qubit will still have a 50/50 chance of being in a 0 or 1 state. The Bell state is the most simple and famous example of an entangled state. Mathematically, a state is entangled if it cannot be written as a tensor product of single qubit states. Correlations in entangled states are stronger than any classical equivalent and are preserved (and evolved) when we act on the system with quantum operations. As we shall see later through some examples, these quantum correlations are one of the key features that make quantum computation more powerful than classical computation in solving certain problems.

Now that we have looked at one and two qubit systems, consider a system of N qubits. The general state is a superposition of 2^N quantum basis states, specified by 2^N complex amplitudes. With 300 qubits, that is already larger than the number of atoms in the universe so that it would be impossible to encode our general state on a classical computer. What's more, if we now act on the system with quantum operations (e.g. we might shine light of specific frequency and intensity on our trapped ion qubits), we evolve all of the basis states simultaneously. Niels and Chuang compare this to Nature keeping 2^{300} hidden pieces of paper on the side on which she performs her calculations as the system evolves. You can see how this is an enormous amount of information, stored and evolved, and this characteristic is at the heart of the power of quantum computation.

There is a (partial) catch, however. All of this information is processed but not easily accessible. In fact, if we make a measurement of the general state above we will obtain only one of the 2^N possible states and we can say nothing about the state's amplitudes, except that the one for the state we measured is non-zero. In principle, to obtain all amplitudes with good accuracy we would have to prepare a huge number of identical systems and repeat the same measurement on all of them. Then the statistical distribution of the measured states would approach the distribution of the actual probabilities of the basis states in the superposition that is the final state of the system, as the number of measurements goes to infinity. From the probabilities we can then obtain the amplitudes that specify the general state of the system. You can see how unpractical this is. Even if we don't want to know the amplitudes exactly, we have to make at least of order 2^N measurements to sample all relevant states and have some meaningful statistics. But if that is the case, the number of measurements we must make, hence the complexity of the problem, grows exponentially with N, number of qubits, making the process intractable, hence defeating the purpose of a quantum algorithm.

Interesting quantum algorithms (and only a few of these known today) are those where this processed information is combined through clever quantum operations in a way that makes it or part of it (enough to solve an interesting problem) accessible through a number of measurements that grows polynomially with the input data and resources. An additional caveat is that the number of quantum operations to evolve the system and combine the information must also scale polynomially to remain in the domain of a solvable problem. You can see how these limitations and complexities add to the difficulty of searching for efficient quantum algorithms.

2.2 Quantum gates

A classical computer is composed of wires and logic gates, to which information is fed and later extracted from in the form of bits. The gates are responsible for processing the information through the necessary steps to carry out desired algorithms. An example of a logic gate for single bits is the NOT gates which flips the bit state, i.e. it takes $0 \rightarrow 1$ and $1 \rightarrow 0$.

A digital quantum computer has a similar structure, though the way the components are realized and behave is significantly different. In place of bits information is stored in qubits and processed using quantum logic gates. For example, the quantum version of the NOT gate is called the X gate and it takes $|0\rangle \rightarrow |1\rangle$ and $|1\rangle \rightarrow |0\rangle$, but it also acts linearly taking the superposition

$$\alpha|0\rangle + \beta|1\rangle \tag{2.5}$$

to

$$\alpha\,|1\rangle + \beta\,|0\rangle\ . \tag{2.6}$$

The most common representation of the two basis states of a single qubit is in terms of the two dimensional vectors

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.7}$$

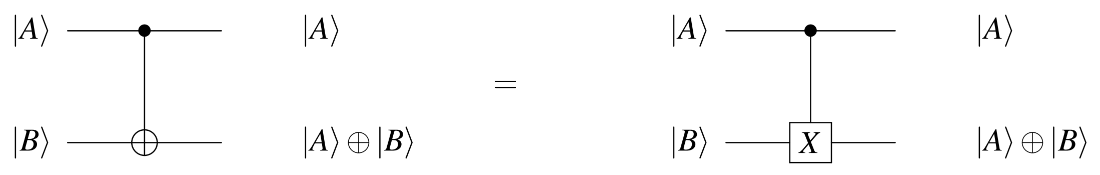
so that we can express the X gate as a two by two unitary matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.8}$$

Quantum states must be normalized such that the sum of the probabilities of obtaining all possible outcomes upon measurement be equal to one, and this normalization must be preserved after we act on the system with quantum operations. This property of quantum mechanics is knows as unitarity and it forces quantum operations to be themselves unitary, hence represented by unitary matrices.

Let's now consider multiple qubits. Common classical multi-bit logic gates are the AND, OR, XOR, NAND, NOR. The AND gate for example takes in two input bits and returns 0 if both inputs are 1 while it returns 1 otherwise. The NAND, which corresponds to an AND gate followed by a NOT gate, has the impressive property that any function of bits can be realized through NAND gates alone, making it a *universal gate*.

In quantum circuits the most common multi-qubit quantum gate is the CNOT gate, a two-qubit gate which takes as inputs a *control qubit*, which is not modified but it is only used to decide whether or not the operation is applied, and a *target qubit*, the qubit on which the operation is applied. Specifically, in the CNOT the operation applied on the target qubit is an X gate, conditional on the control qubit being in a $|1\rangle$ state. This kind of quantum operations in which a gate is applied conditional to the state of one or more qubits are called *controlled operations* and we'll describe them in more detail later. The CNOT gate has the following circuit representation:



where $\oplus$ is addition modulo two. The CNOT gate, just like any other two-qubit quantum gate, can be expressed as a four by four unit matrix. Always using the basis specified by 2.7, and recalling that for a two-qubit system the basis states are the ones appearing in 2.3, the CNOT matrix is given by

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.9}$$

While we can say that the quantum X gate is analogous to the NOT classical gate, since the two behave essentially in the same way flipping quantum or classical bits, we cannot find quantum gates that work like the NAND, XOR or the other multi-qubit classical gates, since these classical gates are irreversible, while quantum operations, being unitary, are always reversible. For example, if we read the output of an NAND gate to be 1 we cannot determine what the input bits were exactly, we can only say they were not both in a 1 state. There is therefore an irreversible loss of information, while with quantum operations we can always apply the inverse operations to bring the system back to the original state. Nonetheless, we'll see in the next section that we can simulate irreversible operations using reversible quantum gates by using some additional ”helper” qubits known as *ancilla* qubits. This can be helpful for example if we want to run classical algorithms on quantum computers (though note that this would not provide any computational advantage in general).

Since single-qubit quantum gates are represented by two by two unitary matrices, the most important quantum operations are given by the Pauli matrices:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \; ; \; Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \; ; \; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.10}$$

Exponentiating the Pauli matrices we obtain the three rotation operators, which span all rotations on the Bloch Sphere and appear in a variety of quantum algorithms:

$$R_x(\theta) = e^{-i\theta X/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

$$R_y(\theta) = e^{-i\theta Y/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.11}$$

$$R_z(\theta) = e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

These rotation operators are especially useful because it can be proved that a general unitary single-qubit operation U can always be decomposed as follows:

$$U = e^{i\alpha} R_x(\beta) R_y(\gamma) R_z(\delta) \tag{2.12}$$

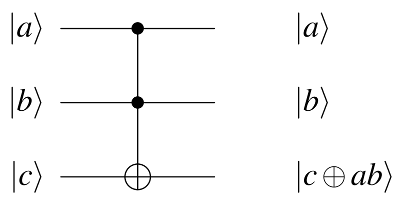
where α, β, γ and δ are real numbers. The three rotation operators can already be implemented on currently available quantum computers, through approximations, to the required precision. Other specific single-qubit quantum gates that are often used are the following:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \; ; \; S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \; ; \; T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{bmatrix} , \tag{2.13}$$

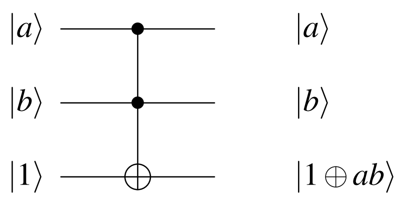
where the first one, labeled with H, is the so-called Hadamard gate, one of the most useful quantum gates, which takes the $|0\rangle$ state into $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and the $|1\rangle$ state into $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

2.3 Quantum computation

As anticipated in the previous section, it is possible to simulate even reversible gates on a quantum computer. In fact, it turns out we can simulate any classical circuit, i.e. any irreversible operation, with a quantum circuit by using a reversible quantum gate known as the *quantum Toffoli gate*. It is a three-qubit gate with the following circuit representation:



It essentially flips the c qubit if both the a and b qubits are in a 1 state, while it leaves all qubits unaffected otherwise. We can use the Toffoli gate to simulate the behavior of the NAND gate by setting the c input qubit to 1 as follows:



The a and b qubits represent the input to the NAND while the third qubit, the *ancilla qubit*, carries the output of the NAND gate. The a and b qubits can then be measured out and neglected, leaving us with the output of the NAND gate in qubit c. Then, since all classical circuits can be constructed using NAND gates only, we can always simulate a classical algorithm with a quantum circuit in principle, even though it might be quite unpractical. Though this is an important result, in general the power of quantum computation lies in algorithms which are essentially quantum in nature and not in classical algorithms running on quantum computers, which often tuns out to be quite inefficient.

Similarly to the universality of the NAND gate in classical computation, there is a *quantum universality result* which states that any unitary operation, hence any operation on a quantum computer, can be approximated to arbitrary accuracy using only H, S, T and CNOT gates (we will refer to these gates as elementary, fundamental or universal quantum gates from now on). We will not go into the details of the proof of this universality result, which can be found in Chapter 4 of [2], but simply outline the three main results upon which the proof is built. The first result is that an arbitrary NxN unitary matrix can always be written as the product of NxN two-level unitary matrices, i.e. matrices which only affect two or less vector components. To give you an example of how these matrices look like,

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{bmatrix} \tag{2.14}$$

is a 3x3 two-level unitary matrix. The second result is that any quantum operation represented by a NxN two-level unitary matrix can be decomposed and carried out using only single qubit gates and CNOT gates (this result is proved using so called *gray codes*). The third and last result is that any single qubit gate can be approximated to arbitrary accuracy using H and T gates only.

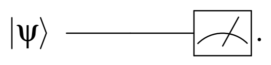
This seems an appropriate place to define what it means for a unitary operation U to be approximated by another unitary operation V. We start by defining the error between the two as

$$E(U,V) \equiv \| (U-V)|\psi\rangle \|_{max} \tag{2.15}$$

where the maximum is over all of the possible normalized states in the state space. It can be shown that if $E(U,V)$ is small then we obtain statistics of equal relevance if we make measurements on $U|\psi\rangle$ or $V|\psi\rangle$. This process of decomposition of an arbitrary unitary operation, can be subjected to error-correcting code and that ensures no information is lost in this process. We then say the process is fault tolerant. This error-correcting procedure is described in Chapter 10 of [2], and it employs phase (S) gates, explaining why the S appears in the list of universal gates.

In terms of efficiency the *Solovay-Kitaev* theorem proves that an arbitrary single qubit operation can be approximated to accuracy ε using $O(log^c(1/\varepsilon))$ gates from the universal set, where c is a constant close to 2. So we can approximate a circuit with m CNOT and single qubit gates using on the order of $O(mlog^c(m/\varepsilon))$ elementary gates, an increase in complexity which is virtually always acceptable. On the other hand however, when it comes to build an arbitrary unitary operation on *n* qubits out of the elementary gates, this cannot always be done with a quantum circuit with polynomial complexity, but it often requires a circuit with exponential complexity, meaning the number of elementary gates or the number of additional qubits necessary grows exponentially with n.

Finally, another important component in quantum circuits are measurements, which are labeled in quantum circuits with the meter symbol:



Measurements can be performed in the computational basis, i.e. the basis in which each qubit has $|0\rangle$ and $|1\rangle$ as basis states, but they can also be performed in any other basis.

In real quantum computers qubits are not perfectly isolated, especially since they must be manipulated in order to perform quantum computation, and as a consequence a certain amount of information is lost during a quantum computation. This phenomenon is known as quantum decoherence. If qubits were to be perfectly isolated they would preserve their state and the system as a whole would maintain coherence. Another similar concept, known as fidelity, defines how accurately a quantum computer is in actually implementing the quantum operations specified in the code, where a fidelity of %100 would mean perfect implementation. Of course perfect coherence and fidelity is not achievable but getting very close to it is of vital importance in order to perform long and complex quantum computations with reliable results. It is the objective of extensive current engineering research to improve on these two fronts, while on the software and algorithmic side a branch developed which studies *quantum error correction*, i.e. computational procedures which drastically improve the fidelity on a given hardware. Error correction in quantum computation is an interesting research field in its own, and one of extreme importance given that current hardware is far from being fault-proof and is subject to frequent losses of information. Error-correcting algorithms use clever techniques to avoid information loss and drastically reduce the occurrences of quantum gate malfunctions, thus allowing quantum algorithms that would otherwise be useless, to produce the desired outputs and results with negligible error.

2.4 Quantum teleportation

Let's now look at an example of a fairly simple but interesting quantum algorithm, known as *quantum teleportation*, to get a better feel for how quantum gates and quantum computation work. Suppose two friends named Alice and Bob create the EPR qubit pair

$$|\phi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.16}$$

and then move far away from one another, each carrying one of the qubits. A long time goes by and one day Alice has a new qubit in a general state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, without knowing what α and β are, that she has to send to Bob, but she must do so by only sending classical information to him. Ideally, she could find out what α and β are and communicate the two complex numbers to Bob, so that he can then recreate the state $|\psi\rangle$ on a qubit he controls. The problem with this is that Alice cannot determine α and β from a single copy of $|\psi\rangle$ and what's more, even if she new the two complex amplitudes, it would require in general an infinite amount of classical information to communicate them to Bob since they take value in a continuous space. We will now show that there is simple quantum algorithm that Alice can employ to send $|\psi\rangle$ to Bob, by using her old qubit from the EPR pair and the ability of sending classical information. The quantum circuit is shown is Figure 2.1, where $|phi\rangle$ is the old EPR pair and the last two controlled operations apply the X and Z gates conditional on the measurements of the control qubits yielding a $|1\rangle$. Note that in general, unless specified otherwise, the input qubits to a quantum circuit are all assumed to be in the $|0\rangle$ state.

Let's see how the algorithm works. It involves both the new qubit as well as the old EPR pair, so the input state is

$$|\psi_0\rangle = |\psi\rangle\,|\phi\rangle = \frac{1}{\sqrt{2}}(\alpha\,|1\rangle\,(|00\rangle + |11\rangle) + \beta\,|0\rangle\,(|00\rangle + |11\rangle))\,, \tag{2.17}$$

where the third qubit is the one Bob has.

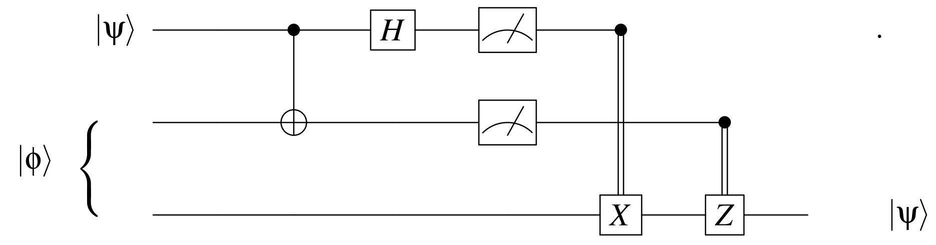


Figure 2.1: quantum circuit for quantum teleportation algorithm with measurements at the end of the circuit.

