ABSTRACT

A NEW ALGORITHM FOR THE TIME EVOLUTION OF QUANTUM TRAJECTORY SIMULATIONS

AND

PHYSICALLY MOTIVATED ERROR MODELS IN 1D QUANTUM CELLULAR AUTOMATA

by Douglas M. McNally

Part I of this thesis discusses a modification of a Monte Carlo algorithm used for calculating the dynamics of open quantum systems. The essential change is using the direct solution to the Schrödinger equation (via matrix diagonalization) in lieu of numerical integration routines in order to simplify aspects of the algorithm. Part II is on the topic of 1-dimensional quantum cellular automata (QCA), which is a generalization of classical cellular automata (CA) and a possible platform for quantum computation and the simulation of various physical systems. Of particular interest to the work described herein is a paper by Brennen and Williams [Phys. Rev. A. 68, 042311 (2003)]. This is used as a starting point, and the mathematical framework they describe is convenient for the goal of this project, which is to add some physically realistic non-idealities including a simple decoherence model, timing errors, and pairwise interactions occurring during single-bit operations.
A NEW ALGORITHM FOR THE TIME EVOLUTION OF QUANTUM TRAJECTORY SIMULATIONS

AND

PHYSICALLY MOTIVATED ERROR MODELS IN 1D QUANTUM CELLULAR AUTOMATA

A Thesis

Submitted to the
Faculty of Miami University
in partial fulfillment of
the requirements for the degree of
Master of Science
Department of Physics
by
Douglas M. McNally
Miami University
Oxford, Ohio
2014

Advisor: ____________________________
(James Clemens)

Reader: ____________________________
(Perry Rice)

Reader: ____________________________
(Samir Bali)
Contents

1 Introduction .................................................. 1
  1.1 Quantum Mechanics ..................................... 1
  1.2 Thesis Outline .......................................... 3

2 Quantum Trajectories Formalism ......................... 4
  2.1 Motivation ............................................... 4
  2.2 Quantum Trajectories and the Master Equation ......... 5

3 Modifying the Quantum Trajectories Algorithm .......... 10
  3.1 Mathematical Background ............................... 10
    3.1.1 Equally Spaced Time Steps ....................... 11
  3.2 Monte-Carlo Pseudocode ................................ 11
  3.3 Comparisons with Master Equation Solutions .......... 12

4 Algorithmic Performance .................................. 15
  4.1 Numerical Integration Comparison ..................... 15
  4.2 Conclusions and Future Work .......................... 17

5 Quantum Cellular Automata Background ................... 19
  5.1 Introduction ............................................ 19
  5.2 BQCA Structure ........................................ 20
    5.2.1 Conditional Update Rules ...................... 22

6 Quantum Information Processing Protocols ............... 23
  6.1 State Transfer ......................................... 23
  6.2 State Swap ............................................. 24
  6.3 GHZ State Generation .................................. 25
  6.4 Entangled Pair ........................................ 26
7 Error Models

7.1 Bit and Phase Flip Errors .............................................. 28
7.2 Timing Errors ............................................................ 29
7.3 Continuous Interactions ............................................... 30
    7.3.1 Compensation .................................................... 32

8 QCA Conclusions ....................................................... 36

8.1 Discussion .............................................................. 36
8.2 Future Work ............................................................. 36

A Trajectories Code ...................................................... 38

B QCA Code Snippets ..................................................... 40

C Why Simulations Used 10 Qubits ...................................... 43

References ................................................................. 44
List of Tables

7.1  Slopes for the linear approximations of $F^2$ vs. $\delta$ . . . . . . . . . . . . . . . . . . . . . . . 28
7.2  Slopes for the linear approximations of $F^2$ vs. $\sigma$ . . . . . . . . . . . . . . . . . . . . . . . 30
List of Figures

3.1 \( \langle aa^\dagger \rangle \) and \( \langle \sigma_+ \sigma_- \rangle \) for a two-level atom coupled to a leaky cavity initially containing 5 photons ........................................... 13
3.2 \( \langle \sigma_+ \sigma_- \rangle \) for a two-level atom being driven on resonance by an external field and allowed to spontaneously emit ........................................... 14
4.1 Linear performance scaling with an increasing number of trajectories. ............. 16
4.2 Performance as a function of Hilbert space dimensionality. ............................. 17
6.1 Space-time diagram for the state transfer protocol ........................................... 24
6.2 Space-time diagram for the state swap protocol .............................................. 25
6.3 Space-time diagram for the GHZ state generation protocol ............................. 26
6.4 Space-time diagram for the entangled pair protocol ........................................... 27
7.1 \( F^2 \) vs. \( \delta \) for the transfer protocol ................................................................. 29
7.2 \( F^2 \) vs. \( \gamma \) for each of the sample QIP protocols ......................................... 31
7.3 \( F^2 \) vs. \( \gamma \) for the swap and transfer protocols averaged over random initial states 32
7.4 \( F^2 \) vs. \( C \) for each of the sample QIP protocols ......................................... 33
7.5 \( F^2 \) vs. \( C \) for the swap and transfer protocols averaged over random initial states 34
Glossary of Terms

**Density operator** An operator which represents the state of a quantum system. For a statistical mixture of states, meaning the system is in one of several states $|\psi_i\rangle$ each with a probability $p_i$, the density operator is defined as $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$. The density operator is guaranteed to have $\text{Tr}(\rho) = 1$ and to be Hermitian.

**Heisenberg Picture** A mathematical view of quantum mechanics where state vectors are constant in time (and therefore stationary in Hilbert space) and operators carry the time evolution.

**Hermitian operator** An operator that is self-adjoint, $H = H^\dagger$. Hermitian operators are guaranteed to have real eigenvalues and a set of orthonormal eigenvectors which span the vector space, and they therefore can be used in quantum mechanics to represent observable quantities, with their eigenvalues corresponding to measurement outcomes, and eigenvectors corresponding to eigenstates in the basis of the Hermitian operator.

**Hilbert Space** A complete vector space with an inner product which can be finite or infinite dimensional.

**Schrödinger Picture** A mathematical view of quantum mechanics where state vectors carry time evolution and operators are constant in time.

**State vector** Represents an abstract quantum state and encodes all known information about a quantum system in a pure state. Denoted in Dirac notation as $|\psi\rangle$.

**Unitary operator** An operator whose Hermitian conjugate is its inverse, i.e. $U^\dagger U = UU^\dagger = \mathbb{I}$. Unitary operators have the useful property of being norm-preserving meaning the norm of $U |\psi\rangle$ is $\langle \psi | U^\dagger U |\psi\rangle = \langle \psi | \psi \rangle$ and is therefore unchanged by the action of $U$, and furthermore this implies that unitary operations are always reversible. Another useful fact, $e^{iH}$ is guaranteed to be unitary if $H$ is Hermitian.
I would like to first acknowledge my advisor, James Clemens, who has been an excellent academic mentor. He has also been someone with whom I could frankly discuss my future and have friendly conversations which is something he had no obligation to do, and I am extremely appreciative of all his help and advice. I also want to acknowledge all the Miami physics faculty (and of course especially my thesis committee), I have had only positive experiences with all of the professors here. I am not someone who endeavors to have an abundance of social relationships, but I am thankful for the small group of friends I have made over the last few years, all of whom know who they are.

There are many famous scientists I admire that have impacted me, be it through recorded lectures or their writing. While I won’t make a list of such individuals because I don’t think that’s very useful, it is important to me to remember the general sentiment that science is a collaborative effort and we (as in the collective we) stand on the shoulders of giants and benefit from the tremendous body of knowledge built up before us. One remark in particular that has stuck with me since I first heard it comes from a video interview with the late Richard Feynman, and to conclude my acknowledgments I want to include it in this space. If I had to single out one thing\(^1\) that inspired me to chose becoming a physicist as my career path, it would be this:

“It’s all really there. That’s what really gets you. But you gotta stop and think about it to really get the pleasure about the complexity, the inconceivable nature of nature.”

– Richard Feynman

\(^1\)Of course it’s not really as simple as a single statement but physics has a celebrated tradition of approximations, and it’s a little bit more romantic that way.
Chapter 1

Introduction

1.1 Quantum Mechanics

The basic foundations of quantum theory have not changed substantially since their development in the first half of the 20th century. As is the case with most rigorous scientific disciplines, progress comes as gradual, incremental steps in the development of the collective body of knowledge represented by the theory at large, with the occasional punctuated breakthrough. This thesis begins now, as it should, with a brief prelude on the foundations and important developments in quantum mechanics. The central frameworks of modern quantum theory include quantum electrodynamics (or QED—the area with which quantum optics is primarily concerned), quantum chromodynamics (QCD), and more generally the Standard Model, all of which are empirically based and are formulated as relativistic quantum field theories (QFT). The details of QFT are well beyond the scope of this thesis which will proceed with a more basic discussion of characteristics common to any quantum system.

By far the most mainstream view of quantum mechanics holds that observable quantities are defined only with reference to their measurement. That is, a given system does not exist in a well-defined state before a measurement of it is made and seemingly random outcomes do not result from ignorance of some hitherto unknown hidden variables but rather from an inherent lack of concrete physical determination of quantum parameters. It is for this reason that controversy over particular philosophical interpretations of quantum mechanics exists and much of that hinges on the precise character of measurement [1]. It should be briefly mentioned that an alternate formulation of quantum theory known as de Broglie-Bohm theory (or pilot-wave theory) does exist and is actively developed by a small subset of quantum physicists [2]. However it is explicitly nonlocal (requiring so-called global hidden variables) which is distasteful since all known physical laws are local and furthermore it does not provide any meaningful gain over the mainstream view in terms of predictive capacity or agreement with empirical evidence.

