3-10-2006

Type II Quantum Computing Algorithm for Computational Fluid Dynamics

James A. Scoville

TYPE II QUANTUM COMPUTING ALGORITHM FOR COMPUTATIONAL FLUID DYNAMICS

THESIS

James A. Scoville, Second Lieutenant, USAF

AFIT/GAP/ENP/06-17

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GAP/ENP/06-17

TYPE II QUANTUM COMPUTING ALGORITHM FOR COMPUTATIONAL FLUID DYNAMICS

THESIS

Presented to the Faculty

Department of Engineering Physics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science (Applied Physics)

James A. Scoville, BS

Second Lieutenant, USAF

March 2006

AFIT/GAP/ENP/06-17

TYPE II QUANTUM COMPUTING ALGORITHM FOR COMPUTATIONAL FLUID DYNAMICS

James A. Scoville, BS
Second Lieutenant, USAF

Approved:

David E. Weeks (Chairman)	10 Mar 06 date
Thomas A. Niday (Member)	10 Mar 06 date
William F. Bailey (Member)	10 Mar 2006 date

Abstract

An algorithm is presented to simulate fluid dynamics on a three qubit type II quantum computer: a lattice of small quantum computers that communicate classical information. The algorithm presented is called a three qubit factorized quantum lattice gas algorithm. It is modeled after classical lattice gas algorithms which move virtual particles along an imaginary lattice and change the particles' momentums using collision rules when they meet at a lattice node. Instead of moving particles, the quantum algorithm presented here moves probabilities, which interact via a unitary collision operator. Probabilities are determined using ensemble measurement and are moved with classical communications channels. The lattice node spacing is defined to be a microscopic scale length. A mesoscopic governing equation for the lattice is derived for the most general three qubit collision operator which preserves particle number. In the continuum limit of the lattice, a governing macroscopic partial differential equation—the diffusion equation—is derived for a particular collision operator using a Chapman-Enskog expansion. A numerical simulation of the algorithm is carried out on a conventional desktop computer and compared to the analytic solution of the diffusion equation. The simulation agrees very well with the known solution.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. David Weeks, for his guidance and support throughout the course of this thesis effort. The insight and experience was certainly appreciated. I would also like to thank Dr. Jeffrey Yepez for the countless hours he spent with me on the phone and typing emails discussing my thesis, and for the years of insight he shared with me.

Most importantly I would like to thank my wife for the long nights she put up with, for the vacation that practically wasn't, for the dinners on the table while holding a job, for the time spent reading and correcting my thesis, and for all the love she had for me. I couldn't have done it without you babe—you're a real life superwoman. I love you.

James A. Scoville

Table of Contents

List of Figures

Figure Page

List of Tables

Table Page

TYPE II QUANTUM COMPUTING ALGORITHM FOR COMPUTATIONAL FLUID DYNAMICS

1 Introduction

1.1 Overview

In 1982 Richard Feynman proposed building a computer based on quantum mechanical principles to efficiently simulate quantum systems. In the two decades since, significant progress has been made both theoretically and experimentally towards this end. Following Feynman's vision, in 2002 the Air Force Research Laboratory and the Air Force Office of Scientific Research established a basic research theme called Quantum Computation for Physical Modeling [1]. The goal of this project is to explore quantum algorithms and practical quantum computers to model dynamic physical systems with an exponential increase in computational efficiency.

This thesis supports this goal by extending an algorithm designed to model fluid dynamics using a lattice of interacting quantum systems. This algorithm was used by Yepez to investigate Navier-Stokes equations of fluid dynamics [2-5], the diffusion equation [6-9], and the Burgers equation [10-12]. It is called the *Factorized Quantum Lattice Gas Algorithm* (FQLGA), and it derives its name from algorithms written to model fluid dynamics on classical computers. These classical algorithms model particles on a spatial lattice. The particles move from lattice node to lattice node in discrete time steps. They include classical collision rules to control how particles interact when they

meet at a lattice node, where the distance between nodes is defined to be a microscopic scale length. Since the algorithms consists of a gas of virtual particles moving on a discrete lattice, it is called a *Lattice Gas Algorithm* (LGA) by the fluid dynamics community, or a *Classical Lattice Gas Algorithm* (CLGA) by the quantum computing community to distinguish it from similar quantum lattice gas algorithms. One may use an ensemble of (quantum or classical) lattice gases to develop a finite difference equation known as the mesoscopic *lattice Boltzmann equation*. From this equation it is possible to develop a macroscopic effective field theory in the continuum limit of the lattice, essentially by taking the Taylor series expansion of the lattice Boltzmann equation around local equilibrium in what is known as a *Chapman-Enskog expansion*.

The FQLGA is said to be "factorized" because it is designed to run on a quantum computer that is not fully coherent—that is, on a computer that is factorized into many smaller quantum computers communicating with classical information: a *type II* quantum computer. This sort of computer is interesting because a prototype already exists which has run fluid dynamics simulations [6, 7, 11]. The algorithm developed by Yepez [2-12] uses a lattice of two *quantum bit* (qubit) computers to perform the computations. Instead of moving particles across a one dimensional lattice probabilities are moved, and instead of using collision rules to govern interactions a unitary operator called the *collision operator* is used. This thesis extends this algorithm to run on a three qubit type II quantum computer. The main contributions presented in this paper are the computation of the lattice Boltzmann equations for the most general three qubit collision operator that conserves particle number, and the derivation of the diffusion equation as an effective field theory for a more specific collision operator. In addition, numerical simulations

comparing the diffusion equation FQLGA simulation and the analytic solution of this partial differential equation are presented.

This thesis is meant to be accessible to someone with a reasonable background in quantum physics, but with little exposure to the subjects of quantum computing or fluid dynamics. As such, I will introduce some of the basics in each of these subjects before discussing the FQLGA.

1.2 Organization

This thesis begins with a summary of quantum computing in Section 2. This section is aimed towards those who have studied quantum mechanics but have had little exposure to quantum computing. It briefly discusses what a qubit is, how quantum logic gates perform computations, and how qubit measurement affects the type of information one may obtain from it. I attempt to make this discussion easier to follow by using analogies with the more familiar classical bits and classical logic gates.

Following this, the main categories of quantum algorithms developed thus far are reviewed to give readers an idea of where the FQLGA fits in the world of quantum computing. Subsequently, several types of quantum computers in development are discussed, along with the various challenges associated with constructing each kind of computer. The focus of this section is on Nuclear Magnetic Resonance (NMR) quantum computers because the most advanced quantum computer prototype to date uses NMR technology, and because these machines are best suited for the quantum algorithm developed in this paper.

Since the FQLGA models fluid dynamics, Section 3 introduces the basics of Navier-Stokes fluid dynamics as well as classical lattice gas and lattice Boltzmann algorithms used to simulate these fluids. These classical algorithms can be used as an analogy to better understand their quantum counterparts. A number of important macroscopic dimensionless parameters used to characterize Navier-Stokes fluids are listed and drawn on during an explanation of diffusive ordering. The lattice Boltzmann equation, which comes from the more familiar Boltzmann equation, is also introduced.

Following this brief introduction to quantum computing and classical lattice gas algorithms, Yepez's factorized quantum lattice gas algorithm is introduced in Section 4. The details of this algorithm are laid out and the unitary collision operator is introduced. Following this is an explanation of how the entire algorithm can be contained in a single equation, the *quantum lattice Boltzmann equation* (QLBE). Next, the *local equilibrium* probabilities are derived and subsequently used in the Chapman-Enskog expansion of the QLBE to derive the governing effective field theory of the lattice in the continuum limit. This governing equation turns out to be the one dimensional Burger's equation, which is a second order nonlinear partial differential equation used to model turbulence and shock formation in inelastic gases.

In Section 5 the three qubit FQLGA I have developed is introduced and the most general three qubit collision operator that conserves particle number is derived. The quantum lattice Boltzmann equation is obtained using this operator.

Section 6 introduces a more specific collision operator which yields the diffusion equation as the algorithm's continuum limit governing partial differential equation. A complete derivation of this equation is given starting from the QLBE in section 6.1. In

section 6.2 the results of a numerical simulation carried out on a conventional computer are presented, and compared to the analytic solution of the diffusion equation. Finally, a discussion of the error and of the convergence properties of the algorithm as compared to the analytic solution is included.

2 Quantum Computing Summary

Quantum computing was first proposed by Richard Feynman in 1982 [13, 14]. He noted that there were certain difficulties in simulating quantum mechanical systems on classical computers due to the exponential growth of the problem with system complexity. He suggested developing a computer based on the principles of quantum mechanics to overcome these difficulties. In 1985 David Deutsch expanded on this idea while trying to use the laws of physics to derive a stronger version of the Church-Turing thesis. This thesis states that the Turing model of computation is at least as efficient as any other model of computation, in the sense that if one computational model can solve a problem in time polynomial to the size of a problem, then a probabilistic Turing machine can too [15]. Since the laws of physics are ultimately quantum mechanical, this led Deutsch to develop the modern concept of quantum computers, which are able to efficiently solve problems that are believed to have no efficient solutions on classical computers and Turing machines.

Deutsch developed a simple algorithm that suggested quantum computers would indeed have more computational power than a classical Turing machine (classical computer). Over the next decade additional algorithms were developed culminating in 1994 with Peter Shor's factoring and discrete logarithm algorithms [16], and in 1997 with

Lov Grover's search algorithm [17]. However, despite the immense progress made in developing these algorithms, physicists have so far managed only modest advancements in developing physical quantum computers.

Sections 2.1 through 2.3 are meant to give a brief overview on how quantum computing works.

2.1 Quantum bits

In a quantum computer, qubits replace classical bits. Qubits are analogous to classical bits in that when read (measured) they can only be 0 or 1. However, before measurement—when a computation is being performed—a qubit can be in a superposition of 0 and 1 states. Equation (2.1) shows the most general state of a qubit where $|1\rangle$ and $|0\rangle$ are the basis states for an arbitrary two level quantum system (for example spin up and down in a spin half particle).

$$|q\rangle = e^{i\xi}\sin(\theta)|1\rangle + e^{i\zeta}\cos(\theta)|0\rangle \tag{2.1}$$

2.1.1 Quantum gates

To carry out a quantum algorithm, quantum computers perform unitary operations on qubits. This is analogous to a classical algorithm being composed of 'gates' (AND, OR, NOT, XOR, etc.) that act on classical bits. A straightforward example of this analogy is the classical *exclusive or* (XOR) gate and the quantum *controlled not* (CNOT) gate. The XOR gate is shown in Figure 1a.

The values c_1 and c_2 represent classical bits flowing down a wire (black line) from left to right. They encounter the XOR gate and undergo modulo two addition (denoted

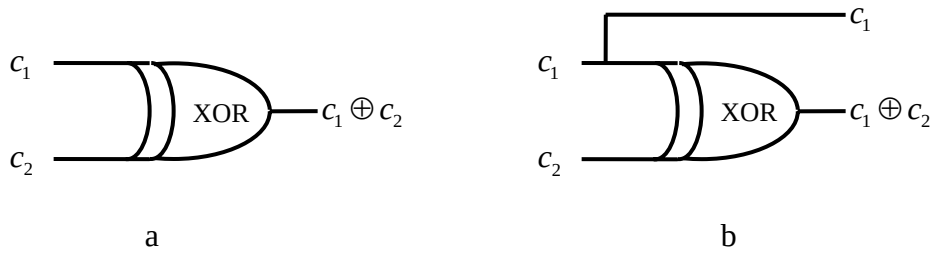

Figure 1. a) An XOR gate. b) A modified XOR gate that preserves the bit c_1.

by $\oplus$) so that the XOR gate output is $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, and $1 \oplus 1 = 0$. This is described in the truth table given in Table 1a. In Figure 1b we have created a modified gate that preserves the bit c_1. Thus, this gate has an equal number of input and output bits. All quantum gates share this property since the number of basis vectors used to represent $|\psi\rangle$ remains constant. Two electrons will always be spin up or down (or both) no matter what unitary operations one performs on them, so it is impossible to create a quantum gate with fewer outputs than inputs. Therefore, the modified XOR gate shown in Figure 1b will be a better analogy to the CNOT gate, as we shall now see.

Table 1. a) Truth table for Figure 1a. b) Truth table for Figure 1b. Inputs are left of the gray bar, outputs are right.

a)

c_1	c_2		$c_1 \oplus c_2$
1	1		0
1	0		1
0	1		1
0	0		0

b)

c_1	c_2		c_1	$c_1 \oplus c_2$
1	1		1	0
1	0		1	1
0	1		0	1
0	0		0	0

Figure 2 shows a quantum CNOT gate. In this diagram, straight lines represent qubit states, and time flows from left to right. Again, the symbol $\oplus$ denotes modulo two addition. A truth table for this gate is shown in Table 2. Note that $|q_2\rangle$ passes

unchanged as long as $|q_1\rangle$ is $|0\rangle$, and $|q_2\rangle$ is changed if $|q_1\rangle$ is $|1\rangle$. Thus $|q_1\rangle$ is called the control qubit and the entire gate is called a control not gate. The truth table given in Table 2 is the same as Table 1b, so that as long as $|q_1\rangle$ and $|q_2\rangle$ are definitely in either state $|1\rangle$ or $|0\rangle$ the gate acts as a modified classical XOR gate. However, if either qubit is in a superposition of $|1\rangle$ or $|0\rangle$ states, then the outputs will also be in a superposition of states. This ability of quantum bits to be a superposition of ones and zeros at the same time is what distinguishes quantum and classical computing.

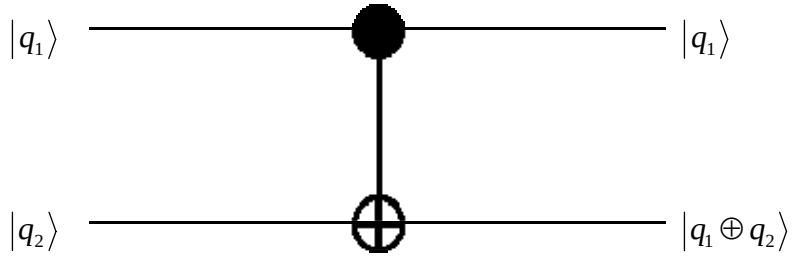

Figure 2. Controlled not gate.