Alice applies a CNOT gate to her two qubits, conditional to the $|\psi\rangle$ qubit, obtaining

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(\alpha\,|0\rangle\,(|00\rangle + |11\rangle) + \beta\,|1\rangle\,(|10\rangle + |01\rangle)) \tag{2.18}$$

and then she applies a Hadamard gate to the first qubit, obtaining

$$|\psi_2\rangle = \frac{1}{2}(\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)) \tag{2.19}$$

This can be rewritten as the sum of the following four terms:

$$|\psi_2\rangle = \frac{1}{2}[|00\rangle\,(\alpha\,|0\rangle + \beta\,|1\rangle) + |01\rangle\,(\alpha\,|1\rangle + \beta\,|0\rangle) + |10\rangle\,(\alpha\,|0\rangle - \beta\,|1\rangle)) + |11\rangle\,(\alpha\,|1\rangle - \beta\,|0\rangle)] \tag{2.20}$$

At this point, if Alice measures her two qubits and communicates the results to Bob through classical information, Bob will know in which state his qubit is in, out of the four possible ones in $|\psi_2\rangle$. For example, if Alice measures 00, Bob's qubit will already be in the desired state

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. On the other hand, if Alice measures one of the other three states, after she communicates the result to Bob, he can fix the state of his qubit to recover $|\psi\rangle$ using the conditional X and Z gates we see at the end of the circuit. For instance, if Alice measures 11 and tells Bob, he can apply an X gate followed by a Z gate to his qubit to obtain $|\psi\rangle$.

As a quick aside, an important feature of measurements in quantum circuits is that they can always be moved from intermediate stages to the end of the quantum circuit. In order to do so, all we have to do is replace the quantum operations conditional on measurement results (i.e. conditional on classical information) with the same operations conditional on the un-measured qubits. Applying this procedure, the quantum teleportation circuit is modified as shown in Figure 2.2 Unlike other quantum operations, measurements are in general irreversible and not unitary,

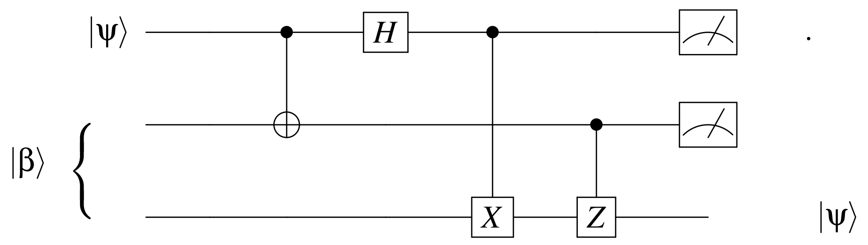


Figure 2.2: quantum circuit for quantum teleportation algorithm with measurements at the end of the circuit.

they destroy quantum information and transform it into classical information. A more precise statement, is that measurements are irreversible except when they do not reveal any information on the state being measured.

2.5 Controlled operations

An operation such as the CNOT gate, where a quantum gate is applied conditional on the state of one or more qubits, called-called *control qubits*, is called a *controlled operation*. In general, controlled operations can have multiple control qubits and the gate which is conditionally applied can be any unitary operation acting on *n* qubits. We will now show how such an arbitrary controlled quantum operation can be decomposed into elementary gates, or at least basic operations such as the Toffoli gate, which can be directly implemented in currently available quantum simulators. These results will be especially important later when we discuss in detail how to decompose the operations in the circuit for our quantum shower algorithm, in order to implement it and simulate it on a current quantum computer. In this section I will state and use certain results without proving them. All proofs can be found in Chapter 4 of [2].

Let's start by considering an arbitrary single-qubit gate U controlled by a single qubit, often referred to as a $C(U)$ operation. This can be decomposed as shown in Figure 2.3,

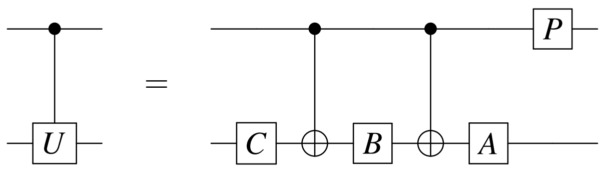


Figure 2.3: Quantum circuit for the decomposition of an arbitrary single-qubit operation controlled on 1 qubit.

where

$$P = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\psi} \end{pmatrix} \tag{2.21}$$

and the following conditions must be satisfied:

$$U = \exp(i\psi)AXBXC \; ; \; ABC = I \tag{2.22}$$

Note that it is always possible to find such A, B, C and ψ. Then any $C(U)$ operation can be realized using solely single qubit and CNOT gates. We can of course also control on a qubit being in the 0 state instead of the 1 state, which is implemented as shown in 2.4.

A single-qubit quantum gate controlled on two qubits is decomposed as shown in Figure 2.5, where V is any unitary gate such that $V^2 = U$. If we define $V \equiv (1-i)(I+iX)/2$, which has

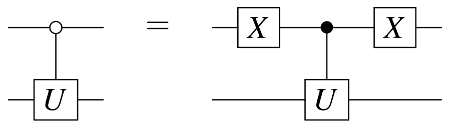


Figure 2.4: controlling a single-qubit operation U on the control qubit being in a 0 state instead of a 1 state

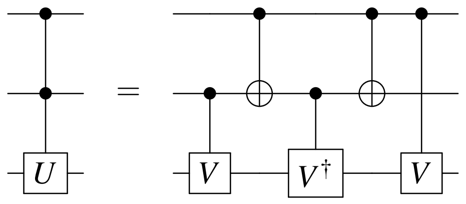


Figure 2.5: quantum circuit for the decomposition of an arbitrary single-qubit operation controlled on 2 qubits.

the property that $V^2 = X$, the above gives a decomposition of the Toffoli gate in terms of CNOTs and single-qubit gates controlled on a one qubit, which can be further decomposed as shown in Figure 2.3.

Finally, let's consider a single qubit gate controlled on n qubits, whose decomposition is shown in Figure 2.6. To decompose such operation we must use $n-1$ work qubits, i.e. additional qubits, initialized in the $|0\rangle$ state, which are only used to carry out intermediate operations and can be then measured out and discarded or reset to the initial state. The first n qubits are the control qubits and here the U operation is applied conditional on these qubits being in a $|1\rangle$ state. We can control of course on any of these qubits being in a 0 state by adding an X gate at the beginning and at the end of the quantum circuit, analogously to what we showed in Figure 2.4. The next $n-1$ qubits are the work qubits we previously mentioned, which start out in the $|0\rangle$ state. By the symmetry of the operations applied, and the fact that the Toffoli gate is its own inverse, we see that both the control and the work qubits will be, at the end of the ciruit in Figure 2.6, in the same state as they started out in. Lastly, the target qubit is the one on which the single-qubit operation U is applied.

We have thus shown how to decompose into elementary gates any controlled operation in

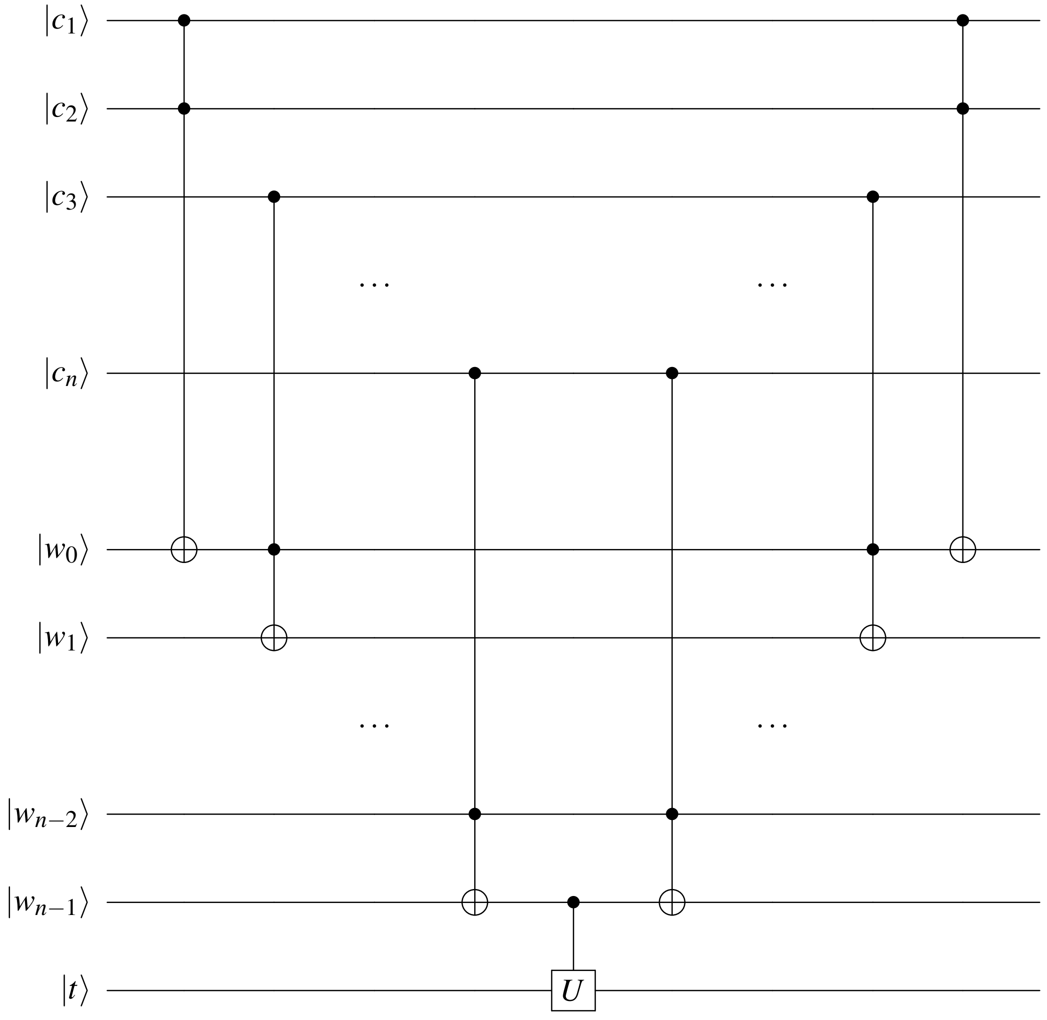


Figure 2.6: quantum circuit for the decomposition of an arbitrary single-qubit operation controlled on n qubits.

which the applied operation is a single-qubit quantum gate. We have previously discussed how to decompose an arbitrary quantum operation U_n on n qubits into elementary gates. Then, if we want to apply such a general U_n gate controlled on 1, 2 or n qubits, we can do this by applying the same controls to all of the elementary gates that make up the decomposition of U_n. Therefore, we now have all the tools to decompose and implement any controlled operation on a real quantum computer.

2.6 Computational complexity

As previously mentioned, a quantum algorithm is truly interesting if it outperforms all known classical algorithms to solve a given problem. It's then important to determine the *computational complexity* of an algorithm, specifically the lower bounds on the amount of resources necessary run the algorithm and how these resources scale. The most common resources we look at in order to determine the complexity of a problem are *time* and *space*. In practical terms, for a classical algorithm these resources correspond to how long it takes for an algorithm to run and how many bits and gates it requires to be implemented. Similarly, for a quantum algorithms they are given by the time it takes the algorithm to run plus how many qubits and quantum gates are required.

Suppose the input to a certain algorithm can be expressed with an n bits or qubits. Usually, the main distinction one makes when studying the complexity of a problem is whether or not the resources necessary to solve the problem grow faster than any polynomial in n. If the resources do grow faster than any polynomial we say that the problem has *exponential complexity* or that the required resources scale exponentially in n. This is slightly abusing the term exponential, since there are algorithms that scale as $n^{\log n}$ for example, which is not exponential though larger than any polynomial in n. Furthermore, certain exponential algorithms might be faster than some polynomial algorithms, which might for instance scale as n^{1000}, for all practical purposes. However, most known polynomial algorithms scale as n or n^2, and rarely have a large degree. So provided these specifications, the distinction between polynomial and exponential algorithm is very useful and widely used when describing computational complexity. A problem that has a know algorithmic solution which uses resources that scale polynomially is called easy, tractable or solvable, while a problem for which the best known algorithmic solution requires exponential resources is known as hard, intractable or unsolvable.

Much of the motivation for using this distinction between polynomial and non-polynomial complexity comes from the strong Church-Turing thesis, which states:

Any model of computation can be simulated on a probabilistic Turing machine with at most a polynomial increase in the number of elementary operations required.

Then if a problem does not have any solution which requires polynomial resources on a probabilistic Turing machine, it doesn't have an efficient solution on any (classical) computing device.

The main caveat in this discussion on complexity is that while it is possible to prove that the majority of problems will have exponential complexity, in general, it is very difficult (and seldom accomplished) to prove that no efficient polynomial algorithm exists to solve a specific problem. We then tend to say a problem is intractable if it is conjectured that the best algorithm to solve it requires exponential resources. This conjecturing often means that extensive research efforts have yielded algorithms for solving the problem, all of which require exponential resources.

2.7 The Quantum Fourier Transform

The most incredible results in quantum computation, and much of the promises from this field, come from the few known quantum algorithms which can efficiently perform tasks that are intractable on a classical computer. Perhaps the most famous example is the factorization of an integer into prime numbers, a task of vital importance in mathematics, cyber security and other areas. If we express the integer in terms of n bits (or qubits), the best known classical algorithm to solve the factorization problem requires approximately $exp(O(n^{1/3}\log^{2/3} n))$ operations. On the other hand, there is a known quantum algorithm which can perform the same task with $O(n^2 \log n \log\log n)$ elementary quantum operations. The key ingredient of this quantum factoring algorithm, responsible for the speedup over the classical algorithm, is the *quantum Fourier transform* (QFT).

The *discrete Fourier transform* takes a complex vector of length N and components $x_0, \dots, x_{N-1}$, and it outputs a complex vector of same length and components $y_0, \dots, y_{N-1}$ defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i jk/N} \tag{2.23}$$

The quantum Fourier transform performs an equivalent transformation but on quantum states. The QFT on an orthonormal basis set $|0\rangle, \ldots, |N-1\rangle$ is a linear unitary operator which transforms each basis state as

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i jk/N} |k\rangle \tag{2.24}$$

and a general state as

$$\sum_{j=0}^{N-1} x_j |j\rangle = \sum_{k=0}^{N-1} y_k |k\rangle \,, \tag{2.25}$$

where the amplitudes y_k are given by the discrete Fourier transform in Qe. 2.23.

The QFT quantum algorithm takes as input the binary representation $j = j_1 j_2 \ldots j_n$ encoded into n qubits ($|0\rangle$ for 0 bits and $|1\rangle$ for 1 bits) and transforms it as follows:

$$|j_1, \ldots, j_n\rangle \rightarrow \frac{(|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n} |1\rangle) \ldots (|0\rangle + e^{2\pi i\, 0.j_1 j_2 \ldots j_n} |1\rangle)}{2^{n/2}}, \tag{2.26}$$

where $0.j_l j_{l+1} \ldots j_m$ represents the *binary fraction* $j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$. The transformation in eq. (2.26) is equivalent to the one in eq. (2.24), and the proof can be found, along with the full quantum circuit which implements eq. (2.26), in Chapter 5 of [2].