The motivation behind studying quantum physics stems from both a desire to under-
stand fundamental physics and thereby elucidate the nature of reality as well as some relatively recently-conceived practical applications. Probing fundamental physics is an extremely worthwhile endeavor in its own right, but sometimes presenting science in terms of an application has broader appeal. At the forefront of applications of quantum physics is the field of quantum computing which has grown to be a substantial portion of contemporary quantum physics research since its inception in the early 1980s [3, 4]. The applications of quantum computing range from cryptography algorithms to simulating physical systems which would require an inordinate amount of computational effort to simulate on classical computers. The basic idea of a quantum computer is to exploit the properties of quantum superposition whereby a system can simultaneously be in (typically) two states—such a system is referred to as a quantum bit or qubit. Then many such systems which mutually interact are cleverly manipulated to have the net effect of preparing a set of qubits such that their measurement gives the answer to some problem which was encoded in the way the system was manipulated. The mathematics of quantum mechanics becomes exponentially harder to solve with classical algorithms as more subsystems are included, but of course the physics proceeds with no difficulty. It is in this sense that at some level the “calculations” of the equations of quantum mechanics are being performed by the physical system, and so if there is a way to formulate a problem in terms of these equations then it can be solved on a quantum computer.

It has proved extremely difficult to experimentally isolate quantum systems in order to preserve whatever delicate properties are being investigated. The primary culprit is so-called quantum dechoerence—the process by which quantum effects are damped out by interacting with large systems which possess many degrees of freedom. Many interesting quantum phenomena rely on maintaining relative phases internal to the system or in other words different elements of a wavefunction interfering with each other. Interactions with the environment in which a system resides generate decoherence through correlations between the system and environment which have the net effect of spreading out the total quantum information. Generally speaking, decoherence is the reason that quantum effects are not observed on macroscopic scales. As a system grows in size, more and more avenues for environmental decoherence are introduced, and it is for this reason that some of the aforementioned applications of quantum mechanics such as quantum computing face a major challenge.
1.2 Thesis Outline

This thesis contains two somewhat disparate bodies of work which comprise an overall research effort aimed at completing the requirements for the degree of Master of Science. The two topics will be addressed individually in the pages that follow. Both topics share the superficial similarities of being related to quantum mechanical dynamics of open systems and being primarily computational in nature. It may be possible in future work to actually use the techniques of quantum trajectory simulations to model decoherence in quantum cellular automata in a continuous-time fashion, but at present no such effort has been made.

Part I is on the topic of quantum trajectory theory. It begins with a discussion of the background and motivation for quantum trajectories and describes the formalism demonstrating that the Monte Carlo approach is indeed equivalent to the analytically obtained master equation. A modification to the typical implementation of the algorithm is then proposed and shown to be equivalent, and finally the utility of the alternate method is investigated by comparing computational performance between it and existing implementations.

Part II, which begins in Chapter 5, is on the topic of error models in quantum cellular automata (QCA). As mentioned above, quantum systems are extremely delicate and so it can be useful to have an idea of how much error in an experiment is acceptable and to characterize the expected effects of various errors. A discussion of a mathematical framework for modeling QCA is presented and four model quantum information protocols which serve as a platform for the various error models are described. Finally the results of error simulations are presented and their implications are discussed.
Chapter 2

Quantum Trajectories Formalism

2.1 Motivation

Real quantum systems are so-called open quantum systems which interact with their environment. It is often difficult or impossible to analytically solve the equations governing these systems and thus such calculations are amenable to computational methods. One formulation of the time evolution of open quantum systems is the Lindblad Master equation [5, 6]:

\[ \dot{\rho} = -\frac{i}{\hbar}[H_{\text{sys}}, \rho] + \frac{1}{2} \sum_i (2\hat{C}_i \rho \hat{C}_i^\dagger - \hat{C}_i^\dagger \hat{C}_i \rho - \rho \hat{C}_i^\dagger \hat{C}_i) \] (2.1)

where \( H_{\text{sys}} \) is the system Hamiltonian and \( \hat{C}_i \) are “collapse” operators that account for the coupling between the environment and the system. This equation is derived under the following assumptions:

1. The system and entire environment can both be represented by density operators
2. The interaction between the system and the environment is weak such that perturbation theory can be used (Born approximation)
3. Initially the system and its environment are uncorrelated
4. The correlation time between the system and environment is very short compared to the timescales of the interactions within the system (Markov approximation)

It is possible to solve this equation directly numerically, but it requires using the density operator \( \rho \) which has \( D^2 \) entries in its matrix representation where \( D \) is the Hilbert space dimensionality. This is not ideal since it means the amount of data one has to keep track of scales as the square of the dimensionality of the Hilbert space. Quantum trajectory simulations are an alternate, but equivalent method of obtaining the time evolution of the density operator. One treats the problem with a Monte-Carlo approach, utilizing ensemble
averaging over many independent realizations of the system in order to arrive at a result that is on average equivalent to the master equation [7, 8].

Interestingly, quantum trajectories can actually reveal information about a system that is hidden to the master equation solution. Since the master equation describes the average behavior of an ensemble of systems, the dynamics of the constituents of the ensemble are not reported. The quantum trajectories approach can therefore describe a possible set of measurement records (of which there are many) corresponding to the emergent behavior of the ensemble.

2.2 Quantum Trajectories and the Master Equation

Quantum trajectory theory provides a stochastic method which generates visualizable realizations of quantum fluctuations [6]. Given any particular quantum optics experiment, the experimenter does not observe expectation values or probability amplitudes, but rather quantized occurrences such as photon emission or absorption events. In this sense, each trajectory can be thought of as a path that an individual system could have taken. Conversely, expectation values and statistical mixtures of quantum states are things that can only be measured on average (and recall that the mathematical apparatus of the master equation can be used to explicitly calculate these ensemble-averaged quantities) and they provide information about the relative likelihood of different trajectories. In an experiment aiming to measure an ensemble behavior, the experimenter obtains many sets of measurement records which correspond to different trajectories of the density operator through Hilbert space. It is in this spirit that the quantum trajectories formalism can be verified, by showing that it gives, on average, the same evolution of the density operator as the master equation (Eq. 2.1).

Quantum trajectories will be loosely referred to as realizations of individual system evolutions, but, to be clear, an affirmative claim about the actual physical dynamics of the system is not being made. Common visualizations of the dynamics include discrete jumps between stationary states or continuous time evolution, however these are just that—visualizations. Of course there is nothing wrong with visualizations and indeed the primary utility of quantum trajectory theory is to provide just such visualizations, however it is important to bear in mind that all that is happening unconditionally is what is measured, not what is statistically inferred. There have been some experimental and theoretical attempts [9, 10, 11] to connect quantum trajectories to a more concrete physical interpretation, but this thesis will remain agnostic on this issue going forward.
The procedure outlined in [7] will be followed in order to demonstrate the equivalency of the master equation and quantum trajectory theory. Each individual trajectory involves starting with the same initial state \( |\psi\rangle \), and allowing it to evolve in time according to the Schrödinger equation:

\[
i\hbar \dot{|\psi}\rangle = H_{\text{eff}} |\psi\rangle \tag{2.2}
\]

with a non-Hermitian effective Hamiltonian

\[
H_{\text{eff}} = H_{\text{sys}} - \frac{i\hbar}{2} \sum_i \hat{C}_i \hat{C}_i^\dagger, \quad H_{\text{eff}}^\dagger = H_{\text{sys}} + \frac{i\hbar}{2} \sum_i \hat{C}_i^\dagger \hat{C}_i \tag{2.3}
\]

where again, \( \hat{C}_i \) are so-called “collapse operators” which represent the interaction between the system and its environment. This will lead to non-unitarity in the time evolution from the Schrödinger equation. Traditional implementations of this algorithm use numerical integration of the Schrödinger equation in order to obtain the time evolution of the state vector. This could be realized by Euler integration as

\[
|\psi(t + \Delta t)\rangle = (1 - \frac{i}{\hbar} H_{\text{eff}} \Delta t) |\psi(t)\rangle \tag{2.4}
\]

The theoretical derivation here assumes that the time step \( \Delta t \) approaches zero such that the error in Euler’s method is vanishingly small, and higher orders of \( \Delta t \) can be neglected (meaning that this introduces error whose magnitude is no greater than that introduced by the approximations used to derive the master equation to begin with). Note that \( |\overline{\psi}\rangle \) denotes an unnormalized state with \( \langle \overline{\psi}|\overline{\psi}\rangle \leq 1 \).

Now consider the norm of the state \( |\overline{\psi}(t + \Delta t)\rangle \):

\[
\langle \overline{\psi}(t + \Delta t)|\overline{\psi}(t + \Delta t)\rangle = \langle \psi(t)| (1 + \frac{i}{\hbar} H_{\text{eff}} \Delta t)(1 - \frac{i}{\hbar} H_{\text{eff}} \Delta t) |\psi(t)\rangle
\]

\[
= \langle \psi(t)|\psi(t)\rangle + (\Delta t)\frac{i\hbar}{2} \langle \psi(t)| H_{\text{eff}} \Delta t |\psi(t)\rangle + \frac{i}{\hbar} \Delta t \langle \psi(t)| H_{\text{eff}}^\dagger |\psi(t)\rangle - \frac{i}{\hbar} \Delta t \langle \psi(t)| H_{\text{eff}} |\psi(t)\rangle
\]

\[
= 1 + \frac{i}{\hbar} \Delta t \left( \langle \psi(t)| H_{\text{sys}} + \frac{i\hbar}{2} \sum_i \hat{C}_i \hat{C}_i^\dagger |\psi(t)\rangle - \langle \psi(t)| H_{\text{sys}} - \frac{i\hbar}{2} \sum_i \hat{C}_i^\dagger \hat{C}_i |\psi(t)\rangle \right)
\]

\[
= 1 - \Delta t \langle \psi(t)| \sum_i \hat{C}_i^\dagger \hat{C}_i |\psi(t)\rangle = 1 - \sum_i \Delta p_i \tag{2.5}
\]

\(^1\)In the case of a density matrix that does not have a corresponding pure state, each of the states in the statistical mixture \( \rho(0) = \sum_i p_i |\psi_i\rangle \langle \psi_i| \) are used with probability \( p_i \)
which defines
\[ \Delta p_i = \Delta t \langle \psi(t) | \hat{C}_i^\dagger \hat{C}_i | \psi(t) \rangle, \quad \Delta p = \sum_i \Delta p_i. \] (2.6)

Here is where the stochastic element of the Monte-Carlo simulation appears. A random number \( r \) which is uniformly distributed in the range \([0, 1]\) is chosen and at each time step, a decision is made to either

- Evolve the current state up to the next time step \( (\Delta p < r) \)
- Apply a collapse operator and re-normalize the state \( (\Delta p > r) \).