Table 2. CNOT gate truth table. Inputs are left of the gray bar; outputs are right.

q_1	q_2		q_1	$q_1 \oplus q_2$
1	1		1	0
1	0		1	1
0	1		0	1
0	0		0	0

The state of the entire system in Figure 2 is $|\psi\rangle = |q_1\rangle \otimes |q_2\rangle = |q_1\ q_2\rangle$. Note that since $|q_1\rangle$ acts as a control, it is possible for the qubits to become entangled. For instance, if the input state of the CNOT gate is $|\psi\rangle = a|1\ 0\rangle + b|0\ 0\rangle$, which can be factored into $\left(a|1\rangle + b|0\rangle\right) \otimes |0\rangle$, the output state will be $|\psi\rangle = a|1\ 1\rangle + b|0\ 0\rangle$, which

cannot be factored. Before application of the CNOT gate, the state of the second qubit was independent of the first qubit, allowing $|\psi\rangle$ to be factored. However, after application of CNOT, the state of qubit two was directly dependent on the state of qubit one and $|\psi\rangle$ could not be factored. Thus, after the CNOT gate, the measurement of one qubit will immediately determine the state of the other and we say the qubits are entangled.

The CNOT gate, like all quantum gates, is a unitary operation, and can be described in matrix form. If we choose the following to be our basis

$$|1\,1\rangle \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |1\,0\rangle \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad |0\,1\rangle \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad |0\,0\rangle \rightarrow \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.2}$$

then using Table 2, the CNOT matrix is

$$\hat{U}_{cnot} \rightarrow \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.3}$$

Returning to the problem discussed earlier, we can see that this formalism gives the same results.

$$\hat{U}_{cnot}\ \left(a|1\,0\rangle + b|0\,0\rangle\right) = \left(a|1\,1\rangle + b|0\,0\rangle\right)$$
$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ a \\ 0 \\ b \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ 0 \\ b \end{pmatrix} \tag{2.4}$$

One should note that quantum computers have an exponential increase in computational power as one adds more qubits [15]. This is because each additional qubit doubles the number of Hilbert space dimensions so that it has 2^B dimensions, where B is the number of qubits. For instance, if we have three qubits there are 2^3 dimensions with one choice of basis being

$$|111\rangle \to \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \quad |110\rangle \to \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix} \quad |101\rangle \to \begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{pmatrix} \quad |011\rangle \to \begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix}$$

$$|001\rangle \to \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix} \quad |010\rangle \to \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix} \quad |100\rangle \to \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix} \quad |000\rangle \to \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}. \tag{2.5}$$

For purposes of comparison, suppose that we had a classical computer that used three parallel lines, each of which simultaneously transmitted a bit to a central processor. Then at any given time the processor can compute using three bits, which we will define to be a byte. So the classical computer can compute with one byte at a time. In contrast, a similar quantum computer with a three qubit memory—all of which may be in a superposition of $|1\rangle$ and $|0\rangle$ states—can compute with all eight bytes simultaneously. Of course the number of bytes the quantum computer can handle at one time rises

exponentially with additional bits. Thus, for some problems the quantum computer is exponentially more powerful than a classical computer.

2.1.2 Measurement

After a calculation is completed, one must measure the quantum bits to get an answer. Of course, if the output is in a superposition of states, as it is in equation (2.6) below, then one will get a random answer weighted by the coefficients in front of each state. For the equation below, the computer will produce the binary output 1, 1 with probability $|a|^2$.

$$|\psi\rangle = a|1\,1\rangle + b|1\,0\rangle + c|01\rangle + d|00\rangle \tag{2.6}$$

To avoid the embarrassment of getting different answers each time, most quantum algorithms include steps to make the coefficients of the calculated incorrect answers go to zero. However, the binary ones and zeros are not the only way one can code information; it can also be saved in the magnitude of the basis coefficients. To get this information, one must either perform the computation many times on the same computer and average the measured results, or perform the same computation on many identical quantum computers and average these measured results. The second method is called *ensemble*

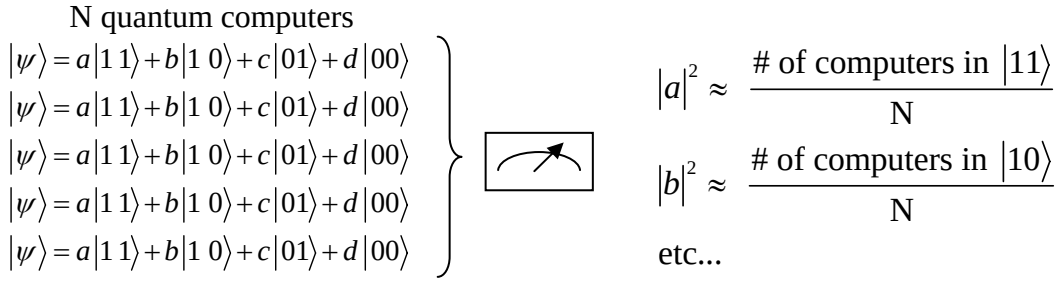

Figure 3. Ensemble measurement averages the measurement results of N identical quantum computers to obtain the magnitude of basis coefficients. The symbol with the arrow in the figure above is used in quantum computing literature to signify the measurement of a quantum system.

measurement and is how information is extracted in a Nuclear Magnetic Resonance (NMR) quantum computer. The FQLGA takes advantage of the ability of NMR machines to do ensemble measurement, so these computers be will discussed in more detail in section 2.3.

For convenience, one may represent the results of an ensemble measurement using projectors or matrices. For instance, suppose we are interested in measuring the probability of finding the second qubit in Figure 3 in the state $|1\rangle$. Then one can write

$$P\left(|q_2\rangle = |1\rangle\right) = |a|^2 + |c|^2 = \left|\langle 11|\psi\rangle\right|^2 + \left|\langle 01|\psi\rangle\right|^2 = \langle\psi|\left(|11\rangle\langle 11| + |01\rangle\langle 01|\right)|\psi\rangle. \quad (2.7)$$

The far right hand side of (2.7) indicates that the probability of the second qubit being $|1\rangle$ is equal to $\langle\psi|\hat{n}_2|\psi\rangle$, where $\hat{n}_2 \equiv |11\rangle\langle 11| + |01\rangle\langle 01|$ is called the number operator and is defined to be the sum of those projectors whose second qubit is $|1\rangle$. Using the basis given in (2.2), we can rewrite the number operator in matrix notation:

$$\hat{n}_2 \equiv |11\rangle\langle 11| + |01\rangle\langle 01| \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (2.8)$$

Similarly, the probability of finding the first qubit in $|1\rangle$ is equal to $\langle\psi|\hat{n}_1|\psi\rangle$ where

$$\hat{n}_1 \equiv |11\rangle\langle 11| + |10\rangle\langle 10| \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (2.9)$$

Note that $\hat{n}_1$ and $\hat{n}_2$ are not unitary and therefore do not represent operations that can be performed by a quantum computer. Rather, they are convenient notations allowing one to predict the results of an ensemble measurement.

2.2 Quantum algorithms

A quantum algorithm consists of a series of unitary transformations performed on qubits followed by a measurement designed to perform a computation. To this date, the types of quantum algorithms developed generally fall into three categories: Fourier transform, search, and simulation algorithms [15].

The Fourier transform algorithm is the backbone of Shor's factoring and discrete logarithm algorithms, and it involves taking the Fourier transform of a set of numbers: $\{ x_0,\ldots\ x_{2^n-1} \}$ to get a new set: $\{ y_0,\ldots\ y_{2^n-1} \}$. Suppose one prepares a state, $|\psi\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle$, so that the coefficients x_j of the basis states are the numbers one wishes to transform. Then one may define a unitary transformation such that

$$|j\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n} |k\rangle . \tag{2.10}$$

If this transformation is performed on $|\psi\rangle$, we see that the new coefficients are the Fourier transformed set $\{ y_0,\ldots\ y_{2^n-1} \}$, which we wanted.

$$\sum_{j=0}^{2^n-1} x_j |j\rangle \rightarrow \sum_{k=0}^{2^n-1} \left[\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} e^{2\pi ijk/2^n} x_j \right] |k\rangle = \sum_{k=0}^{2^n-1} y_k |k\rangle \tag{2.11}$$

Of course one cannot simply read off the coefficients y_k. If one tried, the wave function would collapse into a random collection of bits. It takes an additional amount of

cleverness and a few more quantum logic gates to get useful information from this transformation. Nevertheless, this algorithm can complete the transform in about n^2 steps as opposed to the classical $n2^n$ steps (for 2^n numbers) [16]—an exponential speedup!

Grover's search algorithm is an example of the second kind of quantum algorithm—the search algorithm. Grover's algorithm is designed to search a space of size *n*, looking for an element in it with some desirable attributes, with no information about the structure of the space. Classically, this problem requires about *n* steps while the quantum algorithm can accomplish it in about $\sqrt{n}$ steps [17].

Finally, simulation algorithms can be used to model physical (typically quantum) systems. Quantum computers are ideal for this task because their Hilbert space increases exponentially with the number of qubits involved. If the system we are attempting to model is quantum mechanical and has *n* components, then in general it takes c^n bits of memory on a classical computer to model it, where *c* is some constant associated with the details of the system. On the other hand, a quantum computer only requires *k n* qubits to model the system, where again *k* is a constant that depends on the system [15]. Though simulation algorithms intended for quantum computers are typically designed to model quantum systems, they can also model classical systems. The Factorized Quantum Lattice Gas Algorithm is an example of a quantum algorithm designed to model a classical system.

2.3 Physical quantum computers

Due to the enormous challenge associated with isolating and precisely controlling single particles, quantum computers are currently incapable of rivaling their classical

counterparts. Quantum computers come in two varieties called type I and type II. Type I machines are 'pure' quantum computers and utilize a number of qubits, each of which can be entangled with any other using an arbitrary unitary transformation. Type II machines are not as powerful but are easier to create in practice. They consist of a number of small type I quantum computers (called nodes) with as few as two qubits in each, connected by classical communications channels carrying bits instead of qubits [18]. Figure 4 shows a simple diagram of a type-II computer.

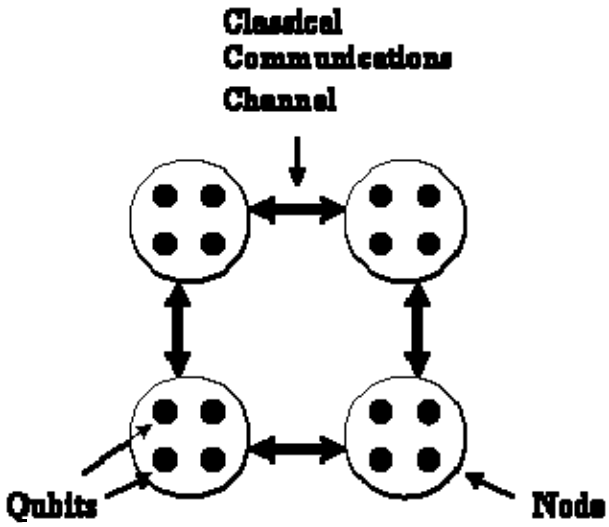

Figure 4. Type-II quantum computer.

The four most developed technologies for quantum computing are optical techniques, ion traps, neutral atom traps, and nuclear magnetic resonance. One of the most significant problems for each of these technologies is decoherence. Decoherence is the uncontrolled entanglement of a system with its environment, destroying the superposition of qubit states within the system and losing the information it contains [19]. The time it takes for this process to occur, called the *decoherence time*, is very short in most systems and therefore limits the number of operations that can be performed on a

given number of qubits. Table 3 lists the various decoherence times (τ_Q) of some of the systems under investigation, along with the time it takes to perform an operation (τ_{op}) on the system [15]. This gives a general idea of the number of operations that can be performed on the system (n_{op}) before quantum information is lost.

Table 3. Estimates for the decoherence time τ_Q, operation time τ_{op}, and maximum number of operations n_{op} for quantum computer candidates [15].

System	τ_Q (sec)	τ_{op} (sec)	n_{op}
Nuclear spin	10^{-2} - 10^{8}	10^{-3} - 10^{-8}	10^{5} - 10^{14}
Electron spin	10^{-3}	10^{-7}	10^{4}
Ion trap (In+)	10^{-1}	10^{-14}	10^{13}
Electron - Au	10^{-8}	10^{-14}	10^{6}
Electron - GaAs	10^{-10}	10^{-13}	10^{3}
Quantum dot	10^{-6}	10^{-9}	10^{3}
Optical cavity	10^{-5}	10^{-14}	10^{9}
Microwave cavity	10^{0}	10^{-4}	10^{4}

Given the long decoherence times of nuclear spins, it is not surprising that the most advanced quantum computers rely on encoding quantum information in atoms with spin half nuclei using NMR technology. This is done by placing a liquid sample in a magnetic field around 10 T, splitting the nuclear spin energy levels with a sort of Zeeman shift. Radio frequency pulses can then be used to manipulate the nuclear states.

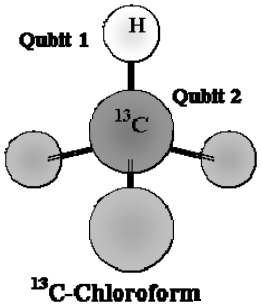

Figure 5. The molecule used by Pravia et al [6] in their implementation of a type II NMR quantum computer was ^{13}C-Chloroform, with hydrogen and carbon 13 nuclear spins serving as the qubits. The energy levels of the nuclear spin states are split using a strong magnetic field.

A basic description of a type-II NMR quantum computer is shown in Figure 5 and Figure 6. The nucleons that act as qubits are in the molecules that make up a liquid sample—in effect each molecule is a small quantum computer. Radio frequency pulses are used to perform unitary transformations on the qubits in the sample. Since the sample contains some 10^{23} identical molecules, using a NMR quantum computer amounts to performing the same calculation on 10^{23} quantum computers. Therefore, when one measures the state of a particular qubit, one does it for the entire sample and gets an average value of the state. This is an example of an ensemble measurement discussed in section 2.2.

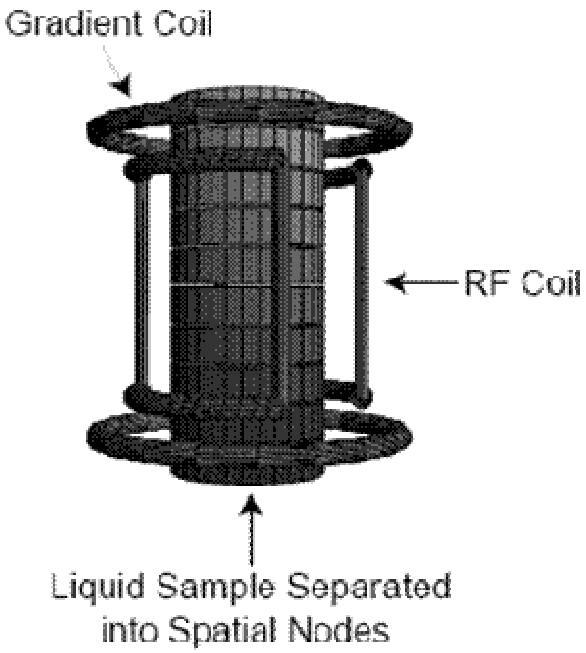

Figure 6. Basic schematic of type-II NMR quantum computer. The gradient coil creates a gradient in the magnetic field so that the nuclear spin energy levels are shifted by different amounts depending on their physical location in the liquid sample. This allows the RF coil to address different parts of the liquid sample with different frequency radio pulses. Each group of molecules that the RF coil can address with one set of frequencies is a node in a type II quantum computer. In each node there are many molecules that are manipulated simultaneously, so that measuring a node is an example of ensemble measurement. This figure was used with permission from [6].