The quantum circuit for the QFT is implemented using $O(n^2)$ elementary quantum gates. If we have n qubits as input, the general initial state will be a superposition of 2^n orthonormal basis states, so that the QFT can Fourier transform 2^n amplitudes using $O(n^2)$ operations. On the other hand, the best know classical algorithms which implement discrete Fourier transform, such as the *Fast Fourier Transform*, require of the order $O(n2^n)$ gates to Fourier transform 2^n amplitudes. This sounds great in principle, especially since Fourier transforms have such a wide range of applications, except that the transformed amplitudes of the QFT are not directly accessible. If we make a measurement on the system after applying the QFT we would simply obtain one state out of the possible 2^n, and not even the respective amplitude. In order to extract the amplitudes with statistical relevance, we would need to perform at least of the order 2^n runs and measurements, hence loosing all the advantage over the classical algorithms.

The key to harness the power of the quantum Fourier transform, is to incorporate it in a clever way into a bigger algorithm, which is able to convert the information processed by the QFT into an accessible form. We will illustrate how this is done in the most important example, known as the *phase estimation* algorithm, which is a key component of the factoring algorithms. Suppose we have a unitary operator U with eigenvector $|u\rangle$ and eigenvalue $e^{2\pi i\varphi}$, then phase estimation allows us to estimate the unknown phase φ, hence determining the eigenvalue of U with computable accuracy. In order to implement phase estimation using a quantum circuit, we must be able to efficiently prepare the $|u\rangle$ state and to apply U^{2^j} operations for some suitable j. Suppose these requirements are satisfied. Then we introduce a second register, beside the one to encode $|u\rangle$, with t qubits in the $|0\rangle$ state, where t determines the accuracy of the estimation of φ and the probability of success of the phase estimation procedure. The algorithm is divided into two main parts and can be found in detail in Chapter 5 of [2]. It starts by transforming the t-register into the state

$$\begin{aligned}\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi i\, 2^{t-1}\varphi}|1\rangle)(|0\rangle + e^{2\pi i\, 2^{t-2}\varphi}|1\rangle)\ldots(|0\rangle + e^{2\pi i\, 2^{0}\varphi}) \qquad (2.27)\\ = \frac{1}{\sqrt{2^{t/2}}}\sum_{k=0}^{2^t-1} e^{2\pi i\varphi k}|k\rangle\,.\end{aligned}$$

Now suppose that φ can be expressed exactly with t bits as $0.\varphi_1\ldots\varphi_t$, then the above state (2.27) becomes

$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi i\, 0.\varphi_t}|1\rangle)(|0\rangle + e^{2\pi i\, 0.\varphi_{t-1}\varphi_t}|1\rangle)\ldots(|0\rangle + e^{2\pi i\, 0.\varphi_1\varphi_2\ldots\varphi_t}|1\rangle) \qquad (2.28)$$

.

If we compare this with the expression in (2.26), we see that they have exactly the same form, so that if we now apply the inverse of the quantum Fourier algorithm (which is invertible being a unitary operation) to the t register we obtain $|\varphi_1 \ldots \varphi_t\rangle$. Then a measurement of the t qubits would yield φ exactly! Now of course in general we cannot express φ with t qubits exactly. So, let b be the best t-bit approximation of φ which is smaller than φ and such that the difference $\delta \equiv \varphi - b/2^t$ satisfies $0 \leq \delta \leq 2^{-t}$. Then the phase estimation procedure yields, in place of the exact φ, a result φ' which is close to b and hence a good approximation of φ.

Incorporating phase estimation, clever applications of the QFT or other known quantum computing routines which are more efficient than classical counterparts, into larger algorithms, is certainly a way of finding new problems which can be solved more efficiently on a quantum computer than on a classical one.

Chapter 3

Parton Showers

In this chapter we briefly describe the physics behind *parton showers* and how they are implemented as Monte Carlo event generators. Hadron colliders have been extensively used as machines for discovering new particles and probing the workings of the fundamental forces at very high energies, the reason being that it is much easier to accelerate hadrons to very high energies as compared to other particles such as leptons. On the theoretical side, hadron collisions are difficult to model due to the large number of particles involved and the complexities of the strong interaction, so that a first-principle approach to compute final states in such processes is currently unfeasible. Meanwhile, Monte Carlo Markov Chain event generators known as parton showers simulate the evolution of partons in hadronic collisions in the collinear limit and have been very successful in computing predictions for high energy scattering experiments, such as the ones performed at the Large Hadron Collider (LHC) in Geneva. This discussion will be useful background for the following chapter, in which we describe a quantum algorithm for parton showers in a simplified field theory model. A complete description of parton shower event generators can be found in [5], [6] and [7].

3.1 QCD and the parton model

Quantum Chromodynamics (QCD) is the quantum field theory of the strong interaction, which acts between quarks and gluons, the fundamental particles that combine to form Hadrons such as the proton, the neutron and the pion. QCD is a very rich physical theory with peculiar properties. One of these properties, known as asymptotic freedom, says that the interaction between particles under the strong force becomes weaker and weaker as the energy becomes larger. Mathematically, this is explained through the renormalization group. The leading term in the beta function expansion in QCD (and in any non-abelian gauge theory in four dimension) comes with a minus sign, meaning the coupling constant becomes weaker as the energy of the system becomes larger. Historically, this was first observed in proton-proton collisions at energies above 10 GeV, where a large number of pions were produces but, instead of filling all of phase space as many people expected, these pions were produced with momenta which were almost collinear to the collision axis. It was then hypothesized that Hadrons were loose clouds of unknown elementary particles that, inn a high energy collisions, could exchange momentum q only in a way such that q^2 would remain small in the end. This theory was put to test in the 1960s in the SLAC-MIT deep inelastic scattering experiments, where a 20 GeV beam of electrons was scattered by an hydrogen target and the scattering rates for large deflection angles were measured. This probed scattering which corresponded to an electron transfering a large total momentum q^2 to one proton in the target. If the proton was a loosely bound ball of component particles as hypothesized above, low scattering rates were expected. However, a large rate of inelastic scattering was measured, as though the proton was an elementary particle scattering with the electron according to the rules of QED. What's more, the largest part of particles which made up this surprisingly high scattering rate, were not single protons, but a large number of hadrons, as if the electron shattered the proton, through a QED interaction, into multiple hadrons.

In order to reconcile the presence of QED hard scattering with the (almost) absence of hard scattering through the strong interaction, Feynman and Bjorken proposed the *parton model*: the proton is made up of constituents called partons, which include fermions carrying electric charge, known as quarks and antiquarks, and possibly other neutral particles responsible for binding the quarks together. These neutral species were then identified to be spin-0 bosons known as gluons. By assumption, partons cannot exchange large invariant momentum though the strong interaction, but the quarks can interact electromagnetically so that an electron can knock a quark out of the proton. Furthermore, following evidence for asymptotic freedom, in the parton model, at very high energies the strong interaction goes to zero. In reality, mathematically the interaction between partons at high energies decreases as the logarithm of the energy scale, meaning that the parton model, though incredibly useful, it remains a simplified model of QCD at high energies.

Going back to the electron that knocked a quark out of the proton, at this stage another fundamental property of QCD comes in, known as *confinement*, which is responsible for the fact that a variety of hadrons, instead of only protons, were found as final products in the scattering experiments described above. Confinement says that particles with color charge, such as gluons and quarks, cannot exist in isolation, and cannot be measured independently, at low energy, but always form colorless objects (hadrons). Then the quark which was ejected from the proton by the scattering with an electron interacts softly with the rest of the proton, producing eventually a jet of hadrons, collinear to the original struck quark. Though no analytic proof of confinement exists, it is understood as follows: the energy to separate gluons (which also carry color charge) and quarks grows as the distance between the particles becomes larger, to the point that it requires less energy to produce quark-antiquark pairs than to separate the particles further. Thus, a shower of spontaneously produced quarks, antiquarks and gluons, which clump together to form colorless Hadrons, is produced and measured. The momenta of the particles in the shower are very close to being collinear to the parton which initializes the shower.

3.2 Hadron collisions and Monte Carlo Markov Chains

Monte Carlo event generators use parton showers to successfully simulate both final state evolution of partons, which corresponds to the showering which takes place after a hard process and before final particle states are measured, and initial state evolution, meaning the radiation of partons on their way to a hard scattering, the two processes being very similar. Let's look at initial state radiation and consider a reaction with a high transfer of invariant momentum in which two hadrons, which we call h_1 and h_2, collide and produce the final state X. In the collinear limit the final cross section (i.e. the probability of state X to be produced) factorizes in the following convolution equation:

$$\sigma_{h_1 h_2 \to X} = \sum_{a,b \in \{q,g\}} \int dx_a \int dx_b \; f_a^{h_1}(x_a, \mu_F^2) \; f_b^{h_2}(x_b, \mu_F^2) \; \int d\Phi_{ab\to X} \frac{d\hat{\sigma}_{ab}(\Phi_{ab\to X}, \mu_F^2)}{d\Phi_{ab\to X}} . \quad (3.1)$$

The function $f_a^h(x_a, \mu_F^2)$ is known as the Parton Distribution Function (PDF) and it represents the probability, at leading perturbative order in QCD, of finding parton a in the parent hadron h, at scale μ_F and with momentum fraction x_a. μ_F is the factorization scale which provides the collinear cutoff and is introduced to regulate IR divergences coming from the emission of soft gluons. $d\sigma_{ab}/d\Phi$ is the differential cross section for the production of final state X from the parton state ab, which can be computed with Feynman diagrams in perturbation theory, and $d\phi_{ab\to X}$ is the respective differential element of phase space.

Now if we do not specify the kinematics of X and any additional particles that can be produced alongside, we refer to eq. (3.1) as the inclusive cross-section for the production of X. Exclusive cross-sections can be obtained by specifying the kinematics of X or the additional particles produced or both. Event generators do the following: starting with eq. (3.1), an inclusive final state consisting of only X is produced. Then, through a Monte Carlo Markov Chain additional particles are produced respecting conservation of probability and four-momentum, to generate a high multiplicity final particle state.

In the collinear limit the evolution of the PDFs is determined by the so-called DGLAP equations:

$$\mu_F^2 \frac{d f_a(x,\mu_F^2)}{d\mu_F^2} = \sum_{b\in\{q,g\}} \int_x^1 \frac{dz}{z} \frac{\alpha_s}{2\pi} P_{ab}(z) f_b(x/z,\mu_F^2)\,, \tag{3.2}$$

where the $P_{ba}(z)$ functions are the Alterelli-Parisi splitting functions, which give the probability of a specific emission in which the emitted particle carries a fraction z of the momentum of the emitting parton. The splitting functions are given by:

$$\begin{aligned}
P_{qq}(z) &= C_F \frac{1+z^2}{(1-z)} \\
P_{gq}(z) &= C_F \frac{1+(1-z)^2}{z} \\
P_{qg}(z) &= T_R z^2 + (1-z)^2 \\
P_{gg}(z) &= C_A \frac{z^4+1+(1-z)^4}{z(1-z)}\,.
\end{aligned} \tag{3.3}$$

The DGLAP equation (3.2) says the following: a parton a which was resolved in the parent hadron at scale μ_F^2 may have been emitted, alongside some other parton, by parton b resolved at the scale $\mu_F^2 + d\mu_F^2$. The transition from parton b to parton a is a Markov process, where we account for the additional emitted parton as part of the final particle state. Then repeated implementation of eq. (3.2), keeping track of all emitted partons, builds the shower evolution.

A Monte Carlo event generator is realized by converting the inclusive prediction of finding parton a in the beam hadron h to an exclusive prediction of parton a to be produced alongside many other particles. The splitting functions provide the relative probability of a certain emission to happen, so that at each splitting a certain branching is chosen through a random number generator where all options are weighted according to the correct splitting probabilities.

At this stage we would run into divergences in integrating the splitting functions over z in eq. (3.2). However, at low enough energies partons cannot exist in isolation but must form color-neutral hadrons due to confinement. This implies that partons which have close transverse momentum, say of the order of 1 GeV or less, cannot be resolved. This provides a natural infrared momentum cutoff for the parton shower, which results in an upper bound for the momentum fraction z in eq. (3.2), thus regulating the IR divergences. At this stage, we have removed all higher-order real corrections and all virtual corrections. These can be accounted for by adding an additional term to the DGLAP equation:

$$\frac{df_a(x,t)}{d\log t} = \sum_{b\in\{q,g\}} \int_x^{z_{max}} \frac{dz}{z}\frac{\alpha_s}{2\pi} P_{ba}(z) f_b(x/z,t) - f_a(x,t) \sum_{b\in\{q,g\}} \int_{z_{min}}^{z_{max}} dz \frac{\alpha_s}{2\pi} P_{ab}(z), \tag{3.4}$$

where we now identify the factorization scale with the more general t, instead of μ_F, which can be associated with different variables, such as the emission angle θ. Let's now introduce the Sudakov factor

$$\Delta_a(t,t') = exp\left(- \sum_{b\in\{q,g\}} \int_t^{t'} \frac{d\tilde{t}}{\tilde{t}} \int_{z_{min}}^{z_{max}} dz \frac{\alpha_s}{2\pi}\frac{1}{2} P_{ab}(z) \right), \tag{3.5}$$

which corresponds to the probability of a parton a not to undergo any branching between the scales t and t'. We can then rewrite eq. (3.4) in terms of the Sudakov factor as follows

$$\frac{d}{d\log t} \log \frac{f_a(x,t)}{\Delta_a(t_c,t)} = \sum_{b\in\{q,g\}} \int_{z_{min}}^{z_{max}} \frac{dz}{z}\frac{\alpha_s}{2\pi} P_{ba}(z) \frac{f_b(x/z,t)}{f_a(x,t)}, \tag{3.6}$$

where t_c corresponds to the IR cutoff discussed above. Note Eq. (3.6) describes the evolution of the PDFs. Final state evolution works very similarly, except now we start from a high evolution scale and evolve down to the cutoff scale t_c. Also in final state evolution we sum over the emitted partons instead of the emitting ones, and PDFs are replaced by Fragmentation Functions (FF). Since final states will consist of colorless hadrons, FFs are non-perturbative functions which give the probability of a certain particle to have contributed to forming these final hadronic states.

Let's now try to get a feel for how, in practice, we implement the parton shower as a

Monte Carlo Markov Chain. Consider forward evolution as in final state evolution and suppose we start with parton a at the evolution scale t'. In order to find the scale t of the next branching we solve the equation

$$r = \Delta_a(t,t')\,, \tag{3.7}$$

where r is a randomly generated number in the range $[0,1]$. At this point, if multiple splittings are possible, we must select a specific splitting according to the correct weights. These weights are given by $P_{ab}(x_a,t)f_a(z)$ for initial state evolution, and by only $P_{ab}(x_a,t)$ for final state evolution. Once a splitting has been selected, we find the momentum fraction z by solving

$$r\int_{z_{min}}^{z_{max}} dz'\frac{\alpha_s}{2\pi}P_{ab}(z') = \int_{z_{min}}^{z} dz'\frac{\alpha_s}{2\pi}P_{ab}(z')\,. \tag{3.8}$$

The process is repeated for each emitted particle, creating a high multiplicity final state, until the cutoff scale t_c is reached. Of course, in practice the evolution variable is discretized into N evolution steps, where at each step an emission can happen or not, according to the probability given by Sudakov factors and splitting functions. It is a Markov process because the probability for a certain splitting to happen at a given scale only depends on the current particle state and dynamical variables, and not on the past evolution history or other particles.

