An Optimizing Pulse Sequence Compiler for NMR QIP

by

Carlos A. Pérez Delgado

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Combinatorics and Optimization

Waterloo, Ontario, Canada, 2003

Abstract

Quantum information processing is a multi-disciplinary science involving physics, mathematics, computer science, and even quantum chemistry. It is centred around the idea of manipulating physical systems at the quantum level, either for simulation of physical systems, or numerical computation. Although it has been known for almost a decade that a quantum computer would enable the solution of problems deemed infeasible classically, constructing one has been beyond today's capabilities. In this work we explore one proposed implementation of a quantum computer: Nuclear Magnetic Resonance (NMR) spectroscopy. We also develop a numerical software tool, a pulse sequence compiler, for use in the implementation of quantum computer programs on an NMR quantum computer. Our pulse sequence compiler takes as input the specifications of the molecule used as a quantum register, the desired quantum gate, and experimental data on the actual effects of RF pulses on a sample of the molecule, and outputs an optimum set of pre and post 'virtual' gates that minimize the error induced.

Acknowledgements

I am extremely grateful to both my supervisor in mathematics, Michele Mosca, and co-supervisor in physics, Raymond Laflamme, for their incredible support and (uncountably) infinite patience.

I wish to thank the University of Waterloo for the economic support that made my Master's studies —and this thesis— possible.

I also wish to thank those who have made my stay at Waterloo most pleasant: Claudie, Theo, Marcela, Amandine, Eduardo and Bere, Francisco, Paquito, Alma, Arturo, Enrique, and the whole CAWAMA gang.

Last but not least, I wish to give my deepest thanks to the first, and very best, teachers I have ever had: my parents.

Contents

List of Figures

Preface

We set out to solve a very specific optimization problem given in the implementation of quantum algorithms on an NMR quantum computer.

Given a desired quantum gate, described by a unitary complex-valued matrix U, and the actual gate that the NMR spectrometer performs U', we wish to find pre and post processing 'virtual' gates E_{pre} and E_{post} (that must have a very specific form as discussed in chapter 4) that can applied before and after the spectrometer's gate, so as to minimize:

$$D(U, E_{post} U' E_{pre})$$

where D is the Manhattan or city-block distance metric, defined by:

$$D(A, B) = \sum_{i,j} |A_{i,j} - B_{i,j}|.$$

Unfortunately, the problem is actually harder: the matrix U is not actually given, rather it must be calculated. In order to calculate U it is necessary to simulate the quantum evolution of the different nuclei in the molecule used as a quantum register when it is submitted to the radio frequency pulses that constitute the quantum gates.

There are several hurdles to be overcome, in order to do this. First of all, the state space of the register molecule grows exponentially in the number of distinguishable spins. This is of course, one of the reasons why quantum computation is desirable: simulating quantum systems on a classical computer is hard. We describe a method to do this simulation, in a decently efficient matter, taking advantage of the specifics of NMR spectroscopy.

The problem described is hard as it is, but there is more. Although the mathematics to describe the evolution of a quantum system with a time-independent Hamiltonian have been studied since almost the beginning of the last century, time-*dependent* Hamiltonians are harder to work with. The overall Hamiltonian is, in our case, heavily time-dependent since it depends on the various RF pulses, that vary with time.

In order to cope with this it is necessary to break up the time evolution into discrete intervals, find the step-wise Hamiltonians of each, and then conjugate them all to achieve overall evolution.

In short, in order to possibly achieve the task at hand, the total evolution of the whole system is split into the step-wise evolutions of the sub-components: *Divide et vinci.*

Although, at first, this may seem as a daunting task, with the right tools it is actually achievable. Unfortunately such tools are found in such distinct areas as optimization, linear algebra, computer science, quantum physics, and molecular chemistry.

In this work we guide the reader, in the briefest possible way, through all the basic principles that allowed us to understand, and finally give solution, to the problem posed.

We begin with an introduction to QIP. We then continue to introduce the basic notions used in QIP, starting with the basic notions of the disciplines on which it is founded: computability theory, and quantum physics. Chapter two can be used independently as a decent introduction to QIP for anyone with a standard mathematical background.

In chapter three we give a comprehensive introduction to NMR QIP. This chapter relies on the information of chapter two, but can be used independently as introduction to NMR for people with an understanding of (theoretical) QIP.

Finally, in chapter five we discuss our own —humble— contribution to the discipline: a pulse sequence compiler for NMR QIP. The discussion in this chapter relies heavily on the notation, and information, introduced in the preceding chapters.

Chapter 1

Introduction

Quantum Information Processing (QIP for short) is a relatively new discipline with profound and exciting implications. It brings together, and closes the gaps between, some of the most important scientific disciplines developed in the twentieth century, which had mostly, until now, advanced independently.

On one side there is information theory. Originally developed by Claude Shannon [CT91] for the study of communication complexity, it has grown to encompass many other areas. Shannon's is the defacto definition of entropy as 'lack of information'. As its name implies, information theory attempts to categorically study what information 'is', and how to quantify it.

However, it is interesting to note, that information theory, by itself, was unable to answer questions it, itself, had posed. For example, the entropy of an object is left undefined. Shannon's entropy applies only to a set, or ensemble, of objects. Information theory would need the hand of another discipline that had been evolving in parallel, in order to attack the deeper questions in its realm.

Fortunately, another discipline was also being developed at the time: (theoretical) computer science. Born originally as a discipline of mathematics, from the minds of mathematicians, computability theory is the answer to Hilbert's tenth problem (his famous Entscheidungsproblem). Based on the idea of finite computation, Church, Turing[Tur37], Post [Pos44], among others gave one of the last century's most important results (one that complemented Kurt Gödel's earlier one [G̈31]): "*not* all functions can be calculated".

Obviously, computability theory needs information theory. Computation is, after all, the "*processing of information*". It was quite surprising, however, to see how computation theory was able to complement and extend information theory. By using the Alan Turing's model of computation, the so-called Turing Machine, Kolmogorov, Gregory Chaitin, among others were able to give a definition of entropy that applied to individual objects. In the process of doing so, also giving (at last after several hundreds of years of conundrum) a reasonable, formal definition of the word *random* (see [LV97]).

Kolmogorov Complexity, also known as algorithmic complexity theory, has profound implications in all areas of science, showing just how powerful the Turing Machine is, as a model for computation.

As powerful as it may be, we must ask: is the Turing machine really a *valid* model for computation? To understand this question we must first realize that Turing's model of computation, as well as Church's and Post's and all others of that era, were *purely mathematical* models.

However, computation cannot be done in the vacuum. Computation is a physical process, subject to the laws of physics. Two questions then arise. First, does the Turing machine (or any other model of computation we might be interested in) obey the laws of physics? And second, if it does, does it fully take advantage of them?

In answering these two questions it makes sense to use the best, most up to date, and successful theory of physics at our disposal: quantum theory.

Although quantum theory is regarded as "modern physics" it predates both information theory, and computation theory. It began with the works of Max Planck and Albert Einstein attempting to explain the nineteenth century's physics most important conundrums, such as the photo-electric effect, black-body radiation, and the so-called ultraviolet catastrophe.

It led to Schrödinger's, Heisenberg's and Dirac's formulations of quantum mechanics, and later to quantum field theory developed by Feynman, Schwinger and others.

Modern quantum physics is, without a doubt, man's most amazing achievement. It studies the very nature of matter, and energy. It explains why atoms don't collapse, and how particles can be waves, and waves be particles.

As to computation and physics, the idea that computation is a physical process is usually first attributed to Landauer. The first truly quantum mechanical model of computation is due to David Deutsch [Deu85]. He was one of the first to develop a quantum-mechanical model of computation: a model based on Turing's Machine, but extended using capabilities allowed by the laws of quantum mechanics. Other early proposals are attributed to Feynman [Fey82], who noted that classical simulation of quantum mechanical systems is inefficient, and Benioff [Ben80a] who gave a first aproximation to what would later be called the Quantum Turing Machine.

It wasn't until 1994, though, that it was realized by Peter Shor [Sho97] that a quantum computer could be used to solve two very important problems that are deemed infeasible classically: factoring, and the discrete logarithm problem.

The difficulty of solving these problems classically led to the standardization and proliferation of cryptographic protocols based on these problems. RSA, Diffie-Hellman, even the much touted Elliptic Curve cryptosystem all depend on the difficulty of these problems (see [MOV96] or [Sti95] for a comprehensive discussion of modern day cryptography). Shor's result implies that anyone with access to a quantum computer could easily compromise today's e-commerce infrastructure. This simple fact has led to an exponential growth in research activity in the area of QIP during the last eight years.

There are, however, deeper more important questions in all the realms of knowledge that we hope QIP will help answer. The quantum Turing machine is might *not* be the ultimate model of computation: like the Turing machine, it is still making unproven assumptions about Nature. The most important one, it posits the fact that a quantum system used for computation must have a discrete spectra. However, as far as we know, systems with continuous spectra (might) do exist (i.e. a free moving particle). Whether or not this last statement is really true, or it is itself a wrong assumption is a question being attacked by the discipline of *quantum gravity.*

How much is a quantum state $|\phi\rangle$, or ρ, really a description of the state, and how much is it dependent on our relationship with the state? QIP, tries to answer these questions by characterizing information of quantum systems.

Decoherence, and, in general, the transition from quantum to classical is best explained as an irreversible loss of *information*. Again, this is the realm of QIP.

Many other examples abound. However, it should be clear by now that QIP has an immense applicability to the areas of physics, mathematics, and computation; perhaps even more.

In order to pursue these avenues of thought it is necessary to perform experimentation on a quantum computer. The abilities of this quantum computer may even be humble. As opposed to the industry and government agencies that are mostly interested in quantum computers with several thousand *qubits* (QUantum BITs), researchers in QIP need only a few qubits in order to experimentally test important foundational theoretical results. For example Deutsch's algorithm requires only two qubits to test.

For these purposes, todays implementation of quantum computers using NMR (Nuclear Magnetic Resonance) are amply sufficient. At Waterloo a 7 qubit quantum computer using trans-Crotonic Acid is currently being used.

Controlling a quantum computer is complicated business. Classical computers are used to design and implement the quantum algorithms on the NMR spectrometer. Classical computers are then used to read and interpret the results. The software involved must be efficient, and correct.

Here we present a module used in the implementation segment of a quantum algorithm. We introduce a general purpose optimizing pulse sequence compiler for use in liquid state NMR.

This compiler takes as input the desired algorithm (broken up into ‘gates’), and outputs the actual RF-pulses (the physical control mechanism in NMR) to be used. It does this in such a way as to minimize introduced errors during gate operations. In other words, it optimizes the efficiency of the gates involved.

In the following two chapters we introduce the necessary nomenclature necessary to understand and appreciate this work. Chapter two gives a general perspective of QIP, introducing notions of computer science and physics relevant to the discussion. Chapter three gives an introduction to QIP using NMR. Finally, in chapter four we give a complete discussion of our pulse-sequence compiler.

The compiler source, together with manuals, and diagrams can also be downloaded at http://www.math.uwaterloo.ca/∼caperezd/research/pulse_compiler.

Chapter 2

Preliminaries

In this chapter we will present mathematical preliminaries that are absolutely fundamental in order to discuss NMR QIP, and the pulse sequence compiler presented here.

QIP is a highly multi-disciplinary area. As such it is often the case that people studying in the area come from very distinct backgrounds.

Hence, we deem necessary to give a —very quick— introduction to several of the areas that comprise QIP. Some basic background that is needed later on, but is not presented (i.e. it is expected from the reader) is some advanced linear algebra, and basic Fourier analysis.

We begin in the next section with the theory of computation. In section 2.2 we will introduce the basic notions of quantum mechanics, and finally in section 2.3 we introduce quantum computation. This chapter serves not only as an introduction to these areas, but also to introduce notation, and ideas that will be necessary in the discussion to follow.

Another important goal of this chapter is to bridge the gap between "physicist's parlance" and the more mathematical perspective.

We leave the specifics of NMR QIP for the next chapter.

2.1 Computability Theory

The main concern of the discipline of *computer science* is to answer the question: what can, and what can't, *in principle*, be calculated/solved/computed?

It is of unrecognized importance to stress the phrase '*in principle*'. Computer science concerns itself with what can be done, and cannot, by any methods, be it machine, man, or an as yet unimagined source of computing power, ever.

It is this fact that makes a computer science such a fundamental mathematical discipline. It tells us which mathematical functions are actually solvable, and which aren't. Computability theory traces its origins to the works of Alonso Church, Emil Post, and Alan Turing. Perhaps it is correct to go back even further to Kurt Gödel. The study of algorithms is even older, dating back to the ancient Greeks.

Computer theoretic answers to the question "what can be computed?" begin with mathematical models of computation. Several such models exist: self-calling functions with base cases: recursive functions; abstract functions that can be applied to each other: λ-calculus; and perhaps the most commonly used today: a finite state machine with an (infinite length) input tape, known as the Turing machine.

In our main discourse we will opt to use a more recent model called the circuit model. This is a very intuitive model, and has the distinct advantage for us that it is easy to carry into the quantum world of computation. Unfortunately it is not a complete model of computation, as we shall see below. For our purposes it suffices to know that there do exist complete models of computation (those stated above).

2.1.1 Families of Acyclic Circuits

Simply put an algorithm is a succinct method for performing a certain function. Every algorithm represents a function. On the other hand, perhaps the most important result of computability theory is that not all functions have algorithms associated with them. In other words, there is simply now way to evaluate such functions. Hence, they are called *uncomputable*.

Of those functions that are computable, not all can be computed with a reasonable amount of resources such as time (duration of the algorithm), and space (amount of workspace e.g. memory in a computer).

Formalizing what exactly 'reasonable amount of resources' exactly means is the focus of *computational complexity theory*[1]. In order to do that we must introduce formal notions of what is meant by algorithm, input, and output.

For simplicity we restrict ourselves to binary inputs and outputs. Hence, the *alphabet* of our algorithms will be always the set $\Sigma = \{0, 1\}$. A string with this alphabet is the concatenation of zero or more symbols 0 or 1. The set of all such strings is denoted by Σ^*. A subset of Σ^* is called simply a *language*.

It is important to note that any function that maps integers to integers, or any finite-precision field onto another finite-precision one can be represented as simply a function over the set of binary strings $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$.

[1]In this work, whenever we mention *complexity theory*, we are referring to *computational complexity theory*.

An algorithm for a function f is a method that will, given the input string representing x in binary will give the binary string for $f(x)$ in return.

In complexity theory it is often more convenient to talk about decision problems, instead of functions.

The decision problem of a language (a subset of all binary strings) $L \subseteq \Sigma^*$, is to, given an input x, decide whether or not x belongs to the language $x \in L$or not. For example, if the language consists of all (binary strings representing) prime numbers, the decision problem is to, given (the binary string representation of) an integer, tell whether or not it is prime.

A formal way to characterize algorithms, both for functions and decision problems, is circuits.

Informally, a mathematical circuit is much like a physical one. Every input symbol is seen as a bit of information travelling through the network. Input goes in at the left end. It then passes through a series of gates, each of which acts only on one or two bits. The gates somehow manipulate the information, and the correct answers comes out at the right end. See figure 2.1.1 for an example.

A formal definition follows.

Definition 1 (Acyclic Circuit). *An acyclic circuit is a directed, acyclic graph $G = (V, E)$, where each vertex is of the following form:*

Inputs *Have in-degree zero. We label inputs as $x_1, x_2, \ldots$, or as one of two constants* 0 *or* 1.

Gates *can be one of the following: not-gates ($\neg$) with in-degree 1, or-gates ($\vee$) with in-degree 2, and-gates ($\wedge$) with in-degree 2.*

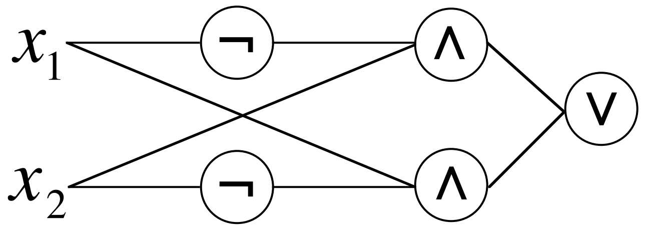


Figure 2.1: Parity Circuit

The vertexes with out-degree zero are called the outputs of the circuit.

It might seem that we lose generality since, we do not allow arbitrary bit operations, instead allowing only three operations. No generality is lost, however, since the set $(\neg, \vee, \wedge)$ is a *universal set* of gates. In other words, any Boolean function can be written in terms of the previous set of gates. Another universal set of gates is the $NAND$ gate by itself (the NOR gate by itself is also universal).

A circuit can calculate functions of its inputs. For example the circuit in figure 2.1.1 calculates the parity of its two inputs. Also, circuits can be seen to decide languages, by taking output zero to mean reject, and different from zero to mean accept. For example the same circuit recognizes the language $x_1x_2 : x_1 \neq x_2$.

Instead of allowing arbitrary input, we could posit that only the string of all zeroes is allowed as leftmost input. The 'real' input is then created by applying a NOT gate to all those bits whose value we wish initialized to one. Both definitions are equivalent. We will prefer the second one, since it carries over better to the quantum realm.

A limitation of circuits is that they can only work on inputs of up to a certain size (two bits, in the previous examples).

To get over this hurdle we define the family of acyclic circuits. In brief, a family of circuits is a set of circuits that all solve the same problem, but for different sizes of input. For example, we could have a family of circuits that squares its input. There is then a circuit in the family for inputs of one bit, another inputs of two bits, another inputs of three bits, and so on.

Definition 2 (Uniform Family of acyclic circuits). *A uniform family of acyclic circuits $C = (C_0, C_1, C_2, \ldots)$, is such that there exists a recursive procedure (a TM) that given n will output C_n in polynomial time*[2]*. Furthermore each C_i has exactly i inputs.*

Notice in the definition that we require circuits to be constructed using a recursive procedure. We need another model of computation to build the circuit. This makes the circuit model incomplete, as it cannot, by itself, define computable functions.

We can define complexity classes solely in terms of circuits[3].

Definition 3 (P). *A language $\mathcal{L}$ is in **P** if and only if there exists a family of uniform acyclic circuits such that the size of the circuit is bounded above by a fixed polynomial P evaluated in n. That is $|C_n| \leq P(n)$, and the circuit C_n decides whether a string of size n is in $\mathcal{L}$. Formally, for any string x such that $|x| = n$, we have that $C_n(x) = 1$ if and only if $x \in \mathcal{L}$*

[2]It could be argued that it is more appropriate to require logarithmic space in the size of the input; however, for our purposes it is sufficient to expect polynomial time.