To make a NMR machine a type-II quantum computer, the liquid sample is effectively split into nodes using a magnetic field gradient. This gradient splits the nuclear spin states by different amounts depending on a molecule's location in the liquid sample, allowing one to address different sections of the sample with different frequency radio pulses.

The FQLGA has two properties that make it ideal for implementation on a type II NMR quantum computer. First, it requires no more than three entangled qubits, making it possible to run the algorithm on a type II computer with three qubits per node. NMR machines with three qubits per node have already been successfully demonstrated using Alanine, Trifluorobromoethylene, and Trichloroethylene [15]. Secondly, this algorithm stores information in the probability coefficients of the basis states. This information can be obtained by an ensemble measurement over all the molecules in a node of a NMR machine.

3 Fluid Dynamics

Since the factorized quantum lattice gas algorithm models fluid dynamics, it is worthwhile to briefly review this subject along with the classical lattice gas algorithms that inspired their quantum counterparts. This section starts with a brief overview of fluid dynamics before discussing classical lattice gas and lattice Boltzmann algorithms.

3.1 Navier-Stokes fluids: macroscopic scale

The following section follows Landau and Lifshitz [20], Yepez [4], and Buick [21]. The long wavelength hydrodynamic behavior of a fluid at the macroscopic scale can be modeled by a set of coupled partial differential equations. These equations model

mass density (ρ) and flow velocity ($\vec{u}$) fields, and are called the continuity and Navier-Stokes equations.

Since the fluid mass change in a region $\Re$ comes from the fluid flux through the boundary $\partial\Re$, ρ and $\vec{u}$ must obey

$$\partial_t \rho + \partial_i (\rho u_i) = 0 \tag{3.1}$$

which is the *continuity equation*. Here the shorthand $\partial_t = \partial / \partial t$ and $\partial_i = \partial / \partial x_i$ is used, along with Einstein indicial notation, which implies summation over repeated indices.

The field equation for Newton's second law, which expresses the change in the momentum density in terms of the stress at the boundary of the region $\partial\Re$, is Euler's equation

$$\partial_t (\rho u_i) + \partial_j \Pi_{ij} = 0 \tag{3.2}$$

where the *momentum flux density tensor* can be written

$$\Pi_{ij} = P(\rho, t)\delta_{ij} + \rho u_i u_j - \sigma'_{ij}. \tag{3.3}$$

The first two terms are the *ideal parts* of the momentum flux density tensor, which are the pressure term $P(\rho, t)$ and the convective term $\rho\vec{u}\vec{u}$. The pressure term is diagonal because the fluid is *isotropic*. The last term is the stress tensor, equal to $\sigma'_{ij} = \eta(\partial_i u_j + \partial_j u_i - \frac{2}{D}\partial_k u_k \delta_{ij}) + \zeta\delta_{ij}\partial_k u_k$, where η is the *shear viscosity*, ζ is the *bulk viscosity*, and *D* is the number of spatial dimension of the system.

Substituting (3.3) into Euler's equation gives the *Navier-Stokes equation*

$$\rho\left(\partial_t u_i + u_j \partial_j u_i\right) = -\partial_i P + \rho\nu\partial^2 u_i + \left(\zeta + \frac{\eta}{D}\right)\partial_i \partial_j u_j \tag{3.4}$$

where $\nu \equiv \eta/\rho$ is the *kinematic viscosity*. This equation has known solutions in only a few simple cases, and computer modeling with various numerical techniques are typically necessary to solve this equation for more complex flows.

The kinematic viscosity ν is a measure of the rate of decay of local shears in a fluid, and determines how fast a fluid will relax from an anisotropic to an isotropic flow field. The shear viscosity alone is responsible for the damping of shear waves in the momentum density field, while both the shear and bulk viscosities cause damping of compression waves in the mass density field.

L and *T* are the characteristic length and time scales of a fluid fluctuation. Examples of the characteristic length for a hydrodynamic flow are the wavelength of a compression wave in the mass density field, the wavelength of a shear wave in the momentum density field, or the diameter of a vortex. Examples of characteristic times are the period of a wave, or the rotation period of a vortex. The mean free path (λ) and time (τ) are the average distance and time that microscopic particles in the fluid travel before colliding. Two important speeds are the characteristic flow speed $\upsilon \sim L/T$ and sound speed $c = \lambda/\tau$.

Relevant dimensionless numbers are: the Knudsen number (Kn) defined as the ratio of the mean free path to the characteristic length, the Strouhal number (Sh) defined as the ratio of the mean free time to the characteristic time, Mach number (M) defined as the ratio of the characteristic velocity and sound speed, and Reynolds number (Re) defined as the product of the characteristic velocity and the characteristic length divided by the kinematic viscosity. A list of all these relevant quantities is given in Table 4.

Table 4. List of relevant quantities in fluid dynamics.

Symbol	Name	Description
ρ	mass density field	scalar field that describes the fluid mass density
$\bar{u}$	flow velocity field	vector field that describes the fluid velocity
η	shear viscosity	causes damping of compression waves in mass density field and shear waves in momentum density field
ζ	bulk viscosity	causes damping of compression waves in mass density field
ν	kinematic viscosity	$\equiv \eta / \rho$, determines how fast perturbed fluid will relax
L	characteristic length	length of fluid perturbations
T	characteristic time	period of fluid perturbations
λ	mean free path	particles' average distance between collisions
τ	mean free time	particles' average time between collisions
υ	characteristic speed	$\sim L/T$
c	sound speed	$\sim \lambda / \tau$
Kn	Knudsen number	$\equiv \lambda / L$
Sh	Strouhal number	$\equiv \tau / T$
M	Mach number	$\equiv \upsilon / c$
Re	Reynolds number	$\equiv \upsilon L / \nu \sim M / Kn$

Returning to equation (3.4), one can see that the one dimensional Navier-Stokes equation simplifies to

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \tag{3.5}$$

if $\eta = \zeta = P = 0$. This is a simplified model of turbulence and shock formation called the Burgers equation [22]. In section 4, we will see that the two qubit Factorized Quantum Lattice Gas Algorithm is capable of accurately modeling this equation.

3.2 Classical lattice gas algorithm: microscopic scale

In the 1980's a class of algorithms called Lattice Gas Algorithms (LGA) were discovered to behave like a Navier-Stokes fluid by Wolfram [23] and by Frisch, Hasslacher, and Pomeau [24], raising the possibility of using massively parallel computers running LGAs to simulate fluid dynamics. These simulations may include

attractive interactions between particles to create multiphase fluids [25] or fixed obstacles to simulate vortex shedding [26].

Lattice gas algorithms move virtual particles along an imaginary lattice and change the particles' momentums using collision rules when they meet at a lattice node. The lattice node spacing (ℓ) is defined to be a microscopic scale length so that LGAs are sometimes said to model fluids at this scale. In fact, lattice gas algorithms grossly oversimplify microscopic particle dynamics. However, this turns out not to matter since the macroscopic behavior of a fluid does not depend directly on its microscopic components. This is evident in experiments carried out using wind tunnels and water tanks with low Mach flows and similar Reynolds numbers, since the results of both types of experiments will be similar [21]. Similarly, LGAs in the *continuum limit* (with very small ℓ) turn out to be accurate models of fluid dynamics.

The simulated particles in a lattice gas algorithm are located on the nodes of a regular lattice. The position and momentum of each particle is specified by its position on the lattice and a *displacement vector*. The displacement vector points in the direction that the particle will move at the beginning of a time step. Particles move from one node to another in a process called *streaming*. In the case of a *single speed lattice gas*, all particles move at the same velocity $c = \frac{\ell}{\tau}$, where ℓ is the distance between lattice sites and τ is the time step interval [23]. All of the particles stream simultaneously at the beginning of a time step. Most LGAs enforce an exclusion principle so that no more than one particle can occupy a state at a given time, though there is usually more than one particle at a lattice node. A state is defined as the location and momentum of a particle.

Each state is typically assigned a bit, so that the bit of a full particle state is 1 and 0 for an empty state.

When two or more particles meet at a lattice node, their momentums change in accordance with predetermined *collision rules*. The updated particle trajectories are then streamed at the beginning of the next time step. This process is shown in Figure 7, with particle trajectories at the beginning of a time step labeled by single arrows while the trajectories at the end of the time step are labeled by double arrows.

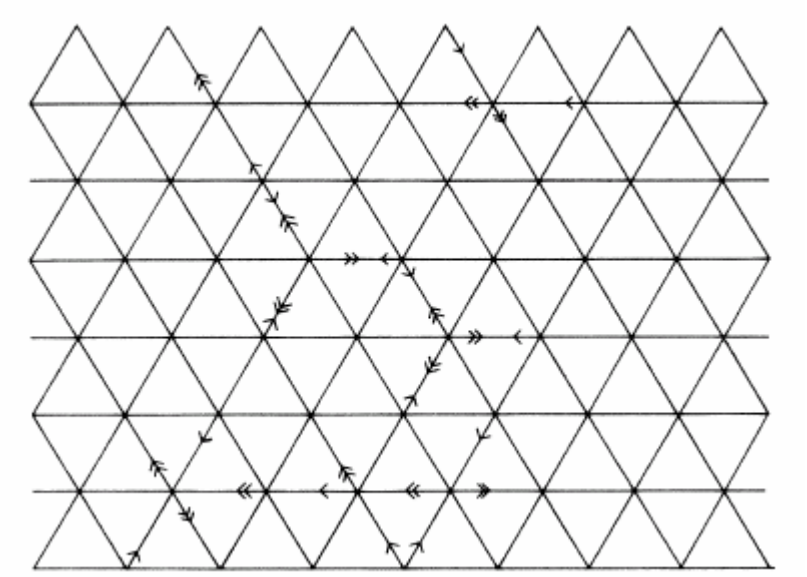

Figure 7. Triangular classical lattice gas developed by Frisch, Hasslascher, and Pomeau. Particles at time t are marked with a single arrow; those at the next time step $t + \tau$ are marked with double arrows. Figure is reproduced from [24].

3.3 Classical lattice Boltzmann algorithm: mesoscopic scale

To transition from the microscopic scale Classical Lattice Gas Models, which contain a number of discrete particles, to the macroscopic scale Navier-Stokes equation, which contains a continuous density parameter ρ, it is necessary to convert the number of particles in a given area to a particle density. In other words, particle number must be replaced by a continuous statistical particle distribution function. This is analogous to

describing the motion of a group of microscopic molecules in a fluid by modeling a mesoscopic statistical particle distribution, called the Boltzmann distribution $f(\vec{x},\vec{u},t)$, that is a function of position, velocity, and time.

Boltzmann mechanics can be described following Gurnett and Bhattacharjee [27], where we consider a group of particles $f(\vec{x},\vec{u},t)d^3xd^3u$ in the phase space volume element d^3xd^3u at time *t*. These particles' positions will change to $\vec{x}' = \vec{x} + \vec{u}\tau$ and their velocities to $\vec{u}' = \vec{u} + \frac{F}{m}\tau$ an instant later at time $t+\tau$, so that they occupy a new volume in phase space: $d^3x'd^3u'$. Or in other words,

$$f(\vec{x}+\vec{u}\tau,\vec{u}+\tfrac{F}{m}\tau,t+\tau)d^3x'd^3u' = f(\vec{x},\vec{u},t)d^3xd^3u\,. \tag{3.6}$$

Any change in the particles distribution that equation (3.6) does not account for must be due to collisions $\Omega(f)$, so that[1]

$$\left[f(\vec{x}+\vec{u}\tau,\vec{u}+\tfrac{F}{m}\tau,t+\tau) - f(\vec{x},\vec{u},t)\right]d^3xd^3u = \Omega(f)d^3xd^3u\tau\,. \tag{3.7}$$

Expanding this result in a Taylor series and taking the limit that τ is zero, one arrives at the well known Boltzmann Equation

$$\frac{\partial f}{\partial t} + \vec{u}\cdot\vec{\nabla}f + \frac{\vec{F}}{m}\cdot\vec{\nabla}_u f = \Omega(f) \tag{3.8}$$

In contrast, if we reconsider the discrete space-time of a lattice gas algorithm and set external forces equal to zero, equation (3.7) becomes the finite difference equation

$$f_u(\vec{x}+\vec{u}\tau,t+\tau) - f_u(\vec{x},t) = \Omega(f_u) \tag{3.9}$$

[1] The Jacobian of the change in the phase space volume $Jd^3xd^3u = d^3x'd^3u'$ is equal to one.

where we now choose to index f by the velocity $\vec{u}$. This is called the *lattice Boltzmann equation*.

There are two methods of modeling this mesoscopic equation. The first approach is to directly simulate the equation on a lattice using continuous values for the particle occupation of a state instead of the binary 1 for "particle present" and 0 for "no particle." Algorithms that follow this approach are called lattice Boltzmann algorithms. The second approach is to model discrete microscopic particles on a lattice gas simulation. The governing lattice Boltzmann equation is then derived as an approximate description of the averaged mesoscopic dynamics [21].

One way to average over the lattice is called *coarse grain averaging* and works by placing a mesoscopic "superlattice" over the microscopic lattice as shown in Figure 8a, and taking the average occupation probability as the value of *f* at a particular superlattice site. The second method takes an ensemble average of many independent microscopic

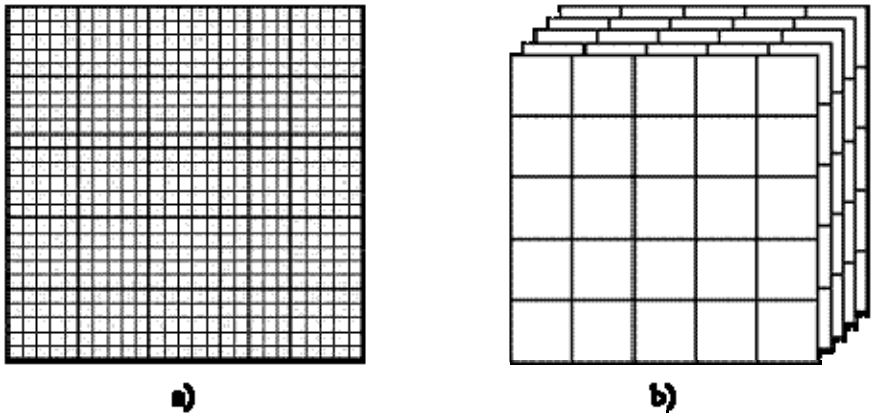

Figure 8. a) Coarse grain averaging works by taking the average over all the microscopic states inside the mesoscopic superlattice. b) Ensemble averaging works by taking the average over many independent microscopic realizations to obtain the particle distribution at each site.

realizations to arrive at a value for *f*. The factorized quantum lattice gas algorithm uses the second method to obtain the distribution *f*.