Chapter 4

A Quantum parton shower

In our paper *'A Quantum algorithm for high energy physics simulations'* [8], we consider a toy model field theory in which fermionic and scalar fileds are radiated collinear to the initiating particle. Monte Carlo Markov Chains (MCMC) have been used very effectively to model such systems, although they cannot capture all quantum effects. In particular, our toy model presents quantum interferences between amplitudes corresponding to identical final particle states, but reached with different intermediate particle states This is similar to interferences in electroweak showers (see [9] for a detailed description of electroweak parton showers) involving intermediate photons or Z bosons. In our simplified model the kinematics consist of the variable describing the collinear shower evolution, which is discretized yielding a series of steps where at each step an emission can happen or not. We propose a quantum algorithm that simulates a parton shower keeping track of intermediate state interferences which are missed by a MCMC approach. Our algorithm samples from the full, final state probability distribution in polynomial time, while the best classical algorithm we could find requires exponential time to perform the same sampling.

4.1 A toy model

Consider a quantum field theory described by the Lagrangian

$$\begin{aligned}\mathcal{L} =& \bar{f}_1(i\partial\!\!\!/ + m_1)f_1 + \bar{f}_2(i\partial\!\!\!/ + m_2)f_2 + (\partial_\mu\phi)^2 \\ &+ g_1\bar{f}_1 f_1\phi + g_2\bar{f}_2 f_2\phi + g_{12}\left[\bar{f}_1 f_2 + \bar{f}_2 f_1\right]\phi \end{aligned} \tag{4.1}$$

where f_1 and f_2 are two fermion fields and ϕ is a scalar boson field. Partial motivation for our model comes from the fact that similar interactions of fermions with scalar fields occur in the Higgs sector of the Standard Model, where it has been shown that final state radiation at high energy can be modeled with a parton shower in the collinear limit ([9], [10]).

We will consider final state radiation in the collinear limit governed by eq. (4.1) and initiated by one or more fermions of type (or spin as we shall refer to sometimes) 1 or 2. In our toy model the scalar boson ϕ can couple to either fermion 1 with coupling constant g_1 or fermion 2 with coupling g_2, or to both fermions with coupling g_{12}. The shower dynamics consist of fermions radiating scalars in $f \to f'\phi$, where the fermion can remain of the same type or switch type, and scalar bosons splitting into fermion-antifermion pairs in $\phi \to f\bar{f}'$. At the end of the shower, the final state observable consists of a set of fermions and bosons with their energies (or angle of emission) and positions in the jet of particles.

With all coupling being non-zero, there are quantum interferences among final states due to the fact that intermediate fermion states are not observed and can be in a superposition of f_1 and f_2. We must choose a scale to order emissions in our shower and we follow the common choice of using the angle of emission θ as our dynamical variable. These angles are strongly ordered such that $\theta_0 > \theta_1 > ... > \theta_n$ all the way to a collinear cutoff $\theta = \varepsilon > 0$. Then the kinematic part of the shower amplitude, i.e. without accounting for the coupling constants, factorizes as

$$\hat{\mathcal{A}}_n(\theta_1,\ldots,\theta_n) = \hat{\mathcal{A}}(\theta_0|\theta_1)\hat{\mathcal{A}}(\theta_2|\theta_1)\ldots\hat{\mathcal{A}}(\theta_n|\theta_{n-1}), \tag{4.2}$$

where $\hat{A}(\theta_n|\theta_{n-1})$ is the amplitude to emit one particle at angle θ_n given the angle of the previous emission. In general, the full amplitude $A\hat{\mathcal{A}}$ also depends on the momentum fraction of the emitted particles, but since in this case the momentum dependence is completely factorized from the angular dependence, we decide to ignore it for the sake of simplicity. Realizing a more complete quantum simulation of our model which also includes momentum dependence is a possible next step to be explored in the future.

4.2 MCMC simulation in the limit with $g_{12} \to 0$ and no $\phi \to f\bar{f}'$

For the sake of clarity in illustrating the salient features of our model, we will start from a simplified version and build up the full model in the next sections. Let's ignore for now the $\phi \to f\bar{f}'$ splitting and consider a one fermion initial state, possibly in a superposition of f_1 and f_2. In the limit where $g_{12} \to 0$, $A_n^{i\to i'}$ reduces to $g_i^n \hat{\mathcal{A}}_n$ and we can then use eq. (4.2) to efficiently sample from the full cross-section using MCMC methods. In order to do so we introduce the splitting functions

$$P_{i\to i\phi}(\theta) = g_i^2 \hat{P}(\theta)\,, \tag{4.3}$$

where $\hat{P}(\theta)$ gives the dependence of the emission probability on the angle of emission (or alternatively the scale of the process) and the splitting amplitudes are computed to leading order in the coupling constants, meaning only including tree level Feynman Diagrams in perturbation theory. We also introduce the Sudakov factor

$$\Delta_i(\theta_1,\theta_2) = \exp\left[-g_i^2 \int_{\theta_1}^{\theta_2} d\theta' \hat{P}(\theta')\right]\,, \tag{4.4}$$

which gives the virtual and unresolved real contribution, i.e. it accounts for the probability of having no emission. The splitting functions and Sudakov factor follow the unitary relation

$$\Delta_i(\theta_1,\theta_2) + \int_{\theta_1}^{\theta_2} d\theta \, P_i(\theta) \, \Delta(\theta_1,\theta) = 1\,, \tag{4.5}$$

and are used to find the clssical parton shower expression for the final state cross-section:

$$\sigma_{n,i}(\theta_1,...,\theta_n) = g_i^{2n} \left[\prod_{j=1}^{n} \Delta_i(\theta_{j-1},\theta_j)\hat{P}(\theta_j) \right] \Delta(\theta_n,\varepsilon)\,. \tag{4.6}$$

We can efficiently sample from this expression by generating one emission at the time using a MCMC.

4.3 MCMC fails when $g_{12} \neq 0$

Now we let $g_{12} \neq 0$ while still suppressing the $\phi \to f\bar{f}'$ splitting. The initial fermion(s) can now change type (or spin) yielding interferences between intermediate states which cannot be accounted for with eq. (4.6), but must be accounted for by working with the amplitudes directly as we shall now describe. We once again consider an initial state consisting of one fermion. The kinematics of the jet will be specified by the number of bosons emitted and their emission angles, and for a final state with one fermion and a maximum of n bosons (i.e. n steps in the evolution) the amplitude is written as

$$A_n^{i\to i'} \equiv \mathcal{A}(i \to i' + n\phi),\,. \tag{4.7}$$

Throughout the evolution there is always one fermion but at each boson emission the fermion can change spin. These $n-1$ intermediate states cannot be observed and the final particle state is therefore a superposition of the 2^{n-1} possible particle configurations.

For example, if we start with a f_1 fermion and end up with a f_1 fermion, the amplitudes for one and two emission steps are

$$
\begin{aligned}
A_1^{1\to 1} &= g_1 \hat{\mathcal{A}}_1(\theta_1) \\
A_2^{1\to 1} &= \sqrt{(g_1^2 + g_{12}^2)}\hat{\mathcal{A}}_2(\theta_1, \theta_2),
\end{aligned} \tag{4.8}
$$

to leading order in the coupling constants. We see that already for two emissions a non-trivial interference appears between the fermion switching type twice and not switching at all. As the number of boson emissions grows the combinatorics to obtain the correct coupling in front of $\hat{\mathcal{A}}_n$ grows exponentially. Factorization (eq. (4.2)) still holds but because the amplitude for a boson to be emitted at a given step depends on the spin of the fermion, which will be in general in a superposition of up (type 1) or down (type 2), the classical parton shower of eq. (4.6) cannot be used to correctly sample the final state cross-section. The MCMC approach multiplies together the probabilities of emission or not emission at each step, which would end up neglecting the interference of amplitudes such as the one in $A_2^{1\to 1}$ above, and these interference effects can only be accounted for by working with the amplitudes directly. Emissions in which the fermion type can change are modeled using the density matrix formalism. The splitting functions are then represented by splitting matrices as

$$
P_{i\to j\phi}(\theta)\left|f_i\right\rangle\left\langle f_j\right| = G_{ij}\hat{P}(\theta) = \begin{bmatrix} g_1 & g_{12} \\ g_{12} & g_2 \end{bmatrix} \tag{4.9}
$$

where in the limit $g_{12} \to 0$ we get $P_{i\to j\phi}(\theta) \to \delta_{ij} g_i^2 \hat{P}(\theta)$, and G is necessarily unitary..

Even though we cannot use MCMC methods to simulate this system, because we are neglecting the $\phi \to f\bar{f}'$ splitting, our system still holds limited complexity and can be solved efficiently with classical methods. We first found a simple quantum circuit able to simulate emissions in this model in linear time, which quickly led us to find a quantum-inspired classical algorithm which also solves the problem efficiently. This is a fairly common situation where looking for quantum algorithms shines light on new classical algorithms which can solve the problem at hand, and it is one of the added benefits of conducting research in quantum computing. We will now illustrate both the quantum circuit, as a warm up for the full quantum simulation we will describe later, as well as the classical algorithm.

4.3.1 Quantum algorithm solution

We start with one fermion (possibly in a spin superposition) and we ignore $\phi \to f\bar{f}'$, as well as the running of the coupling. Then the full evolution can be simulated with the quantum circuit in Figure 4.1. The $|f\rangle$ encodes the fermion state, $|0\rangle$ for f_1 and $|1\rangle$ for f_2 (possibly in a superposition of the two), while the $|\phi_i\rangle$ states, which are initialized in the $|0\rangle$ state, encode whether or not an emission occurred at step i. We start the evolution by applying the matrix U

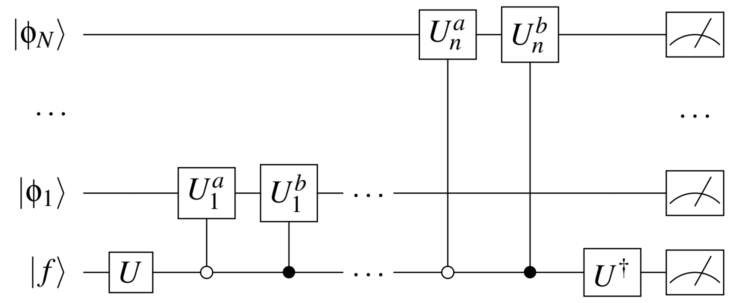


Figure 4.1: Quantum circuit for full evolution with no $\phi \to f\bar{f}$ splitting and no running coupling.

which rotates the fermion state into a new basis $f_{a/b}$ (now $|0\rangle$ corresponds to f_a and $|1\rangle$ to f_b) in which there is no mixing between fermion states. In this basis fermions do not change type and applying the U operation is equivalent to diagonalizing the splitting matrix G from Eq. (4.9):

$$G^{\text{diag}} = UGU^\dagger = \begin{pmatrix} g_a & 0 \\ 0 & g_b \end{pmatrix}, \tag{4.10}$$

where

$$U = \begin{pmatrix} \sqrt{1-u^2} & u \\ -u & \sqrt{1-u^2} \end{pmatrix}, \tag{4.11}$$

with

$$u = \sqrt{\frac{(g_1 - g_2 + g')}{2g'}}, \tag{4.12}$$

and

$$g_a = \frac{g_1 + g_2 - g'}{2}, \qquad g_b = \frac{g_1 + g_2 + g'}{2}, \tag{4.13}$$

$$g' = \text{sign}(g_2 - g_1)\sqrt{(g_1 - g_2)^2 + 4g_{12}^2}. \tag{4.14}$$

The evolution is then performed through the $U_i^{a/b}$ gates, which are given by the matrices

$$U_k^{a/b} = \begin{pmatrix} \sqrt{\Delta_{a/b}(\theta_k)} & -\sqrt{1-\Delta_{a/b}(\theta_k)} \\ \sqrt{1-\Delta_{a/b}(\theta_k)} & \sqrt{\Delta_{a/b}(\theta_k)} \end{pmatrix} \tag{4.15}$$

encoding the probability of a boson to be emitted at a given step, i.e. for a given value of the evolution variable θ. These matrices are applied conditionally on the fermion type, rotating a qubit $|\phi_i\rangle$ from the initial $|0\rangle$ state to a superposition of $|0\rangle$ (no emission happened) and $|1\rangle$ (a boson was emitted) with the correct amplitudes in terms of Sudakov factors (note that $1 - \Delta_{a/b}(\theta_k)$ is the probability of a fermion of type a/b to emit a boson at $\theta = \theta_k$). If we were to include the running of the coupling, as we will in the full simulation, we would have to rotate back and forth between the $1/2$ and a/b basis at each step, since U would depend on the scale θ though the couplings g_i.

The evolution in the $f_{a/b}$ basis is straight forward and does not involve any interferences between fermion states. Since we are ignoring the running of the coupling here, we simply rotate back to the $1/2$ basis at the end, by applying $U^\dagger$ to the fermion state. This operation creates interferences between different intermediate fermion states as we described above. Finally, by measuring all qubits we sample the final state distribution (which includes interference effects) thus obtaining one event, analogous to the events generated in parton showers.

Because of the circuit structure it is clear that the complexity of the algorithm grows linearly with the number of steps to simulate. This is a very simple quantum system and in fact we notice that, once the $|\phi\rangle$ qubits are acted on with $U^{a/b}$, they are left alone until the end of the circuit where they are measured. This means we could actually measure such qubtis right away, store the result in a classical register, set the qubit back to $|0\rangle$ and reuse it for the next step. This way, employing repeated measurements, we can run the same algorithm using only two qubits, as shown in the circuit in Figure 4.2. The combination of the n measurements on the $|\phi\rangle$ qubit and

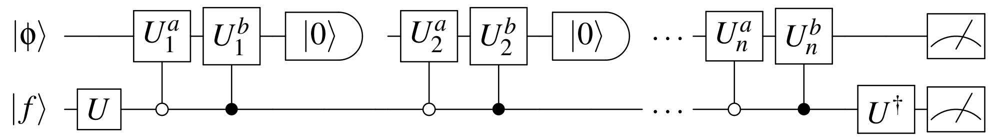


Figure 4.2: Quantum circuit for full evolution with no $\phi \to f\bar{f}$ splitting employing repeated measurements and only two qubits.

the final measurement on the $|f\rangle$ qubit yieldsß one event.

4.3.2 Quantum inspired classical algorithm

Because of the simplicity of the quantum circuit in Figure 4.2 (in particular the fact that it only requires two qubits) we knew there had to be an efficient classical algorithm to simulate the same shower evolution, and we found in fact a quantum inspired classical algorithm that uses random variables to generate events like the quantum circuits of the previous section.

Let's start by considering the evolution in step k of the quantum circuit in Figure 4.2.