[3]This is one of the places where we take an unorthodox approach. Complexity classes are usually defined in terms of Turing Machines. However, both definitions (in the classical setting) can be shown to be equivalent.

Definition 4 (EXP). *A language $\mathcal{L}$ is in **EXP** if and only if there exists a family of uniform acyclic circuits such that the size of the circuit is bounded above by $2^{P(n)}$, where P is a fixed polynomial P. That is $|C_n| \leq 2^{P(n)}$, and the circuit C_n decides whether a string of size n is in $\mathcal{L}$.*

We usually consider problems that are known to have polynomial size circuits to be tractable, while those that are only known to have exponential size circuits to be intractable; even though both kinds of problems are definitely decidable.

In order to define further complexity classes it is necessary to introduce the concept of a random or non-deterministic gate, also known as a coin-flip.

Definition 5 (Non-deterministic Gates). *A non-deterministic gate $¢$ is a gate with no inputs (in-degree zero). A circuit with non-deterministic gates is called a non-deterministic circuit. The set of non-deterministic gates $\{¢_0, ¢_1, \ldots\}$ used in a circuit C is called the non-deterministic base of C, denoted by $\$(C)$. The size of the non-deterministic base of a circuit, $|\$(C)|$ is the level of non-determinism of C, and we denote it by n_C, or simply n when there is no ambiguity. A valuating function appropriate to a circuit C is a complete function $f : \$(C) \to \{0, 1\}$.*

The output of a non-deterministic circuit depends not only on its inputs but also on the valuating function, hence it is not (entirely) correct to say $y = C(x)$, for a non-deterministic circuit, but rather we should say $y = C(x, f)$, where f is a valuating function appropriate for C.

Very powerful complexity classes can be defined in terms of non-determinism.

Definition 6 (NP). *A language $\mathcal{L}$ is in **NP** if and only if there exists a family of uniform non-deterministic acyclic circuits such that the size of the circuit is bounded* *above by a fixed polynomial P evaluated in n. That is $|C_n| \leq P(n)$. Furthermore, for each x such that $|x| = n$, the circuit C_n decides whether x is in $\mathcal{L}$ in the following sense: if $x \notin \mathcal{L}$ then $C_n(x, f) = 0$ for all valuating functions f appropriate for C. If $x \in \mathcal{L}$ then $C_n(x, f) = 1$, for at least one valuating function f appropriate for C.*

The reader can easily see that **NP** does not characterize any reasonable model of computation. This is because in order to carry out the calculations of the aforementioned type of circuits it is necessary to check all possible valuating functions appropriate for the circuit in question. The number of valuating functions N appropriate for a circuit C is directly related to its degree of non-determinism, succinctly: $N = 2^n$ (where n is the level of non-determinism of C). A simple result follows:

Theorem 1. $\boldsymbol{P} \subseteq \boldsymbol{NP} \subseteq \boldsymbol{EXP}$.

The first inclusion comes from the fact that deterministic circuits are simply a special case of non-deterministic ones (with level of non-determinism zero). The second comes from the fact that we can easily simulate a non-deterministic circuit of size p and non-determinism n with a deterministic circuit of size $2^n p$.

The importance of **NP** comes from the fact that it characterizes a set of problems very well.

A well-known problem is $SATISFIABILITY$: given a Boolean formula in conjunctive normal form (that is of the form $\bigwedge_{i=1}^{n}(\bigvee_{j=1}^{m(j)} v_{ij})$ where each v_{ij} is either a variable or a negated variable) decide whether there exists a Boolean assignment of each variable that makes the formula true.

By our definition of **NP**, it is easy to see that $SATISFIABILITY \in \mathbf{NP}$. Also, it is not too difficult to see that any algorithm that solves $SATISFIABILITY$

would also solve *any other* problem in **NP**. We say that *SATISFIABILITY* is **NP**-Complete. This result, proved originally in the Turing-Machine model of computation, is of profound importance, and is known as Cook's Theorem. In the circuit-model it is almost a direct consequence of the definition.

Two more important classes follows:

Definition 7 (PP). *A language $\mathcal{L}$ is in **PP** if and only if there exists a uniform family of non-deterministic acyclic circuits such that the size of the circuit is bounded above by a fixed polynomial P evaluated in n. That is $|C_n| \leq P(n)$. Furthermore, for each x such that $|x| = n$, the circuit C_n decides whether x is in $\mathcal{L}$ in the following sense: $x \in \mathcal{L}$ if and only if $C_n(x, f) = 1$ for more than half the valuating functions f appropriate for C. We say that the circuit decides $\mathcal{L}$ by simple majority.*

Definition 8 (BPP). *A language $\mathcal{L}$ is in **BPP** if and only if there exists a family of uniform non-deterministic acyclic circuits such that the size of the circuit is bounded above by a fixed polynomial P evaluated in n. That is $|C_n| \leq P(n)$. Furthermore, for each x such that $|x| = n$, the circuit C_n decides whether x is in $\mathcal{L}$ in the following sense: if $x \in \mathcal{L}$ then $C_n(x, f) = 1$ for at least two thirds of all valuating functions f appropriate for C. If $x \notin \mathcal{L}$ then $C_n(x, f) = 0$, for at least two thirds of all valuating functions f appropriate for C. We say that the circuit decides $\mathcal{L}$ by clear majority.*

The importance of the previous class follows.

In an actual implementation of a circuit we could substitute all non-deterministic gates with a random source, one that outputs either a 1 or a 0 with equal probability.

By running the circuit several times with the same input x we can figure whether $x \in \mathcal{L}$ with increasingly good probability.

Using basic probability it is not too hard to show that if the probability of success of the algorithm is polynomially-bounded away from 1/2 (that is, for input x, the probability of success is at least $1/2 + \frac{1}{poly(|x|)}$) —as is the case for algorithms for problems in **BPP**— then by repeating the algorithm a polynomial (in the size of the input) number of times the probability of success can be made *exponentially* close to 1. On the other hand, if the probability of success of the original algorithm is allowed to be exponentially close to 1/2 then an *exponential* number of trials is necessary to bring the total probability of success up close to 1. This result is due to Chernoff, and is known as the *Chernoff bound.* It is the reason why **BPP** captures our notion of the set of problems efficiently solvable by random algorithms, rather than **PP**.

This concludes our discussion of classical complexity theory. Much remains to be said, but this will have to do. Now we turn our attention to the quantum realm.

2.2 Brief Overview of Quantum Mechanics

Here we will give a very brief introduction to some of the key concepts of quantum mechanics. We will begin with a physical experiment that shows the principles that allow us to use quantum systems for efficient computation.

2.2.1 The Stern-Gerlach Experiment

We now describe a classic experiment in quantum mechanics. This experiment is very much referred to in textbooks on quantum mechanics as it serves to illustrate the most fundamental concepts of quantum mechanics.

The next discussion takes the same approach given in [Sak94] and [NC00].

The Stern-Gerlach experiment was proposed be Stern in 1921 and implemented by himself and W. Gerlach in 1922.

Though the original was carried about using silver atoms, the same experiment can be done using hydrogen [NC00] and the discussion thus is simpler.

Recall that a hydrogen atom consists of a single proton and an electron 'orbiting' around it. Both the proton and the electron have an electric charge (positive and negative respectively).

There are two magnetic fields associated with an atom in general. First the movement of the electron around the proton creates a magnetic field. This field is completely described classically as one created by the movement of one charge around another. However, in the case of the hydrogen atom, this movement is completely symmetric (the orbital path describes a sphere around the proton), and thus this field cancels itself out, and has a value of 0.

There is, however, another magnetic field called *magnetic dipole moment.* This field is described, in quantum mechanics, as created by the *intrinsic* movement of the electron, called *spin.*

The magnetic dipole moment μ of the atom is a three dimensional vector that is directly proportional to the electron spin S. Say we want to measure the z-

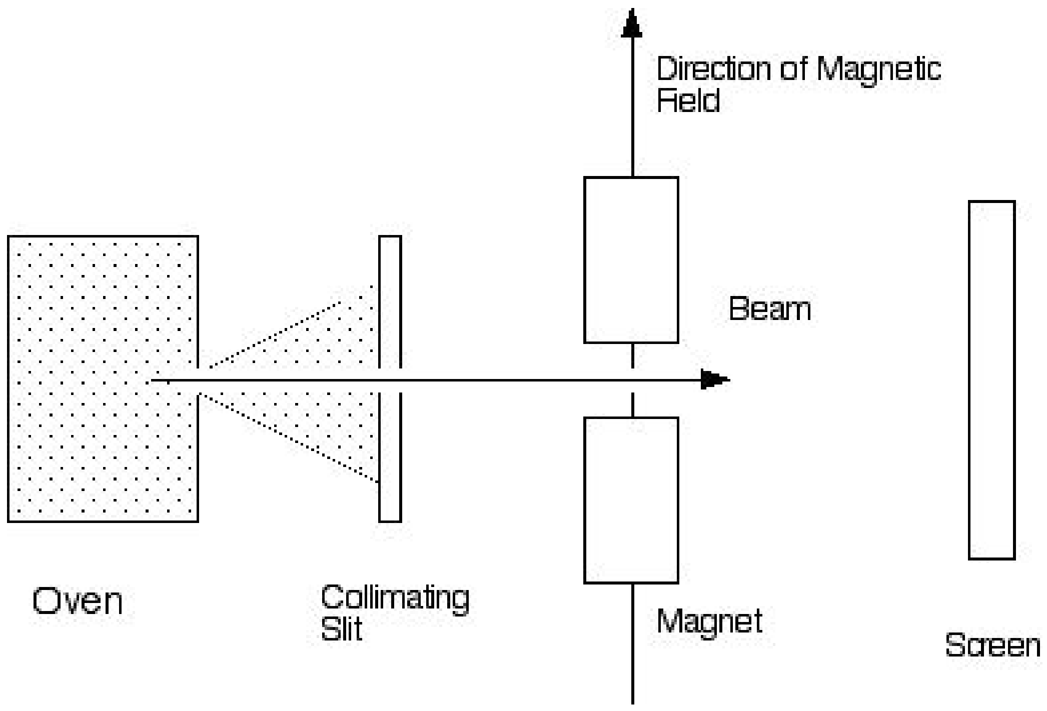


Figure 2.2: The Stern-Gerlach Experiment

component of μ (and thus measure the z-component of the spin S_z).

Then we can submit the atom(s) to a magnetic field which creates a force on this direction. Much like two equal polarity magnets repel and different polarity magnets attract, an atom with a positive z-component of μ will feel an upward force while an atom with a negative value will feel a downward force.

The schematic view of this experiment is shown in figure 2.2.1. We call this apparatus SG_z since it is meant to measure the z-component of μ and thus of the spin S.

Hydrogen atoms are heated up in an oven, from where they escape. We interpose a collimating slit (an aperture that allows only a stream in one direction) in order to

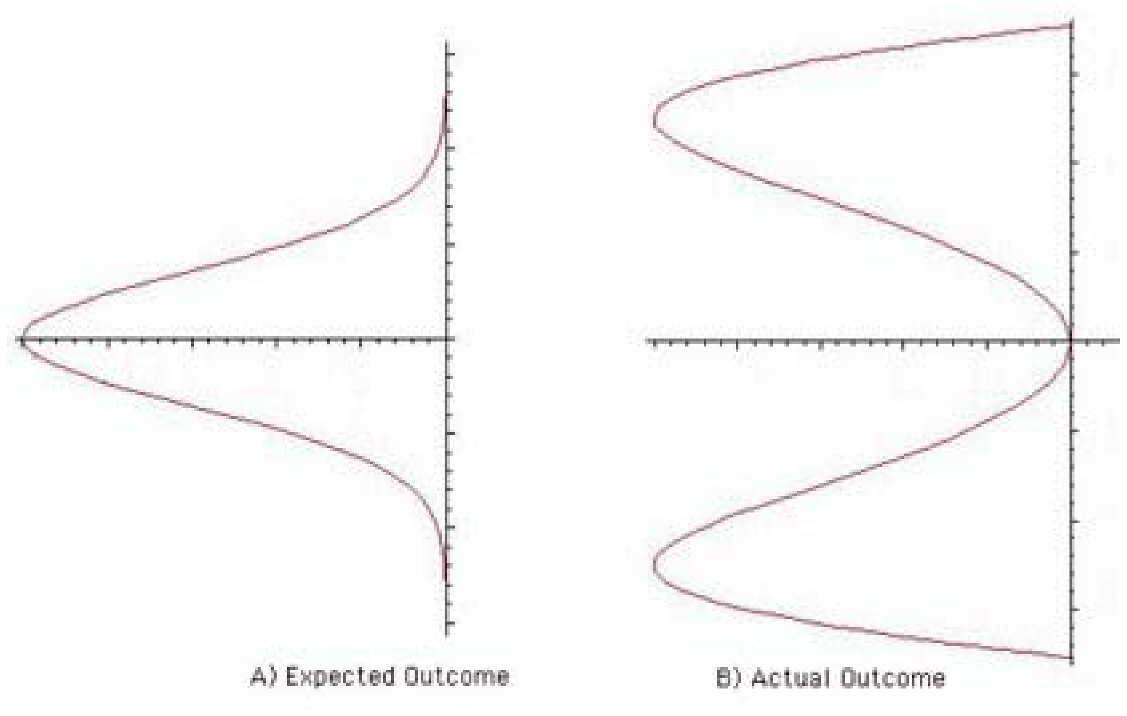


Figure 2.3: Expected and Actual Outcomes

guarantee a nice parallel stream of atoms. Then the hydrogen atoms are subjected to the magnetic field.

It is interesting to see what happens when the atoms pass through the magnetic field: any atoms with a positive μ_z value would have their trajectory slightly tilted upward, while those with negative μ_z value would have their trajectory tilted downward, in both cases the tilt is proportional to the magnitude of μ_z. Since the atoms leaving the oven do so in a very random fashion, we expect their electron spins to be fairly random. Thus, in accordance to probabilistic laws, we expect to have atoms distributed randomly in the z direction like in figure 2.2.1 a.

However, what is *actually* obtained is more like figure 2.2.1 b. How can we explain this? It is impossible to do so with classical probability theory. Rather we must accept the quantum mechanical concept that electron spin is not continuous in value, rather it is discrete or *quantized*, in this case it admits only two possible values: z-spin up S_z+, and z-spin down S_z-.

Let us make now a more interesting experiment. Suppose we were to setup the Stern-Gerlach apparatus in a way as to block all atoms with spin down. What if we were to subject the remaining particles to another —exactly the same— Stern-Gerlach apparatus, namely another magnetic field. Since we eliminated all particles with spin down coming out of the first apparatus we would expect to see all particles coming out of the second apparatus with spin up only. In this case actual results reflect the expected outcome.

Now what happens if we tilt the second Stern-Gerlach apparatus 90 degrees, so instead of measuring μ_z we measure μ_x? Since initially all particles have random μ and the first Stern-Gerlach apparatus measures only the z-component and —supposedly— leaves the x-component unaltered, we would expect to have exactly half the atoms to have S_x+ and half to have S_x-. Again the actual outcomes of the experiment reflect expected outcomes, and there are no surprises.

Now let us suppose we block all atoms with S_x- from the second apparatus, and submit the remaining atoms to a third Stern-Gerlach apparatus, this time again in the z-direction. Recall that we block all atoms having S_z- coming out of the first apparatus, and all atoms having S_x- coming out of the second, so we expect all atoms to have only S_x+ and S_z+ values, and thus our third Stern-Gerlach apparatus should detect only positive S_z values.

However, again Nature is not quite predictable. The real outcome of the experiment is to have only half the atoms with S_z+ and the rest with S_z-. Classically, there seems to be no reasonable explanation.

Quantum mechanics explains this with the *uncertainty principle*. In this case it states simply that we cannot know both the x-component and the z-component (or y-component etc.) of a given atom *at the same time*. Notice that from the moment the atoms leave the first apparatus and up to the moment they enter the second we know their S_z value. Once they enter the second apparatus and we measure the S_x value the S_z value is irrevocably lost.

The pairs of values $\{S_z+, S_z-\}$ and $\{S_x+, S_x-\}$ are called *conjugate bases*. The reason is that a quantum system (a hydrogen atom here) with S_z+ or S_z- value can be deterministically measured if this is done with respect to the *basis* S_z+, S_z- (with SG_z apparatus for example) but will behave completely randomly if measured with respect to the basis S_x+, S_x- taking any of the two values with one half probability. The converse is also true.

This is the fundamental concept behind quantum cryptography[4].

We now discuss a more mathematical explanation of the Stern-Gerlach experiment.

[4]In quantum cryptography conjugate basis of photon polarization, as opposed to nuclear spin, is used [BB84, BBG$^+$90, MBG93]. For a comprehensive bibliography of quantum cryptography see [Bra93]. Claude Crépeau keeps a an updated version of this paper at: http://www.cs.mcgill.ca/~crepeau/CRYPTO/Biblio-QC.html

2.2.2 A Quantum world

In the Schrödinger picture of quantum mechanics, every quantum system, like nuclear spin, is represented by a state vector, or ket, that evolves in time. An arbitrary state is denoted by $|\psi\rangle$ (pronounced 'ket psi').

Possible states include $|S_z+\rangle$, $|S_z-\rangle$, and $|S_x+\rangle$.

A (projective) measurement is represented by the conjugate transpose of a ket, called bra. An arbitrary measurement is denoted as $\langle\psi|$ (pronounced 'bra psi'). A projective measurement is, for example, when we setup the Stern-Gerlach experiment to allow only atoms with S_z+ magnetization to pass through.

In this example, the projective measurement is $\langle S_z + |$. The absolute value squared of the product of a bra $\langle\phi|$with a ket $|\psi\rangle$ gives the probability of measuring ϕ given that the system is in state $|\psi\rangle$.

$$P(m(\gamma) = \phi \mid \gamma = \psi) = |\langle\psi|\phi\rangle|^2 \tag{2.1}$$

where $m(\gamma)$ is the outcome of measuring system γ, and $\langle\psi|\phi\rangle$ is an abbreviation for $\langle\phi||\psi\rangle$.

From equation 2.1 and the experimental observation we have that

$$|\langle S_z + |S_z+\rangle|^2 = 1$$

and,

$$|\langle S_z + |S_z-\rangle|^2 = 0.$$

Since the probabilities of measuring the outcome S_z+ is 1 given that the system is state S_z+ and 0 if the system is in state S_z-. We conclude that $\{|S_z+\rangle, |S_z-\rangle\}$ forms an orthonormal basis of the spin state-space. Therefore, we must be able to write any other spin state as a linear combination of these two states.