To transition from a discrete (in space) mesoscopic scale simulation to a continuous macroscopic scale Navier-Stokes simulation one must let the lattice cell size approach zero in the continuum limit as shown in Figure 9. In this limit, it is possible to perform a Chapman-Enskog expansion to derive the macroscopic governing equation of the fluid [24].

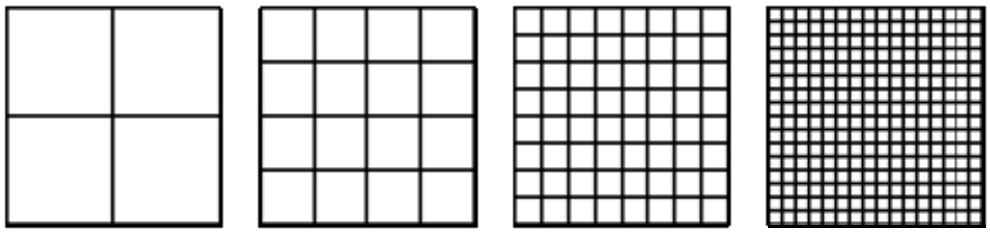

Figure 9. Decreasing the mesoscopic lattice cell size towards zero increases the simulation resolution and, in the continuum limit, will approximate a continuous macroscopic field.

When performing this expansion, one should note that the particles in a lattice gas algorithm undergo *random walk*. That is, a tagged particle will move a distance which asymptotically approached $L = n\ell\sqrt{2/\pi}$ after streaming n^2 times. Since particles stream at the end of every time step, $n^2\tau = T$. This implies that in random walk processes Sh ~ $\mathrm{Kn}^2 \sim (1/n)^2 \equiv \varepsilon^2$ —a condition called *diffusive ordering*, which produces *viscous* hydrodynamic behavior [5, 28]. This will become important as the Chapman-Enskog analysis is described in further detail in the next section.

4 Factorized Quantum Lattice Gas Algorithm

As discussed in section 2.3, type II NMR quantum computers are well suited to simple algorithms that requires massive parallelism. LGAs fit this description well, and it was this observation that lead Yepez [2-5] to develop the *factorized quantum lattice gas algorithm* (FQLGA) to test the modeling utility of quantum computers. The algorithm is called "factorized" because it is not meant to run on a fully coherent computer, but rather on one that is made up of many smaller quantum computers. Constant measurement of the system allows one to transfer classical information between the smaller quantum computers so that the system need not be fully coherent.

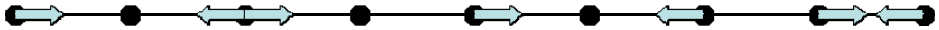

Figure 10. The 1-D factorized quantum lattice gas model developed by Yepez. Each lattice site is simulated by a node on a type II NMR quantum computer. The probability of finding a particle moving right at lattice site *l* is given by qubit 1 in node *l*, and the probability of finding a particle moving left at that lattice site is given by qubit 2. Since there are many computers per node in a type II NMR machine, one can perform an ensemble measurement on each node to obtain the probabilities that will be streamed via classical communications channels to neighboring nodes.

The FQLGA is similar to a classical LGA in that it simulates particles moving along a lattice as shown in Figure 10. As in a classical LGA, the particles move via streaming and obey collision rules when two particles meet at a node. However, unlike a classical LGA, the collision rules are unitary operations which mix those states at a lattice site. Following the collision operation, the updated probabilities for particles moving right (particle 1) and left (particle 2) are obtained via ensemble measurement. These updated probabilities are classically streamed to nearest neighbor lattice sites, marking the end of a time step. Since the collision operator will only mix those states at a given lattice site, each lattice site can be simulated by a node on a type II quantum computer. Thus, the 1-D FQLGA developed by Yepez with two particles per lattice site (moving

right and left) requires a type II quantum computer with two qubits per node. This entire process is described in more detail in the following section.

4.1 The four steps process for the FQLGA

4.1.1 Step 1: computational memory state encoding

The first step in the FQLGA is to encode the computational memory state for the two qubits $|q_1(x_l,t)\rangle$ and $|q_2(x_l,t)\rangle$, representing particles 1 and 2 at the lattice site *l* (located at x_l), at time *t*. At time t_o, the probabilities are given by initial conditions provided by the user. Subsequent probabilities are determined by the algorithm. The probability for particle *m* to exist at a lattice site located at x_l at time *t* is written $p_m(x_l,t)$, so that the m^{th} qubit is encoded as

$$|q_m(x_l,t)\rangle = \sqrt{p_m(x_l,t)}\,|1\rangle + \sqrt{1-p_m(x_l,t)}\,|0\rangle. \tag{4.1}$$

Here the basis state $|1\rangle$ means a particle exists in the simulated state and $|0\rangle$ means it doesn't. The state of the entire node is called the *local state* and is given by the tensor product of the qubits $|q_1(x_l,t)\rangle$ and $|q_2(x_l,t)\rangle$. It has the form

$$\begin{aligned} |\psi(x_l,t)\rangle &= |q_1(x_l,t)\rangle \otimes |q_2(x_l,t)\rangle \\ &= \sqrt{p_1 p_2}\,|11\rangle + \sqrt{p_1(1-p_2)}\,|10\rangle \\ &\quad + \sqrt{(1-p_1)p_2}\,|01\rangle + \sqrt{(1-p_1)(1-p_2)}\,|00\rangle \end{aligned} \tag{4.2}$$

where I have dropped the explicit time and position dependence of $p_m(x_l,t)$ for convenience, and the labels inside the kets are ordered for particles 1 and 2. Note that I

have assumed the qubits are distinguishable since the local ket is neither symmetric nor antisymmetric[1].

4.1.2 Step 2: collision

Following memory state encoding the local state undergoes unitary evolution. This is analogous to the collision operator in the classical LGA and is therefore labeled $\hat{C}$. Thus, the local state becomes

$$\left|\psi'(x_l,t)\right\rangle = \hat{C}\left|\psi(x_l,t)\right\rangle. \tag{4.3}$$

If we use the basis in (2.2) and choose a collision operator that conserves particle number then the operator will be block diagonal and have the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\sigma}e^{i\xi}\cos\theta & e^{i\sigma}e^{i\phi}\sin\theta & 0 \\ 0 & -e^{i\sigma}e^{-i\phi}\sin\theta & e^{i\sigma}e^{-i\xi}\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.4}$$

The block that mixes the states $\left|10\right\rangle$ and $\left|01\right\rangle$ is a U(2) matrix.

4.1.3 Step 3: measurement

This step destroys the quantum superposition and measures the probability of each qubit to be in the state $\left|1\right\rangle$. As discussed in section 2.1.2, this probability can be obtained via an ensemble measurement, and for convenience may be expressed as

$$p'_m(x_l,t) = \left\langle\psi'(x_l,t)\right|\hat{n}_m\left|\psi'(x_l,t)\right\rangle \tag{4.5}$$

[1] This is reasonable since one designs a quantum computer so that its qubits are distinguishable. For instance, the nucleons chosen to represent qubits have different spin energy levels in a NMR computer.

where the operators $\hat{n}_m$ are given in (2.8) and (2.9). Notice that the equation

$$p_m(x_l,t) = \langle \psi(x_l,t) | \hat{n}_m | \psi(x_l,t) \rangle \tag{4.6}$$

also holds. This is of no consequence now but will be useful later.

4.1.4 Step 4: streaming

Each lattice site is updated following steps 1 through 3 and the resulting probabilities are streamed to the adjacent lattice sites via classical communications channels so that

$$p_m(x_l + e_m \ell,\ t+\tau) = p'_m(x_l,\ t) \tag{4.7}$$

where e_1 = 1 for the right streaming qubit, e_2 = -1 for the left streaming qubit, and ℓ is the lattice spacing. This signals the end of a time step τ, after which the entire process is repeated. The simulation will include many time steps and is typically completed when the system reaches equilibrium and exhibits no further change.

4.2 Quantum lattice Boltzmann equation

By simple substitution, all four steps in the FQLGA can be encapsulated in one equation. This is carried out as follows:

$$\begin{aligned} p_m(x_l + e_m \ell,\ t+\tau) &= p'_m(x_l,\ t) \\ &= \langle \psi'(x_l,\ t) | \hat{n}_m | \psi'(x_l,\ t) \rangle \\ &= \langle \psi(x_l,\ t) | \hat{C}^\dagger \hat{n}_m \hat{C} | \psi(x_l,\ t) \rangle . \end{aligned} \tag{4.8}$$

With a few modifications, this equation becomes the finite difference quantum lattice Boltzmann equation, analogous to the classical equation (3.9). The first step is to

reinterpret the probabilities $p_m(x_l,\ t)$ as a mesoscopic Boltzmann field $p_m(x_l,\ t) \equiv f_m(x_l,t)$. Therefore, equation (4.6) can be rewritten

$$f_m(x_l,t) = \langle \psi(x_l,t) | \hat{n}_m | \psi(x_l,t) \rangle \tag{4.9}$$

and the result of (4.8) can be rewritten

$$f_m(x_l + e_m\ell,\ t+\tau) = \langle \psi(x_l,\ t) | \hat{C}^\dagger \hat{n}_m \hat{C} | \psi(x_l,\ t) \rangle . \tag{4.10}$$

Then all that is left to do is to subtract equation (4.9) from (4.10) so we obtain

$$\begin{aligned} f_m(x_l + e_m\ell,\ t+\tau) - f_m(x_l,\ t) &= \langle \psi(x_l,\ t) | \left(\hat{C}^\dagger \hat{n}_m \hat{C} - \hat{n}_m \right) | \psi(x_l,\ t) \rangle \\ &= \Omega_m(\psi) \end{aligned} \tag{4.11}$$

This is the *quantum lattice Boltzmann equation*. This can be further expanded by inserting the vectors for $\langle \psi(x_l,\ t)|$ and $|\psi(x_l,\ t)\rangle$ along with the matrices for $\hat{C}$ and $\hat{n}_m$ into (4.11). With much algebraic manipulation [12], the collision function becomes

$$\Omega_{1,2} = \mp \left(\sin^2\theta \left[(1-f_2) f_1 - (1-f_1) f_2 \right] + \sin(2\theta)\cos(\phi - \xi) \sqrt{f_1(1-f_1) f_2(1-f_2)} \right) \tag{4.12}$$

or, written more simply

$$\Omega_{1,2} = \mp \left(\sin^2\theta (f_1 - f_2) + \sin(2\theta)\cos(\phi - \xi) \sqrt{f_1(1-f_1) f_2(1-f_2)} \right) . \tag{4.13}$$

4.3 Chapman-Enskog expansion

As mentioned in section 3.3, one can derive the macroscopic governing equation in the continuum limit of the lattice by performing a Chapman-Enskog expansion of the lattice Boltzmann equation. This section will follow Yepez [4, 10] to derive the macroscopic equation for his model.

The Chapman-Enskog expansion works by taking a Taylor series expansion of the lattice Boltzmann equation around local equilibrium. In physical systems, local equilibrium is the state where particles are in thermodynamic equilibrium with one another across mesoscopic or microscopic scale lengths. In classical lattice Boltzmann and quantum lattice gas algorithms, local equilibrium is obtained at a lattice site when the collision function no longer changes the particle distribution at that site.

We expand around local equilibrium because at the mesoscopic scale, most systems are at, or very near to, thermodynamic equilibrium. It is only at the macroscopic scale that there are free thermodynamic variables such as local density, temperature, and momentum. Thus, a macroscopic description of a fluid comes from a patchwork of slowly varying systems at or very near equilibrium [24].

The dimensionless numbers Kn (Knudsen number), Sh (Strouhal number), and Re (Reynolds number) discussed in section 3.1 can be used to determine how close a system is to equilibrium—at equilibrium these numbers vanish. For instance, at equilibrium the characteristic length scale is infinitely large compared to mean free path, so Kn ~ 0. However, hydrodynamic behavior is also attained in the long wavelength limit where these numbers are close to zero. Thus, it should be of no surprise that expanding the lattice Boltzmann equation around local equilibrium should result in hydrodynamic behavior.

In the following section the local equilibrium value of f_m is derived. In section 4.3.2 this result is used in the Chapman-Enskog expansion of the quantum lattice Boltzmann equation.

4.3.1 Local equilibrium

For simplicity, we will label the equilibrium values of f_m as d_m. Local equilibrium is defined as the condition where $d_m(x_l + e_m\ell,\ t+\tau) - d_m(x_l,\ t) = 0$; that is, when collisions cause no further change in the distribution function f_m. From this we can see that $\Omega_m\,|_{f_{1,2}=d_{1,2}} = 0$, or from (4.12)

$$\sin^2\theta\left[(1-d_2)d_1 - (1-d_1)d_2\right] + \sin(2\theta)\cos(\phi-\xi)\sqrt{d_1(1-d_1)d_2(1-d_2)} = 0 \qquad (4.14)$$

Dividing this equation by $(1-d_1)(1-d_2)$ and rearranging we get

$$\frac{d_1}{(1-d_2)} - \frac{d_2}{(1-d_1)} = 2\cot(\theta)\cos(\phi-\xi)\sqrt{\frac{d_1}{(1-d_1)}\frac{d_2}{(1-d_2)}}\,. \qquad (4.15)$$

Taking the equilibrium probabilities to have the form

$$d_1 = \frac{1}{\gamma z + 1} \qquad \text{and} \qquad d_2 = \frac{1}{\frac{z}{\gamma}+1}\,. \qquad (4.16)$$

and substituting this into (4.15) with some manipulation gives the quadratic equations

$$\gamma^2 + 2\alpha\gamma - 1 = 0 \qquad (4.17)$$

where $\alpha \equiv \cot\theta\cos(\phi-\xi)$. We take the positive root solution of this (so that d_m will be positive) so that

$$\begin{aligned} \gamma &= \sqrt{\alpha^2+1} + \alpha \\ \frac{1}{\gamma} &= \sqrt{\alpha^2+1} - \alpha. \end{aligned} \qquad (4.18)$$

Noting that the total number density at a lattice site is $d_1 + d_2 \equiv \rho$ and substituting (4.16) and (4.17) into this expression, we obtain a quadratic equation in z

$$\rho z^2 + \left(\gamma + \frac{1}{\gamma}\right)(\rho-1)z + (\rho-2) = 0\,. \qquad (4.19)$$

When the positive root solution of (4.19) along with (4.17) is substituted into (4.16) one finally arrives at

$$d_{1,2} = \frac{\rho}{2} \mp \frac{1}{2\alpha}\left(\sqrt{1+\alpha^2} - \sqrt{1+\alpha^2(\rho-1)^2}\right) \tag{4.20}$$

which are the equilibrium values for the two qubit quantum lattice Boltzmann equation.