In the first case, the state is normalized and the process is repeated with a new random number at the next increment of \( \Delta t \). As an aside, this procedure is equivalent to leaving the state vector unnormalized keeping the same random number \( r \) until \( \langle \bar{\psi}(t) | \bar{\psi}(t) \rangle < r \); in effect with an unnormalized state, \( \Delta p \) accumulates with each subsequent time step by referencing the norm at the previous time step (i.e. the first term in Eq. 2.5 would not be unity). Therefore the unconditional probability of the next collapse occurring monotonically decreases. This means that there is information to be gleaned from the knowledge of a null result up to the current time, and thus the outcome at subsequent time steps is conditioned on this knowledge. This fact will be useful when considering a modification to the procedure, but it is easiest to proceed by re-normalizing the evolved state for illustrative purposes, and thus

\[ |\psi(t + \Delta t)\rangle = \frac{1}{\sqrt{1 - \Delta p}} |\bar{\psi}(t + \Delta t)\rangle. \] (2.7)

In the latter case a collapse occurs. Which \( \hat{C}_i \) occurs depends on its relative probability given by

\[ \delta_i = \frac{\Delta p_i}{\Delta p} = \frac{\langle \psi(t) | \hat{C}_i^\dagger \hat{C}_i | \psi(t) \rangle}{\sum_j \langle \psi(t) | \hat{C}_j^\dagger \hat{C}_j | \psi(t) \rangle} \] (2.8)

and notice that in the last expression the reference to \( \Delta t \) cancels, this will again be important when discussing the proposed modification in the next chapter. And so, in the case of a collapse the evolved state vector is

\[ |\psi(t + \Delta t)\rangle = \sqrt{\frac{\Delta t}{\Delta p_i}} \hat{C}_i |\psi(t)\rangle. \] (2.9)

The present goal is to demonstrate that the master equation and quantum trajectories give the same time-evolved density operator in the limit of averaging many trajectories. The
quantum trajectories procedure for evolving a state vector in time by an infinitesimal $\Delta t$ has just been outlined, and at each $\Delta t$ the state will just evolve in time with probability $1 - \Delta p$, and a collapse will occur with probability $\Delta p$. Therefore in the average case (weighting each possible outcome by its corresponding probability), the density operator is

$$
\rho_{\text{ave}}(t + \Delta t) = (1 - \Delta p) \frac{\langle \bar{\psi}(t + \Delta t) \rangle}{\sqrt{1 - \Delta p}} \frac{\langle \bar{\psi}(t + \Delta t) \rangle}{\sqrt{1 - \Delta p}} + \Delta p \sum_i \delta_i \frac{\Delta t}{\Delta p_i} \hat{C}_i |\psi(t)\rangle \langle \psi(t)| \hat{C}_i^\dagger
$$

(2.10)

and writing the time-evolved state vector in terms of Eq. 2.4 gives

$$
\rho_{\text{ave}}(t + \Delta t) = (1 - \frac{i}{\hbar}H_{\text{eff}}\Delta t) |\psi(t)\rangle \langle \psi(t)| (1 + \frac{i}{\hbar}H_{\text{eff}}^\dagger) + \Delta t \sum_i \hat{C}_i |\psi(t)\rangle \langle \psi(t)| \hat{C}_i^\dagger
$$

(2.11)

where the substitution $|\psi(t)\rangle \langle \psi(t)| = \rho(t)$ has been made. Recall that the case of averaging over many trajectories (and therefore their corresponding state vectors) is of interest here. This can be accomplished by making the substitution of $\rho(t) \mapsto \rho_{\text{ave}}(t)$ on the RHS (meaning that at time $t$ all possible trajectories were appropriately represented)

$$
\rho_{\text{ave}}(t + \Delta t) = \rho_{\text{ave}}(t) - \frac{i}{\hbar} [H_{\text{sys}}, \rho_{\text{ave}}(t)] \Delta t + \frac{\Delta t}{2} \sum_i \rho_{\text{ave}}(t) \hat{C}_i \hat{C}_i^\dagger + \hat{C}_i \hat{C}_i^\dagger \rho_{\text{ave}}(t) + 2 \hat{C}_i \rho_{\text{ave}}(t) \hat{C}_i^\dagger
$$

(2.12)

where the sums sharing a common index have also been combined. Re-arranging terms and recalling that the limit of small $\Delta t$ was taken gives

$$
\lim_{\Delta t \to 0} \frac{\rho_{\text{ave}}(t + \Delta t) - \rho_{\text{ave}}(t)}{\Delta t} = -\frac{i}{\hbar} [H_{\text{sys}}, \rho_{\text{ave}}(t)] + \frac{1}{2} \sum_i \rho_{\text{ave}}(t) \hat{C}_i \hat{C}_i^\dagger + \hat{C}_i \hat{C}_i^\dagger \rho_{\text{ave}}(t) + 2 \hat{C}_i \rho_{\text{ave}}(t) \hat{C}_i^\dagger
$$

(2.13)
which (by the definition of the derivative\(^2\)) is exactly Eq. 2.1 given that \(\rho_{\text{ave}}(t) = \rho(t)\) initially. Therefore it has been shown that in the limit of infinite trajectories (meaning that the probability of each outcome is represented with unit fidelity) and sufficiently small time steps the equation of motion for the density operator is identical from the perspectives of quantum trajectory theory and the master equation.

Notice that the solution to the Schrödinger equation (Eq. 2.2) in closed form could have been found as

\[
|\psi(t)\rangle = e^{-\frac{i}{\hbar}H_{\text{eff}}t} |\psi(0)\rangle,
\]

this solution is what is used in the alternate approach discussed here. The arguments above also apply here when considering a small time step

\[
|\psi(t + \Delta t)\rangle = e^{-\frac{i}{\hbar}H_{\text{eff}}\Delta t} |\psi(t)\rangle = (1 - \frac{i}{\hbar}H_{\text{eff}}\Delta t + \ldots) |\psi(t)\rangle
\]

which to the first order in \(\Delta t\) recovers Eq. 2.4.

\(^2\)The limit is written this way for clarity, but it should be noted that physically this limit cannot be taken and is approximate [8]. Recall that the Markov approximation was invoked to obtain the master equation in the first place meaning that only dynamics which occur on timescales long compared to the timescales of the environment correlations can be monitored.
Chapter 3
Modifying the Quantum Trajectories Algorithm

3.1 Mathematical Background

This alternate technique hinges on the notion of matrix diagonalization. Namely that for a diagonalizable matrix \( A \), there exists an invertible similarity transformation matrix \( S \) whose columns are the right eigenvectors of \( A \), such that \( S^{-1}AS = \Lambda \). Where \( \Lambda \) is a diagonal matrix with the eigenvalues of \( A \) on its diagonal. Extending this to the notion of matrix exponentiation which has meaning through its Taylor series representation, it can be shown [12] that

\[
e^{At} = S e^{\Lambda t} S^{-1}
\]  

(3.1)

this of course has a consequence for the solution to the Schrödinger equation,

\[
|\psi(t)\rangle = S e^{\Lambda t} S^{-1} |\psi(0)\rangle
\]  

(3.2)

where here \( A = -\frac{i}{\hbar} H_{\text{eff}} \), and since the exponential of a diagonal matrix is just the exponential of each of the diagonal elements, this greatly simplifies the time evolution. Not only that, but it also means that it is possible to know the state at a later time with unlimited accuracy and thus find the times at which “jumps” (applications of collapse operators) will occur in advance using a zero-finding algorithm

\[
\langle \bar{\psi}(t) | \bar{\psi}(t) \rangle - r = 0^1
\]  

(3.3)

where \( r \) is a random number in the range \([0,1]\), and a collapse occurs when the norm of the state vector falls below \( r \). This eliminates the potential for any backtracking that might be necessary in a numerical integration approach in the case that the timestep being

---

\(^1\)Brent’s method was used in the implementation discussed here.
used overshoots the time of a jump. It should be noted that it is unclear if the effective Hamiltonian will always be diagonalizable; however this approach has utility even if it is not, because it is known to be for many problems of interest. Another caveat is this method will only work for problems that have no explicit time dependence in the system Hamiltonian or the collapse operators, as this would cause the similarity transformation matrices to not be constant in time.

3.1.1 Equally Spaced Time Steps

When using quantum trajectories, the user typically specifies some operators \( \hat{O} \) corresponding to observables for which they wish to know the expectation value at a discrete set of times. These expectation values are calculated as always, \( \langle \hat{O} \rangle (t) = \langle \psi(t) | \hat{O} | \psi(t) \rangle \) at each \( t \) that is requested. If the list of times are equally spaced by an amount \( \Delta t \), i.e. \( t_i = i \Delta t \), then the state vector at a time \( t_i \) can be calculated as

\[
|\psi(t_i)\rangle = [S e^{\Lambda \Delta t} S^{-1}]^i |\psi(0)\rangle
\]  

(3.4)

and note that the quantity \( S e^{\Lambda \Delta t} S^{-1} \) is constant and therefore only needs to be calculated once. When jumps occur the time evolution goes up to \( t_{\text{jump}} \) which is generally not a multiple of \( \Delta t \) and so there will be an occasional need to compute the quantity \( S e^{\Lambda t} S^{-1} \) but often the vast majority of time displacements are by \( \Delta t \). Therefore this can provide a substantial improvement in the total amount of matrix multiplications that need to be performed.