From experimental observation we also obtain that:

$$|\langle S_z + |S_x+\rangle|^2 = |\langle S_z - |S_x+\rangle|^2 = \frac{1}{2} \tag{2.2}$$

$$|\langle S_z + |S_x-\rangle|^2 = |\langle S_z - |S_x-\rangle|^2 = \frac{1}{2}. \tag{2.3}$$

From equations 2.2 and 2.3 we deduce a possible way to write $|S_x+\rangle$ and $|S_x-\rangle$, in terms of $|S_z+\rangle$ and $|S_z-\rangle$:

$$\begin{aligned} |S_x+\rangle &= \frac{1}{\sqrt{2}}|S_z+\rangle + \frac{1}{\sqrt{2}}|S_z-\rangle \\ |S_x-\rangle &= \frac{1}{\sqrt{2}}|S_z+\rangle - \frac{1}{\sqrt{2}}|S_z-\rangle. \end{aligned}$$

As for the states S_y+ and S_y-, we know that:

$$
\begin{aligned}
|\langle S_z + |S_y+\rangle|^2 = |\langle S_z - |S_y+\rangle|^2 = \frac{1}{2} \\
|\langle S_z + |S_y-\rangle|^2 = |\langle S_z - |S_y-\rangle|^2 = \frac{1}{2} \\
|\langle S_x + |S_y+\rangle|^2 = |\langle S_x - |S_y+\rangle|^2 = \frac{1}{2} \\
|\langle S_x + |S_y-\rangle|^2 = |\langle S_x - |S_y-\rangle|^2 = \frac{1}{2}
\end{aligned}
$$

from which we conclude that the only way to write the states $|S_y+\rangle$ and $|S_y-\rangle$ in terms of $|S_z+\rangle$ and $|S_z-\rangle$ is

$$
\begin{aligned}
|S_y+\rangle &= \frac{1}{\sqrt{2}}|S_z+\rangle + i\frac{1}{\sqrt{2}}|S_z-\rangle \\
|S_y-\rangle &= \frac{1}{\sqrt{2}}|S_z+\rangle - i\frac{1}{\sqrt{2}}|S_z-\rangle.
\end{aligned}
$$

We fully described the observed states of the previous section. However, these are not the only possible quantum states of a simple 1/2 spin system. Fixing a particular basis, say $\{|S_z+\rangle, |S_z-\rangle\}$ any unit-length linear combination

$$|\psi\rangle = \alpha|S_z+\rangle + \beta|S_z-\rangle \tag{2.4}$$

such that $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$ is a possible quantum state. Unit length is necessary to guarantee that all probabilities add up to one.

It is easy to show that if $|\gamma| = 1$ then $|\psi\rangle$ and $\gamma|\psi\rangle$ represent the same state. Hence, by convention α is taken to be real-valued.

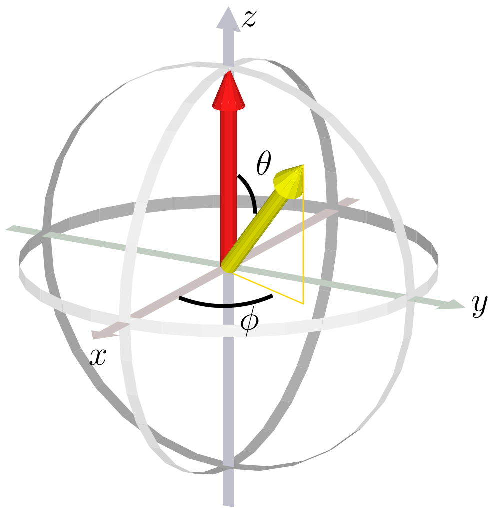


Figure 2.4: The Bloch Sphere

It is important to note that while global phase is not measurable, relative phase *is*. That is the state $|S_z+\rangle + |S_z-\rangle$ is different from $|S_z+\rangle - |S_z-\rangle$; the first state is equivalent to $|S_x+\rangle$, and the second to $|S_x-\rangle$, which are orthogonal states.

A convenient way of visualizing the state $|\psi\rangle$ of equation 2.4 is the Bloch sphere. Let $|\psi\rangle = \cos\frac{\theta}{2}|S_z+\rangle + e^{-i\phi}\frac{\theta}{2}|S_z-\rangle$. Then $|\psi\rangle$ is represented in the Bloch sphere (see figure 2.2.2) as the unit vector $(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$.

Say we measure the z-magnetization of the same (prepared) state $|\psi\rangle$. Whenever we measure positive z-magnetization we note down $+1$, and we use -1 to refer to negative magnetization. The expected value of such a measurement would then be:

$$|\langle S_{z+}||\psi\rangle|^2 - |\langle S_{z-}||\psi\rangle|^2. \tag{2.5}$$

A more compact way to express equation 2.5 is to use *observables*.

The z-magnetization is said to be an 'observable' of the spin. So are the x and y magnetizations.

Observables are represented by Hermitian matrices. The x, y and z magnetization observables are represented by the Pauli matrices σ_x, σ_y and σ_z respectively:

$$\begin{aligned}
\sigma_x &= |S_x+\rangle\langle S_x+| - |S_x-\rangle\langle S_x-| \\
\sigma_y &= |S_y+\rangle\langle S_y+| - |S_y-\rangle\langle S_y-| \\
\sigma_z &= |S_z+\rangle\langle S_z+| - |S_z-\rangle\langle S_z-|.
\end{aligned}$$

The expected value of an observable σ when measuring the state $|\psi\rangle$ is given by[5]:

$$\langle\sigma\rangle = \langle\psi|\sigma|\psi\rangle. \tag{2.6}$$

Equation 2.6 can easily be derived from equation 2.5.

Any closed quantum system evolves in time. Its evolution is given by a unitary matrix, called the time-evolution operator U. Hence, if $|\psi(0)\rangle$ is the initial state of the system the system at time t is given by $|\psi(t)\rangle = U(t)|\psi(0)\rangle$. The operator U is related to the total energy function, or Hamiltonian, of the system by the Schrödinger equation:

[5]In figure 2.2.1, in equation 2.6, and in the rest of this chapter we have omitted a factor of $\hbar/2$.

$$i\hbar \frac{d}{dt} U = UH \tag{2.7}$$

which has solution:

$$U = e^{-i\frac{1}{\hbar}H}. \tag{2.8}$$

The exponential of a matrix is easily defined using the series expansion

$$e^A = \sum_{i=0}^{\infty} \frac{A^n}{n!}.$$

We will not concern ourselves with solving the Schrödinger equation. It suffices to know that evolution of a system is given by a unitary matrix, that depends solely on the Hamiltonian of the system. Also, in what follows we will set the physical constant $\hbar$ to 1, in other words, we set $\hbar$ as our unit (everything is then measured in multiples of $\hbar$). Thus, from henceforward we shall omit $\hbar$ from our calculations.

Examples of unitary matrices are the Pauli matrices. Seen as an operator the Pauli matrix σ_x has the effect of mapping the state $|S_z+\rangle$ to the state $|S_z-\rangle$ and vice versa:

$$\begin{aligned}
\sigma_x|S_z+\rangle &= (|S_x+\rangle\langle S_x+| - |S_x-\rangle\langle S_x-|)|S_z+\rangle \\
&= |S_x+\rangle\langle S_x+||S_z+\rangle - |S_x-\rangle\langle S_x-||S_z+\rangle \\
&= |S_x+\rangle\frac{1}{\sqrt{2}} - |S_x-\rangle\frac{1}{\sqrt{2}} \\
&= \frac{1}{\sqrt{2}}|S_x+\rangle - \frac{1}{\sqrt{2}}|S_x-\rangle \\
&= |S_z-\rangle.
\end{aligned}$$

Finally, if two quantum systems (for example two quantum spins) are composed to form single larger system, the state space of the composite system is the *tensor* product of the two subsystems. The tensor of two state $|\psi\rangle$ and $|\phi\rangle$, formally denoted by $|\psi\rangle \otimes |\phi\rangle$ is usually abbreviated as $|\psi\rangle|\phi\rangle$ or even $|\psi\phi\rangle$.

Likewise, operators acting on two spins are tensor products of single spin operators. For example $\sigma_x \otimes \sigma_y$, and $\sigma_x \otimes \mathbb{1}$ are both two spin operators. However, notice that the second operator acts on the first spin, leaving the second one untouched. For notational convenience we refer to such operator as $\sigma_x^{(1)}$, that is the Pauli operator x acting on the first spin. Likewise, $\sigma_z^{(2)}$ is the Pauli operator z acting on the second spin, formally it is $\mathbb{1} \otimes \sigma_z$.

Quantum mechanics is a huge and growing area of physical science. For a more in depth look, see for example [Sak94]. An encyclopedic approach is given in [Mes99], and an introduction in the context of QIP is given in [NC00].

We next turn to QIP proper.

2.3 Qubits

In computation, as well as information theory the basic unit of information is the *bit*[6]. The bit has two possible values: zero (0), and one (1).

In QIP the basic unit of information is the qubit. Like the spin-1/2, it is a two dimensional complex-valued scalar vector of unit length. It is the generalization of the simple bit to the quantum realm. A quantum bit can also have the values $|0\rangle$ and $|1\rangle$, but can also have as value any superposition of these two basis states.

A qubit can be represented physically in many ways, one such way is precisely through a spin system, as is done in NMR QIP.

A special basis is fixed for the qubit, called the computational basis: $\{|0\rangle, |1\rangle\}$, representing the logical zero and one. In a spin system we usually map the logical $|0\rangle$ and $|1\rangle$ to the physical $|S_z+\rangle$ and $|S_z+\rangle$ respectively.

All rules of measurements, evolution, and system composition apply equally well to qubits. A system of n qubits rest in a 2^n dimension vector space whose basis states are $|00\ldots00\rangle, |00\ldots01\rangle, |00\ldots10\rangle, \ldots, |11\ldots10\rangle, |11\ldots11\rangle$; or more compactly $\{|i\rangle\}_{i\in 0..2^n-1}$.

Hence any n qubit register can be written as unit-length linear combination:

$$\sum_{i\in I} \alpha_i |a_i\rangle$$

such that

[6]In information theory the *gnat* is also sometimes used, but the bit is, by far, more common.

$$\sum_{i\in I} |\alpha_i|^2 = 1.$$

Whereas in traditional quantum mechanics we are interested in the natural evolution of a quantum system, in QIP, we induce a special evolution, designed to carry out a a computation.

In such view, unitary operators acting on one or more qubits are referred to as quantum *gates*. Examples of quantum gates are the, already seen, Pauli matrices. The Pauli matrix σ_x is also called the *NOT* gate, as it has the effect (seen in the previous section) of negating the values of the computational basis. The identity matrix $\mathbb{1}$ is also a quantum gate, corresponding to the *nop* (no operation) of traditional computing.

Another very important gate is the Hadamard (H) operator:

$$H|0\rangle = 1/\sqrt{2}|0\rangle + 1/\sqrt{2}|1\rangle$$
$$H|1\rangle = 1/\sqrt{2}|0\rangle - 1/\sqrt{2}|1\rangle.$$

Other important operators, acting on one qubit are the exponentials of operators. If σ is a Hermitian *idempotent* matrix[7] then $e^{-i\theta\sigma}$, is a unitary matrix equal to $\cos\theta\mathbb{1} - i\sin\theta\sigma$. It corresponds to a rotation of 2θ along the axis σ in the Bloch sphere picture. For example, $e^{-i\pi/4\sigma_z}$ corresponds to a rotation of $\pi/2$ along the

[7]All Pauli matrices are idempotent. Also, it is not hard to prove that any unit-length linear combination of the Pauli matrices $\sigma = \alpha\sigma_x + \beta\sigma_y + \gamma\sigma_z$ with $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$ is also idempotent.

z-axis.

An important fact useful in understanding NMR QIP is that *any* quantum single-qubit operation can be seen as a rotation, in the Bloch sphere picture, of an angle θ along an arbitrary axis $\vec{n} = (n_x, n_y, n_z)$. Such a rotation is written as $e^{-i\theta(n_x\sigma_x + n_y\sigma_y + n_z\sigma_z)}$.

We shall see in chapter 4 that even operations of the type $e^{-i\theta\sigma_z\otimes\sigma_z}$, where the exponent includes the tensor of two or more matrices can be seen as a rotation in a generalization of the Bloch sphere called *product operator space*.

An important two-qubit operator is the *c-not*. It is given as follows:

$$\begin{aligned}
CNOT|00\rangle &= |00\rangle \\
CNOT|01\rangle &= |01\rangle \\
CNOT|10\rangle &= |11\rangle \\
CNOT|11\rangle &= |10\rangle.
\end{aligned}$$

Classically, we can think of the c-not as flipping the second register if and only if the first register is set to 1. In the quantum case, it is subtler. We shall see in the next section, how this, and other, subtleties allow us to do fast computations.

The $CNOT$ gate, in conjunction with arbitrary rotations of single qubits composes a complete set of quantum gates. A *complete* set of quantum gates is one that, given any arbitrary Hamiltonian H, H can be simulated with arbitrary precision, using only quantum gates in the set.

The proof of the fact the $CNOT$ and single qubit gates form a complete set of gates can be found in [NC00].

We see then quantum computation as inducing a set of quantum gates performed on a system, that is initially in a well defined state (see below), and performing a measurement at the end.

For example, if the set of gates (unitary transformations) that we are to perform are $U_1, U_2, \ldots, U_n$ then what we are doing is performing the operation:

$$|\psi_0\rangle \longrightarrow U_n U_{n-1} \ldots U_1 |\psi_0\rangle$$

By setting $U = U_1 U_2 \ldots U_n$ we see that U is the evolution operator of the whole system. Now we can also turn around and see computation from a different perspective:

Given a Hamiltonian, break up its evolution operator $U = e^{-i\hbar H}$ into a set of achievable quantum gates $U_1, U_2, \ldots, U_n$, and perform them on an appropriately initialized system.

This view is the one normally taken when one is interested in *simulating* quantum systems. However, it is interesting to see that both views apply equally well to both simulation and computation. In a deep and meaningful way, we conclude that experimentation and computation are one and the same.

We now introduce a formalism that is very useful in designing, and presenting, quantum algorithms.

2.4 Quantum Circuits

David Deutsch [Deu85] was the first to formalize the idea of a Quantum Computer. He used the Turing Machine formalism, and extended it to encompass quantum states. In short, the tape of a Quantum Turing machine, consists of an infinite discrete number of two-state quantum systems, or qubits. The head of the QTM is also a finite-dimensional quantum system.

Computational complexity of quantum Turing machines was analyzed in depth in [BV97]. We will, however, limit ourselves to the quantum circuit model. For the most part these two computational models are equivalent (see [Yao93]), though there are some perks.

The inputs to a quantum circuit are assumed to be $|0\rangle$. Hence a quantum circuit with n qubits begins in the state $|0^n\rangle$. The 'input' string is created by applying unitary transformations to the qubits. In section 2.1.1 we stated that for classical networks it is equivalent to simply allow any arbitrary input to circuit. In the quantum case we wish to be more careful. We wish to guarantee that all values carried along the network are computable. If we were to allow arbitrary inputs to the quantum circuit, then in principle a qubit could be input in the state $|0\rangle + e^{-i\Omega}|1\rangle$ where Ω is an uncomputable number.

Furthermore we restrict gates to be unitary operators acting on only one or two qubits, *where all entries in the matrix representation of the operator are computable.* We shall call such operators *computable.*

Definition 9 (Quantum Acyclic Circuit). *A quantum acyclic circuit is a directed, acyclic graph $G = (V, E)$, where each is vertex is of the following form:*

Inputs *Have in-degree zero. We label inputs as* $|x_1\rangle, |x_2\rangle, \ldots$. *These vertexes have out-degree one.*

Gates *Can be of two types: Arbitrary, computable one-qubit gates, or CNOT. One qubit gates have in-degree and out-degree one. CNOT gates have in-degree and out-degree two.*

Previously we stated that Nielsen and Chuang [NC00] showed that CNOT and arbitrary single qubit gates are universal in the sense that an arbitrary gate U can be expressed exactly as a product of these gates.

Now if U is a *computable* unitary $n \times n$ matrix, the same proof shows that it can be decomposed into a product of *computable* single qubit gates, and CNOT.

Hence our definition correctly captures quantum computation.

One possible flaw of the above definition is that it uses an infinite set of gates. We can restrict ourselves to circuits that only use a finite set of gates, if we are willing to simulate any unitary transformation approximately only [NC00].

Notice also that in contrast to the classical case, every gate is required to have exactly the same out-degree as its in-degree, in other words $FANOUT$ is not allowed. This is because, in general, it is impossible to copy a quantum state.

A simple, yet powerful, example of a quantum circuit is given in inset 2.4.1.

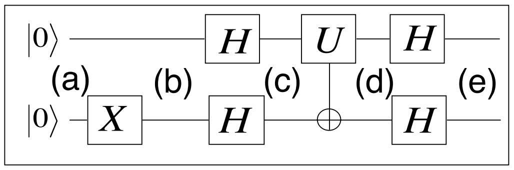
2.5 Quantum Computational Complexity

Among the quantum complexity classes developed so far, three are essential to our discussion. We shall now go over them, and the most important results pertaining

Deutsch's Problem is defined as follows: Let $f : \{0, 1\} \to \{0, 1\}$ be any function that maps a single bit to a single bit. Find out, using the minimum number of queries, whether f is *constant*, that is $f(0) = f(1)$, or f is *balanced*, in other words $f(0) \neq f(1)$ [Deu85]. Note that this is equivalent to calculating $f(0) \oplus f(1)$.

It is easy to see that classically, even with a probabilistic algorithm, it is impossible to decide Deutsch's problem with any advantage with less than two queries.

However, the quantum circuit below, due to Cleve, Ekert, Macchiavello and Mosca [CEMM97], gives the correct answer with just one query (the gate U is such that $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$):



.

Before we analyze the algorithm lets note that $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$, hence:

$$U|x\rangle(|0\rangle - |1\rangle) = |x\rangle(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle = \\ |x\rangle(-1)^{f(x)}(|0\rangle - |1\rangle) = (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle).$$

Now, we analyze the evolution of the quantum register through the quantum circuit above:

$$\begin{aligned}
(a) &= |0\rangle|0\rangle \\
(b) &= |0\rangle|1\rangle \\
(c) &= (|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\
(d) &= ((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle)(|0\rangle - |1\rangle) \\
&= (-1)^{f(0)}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)(|0\rangle - |1\rangle) \\
&= (|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)(|0\rangle - |1\rangle) \\
(e) &= |f(0) \oplus f(1)\rangle|1\rangle
\end{aligned}$$

Hence, the circuit outputs the correct answer in the first qubit.