4.3.2 Taylor series expansion around local equilibrium

To keep track of the order of the expansion, one uses a "smallness parameter" ε. This is defined to be on the order of the Knudsen number: Kn = $\ell / L \sim \varepsilon$. Like the classical lattice gas algorithm, the factorized quantum lattice gas algorithm obeys diffusive ordering. Therefore, ε^2 must be on the order of the Strouhal number Sh = $\frac{\tau}{T} \sim \varepsilon^2$ [4]. The QLBE can thus be written

$$f_m(x_l + \varepsilon e_m \ell,\ t + \varepsilon^2 \tau) - f_m(x_l,\ t) = \Omega_m . \tag{4.21}$$

With this we can now find the Taylor series of the left side of (4.21), which is just the quantum lattice Boltzmann equation rewritten to include ε. This gives

$$\varepsilon^2 \tau \frac{\partial f_m}{\partial t} + \varepsilon e_m \ell \frac{\partial f_m}{\partial x} + \varepsilon^2 e_m^2 \ell^2 \frac{1}{2} \frac{\partial^2 f_m}{\partial x^2} + \mathrm{O}(\varepsilon^3) = \Omega_m . \tag{4.22}$$

We did not expand the right hand side of (4.21) because it turns out to be easier in the end not to. Notice that the equation is still exact, since we implicitly keep higher order terms in the factor $\mathrm{O}(\varepsilon^3)$.

The crucial ansatz is to now assume that f_m can be expanded around local equilibrium as the sum of d_m and a small deviation δf_m, which may in turn be expanded in powers of ε:

$$f_m = d_m + \varepsilon\delta f_m^{(0)} + \varepsilon^2\delta f_m^{(1)} + \mathrm{O}(\varepsilon^3). \tag{4.23}$$

Inserting this into (4.22) and explicitly writing out only those terms up to second order in ε we obtain

$$\varepsilon^2\tau\frac{\partial d_m}{\partial t} + \varepsilon e_m \ell \frac{\partial d_m}{\partial x} + \varepsilon^2 e_m \ell \frac{\partial \delta f_m^{(0)}}{\partial x} + \varepsilon^2 e_m^2 \ell^2 \frac{1}{2}\frac{\partial^2 d_m}{\partial x^2} + \mathrm{O}(\varepsilon^3) = \Omega_m. \tag{4.24}$$

This is the QLBE expanded around equilibrium. However, we do not have an expression for $\delta f_m^{(0)}$. To obtain one, we take the first moment of the QLBE and solve for $\delta f_m^{(0)}$. First, expand

$$\Omega_m = \Omega_m\big|_{f_{1,2}=d_{1,2}} + \varepsilon\sum_n \left.\frac{\partial\Omega_m}{\partial f_n}\right|_{f_{1,2}=d_{1,2}} \delta f_n^{(0)} + \mathrm{O}(\varepsilon^2). \tag{4.25}$$

Note that the first term in this expression is equal to zero from the definition of local equilibrium. Equating like powers of ε, the first moment of (4.21) is therefore

$$\varepsilon e_m \ell \frac{\partial d_m}{\partial x} = \varepsilon\sum_n \left.\frac{\partial\Omega_m}{\partial f_n}\right|_{f_{1,2}=d_{1,2}} \delta f_n^{(0)}. \tag{4.26}$$

From equation (4.13) we see that $\Omega_1 = -\Omega_2 \equiv \Omega$. Then, for simplicity, equation (4.26) can be rewritten in vector form

$$\varepsilon\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle = \varepsilon\hat{J}\left|\delta f^{(0)}\right\rangle \tag{4.27}$$

where the vectors $\left|\frac{\partial d}{\partial x}\right\rangle$ and $\left|\delta f^{(0)}\right\rangle$ are

$$\left|\frac{\partial d_m}{\partial x}\right\rangle = \begin{pmatrix} \frac{\partial d_1}{\partial x} \\ \frac{\partial d_2}{\partial x} \end{pmatrix} \qquad \text{and} \qquad \left|\delta f^{(0)}\right\rangle = \begin{pmatrix} \delta f_1^{(0)} \\ \delta f_2^{(0)} \end{pmatrix}. \tag{4.28}$$

and the matrices $\hat{J}$ (the Jacobian) and $\hat{e}$ are

$$\hat{J} = \left.\begin{pmatrix} \frac{\partial \Omega}{\partial f_1} & \frac{\partial \Omega}{\partial f_2} \\ \frac{-\partial \Omega}{\partial f_1} & \frac{-\partial \Omega}{\partial f_2} \end{pmatrix}\right|_{f_{1,2}=d_{1,2}} \equiv \begin{pmatrix} J_1 & J_2 \\ -J_1 & -J_2 \end{pmatrix} \qquad \text{and} \qquad \hat{e} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{4.29}$$

Solving for $\left|\delta f^{(0)}\right\rangle$ is not as easy as finding the inverse of $\hat{J}$, however, because this matrix is singular. Yepez utilizes two (equivalent) methods to find a consistent $\left|\delta f^{(0)}\right\rangle$ [10]. The first is to multiply both sides of (4.27) by $\hat{J}$. The second method is to multiply both sides of (4.27) by a "generalized inverse" $\hat{J}^{-1}_{gen}$, which Yepez has invented. This matrix is similar to the Moore-Penrose pseudoinverse [29].

The eigenvalues and left and right eigenvectors of $\hat{J}$ are

$$\begin{aligned} &\lambda_1 = 0 \qquad \langle E_1| = \begin{pmatrix} 1 & 1 \end{pmatrix} \qquad |\mathrm{E}_1\rangle = \frac{1}{J_1 - J_2}\begin{pmatrix} J_2 \\ -J_1 \end{pmatrix} \\ &\lambda_2 = J_1 - J_2 \qquad \langle E_2| = \frac{1}{J_2 - J_1}\begin{pmatrix} J_1 & J_2 \end{pmatrix} \qquad |\mathrm{E}_2\rangle = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{aligned} \tag{4.30}$$

where the right eigenvectors (often simply called eigenvectors) satisfy $\hat{J}|E_m\rangle = \lambda_m |E_m\rangle$, the left eigenvectors satisfy $\langle E_m|\hat{J} = \lambda_m \langle E_m|$, and the eigenvector lengths are selected so that $\langle E_m | E_n\rangle = \delta_{mn}$. From this we see that $\hat{J}$ is

$$\hat{J} = \lambda_1 |E_1\rangle\langle E_1| + \lambda_2 |E_2\rangle\langle E_2| = \lambda_2 |E_2\rangle\langle E_2| = \begin{pmatrix} J_1 & J_2 \\ -J_1 & -J_2 \end{pmatrix} \tag{4.31}$$

since $\lambda_1 = 0$. This equation is analogous the spectral decomposition of a Hermitian matrix, and is equivalent to square matrix diagonalization. Any *n* by *n* square matrix **M** with *n* independent eigenvectors can be diagonalized using $\mathbf{S}^{-1}\mathbf{M}\mathbf{S} = \mathbf{\Lambda}$ [30]. The columns of **S** are the right eigenvectors of **M**, $\mathbf{\Lambda}$ is a diagonal matrix with corresponding

eigenvalues, and the rows of $\mathbf{S}^{-1}$ are the left eigenvectors of $\mathbf{M}$ since $\langle E_m | E_n \rangle = \delta_{mn}$.

Therefore $\mathbf{M} = \mathbf{S}\Lambda\mathbf{S}^{-1}$ is equivalent to (4.31).

From (4.31), it is clear that

$$\hat{J}^2 = \lambda_2^2 |E_2\rangle\langle E_2| = \lambda_2 \hat{J}. \tag{4.32}$$

As was mentioned, the first method Yepez uses to find $|\delta f^{(0)}\rangle$ is to multiply both sides of (4.27) by $\hat{J}$. Then one obtains

$$\begin{aligned} \hat{J}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \hat{J}^2 |\delta f^{(0)}\rangle \\ \hat{J}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \hat{J}\lambda_2 |\delta f^{(0)}\rangle \\ &\text{which implies} \\ \ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \lambda_2 |\delta f^{(0)}\rangle \\ \frac{1}{\lambda_2}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= |\delta f^{(0)}\rangle \ . \end{aligned} \tag{4.33}$$

The second (and equivalent) method Yepez uses is to multiply both sides of (4.27) by his generalized inverse. The generalized inverse is analogous to the inverse of a nonsingular square matrix $\mathbf{M}^{-1} = \mathbf{S}\Lambda^{-1}\mathbf{S}^{-1}$. Yepez uses an identical construction for his generalized inverse except that he replaces Λ^{-1} with a matrix, Λ_{gen}^{-1}, in which only the nonzero diagonal components are inverted. The procedure for constructing the Moore-Penrose pseudoinverse is similar, except the vectors which make up $\mathbf{S}^{-1}$ and $\mathbf{S}$ are the left and right eigenvectors of $\mathbf{M}\mathbf{M}^{\dagger}$. It can be shown that when $\mathbf{M}$ is invertible, the least squares solution for $\mathbf{M}\mathbf{x} = \mathbf{b}$ is $\mathbf{x} = \mathbf{M}_{psuedo}^{-1}\mathbf{b}$, where $\mathbf{M}_{psuedo}^{-1}$ is the Moore-Penrose pseudoinverse [30].

Yepez's generalized inverse for $\hat{J}$ is

$$\hat{J}_{gen}^{-1} = \frac{1}{\lambda_2}\left|E_2\right\rangle\left\langle E_2\right| = \frac{1}{\lambda_2^2}\left(\lambda_2\left|E_2\right\rangle\left\langle E_2\right|\right) = \frac{1}{\lambda_2^2}\hat{J}\,. \tag{4.34}$$

Multiplying both sides of (4.27) from the left by this generalized inverse, one obtains

$$\begin{aligned}
\hat{J}_{gen}^{-1}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \hat{J}_{gen}^{-1}\hat{J}\left|\delta f^{(0)}\right\rangle \\
\frac{1}{\lambda_2^2}\hat{J}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \frac{1}{\lambda_2^2}\hat{J}^2\left|\delta f^{(0)}\right\rangle \\
\hat{J}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \hat{J}^2\left|\delta f^{(0)}\right\rangle
\end{aligned} \tag{4.35}$$

which may be further simplified following the same steps given in (4.33). Thus, the two methods are equivalent. If one takes the solution obtained in (4.33) and substitutes it into (4.24) one arrives at

$$\varepsilon^2\tau\frac{\partial d_m}{\partial t} + \varepsilon e_m\ell\frac{\partial d_m}{\partial x} + \varepsilon^2\frac{1}{\lambda_2^2}e_m^2\ell^2\frac{\partial\lambda_2}{\partial x}\frac{\partial d_m}{\partial x} + \left(\frac{1}{\lambda_2}+\frac{1}{2}\right)\varepsilon^2 e_m^2\ell^2\frac{\partial^2 d_m}{\partial x^2} + \mathrm{O}(\varepsilon^3) = \Omega_m \tag{4.36}$$

The next step is to sum these equations over *m*, noting that $d_1 + d_2 \equiv \rho$, and $e_m^2 = 1$. This gives

$$\varepsilon^2\tau\frac{\partial\rho}{\partial t} + \varepsilon\ell\frac{\partial}{\partial x}(d_1 - d_2) + \varepsilon^2\frac{\ell^2}{\lambda_2^2}\frac{\partial\lambda_2}{\partial x}\frac{\partial\rho}{\partial x} + \left(\frac{1}{\lambda_2}+\frac{1}{2}\right)\varepsilon^2\ell^2\frac{\partial^2\rho}{\partial x^2} + \mathrm{O}(\varepsilon^3) = 0\,. \tag{4.37}$$

In what follows, Yepez restricts himself to small α to simplify the resulting equations. That is, he assumes that the angles θ, ϕ, and ξ in the collision operator are such that $|\alpha| = |\cot\theta\cos(\phi - \xi)| << 1$. Then, using the equilibrium values (4.20), the second term in (4.37) is

$$\varepsilon\ell\frac{\partial}{\partial x}(d_1-d_2)=\varepsilon\ell\frac{\partial}{\partial x}\left[-\frac{1}{\alpha}\left(\sqrt{1+\alpha^2}-\sqrt{1+\alpha^2(\rho-1)^2}\right)\right]$$
$$=-\varepsilon\ell\alpha(\rho-1)\frac{\partial\rho}{\partial x}\left(1+\alpha^2(\rho-1)^2\right)^{-1/2} \tag{4.38}$$
$$=\varepsilon\ell\alpha(1-\rho)\frac{\partial\rho}{\partial x}\left(1+\mathrm{O}(\alpha^2)\right)$$

where the Taylor series expansion of $\left(1+\alpha^2(\rho-1)^2\right)^{-1/2}$ with respect to α was taken in the last line. Calculating the components of *J* one obtains

$$J_{1,2}=\sin^2\theta\left(\mp1-\alpha\frac{(2d_{1,2}-1)d_{2,1}(1-d_{2,1})}{\sqrt{d_1(1-d_1)d_2(1-d_2)}}\right). \tag{4.39}$$

This implies that

$$\lambda_2=J_1-J_2=-2\sin^2\theta(1+\alpha^2 f(\alpha,\rho)) \tag{4.40}$$

where $f(\alpha,\rho)$ is very complicated but has the important property that $f(\alpha,\rho)=1+\mathrm{O}(\alpha)$. The resulting equation is

$$\frac{\partial\rho}{\partial t}+\frac{\ell}{\tau}\cot\theta\cos(\phi-\xi)(1-\rho)\frac{\partial\rho}{\partial x}=\cot^2\theta\frac{\ell^2}{\tau}\frac{\partial^2\rho}{\partial x^2}+\mathrm{O}(\varepsilon^3,\varepsilon\alpha^2) \tag{4.41}$$

Dropping the terms implicit in $\mathrm{O}(\varepsilon^3,\varepsilon\alpha^2)$ one obtains a partial differential equation that models the FQLGA in the continuum limit of the quantum lattice Boltzmann equation (4.11), accurate to first order in time and second order in space for small α.