3.2 Monte-Carlo Pseudocode

To illustrate the differences that the matrix diagonalization technique introduces to the algorithm, consider a pseudocode outline of the procedure in the two cases. These algorithms assume a discrete set of increasing times at which to compute the state is provided, i.e. \( t_{\text{list}} = [0, t_1, t_2, \cdots, t_n] \).

**Quantum trajectories using numerical integration**

\[
t \leftarrow t_0
\]

\[
\text{while } t < t_n \text{ do}
\]

\[
\quad \text{evolve } |\overline{\psi}(t)\rangle \rightarrow |\overline{\psi}(t_{i+1})\rangle \text{ (using Eq. 2.4 or another ODE algorithm)}
\]

\[
\quad \text{compute } p(t_{i+1}) = \langle \overline{\psi}(t_{i+1}) | \overline{\psi}(t_{i+1}) \rangle
\]

11
if \( p(t_i) > r \) then
  calculate expectation values
  \( t \leftarrow t_{i+1} \)
  increment \( i \)
else
  backtrack\(^2\) to \( t \) where \( p(t) = r \)
  apply collapse \( |\psi(t)\rangle = \hat{C} |\psi(t)\rangle \)
  normalize \( |\psi(t)\rangle \)
end if
end while

Note that additional calculations may be needed in the case that the time of a jump lies somewhere between \( t_i, t_{i+1} \).

Quantum trajectories using matrix diagonalization

\( t \leftarrow t_0 \)
while \( t_{i+1} \leq t_n \) do
  calculate time of next jump = \( t_{\text{jump}} \) with \( \langle \overline{\psi}(t_{\text{jump}}) | \overline{\psi}(t_{\text{jump}}) \rangle - r = 0 \)
  while \( t_{i+1} \leq t_{\text{jump}} \) do
    evolve to \( t_{i+1} \) (using Eq. 3.2)
    calculate expectation values
    increment \( i \)
  end while
  evolve to \( t_{\text{jump}} \)
  apply collapse \( |\psi(t_{\text{jump}})\rangle = \hat{C} |\psi(t_{\text{jump}})\rangle \)
  normalize \( |\psi(t_{\text{jump}})\rangle \)
end while

Both procedures run until a maximum time \( t_n \), and at each time in \texttt{tlist} various expectation values would be calculated.

3.3 Comparisons with Master Equation Solutions

Figures 3.1a & 3.1b help to illustrate how the quantum trajectories algorithm works in practice. A simulation was run for a two-level atom coupled to an optical cavity that initially

\(^2\)This could be done using Eq. 2.4 with increasingly smaller \( \Delta t \).
Figure 3.1: (a) A single trajectory for a two-level atom coupled to a leaky cavity initially containing 5 photons. The discrete jumps in the cavity photon number correspond to a collapse operator being applied to the state—specifically a photon leaving the cavity. (b) 5000 trajectories like those in (a) averaged together and plotted along with the master equation solutions (red and light green) for comparison. Here $\hbar \omega = 2\pi$, $\hbar g = \frac{\pi}{2}$, and $\hbar \kappa = \frac{1}{20}$.

contained 5 photons. The effective quantum trajectory Hamiltonian for this is

$$H_{\text{eff}} = \hbar \omega (a^\dagger a + \sigma_+ \sigma_-) + \hbar g (\sigma_- a^\dagger + \sigma_+ a) - i \hbar \kappa a^\dagger a$$  \hspace{1cm} (3.5)$$

where $\omega$ is the atomic resonance frequency, $g$ is the coupling strength, and $\kappa$ is the photon leakage rate. Expectation values are plotted against time; specifically Fig. 3.1a shows the expectation values for a single trajectory. There are clear discontinuous jumps in the plot of the expectation value of the number of photons in the cavity. These jumps correspond to a photon leaving the cavity, or in terms of the algorithm, they correspond to a collapse operator being applied to the state. Fig. 3.1b shows the result of the ensemble average of many (5000) trajectories like those in Fig. 3.1a. Also plotted is the solution from the master equation, and clearly there is excellent agreement between the two—they are nearly coincident and the agreement will increase with more trajectories. These plots were generated using the matrix diagonalization techniques described above, and therefore provide some evidence that this alternate algorithm does function as claimed.

Fig. 3.2 shows the results for the canonical problem of a two-level system (initially in the ground state) being driven on resonance by an external field. The effective quantum
Figure 3.2: A two-level atom being driven on resonance by an external field and allowed to spontaneously emit (resonance florescence). 5000 trajectories plotted alongside the master equation solution (green) for comparison. Here $\Omega = 2\pi$ and $\hbar \gamma = \frac{1}{2}$.

The trajectories Hamiltonian for this problem is

$$H_{\text{eff}} = \Omega \sigma_x - i\hbar \gamma \sigma_+ \sigma_-$$

where $\Omega$ is the Rabi frequency and $\gamma$ is the spontaneous emission rate. The system is allowed to emit and absorb light quanta. The emission and absorption rates are the same, and the excited state lifetime is assumed to be long compared to the inverse Rabi frequency. Under these conditions, the steady state excitation probability should clearly be $\frac{1}{2}$ (the system is just as likely to be in the excited state or the ground state) which is in agreement with the figure. Again there is clearly excellent agreement with the solution from the master equation, showing again that everything is working as intended.
Chapter 4

Algorithmic Performance

4.1 Numerical Integration Comparison

For the reasons discussed previously, there is hope that this alternate algorithm will be faster for some problems, or perhaps in general. Significant improvement in speed has been shown in the two examples above. This algorithm has been developed in the context of possibly being used in the popular quantum optics package The Quantum Toolbox in Python [13, 14] (QuTiP). To that end, a version was developed purely in Python. QuTiP presently has an optional Monte-Carlo solver with a Fortran 90 backend using F2PY [15]. The algorithm described here was also implemented in Fortran 90 by modifying the existing code for that implementation—the performance will depend strongly on the language used. Specifically Fortran will always be substantially faster than Python. For this reason (and others) it can be difficult to compare performance, but an attempt is made here. Note that further optimizations which improve upon these results may possible. For problems with small Hilbert spaces, this method can be considerably faster. This is demonstrated in Fig. 4.1 which shows the computation time for varying amounts of trajectories for the problem described in Fig. 3.2. Clearly the new method is faster for this problem, however it does represent the smallest possible Hilbert space that could be encountered in a quantum mechanical calculation. It is not clear that this scales to large Hilbert spaces, which presents a substantial limitation. Small problems can already be solved quickly with a master equation approach. One possible use would be for applications where one needs to solve an ensemble of many small problems to generate a 3D plot.

An investigation into the crossover in speed up is presented here in Fig. 4.2. A driven Dicke model [16] was used as the example problem for convenience, since it exhibits linear scaling of the size of the Hilbert space. In Fig. 4.2 it can be seen that the point at which the new method starts to become slower than the existing algorithm is around a Hilbert space dimensionality of $D = 180$ for the Fortran 90 implementation and $D = 85$ for Python. Unfortunately this often is not even large enough to warrant the use of quantum trajectories
Figure 4.1: Computation times for varying amounts of trajectories for the problem described in Figure 2. Both methods show linear scaling as expected, with a smaller slope for the new method. Both are Fortran 90 implementations. The simulation was run on Ubuntu 14.04 LTS with an AMD FX-8350 CPU clocked at 4.0GHz.

in the first place (usually for $D$ on the order of several hundred). There are some exceptions of course for situations where information about the conditional dynamics of a system is desired; this information cannot be obtained from a master equation solution. It is worth noting however that the crossover point does depend heavily on the character of the problem in question, and even for the problem used here it had a non-negligible dependence on the amount of driving. The likely reason for this method being slower for large problems is that the similarity transformation matrices are generally dense. By contrast, the Hamiltonian and collapse operators are generally extremely sparse which can give the numerical integration approach an advantage when it comes to the actual calculations even though it ostensibly has more steps at a high level. This means that the inevitability of using dense matrices in the calculations ultimately seems to be too high of a computational cost and wins out over whatever savings are gained by eliminating the need to compute the norm at each time step and eliminating backtracking. It may be possible to improve on the performance with clever optimizations, however none are apparent at the present time and this is not being pursued.
Figure 4.2: Demonstration of the performance crossover in terms of the size of the Hilbert space between the new (diagonalization) method and the existing (ODE) method. A driven Dicke Model was used. (a) Shows performance for Fortran 90 implementations, (b) is for pure Python implementations. The simulation was run on Ubuntu 14.04 LTS with an AMD FX-8350 CPU clocked at 4.0GHz.

4.2 Conclusions and Future Work

An alternate method for finding the time evolution of a state vector by computing the exponential of the diagonal time-independent effective Hamiltonian in quantum trajectory simulations was presented. Evidence that this method is valid was given by computing solutions for several problems and comparing this with the known results. Computational performance was then compared between Fortran 90 implementations of this new algorithm and the \texttt{mcsolve} method from QuTiP [13]. For problems with small Hilbert space dimensionality, the matrix diagonalization technique is definitively faster when tested on several consumer-grade CPUs. When it comes to problems with larger Hilbert spaces the performance benefit drops off and at some point the numerical integration approach becomes favorable again. In order to quantify this crossover, a problem whose Hilbert space scales linearly was chosen and the results demonstrated that there is a clear range over which this new approach is useful. However since the calculations rely heavily on matrix multiplication it was found that the particular version of the Basic Linear Algebra Subprograms (BLAS) called by Fortran 90 substantially influenced performance, and the version that offered the best performance on the hardware used was OpenBLAS [17]. Which method is preferable will ultimately depend upon the particular problem in question, but nevertheless there is strong
evidence that at least in some circumstances using the technique described is beneficial in terms of computational time.