Inset 2.4.1: Deutsch's Problem

to them. We will introduce one further class when we discuss NMR QIP.

Definition 10. ***EQP*** *is the set of languages for which there exists a polynomial size quantum circuit that decides L* deterministically*, i.e. it outputs the correct answer with probability* 1 *[BV97].*

Theorem 2. $\boldsymbol{P} \subseteq \boldsymbol{EQP}$.

The inclusion is not that obvious. It was proved by Bennett that all non-reversible operations can implemented using only reversible ones. Hence, unitary operators (which, by nature must be reversible) can simulate any type of computation, reversible or not, and the previous result is valid. Furthermore, it must be stated that although there is no conclusive proof, there exists good evidence that the inclusion is strict [BV97].

Definition 11. ***ZQP*** *is the set of languages for which there exists a polynomial size quantum circuit that either accepts, rejects or fails. If it accepts, then the input is* guaranteed *to be in the language, likewise, if it rejects then it is guaranteed not to be. Furthermore the probability that the circuit fails for any given input is polynomially bounded away from 1.*

Note that in this definition a quantum computer is not required to give an answer, however if it does give an answer it is guaranteed to be the correct one (hence the name which stands for zero error quantum polynomial time). An example of an algorithm of this type is Deutsch's original solution to his problem [Deu85].

Definition 12. ***BQP*** *is the set of languages for which there exists a polynomial size quantum circuit that gives the correct answer with probability* $\frac{2}{3}$.

Similar to the results above we have the following one:

Theorem 3. ***BPP*** $\subseteq$ ***BQP***.

As in the classical setting, we regard **BQP** as the set of languages that can efficiently be decided on a quantum computer. The next result follows directly from the definitions:

Theorem 4. ***EQP*** $\subseteq$ ***ZQP*** $\subseteq$ ***BQP***.

Not much is known about the relationship between classical and quantum computational complexity classes. One important result however is the following one taken from [Cle99]:

Theorem 5. *For any n-qubit quantum circuit with m gates there is a classical probabilistic circuit that 'simulates' it with accuracy* ϵ *that uses* $O(2^n m^3 log^2(1/\epsilon))$ *gates.*

An important consequence is the following:

Corollary 1. ***BQP*** $\subseteq$ ***EXP***.

Actually **BQP** is in a smaller class called $P^{\sharp P}$. For the definition of such a class see [Pap94].

A very important problem, both in science and industry is $FACTORING$. In short, the problem is given an n-bit composite number, find a prime factor. $FACTORING$ is considered a —classically— difficult problem, since it is not known to be in **P** or even **BPP**. However, one of the most famous results in quantum computing is the following:

Theorem 6. $FACTORING \in \boldsymbol{BQP}$.

This last result is from [Sho97]. He gives in fact a circuit of size $O(n^2)$. A different analysis of the algorithm is given in [CEMM97], [Mos99] and [NC00]. Another very important problem related to $FACTORING$ is $DISCRETE\ LOGARITHM$. This is also shown to be in **BQP**.

Chapter 3

NMR QIP

3.1 NMR Spectroscopy

Nuclear Magnetic Resonance (NMR) spectroscopy was discovered in 1945 [Fre98]. Since then it has matured and become one of the most important techniques in diverse areas, from biology and medicine, to chemistry and physics. NMR is used today to investigate the structure of a molecule in a liquid state and for solid state structure analysis.

NMR spectroscopy uses the intrinsic magnetic moment, or spin, of the nuclei in the molecule. A (large) sample of the molecule is dissolved in a liquid and then set in a strong magnetic field. This causes the nuclei to precess (except those nuclei that happen to be aligned with the external force), causing a magnetic field. Although the magnetic field of a single nucleus is far too weak to be registered with present day technology, by using a very large sample (in the order of 10^{19}), (an approximation to) the overall magnetization —referred to as *bulk* magnetization—

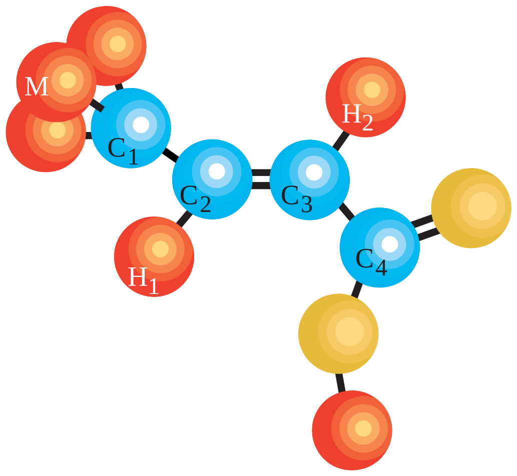


Figure 3.1: Labelled trans-crotonic acid

can be recorded over time.

Different types of nuclei precess at very different frequencies. The frequency of precession of each nucleus is usually called its Larmor frequency after the British physicist Joseph Larmor (1857-1942).

Commonly used nuclei in NMR include hydrogen, carbon-13, fluorine-19, phosphorus-31 and nitrogen-15.

Just as any two nearby magnetic fields interact so do nearby nuclei. These interaction also influence the precession frequency of each nucleus.

In common NMR spectroscopy this difference in precession frequency is used to investigate the structure of the molecule. In QIP the knowledge of the structure of the molecule is used to distinguish the contribution to overall magnetic field by each of the different nuclei of the same type.

For QIP each nucleus is used as a different qubit. Since it is necessary to be able to distinguish each qubit separately, molecules are chosen that are highly *asymmetrical*. This is to ensure that the bulk magnetization of each nucleus is distinguishable from other nucleus of the same type (we see later on that this is one of the hurdles on the road toward scalable QIP with NMR).

In experiments currently done at the University of Waterloo, as well as other places (MIT, LANL), the molecule of choice is trans-crotonic acid (CH3-CH=CH-COOH). Figure 3.1 shows a representation of the molecule. The nuclei used for QIP are the four carbons, the two hydrogen nuclei adjacent to the double bond, and the spin-1/2 component of the methyl group.

In implementing a quantum algorithm on a NMR spectrometer the final measurement is obtained using coils attuned to the different Larmor frequencies of each type of nucleus to measure the overall phase and magnitude of the magnetization. Post processing is done to obtain actual values for the different nuclei of the same type.

In order to implement gates, radio-frequency, or RF, pulses attuned to the precessing frequency of each nucleus are used. To prepare the initial state, these same RF pulses, together with a gradient magnetic fields are used.

In figure 3.2 a schematic picture of a spectrometer is shown. The sample is introduced into the central core, inside the *probe*. The probe (shown in the insert) contains the coils used both for the RF-pulses and the magnetic fields gradients. The probe is surrounded by a superconducting magnet. The whole machinery is controlled by a regular UNIX machine.

In the next sections the details of computation with NMR are more fully ex-

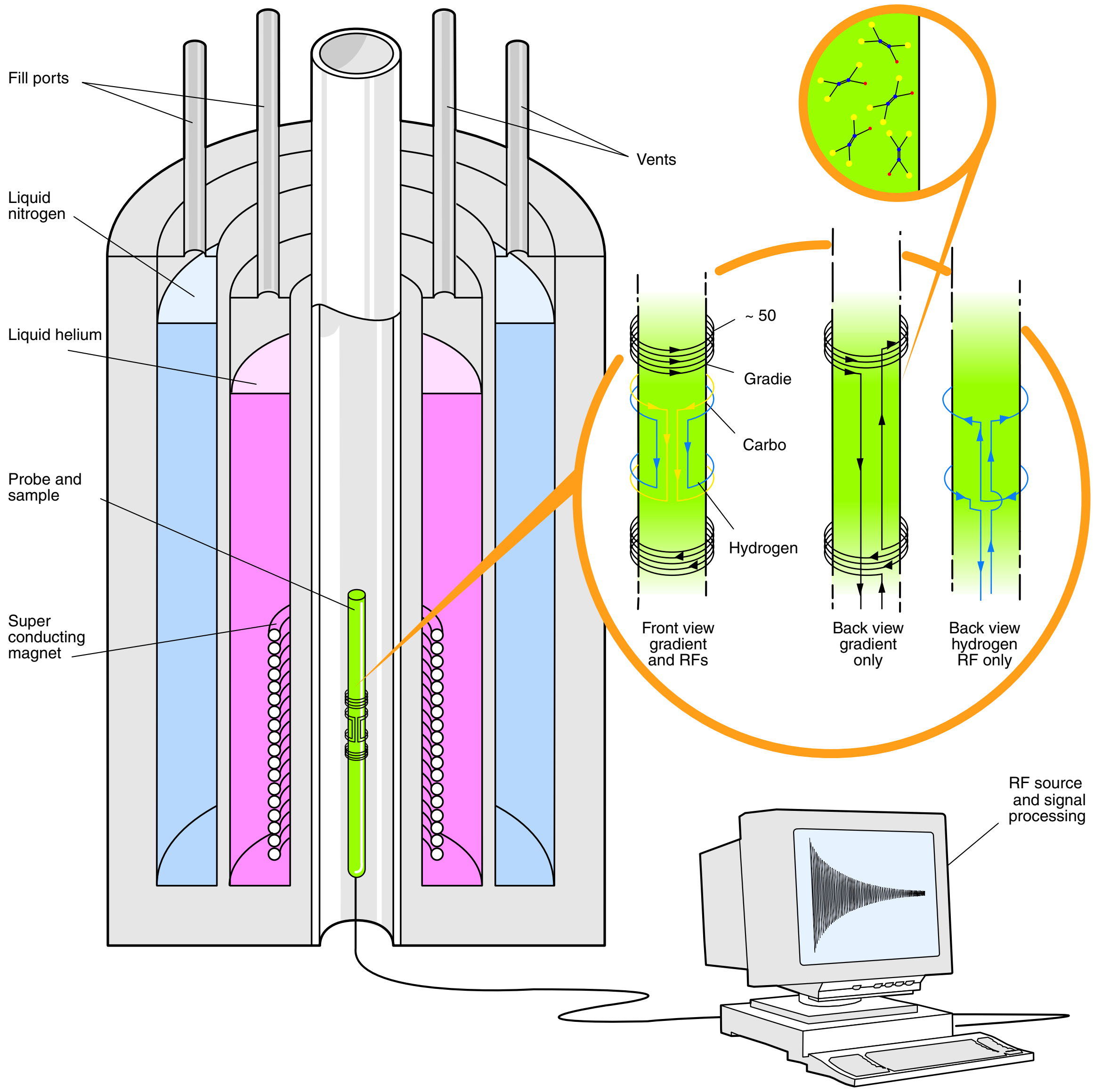


Figure 3.2: An NMR Spectrometer

plained.

3.1.1 Mixed Quantum States

Before continuing it is necessary to take a moment to study a different formalism to represent quantum states and evolution than the one introduced in chapter 1. This is the *density matrix* formalism.

In an NMR sample, the number of molecules is enormous (around 10^{20}). The RF-pulses, and the final measurements act on a specific nucleus of the whole *ensemble* of molecules in the sample. For example, an RF-pulse attuned to one of the protons in the register-molecule will act on that same proton, of all the molecules in the sample.

However, not every molecule will have that proton in the same state. Some molecules, say 30% may have the proton in the state $|\psi\rangle$ while the other 70% might have it in the state $|\phi\rangle$. We can refer to this mixture as the ensemble $\{(0.3, |\psi\rangle), (0.7, |\phi\rangle)\}$.

A NOT (X) gate applied to the proton would then change the ensemble to the mixture $\{(0.3, X|\psi\rangle), (0.7, X|\phi\rangle)\}$. A measurement in the computational basis of the whole ensemble will have an expected value, $\langle z\rangle = 0.3\langle\psi|\sigma_z|\psi\rangle + 0.7\langle\phi|\sigma_z|\phi\rangle$.

This method of handling mixed states is cumbersome, especially when referring to highly mixed states. Also, due to the nature of quantum measurement, it over-specifies quantum systems.

Instead of referring to parts of an ensemble directly it is possible assign to the whole ensemble the mixed quantum state $\rho = 0.3|\psi\rangle\langle\psi| + 0.7|\phi\rangle\langle\phi|$. In this case ρ

is called the density matrix of the ensemble. It is related to the probability density function in classical statistics, in the sense that it gives a statistical prediction of measurements on an ensemble of quantum systems.

Density matrices are also a convenient representation of quantum system where *decoherence* is involved. Decoherence, in short, is the *randomization* of quantum state due to unwanted interaction with other systems.

The evolution of a density matrix under a unitary transformation is given by the formula:

$$\rho \xrightarrow{U} U\rho U^\dagger. \tag{3.1}$$

The trace of an operator U is defined to be

$$Tr(U) = \sum_i \langle b_i|U|b_i\rangle$$

where $\{|b_i\rangle\}$ is taken to be any orthonormal basis (it can be easily shown that the trace is invariant under change of basis). In terms of matrices the trace is simply the sum of the diagonal elements.

The definition of trace has the following easy to prove consequences:

$$Tr(AB) = Tr(BA)$$

$$Tr(A + B) = Tr(A) + Tr(B)$$

$$Tr(zA) = zTr(A)$$

for any two matrices A and B, and scalar z.

The expected outcome of an observable $\langle \sigma \rangle$ is given by the following equation

$$\langle A \rangle = Tr(\rho\sigma) \tag{3.2}$$

We can derive equation 3.2 from the law of total probabilities $P(a) = \sum_i P(a|x_i)P(x_i)$, in the following way.

Let

$$\rho = \sum_i p_i |b_i\rangle\langle b_i|$$

then using the law of total probabilities we have that

$$\begin{aligned}
\langle \sigma \rangle &= \sum_i p_i \langle b_i|\sigma|b_i\rangle \\
&= \sum_i p_i \langle b_i|\sigma \mathbb{1}|b_i\rangle \\
&= \sum_i p_i \langle b_i|\sigma \sum_j |b_j\rangle\langle b_j||b_i\rangle \\
&= \sum_{i,j} p_i \langle b_i|\sigma|b_j\rangle\langle b_j||b_i\rangle \\
&= \sum_{i,j} \langle b_j||b_i\rangle p_i \langle b_i|\sigma|b_j\rangle \\
&= \sum_j \langle b_j| \sum_i p_i |b_i\rangle\langle b_i|\sigma|b_j\rangle
\end{aligned}$$

$$= \sum_j \langle b_j|\rho\sigma|b_j\rangle$$

$$= Tr(\rho\sigma)$$

where we have used the fact that for any orthonormal basis $\{|b_j\rangle\}$, $\mathbb{1} = \sum_j |b_j\rangle\langle b_j|$.

An important fact to note is that the density matrix completely describes the statistics of a state. For example, an equal mixture of states $|S_z+\rangle$ and $|S_z-\rangle$ and equal mixture of states $|S_x+\rangle$ and $|S_x-\rangle$ have exactly the same measurement statistics. From measurements alone, we would not be able to distinguish both mixtures.

Notice that they have the very same density matrix. Hence the density matrix is a better way to represent mixtures: it has all the information necessary to characterize a mixed quantum state, and no more.

A state represented by a density matrix ρ is usually referred to as a *mixed* quantum state, as opposed to a *pure* quantum state given by a state vector $|\psi\rangle$. However, a density matrix *can* represent pure states. For example $\rho = |\psi\rangle\langle\psi|$ and the vector $|\psi\rangle$ represent the same state.

Finally, a density operator also has a representation in the Bloch sphere. The state ρ corresponds to the point $(Tr(\rho\sigma_x), Tr(\rho\sigma_y), Tr(\rho\sigma_z))$. Although pure states lie always on the sphere's surface a mixed state can lie anywhere inside the sphere. The point $(0, 0, 0)$ is the completely mixed or random state. It corresponds to the density matrix $\rho = \mathbb{1}$.

Recall that in quantum computing we wish to measure in the computational basis. Assuming the usual correspondence, logical $|0\rangle$ and $|1\rangle$ are represented by the physical spins $|z\uparrow\rangle$ and $|z\downarrow\rangle$ respectively, the probability p of measuring a 1 on qubit n satisfies the equation $1-2p = Tr(\rho\sigma_z^{(n)})$, therefore

$$p = \frac{1 - Tr(\rho\sigma_z^{(n)})}{2}.$$

Although we introduced the density operator in the setting of mixtures, it must me noted the density operators are used to refer to the state of a single quantum system, where —through decoherence for instance— there has been loss of information, and a complete description of the state is lacking.

For a more complete description of the density matrix formalism see for example [NC00]. We now turn back to NMR QIP.

3.2 Computing with NMR

DiVincenzo gives five requirements that a system should obey in order for it to be suitable for general purpose, scalable, quantum computation:

1. A scalable physical system with a mapping of qubits onto this system
2. A method for initializing the state of the system to an a priori known state, usually corresponding to $|0\rangle^{\otimes n}$ under the above mapping.
3. A large decoherence-time to gate-time ratio,
4. Sufficient control of the system via time-dependent Hamiltonians in order to effectively implement a universal set of gates on the system's qubits, and finally

5. A measurement operation on the system's qubits.

The requirement for state initialization seems to imply that the ability of a system to exhibit pure quantum states is necessary for quantum computation, a requirement that is *not* fulfilled by NMR spectroscopy. We shall see, however, in section 3.2.5, that it is possible to do computation with a weaker supposition, the ability to create so-called *pseudo-pure* states.

We shall now discuss in detail how to provide four of DiVincenzo's five requirements for QIP, starting with qubit implementation. Fault tolerant QIP will be the focus of the next chapter.

3.2.1 Implementing Qubits

The spin of each nucleus in the register molecule represents a single qubit. Recall that the nuclear-magnetic spin of a particle is a vector (ray) in real-space. Typically, a qubit to spin mapping associates the logical state $|0\rangle$ with the spin state $|s_z+\rangle$, and similarly $|0\rangle$ with $|s_z-\rangle$.

One problem remains however. The strong fixed magnetic field of the spectrometer induces the evolution $|\psi_t\rangle = e^{i\omega\sigma_z t/2}|\psi_0\rangle$. Therefore, each qubit is constantly changing its value, precessing around the z-axis at the so-called Larmor frequency; this is a handicap if these qubits are to be used for computation.

A simple solution is to carry the computations in a frame of reference that rotates at the same frequency as the qubit in question. This frame of reference is aptly named the 'rotating frame', or sometimes the 'logical' frame.