Because we reset the second qubit to $|0\rangle$, using the computational basis of Eq. (2.7) the two-qubit state at the beginning of the step has the following form:

$$\left|\psi_k^i\right\rangle = \begin{pmatrix} a_1^{(k)} \\ 0 \\ a_3^{(k)} \\ 0 \end{pmatrix}. \tag{4.16}$$

Applying the conditional U_k 's operations yields the state

$$\left|\psi_k^f\right\rangle = \begin{pmatrix} b_1^{(k)} \\ b_2^{(k)} \\ b_3^{(k)} \\ b_4^{(k)} \end{pmatrix}, \tag{4.17}$$

where the $b_i^{(k)}$ are determined by multiplying the 4x4 matrix which represents the conditional U_k operations with the state $\left|\psi_k^i\right\rangle$. Then the probabilities P_0 and P_1 to measure the second qubit as $|0\rangle$ or $|1\rangle$ are

$$P_0 = {b_1^{(k)}}^2 + {b_3^{(k)}}^2, \qquad P_1 = {b_2^{(k)}}^2 + {b_4^{(k)}}^2, \tag{4.18}$$

and the corresponding states after resetting the second qubit to $|0\rangle$ are given by

$$|\psi_k\rangle_0 = \frac{1}{\sqrt{P_0}} \begin{pmatrix} b_1^{(k)} \\ 0 \\ b_3^{(k)} \\ 0 \end{pmatrix}, \qquad |\psi_k\rangle_1 = \frac{1}{\sqrt{P_1}} \begin{pmatrix} b_2^{(k)} \\ 0 \\ b_4^{(k)} \\ 0 \end{pmatrix}. \tag{4.19}$$

Both of these states have form

$$|\psi_{k+1}\rangle = \begin{pmatrix} a_1^{(k+1)} \\ 0 \\ a_3^{(k+1)} \\ 0 \end{pmatrix}, \tag{4.20}$$

which has exactly the same form of the state we started with, so that this process can be repeated again. To implement this classically we simply use a random number generator to choose between $|\psi_k\rangle_0$ and $|\psi_k\rangle_1$ with the correct probabilities. The complete algorithm is as follows:

```
Create empty vector for classical register $c_\phi[m]$
Set $a_1 = a$ and $a_3 = \sqrt{1-a^2}$
**for** step $= 1 \ldots m$ **do**
    Set $b_i = U_{ij} a_j$
    Set $P_0 = (b_1^2 + b_3^2)$ and $P_1 = b_2^2 + b_4^2$
    **if** rand() $< P_0$ **then**
        $c[step] = 0$
        $a_1 = b_1/\sqrt{P_0}$ and $a_3 = b_3/\sqrt{P_0}$
    **else**
        $c[\text{step}] = 1$
        $a_1 = b_2/\sqrt{P_1}$ and $a_3 = b_4/\sqrt{P_1}$
    **end if**
**end for**
**if** rand() $< a_1^2/(a_1^2 + a_3^2)$ **then**
    $c_f = 0$
**else**
    $c_f = 1$
**end if**
```

ALGORITHM 4.1: Quantum inspired classical algorithm.

4.4 Full Quantum Simulation

We now consider the full model with $g_{12} \neq 0$ and including the $\phi \to f\bar{f}'$ splitting. The complexity of the kinematics grows significantly, and we do not believe a classical algorithm exists that is able to simulate the system and sample from the full final state distribution with polynomial efficiency, a task that our quantum algorithm will instead be able to accomplish. The initial state will consist of n_I initial particles, which can be scalar bosons ϕ, $f_{1/2}$ fermions or any superposition of these. The quantum circuit block in Figure 4.3 simulates one step from the N step evolution of the inital particle state, where at each step particles can radiate according to the interactions specified by the Lagrangian in Eq. (4.1). The full quantum circuit is obtained by repeating this block N times.

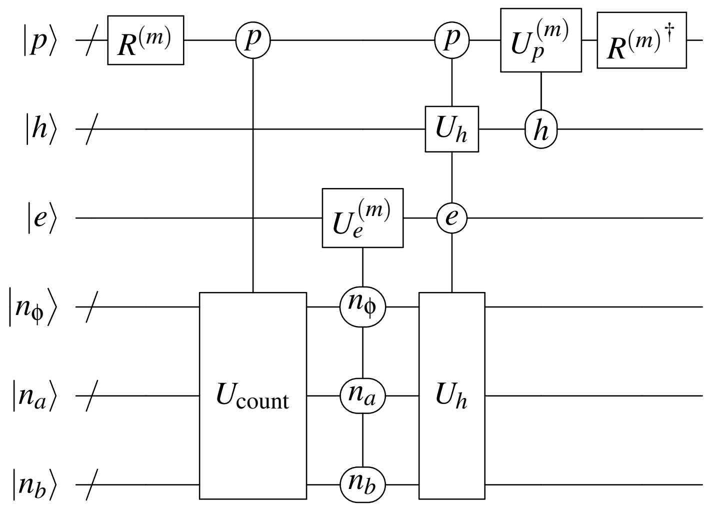


Figure 4.3: Quantum circuit for one step.

We still begin by rotating the particle states into the $f_{a/b}$ basis, through the $R^{(m)}$ gate, so that we can perform the evolution in a basis where emissions are uncorrelated and fermions do not mix. Then, the rest of the circuit is dedicated to count how many particles of each type we have at the beginning of the step, compute from this the correct amplitudes for each particle to emit, determine whether an emission occurred or not, generate a superposition of all the possible emissions that can happen with the correct amplitudes and adjust all the superimposed particle states based on the respective emissions. Then we rotate back into the $f_{1/2}$ basis before measuring the particle states, thus creating interference between intermediate fermion states similarly to the ones we described in Section 3.3.1. We will now describe the algorithm and its components in detail.

4.4.1 Particle Registers

The quantum circuit for the full simulation requires 6 qubit registers, which are listed in Table 4.1 together with the necessary number of qubits in each register in order to perform an N-step evolution starting from n_I initial particles. The first two registers are physical, meaning

Table 4.1: Registers in quantum circuit with the number of qubits they require for N steps and n_I initial particles. The symbol $\lceil \ldots \rceil$ denotes the ceiling function.

Register	Purpose	# of qubits
$\|p\rangle$	Particle state	$3(N+n_I)$
$\|h\rangle$	Emission history	$N\lceil \log_2(N+n_I) \rceil$
$\|e\rangle$	Did emission happen?	1
$\|n_\phi\rangle$	Number of bosons	$\lceil \log_2(N+n_I) \rceil$
$\|n_a\rangle$	Number of f_a	$\lceil \log_2(N+n_I) \rceil$
$\|n_b\rangle$	Number of f_b	$\lceil \log_2(N+n_I) \rceil$

they are only measured at the end of the full evolution to provide the sampling of the final state distribution. The other four registers instead, are work registers, i.e. they are necessary to implement the desired operations but they do not hold any physical information and are reset to zero at the end of each step, in order to be used again at the successive step.

The first register, $|p\rangle$, contains the information about the particle state. Each particle can be in one of 6 states $|0\rangle$, $|\phi\rangle$, $|f_{a/b}\rangle$, and $|\bar{f}_{a/b}\rangle$. To encode these 6 states one requires 3 qubits, and we choose the following representation

$$|p\rangle_i = \begin{pmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{pmatrix} = \begin{pmatrix} 0 \\ \phi \\ - \\ - \\ f_1/f_a \\ f_2/f_b \\ \bar{f}_1/\bar{f}_a \\ \bar{f}_2/\bar{f}_b \end{pmatrix}, \tag{4.21}$$

where the third and fourth states are not used and one chooses $f_{1/2}$ and $f_{a/b}$ before and after the basis change. There can be up to $N+n_I$ particles in the system, so we need a total of

$$\dim[|p\rangle] = 3(N+n_I) \tag{4.22}$$

qubits in this register.

The second register, $|h\rangle$, is the history register which encodes the information of which particle, if any, emitted during each step. Then, we actually have a subregister $|h\rangle_m$ for each step, which must be able to hold integers from 0 (meaning no particle emitted in the given step) to $N+n_I-1$ (the number of particles that can radiate during the last step). If $|h\rangle_m$ is in state $|j\rangle$ it means that at step m particle j was responsible for an emission. Then, if we want all subregisters to be of same size, the history register will require

$$\dim[|h\rangle] = N\lceil\log_2[(N+n_I)]\rceil \tag{4.23}$$

qubits.

The third register, $|e\rangle$, holds the information on whether an emission occurred or not during a given step, and being a work register is reset to the initial state at the end of the step. Then we only need a single qubit for this register and

$$\dim[|e\rangle] = 1\,. \tag{4.24}$$

The remaining three registers are count registers, which hold the information on how many bosons, fermions of type a and fermions of type b (including both f and $\bar{f}$) are in the current state. These registers must be able to hold integers $0,\ldots,N+n_I-1$, so if we use the binary representation to represent these integers one needs

$$\dim[|n_\phi\rangle] = \dim[|n_{a/b}\rangle] = \lceil \log_2[(N+n_I)] \rceil \tag{4.25}$$

qubits.

At the beginning of the evolution all qubits in the work registers $|e\rangle$, $|n_\phi\rangle$, $|n_a\rangle$, and $|n_b\rangle$ start out in the $|0\rangle$ state. For the physical registers, all qubits in the $|h\rangle$ register are initialized to the $|0\rangle$ state, while for the particle register $|p\rangle$, the particle states $|p_i\rangle$ with $i > n_I$ are initialized to zero, while the ones with $i \leq n_I$ are set to encode the initial particle states, all according to the representation in Eq. (4.21). We now move on to describing the operations in the quantum circuit.

4.4.2 Changing basis

The first thing we do, after inputting the initial particle states into the particle register, is to rotate to the $f_{a/b}$ basis, where there is no mixing between different types of fermions. This is very similar to what we did for the simplified algorithm of Section 3.3.1, but now particle states are encoded in three qubits instead of one and this change of basis operation is represented by the

8x8 unitary matrix

$$R = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & U & 0 \\ 0 & 0 & 0 & U \end{pmatrix}, \tag{4.26}$$

where the 2x2 U matrix is the one given in Eq. (4.11). This must be applied to all three-qubit particle states in $|p\rangle$ and as we have previously mentioned, if we include the running of the coupling, the U matrix, hence the R matrix, will be different at each step, so that you have to rotate back and for between the two basis at the beginning and end of each step as shown in the circuit in Figure 4.3.

4.4.3 First operation: counting particles

After the change of basis the quantum circuit consists of four main operations. The first operation counts the number of particles of each type and stores these numbers in the three count registers $|n_\phi\rangle$, $|n_a\rangle$ and $|n_b\rangle$. We do this by applying the controlled-U_{count} operation conditional on the particle states, whose circuit decomposition is shown in Figure 4.4. We loop over all

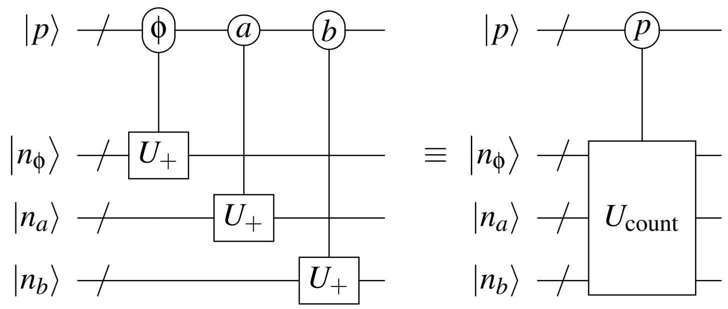


Figure 4.4: Quantum circuit for counting particles.

particle states $|p\rangle_i$ and if $|p\rangle_i$ is in a fermion or boson state the U_+ operation is applied to the respective count register. U_+ acts on an integer in binary representation (where 0 corresponds to

$|0\rangle$ and 1 corresponds to $|1\rangle$) as

$$U_+ |n\rangle = |n+1\rangle_{\text{mod}_{N+n_I}} , \tag{4.27}$$

or in matrix form, $(U_+)_{ij} = 1$ if $j = i+1 \bmod (N+n_I)$ and 0 otherwise. The U_+ can be decomposed into elementary and Toffoli quantum gates as shown in Figure 4.5, while the controls

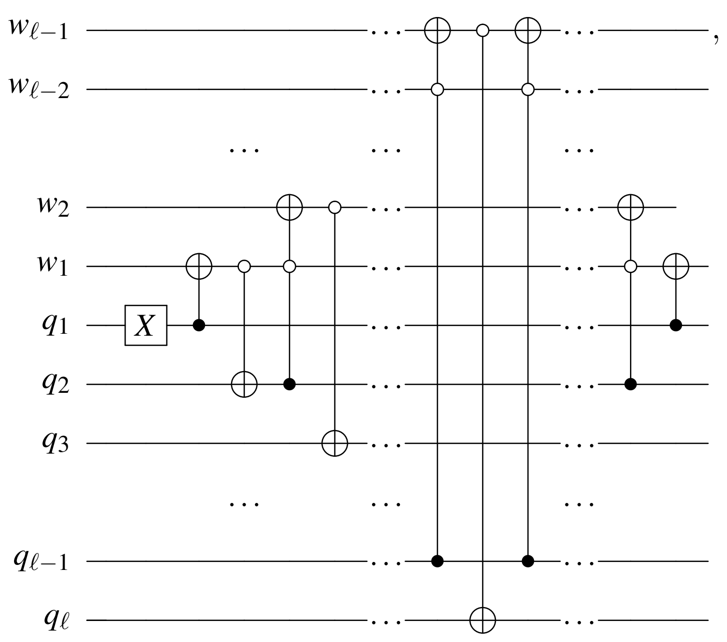


Figure 4.5: U_+ decomposition into elementary and Toffoli gates. The largest integer we can count here is n, where $\ell = \lceil \log_2(n) \rceil$.

on the particle type which appear in Figure 4.4 are implemented as illustrated in Figure 4.6. These controls are applied to all of the gates in the decomposition of U_+, yielding many instances of an X gate controlled on 2, 3, 4 or 5 qubits. We saw how to decompose such operations into elementary gates in Section 1.6.

4.4.4 Second operation: to emit or not to emit

In the second operation we determine whether or not an emission happened during the current step. Similarly to what we described in Section 3.2, in the a/b basis the interactions

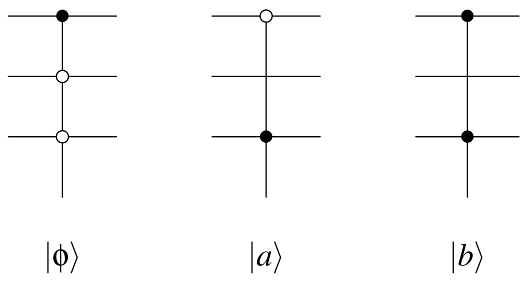


Figure 4.6: Decomposition of the controls on the particle states ϕ, f_a and f_b.

cannot change the flavor of the fermion and the evolution is described by splitting functions and Sudakov factors as in MCMCs. The splitting functions are now

$$P_{i\to i\phi}(\theta) = g_i^2 \hat{P}_f(\theta)\,, \tag{4.28}$$

and

$$P_{\phi\to i\bar{i}}(\theta) = g_i^2 \hat{P}_\phi(\theta)\,, \tag{4.29}$$

where $i \in \{a,b\}$ specifies the fermion flavor. Recall that $P_f(\theta)$ and $P_\phi(\theta)$ specify the functional dependence of the splitting amplitude on the evolution scale, which can be specified by the time t, angle of emission θ or other dynamical variables. For instance, in the implementation of our algorithm we will use $P_f(\theta) = P_\phi(\theta) = log(\theta)$.