For $u=c_s(1-\rho)$ where $c_s=\frac{\ell}{\tau}\cot\theta\cos(\phi-\xi)$ is the speed of sound and $\nu=\frac{1}{2}\cot^2\theta\frac{\ell^2}{\tau}$ is the kinematic viscosity, equation (4.41) becomes the Burger's Equation introduced in section 3.1.

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=\nu\frac{\partial^2 u}{\partial x^2} \tag{4.42}$$

Thus we have completed the Chapman-Enskog expansion by taking the Taylor series expansion of the quantum lattice Boltzmann equation around local equilibrium, valid when $\left|\cot\theta\cos(\phi-\xi)\right| << 1$. This entire process can be a bit difficult to follow so it is summarized here:

1. Expand f_m around local equilibrium: $f_m = d_m + \varepsilon\delta f_m^{(0)} + ...$
2. Insert the expanded f_m into a Taylor series expansion of the QLBE, explicitly writing out only those terms of order ε^2 or lower (since the first derivative in time is on the order of ε^2).
3. Use the first moment of the QLBE to solve for the unknown $\delta f_m^{(0)}$ in terms of d_m. Placing the equations in matrix form can help, but it is nevertheless tricky since the matrix *J* will be singular.
4. Insert $\delta f_m^{(0)}$ into the results of step 2.
5. Sum the results of step 5 over *m* and simplify, taking advantage of the fact that $\rho = \sum_m d_m$.

The resulting equation will be the governing partial differential of the quantum lattice Boltzmann equation in the continuum limit.

4.4 Numerical and experimental simulation of the Burgers equation

Yepez has run a numeric simulation of this algorithm with a lattice length equal to 256ℓ on a conventional desktop computer, and compared its results to a known analytic solution of the Burgers equation [10]. The initial condition of the simulation was a

sinusoidal wave, which generates a shock front at later times. For the simulation he chose $\tau = 1$, $\ell = 1$, $\theta = \pi/4$, and $\phi = \xi$.

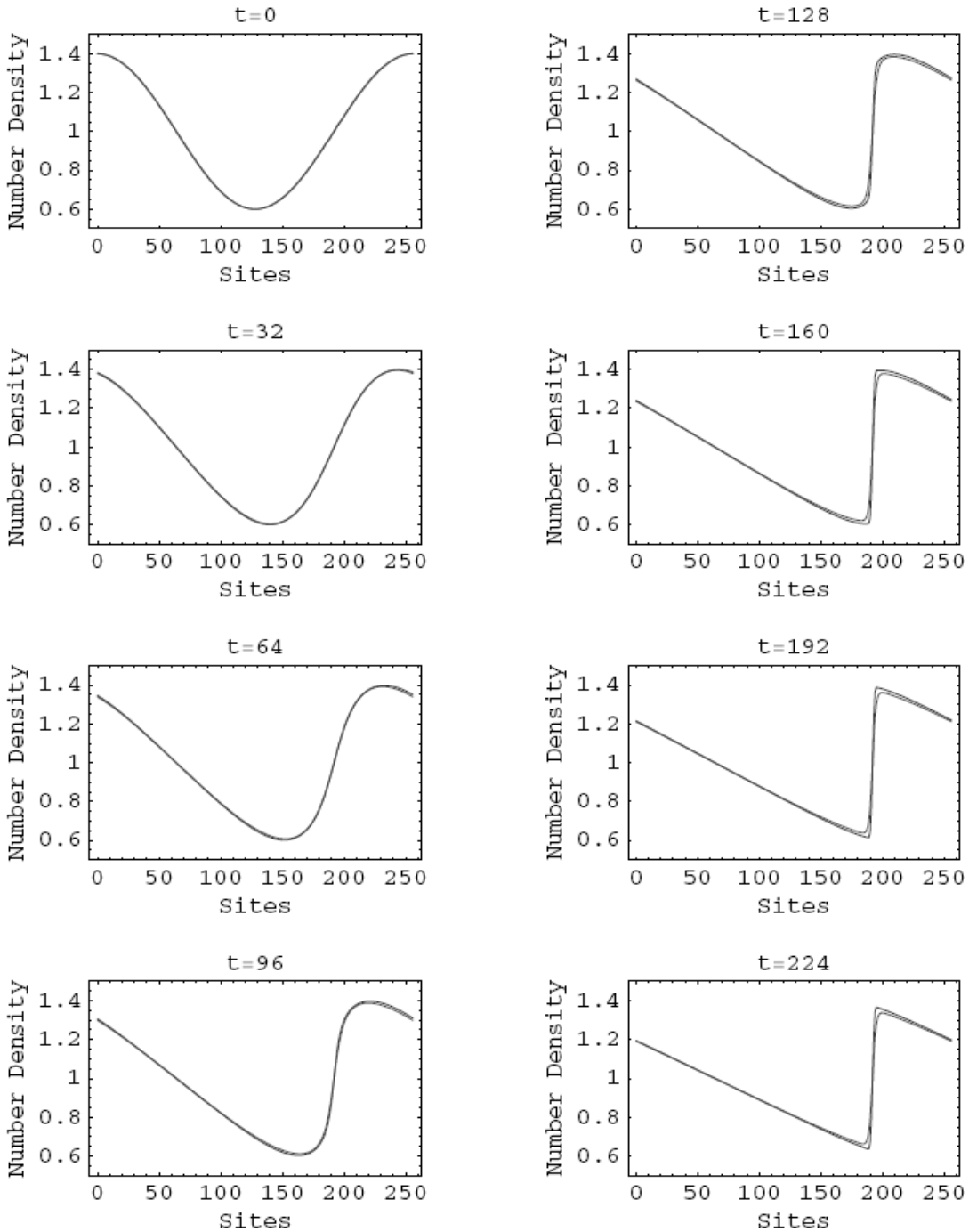

Figure 11. Numerical results of the Burgers equation simulation carried out by Yepez, along with the analytic solution. Agreement between the simulation and the analytic solution is generally very good, with slight deviations occurring at later times as a steep shock front is formed. The simulation shock front is sharper than the analytic shock. This figure was used with permission from [10].

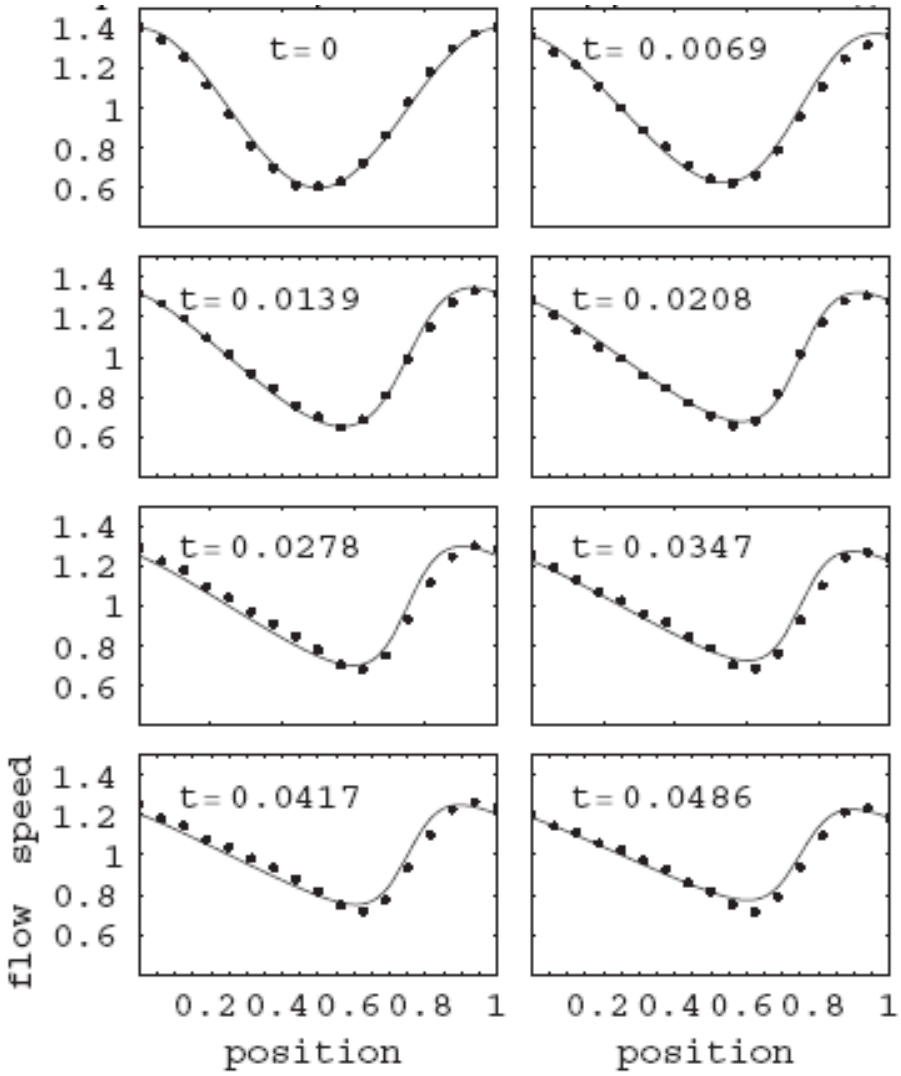

Figure 12. Experimental results of the two qubit FQLGA simulating the Burgers equation carried out on a type II NMR quantum computer. The black dots are the experimental results and the solid gray line is the analytic solution. This figure was used with permission from [11].

The results of this simulation are presented in Figure 11. The agreement between the simulation and analytic solution is generally very good, although there is some divergence at later times as the steep shock front forms. The simulation produces a sharper edged shock than the curved edge analytic solution. The agreement is nevertheless impressive since the shock front appears to be greatly under-resolved. Comparable classical algorithms used to model the Burgers equation require significantly

more lattice sites, $2^{16}\ell = 65536\ell$, and time steps, $2^{18}\tau = 262144\tau$, to model a shock formation with this accuracy.

This simulation has also been run on a working two qubit per node type II NMR quantum computer with 16 nodes. The results of this are shown in Figure 12. The dominant errors in this simulation come from errors in applying the collision operator, which accumulate over time.

5 Three Qubit FQLGA Using Most General Collision Matrix

The two qubit FQLGA developed by Yepez can be extended in one dimension by adding on an additional qubit per lattice site that does not move during streaming. Thus, this sort of algorithm will have three particles per lattice site: particle one streams right, particle two does not move, and particle three streams left. The difficulty in extending this algorithm, however, is that each additional qubit greatly increases the complexity of the most general collision operator that conserves particle number. This makes the analytic treatment increasingly difficult to carry out. Nevertheless, I have derived the quantum lattice Boltzmann equation for the most general three qubit collision operator that conserves particle number, as well as the diffusion equation for a specific collision matrix.

In what follows, I will discuss the analytic treatment of the most general three qubit operator before deriving the diffusion equation in full detail. Then I will show the results of numeric simulations of the three qubit FQLGA using the collision matrix which models the diffusion equation. I will compare this simulation to the analytic solution and investigate the convergence and numerical stability of the simulation.

5.1 Microscopic scale: matrices and basis states

Like the two qubit algorithm developed by Yepez, my three qubit algorithm is designed to conserve particle number. For this reason, there may only be mixing between the states $|110\rangle$, $|101\rangle$, and $|011\rangle$, and separately $|001\rangle$, $|010\rangle$, and $|100\rangle$. I therefore choose the basis given in (2.5), so that

$$\begin{pmatrix} \langle 111|\psi\rangle \\ \langle 110|\psi\rangle \\ \langle 101|\psi\rangle \\ \langle 011|\psi\rangle \\ \langle 001|\psi\rangle \\ \langle 010|\psi\rangle \\ \langle 100|\psi\rangle \\ \langle 000|\psi\rangle \end{pmatrix} = \begin{pmatrix} \sqrt{p_1 p_2 p_3} \\ \sqrt{p_1 p_2 (1-p_3)} \\ \sqrt{p_1 (1-p_2) p_3} \\ \sqrt{(1-p_1) p_2 p_3} \\ \sqrt{(1-p_1)(1-p_2) p_3} \\ \sqrt{(1-p_1) p_2 (1-p_3)} \\ \sqrt{p_1 (1-p_2)(1-p_3)} \\ \sqrt{(1-p_1)(1-p_2)(1-p_3)} \end{pmatrix} \tag{5.1}$$

where p_m is of course the probability of finding a particle in state *m*. This choice makes the collision matrix block diagonal. Thus it must have the form

$$\hat{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & U(3) & 0 & 0 \\ 0 & 0 & U(3) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.2}$$

This is shorthand, since $\hat{C}$ must be an eight by eight matrix. The entries U(3) represent the most general three by three unitary matrices. In fact, this collision matrix can be simplified even further since $U(3) = e^{i\sigma} SU(3)$. From equation (4.10), we see that the system dynamics are determined by the matrix $\hat{C}^{\dagger}\hat{n}_m\hat{C}$, where $\hat{n}_m$ gives the probability of finding a particle in state *m* and represents

$$\hat{n}_1 \to \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix}, \hat{n} \to \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix},$$

$$\hat{n}_3 \to \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix} \tag{5.3}$$

Since $\hat{n}_m$ is diagonal we know that $\hat{C}^\dagger \hat{n}_m \hat{C}$ must have the form

$$\hat{C}^\dagger \hat{n}_m \hat{C} = \begin{pmatrix} 1 & 0 & 0 & 0\\ 0 & SU(3)^\dagger e^{-i\sigma} \cdot \hat{D}_{m,1} \cdot e^{i\sigma} SU(3) & 0 & 0\\ 0 & 0 & SU(3)^\dagger e^{-i\sigma} \cdot \hat{D}_{m,2} \cdot e^{i\sigma} SU(3) & 0\\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.4}$$

where $\hat{D}_{m,n}$ are diagonal matrices that depend on $\hat{n}_m$. For instance, for $m = 1$

$$\hat{D}_{1,1} = \begin{pmatrix} 1&0&0\\ 0&1&0\\ 0&0&0 \end{pmatrix} \text{ and } \hat{D}_{1,2} = \begin{pmatrix} 0&0&0\\ 0&0&0\\ 0&0&1 \end{pmatrix} \tag{5.5}$$

Obviously, the terms $e^{-i\sigma}$ can be factored through the $\hat{D}_{m,n}$ matrices and cancel the terms $e^{i\sigma}$, so without any loss of generality one can use the simplified collision matrix

$$\hat{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & SU(3) & 0 & 0 \\ 0 & 0 & SU(3) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.6}$$

The matrix SU(3) has eight free parameters and is too complicated to write down here but is presented in Appendix A.