Since each nucleus has a different Larmor frequency, the rotating frame for each nucleus is different.

The 'logical' frame and the 'standard' frame for each qubit are related by the equations:

$$z_l = z_s$$

$$x_l = x_s + \omega t$$

$$y_l = y_s + \omega t$$

where x_l, y_l and z_l are the Bloch sphere coordinates in the logical basis, and x_s, y_s and z_s are the coordinates in the standard frame.

Care must be taken, hence, to effectively carry out gates in the correct 'logical' frame, and to correctly translate measurements results, since from the spin's perspective the laboratory and all its equipment is rotating.

3.2.2 Implementing Gates

In chapter two we mentioned that arbitrary one qubit gates are required in order to achieve a universal set of gates. However, in order to achieve an arbitrary rotation θ along an arbitrary axis $\vec{n} = (n_x, n_y, n_z)$, it is sufficient to be able to achieve $\pi/2$ along the x-axis along with arbitrary rotations along the z-axis. Inset 3.2.1 shows how to achieve this.

Now, a rotation of ϕ along the z-axis is easily achieved by simply updating the value of the rotating frame. No actual physical operation is needed, and thus, as a

A rotation along the direction $\vec{n} = (n_x, n_y, n_z)$ by an angle θ can be achieved by the following combination of the rotations:

$$e^{-i\frac{\alpha}{2}Z}e^{i\frac{\pi}{4}X}e^{i\frac{\beta}{2}Z}e^{-i\frac{\pi}{4}X}e^{-i\frac{\theta}{2}Z}e^{i\frac{\pi}{4}X}e^{-i\frac{\beta}{2}Z}e^{-i\frac{\pi}{4}X}e^{-i\frac{\alpha}{2}Z}$$

where

$$\alpha = \tan^{-1}\frac{n_y}{n_x}$$
$$\beta = \tan^{-1}\frac{n_x}{n_z}.$$

To prove that such sequence indeed achieves the desired rotation it is sufficient to expand, and then simplify the expression.
Intuitively it is easy to see, however, how the sequence works. The first three rotations serve to align the z-axis with the vector $\vec{n} = (n_x, n_y, n_z)$, the fifth rotation achieves the desired operation by rotating along the z-axis (which points along $\vec{n}$), and the last four rotations return the z-axis to its proper alignment.

Inset 3.2.1: Achieving arbitrary rotations

basic computational operation, it is considered to take zero time.

To achieve a rotation along the x-axis it is necessary to ‘tilt’ the spin by means of an electromagnetic pulse. However, a constant electromagnetic pulse (as observed from the laboratory) would seem to *rotate* as seen from the spin’s logical frame.

The electromagnetic pulses emitted by the spectrometer are relatively weak compared to the constant external field. Hence, the rotation along the z-axis is much faster than the rotation along the x-axis induced by the pulse. The effects of the pulse would mostly cancel out.

This is because as the nucleus precesses along the z-axis the electromagnetic pulse changes from pushing in the $x+$, to pushing in $y+$, the $x-$, the $y-$, and again the $x+$ direction and so forth.

The solution is to emit the electromagnetic pulse at the same frequency as the

Larmor frequency of the nucleus in question. This way, the pulse looks constant from the nucleus rotating frame perspective. This sort of pulse is called a resonant pulse. Since the Larmor frequency of the nuclei in question are normally in the hundreds of MHz, the pulses used are in the radio frequency. Hence the name: RF-pulses.

Since different kinds of nuclei have very different Larmor frequencies, a pulse that is resonant with a carbon nucleus will seem to be rotating highly from the perspective of a hydrogen nucleus rotating frame. Hence, for this latter nucleus the effects of the pulse will cancel out almost completely.

Although different types of nuclei have very different Larmor frequencies, similar nuclei do not. Hence, an RF pulse designed to implement a rotation of one carbon nucleus, will likely affect the other carbon nuclei in the molecule. This is called 'crosstalk' and is one of the most important causes of errors in NMR QIP.

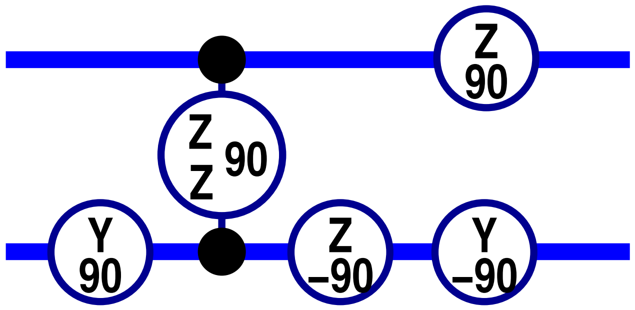
The solution to such a problem is one of the main focuses of this work, and will be more amply discussed in the next chapter.

Two-Qubit Gates

The effects of each nuclei magnetic dipole moment make nearby nuclei interact, not unlike nearby magnets do [LKC+02].

If the interaction between two nuclei is not very strong, then the evolution on the two spins under it can be described by the Hamiltonian $H = 2J\pi Z_1 Z_2 + \phi Z_1 + \theta Z_2$ where ϕ and θ depend on the Larmor frequencies of the first and second spin respectively. Since this Hamiltonian is invariant under rotation along the z-axis we can study the evolution under the rotating frame of each nucleus. The

Here is a pulse sequence that implements the C-NOT gate.



A very simple analysis comes from studying the gate on inputs $|0\rangle|\psi\rangle$, $|1\rangle|\psi\rangle$. By linearity, the analysis then extends to arbitrary inputs.

On input $|0\rangle|\psi\rangle$ the σ_z on the second qubit cancels out with the the effect of the j-coupling: the j-coupling rotates the second qubit 90 degrees along the z-axis, and the σ_z gate rotates it back. The σ_y gates then cancel out.

On input $|1\rangle|\psi\rangle$ the j-coupling and the σ_z add up, to create a 180 degree rotation of the second qubit along the z-axis. The σ_y gates transform this into a 180 x rotation since $\sigma_x = \sigma y \sigma_z \sigma y$.

The σ_z on the first qubit is to ensure the correct phase of the first qubit.

Inset 3.2.2: Implementing the C-NOT Gate

Hamiltonian then becomes $H = 2J\pi Z_1 Z_2$. Hence the evolution can be seen as $|\psi_t\rangle = e^{-i2tJ\pi Z_1 Z_2}|\psi_0\rangle$.

Intuitively, if the first spin is in the state $|0\rangle$, then the second spin rotates clockwise, if the first spin is in the state $|1\rangle$, then the second spin rotates *counter*-clockwise. By linearity of the quantum evolution this behaviour extends to any arbitrary state.

As we have already seen, the C-NOT together with arbitrary one-qubit gates is universal. In the last section we showed how to implement arbitrary one-qubit gates in NMR. An NMR pulse sequence to implement the C-NOT gate is given in inset 3.2.2.

The problem with the j-coupling is that it cannot be 'turned on and off' at will. It is a constant physical effect. There is a method, however, that corrects for the unwanted j-coupling evolution. It is called 'refocusing' and is fully discussed in inset 3.2.3.

3.2.3 Measurement

As mentioned earlier, the precession of the nuclei around the z-axis creates a magnetic field in the x-y plane. A coil, electronically tuned in to the frequency of this precession is able to measure the *overall* magnetic field created by a very large number of precessing nuclei.

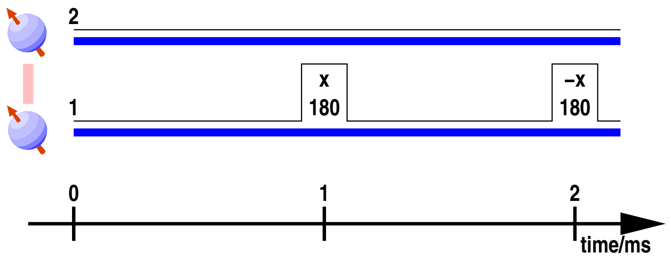
Since the different type of nuclei precess at very different frequencies, different coils tuned in to different frequencies are used to measure the amplitude and phase of the magnetic field induced over time.

Each coil then measures the magnetic field induced by all the nuclei of the same type. To actually obtain a value for each nucleus, some processing is needed.

For example, in the case of crotonic acid it is possible to attune a coil to the frequency of the protons (about 500 MHz, in an 11 Tesla field). We set up a reference frame in which the first proton is stationary. The second proton then rotates at a frequency of —about— 600Hz, since this is the difference of the Larmor frequencies of the two protons.

The Hamiltonian

The idea behind the 'refocusing' technique is to reverse the evolution of the second qubit due to the j-coupling for half the total time. By flipping the z value (NOT gate) of the first qubit at exactly the midpoint of the evolution, the first qubit will eventually return to its original value. A second gate at the end of the evolution ensures that both qubits have the values they started of with (see the picture below). Refocusing must occur in between all gates that involve either of the two qubits.



Mathematically, we have that the evolution of the second qubit due to its j-coupling to the first one is given by $e^{-it/2J\sigma_z}$ for the first half of the evolution, before the refocusing gate. After the refocusing gate the evolution is given by $e^{it/2J\sigma_z}$. The second operator cancels out with the first one. This is assuming that the first qubit begin in the state $|0\rangle$. If it begin in the state $|1\rangle$ just reverse the order of the operators, again they cancel out. By linearity the same will happen for any superposition of the first qubit.
It must be noted that in the mathematical analysis we have assumed the refocusing gate to be instantaneous. This is, of course, not achievable. In a real world implementation the gate would take some time t, during which the evolution of second qubit is not so easy to define, and could —in principle— be a source of errors.
A larger consideration is the fact that j-coupling (potentially) exists between each pair of nuclei. Fixing one pair could —in theory— break another.
The question arises then whether refocusing can be done in an efficient way, on the number of qubits. Jones and Knill prove that it is indeed possible, and show how, in [JK99].

Inset 3.2.3: Refocusing

$$H_{cs} = \pi 600\text{Hz}\ \sigma_z^{(2)}$$

is called the '*chemical-shift*' Hamiltonian.

Now, following [LKC$^+$02], since the coils are rotating around the qubits, as seen from the rotating frame, it is possible to measure both the magnetization in the x and y directions (actually in any direction in the x-y plane):

$$\begin{aligned} M_x(t) &= tr(\rho(t)(\sigma_x^{(1)} + \sigma_x^{(2)})) \\ M_y(t) &= tr(\rho(t)(\sigma_y^{(1)} + \sigma_y^{(2)})). \end{aligned}$$

and then combine them, using $\sigma_+ = \sigma_x + i\sigma_y$, to form a representation of the plane magnetization as a complex number:

$$\begin{aligned} M(t) &= M_x(t) + M_y(t) \\ &= tr(\rho(t)(\sigma_+^{(1)} + \sigma_+^{(2)})) \end{aligned}$$

and obtain a picture of the evolution of the two spin system over time.

Now, using the facts that under the chemical shift Hamiltonian

$$\rho(t) = e^{-i\pi 600\text{Hz}\ \sigma_z^{(2)} t} \rho(0) e^{i\pi 600\text{Hz}\ \sigma_z^{(2)} t}$$

the first spin remains invariant, we may write

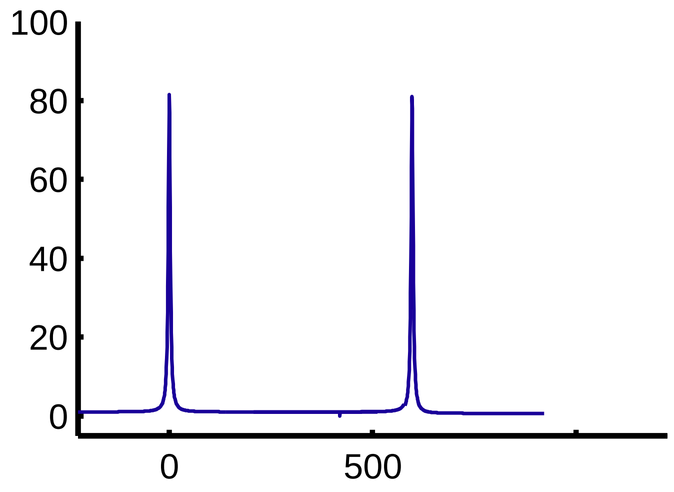


Figure 3.3: Chemical shift Hamiltonian Fourier analysis

$$\begin{aligned}
M(t) &= tr(\rho(t)(\sigma_+^{(1)} + \sigma_+^{(2)})) \\
&= tr(\rho(0)\sigma_+^{(1)}) + tr(e^{-i\pi 600\mathrm{Hz}\ \sigma_z^{(2)}t}\rho(0)e^{i\pi 600\mathrm{Hz}\ \sigma_z^{(2)}t}\sigma_+^{(2)}) \\
&= tr(\rho(0)\sigma_+^{(1)}) + tr(\rho(0)e^{i\pi 600\mathrm{Hz}\ \sigma_z^{(2)}t}\sigma_+^{(2)}e^{-i\pi 600\mathrm{Hz}\ \sigma_z^{(2)}t}) \\
&= tr(\rho(0)\sigma_+^{(1)}) + tr(\rho(0)e^{i2\pi 600\mathrm{Hz}\ \sigma_z^{(2)}t}\sigma_+^{(2)}).
\end{aligned}$$

Recall that a Fourier transform maps from the time-domain to the frequency domain. By taking the measurement of the x-y magnetization over time, and Fourier transforming it we should get two peaks, one at 0Hz (reference frame) for the first spin, and one at 600Hz representing the magnetization of the second spin, see figure 3.3.

We have conveniently disregarded the j-coupling between the two protons. The coupling Hamiltonian is given by:

$$H_j = \pi 15\text{Hz}\ \sigma_z^{(1)}\sigma_z^{(2)}$$

hence the full Hamiltonian is

$$H = \pi 600\text{Hz}\ \sigma_z^{(2)} + \pi 15\text{Hz}\ \sigma_z^{(1)}\sigma_z^{(2)}.$$

To analyze the spectrum of the full Hamiltonian we introduce the operators [LKC+02]:

$$\sigma_\uparrow = \frac{\mathbb{1} + \sigma_z}{2} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$
$$\sigma_\downarrow = \frac{\mathbb{1} - \sigma_z}{2} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Notice that both operators $\sigma_\uparrow$ and $\sigma_\downarrow$ are invariant under the full Hamiltonian, and that $\sigma_\uparrow + \sigma_\downarrow = \mathbb{1}$. Hence we have:

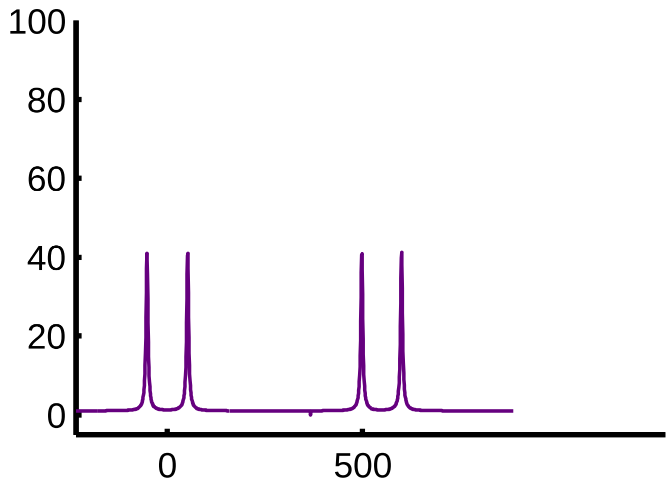


Figure 3.4: Full Hamiltonian Fourier analysis

$$\begin{aligned}
M(t) &= tr(\rho(t)(\sigma_+^{(1)} + \sigma_+^{(2)})) \\
&= tr(\rho(t)\sigma_+^{(1)}) + tr(\rho(t)\sigma_+^{(2)})) \\
&= tr(\rho(t)\sigma_+^{(1)}\mathbb{1}^{(2)}) + tr(\rho(t)\mathbb{1}^{(1)}\sigma_+^{(2)})) \\
&= tr(\rho(t)\sigma_+^{(1)}(\sigma_\uparrow^{(1)} + \sigma_\downarrow^{(1)})) + tr(\rho(t)(\sigma_\uparrow^{(2)} + \sigma_\downarrow^{(2)})\sigma_+^{(2)})) \\
&= tr(e^{-Ht}\rho(0)e^{Ht}\sigma_+^{(1)}(\sigma_\uparrow^{(1)} + \sigma_\downarrow^{(1)})) + tr(e^{-Ht}\rho(0)e^{Ht}(\sigma_\uparrow^{(2)} + \sigma_\downarrow^{(2)})\sigma_+^{(2)})) \\
&= tr(e^{-Ht}\rho(0)e^{Ht}\sigma_+^{(1)}\sigma_\uparrow^{(1)}) + tr(e^{-Ht}\rho(0)e^{Ht}\sigma_+\sigma_\downarrow^{(1)}) \\
&\quad + tr(e^{-Ht}\rho(0)e^{Ht}\sigma_\uparrow^{(2)}\sigma_+^{(2)}) + tr(e^{-Ht}\rho(0)e^{Ht}\sigma_\downarrow^{(2)}\sigma_+^{(2)}) \\
&= e^{i2\pi 15\text{Hz}}\ tr(\rho(0)\sigma_+^{(1)}\sigma_\uparrow^{(1)}) + e^{-i2\pi 15\text{Hz}}\ tr(\rho(0)\sigma_+\sigma_\downarrow^{(1)}) \\
&\quad + e^{i2\pi 600\text{Hz}\ +15\text{Hz}}\ tr(\rho(0)\sigma_\uparrow^{(2)}\sigma_+^{(2)}) + e^{i2\pi 600\text{Hz}\ -15\text{Hz}}\ tr(\rho(0)\sigma_\downarrow^{(2)}\sigma_+^{(2)}).
\end{aligned}$$

From the last equation we can see that the two peaks from figure 3.3 split into two (see figure 3.4). The first peak group having two peaks, one each for the $\sigma_+\sigma_\uparrow$ and $\sigma_+\sigma_\downarrow$ states, and the second two peaks corresponding to $\sigma_\uparrow\sigma_+$ and $\sigma_\downarrow\sigma_+$.

With n nuclei the spectral signal of each nucleus splits into 2^{n-1} peaks. For example, the 2^{n-1} peaks of the first nucleus' group represent the states $\sigma_+\sigma_\uparrow\sigma_\uparrow\ldots\sigma_\uparrow$, $\sigma_+\sigma_\uparrow\sigma_\uparrow\ldots\sigma_\downarrow$, $\ldots$ $\sigma_+\sigma_\uparrow\sigma_\downarrow\ldots\sigma_\downarrow$, $\sigma_+\sigma_\downarrow\sigma_\downarrow\ldots\sigma_\downarrow$ (assuming all peaks can be resolved in the spectrometer).