With the splitting functions we can define the Sudakov factors, which encode the probability of not emitting, as

$$\begin{aligned}\Delta_i(\theta_m,\theta) &\equiv \exp\left[-\Delta\theta\, P_i(\theta_m)\right]\\ \Delta_\phi(\theta_m,\theta) &\equiv \exp\left[-\Delta\theta\, P_\phi(\theta_m)\right]\,,\end{aligned} \tag{4.30}$$

where

$$
\begin{aligned}
P_i(\theta_m) &\equiv P_{i\to i\phi}(\theta_m) \\
P_\phi(\theta_m) &\equiv P_{\phi\to a\bar{a}}(\theta_m) + P_{\phi\to b\bar{b}}(\theta_m)\,,
\end{aligned}
\tag{4.31}
$$

and

$$
\Delta\theta = \theta_m - \theta\,. \tag{4.32}
$$

If in the current step we have a particle state consisting of n_ϕ bosons and $n_{a/b}$ fermions, where these numbers have been stored in the count registers in the previous operation, the probability of having no emission in this step is given by

$$
\Delta^{(m)}(\theta_m,\theta_{m+1}) = \Delta_\phi^{n_\phi}(\theta_m,\theta_{m+1})\Delta_a^{n_a}(\theta_m),\theta_{m+1}\Delta_b^{n_b}(\theta_m,\theta_{m+1})\,. \tag{4.33}
$$

Meanwhile, the probability of having an emission (any emission) is

$$
\begin{aligned}
q_p(\theta_m) &\equiv \int_{\theta_m}^{\theta_{m+1}} d\theta P_p(\theta_m)\Delta_p(\theta_m,\theta) \\
&= 1 - \Delta_p(\theta_m,\theta_{m+1})\,.
\end{aligned}
\tag{4.34}
$$

We can encode these emission and non-emission probabilities in our quantum circuit through the operation U_e controlled on the count registers, as shown in Figure 4.7, and represented by the matrix

$$
U_e^{(m)} = \begin{pmatrix} \sqrt{\Delta^{(m)}(\theta_m)} & -\sqrt{1-\Delta^{(m)}(\theta_m)} \\ \sqrt{1-\Delta^{(m)}(\theta_m)} & \sqrt{\Delta^{(m)}(\theta_m)} \end{pmatrix}, \tag{4.35}
$$

which acts on the qubit $|e\rangle$, initially in the state $|0\rangle$, rotating it into a superposition of $|0\rangle$ (no emission happened) and $|1\rangle$ (an emission happened) with the correct respective amplitudes. The

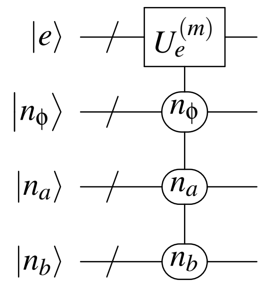


Figure 4.7: circuit operation for determining whether an emission occurred at the m^{th} step.

entries of U_e depend on the number of each particle type in the current state through Eq. (4.33), and this is the first reason we needed to count particles and store these numbers into count registers. When we implement this operation, we must account for all possible combinations of particles that can populate the particle state at this stage, this means we must apply U_e controlled on all possible combinations of three integers, ranging from 0 to $m+n_I-1$, whose sum must be in the range $[n_I, m+n_I-1]$, where m is the current step's number. This is an instance of a single qubit gate controlled on n qubits, which we have previously shown how to decompose into elementary quantum gates.

4.4.5 Third operation: deciding the emitting particle

If an emission did happen, the third operation decides which particle emitted and assigns the correct amplitude for that emission. In order to do this we "loop" over all particles up to the Mth particle state, applying the $U_h^{(m,k)}$ sub-operations as shown in Figure 4.8, where $M = m+n_I-1$, m is the current step and the -1 is because we don't loop over the particle which might have just been emitted in the current step. The sub-operations are decomposed as shown in Figure 4.9

This is the most subtle operation so let's analyze it in detail. We are in the mth step, $M \equiv m+n_I-1$ is the maximum number of particles that can be present in the current state and

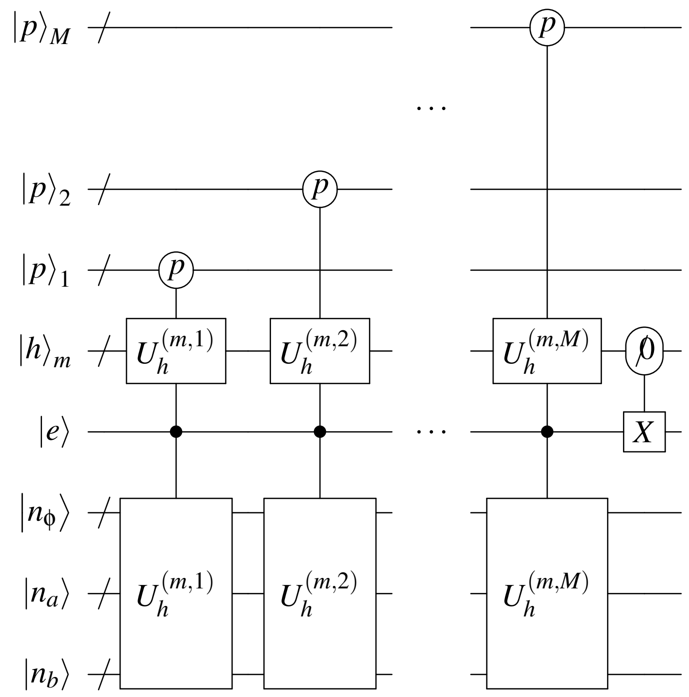


Figure 4.8: Sub-operations that "loop" over particle states up to the *M*th particle and make up the third main operation in the *m*th step.

we work with the history sub-register $|h\rangle_m$, which contains states labeled by integers from $|0\rangle$ to $|M\rangle$. In the k^{th} (out of M) sub-operation we apply the gate $U^{(m,k)}$ whose action is best described by a 2x2 unitary sub-matrix which always acts between state $|0\rangle$ and state $|k\rangle$ in $|h\rangle_m$, and which is given by

$$U^{(m,k)} = \begin{pmatrix} \sqrt{1 - \frac{P_{p_k}(\theta_m)}{P(n_\phi,n_a,n_b)}} & -\sqrt{\frac{P_{p_k}(\theta_m)}{P(n_\phi,n_a,n_b)}} \\ \sqrt{\frac{P_{p_k}(\theta_m)}{P(n_\phi,n_a,n_b)}} & \sqrt{1 - \frac{P_{p_k}(\theta_m)}{P(n_\phi,n_a,n_b)}} \end{pmatrix} . \tag{4.36}$$

where

$$P(n_\phi, n_a, n_b)(\theta_m) = \sum_p n_p P_p(\theta_m) , \tag{4.37}$$

and the $P_p(\theta_m)$'s were given in the previous section. The last controlled-X operation sets the

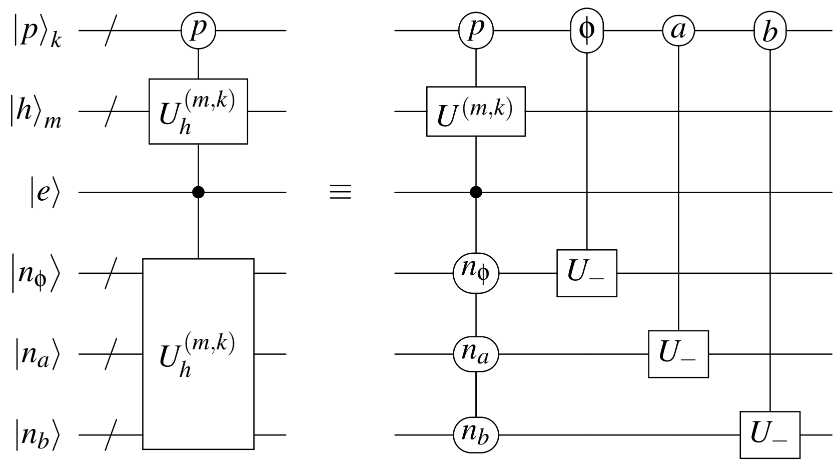


Figure 4.9: Circuit decomposition for the k^{th} sub-operation from the third main operation.

emission register $|e\rangle$ back to the initial $|0\rangle$ state.

After having determined an emission happened in the previous operation, the probability of having a specific emission is essentially given by the relative probability of that emission to happen with respect to the sum of all emission probabilities, hence the form of the entries of $U^{(m,k)}$ (where the square roots are because we must work with amplitudes).

We control on $|p\rangle_k$ the same way we did for the count operation as shown in Figure 4.4 and Figure 4.6, plus controlling on the number of particles of each type in the current particle state, like we did in the second operation (see Figure **??**). However, here the operations we apply are different for each combination of controls, since the entries of the $U^{(m,k)}$ matrix depend on the particle flavor and the number of particles through P_{p_k} and $P(n_\phi, n_a, n_b)(\theta_m)$. Of course we also control on the qubit $|e\rangle$ being in a $|1\rangle$ state, which means an emisison did happen.

In each sub-operation, after applying the $U^{(m,k)}$ gate, we apply U_- gates conditional on the particle flavor, which acts as following:

$$U_- |n\rangle = |n-1\rangle_{\text{mod}_{N+n_I}} , \tag{4.38}$$

and which is decomposed similarly to the U_+ gate (see Figure 4.5) except that the controls on the work qubits here control on the qubits being in the $|1\rangle$ state, instead of the $|0\rangle$ state like in the U_+ decomposition.

In each step, the sub-operations build up recursively a superposition of all the possible emissions, i.e. of all possible histories for the step, with the correct amplitudes. In each sub-operation, after the U gate is applied, the controlled U_- gates are applied so that the entries for $U^{(m,k)}$ in the k^{th} sub-operation come from the relative probability that the p_k particle emitted out of the particles between (and including) p_k and p_M. The value of $P(n_\phi, n_a, n_b)(\theta_m)$ then changes at each step, until in the last step we have

$$P(n_\phi, n_a, n_b)(\theta_m) = P_{p_k}(\theta_m)\,, \tag{4.39}$$

so that the last 2×2 sub-matrix U is always of form

$$U^{(m,m)} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{4.40}$$

The last sub-operation then rotates or transfers the amplitude of the $|0\rangle$ state of $|h\rangle_m$, which was built up out of non-emission amplitudes from all the previous U's, to the $|M\rangle$ state. Recall that we are applying all of these sub-operations conditionally on an emission having occurred. Therefore, at the last sub-operation, the amplitude for last particle to be the one which emitted is the same as the amplitude for all of the previous particles not having emitted, which is what we are transferring to the $|M\rangle$ state.

Notice also that at the end of this operation, the U_- gates have been applied conditionally on all of the particle states in $|p\rangle$, which is exactly the inverse of the counting operation, so that the three count registers are back to the initial $|0\rangle$ state, ready to be used again in the next step.

As a quick aside, I want to emphasize again that all of these unitary operations which are applied, often with controls, are essentially rotations between quantum states which always create superpositions of states, and these states are then evolved in subsequent operations and evolution steps. In a sense all possible emission histories are created and evolved with the correct respective amplitudes, and at the end only one is picked out by a measurement. So when we say that in the second operation we decide whether or not an emission happened, the algorithm actually creates and processes both possibilities and the choice is really made when we measure at the end. This of course, is one of the key features of quantum computation and principal source of its potential, but that said, it is unintuitive to think in these terms, making it often challenging to come up with new efficient quantum algorithms or even to follow how existing ones work.

4.4.6 Fourth operation: adjusting the particle state

After we determined the emission history of the step we must adjust the particle states in $|p\rangle$ accordingly; this is done in the fourth and last main operation, whose circuit schematic is shown in Figure 4.10. The operation labeled with U_p is controlled on the k^{th} state of $|h\rangle_m$ and acts on the particle sub-registers $|p\rangle_k$ and $|p\rangle_{M+1}$. We must loop over the states in $|h\rangle_m$ and apply the controlled-U_p operation for all k's from 1 to M. Note that we control on the k^{th} state in the history sub-register, which specifies which particle emitted but does not have the information of the flavor of the particle, which is provided by the particle state $|p\rangle_k$.

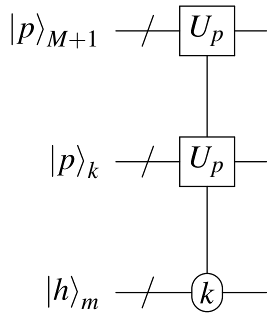


Figure 4.10: Circuit for the operation at step m which fixes the particle register after the emission has happened. As before, $M = m + n_I - 1$. Notice that if we control on $|h\rangle$ being in the $|k\rangle$ state, we apply U_p to the k^{th} sub-register $|p\rangle_k$ and the $(M+1)^{\text{th}}$ sub-register $|p\rangle_{M+1}$.

For example, if the we determined that the boson encoded in $|p\rangle_k$ split into a $f_a\bar{f}_a$ pair we must remove the ϕ from the full particle state and add the fermion-antifermion pair. Either f_a or $\bar{f}_a$ are encoded in $|p\rangle_k$ in place of the ϕ while the other one is encoded in the next sub-register $|p\rangle_{M+1}$ (in fact, a superposition of the two cases is actually created). On the other hand, if a fermion emits a boson, we simply have to add a $|\phi\rangle$ to the particle register by encoding it in $|p\rangle_{M+1}$. We want the U_p operation to act on the $|p\rangle_k |p\rangle_{M+1}$ states as follows:

$$\begin{aligned}
&|f_i\rangle |0\rangle \to |f_i\rangle |\phi\rangle \\
&|\bar{f}_i\rangle |0\rangle \to |\bar{f}_i\rangle |\phi\rangle \\
&|\phi\rangle |0\rangle \to \sum_{i=a,b} \hat{g}_i \left(|f_i\rangle |\bar{f}_i\rangle + |\bar{f}_i\rangle |f_i\rangle\right) ,
\end{aligned} \tag{4.41}$$

where

$$\hat{g}_i \equiv \frac{g_i}{\sqrt{2(g_a^2+g_b^2)}} . \tag{4.42}$$

This can be carried out by the following unitary operator:

$$\begin{aligned}
U_p = &\sum_{i=a,b} |f_i\rangle |\phi\rangle \langle f_i| \langle 0| + \sum_{i=1,b} |\bar{f}_i\rangle |\phi\rangle \langle \bar{f}_i| \langle 0| \\
&+ \sum_{i=a,b} \hat{g}_i \left(|f_i\rangle |\bar{f}_i\rangle + |\bar{f}_i\rangle |f_i\rangle\right) \langle \phi| \langle 0| ,
\end{aligned} \tag{4.43}$$

where the unitarity can be seen by recalling that states of particles with different flavors are orthogonal. The circuit implementation of this operation is given in Figure 4.11, where H is the Hadamard gate and U_r is given by:

$$U_r = \frac{1}{\sqrt{g_a^2+g_b^2}} \begin{pmatrix} g_a & -g_b \\ g_b & g_a \end{pmatrix} . \tag{4.44}$$

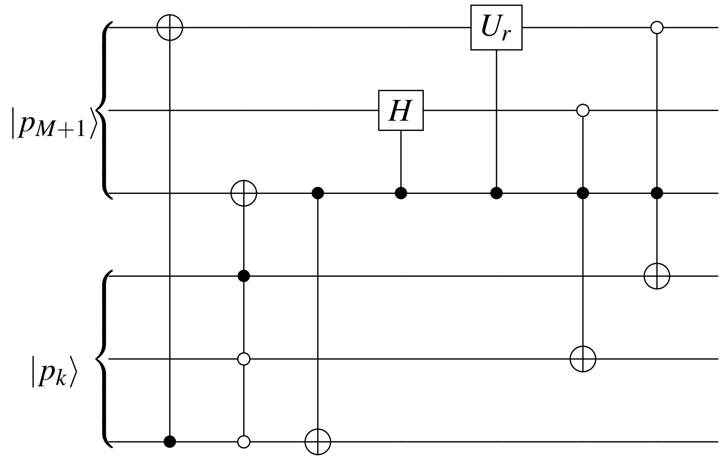


Figure 4.11: The circuit for the U_p operation.