Early work on this project concerned implementing the quantum trajectories algorithm with massively parallel computing on General-Purpose Computing on Graphics Processing Units (GPGPU) with CUDA and OpenCL [18, 19]. It was found that the extremely conditional time evolution in the quantum trajectories approach was not easily amenable to such an application. Specifically GPGPU performance is very poor with branching, and in the QT approach, decisions about the evolution need to be made at every time step. Further attempts at implementing the algorithm in that way may be worthwhile, as well as optimizing the present implementation. In a related area, it may be much more useful and easy to implement the master equation algorithm with OpenCL or CUDA, since it does not rely on branching but just straight numerical integration which at least superficially should be straightforward to implement in a massively parallel way.
Chapter 5

Quantum Cellular Automata

Background

This work expands a previous block-partitioned quantum cellular automata (BQCA) model proposed by Brennen and Williams [Phys. Rev. A. 68, 042311 (2003)] to incorporate physically realistic error modes. These include timing errors in the form of over- and under-rotations of quantum states during computational gate sequences, stochastic phase and bit flip errors, as well undesired as pairwise interactions occurring during single-bit gate portions of an update sequence. For the last error mode, a compensation method is proposed and investigated as a method to mitigate the impact of this non-ideality. Each of these error regimes is implemented using either Monte Carlo simulations or modifications to the prescribed gate sequences for four quantum information processing protocols of interest in order to simulate their effect. The impact of these various error modes on the function of a QCA gate sequence is investigated in terms of the fidelity of the final state. These results have implications for the feasibility of implementing a QCA in the face of environmental decoherence and timing errors. Broadly, the results show that a larger system corresponds to a lower error tolerance and fidelity falls rapidly with increasing error rates.

5.1 Introduction

Quantum Cellular Automata (QCA) are of interest for a variety of reasons, one of which is that they are able to be used as a platform for quantum computation (QC) [20]. Many implementations of QCA have been proposed [21], but they all have some fundamental similarities. In particular, the essential aspect of QCA is that it largely eliminates the need for single-bit addressability which is often necessary in traditional descriptions of quantum computing. The typical structure of a QCA is straightforward: a linear array of qubits that carry out computation through nearest-neighbor interactions and globally applied single-bit fields. This could potentially be a substantial improvement over the traditional quantum
computing paradigm in terms of the practical feasibility of implementing such a system. Namely, it can be difficult to manipulate and address individual qubits in order to apply quantum logic gates. Moreover, QCA takes advantage of naturally occurring pairwise interactions to implement two qubit gates. It may therefore be preferable to carefully arrange a system of qubits and apply only global external fields in order to perform the desired process, as in QCA.

The QCA proposal of interest here is that presented by Brennen and Williams (so-called block partitioned QCA) [22], a 1-dimensional QCA that assumes nearest-neighbor interactions only between the qubits. The QCA consists of 2 species of qubits, an even-ordered species $A$ and an odd-ordered species $B$. The two species are updated in parallel one after the other, i.e. an update on the $B$ species, followed by an update on the $A$ species. Note that this is a common feature of QCA as a method to circumvent the no-cloning theorem [23, 24] which forbids the copying of a quantum state, meaning that there is no way to copy and store the state before an update in order to update all cells simultaneously. It allows for either periodic boundary conditions, or boundaries with auxiliary qubits fixed in a particular state at the ends of a chain. Worth noting is that this QCA structure meets the minimum requirements [25, 20] for being able to implement a universal quantum computer, provided one allows for a sequence of update rules to be used.

Of course, as is the case with any quantum system, decoherence could potentially affect a QCA in the form of unintended interactions with the environment in which the system exists, spontaneous emission, etc. How viable QCA is as a platform for quantum computing will depend at least in part upon how tolerant it is to errors. Much work has been done on error correction and the modeling of decoherence when it comes to traditional quantum computing [26], but present little investigation of these effects on QCA has been done with the notable exceptions of [27, 28]. The goal herein is to supplement the previous calculations from [22] with various types of errors in order to investigate the consequences of decoherence effects on 1D block-partitioned QCA.

5.2 BQCA Structure

A gate sequence consists of both single-bit portions which perform some unitary operation on a given qubit independent of the rest of the system and two-qubit gates which, in this scheme, correspond to local pairwise interactions between neighboring qubits. Constructing a general gate sequence for BQCA is discussed in [22], and this will not be recapitulated in
great detail here—only the crucial parts needed to understand the operations used in the context of this thesis are presented.

It is first necessary to be able to define an operator which acts on a single qubit subsystem of the QCA. As an example, the $\sigma_x$ operator which acts on the $i$-th subsystem is defined as

$$\sigma_x^i = \bigotimes_{0}^{i-1} 1_2 \otimes \sigma_x \otimes \bigotimes_{i+1}^{n-1} 1_2.$$  \hfill (5.1)

Similarly, the $\sigma_x$ operator which acts on all even-ordered subsystems is the sum of even operators defined as above:

$$\sigma_x^A = \sum_{i=0,2,...} \sigma_x^i.$$  \hfill (5.2)

The operator corresponding to the interaction Hamiltonian which implements two-qubit gates is

$$H_I = \sum_{j=0}^{N-1} \sigma_x^j \sigma_x^{j+1}.$$  \hfill (5.3)

For the gate sequences discussed herein an operator which rotates about the Bloch vector $\vec{m} = (\sin \frac{\pi}{4}, 0, \cos \frac{\pi}{4})$ is also required. For the even-ordered species this is

$$\sigma_{\vec{m}}^A = \frac{1}{\sqrt{2}} (\sigma_x^A + \sigma_z^A).$$  \hfill (5.4)

Assembling these pieces as prescribed in [22] gives the desired update rule which, among other things, can be used to transport quantum information along the chain of qubits. It updates the $A$ species of a QCA with boundaries fixed in the state $|0\rangle$

$$M^A = e^{-i\frac{\pi}{2} \sigma_{\vec{m}}^A} e^{-i\frac{3\pi}{4} \sigma_z^0} e^{-i\frac{3\pi}{4} H_I} e^{-i\frac{\pi}{2} \sigma_z^A} e^{i\frac{\pi}{2} \sigma_{\vec{m}}^A}.$$  \hfill (5.5)

and similarly for the $B$ species

$$M^B = e^{-i\frac{\pi}{2} \sigma_{\vec{m}}^B} e^{-i\frac{3\pi}{4} \sigma_z^{n-1}} e^{-i\frac{3\pi}{4} H_I} e^{-i\frac{\pi}{2} \sigma_z^B} e^{i\frac{\pi}{2} \sigma_{\vec{m}}^B}.$$  \hfill (5.6)

These describe a sequence of steps to perform on the system, meaning that read from right to left $M^A$ says to apply a $\sigma_{\vec{m}}$ update to the $A$ species, then apply a $\sigma_z$ update to the $A$ species, followed by allowing the system to evolve undisturbed via pairwise coupling between qubits, a $\sigma_z$ update on just the first qubit (this comes from the boundary conditions for the
pairwise interaction), and finally apply a $\sigma_m$ update to the $A$ species. This sequence of steps, each executed for an appropriate duration denoted by the angle accompanying the operator in the exponent, completes one $M^A$ update. It is often convenient to define one update step as $M = M^A M^B$. One final remark on the notation used; $|\psi\rangle_j$ represents the state of the $j$-th subsystem given the state of the entire system $|\psi\rangle$ which is constructed as the tensor product between the state of each subsystem, i.e.

$$
|\psi\rangle = \bigotimes_{j=0}^{n-1} |\psi\rangle_j. \quad (5.7)
$$

### 5.2.1 Conditional Update Rules

The formalism described in [22] is more clearly in the spirit of the cellular automata paradigm, and the seemingly complicated update rule $M$ turns out to be the way to implement conditional updates. This will now be briefly mentioned, but it is not important for the actual calculations in the coming chapters. The update rule $M$ that is applied over a three-cell neighborhood is defined as

$$
M(u_{00}, u_{01}, u_{10}, u_{11}) = |00\rangle \langle 00| \otimes u_{00} + |01\rangle \langle 01| \otimes u_{01} + |10\rangle \langle 10| \otimes u_{10} + |11\rangle \langle 11| \otimes u_{11}. \quad (5.8)
$$

The meaning of this update rule is straightforward—updating a qubit based on the state of its two neighbors. For instance a qubit with its left neighbor in the state $|1\rangle$ and its right neighbor in the state $|0\rangle$ would have the unitary operator $u_{10}$ applied to it. This of course generalizes to the case where the neighbors are not in basis states, which would result in varying amounts of each unitary operator being applied to the central qubit.

Considering the update rule in this way is clearly more like typical descriptions of CA, and [22] provides recipes for constructing gate sequences given the four unitary operators $u$ which are conditionally applied. The gate sequence description of $M$ is more akin to a set of rules that an experimenter would implement in practice and is therefore useful for modeling errors by way of slightly modifying the gate sequence in ways that might correspond to physically realistic errors.
Chapter 6

Quantum Information Processing

Protocols

The protocols that follow, including the spatial transfer and exchanging of quantum information and the distribution of entanglement among a system of qubits, represent some of the basic operations one would need to be able to perform in order to build up more complicated computations. The hope is therefore that they will be a representative platform for investigating the effects of the various proposed error modes.

All numerical simulations were performed using Python. In particular, the Quantum Toolbox in Python (QuTiP) package [13] was used for boilerplate operations and to streamline the implementation of this model.

6.1 State Transfer

The character of this protocol is straightforward—start with a seed qubit on one end of a chain initialized to some arbitrary state

$$|\psi_i\rangle = (\alpha |0\rangle + \beta |1\rangle) \otimes \prod_{1}^{n-1} |0\rangle$$  \hspace{1cm} (6.1)

and transfer this state through intermediary qubits initialized to the state $|0\rangle$. End with the state transferred to the qubit at the opposite end of the chain. The update sequence that will accomplish this in terms of $M$ as defined in Eqs. 5.5 & 5.6 is

$$|\psi_f\rangle = \sigma_z^{n-1}[M]^{1/2} |\psi_i\rangle$$  \hspace{1cm} (6.2)

The resulting space-time diagram which plots the excitation probability for each qubit at each application of $M$ is shown in Fig. 6.1. The simulation was run for $n = 10$ qubits,
Figure 6.1: Diagram showing $\langle \sigma_+ \sigma_- \rangle$ as the cell shading for each qubit ($n = 10$) at each update step for the transfer of a quantum state $|\psi_i\rangle_0 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ from one end of a chain of qubits to the other end. Not shown are boundary qubits at both ends fixed in the state $|0\rangle$.

resulting in 5 applications of $M$.