Recall, that in QIP the computational basis is precisely the states $\sigma_\uparrow\sigma_\uparrow\ldots\sigma_\uparrow$, $\sigma_\uparrow\sigma_\uparrow\ldots\sigma_\downarrow$, $\ldots$ $\sigma_\uparrow\sigma_\downarrow\ldots\sigma_\downarrow$, $\sigma_\downarrow\sigma_\downarrow\ldots\sigma_\downarrow$.

Hence, it is possible to measure the $n-1$ bits 2 to n in the computational basis, by analyzing the first nucleus' peak group. This fact will be used later on in section 3.2.4.

The true 'full' Hamiltonian of the two spin system, however, is not *really* just the sum of the chemical shift Hamiltonian and the j-coupling Hamiltonian. There is also the (non-unitary) evolution due to decoherence. This last evolution has the effect of exponentially decreasing the observable magnetization over time. Since the normal decay times for molecules used in QIP are in the range of 0.4 to 2s, the measurement process consists of sampling the magnetic field at several discrete intervals within that time frame.

Recapitulating, the coil is not able to measure the state of a single spin, rather it measures the collective, or 'bulk' magnetization of a very big number of nuclei (somewhere in the order of 10^{19}). As such this measurement is, in some ways, much more powerful than normal projective measurements. Whereas a projective measurement on a qubit gives only a single bit of information (zero or one) in NMR spectroscopy the measurement gives a 'good approximation' to an actual expected value $\bar{x_\phi} = \langle\phi|X|\phi\rangle$ for each measured qubit $|\phi\rangle$. In seeming paradox this type of measurement is called 'weak measurement'. This is because the measurement apparatus is not strong enough to cause a significant collapse of the measured qubits (of the whole ensemble) to one of its eigenstates.

What a 'good approximation' actually means is of great computational importance. Whereas a quantum computer model that give a polynomial size approximation (in the size of the input) to the expected value of the output is provenly equivalent to a quantum computer model that gives only simple projective measurement outputs (see [KL98]), a model computer that can give an exponentially close approximation (or better) is not. This later model can easily be shown to be much more powerful (at least in the black box model).

As we saw in chapter 1 this seems to hold in the classical realm. BPP, the class of problems that can be solved in polynomial time by an algorithm that gives the expected value up to polynomial precision, is believed to be much smaller than PP, the equivalent class that has algorithms that give exponentially good approximations.

It seems that the approximation to the expected value of the output of any arbitrary quantum algorithm should be exponentially good. This in light of the fact that the number of actual trials is around 10^{19}, whereas —so far— the number of qubits does not exceed 10.

If this were so, NMR could be used to solve (small instances of) problems in PP, which are deemed intractable, without even taking advantage of quantum superposition.

However, the approximation is not as good as would be expected. One of the reasons, is as stated before, the accuracy at which the spectrometer is able to read the plane magnetization of the sample. Another reason is —at least at present— the way the initial state is prepared.

Without dwelling too much on the subject we should further note that even if these hurdles were to be overcome, using NMR spectrography in this fashion would be no different from classical massive parallelism. There is no real gain, only a tradeoff between space and time. Much like molecular computation (see for example [Adl94]), the exponential growth of the problems in PP eventually catch up with the sample size, and surpass it.

3.2.4 State Preparation

The molecules dissolved in the NMR sample are in constant flux. The state of each nucleus of each molecule are all in very different states. As the observable magnetization depends on the overall magnetic moment induced by all the nuclei precessing at the same frequency, it is necessary to represent the 'overall' state of sample as a mixed state ρ.

However, normally in QIP, especially quantum algorithms, the state of the system at all times is expected to be in a pure state; furthermore, the initial state of the quantum register is generally expected to be $|0\rangle$.

It seems that what is needed is to somehow pre-process the sample to force it into the state $|0\rangle$. Although methods for doing so have been proposed, none of them seem to be implementable with current technology.

This could lead us to —wrongly— assume that that quantum computation is not possible in the setting of NMR. This is not the case.

Although truly pure states are difficult to achieve in the context of NMR, it is possible to do computation using a different type of approach; the so-called '*pseudo*-pure' states.

3.2.5 Pseudo-pure States

Normally in quantum algorithms we induce a time-dependent Hamiltonian on the system, which we assume to start off in the state $|0\rangle^{\otimes n}$, and perform a projective measurement in the computational basis at the end of the calculation.

There is another way to view this. If the time-dependent evolution of the system (i.e, the complete series of quantum gates) is represented by the unitary matrix U then we can solve our problem by obtaining the value of $\langle z \rangle = \langle 0|U^\dagger \sigma_z U|0\rangle$. If the value of $\bar{z}$ is greater than zero, than the most likely outcome would be one. For problems in EQP we would expect the above value $\langle z \rangle$ to be either a one (measured outcome is zero with probability one) or minus one (the algorithm outputs one, with probability one).

In the mixed state formalism the above procedure translates into having the initial state $\rho = |0\rangle\langle 0|$ and performing the measurement $\bar{z} = Tr(\sigma_z U|0\rangle\langle 0|U^\dagger)$.

As stated earlier, the measurement apparatus allows us to obtain an actual approximation to the value $\langle z \rangle$ to be obtained.

An important observation is that if the density matrix is the identity then no magnetization is observable. This is because for any projective measurement A performed on the identity, the mean value $\langle A\rangle$ is $\langle A\rangle = Tr(IA) = Tr(A) = 0$ (since all our observables have trace zero).

Now, suppose the state of the sample is mostly, but not completely in the state I. A density matrix γ is said to have deviation ρ if it is of the form $\gamma = (1-\epsilon)\mathbb{1}+\epsilon\rho$ In this case γ is said to be a *deviation* density matrix for ρ.

The mean value of the observable A on the state γ depends only on its deviation ρ, since

$$\begin{aligned}\langle A_\gamma\rangle &= Tr(\gamma A)\\ &= Tr(((1-\epsilon)\mathbb{1}+\epsilon\rho)A)\\ &= Tr(((1-\epsilon)\mathbb{1})A) + Tr(\epsilon\rho A)\\ &= \epsilon Tr(\rho A).\end{aligned}$$

Another important property of the completely random state $\mathbb{1}$ is that it remains random under any unitary transformations U, since $\mathbb{1} \to U\mathbb{1}U^\dagger = UU^\dagger = \mathbb{1}$.

A deviation density matrix γ, with deviation ρ, will become a deviation matrix for $U\rho U^\dagger$ under the unitary operator U, since:

$$\begin{aligned}\gamma &\to U\gamma U^\dagger \\ &= U((1-\epsilon)\mathbb{1} + \epsilon\rho)U^\dagger \\ &= (1-\epsilon)U\mathbb{1}U^\dagger + \epsilon U\rho U^\dagger \\ &= (1-\epsilon)\mathbb{1} + \epsilon U\rho U^\dagger.\end{aligned}$$

In short, a deviation matrix for the state ρ behaves (almost) just like the actual state ρ for all purposes of QIP. If ρ is a pure state $|\psi\rangle\langle\psi|$ then the deviation density matrix $\gamma = (1-\epsilon)\mathbb{1} + \epsilon|\psi\rangle\langle\psi|$ is called a '*pseudo*-pure' state, since it acts, for all practical purposes, like the pure state $|\psi\rangle$.

Now, the solution seems clear. Although forcing the NMR solution into the totally pure state $|0\rangle^{\otimes n}$ is virtually impossible, creating a pseudo-pure state with *deviation* $|0\rangle\langle 0|^{\otimes n}$ is at the reach of current technology.

However, in practice, the pseudo-pure state usually strived for is the one with deviation $\sigma_x \otimes |0\rangle\langle 0|^{\otimes n}$, where we have added one extra qubit. This can be seen as the standard initial state $|0\ldots 0\rangle$ on n qubits, by simply disregarding the first qubit.

Recall that in section 3.2.3 we saw that by measuring the magnetization of the first qubit it is possible to obtain the state of the n remaining qubits. This is what is done in current experiments [LKC$^+$02].

The problem with such a method is that it does not scale. Recall again our discussion in section 3.2.3. Say the total polarization of the first qubit (the one we measure) is K. If there are two distinguishable states of the remaining qubits then there are two peaks for the first qubit, one for each state, *but each peak is roughly half the magnitude of the original.* If there are four distinguishable states, then each peak is roughly one fourth the size of the original. Hence we have that for a pseudo-pure state with n qubits, the observable magnetization is decreased by $\frac{1}{2^n}$. Hence it is impossible to have too many qubits before the observable magnetization is indistinguishable from noise.

Other methods have been proposed. For example, Schulman and Vazirani give one such proposal in [SV99]. Unfortunately, such method, though scalable in theory, is not yet implementable in practise.

Another possibility is to dispense with pure states altogether, and use a different model of computation based (almost) entirely on mixed states. One such proposal is DQ1, introduced by E. Knill and R. Laflamme [KL98].

DQ1 is, at first sight, similar to BQP. The same rules of state evolution, and final measurement apply. However, in DQ1, as opposed to BQP, only one qubit is allowed to begin in a pure state $|0\rangle$, all other qubits begin in the completely mixed state $\mathbb{1}$.

Though DQ1 is shown to be more powerful than BPP (in the black box model), it seems to be less powerful than BQP.

It seems then, that efficient state initialization is one of today's most important obstacles for scalable NMR QIP. However, state initialization is not the most fundamental obstacle for NMR QIP scalability. Even with perfect gates, perfect pure state initialization, and perfect measurement apparatus, there are still obstacles to be overcome if a scalable NMR quantum computer is to be built.

One such obstacle is that, opposed to other quantum computer proposals that use spatial separation as a means to differentiate qubits, NMR relies on frequency separation. There are just so many distinguishable frequencies that can be used.

However, as we stated in the beginning of this work, NMR QIP is *today* the most advanced implementation of a quantum computer, and it can be used *today* as a tool to study QIP.

Chapter 4

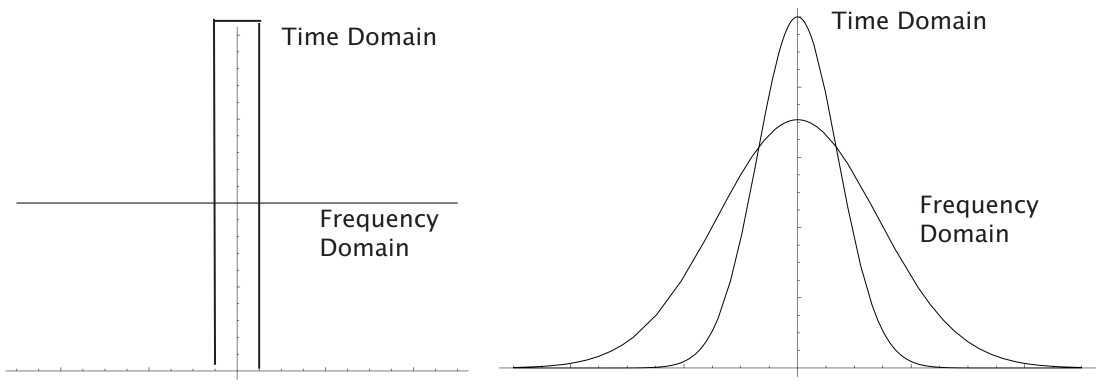
Pulse Compiler

As was discussed in the last chapter, creating a gate out of a sequence of radio frequency pulses is complicated and error prone.

For example if one wishes to implement a ninety degree rotation of the first qubit about the x-axis then, in principle one could apply a RF pulse in resonance with the target qubit. For example if the pulse induces the Hamiltonian $\omega_x \sigma_x/2$, then the pulse must be applied for time t where $t = \pi/2\omega_x$, so that $e^{-i\omega_x \sigma_x t/2} = -i\sigma_x$ (the global phase is ignored).

Such pulses are called *hard pulses*. The problem with such an approach is that crosstalk, that is the unwanted effect of the RF pulse on other pulses is very big. To see why this is so, consider a pulse that is very strong, for a short period of time. If we see the same pulse in the frequency domain (by taking its Fourier transform), and plot its strength, then we see that it effects all nuclei of the same type about evenly (see figure 4).

An alternative is to use so-called '*soft*' pulses, also called '*shaped*' pulses. The



The picture on the left shows a hard pulse, which is on for one discrete time step. Note that in the frequency domain the pulse is constant, and hence affects all nuclei (of the same type) equally. The picture on the right shows a soft, Gaussian envelope type, pulse. Note that the pulse is a Gaussian centred about the desired frequency.

Figure 4.1: Hard vs. Soft Pulses

idea is to modulate the pulse strength over time such as to minimize the effect on other spins, or crosstalk. Perhaps the best shape to use is the so-called 'Gaussian Envelope' defined by

$$S(t) = e^{-a(t-t_0)^2} \tag{4.1}$$

where t_0 is the centre of the pulse envelope. The reason for this is the general theorem of Fourier transforms: if the n first derivatives of $S(t)$ are all continuous, then its Fourier transform decreases inversely as the nth power of frequency in the tails [Fre98], hence the Fourier transform of the pulse ends up being a Gaussian too, centred about the desire frequency, see figure 4.

Even with such a technique it is difficult, if not impossible, to eliminate crosstalk absolutely.

Many techniques have been developed in order to cope with errors during quantum computation.

NMR, however, has a very specific scenario when it comes to gates. Recall, that whereas rotations in the x-y plane are prone to error, z-axis rotations, and two qubit zz-axis rotations are, in a way, ‘perfect’ and ‘free’, since no actual quantum operation is involved.

In the case of single qubit σ_z gates just an update of the classical (as opposed to quantum) values of the rotating frame is necessary. Such an operation can be done classically in a much more robust way. As for two qubit $\sigma_z \otimes \sigma_z$ gates, all that is necessary is a time delay. The inherent j-coupling will perform the desired operation without outside intervention.

Hence, in NMR QIP we can, in principle, freely introduce z, and zz rotations wherever we see fit, in order to minimize induced errors.

Such method is the central idea of this work, and the topic of this chapter. Before proceeding it is important to stress that this is *not* quantum error correction. Rather it is error *prevention.*

In order to fully understand the inner workings of the pulse sequence compiler discussed in this chapter it is important to understand the *product operator* formalism. This is discussed in the next section.

Following that we give an in depth look of the pulse sequence compiler.

4.1 Product operator formalism

We have already seen one method of describing general transformations of quantum systems: as a unitary, complex-scalar, matrix σ.

For single qubit systems we have also seen that it is possible to describe transformations as a rotation angle θ and a direction axis $\vec{n} = (n_x, n_y, n_z)$.

The former representation, while a general description that applies to any Hilbert space (even infinite dimensional), has the flaw that it is not at all intuitive: it is difficult to obtain a mental picture of the actual transformation from just looking at the representing matrix σ.

The latter has the distinct advantage of clearly showing the outcome of performing the given transformation, in a visual form as seen in the Bloch sphere picture of a single qubit system. However, the fact that it only works for single qubit systems is somewhat of a disadvantage.

Fortunately, since 1983 a generalization of the Bloch sphere to many qubit systems has been known. It was first described by Sørensen et. al. in [rEL+83].

Here, we give a short description of the formalism.

Recall, how the Pauli matrices $\sigma_i, \sigma_x, \sigma_y, \sigma_z$ form a basis for the four-dimensional vector space $M_{(2,2)}$. The group of all operators acting on one qubit, is a subset of such space, hence all one qubit operators can be written as a unit-length linear combination of the Pauli operators.

By writing the operator as a linear combination we obtain the Bloch sphere picture of the operator in question.

It is possible to define a similar basis for operators acting on an arbitrary number of 1/2-spin's. For an n 1/2-spin system there are 4^n basis elements b_i, and each element is defined as:

$$b_i = 2^{q-1} \prod_{k=1}^{q} \sigma_v^k. \tag{4.2}$$

Each basis element is called a '*product operator*' [rEL+83].

For example for a one qubit system the product operators are:

$$\begin{aligned} q = 0: &\quad \frac{1}{2}\mathbb{1}, \\ q = 1: &\quad \sigma_x, \sigma_y, \sigma_z. \end{aligned}$$

Hence, for a one 1/2 spin system the product operator formalism reduces (roughly) to the Bloch sphere picture of a transformation: that of a rotation in three-space. Each operator corresponds to a specific rotation along an axis.

For a two qubit system the product operators are:

$$\begin{aligned} q = 0: &\quad \frac{1}{2}\mathbb{1}, \\ q = 1: &\quad \sigma_x^{(1)}, \sigma_y^{(1)}, \sigma_z^{(1)}, \sigma_x^{(2)}, \sigma_y^{(2)}, \sigma_z^{(2)}, \\ q = 2: &\quad 2\sigma_x^{(1)}\sigma_x^{(2)}, 2\sigma_x^{(1)}\sigma_y^{(2)}, 2\sigma_x^{(1)}\sigma_z^{(2)}, \\ &\quad 2\sigma_y^{(1)}\sigma_x^{(2)}, 2\sigma_y^{(1)}\sigma_y^{(2)}, 2\sigma_y^{(1)}\sigma_z^{(2)}, \\ &\quad 2\sigma_z^{(1)}\sigma_x^{(2)}, 2\sigma_z^{(1)}\sigma_y^{(2)}, 2\sigma_z^{(1)}\sigma_z^{(2)}. \end{aligned}$$

As in the one qubit case, the single spin operators ($q = 1$) correspond to rotations of individual spins, about a specific axis in the Bloch sphere of the individual spin (the three dimensional subspace of the whole two-qubit space). Two spin operators can also be seen as rotations, but now along an axis in the bigger space. Hence, the operator $2\sigma_z^{(1)}\sigma_z^{(2)}$ corresponds to a rotation along the 'zz axis', and clearly corresponds to a two-spin J-coupling.

By representing transformations in this basis, it is easier, both for human and computer observers, to quickly obtain rotational effects of unitary matrices.

4.2 Pulse Compilation

The requirements of pulse compilation is, roughly, the following.

First, single gates/pulses are performed experimentally on the target solution on which a quantum algorithm is to be performed. From this data, the error of individual gates is obtained.

This information, along with the information pertaining to the molecule used as the quantum register, and pulse sequences to be used in the algorithm is fed into the pulse compiler.