5.2 Mesoscopic scale: quantum lattice Boltzmann equation

The quantum lattice Boltzmann equation

$$f_m(x_l + e_m\ell,\ t+\tau) - f_m(x_l,\ t) = \langle \psi(x_l,\ t) | \left(\hat{C}^\dagger \hat{n}_m \hat{C} - \hat{n}_m \right) | \psi(x_l,\ t) \rangle \tag{5.7}$$

developed in 4.2 is still valid for my three qubit algorithm. Due to the complexity of the most general SU(3) matrix, it is useful to temporarily replace it with the over-parameterized matrix

$$\text{SU(3)} \rightarrow \begin{pmatrix} ae^{i\theta_a} & be^{i\theta_b} & ce^{i\theta_c} \\ de^{i\theta_d} & fe^{i\theta_f} & ge^{i\theta_g} \\ he^{i\theta_h} & je^{i\theta_j} & ke^{i\theta_k} \end{pmatrix} \tag{5.8}$$

where I have written the entries of the SU(3) matrix as complex numbers in polar form, parameterized by the real numbers $a-d, f-h, j, k$ and $\theta_{a-d, f-h, j, k}$. Since the rows and columns of a unitary matrix are orthonormal, the constraints

1) $a^2 + b^2 + c^2 = 1$
2) $d^2 + f^2 + g^2 = 1$
3) $h^2 + j^2 + k^2 = 1$
4) $a^2 + d^2 + h^2 = 1$
5) $b^2 + f^2 + j^2 = 1$
6) $c^2 + g^2 + k^2 = 1$

$$
\begin{aligned}
&7)\quad ade^{i(\theta_a-\theta_d)}+bfe^{i(\theta_b-\theta_f)}+cge^{i(\theta_c-\theta_g)}=0\\
&8)\quad ahe^{i(\theta_a-\theta_h)}+bje^{i(\theta_b-\theta_j)}+cke^{i(\theta_c-\theta_k)}=0\\
&9)\quad hde^{i(\theta_h-\theta_d)}+jfe^{i(\theta_j-\theta_f)}+kge^{i(\theta_k-\theta_g)}=0\\
&10)\quad abe^{i(\theta_a-\theta_b)}+dfe^{i(\theta_d-\theta_f)}+hje^{i(\theta_h-\theta_j)}=0\\
&11)\quad ace^{i(\theta_a-\theta_c)}+dge^{i(\theta_d-\theta_g)}+hke^{i(\theta_h-\theta_k)}=0\\
&12)\quad bce^{i(\theta_b-\theta_c)}+fge^{i(\theta_f-\theta_g)}+jke^{i(\theta_j-\theta_k)}=0
\end{aligned}
\tag{5.9}
$$

must be true. Note that the conjugates of identities 7 through 12 must also be true. Though rewriting the SU(3) matrix in the form (5.8) appears to have complicated matters, it significantly simplifies the matrix multiplication $(\hat{C}^{\dagger}\hat{n}_m\hat{C}-\hat{n}_m)$. For m = 1, this matrix is equal to

$$
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -1+a^2+d^2 & abe^{-i(\theta_a-\theta_b)}+dfe^{-i(\theta_d-\theta_f)} & ace^{-i(\theta_a-\theta_c)}+dge^{-i(\theta_d-\theta_g)} & 0 & 0 & 0 & 0\\
0 & abe^{i(\theta_a-\theta_b)}+dfe^{i(\theta_d-\theta_f)} & -1+b^2+f^2 & bce^{-i(\theta_b-\theta_c)}+fge^{-i(\theta_f-\theta_g)} & 0 & 0 & 0 & 0\\
0 & ace^{i(\theta_a-\theta_c)}+dge^{i(\theta_d-\theta_g)} & bce^{i(\theta_b-\theta_c)}+fge^{i(\theta_f-\theta_g)} & c^2+g^2 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & h^2 & hje^{-i(\theta_h-\theta_j)} & hke^{-i(\theta_h-\theta_k)} & 0\\
0 & 0 & 0 & 0 & hje^{i(\theta_h-\theta_j)} & j^2 & jke^{-i(\theta_j-\theta_k)} & 0\\
0 & 0 & 0 & 0 & hke^{i(\theta_h-\theta_k)} & jke^{i(\theta_j-\theta_k)} & -1+k^2 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\tag{5.10}
$$

Then, using the identities 4 through 6 and 10 through 12 in (5.9), this matrix becomes

$$
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -h^2 & -hje^{-i(\theta_h-\theta_j)} & -hke^{-i(\theta_h-\theta_k)} & 0 & 0 & 0 & 0\\
0 & -hje^{i(\theta_h-\theta_j)} & -j^2 & -jke^{-i(\theta_j-\theta_k)} & 0 & 0 & 0 & 0\\
0 & -hke^{i(\theta_h-\theta_k)} & -jke^{i(\theta_j-\theta_k)} & 1-k^2 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & h^2 & hje^{-i(\theta_h-\theta_j)} & hke^{-i(\theta_h-\theta_k)} & 0\\
0 & 0 & 0 & 0 & hje^{i(\theta_h-\theta_j)} & j^2 & jke^{-i(\theta_j-\theta_k)} & 0\\
0 & 0 & 0 & 0 & hke^{i(\theta_h-\theta_k)} & jke^{i(\theta_j-\theta_k)} & -1+k^2 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}.
\tag{5.11}
$$

Multiplying this matrix from the left by $\langle\psi|$ and from the right by $|\psi\rangle$ and using identity 3 from (5.9) one obtains the collision function for the first QLBE collision function.

$$
\begin{aligned}
\Omega_1 &= (k^2-1)p_1 + j^2 p_2 + h^2 p_3 + \\
&2hj\cos[\theta_h - \theta_j](1-2p_1)\sqrt{(1-p_2)p_2(1-p_3)p_3} + \\
&2hk\cos[\theta_h - \theta_k](1-2p_2)\sqrt{(1-p_1)p_1(1-p_3)p_3} + \\
&2jk\cos[\theta_j - \theta_k](1-2p_3)\sqrt{(1-p_1)p_1(1-p_2)p_2} \quad .
\end{aligned}
\tag{5.12}
$$

With similar work one obtains the second and third collision functions:

$$
\begin{aligned}
\Omega_2 &= g^2 p_1 + (f^2-1)p_2 + d^2 p_3 + \\
&2df\cos[\theta_d - \theta_f](1-2p_1)\sqrt{(1-p_2)p_2(1-p_3)p_3} + \\
&2dg\cos[\theta_d - \theta_g](1-2p_2)\sqrt{(1-p_1)p_1(1-p_3)p_3} + \\
&2fg\cos[\theta_f - \theta_g](1-2p_3)\sqrt{(1-p_1)p_1(1-p_2)p_2}
\end{aligned}
\tag{5.13}
$$

$$
\begin{aligned}
\Omega_3 &= c^2 p_1 + b^2 p_2 + (a^2-1)p_3 + \\
&2ab\cos[\theta_a - \theta_b](1-2p_1)\sqrt{(1-p_2)p_2(1-p_3)p_3} + \\
&2ac\cos[\theta_a - \theta_c](1-2p_2)\sqrt{(1-p_1)p_1(1-p_3)p_3} + \\
&2bc\cos[\theta_b - \theta_c](1-2p_3)\sqrt{(1-p_1)p_1(1-p_2)p_2} \quad .
\end{aligned}
$$

Using identities 10 through 12 in (5.9) and their conjugates one can see that

$$
\begin{aligned}
&13)\quad ab\cos(\theta_a - \theta_b) + df\cos(\theta_d - \theta_f) + hj\cos(\theta_h - \theta_j) = 0 \\
&14)\quad ac\cos(\theta_a - \theta_c) + dg\cos(\theta_d - \theta_g) + hk\cos(\theta_h - \theta_k) = 0 \\
&15)\quad bc\cos(\theta_b - \theta_c) + fg\cos(\theta_f - \theta_g) + jk\cos(\theta_j - \theta_k) = 0
\end{aligned}
\tag{5.14}
$$

so that $\Omega_1 + \Omega_2 + \Omega_3 = 0$.

5.3 Macroscopic scale: Chapman-Enskog

As discussed in section 4.3.2, the quantum lattice Boltzmann equation can be expanded in a Taylor series around local equilibrium to yield

$$
\varepsilon^2 \tau \frac{\partial d_m}{\partial t} + \varepsilon e_m \ell \frac{\partial d_m}{\partial x} + \varepsilon^2 e_m \ell \frac{\partial \delta f_m^{(0)}}{\partial x} + \varepsilon^2 e_m^2 \ell^2 \frac{1}{2}\frac{\partial^2 d_m}{\partial x^2} + \mathrm{O}(\varepsilon^3) = \Omega_m \, . \tag{5.15}
$$

One may sum these equations over *m* with $e_1 = 1$, $e_2 = 0$, $e_3 = -1$, and $\rho = d_1 + d_2 + d_3$ to obtain

$$\varepsilon^2 \tau \frac{\partial \rho}{\partial t} + \varepsilon \ell \frac{\partial}{\partial x}(d_1 - d_3) + \varepsilon^2 \ell \frac{\partial}{\partial x}(\delta f_1^{(0)} - \delta f_3^{(0)}) + \varepsilon^2 \ell^2 \frac{1}{2} \frac{\partial^2 \rho}{\partial x^2} + \mathrm{O}(\varepsilon^3) = 0 . \quad (5.16)$$

The key difficulty now is finding the local equilibrium values d_m, which one needs to solve for $(d_1 - d_3)$ and $(\delta f_1^{(0)} - \delta f_3^{(0)})$ to complete the Chapman-Enskog expansion. Since the collision functions sum to zero, only two of these functions are linearly independent. The collision functions are each equal to zero at local equilibrium. Therefore, the three equations that one must solve to obtain the three unknown equilibrium values are

$$\begin{aligned} &\Omega_1\big|_{p_m = d_m} = 0 \\ &\Omega_2\big|_{p_m = d_m} = 0 \\ &\rho = d_1 + d_2 + d_3 \ . \end{aligned} \quad (5.17)$$

The complexity of the collision functions necessitates making a variable substitution to simplify the first two equations. Making the substitutions

$$d_1 = \frac{1}{1+x^2}, \ d_2 = \frac{1}{1+y^2}, \ d_3 = \frac{1}{1+z^2}, \quad (5.18)$$

and multiplying the first two equations in (5.17) by $(1+x^2)(1+y^2)(1+z^2)$ simplifies these equations so that they no longer depend on the square root of the variables we are trying to solve for. The three equations become

$$\begin{aligned} &((a^2-1)y^2 + 2ab\cos(\theta_a - \theta_b)yz + b^2z^2 - c^2)x^2 \\ &+(2bc\cos(\theta_b - \theta_c)(z^2-1)y + 2ac\cos(\theta_a - \theta_c)(y^2-1)z)x \\ &-2ab\cos(\theta_a - \theta_b)yz + (-b^2 + c^2z^2)y^2 + (1-a^2)z^2 = 0 \end{aligned}$$

$$
\begin{aligned}
&((f^2-1)z^2+2df\cos(\theta_d-\theta_f)yz+d^2z^2-g^2)x^2\\
&+(2dg\cos(\theta_d-\theta_g)(y^2-1)z+2fg\cos(\theta_f-\theta_g)(z^2-1)y)x\\
&-2df\cos(\theta_d-\theta_f)yz+(1-f^2+g^2z^2)y^2-d^2z^2=0
\end{aligned}
\tag{5.19}
$$

$$
\frac{1}{1+x^2}+\frac{1}{1+y^2}+\frac{1}{1+z^2}=\rho
$$

Unfortunately, the first two equations are quadratic in *x*, *y*, and *z*, and it is not yet known if it is possible to find an analytic solution to these three equations, even with the additional constraints that come from replacing *a*-*g* with the most general values from a SU(3) matrix. Additional research in this area is left for future work, and I have focused on a specific SU(3) matrix for the remainder of this thesis.

6 Diffusion Equation

6.1 Analytic treatment

The diffusion equation can be modeled if the SU(3) matrices inside the collision function are equal to

$$
\mathrm{U}(3)=\frac{e^{-i\pi/6}}{\sqrt{3}}\begin{pmatrix} e^{i2\pi/3} & 1 & 1\\ 1 & e^{i2\pi/3} & 1\\ 1 & 1 & e^{i2\pi/3}\end{pmatrix} \tag{6.1}
$$

Inserting this into (5.12) and (5.13) gives the collision functions

$$
\begin{aligned}
\Omega_1=\tfrac{1}{3}[&-2p_1+p_2+p_3-2(2p_1-1)\sqrt{p_2(1-p_2)p_3(1-p_3)}+\\
&(2p_2-1)\sqrt{p_1(1-p_1)p_3(1-p_3)}+(2p_3-1)\sqrt{p_1(1-p_1)p_2(1-p_2)}]
\end{aligned}
$$

$$
\begin{aligned}
\Omega_2=\tfrac{1}{3}[&p_1-2p_2+p_3+(2p_1-1)\sqrt{p_2(1-p_2)p_3(1-p_3)}+\\
&-2(2p_2-1)\sqrt{p_1(1-p_1)p_3(1-p_3)}+(2p_3-1)\sqrt{p_1(1-p_1)p_2(1-p_2)}]
\end{aligned}
$$

$$\Omega_3 = \tfrac{1}{3}[p_1 + p_2 - 2p_3 + (2p_1 - 1)\sqrt{p_2(1-p_2)p_3(1-p_3)} + (2p_2 - 1)\sqrt{p_1(1-p_1)p_3(1-p_3)} - 2(2p_3 - 1)\sqrt{p_1(1-p_1)p_2(1-p_2)}] \quad (6.2)$$

where, as expected, $\Omega_1 + \Omega_2 + \Omega_3 = 0$.