When considering various errors in the update sequence, it will be necessary to calculate fidelity. For the state of the last qubit calculated with no errors $|\psi_c\rangle = |\psi_i\rangle_0$ (this denotes the initial state of the first qubit which should be losslessly transferred) and the state of the entire system calculated with errors $|\psi_f\rangle$, the fidelity squared is

$$F^2 = \langle \psi_c | \rho_{n-1} | \psi_c \rangle$$  \hspace{1cm} (6.3)

where $\rho_{n-1} = \text{Tr}_{0, \ldots, n-2}(|\psi_f\rangle \langle \psi_f|)$ is the reduced density matrix for the last qubit obtained taking a partial trace over all but the last subsystem.

### 6.2 State Swap

The goal of this protocol is to start with qubits seeded to arbitrary states on either end of the chain, and swap the states between the ends.

$$|\psi_i\rangle = (\alpha |0\rangle + \beta |1\rangle) \otimes \bigotimes_{1}^{n-2} |0\rangle \otimes (\epsilon |0\rangle + \mu |1\rangle)$$  \hspace{1cm} (6.4)
Figure 6.2: Diagram showing $\langle \sigma_+ \sigma_- \rangle$ as the cell shading for each qubit ($n = 10$) at each update step for the task of swapping the state of two qubits $|\psi_i\rangle_0 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|\psi_i\rangle_{n-1} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ at the opposite ends of a 1D lattice. Not shown are boundary qubits at both ends fixed in the state $|0\rangle$.

The gate sequence to achieve the swap is

$$|\psi_f\rangle = \sigma_0^{-\sigma_n^{-1}} M^B[M]^{n/2} |\psi_i\rangle$$  \hspace{1cm} (6.5)

The spacetime diagram for the state swap protocol is shown in Fig. 6.2.

Again it is useful to define fidelity squared in terms of the correct final state of the first and last qubits $|\psi_c\rangle = |\psi_i\rangle_{n-1} \otimes |\psi_i\rangle_0$ and the state of the entire system obtained with errors $|\psi'_f\rangle$ as

$$F^2 = \langle \psi_c | \rho_{0,n-1} | \psi_c \rangle$$  \hspace{1cm} (6.6)

where $\rho_{0,n-1}$ is the reduced density matrix for the first and last subsystems.

### 6.3 GHZ State Generation

Start with the central qubit seeded in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and end with all qubits maximally entangled (so-called Greenberger-Horne-Zeilinger (GHZ) state [29]). For $n = 4k + 2$ qubits, the location of the “central” qubit is the $n/2 - 1$ site. The initial state can therefore be
constructed as

\[ |\psi_i \rangle = \bigotimes_0^{n/2-2} |0 \rangle \otimes \frac{1}{\sqrt{2}} (|0 \rangle + |1 \rangle) \otimes \bigotimes_{n/2}^{n-1} |0 \rangle \] (6.7)

and the update sequence is

\[ |\psi_f \rangle = e^{-i\frac{\pi}{4} \sigma_0^z} |\psi_i \rangle \] (6.8)

The spacetime diagram for GHZ state preparation is shown in Fig. 6.3. Here fidelity squared in terms of the correct final state \( |\psi_f \rangle \) and the state obtained with errors \( |\psi'_f \rangle \) is

\[ F^2 = |\langle \psi'_f | \psi_f \rangle|^2 \] (6.9)

Note that there is no partial trace in this case, because the final state involves entanglement between all qubits and hence the fidelity must account for all subsystems.

### 6.4 Entangled Pair

The first update which distributes the initial state in the central qubit to the adjacent pairs of qubits requires a new update rule other than \( M \) from Eqs. 5.5 & 5.6. This update rule \( M_2 \) (again the method for obtaining this rule is described in [22]) is
This starting state is the same as it was for the task of generating a GHZ state as in Eq. 6.7. Again, it starts with the central qubit seeded in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. For $n = 4k+2$ qubits, the location of the “central” qubit is the $n/2 - 1$ site. After the update sequence, the two qubits at the ends will be maximally entangled with one and other. The total update sequence to accomplish this is

$$M_2 = e^{-i\frac{\pi}{4}\sigma_z} e^{i\frac{\pi}{2}\sigma_x} e^{-i\frac{\pi}{2}\sigma_y} e^{-i\frac{\pi}{8}H_I} e^{i\frac{3\pi}{8}\sigma_z} e^{-i\frac{\pi}{2}\sigma_x} e^{-i\frac{\pi}{2}\sigma_y} e^{-i\frac{\pi}{4}H_I} e^{i\frac{3\pi}{2}\sigma_z} \times e^{i\frac{\pi}{2}\sigma_x} e^{-i\frac{\pi}{4}H_I} e^{-i\frac{\pi}{2}\sigma_y} e^{-i\frac{\pi}{2}\sigma_x} e^{i\frac{\pi}{2}\sigma_y} e^{-i\frac{\pi}{4}H_I}$$

(6.10)

The spacetime diagram for the entangled pair preparation is shown in Fig. 6.4.
Chapter 7

Error Models

7.1 Bit and Phase Flip Errors

In this error model, spontaneous bit flips ($\sigma_x$) and phase flips ($\sigma_z$) have some probability of occurring. At each update, each qubit has a probability $\delta$ of receiving an error; bit and phase flips are equally likely. As is shown in Fig. 7.1, fidelity falls rapidly as $\delta$ increases.

However in a practical scenario, it is likely that the probability for this type of error would be non-zero, but small. As such it is perhaps most interesting to consider the region where $\delta$ is small. To this end, the plots in Fig. 7.1 were truncated at the vertical dotted line, leaving $\delta \in [0, 0.05]$. Table 7.1 lists the slope of a linear regression fit to this small error region for each of the model protocols.

<table>
<thead>
<tr>
<th># Qubits</th>
<th>State Transfer</th>
<th>State Swap</th>
<th>GHZ State</th>
<th>Entangled Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-2.9 (.98)</td>
<td>-6.0 (.99)</td>
<td>-5.3 (.99)</td>
<td>-3.9 (.99)</td>
</tr>
<tr>
<td>6</td>
<td>-4.1 (.98)</td>
<td>-7.9 (.99)</td>
<td>-6.9 (.99)</td>
<td>-5.7 (.99)</td>
</tr>
<tr>
<td>8</td>
<td>-4.9 (.95)</td>
<td>-9.8 (.98)</td>
<td>-11.5 (.98)</td>
<td>-7.9 (.99)</td>
</tr>
<tr>
<td>10</td>
<td>-6.2 (.97)</td>
<td>-10.4 (.96)</td>
<td>-12.1 (.96)</td>
<td>-8.2 (.98)</td>
</tr>
</tbody>
</table>

Table 7.1: This table lists the slope $m$ and squared sample correlation coefficient ($R^2$) of the linear regression for the Fidelity squared vs. error probability plots, with small error probabilities $\delta \in [0, 0.05]$. The values can be used to describe fidelity squared as a function of error probability in this range, as $F^2 = 1 + m\delta$. Simulations were averaged over 1000 independent trials. Note the initial states used for the swap and transfer protocols are the same as those shown in Figs. 6.1 & 6.2.

To be clear, the decay in fidelity shown in Fig. 7.1 is an ensemble behavior resulting from the average of many independent trials. This is essentially a Monte Carlo approach, and members of the ensemble at a given $\delta$ may have any fidelity in the interval $[0, 1]$. 

28
Figure 7.1: Plots showing fidelity squared as a function of error probability (assessed on a per-qubit basis at each update step) $\delta$ for $n = 4, 6, 8, 10$ qubits. These results are averaged over 5000 independent trials. The fidelity rapidly decreases in all cases, and does so faster for more qubits. The horizontal dashed line represents the expected fidelity between two random quantum states [30], in this case $\frac{1}{2}$.

### 7.2 Timing Errors

The application of single-bit gates to the system of qubits would likely be realized by globally applied control fields. The prescribed rotation angle could then be achieved by applying a laser field with a given intensity for some duration of time which depends upon the intensity of the laser. Moreover, the interaction portion of the update is achieved by allowing the system to evolve undisturbed for some amount of time. It is therefore reasonable to suggest that there may not be perfect precision when it comes to the intensity of the globally applied fields, nor the temporal duration of the various updates. To model this effect, errors in the rotation amount sampled from a Gaussian distribution centered at 0 with width $\sigma$ are considered. An error is assessed on every angle at every update, for example $\frac{\pi}{2} \mapsto \frac{\pi}{2} + r$ where $r$ is a random value sampled from the Gaussian distribution. As was the case with random errors of the type described in the previous section, the fidelity rapidly falls with increasing $\sigma$. And so again the data is truncated and fit to a line, this time for small error distribution widths $\sigma \in [0, 0.05]$. The values for the slope and squared sample correlation coefficient of the linear regression are reported in Table 7.2. Again these results were obtained from a Monte Carlo type simulation. These results can be interpreted to mean that for an ensemble of systems under the effects of a Gaussian distribution of timing errors of width $\sigma \in [0, 0.05]$, the expected average fidelity is $F^2 = 1 + m\sigma$. 