For example, let the input be the desired pulse effect $e^{-iU_{ideal}}$ and the actual pulse effect obtained experimentally is $e^{-iU_{actual}}$.

The compiler must output a series of single qubit gates

$$e^{-i\theta_{pre}^{(1)}\sigma_z^{(1)}}, e^{-i\theta_{post}^{(1)}\sigma_z^{(1)}}, e^{-i\theta_{pre}^{(2)}\sigma_z^{(2)}}, e^{-i\theta_{post}^{(2)}\sigma_z^{(2)}}, \ldots, e^{-i\theta_{pre}^{(n)}\sigma_z^{(n)}}, e^{-i\theta_{post}^{(n)}\sigma_z^{(n)}},$$

and a series of two qubit gates,

$$e^{-i\theta_{pre}^{(1,2)}\sigma_z^{(1)}\sigma_z^{(1,2)}}, e^{-i\theta_{post}^{(1,2)}\sigma_z^{(1)}\sigma_z^{(2)}}, e^{-i\theta_{pre}^{(1,3)}\sigma_z^{(1)}\sigma_z^{(3)}}, e^{-i\theta_{post}^{(1,3)}\sigma_z^{(1)}\sigma_z^{(3)}}, \ldots,$$
$$e^{-i\theta_{pre}^{(n-1,n)}\sigma_z^{(n)-(1)}\sigma_z^{(n)}}, e^{-i\theta_{post}^{(n-1,n)}\sigma_z^{(n-1)}\sigma_z^{(n)}}$$

where n is the number of nuclei in the register molecule. So that

$$R = \prod_{i=1}^{n} e^{-i\theta_{post}^{(i)}\sigma_z^{(i)}} \prod_{i=1}^{n-1}\prod_{j=i}^{n} e^{-i\theta_{post}^{(i,j)}\sigma_z^{(i)}\sigma_z^{(j)}} e^{-iU_{actual}} \prod_{i=1}^{n-1}\prod_{j=i}^{n} e^{-i\theta_{pre}^{(i,j)}\sigma_z^{(i)}\sigma_z^{(j)}} \prod_{i=1}^{n} e^{-i\theta_{pre}^{(i)}\sigma_z^{(i)}}$$

best approximates $e^{-iU_{ideal}}$. Formally, we minimize

$$D(R, e^{-iU_{ideal}})$$

where $D(A, B)$ is defined to be the Manhattan metric, also known as the 'city block' distance metric, for matrices, that is:

$$D(A, B) = \sum_{i.j} |A_{i,j} - B_{i.j}|$$

This formula was chosen because it is a well-defined, and well-behaved distance metric, and it is computational efficient to compute.

Note that, since all σ_z and $\sigma_z\sigma_z$ commute, the order of the pre and post gates do not matter.

For composite pulses, where the gate consists of several pulses one after the other, the compiler finds a set of σ_z and $\sigma_z\sigma_z$ gates to put in between each pulse. For example, if the pulse effect sequence is $e^{-iU_1}, e^{-iU_2}, \dots, e^{-iU_n}$, then the compiler finds sequence of operators $E_0, E_1, \dots, E_n$, where each E_i is of the form

$$\prod_{i=1}^{n} e^{-i\theta^{(i)}\sigma_z^{(i)}} \prod_{i=1}^{n-1}\prod_{j=i}^{n} e^{-i\theta^{(i,j)}\sigma_z^{(i)}\sigma_z^{(j)}}$$

so that the pulse sequence

$$E_n e^{-iU_n} E_{n-1} e^{-iU_{n-1}} \dots e^{-iU_2} E_1 e^{-iU_1} E_0$$

best approximates the desired evolution, in the same formal meaning as above. In the next section we discuss how this is actually done.

4.3 Pulse Compiler Algorithm

There is a lot of information to process in order to compose a complete pulse sequence. Three input files must be parsed. First, there is the nuclear data file. This contains all the information pertinent to the molecule used as the quantum register. It gives the different type of nuclei (hydrogen, carbon, etc.) in the molecule, and a reference precession frequency for each type. It lists each nuclei, its type, and its Larmor frequency, given with respect to the reference frequency for its type. It also gives the j-coupling constants for each pair of nuclei. Inset 4.3.1 shows an excerpt of the nuclei file for Crotonic Acid.

From this file we can read that the reference frequency for the Hydrogen atoms is 500.13×10^6 Hertz. The Larmor frequency of the first Hydrogen atom is -2370.80

```
# Nuclear types
500.13e6 H
125.757739e6 C
# Nuclei: <frequency><tab><name><tab><type>
  -781.06 RH H
   230.43 M H
 -2370.80 H1 H
 -1774.47 H2 H
 -6204.60 RC C
 -1914.06 C1 C
-18115.10 C2 C
-15157.41 C3 C
-21148.90 C4 C
# Reference frequencies:
refFreq H M;H1;H2
refFreq C C1;C2;C3;C4
offsetFreq H 0
# Couplings:  <frequency><tab><name><tab><name>
    7.72 H1 H2
    0.00 H1 RC
    1.98 H1 C1
   77.97 H1 C2
   -0.91 H1 C3
    3.25 H1 C4
    3.45 M H1
   -0.88 M H2
  .
  .
  .
  # End data.
```

Inset 4.3.1: Example nuclei file

relative the previous frequency, and its j-coupling constant with the second Hydrogen is 7.72.

Then there is the pulse shape information. This gives the information about

```
# Shape name<tab>
#   number of points<tab>
#     t/(t for angle at max power) for the intended use.<tab>
#       calibration angle.<tab>
#         spectral width
###
isech180        128         2.4150418205742e+00 0.500000 2.360000
xsech180        128         5.0406945915562e+00 0.500000 4.520000
isech270        128         2.2150050633202e+00 0.750000 3.100000
isech90         128         2.2618000209690e+00 0.250000 2.250000
ssquare4        4           1.0000000000000e+00 0.250000 10.000000
ssquare8        8           1.0000000000000e+00 0.250000 10.000000
rfsel32p        972         2.4300000000000e+02 0.250000 1.000000
rfsel32n        972         2.4300000000000e+02 0.250000 1.000000
rfsel64p        1932        4.8300000000000e+02 0.250000 1.000000
rfsel64n        1932        4.8300000000000e+02 0.250000 1.000000
rfsel100p       3012        7.5300000000000e+02 0.250000 1.000000
rfsel100n       3012        7.5300000000000e+02 0.250000 1.000000
```

Inset 4.3.2: Example Shape Definition file

the shapes of each pulse. Inset 4.3.2 gives a detailed file. The name of the shape, and the number of points are two very important items. The name of the pulse is the same name of a binary file where the actual pulse is defined. For example, there is a file, called isech180, that contains 128 pairs of numbers, representing the phase and the power of the pulse at each of the 180 time-steps.

Finally, there is the gate information itself. Each gate consists of a sequence of pulses defined in the shape file. An example file is given in inset 4.3.3. For basic pulses, the shape of the pulse is given. For example, a 90 degree rotation around the x-axis of the first hydrogen atom (pulse H1_90) has the shape defined in isech90.

For composite and parallel pulse sequences, the names of the individual pulses are given. For example, the composite pulse Hhc_180 consists of five individual

```
.
.
.
basic M_90 soft isech90 .25 M .25 M M_90 1 M R
basic H1_90 soft isech90 .25 H1 .25 H H2_90 1 H1 R
basic H2_90 soft isech90 .25 H2 .25 H H2_90 1 H2 R
basic C1_90 soft isech90 .25 C1 .25 C1 C1_90 2 C1 R
basic Hh_180 hard ssquare8 .5 M;H .5;.5;.5 vals.hard90H*2 Mh_180 1
M;H R
basic RHh_180 hard ssquare8 .5 RH .5 vals.hard90H*2 Mh_180 1 RH
M;H;C
.
.
.
composite Hhc_180 5 M;H1;H2 .5;.5;.5 0;0;0 R
Hh_180 vals.minPrePulse vals.minPostPulse 1/12 0
Hh_180 vals.minPrePulse vals.minPostPulse 0.0000 0
Hh_180 vals.minPrePulse vals.minPostPulse 0.2500 0
Hh_180 vals.minPrePulse vals.minPostPulse 0.0000 0
Hh_180 vals.minPrePulse vals.minPostPulse 1/12 0
composite RHhc_180 5 RH .5 0;0;0 M;H;C
RHh_180 vals.minPrePulse vals.minPostPulse 1/12~RH 0
RHh_180 vals.minPrePulse vals.minPostPulse 0.0000~RH 0
RHh_180 vals.minPrePulse vals.minPostPulse 0.2500~RH 0
RHh_180 vals.minPrePulse vals.minPostPulse 0.0000~RH 0
RHh_180 vals.minPrePulse vals.minPostPulse 1/12~RH 0
```

Inset 4.3.3: Example Pulse Definition file

pulses, each of type Hh_180.

All of these files must be read in and parsed before any real computation can be done. After this is done each pulse that is to be processed (the list of pulses can be passed to the program as an inline argument) is computed in turn. Basic pulses, or hard pulses, can be processed immediately. Parallel pulses, where several RF-pulses act on different spins at the same time, cannot be processed until the individual pulse constituents have been analyzed. Likewise for composite pulses.

The top level algorithm is given as a flow-chart in figure 4.2.

The procedure to actually process each pulse is as follows. Since the Hilbert space of the whole molecule is in general, very big (a density matrix of the system uses 2^{2n} complex numbers, where n is the number of nuclei in the register), it is generally impossible to actually simulate the effects of the pulse on the whole molecule in one go.

Instead, what is done is to first calculate the independent evolution of each nuclei separately due to the RF-pulse, and calculate the desired pre and post σ_z gates.

Then, for each pair of nuclei the evolution is calculated, taking into account the j-coupling between the nuclei. An optimal pre and post $\sigma_z\sigma_z$ gate is then calculated.

It is possible to do this processing in steps due to the fact that operators acting on different qubits always commute.

The pulse processing algorithm is depicted in figure 4.3.

This algorithm uses two procedures yet to be defined. The first one, is the actual simulation of the pulse sequence on the desired spin. The procedure to do this is outlined in figure 4.4. The idea of the algorithm is as follows. First, instead of attempting to find the evolution operator for the full time-dependent Hamiltonian, the evolution is discretized. That is, the time lapse of the pulse sequence is split into small discrete intervals, such that the Hamiltonian of the system at each of the intervals is constant. The time evolution operator for each constant Hamiltonian is found, and applied in turn. The final state is then output.

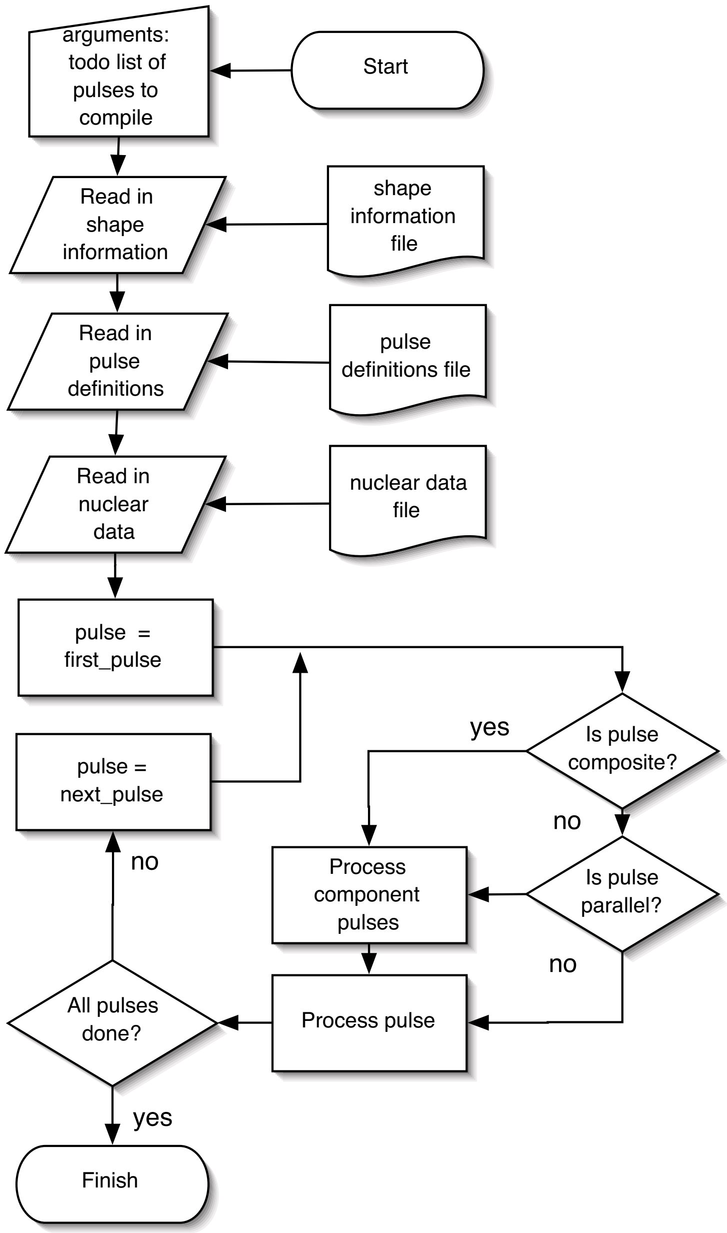


Figure 4.2: Top-level Pulse compiler flow-chart

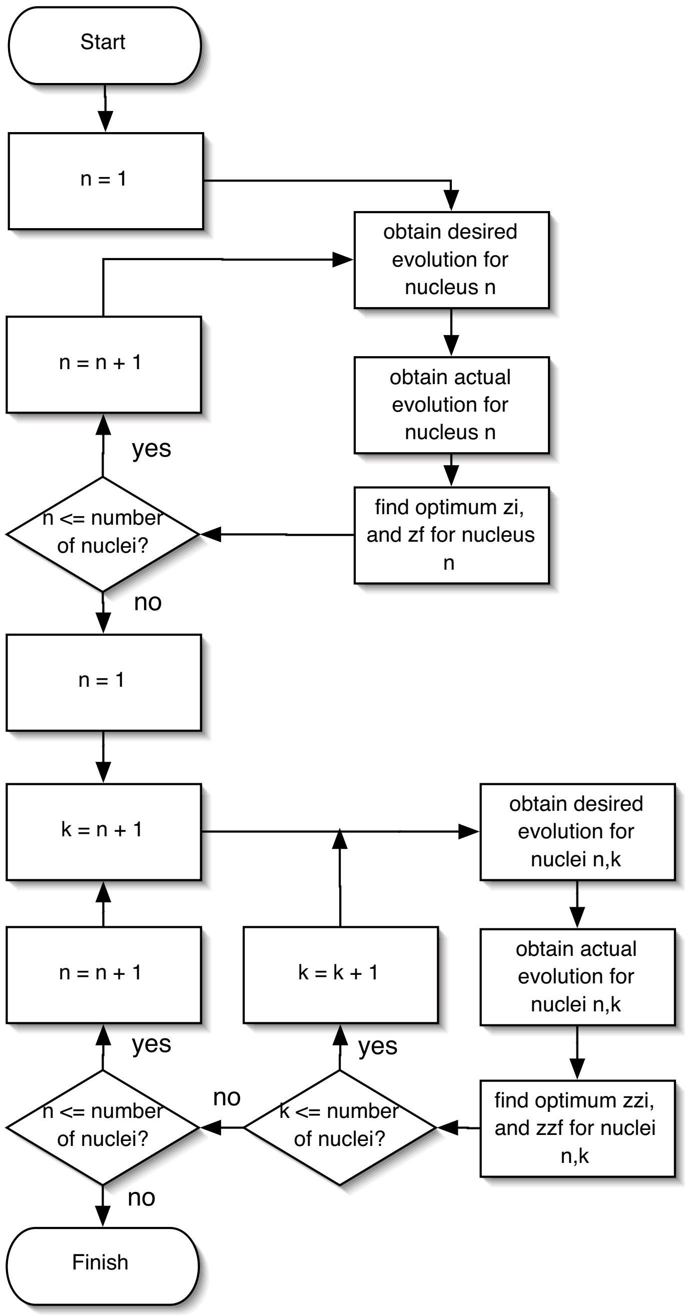

Start
n = 1
obtain desired evolution for nucleus n
n = n + 1
obtain actual evolution for nucleus n
yes
n <= number of nuclei?
find optimum zi, and zf for nucleus n
no
n = 1
k = n + 1
obtain desired evolution for nuclei n,k
n = n + 1
k = k + 1
obtain actual evolution for nuclei n,k
yes
yes
no
n <= number of nuclei?
k <= number of nuclei?
find optimum zzi, and zzf for nuclei n,k
no
Finish

Figure 4.3: Pulse processing flow-chart

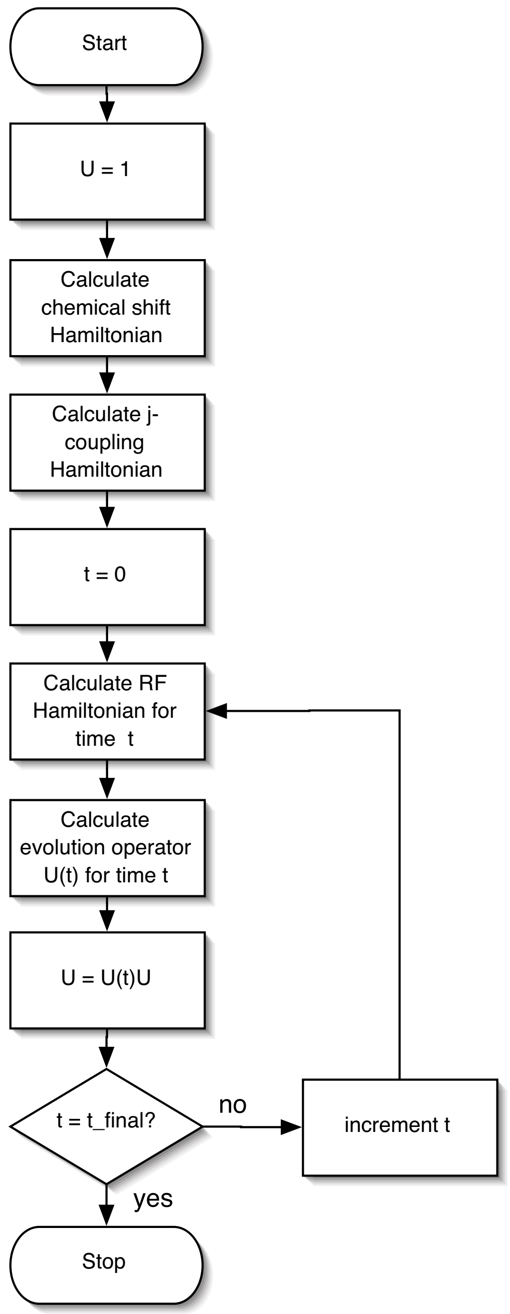

Start
U = 1
Calculate chemical shift Hamiltonian
Calculate j-coupling Hamiltonian
t = 0
Calculate RF Hamiltonian for time t
Calculate evolution operator U(t) for time t
U = U(t)U
t = t_final?
no
increment t
yes
Stop

Figure 4.4: Simulation of RF Pulse sequence flow-chart

Now, since the evolution due to the chemical shift and j-couplings is constant throughout the pulse sequence, the chemical shift, and j-coupling Hamiltonian is calculated first. Then for each interval the evolution due to the active RF pulses is calculated. Then the evolution operator is calculated from the complete Hamiltonian. The final state of the system is then slowly constructed from these time-step operators.