4.4.7 Sampling

To generate one event, we start by selecting the emission angle cutoff where we want the evolution to stop, which corresponds to the cutoff scale t_c that we introduced when discussing parton showers in the previous chapter. Then, we specify the initial scale, in this case identified with the initial emission angle, and the total number N of evolution step we want to simulate going from the initial to the cutoff scale. The larger N, the more accurate the simulation will be. In each of the N steps the circuit in Figure 4.3 is run, with the gate parameters accurately computed based on the scale of the current step. Each step starts with rotating from the $f_{1/2}$ to the $f_{a/b}$ basis with the R operation, then the four main operations are applied and finally we rotate back to the $f_{1/2}$ basis with the $R^\dagger$ operation, which is the inverse of the R rotation, thus creating interferences between identical final particle states which had intermediate fermions of different flavor in their shower history.

At the very end of the evolution, after the Nth step is completed, we are in the $f_{1/2}$ basis and we measure the qubits which make up the physical registers, i.e. the particle and history registers. Note that all other qubits are work qubits and at this point have already been rotated back into the $|0\rangle$ state. Measuring the particle register tells us the final particle state, meaning all the emitted/radiated particles which are present in the shower at the end of the evolution. Measuring the history register instead, tells us which particles in $|p\rangle$ radiated and at which angle θ, where the latter can be deduced from the step at which the emission happened. The information obtained through measurement however, is not enough to reconstruct the full evolution, since in the $1/2$ basis fermions can change flavor during emissions and in the end we will have a superposition of states with the same final particles and same emission history, but different final amplitudes, because of intermediate particle states with fermions of different type. The example in eq. (4.8) for the simplified model helps to illustrate the form of these interferences. It is almost like we decided to be ignorant of the flavor of the fermions during the shower evolution, in order to preserve these interference effects.

Measuring the particle and history registers corresponds to generating one event, i.e. one sample of the final state distribution of the shower evolution. We have argued that the complexity to generate this one event scales polynomially with the number of resources (qubits and elementary quantum gates), however, we must still make sure that obtaining the final state distribution of the shower with acceptable accuracy does not require to generate exponentially many events, otherwise we would loose all advantage and the problem would become once again intractable. In other words, if we are measuring a total of say T qubits between particle and history registers, there are 2^T possible states to be measured, so that if all states had a similar chance of being produced it would require more than 2^T events to be generated to obtain the distribution of final states with acceptable statistics. The solution to this problem lies in the fact that, in our shower evolution, out of the 2^T possible states, only a small fraction will have non-vanishing probability of being produced, and because quantum measurements yield states according to their relative probability of occurring in the Hilbert space of all possible quantum states, the number of events we must generate to sample with sufficient accuracy the final state distribution is small and scales polynomially with the number of qubits T.

4.4.8 Complexity

In this section we derive a measure of the complexity of the quantum circuit by counting how many elementary gates are necessary to implement each of the four main operations. We'll start by computing the complexity of some common operations, which are used multiple times in the implementation of our algorithm. In Section 2.5 we saw how to decompose an operation U controlled on n qubits, it required $n-1$ work qubits, $2\times(n-1)$ Toffoli gates and one $C(U)$ operation. A Toffoli gate requires 16 elementary gates to be implemented while a $C(U)$ operation requires 5. Then, assuming $n>2$ and that we control on all qubits being in a $|1\rangle$ state, quantum gates controlled on n qubits require

$$\begin{aligned} \left|C^{(n)}[X]\right| &= 32n-31 \\ \left|C^{(n)}[U]\right| &= 32n-27 \end{aligned} \tag{4.45}$$

elementary gates. To these numbers we must add 2 X-gates each time we control on a qubit being in the $|0\rangle$ state instead of the $|1\rangle$ state.

Let's now look at the first operation. The U_+ gate was broken down in Figure 4.5 and each operation in that figure was controlled on the particle states as shown in Figure 4.6. Using the above results and the fact that in the mth step we must be able to store integers up to $\ell = \lceil \log_2(m+n_I-1)\rceil$ in the count registers, the number of elementary gates required for the counting operation in the mth evolution step is

$$c_{\text{count}}(m,n_I) = 909\lceil \log_2(m+n_I-1)\rceil - 1010\,. \tag{4.46}$$

In the second operation, where we determine whether an emission happened, we apply the U_e gates controlled on all possible combinations of three integers between 0 and $m+n_I-1$

whose sum is in the range $[n_I, m+n_I-1]$. The number of such combinations is

$$c_2(m,n_I) = \frac{m+1}{6}(m^2+3mn_I+5m+3n_I^2+9n_I+6)\,. \tag{4.47}$$

For each of these combinations we apply a U_e gate controlled on $3\log_2(m+n_I-1)$ qubits, and using the above result for the complexity of a $C^{(n)}(U)$ operation, we computed the total number of elementary gates necessary to implement the second operation at step m to be

$$c_{emission}(m,n_I) = c_2(m,n_I)(96\lceil\log_2(m+n_I)\rceil - 27)\,. \tag{4.48}$$

In order to determine the complexity of the third main operation, which creates the emission history, we compute the number of gates required to implement the kth sub-operation. The gate $U^{(m,k)}$ gate has the same controls on the count registers as the controlled-U_e operation plus being also controlled on the emission qubit and on the particle state. We take the control on the particle state as a control on three qubits in the $|1\rangle$ state, thus ignoring the fact that for the fermions we actually only control on two qubits and that some controls are on $|0\rangle$ states. This is justified by the fact that the resulting error difference in gates is negligible when compared to the number of gates required for implementing the rest of the operation. So the $U^{(m,k)}$ gate is applied controlled by $4+3\lceil\log_2(m+n_I-k-1)\rceil$ qubits, where the $-k$ come from the reduction in the number of particle states due to the U_- gates.

For the kth sub-operation the matrix $U^{(m,k)}$ is an $a\times a$ unitary matrix, where $a=2^b$ and b is the number of qubits on which the matrix acts, given by $b=\lceil\log_2(k)\rceil$ since $U^{(m,k)}$ acts between the states $|0\rangle$ and $|k\rangle$, represented by integers in binary representation. There is a standard procedure to break down a specific two-level unitary matrix like $U^{(m,k)}$, outlined in Chapter 4 of [2], from which it turns out that the number of elementary gates to implement $U^{(m,k)}$ controlled on n qubits is

$$
\begin{aligned}
|C^{(n)}(U^{(m,k)})| &= 32(n-1) + |C^{(1)}(U^{(m,k)})| \\
&= 32(n-1) + 2(b-1)|C^{(1)}(X)| + |C^{(1)}(U)| \\
&= 64b^2 - 94b + 32n + 3\,. \qquad (4.49)
\end{aligned}
$$

As we mentioned above in this case $n = 4 + 3\lceil \log_2(m + n_I - k - 1)\rceil$.

The controlled-U_- gates have very similar complexity to the controlled-U_+'s of course, with the difference that we control on work qubits being in the $|1\rangle$ state, so that we need

$$
c_{countdown}(m, n_I) = 873\lceil \log_2(m + n_I - 1)\rceil - 968\,. \qquad (4.50)
$$

elementary gates to implement the controlled-U_- operations. All together the total number of elementary gates to implement the third operation at step m, summing over all sub-operations, is

$$
\begin{aligned}
c_{hist}(m, n_I) = \sum_{k=1}^{m+n_I} \Bigg[& c_{\mathrm{count}}(m, n_I) \\
& + 3c_2(m, n_I) \left| C^{(4+3\lceil \log_2(m+n_I-k)\rceil)}(U^{(m,k)}) \right| \Bigg]\,. \qquad (4.51)
\end{aligned}
$$

Finally, for the fourth operation the gate U_p is decomposed as shown in Figure 4.11, which is then applied controlled on all possible states in $|h\rangle$. There are a total of $m + n_I$ such states, encoded in $\lceil \log_2(m + n_I)\rceil$ qubits. U_p is applied $m + n_I - 1$ times and each time one controls all gates in the decomposition in Figure 4.11 on $\lceil \log_2(m + n_I)\rceil$ control qubits from the history register. Then to implement the fourth main operation at step m one needs

$$
c_{fix}(m, n_I) = (m + n_I)\left(224\lceil \log_2(m + n_I)\rceil + 143\right)\,. \qquad (4.52)
$$

elementary gates.

If we add together the number of elementary gates required for the each of the four operations and sum over $0 < m < N-1$, we find that the total number of universal gates for the circuit scales as $N^5 lnN$.

Because there is no interference between states with different emission histories, we could in principle measure the history register at each step, store the result and set it back to the initial $|0\rangle$ state. If we count again the elementary gates in the third operation (which is the one with highest complexity) we find that this modification reduces the circuit complexity to $N^3 lnN$. However, such repeated measurements are not implementable on current hardware, so we stick for now to the implementation using multiple history sub-registers.

4.5 Results

The full quantum circuit described in Section 3.4 requires too many interconnected qubits and quantum operations to be implemented and tested on current state-of-the-art quantum computers. Therefore, we were only able to test on a real quantum machine, the simplified algorithm we discussed in section 3.3, where we ignore the splitting $\phi \to f\bar{f}$, the running of the coupling and we start with only one fermion as our initial state. For this model we ran classical simulations (in Python and Mathematica), quantum simulations (using the IBM Q simulator called Qiskit) and we also ran the algorithm on a current quantum computer, the IBM Q Johannesburg Chip. With each of these methods we generated a large number of events which we then used to compute and plot the differential cross sections for various observables. We ran the algorithm with both $g_{12} = 0$, for which a classical simulation using MCMC is possible, and for $g_{12} \neq 1$, for which we have results from the quantum simulation and the actual quantum computer, but for which the MCMC simulation is not available. Since the IBM Q machine has limited number of qubits, gate depth and hardware fidelity, we chose to simulate $N = 4$ steps.

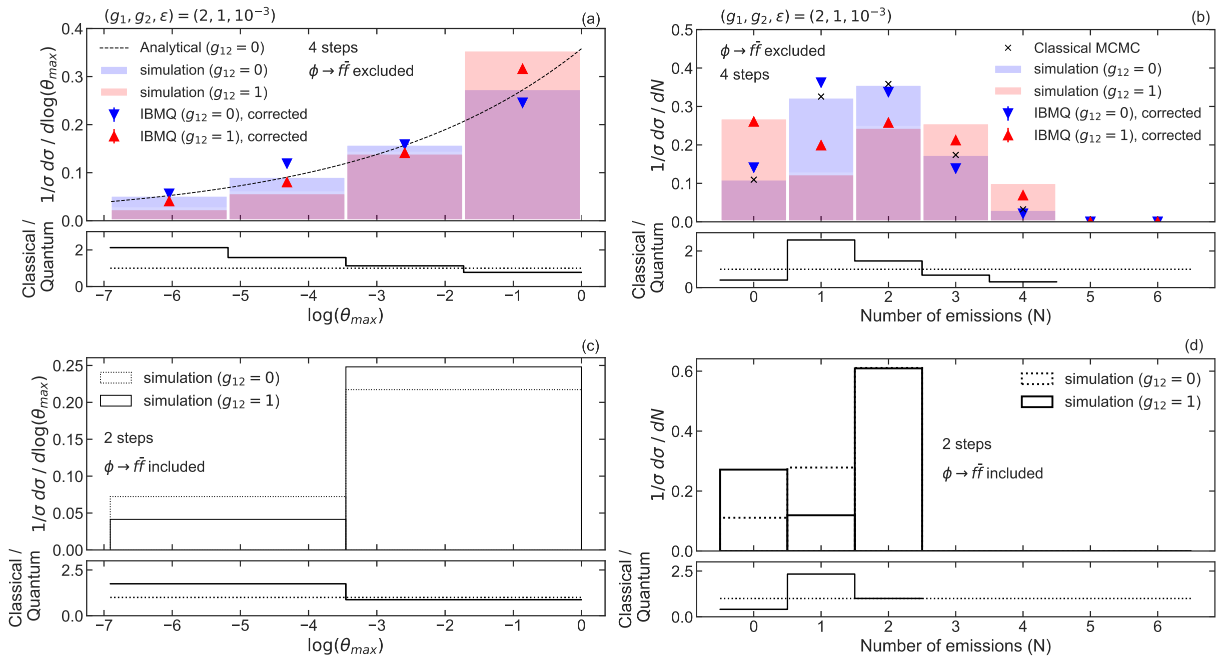


Figure 4.12: Normalized differential cross section for two observables: $\log \theta_{\max}$ (a,c) and the total number of emissions (b,d), for both ($g_{12} = 1$) and ($g_{12} = 0$), where the classical simulations/calculations are expected to agree with the quantum simulations and measurements.

For the full model instead, i.e. including the splitting of the scalar field into fermions, we were only able to run a 2-step complete quantum simulation using the IBM Quiskit simulator. We did this for both $g_{12} = 0$ and $g_{12} = 1$.

All results are displayed in Figure 4.12, which shows the plots of the normalized differential cross sections for two observables: the natural logarithm of the maximum emission angle (a,c) and total number of emissions (b,d). Over 10^5 events contribute to each cross section, yielding negligible statistical uncertainties. Figures 4.12(a,b) present the results of the four-step simulations and runs, where the legend entry IBMQ ”corrected” means that the run on the actual quantum computer was error corrected. Meanwhile, figures 4.12 (c,d) display the results of the 2-step full model simulations.

Let’s look at Figure 4.12(b) for example. We see that the results for the $g_{12} = 0$ simulation agree very well with the MCMC results (which also has $g_{12} = 0$) as expected, and more interestingly it also agrees fairly well with the results of the runs on the actual quantum computer for the case $g_{12} = 0$ (blue triangular marks). The discrepancy can be attributed to noise, still significant in current quantum computers, where a certain degree of gate failure and information-loss is unavoidable. This supposition is supported by the fact that performing error correction brought the blue marks closer to the results of the quantum simulation. We find something similar when comparing the results from the actual quantum computer with $g_{12} = 1$ (given by the red triangular marks) with the results from the respective simulation in Quiskit (red histogram bars). The two seem to follow a similar trend, but with $g_{12} = 1$ the quantum circuit becomes longer and deeper, so that the number of gate failure as well as the amount of information lost due to decoherence is larger, resulting in a larger discrepancy between the real and simulated results. Figures 4.12(c,d), while not validating or comparing the results of the two-step full simulation, are interesting in the regard that they provide a first step toward a full simulation which we hope one day to be able run on a more advanced quantum machine, plus furnishing some visual cues on the differences between the full model with $g_{12} = 0$ and the full model with $g_{12} = 1$.