29
### Table 7.2

<table>
<thead>
<tr>
<th># Qubits</th>
<th>State Transfer</th>
<th>State Swap</th>
<th>GHZ State</th>
<th>Entangled Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-0.8 (.98)</td>
<td>-1.8 (.98)</td>
<td>-1.7 (.96)</td>
<td>-4.6 (.96)</td>
</tr>
<tr>
<td>6</td>
<td>-1.6 (.97)</td>
<td>-3.1 (.95)</td>
<td>-3.3 (.98)</td>
<td>-6.8 (.96)</td>
</tr>
<tr>
<td>8</td>
<td>-2.2 (.98)</td>
<td>-4.0 (.98)</td>
<td>-6.2 (.98)</td>
<td>-7.9 (.97)</td>
</tr>
<tr>
<td>10</td>
<td>-2.7 (.97)</td>
<td>-4.8 (.99)</td>
<td>-9.6 (.98)</td>
<td>-8.3 (.97)</td>
</tr>
</tbody>
</table>

This table lists the slope \( m \) and squared sample correlation coefficient (\( R^2 \)) of the linear regression for the Fidelity squared vs. Gaussian error width plots, with small Gaussian widths \( \sigma \in [0, 0.05] \). The values can be used to describe fidelity as a function of error rate in this range, as \( F = 1 + m\sigma \). Simulations were averaged over 150 independent trials.

#### 7.3 Continuous Interactions

The model in [22] assumes that the pairwise interactions between qubits is somehow “turned off” during applications of single-bit fields. Turning off the pairwise coupling is not something that is trivial to achieve physically, and so at best one might hope that the timescale of the single-bit gates is much less than the timescale of the interactions between qubits. The following modification to the update rule (Eqs. 5.5 & 5.6) aims to model interactions occurring during the application of single-bit gates by way of the introduction of an interaction parameter \( \gamma \) as follows:

\[
M = e^{-i\frac{\pi}{2}(\sigma_{\hat{m}}+\gamma H_I)} e^{-i\frac{\pi}{4}H_I} e^{-i\frac{\pi}{2}(\sigma_z+\gamma H_I)} e^{i\frac{\pi}{2}(\sigma_{\hat{m}}-\gamma H_I)}
\]  

Eq. 6.10 is modified in the same way for the entangled pair protocol but is not shown explicitly for the sake of brevity. In the case of modeling continuous interactions, consistently decreasing fidelity with increasing \( \gamma \) is not observed as was the case with the last two error modes. As such, approximating a line for small values is not as useful here, and instead the plots for each of the sample protocols are presented in Fig. 7.2, with \( \gamma \in [0, 1] \).

Interestingly in the case of the state transfer and swap problems shown in Figs. 7.2a & 7.2b, there are large values of \( \gamma \) that correspond to relatively high fidelity. Recall that the update sequence for these problems is able to transport arbitrary initial states and so it is possible that these high fidelity regions are just fortuitous occurrences that depend on the initial state that happened to be used. To investigate this, simulations were run for many random initial states in the seed qubits and averaged together. Random initial states for the seed qubits were generated as

\[
|\psi_0\rangle = \alpha |0\rangle + e^{i\phi} \sqrt{1 - \alpha^2} |1\rangle
\]  

where \( \alpha \) is a random number in the range \([0, 1]\) and \( \phi \) is a random angle in the range \([0, 2\pi]\).
Figure 7.2: Fidelity squared as a function of the interaction parameter $\gamma$ defined in Eq. 7.1, for the (a) the state transfer problem, (b) the state swap problem, (c) the GHZ state generation problem, and (d) the entangled pair problem. The dashed lines correspond to the average fidelity between two random quantum states. Plots are shown for $n = 4, 6, 8, 10$ qubits, as labeled.
Figure 7.3: Fidelity squared as a function of the interaction parameter $\gamma$ defined in Eq. 7.1, for the (a) the state transfer protocol and (b) the stateswap protocol. Both figures are averaged over 1000 randomly chosen initial states for the seed qubits in an effort to eliminate any coincidental dependence on the initial state. The dashed lines correspond to the average fidelity between two random quantum states. Plots are shown for $n = 4, 6, 8, 10$ qubits, as labeled.

This will therefore be guaranteed to be normalized and will sample evenly from all possible relative phases.

Fig. 7.3 shows the results for the same simulations as Figs. 7.2a & 7.2b but averaged over 1000 random initial states as described. For the most part it appears that the high fidelity regions are damped out by the averaging and that at high $\gamma$ the fidelity is generally not much better than for random states.

### 7.3.1 Compensation

After modeling interactions occurring during the single-bit portions of the update, the question that naturally arises is whether or not this can be corrected for by decreasing the duration of the interaction portion that was originally required. To test this, consider the introduction of a compensation parameter $C$ which decreases the duration of the interaction portion of the update as follows

$$M = e^{-i \frac{\pi}{4} (\sigma_m + \gamma H_I)} e^{-i \frac{3\pi}{4}(1-C)H_I} e^{-i \frac{\pi}{4} (\sigma_z + \gamma H_I)} e^{i \frac{\pi}{4} (\sigma_m - \gamma H_I)}.$$  \hspace{1cm} (7.3)

Again Eq. 6.10 is modified in the same way for the entangled pair protocol.
Figure 7.4: Plots showing fidelity squared as a function of compensation parameter $C$ with various small values of $\gamma$ (denoted by line color) for the tasks of (a) transferring a quantum state from one end of a chain of qubits to the other, (b) exchanging quantum states between opposite ends of a chain, (c) generating a GHZ state in a chain, and (d) entangling a pair of qubits at the ends. All plots used $n = 10$ qubits, with boundary qubits fixed in the state $|0\rangle$ at the ends.
Fig. 7.4 shows how $C$ affects fidelity for the sample problems. Six different values of the interaction parameter $\gamma$, denoted by line color, are shown in each plot. Note that in all cases except for 7.4d for small values of $\gamma$, the fidelity increases to a value higher than it would have been with no compensation ($C = 0$). This means that in practice if pairwise couplings occur during the applications of global single-bit fields (i.e. the pairwise coupling is not turned off), it is always beneficial to reduce the duration of the pairwise interaction.

Again in the case of the transfer and swap protocols there is a question of how the behavior depends on the initial states used. To investigate this it is again useful to consider averaging over random initial states, generated as described in Eq. 7.2, for these protocols.

From Fig. 7.5, it seems when $\gamma$ is large, averaging over random states results in $F^2$ of roughly the expected value ($F^2 = 0.5$ for Fig. 7.5a and $F^2 = 0.25$ for Fig. 7.5b), thereby removing artifacts which were a happenstance result of the initial states used in Fig. 7.4. Notably however there still appears to be at least some benefit to using some compensation in the average case. Perhaps the most physically relevant examples of this are when $\gamma$ is small and there is a slight peak when $C$ is small. In an experiment, this fact could be utilized to increase the effectiveness of these protocols by reducing the duration of the pairwise interaction update by an amount that corresponds to the $C$ value at which the peak
occurs, assuming the control pulses are fast (corresponding to a particular small $\gamma$). And since this peak still exists even in the average case, there is therefore hope that a $C$ which will maximize fidelity can be computed for any state.
Chapter 8

QCA Conclusions

8.1 Discussion

This work has successfully reproduced the results of [22] and expanded upon those results in order to model several types of physically realistic errors. Unsurprisingly these results show that the success of a BQCA update rule is very sensitive to random bit and phase flip errors assessed on the constituent qubits as well as errors in the rotation amounts prescribed in the update sequence (so-called “timing errors” herein). In the case of modeling pairwise interactions being always on there are some values of the interaction parameter $\gamma$ for which fidelity is high. This could possibly be exploited by performing slow single-bit updates, however the particular values of $\gamma$ which correspond to high fidelity depend upon the state in question when it comes to the problems of transferring quantum states between the ends of a chain and this behavior is vanishes when averaged over many random states. The interactions occurring during the single-bit gates can be somewhat ameliorated by the introduction of compensation which reduces the duration of the intended interaction portion of an update sequence. This seems to imply, at least for three of the sample problems used, that the success of a QCA update sequence can always be slightly improved by reducing the duration of the interaction portion of an update by an amount which depends on the timescale of the single-bit updates.

8.2 Future Work

It may be possible to formulate QCA in a continuous-time fashion—quantum trajectories could actually be well-suited for this purpose. Essentially the decoherence effects would correspond to collapse operators. This may be a more faithful representation of errors or even open up modelling different types of errors, for instance a bit flip occurring during an update step and not in-between as it is now. It could also be interesting to propose error
correction protocols akin to those developed in the traditional QC description. Last, similar simulations could be conducted for various other formulations of QCA in order to verify that similar results are obtained and these conclusions are model-independent as they should be if they are indeed representing physical reality.
Appendix A

Trajectories Code

This is the code in Python for a single trajectory, given the diagonal effective Hamiltonian \( \text{Heff}_{\text{diag}} \), the similarity transformation matrices \( S \) and \( S^{-1} \), the initial state (ket) vector multiplied on the left by \( S^{-1} \psi_0 \), a list of times at which to calculate expectation values \( t_{\text{list}} \), the operators for which to calculate expectation values \( e_{\text{ops}} \), and the collapse operators \( c_{\text{ops}} \). This code depends on the Python packages \texttt{numpy}, \texttt{scipy}, and \texttt{random}.

```python
def one_traj(Heff_diag, S, Sinv, psi0_nb, tlist, e_ops, c_ops, num_eops, num_cops):
    random = Random()
    exp_vals = list(np.zeros(len(tlist), dtype=complex) for i in range(num_eops))
    which_oper = list()
    collapse_times = list()
    # Find the initial expectation values:
    for i in range(num_eops):
        exp_vals[i][0] += mi_expect(e_ops[i], psi0_nb)
    tmax = tlist[-1]
    prevjump = 0
    ind = 1
    try:
        nextjump = brentq(find_t, 0, tmax, xtol=1e-6, args=(psi0_nb, S, Heff_diag, random.random()))
        collapse_times.append(nextjump)
        # dmm this has to be here, or there are times listed for the tmax case
    except:
        nextjump = tmax
        # the case that there is no zero on the interval [0, tmax]
    flag = num_cops >= 1
    while ind < len(tlist):
        if nextjump < tlist[ind] and flag:
            psi0_nb = np.dot(diag_expm(np.multiply(Heff_diag, (nextjump - prevjump))), psi0_nb)
        if num_cops == 0: n_col = 0
        else: # choose which collapse operator acts on the state
            rand, tot, count, weight = random.random(), 0, -1, 0
            prob = numpy.zeros(num_cops)
            for i in range(num_cops):
                prob[i] = (np.linalg.norm(np.dot(c_ops[i], psi0_nb)))**2
                weight += prob[i]
            prob = prob/weight
            while rand > tot:
                count += 1
```