Now that the actual evolution has been calculated it is necessary to compare it with the desired evolution, and find the initial and final z and zz rotations we are looking for.

The desired evolution is always given, since it is part of the gate definition, so only the comparison remains to be done. This is where the product operator formalism becomes invaluable.

Both the actual evolution operator, and the desired evolution operator are transformed to coordinates with respect to the product operator basis. This can easily be done by taking the trace of the product of the evolution operator in question and the basis elements.

In this basis, the optimization process we wish to achieve is reduced to a simple projection. In order to analyze why this is true, let us first analyze the case of a single qubit.

In this case the only operation that we allowed to do is a rotation along the σ_z axis. Therefore, in order to find the desired rotation, what we do is project both the ideal and the actual pulses to the σ_z axis (recall that we are working in product-operator basis), the difference of these two projections, is the operator that will bring U_{actual} closest to U_{ideal}.

Now, from the product operator representation of a unitary matrix it is relatively easy to obtain the representation as a rotation on the x-y plane conjugated with z-rotations. That is an operator of the form $e^{-i\theta_z\sigma_z}e^{-i\cos\phi\sigma_x+\sin\phi\sigma_y}e^{-i\theta_z\sigma_z}$.

Let $U_{ideal} = e^{-i\theta_{z_i}\sigma_z}e^{-i\cos\phi\sigma_x+\sin\phi\sigma_y}e^{-i\theta_{z_i}\sigma_z}$, and
$U_{actual} = e^{-i\theta_{z_a}\sigma_z}e^{-i\cos\phi\sigma_x+\sin\phi\sigma_y}e^{-i\theta_{z_a}\sigma_z}$, then the sought after value z is $\theta_{z_i} - \theta_{z_a}$.

In the case for pairs of qubits, we repeat the same analysis —and algorithm— except that now we allow ourselves to move in the $\sigma_z - \sigma_z \otimes \sigma_z$ plane. Therefore we project onto this plane, repeat the same process outlined above.

4.4 Pulse Compiler Code

There are currently two versions of the pulse compiler in active development. A Matlab version which is based on Emanuel Knill's original code for Crotonic acid, and a prototype C/C++ version that only implements some parts of the program.

Since the merits of a native C version are not yet clear we shall discuss only the Matlab version here. It should be noted that Matlab counts with a 'compiler' that translates Matlab code to C; it is yet to be thoroughly tested though.

There are more than thirty program files in the Matlab version of the pulse compiler; the main program files being `mkPulse`, `readNuclei`, `readShapes`, `readPulseDefs`, `sysOpx`, and `effOpSx`.

See figure 4.5 for a complete 'birds-eye' picture of the code structure.

The main program file is `mkPulse`. Its arguments are: the list of pulses to be processed, and (a possibly empty) list of nuclei to ignore during the processing (this

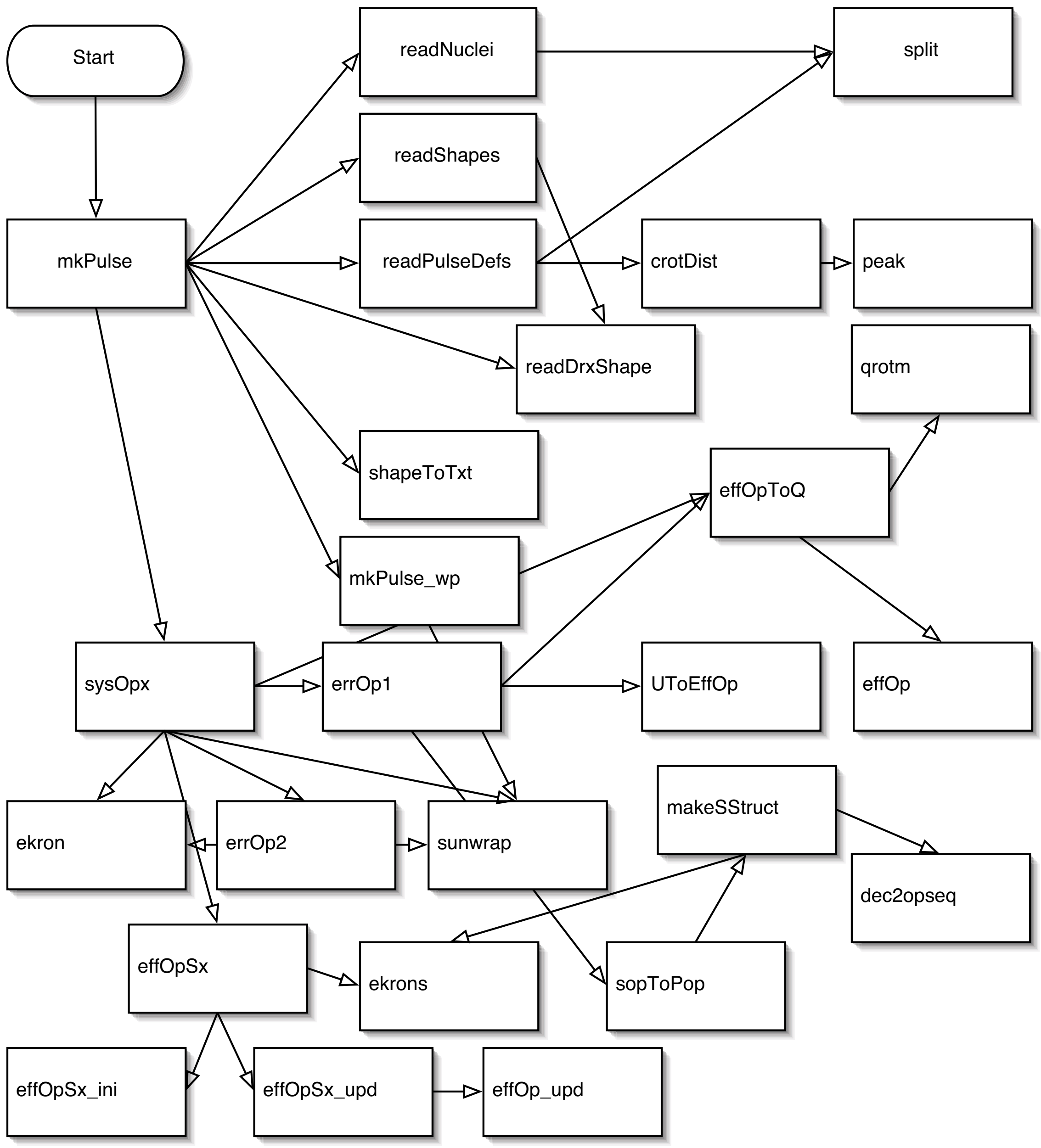


Figure 4.5: Bird's Eye view of the Pulse Compiler

is, nuclei that won't be taken into account when calculating errors.). It proceeds as follows (some miscellaneous steps are omitted for clarity).

First it calls `readNuclei` to read in the nuclear data file. This file defines the basic information of the molecule to be used as the quantum register, such as nucleus types, Larmor frequencies of each nucleus etc.

It then calls `readShapes`. This program file reads in the definition of the shapes of all soft pulses.

The module `mkPulse` then calls `readPulseDefs` to read in the pulse sequence definitions. These are the pulses that are to be processed. Which ones out of this list are actually processed is determined by the arguments to `mkPulse`.

After all data has been read in `mkPulse` creates a to-do list of pulses to be processed. Initially this list is set to be the list passed in as argument. However, if any pulse sequence in the to-do list depends on a second pulse sequence (say, a 180 pulse might be composed of two 90 pulses) then the latter pulse sequence is added to the list. This is done repeatedly until all dependencies are resolved.

Next, each pulse sequence in the to-do list is processed in order.

Each pulse sequence is categorized as either basic, composite, or parallel.

Some preprocessing of the pulse varies depending on the type of pulse. However, regardless of the type, the main step is invoking `sysOpx` on the pulse. The module `sysOpx` takes as input the pulse sequence, and the 'ideal' rotation it is trying to achieve. It returns a set of pre and post z rotations that best close the gap between the true and ideal operation, and the minimized error.

The program module `sysOpx` operates as follows. For each nuclear spin it calculates the best pre and post z rotations, and the total error. It does so by first calling `effOpSx` to simulate the actual pulse and determine its effects on the nucleus in question. It then calls `errOp1` to calculate the best initial and final z rotations, and the error incurred.

Then for each *pair* of nuclei `sysOpx` obtains a best initial and final zz rotation, again by simulating the pulse sequence and examining the effects on the two nuclei in question using `effOpSx`, and then comparing this with the 'ideal' rotation using, this time `errOp2`.

The module `effOpSx` runs in the following way: First, it determines the chemical shift Hamiltonian, and j-coupling Hamiltonian from nuclear information. Then it uses the pulse sequence information to calculate the RF Hamiltonian, that is, the evolution of the spin-states under the influence of the RF-pulses.

It then combines the effects to create the overall Hamiltonian of the system. Finally, it simulates the evolution the system by evolving the system by discrete intervals.

Finally, `errOp1` operates by first transforming both the ideal rotation and the actual pulse sequence rotation to the form $e^{-i\theta_t Z} e^{-i\cos\alpha X + sin\alpha Y} e^{-i\theta_i Z}$. That is, a rotation along the z-axis, followed by a rotation on the x-y plane, followed by a second rotation on the z-axis (it is easy to prove that this is always possible). The wanted initial z rotation is then the difference of the initial z rotation θ_i of the ideal pulse, and that of the actual pulse sequence. Similarly for the wanted final z rotation.

This concludes the overview of the pulse compiler. For the complete documented source, as well as user, and install guides, please visit

http://www.math.uwaterloo.ca/∼caperezd/research/pulse_compiler.

Chapter 5

Conclusions and Further Work

Without a doubt, the most important, pressing, and interesting question that remains in the air is whether or not NMR spectroscopy is a *scalable* quantum computer architecture.

In short, the answer is *'we don't know'*. We have addressed some of the hurdles towards a scalable implementation, and how to overcome some of them. There are, however, several others we have not even begun to contemplate.

Although Divincenzo's criteria is a good starting point for addressing an implementation of a quantum computer, there exist several more questions that must be addressed.

Daniel Gottesman looks further into the necessary requirements for scalable, fault-tolerant QIP in [Got02].

Gottesman's extended criteria, along with DiVincenzo's original criteria is given in inset 5.0.1.

Does a quantum computer based on NMR spectroscopy fullfill the extended

1. A scalable physical system with a mapping of qubits onto this system
2. A method for initializing the state of the system to an a priori known state, usually corresponding to $|0\rangle^{\otimes n}$ under the above mapping.
3. A big decoherence time to gate time ratio,
4. Sufficient control of the system via time-dependent Hamiltonians in order to effectively implement a universal set of gates on the system's qubits,
5. A measurement operation on the system's qubits,
6. The ability to interconvert stationary and flying qubits[1],
7. The ability to faithfully transmit flying qubits between specified locations.
8. Low gate error rates,
9. Ability to perform operations in parallel,
10. A way of remaining in, or returning to, the computational Hilbert space,
11. A source of fresh initialized qubits during the computation,
12. Benign error scaling: error rates that do not increase as the computer gets larger, and no large-scale correlated errors.

Inset 5.0.1: DiVincenzo-Gottesman Criteria for Fault Tolerant QIP

DiVincenzo-Gottesman criteria?

So far, we know how to implement qubits, achieve a universal set of gates, and measure in the computational basis.

In NMR there are simply no obvious 'flying qubits'. All interactions are local, based on the —very limited— j-coupling. Previously we had seen this weak interaction as a plus, since it meant that we only had to deal with a few coupled spins when refocusing. Now, however, we can see it is a disadvantage, since it does not allow for far away qubits to interact efficiently. In trans-Crotonic Acid (recall figure 3.1 on page 43) for example, in order to implement a gate between Hydrogen 1 and Carbon 4, the Hydrogen qubit must first be swapped with Carbons 2 and 3, and must be swapped back afterwards. This is not, an efficient procedure: elementary operations take $O(n)$ time where n is the number of qubits in the quantum computer, instead of the $O(1)$ time normally associated with elementary operations.

The operations that can be done in parallel, seem to be quite limited. For example, it is trivial to perform the same operation on all Hydrogen nuclei (recall that a simple hard pulse would suffice), or perform distinct gates on one Hydrogen and one Carbon (since they are attuned to different coils), but performing distinct operations on two Carbons seems more challenging.

In the same fashion, while there exist several methods of initializing qubits in NMR, a method for doing so *in the middle of a computation*, does not seem so clear.

Fortunately, errors do seem to scale reasonably.

While there do exist methods that address the problems posed above, there does not exist, at the present moment (and as far the author knows) any method that addresses all the problems simultaneously.

In conclusion, the most pressing question to be investigated in the future is: *"Do NMR quantum computers scale?"*

Bibliography

[ADH97] Leonard M. Adleman, Jonathan Demarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM J. Comput.*, 26(5):1524–1540, 1997.

[Adl94] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 11, 1994.

[BB84] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *International Conference on Computers, Systems and Signal Processing*, 1984.

[BB92] André Berthiaume and Gilles Brassard. The quantum challenge to structural complexity theory. In *Proceedings of the Seventh Annual Structure in Complexity Theory Conference (Boston, MA, 1992)*, pages 132–137, Los Alamitos, CA, 1992. IEEE Comput. Soc. Press.

[BBG$^+$90] Charles H. Bennett, François Bessette, Gilles Grassard, Louis Salvail, and John Smolin. Experimental quantum cryptography. In *EUROCRYPT*, 1990.

[Ben80a] P. Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22:563–591, 1980.

[Ben80b] P. Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 1980.

[Ben81] P. Benioff. Quantum mechanical Hamiltonian models of discrete processes. *Journal of Mathematical Physics*, 22:495–507, 1981.

[Ben82] P. Benioff. Quantum mechanical Hamiltonian models of Turing machines that dissipate no energy. *Physical Review Letters*, 48:1581–1585, 1982.

[BEZ00] Dirk Bouwmeester, Artur Ekert, and Anton Zeilinger. *The Physics of Quantum Information*. Springer-Verlag, 2000.

[Bra93] Gilles Brassard. A bibliography of quantum cryptography. *Journal of Modern Optics*, 1993.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.

[CEH$^+$98] Richard Cleve, Artur Ekert, Leah Henderson, Chiara Macchiavello, and Michele Mosca. On quantum algorithms. *Complexity 4*, 33, 1998.

[CEMM97] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. 1997.

[Cle99] Richard Cleve. An introduction to quantum complexity theory. 1999.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Interscience, 1991.

[Dav58] Martin Davis. *Computability and Unsolvability*. McGraw-Hill, 1958.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Roy. Soc. London Ser. A*, 400(1818):97–117, 1985.

[Deu89] David Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A*, 1989.

[Fey82] Richard Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[Fre98] Ray Freeman. *Spin Choreography*. Oxford University Press, 1998.

[G3̈1] Kurt Gödel. Über formal unentscheidbare sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

[Got02] Daniel Gottesman. Beyond the DiVincenzo criteria: Requirements and desiderata for fault-tolerance. In *Joint IPAM/MSRI Workshop on Quantum Computing*, 2002.

[Hir01] Mika Hirvensalo. *Quantum Computing*. Springer-Verlag, 2001.

[JK99] J. A. Jones and E. Knill. Efficient refocussing of one spin and two spin interactions for nmr quantum computation. *Journal of Magnetic Resonance*, 141:322–325, 1999.

[KL98] Emanuel Knill and Raymond Laflamme. On the power of one bit of quantum information. *Physical Review Letters*, 81(25), 1998.

[LKC$^+$02] Raymond Laflamme, Emanuel Knill, David Cory, E. Fortunato, T. Havel, C. Miquel, R. Martinez, C. Negreverne, G. Ortiz, M. Pravia, Y. Sharf, S. Sinha, R. Somma, and L. Viola. Introduction to NMR quantum information processing. *xxx.lanl.gov*, 2002.

[LV97] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, 1997.

[MBG93] A. Muller, J. Breguet, and N. Gisin. Experimental demonstration of quantum cryptography using polarized photons in optical fibre over more than one km. *Europhysics Letters*, 23(6), August 1993.

[Mes99] Albert Messiah. *Quantum Mechanics.* Dover, 1999.

[Mol01] Richard A. Mollin. *An Introduction to Cryptography.* Chapman and Hall, 2001.

[Mos99] Michele Mosca. *Quantum Computer Algorithms.* PhD thesis, University of Oxford, 1999.

[MOV96] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1994.

[Pos44] Emil L. Post. Recursively enumerable sets of positive integers and their decision problems. *Bulletin Of The American Mathematical Society*, 50(5):284–316, 1944.

[rEL$^+$83] O. W. Sørensen, G. W. Eich, M. H. Lefitt, G. Bodenhausen, and R. R. Ernst. Product operator formalism for the description of NMR pulse experiments. *Progress in ekert Spectroscopy*, 16(2):163–192, 1983.

[Sak94] J. J. Sakurai. *Modern Quantum Mecanics*. Addison-Wesley, 1994.

[Sho97] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5), 1997.

[Sim97] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5), 1997.

[Sti95] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.

[SV99] Leonard J. Schulman and Umesh V. Vazirani. Molecular scale heat engines and scalable quantum computation. In *Proceedings of the 31'st Annual ACM Symposium on Theory of Computation*, 1999.

[Tur37] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1937.

[Yao93] A. C. Yao. Quantum circuit complexity. In *Prodeedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, 1993.

[Zal98] Christof Zalka. Simulating quantum systems on a quantum computer. *Proceedings of the Royal Society of London A*, 454(1969), 1998.