The relevance of the presented results lies in that they represent an experimental proof of concept of our algorithm. It's the first step toward running the full model for a large number of steps, and with great accuracy, once more advanced quantum machines are available, a feat which is intractable with classical computers. Furthermore, the algorithm can be improved and expanded to come closer to simulating processes in the Standard Model. For instance, the full three-dimensional kinematics of splitting could be included by adding the sampling of momentum fractions and azimuthal angles with a hybrid quantum-classical approach. Extension to SU(2) could also be achieved by adding more fermions to account for leptons and quarks, and more bosons to represent the $W^{\pm}$ and Z bosons. These advancements would ideally lead to a full quantum simulation of electroweak showers, which follow collinear radiation and might be of increasing importance in future high-energy colliders or dark matter experiments.

Chapter 5

Beyond our algorithm

In this final chapter, before concluding the article, I want to briefly discuss a few other promising and fascinating applications of quantum computing in the physical sciences.

5.1 The Jordan-Lee-Preskill approach

In their 2011 paper ”Quantum Computation of Scattering in Scalar Quantum Field Theories” [4] perhaps the most famous paper in the field, Jordan, Lee and Preskill propose a quantum algorithm that is able to compute relativistic scattering amplitudes in massive ϕ^4 theories, a class of field theories which are relevant in various areas of quantum field theory, such as in describing the self-interactions of the Higgs boson. The proposed algorithm's complexity scales polynomially with the number of particles, their energy and desired precision.

To make the distinction clear, while our algorithm computed scattering cross sections for simplified models in a specific perturbative regime, JLP's computes scattering amplitudes for a full field theory, not in the perturbative regime, but by discretizing space in a way similar to what is done in lattice gauge theories, and then simulate Hamiltonian time evolution. On the other hand, JLP's algorithm cannot be run on NISQ era quantum machines, requiring tens on thousands, if not more, of well connected qubits, even for the simplest amplitudes. Furthermore, a detailed implementation procedure is not provided and many are the challenges that would have to be surmounted in that pursuit. In other words, the algorithm is impressive in its intricacy and potential but its realization in practical terms remains far in the future.

Let's look into how the algorithm works. I won't dive too deep into the intricacies of the algorithm or the lattice gauge theory necessary, both of which can be found in the original paper [4], I simply want to outline the main steps and techniques involved. The first step is to put the theory on a finite qubic lattice by discretizing space only (and not spacetime as in usual lattice gauge theory). For each point in space **x**, the scalar field, a real and continuous degree of freedom, takes on some value $\phi(\mathbf{x})$, and so does the conjugate momentum $\pi(\mathbf{x})$. By canonical quantization, these two degrees of freedom are promoted to operators and the Hamiltonian for ϕ^4 theory can be written as

$$H = \sum_{x\in\Omega} a^d \left[\frac{1}{2}\pi(\mathbf{x})^2 + \frac{1}{2}(\nabla_a\phi)^2(\mathbf{x}) + \frac{1}{2}m_0^2\phi(\mathbf{x})^2 + \frac{\lambda_0}{4!}\phi(\mathbf{x})^4\right]. \tag{5.1}$$

By unitarity of quantum mechanics, the time evolution of the Hamiltonian is already a unitary operation and the next step is to find a way to implement it on a digital quantum computer. The time evolution itself is implemented by breaking down H into $H_\pi + H_\phi$, where H_π corresponds to the first term in Eq. (5.1) and H_ϕ to the latter three terms in Eq. (5.1). Since H_ϕ acts on the computational basis, which is the basis we work in, we can implement its time evolution through repeated application of $e^{-iH_\phi \delta t}$, which simply yields a phase. We can simulate $e^{-iH_\pi \delta t}$ in a similar manner, with the difference that we must first Fourier Transform to the momentum basis, and transform back to the computational basis after the evolution (we discussed how the Quantum Fourier Transform works in chapter 2).

Time evolution is one of the key ingredients of the quantum algorithm, and now that we have properly introduced it, we can sketch the full algorithmic procedure to simulate high energy scattering amplitudes. First we must define the state of the lattice by using a register of qubits to store the value of the field ϕ at each point $\mathbf{x}$. Then the main steps of the algorithm are the following (a detailed description of the algorithm is found in the original paper [4]) :

1. Prepare the ground state of the theory. Preparing an arbitrary ground state is an intractable operation even on a quantum computer, so instead we prepare the ground state of the free theory through the method of Kitaev and Webb [11], which needs a polynomial number of gates. We will later evolve the states through adiabatic evolution to states of the full interacting theory.

2. Excite wavepackets in the free theory by applying creation operators.

3. Obtain wavepackets in the new theory by performing a Hamiltonian time evolution for a time τ during which the interaction is turned on adiabatically. This is realized by implementing a kth order Suzuki-Trotter formula (see [12] and [13]) by letting

$$H = \sum_{x\in\Omega} a^d \left[\frac{1}{2}\pi(\mathbf{x})^2 + \frac{1}{2}(\nabla_a\phi)^2(\mathbf{x}) + \frac{1}{2}m_0^2(s)\phi(\mathbf{x})^2 + \frac{\lambda_0(s)}{4!}\phi(\mathbf{x})^4\right], \tag{5.2}$$

   for $0 \le s \le 1$ and $\lambda_0(0) = 0$.

4. Evolve the Hamiltonian $H(1)$ for time t, during which the actual scattering occurs, with the method outlined above for Hamiltonian time evolution.

5. Perform the time-reversed version of the adiabatic turn-on in step 3.

6. Measure via phase estimation the number operator $L^{-d}a_p^\dagger a_p$, of momentum modes in the free theory, where L is the length of the Lattice. This reduces to simulating $e^{iL^{-d}a_p^\dagger a_p t}$ for various t, and can also be implemented using a Suzuki-Trotter formula.

The outlined algorithm simulates high energy scattering amplitudes in scalar field theory with polynomial run-time, and it is applicable to both strongly and weakly interacting theories. It presents an advantage over classical algorithms especially in the non-perturbetive regime and for high precision calculation, where perturbation theory becomes too cumbersome.

This approach of simulating full scatterings with a digital quantum computer, using adiabatic turn-on and the mentioned Hamiltonian evolution, presents a fascinating research direction which promises to yield important results once large and well connected quantum machines become available in the future. It also offers many possibilities for further work, from extending this approach to more general field theories to working on the actual implementation of scatterings in a specific theory.

5.2 Other approaches

There are of course other interesting and relevant papers on quantum computing solutions for HEP, the field being relatively new but vibrant. For instance, in their 2005 paper "Simulating lattice gauge theories on a quantum computer" [3], Byrnes and Yamamoto explore the simulation of lattice gauge theories on a universal quantum computer by transcribing the Hamiltonian of the theory on the lattice in a Hamiltonian involving only Pauli spin operators, so that it can be then simulated using only one and two qubit operations. They provide examples of algorithms for $U(1)$, $SU(2)$ and $SU(3)$ lattice gauge theories up to cutoff, which were found to have polynomial complexity in the number of qubits and operations required.

If we extend the discussion to applications of quantum computing to the physical sciences as a whole, one of the most active and promising areas is quantum chemistry. Molecules, unlike scattering particles, are bound quantum systems with a finite number of degrees of freedom, making them much easier to simulate on quantum computers with a limited number of qubits. One of the main results in this field has been the development of hybrid classical-quantum systems such as the variational quantum eigensolver (VQE), which was proved to be able to compute Hamiltonian ground states, as well as certain excited states, of simple molecules efficiently. In the paper ”Computation of Molecular Spectra on a Quantum Processor with an Error-Resilient Algorithm” [14], Siddiqui *et al* describe the successful application of a VQE on a quantum machine with superconducting qubits, to extract both ground and excited states of the H_2 molecule.

5.3 Final thoughts

In our research effort, we took a successful and popular classical tool as the Parton Shower and constructed an equivalent algorithm on a quantum computer, which included features that cannot be included in the classical version, specifically keeping track of quantum interferences between final shower states. So far, we were only able to do this for a simplified field theory model and we hope one day to be able to simulate a full quantum shower for the standard model. This approach of modifying a classical algorithm into a quantum one is a popular one, which tends to yield algorithm suitable for NISQ machines, i.e. quantum computers of the near future.

The other approach, taken by JLP for instance, is to start from scratch and develop an inherently quantum algorithm. This, though probably harder to carry out, tends to result in more novel and powerful algorithms. At the same time, these algorithms often require quantum computers with a large number of interconnected qubits, with high fidelity, for practical implementation, something that will not be available in the near future.

Both directions offer fascinating research opportunities, and I hope with this work to help or encourage some readers to explore the new and exciting field of quantum computing applications, to high energy physics and beyond.

Bibliography

[1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[3] Tim Byrnes and Yoshihisa Yamamoto. Simulating lattice gauge theories on a quantum computer. *Phys. Rev. A*, 73:022328, Feb 2006.

[4] Stephen P. Jordan, Keith S. M. Lee, and John Preskill. Quantum computation of scattering in scalar quantum field theories, 2019.

[5] Stefan Höche. Introduction to parton-shower event generators, 2015.

[6] Christian W. Bauer and Frank J. Tackmann. Gaining analytic control of parton showers. *Physical Review D*, 76(11), Dec 2007.

[7] Sebastian Schmidt. *An analytic parton shower: Algorithms, implementation and validation*. PhD thesis, Hamburg U., 2012.

[8] Christian W. Bauer, Wibe A. de Jong, Benjamin Nachman, and Davide Provasoli. A quantum algorithm for high energy physics simulations. 2019.

[9] Junmou Chen, Tao Han, and Brock Tweedie. Electroweak splitting functions and high energy showering. *Journal of High Energy Physics*, 2017(11), Nov 2017.

[10] Christian W. Bauer, Davide Provasoli, and Bryan R. Webber. Standard model fragmentation functions at very high energies. *JHEP*, 11:030, 2018.

[11] Alexei Kitaev and William A. Webb. Wavefunction preparation and resampling using a quantum computer, 2009.

[12] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, Dec 2006.

[13] M. Suzuki. Fractal decomposition of exponential operators with applications to many-body theories and monte carlo simulations. *Physics Letters A*, 146:319–323, 1990.

[14] J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi. Computation of molecular spectra on a quantum processor with an error-resilient algorithm. *Phys. Rev. X*, 8:011021, Feb 2018.

[15] IBM Research. Qiskit, an open-source computing framework, 2018.

[16] Lov K. Grover. A fast quantum mechanical algorithm for database search. pages 212–219, 1996.

[17] Peter W. Shor. Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Sci. Statist. Comput.*, 26:1484, 1997.

[18] Carlos Blanco, J. Patrick Harding, and Dan Hooper. Novel Gamma-Ray Signatures of PeV-Scale Dark Matter. *JCAP*, 1804(04):060, 2018.

[19] Leandro G. Almeida, Stephen D. Ellis, Christopher Lee, George Sterman, Ilmo Sung, and Jonathan R. Walsh. Comparing and counting logs in direct and effective methods of QCD resummation. *JHEP*, 04:174, 2014.

[20] George F. Sterman and Steven Weinberg. Jets from Quantum Chromodynamics. *Phys. Rev. Lett.*, 39:1436, 1977.

[21] Joshua Isaacson and Stefan Prestel. Stochastically sampling color configurations. *Phys. Rev.*, D99(1):014021, 2019.

[22] René Ángeles Martínez, Matthew De Angelis, Jeffrey R. Forshaw, Simon Plätzer, and Michael H. Seymour. Soft gluon evolution and non-global logarithms. *JHEP*, 05:044, 2018.

[23] Simon Platzer and Malin Sjodahl. Subleading N_c improved Parton Showers. *JHEP*, 07:042, 2012.

[24] Simon Plätzer, Malin Sjodahl, and Johan Thorén. Color matrix element corrections for parton showers. *JHEP*, 11:009, 2018.

[25] Zoltán Nagy and Davison E. Soper. Parton showers with more exact color evolution. *Phys. Rev.*, D99(5):054009, 2019.

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

[27] Peter Richardson. Spin correlations in Monte Carlo simulations. *JHEP*, 11:029, 2001.

[28] I. G. Knowles. A Linear Algorithm for Calculating Spin Correlations in Hadronic Collisions. *Comput. Phys. Commun.*, 58:271–284, 1990.

[29] John C. Collins. Spin Correlations in Monte Carlo Event Generators. *Nucl. Phys.*, B304:794–804, 1988.

[30] I. G. Knowles. Spin Correlations in Parton - Parton Scattering. *Nucl. Phys.*, B310:571–588, 1988.

[31] Cornelius Hempel, Christine Maier, Jonathan Romero, Jarrod McClean, Thomas Monz, Heng Shen, Petar Jurcevic, Ben P. Lanyon, Peter Love, Ryan Babbush, Alán Aspuru-Guzik, Rainer Blatt, and Christian F. Roos. Quantum chemistry calculations on a trapped-ion quantum simulator. *Phys. Rev. X*, 8:031022, Jul 2018.

[32] John C. Collins, Davison E. Soper, and George F. Sterman. Factorization of Hard Processes in QCD. *Adv. Ser. Direct. High Energy Phys.*, 5:1–91, 1989.

[33] Zoltan Nagy and Davison E. Soper. A parton shower based on factorization of the quantum density matrix. *JHEP*, 06:097, 2014.

[34] Esteban A. Martinez, Christine A. Muschik, Philipp Schindler, Daniel Nigg, Alexander Erhard, Markus Heyl, Philipp Hauke, Marcello Dalmonte, Thomas Monz, Peter Zoller, and Rainer Blatt. Real-time dynamics of lattice gauge theories with a few-qubit quantum computer. *Nature*, 534:516–519, 2016.

[35] I. M. Georgescu, S. Ashhab, and Franco Nori. Quantum Simulation. *Rev. Mod. Phys.*, 86:153, 2014.

[36] Stephen P. Jordan, Keith S. M. Lee, and John Preskill. Quantum Algorithms for Quantum Field Theories. *Science*, 336:1130–1133, 2012.

[37] Walter T. Giele, David A. Kosower, and Peter Z. Skands. A simple shower and matching algorithm. *Phys. Rev.*, D78:014026, 2008.

[38] Stefan Gieseke David Grellscheid Stefan Hoche Hendrik Hoeth Frank Krauss Leif Lonnblad Emily Nurse Peter Richardson Steffen Schumann Michael H. Seymour Torbjorn Sjostrand Peter Skands Bryan Webber Andy Buckley, Jonathan Butterworth. General-purpose event generators for LHC physics. *Phys. Rept.*, 504:145–233, 2011.

[39] Stefan Höche and Stefan Prestel. The midpoint between dipole and parton showers. *Eur. Phys. J.*, C75(9):461, 2015.

[40] T. Gleisberg, Stefan. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert, and J. Winter. Event generation with SHERPA 1.1. *JHEP*, 02:007, 2009.

[41] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. PYTHIA 6.4 Physics and Manual. *JHEP*, 05:026, 2006.

[42] G. Marchesini and B. R. Webber. Simulation of QCD Jets Including Soft Gluon Interference. *Nucl. Phys.*, B238:1–29, 1984.

[43] Kevin Marshall, Raphael Pooser, George Siopsis, and Christian Weedbrook. Quantum simulation of quantum field theory using continuous variables. *Phys. Rev.*, A92(6):063825, 2015.