38
tot += prob[count]
    n_col = count
    which_oper.append(n_col)
    # allow collapse operator to act on state vector at t = nextjump
    psi0_nb = np.dot(c_ops[n_col], psi0_nb)
    psi0_nb = psi0_nb / np.linalg.norm(psi0_nb)
    psi0_nb = np.dot(Sinv, psi0_nb) # return to the basis where Heff is diagonal
    prevjump = nextjump
    try:
        nextjump += brentq(find_t, 0, tmax - prevjump, xtol = 1e-6, args=(psi0_nb, S, Heff_diag, random.random()))
        collapse_times.append(nextjump)
    except:
        flag = False
        # calculate expectation values at t = tlist[ind]
        else:
            psi0_nb = np.dot(diag_expm(np.multiply(Heff_diag, (tlist[ind] - prevjump))), psi0_nb)
            psi0_nb = np.dot(S, psi0_nb)
            psi0_nb = psi0_nb / np.linalg.norm(psi0_nb)
            psi0_nb = np.dot(Sinv, psi0_nb)
            for i in range(num_eops):
                exp_vals[i][ind] += mi_expect(e_ops[i], psi0_nb)
            prevjump = tlist[ind]
            ind += 1
    return exp_vals, collapse_times, which_oper
Appendix B

QCA Code Snippets

The Python function which constructs the update rules $M^A, M^B$ given a list of operators `oper` and coefficients `coeff` for each of the unitary operators which comprise the update rules. An interaction parameter $\gamma$ and a compensation parameter $C$ can optionally be specified.

```python
def build_M(sigma, oper, coeff, gamma=False, C=False):
    # Construct an update sequence M for running a QCA simulation
    # returns M_A (update rule for A species), M_B (B species)
    if not gamma and not C:
        M_A = (1.j * pi * coeff[0] * sigma[oper[0] + '_a']).expm()
        for i in range(1, len(oper)):
            M_A = M_A * (1.j * pi * coeff[i] * sigma[oper[i] + '_a']).expm()
        M_B = (1.j * pi * coeff[0] * sigma[oper[0] + '_b']).expm()
        for i in range(1, len(oper)):
            M_B = M_B * (1.j * pi * coeff[i] * sigma[oper[i] + '_b']).expm()
        return M_A, M_B
    elif isinstance(gamma, float):
        # assumes first operator is not zsum
        M_A = (1.j * pi * coeff[0] * sigma[oper[0] + '_a'] - 1.j * pi * abs(coeff[0]) * gamma * sigma['zsum_a']).expm()
        M_B = (1.j * pi * coeff[0] * sigma[oper[0] + '_b'] - 1.j * pi * abs(coeff[0]) * gamma * sigma['zsum_b']).expm()
        for i in range(1, len(oper)):
            if oper[i] != 'zsum': # include the interaction Hamiltonian in the control pulses (single bit gates)
                M_A = M_A * (1.j * pi * coeff[i] * sigma[oper[i] + '_a'] - 1.j * pi * abs(coeff[i]) * gamma * sigma['zsum_a']).expm()
                M_B = M_B * (1.j * pi * coeff[i] * sigma[oper[i] + '_b'] - 1.j * pi * abs(coeff[i]) * gamma * sigma['zsum_b']).expm()
        else:
            if C:
                M_A = M_A * (1.j * pi * coeff[i] * (1 - C) * sigma[oper[i] + '_a']).expm()
                M_B = M_B * (1.j * pi * coeff[i] * (1 - C) * sigma[oper[i] + '_b']).expm()
            else:
                M_A = M_A * (1.j * pi * coeff[i] * sigma[oper[i] + '_a']).expm()
                M_B = M_B * (1.j * pi * coeff[i] * sigma[oper[i] + '_b']).expm()
        return M_A, M_B
    else:
        raise ValueError("Gamma must be provided if C is provided")
```
The Python function which constructs operators that act on an \( N \)-qubit QCA.

```python
def init_operators(N, m_vec):
    # \( N \) is the number of qubits that the operators will be acting on
    # \( m_{\text{vec}} \) is the Bloch vector, supplied as a tuple \((x,y,z)\)

    sigma = {'x': list (), 'y': list (), 'z': list (), '+-': list (), 'm_a':None, 'm_b':None,
             'x_a':None, 'y_a':None, 'z_a':None, 'x_b':None, 'y_b':None, 'z_b':None,
             'zsum_a':None, 'zsum_b':None, 'zsum':None}

    # note the follow correspondence:
    # \( x_a(b) \rightarrow \) even (odd) \( \sigma_x \) total
    # \( y_a(b) \rightarrow \) even (odd) \( \sigma_y \) total
    # \( z_a(b) \rightarrow \) even (odd) \( \sigma_z \) total
    # \( m_a(b) \rightarrow \) even (odd) \( \sigma \) directed along the Bloch vector
    # \( zsum_a(b) \rightarrow \) even (odd) sum of \( \sigma_z \) \( j \) * \( \sigma_z \) \( j+1 \) for "U([t])" with B.C. included
    for i in range(N):
        if i == 0:
            sigma['x'].append ( sigmax ())
            sigma['y'].append ( sigmay ())
            sigma['z'].append ( sigmaz ())
            sigma['+-'].append ( sigmap () * sigmam ())
        else:
            sigma['x'].append ( qeye (2))
            sigma['y'].append ( qeye (2))
            sigma['z'].append ( qeye (2))
            sigma['+-'].append ( qeye (2))

    for j in range(1,N):
        if i == j:
            sigma['x'][i] = tensor ( sigma['x'][i], sigmax ())
            sigma['y'][i] = tensor ( sigma['y'][i], sigmay ())
            sigma['z'][i] = tensor ( sigma['z'][i], sigmaz ())
            sigma['+-'][i] = tensor ( sigma['+-'][i], sigmap () * sigmam ())
        else:
            sigma['x'][i] = tensor ( sigma['x'][i], qeye (2))
            sigma['y'][i] = tensor ( sigma['y'][i], qeye (2))
            sigma['z'][i] = tensor ( sigma['z'][i], qeye (2))
            sigma['+-'][i] = tensor ( sigma['+-'][i], qeye (2))

    # Collect the sums of the operators
    sigma['x_a'], sigma['y_a'], sigma['z_a'] = sigma['x'][0], sigma['y'][0], sigma['z'][0]
    sigma['x_b'], sigma['y_b'], sigma['z_b'] = sigma['x'][1], sigma['y'][1], sigma['z'][1]

    for i in range(2,N):
        if not i % 2:
            sigma['x_a'] += sigma['x'][i]
            sigma['y_a'] += sigma['y'][i]
            sigma['z_a'] += sigma['z'][i]
        else:
            sigma['x_b'] += sigma['x'][i]
            sigma['y_b'] += sigma['y'][i]
            sigma['z_b'] += sigma['z'][i]

    sigma['m_a'] = m_vec[0] * sigma['x_a'] + m_vec[1] * sigma['y_a'] + m_vec[2] * sigma['z_a']
```
Python code which simulates transferring a quantum state between the ends of a chain of qubits (as in Fig. 6.1).

```python
N = 10
steps = N / 2
psi0 = (basis(2) + basis(2,1)).unit()
for i in range(N-1): psi0 = tensor(psi0, basis(2))
sigma = init_operators(N,(1 / sqrt(2), 0, 1 / sqrt(2)))
oper = ['m', 'zsum', 'z', 'm']
coeff_ = [-1./2, -3./4, -1./2, 1./2]
M_A, M_B = build_M(sigma,oper,coeff_)
M = M_A * M_B

results = []
for i in range(N): results.append(numpy.zeros(steps + 1))
for j in range(N): results[j][0] = expect(sigma['+-'][j], psi0)

for i in range(1,steps + 1):
    psi0 = M * psi0
    if i == steps:
        psi0 = sigma['z'][-1] * psi0
    for j in range(N): results[j][i] = expect(sigma['+-'][j], psi0)
```
Appendix C

Why Simulations Used 10 Qubits

For a composite system comprised of \( N \) two-state systems, the Hilbert space dimensionality is

\[
D = 2^N
\]  \hspace{1cm} (C.1)

and the number of entries in the matrix representation of an operator which acts on the composite system is the square of the Hilbert space dimensionality, \( D^2 \). In principle each of the entries in such a matrix can be a complex number, and complex numbers are implemented in Python as two double precision floating point numbers in C (one for the real and imaginary parts respectively). Therefore a single complex number represented in Python requires 8 bytes for each part, or 128 bits altogether. A dense matrix which represents an operator would then require

\[
D^2 \times 128 \text{ bits} = 2^{2N+7} \text{ bits.} \]  \hspace{1cm} (C.2)

For \( N = 10 \), which was used predominantly in this thesis, this comes out to 16 Megabytes per matrix; for \( N = 12 \) it is 256 MB and for \( N = 14 \) it is 4 GB. Clearly it rapidly becomes difficult or impossible to run simulations on modern hardware just on the basis of the amount of physical memory a consumer-grade machine can have. These numbers can be significantly reduced by taking advantage of the fact that the operators are generally sparse, however even with the additional programming that would require, it might at best allow the simulations to go to \( N = 12 \) or 14 qubits since the limiting factor is not just physical memory but also the amount of time it takes to carry out calculations which also scales exponentially. It is for these reasons that the simulations conducted herein primarily used \( N = 10 \) qubits which is thought to be enough to not be trivial and to produce any interesting features, but at the same time allows the calculations to be performed in a reasonable time frame.
References