Running the numerical simulation presented in section 6.2 suggests that the equilibrium values d_m, are $d_1 = d_2 = d_3 = \rho/3$. This can be easily verified by noting that at equilibrium, the collision matrix should not change the occupation probabilities; that is $\hat{C}\left|\psi_{eq}\right\rangle = \left|\psi_{eq}\right\rangle$. Thus

$$\frac{e^{-i\pi/6}}{\sqrt{3}}\begin{pmatrix} \sqrt{3}e^{i\pi/6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{i2\pi/3} & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & e^{i2\pi/3} & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & e^{i2\pi/3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{i2\pi/3} & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & e^{i2\pi/3} & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & e^{i2\pi/3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{3}e^{i\pi/6} \end{pmatrix} \begin{pmatrix} \sqrt{d_1 d_2 d_3} \\ \sqrt{d_1 d_2 (1-d_3)} \\ \sqrt{d_1 (1-d_2) d_3} \\ \sqrt{(1-d_1) d_2 d_3} \\ \sqrt{(1-d_1)(1-d_2) d_3} \\ \sqrt{(1-d_1) d_2 (1-d_3)} \\ \sqrt{d_1 (1-d_2)(1-d_3)} \\ \sqrt{(1-d_1)(1-d_2)(1-d_3)} \end{pmatrix} = \begin{pmatrix} \sqrt{d_1 d_2 d_3} \\ \sqrt{d_1 d_2 (1-d_3)} \\ \sqrt{d_1 (1-d_2) d_3} \\ \sqrt{(1-d_1) d_2 d_3} \\ \sqrt{(1-d_1)(1-d_2) d_3} \\ \sqrt{(1-d_1) d_2 (1-d_3)} \\ \sqrt{d_1 (1-d_2)(1-d_3)} \\ \sqrt{(1-d_1)(1-d_2)(1-d_3)} \end{pmatrix} \quad (6.3)$$

If we insert $d_1 = d_2 = d_3 = \rho/3$ into the above equation, the expression is reduced to the following equations

$$\begin{aligned} &\left(\frac{\rho}{3}\right)^{2/3} = \left(\frac{\rho}{3}\right)^{2/3} \\ &\frac{e^{-i\pi/6}}{\sqrt{3}}(2 + e^{i\pi 2/3})\frac{\rho}{3}\sqrt{\left(1-\frac{\rho}{3}\right)} = \frac{\rho}{3}\sqrt{\left(1-\frac{\rho}{3}\right)} \\ &\frac{e^{-i\pi/6}}{\sqrt{3}}(2 + e^{i\pi 2/3})\left(1-\frac{\rho}{3}\right)\sqrt{\frac{\rho}{3}} = \left(1-\frac{\rho}{3}\right)\sqrt{\frac{\rho}{3}} \\ &\left(1-\frac{\rho}{3}\right)^{2/3} = \left(1-\frac{\rho}{3}\right)^{2/3}, \end{aligned} \quad (6.4)$$

which of course can be further simplified to the identities 1 = 1 and $\frac{e^{-i\pi/6}}{\sqrt{3}}(2 + e^{i\pi 2/3}) = 1$.

This verifies that the diffusion equation collision matrix has no effect on the local state when $d_1 = d_2 = d_3 = \rho/3$. Thus, this must be the equilibrium condition.

After finding the equilibrium condition, the next step is to perform the Chapman-Enskog expansion around local equilibrium. From section 4.3.2, we know that the equations

$$\varepsilon^2 \tau \frac{\partial d_m}{\partial t} + \varepsilon e_m \ell \frac{\partial d_m}{\partial x} + \varepsilon^2 e_m \ell \frac{\partial \delta f_m^{(0)}}{\partial x} + \varepsilon^2 e_m^2 \ell^2 \frac{1}{2} \frac{\partial^2 d_m}{\partial x^2} + \mathrm{O}(\varepsilon^3) = \Omega_m \tag{6.5}$$

and

$$\varepsilon \ell \hat{e} \left| \frac{\partial d}{\partial x} \right\rangle = \varepsilon \hat{J} \left| \delta f^{(0)} \right\rangle \tag{6.6}$$

are the Taylor series expansion and first moment equation of the FQLBE respectively. The vectors in (6.6) are now

$$\left| \frac{\partial d_m}{\partial x} \right\rangle = \begin{pmatrix} \frac{\partial d_1}{\partial x} \\ \frac{\partial d_2}{\partial x} \\ \frac{\partial d_3}{\partial x} \end{pmatrix} \quad \text{and} \quad \left| \delta f^{(0)} \right\rangle = \begin{pmatrix} \delta f_1^{(0)} \\ \delta f_2^{(0)} \\ \delta f_3^{(0)} \end{pmatrix} \tag{6.7}$$

and the matrices are

$$\hat{e} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad \text{and} \quad \hat{J} = \left. \begin{pmatrix} \frac{\partial \Omega_1}{\partial f_1} & \frac{\partial \Omega_1}{\partial f_2} & \frac{\partial \Omega_1}{\partial f_3} \\ \frac{\partial \Omega_2}{\partial f_1} & \frac{\partial \Omega_2}{\partial f_2} & \frac{\partial \Omega_2}{\partial f_3} \\ \frac{\partial \Omega_3}{\partial f_1} & \frac{\partial \Omega_3}{\partial f_2} & \frac{\partial \Omega_3}{\partial f_3} \end{pmatrix} \right|_{f_{1,2,3} = \rho/3} = \begin{pmatrix} -1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -1 \end{pmatrix} \tag{6.8}$$

Once again $\hat{J}$ is singular. The eigenvalues and left and right eigenvectors of $\hat{J}$ are

$$\lambda_1 = -\frac{3}{2} \qquad \left\langle E_1 \right| = \frac{1}{3}\begin{pmatrix} -1 & -1 & 2 \end{pmatrix} \qquad \left| \mathrm{E}_1 \right\rangle = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

$$\lambda_2 = -\frac{3}{2} \qquad \langle E_2| = \frac{1}{3}(-1 \quad 2 \quad -1) \qquad |\mathrm{E}_2\rangle = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$$
$$\lambda_3 = 0 \qquad \langle E_3| = \frac{1}{3}(1 \quad 1 \quad 1) \qquad |\mathrm{E}_3\rangle = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{6.9}$$

with lengths selected so that $\langle E_i | E_j \rangle = \delta_{ij}$. Since $\lambda_1 = \lambda_2$, we can use the same sort of trick Yepez used (described in section 4.3.2) to solve for $|\delta f^{(0)}\rangle$: multiply both sides of the first moment equation (6.6) by $\hat{J}$. $\hat{J}$ squared is equal to

$$\begin{aligned} \left(\lambda_1 |E_1\rangle\langle E_1| + \lambda_2 |E_2\rangle\langle E_2|\right)^2 &= \lambda_1^2 \left(|E_1\rangle\langle E_1| + |E_2\rangle\langle E_2|\right)\left(|E_1\rangle\langle E_1| + |E_2\rangle\langle E_2|\right) \\ &= \lambda_1^2 \left(|E_1\rangle\langle E_1| + |E_2\rangle\langle E_2|\right) \\ &= \lambda_1 \hat{J} \end{aligned} \tag{6.10}$$

Thus equation (6.6) can be solved for $|\delta f^{(0)}\rangle$ as follows.

$$\begin{aligned} \hat{J}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \hat{J}^2 |\delta f^{(0)}\rangle \\ \hat{J}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \hat{J}\lambda_1 |\delta f^{(0)}\rangle \\ &\text{implies} \\ \ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= \lambda_1 |\delta f^{(0)}\rangle \\ \frac{1}{\lambda_1}\ell\hat{e}\left|\frac{\partial d}{\partial x}\right\rangle &= |\delta f^{(0)}\rangle \end{aligned} \tag{6.11}$$

Inserting this solution into (6.5) along with the equilibrium condition $d_m = \rho/3$ produces

$$\varepsilon^2\tau\frac{1}{3}\frac{\partial d_m}{\partial t} + \varepsilon^2 e_m^2 \ell^2 \frac{1}{3}\frac{\partial d_m}{\partial x} - \frac{1}{18}\varepsilon^2 e_m^2 \ell^2 \frac{\partial d_m}{\partial x} + \mathrm{O}(\varepsilon^3) = \Omega_m . \tag{6.12}$$

Summing these equation over m with $e_1 = 1$, $e_2 = 0$, and $e_3 = -1$ produces the diffusion equation

$$\frac{\partial \rho}{\partial t} = \frac{\ell^2}{9\tau}\frac{\partial^2 \rho}{\partial x^2} + \mathrm{O}(\varepsilon^3). \tag{6.13}$$

This derivation suggests that the simulation is accurate to at least first order in time and second order in space. The diffusion coefficient for the three qubit FQLGA is equal to $d = \ell^2 / 9\tau$ where ℓ is the lattice spacing and τ is the time step. It is possible to arbitrarily change either ℓ or τ to adjust the diffusion constant. For instance, if one wishes the diffusion constant to be 1/9 m^2/s, then ℓ is one meter if the time step τ is defined to be one second.

6.2 Numerical treatment

The results of a numerical simulation of the diffusion equation three qubit FQLGA model are presented in this section. All numerical simulations were run in Mathematica 5.1 or 5.2 on a conventional desktop computer.

6.2.1 Sum of Gaussian and sinusoid initial condition

The three qubit FQLGA is carried out using circular boundary conditions—that is, $\rho(0,t) = \rho(L,t)$ where L is the length of the lattice, set to 250ℓ for this simulation. The initial condition was

$$\rho(x,0) = \frac{1}{6}\sin\left(\frac{2\pi x}{L}\right) + \frac{7}{10}e^{-\left(\frac{x-L/2}{L/10}\right)^2} + \frac{13}{60}. \tag{6.14}$$

The solution to the diffusion equation with circular boundary conditions is

$$\rho(x,t)_{exact} = B + \sum_{m=1}^{\infty} e^{d(2\pi m/L)^2 t}\left(C_m \sin\left(\frac{2\pi mx}{L}\right) + D_m \cos\left(\frac{2\pi mx}{L}\right)\right) \tag{6.15}$$

where

$$\begin{aligned} B &= \frac{1}{L}\int_0^L \rho(x,0)dx \\ D_m &= \frac{2}{L}\int_0^L \rho(x,0)\cos\left(\frac{2\pi mx}{L}\right)dx \\ C_m &= \frac{2}{L}\int_0^L \rho(x,0)\sin\left(\frac{2\pi mx}{L}\right)dx\ . \end{aligned} \tag{6.16}$$

The derivation of this solution is presented in Appendix B. From this solution, one should expect the simulation to exhibit a roughly exponential decay from the initial condition to the average of the initial state D_0, where the exponential decay constant of higher frequency terms (in space) is larger than that of low frequency terms. The qubits are each initialized to be $\rho(z,0)/3$ so that they will be close to local equilibrium after the first and subsequent time steps. The results of the simulation are presented in Figure 13.

The FQLGA results are shown in gray while the black line is the sum of the first 14 terms of the analytic solution (6.15). The sum is cut off after 14 terms because the coefficients C_m and D_m for $m > 14$ are less than the negligible value 10^{-10}. The average percent error between the simulation and the analytic solution is shown in Figure 14 and Figure 16. The maximum percent errors are shown in Figure 15 and Figure 17. From these figures, one can see that the magnitude of these errors oscillates within the first ten time steps, after which they begin to steadily decrease as the simulation approaches equilibrium.

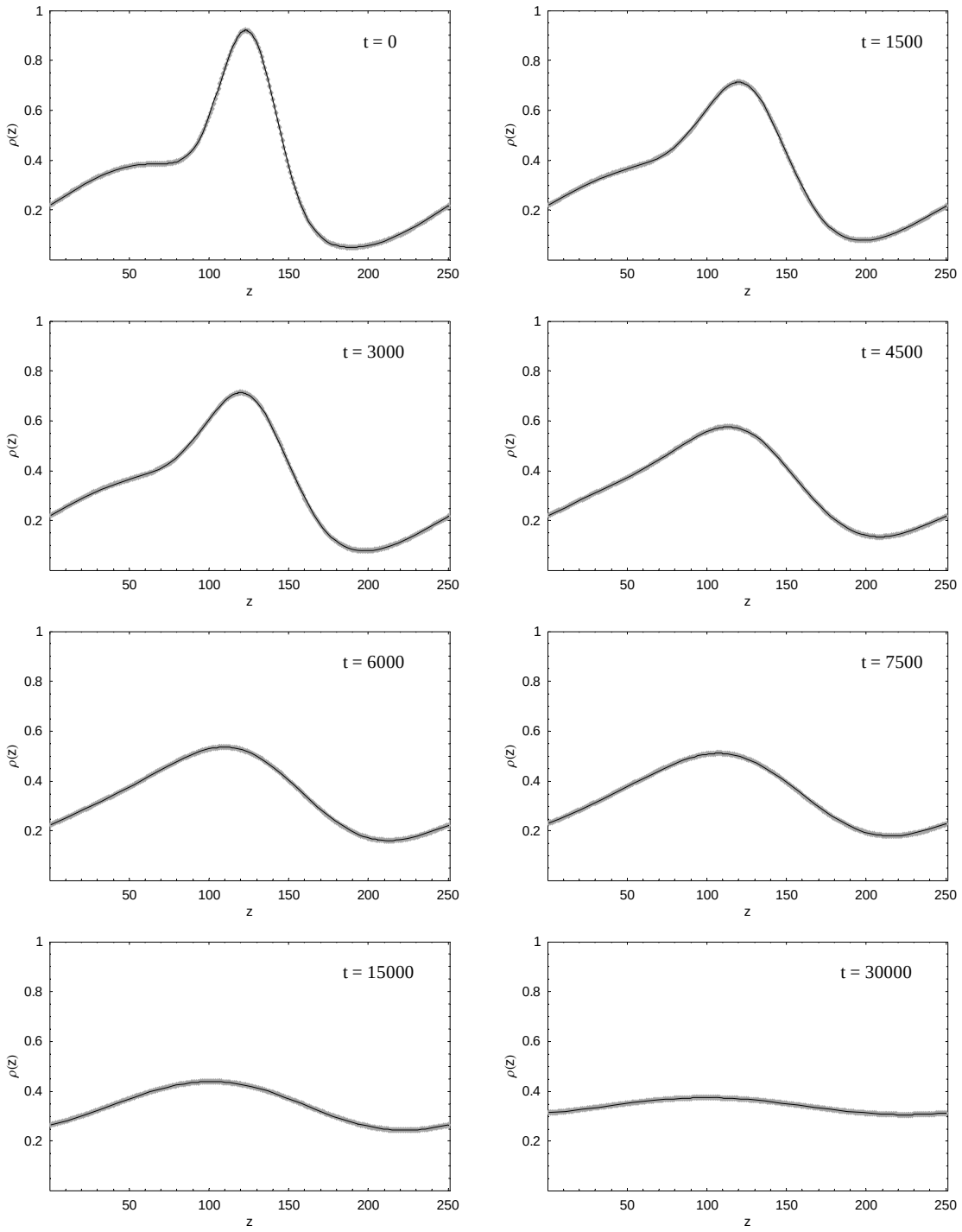

Figure 13. The results of the FQLGA simulation with circular boundary conditions for a number of time steps. The simulation lattice points make up the thick gray line while the exact solution is in black. The solution decays to the average density of the initial condition.

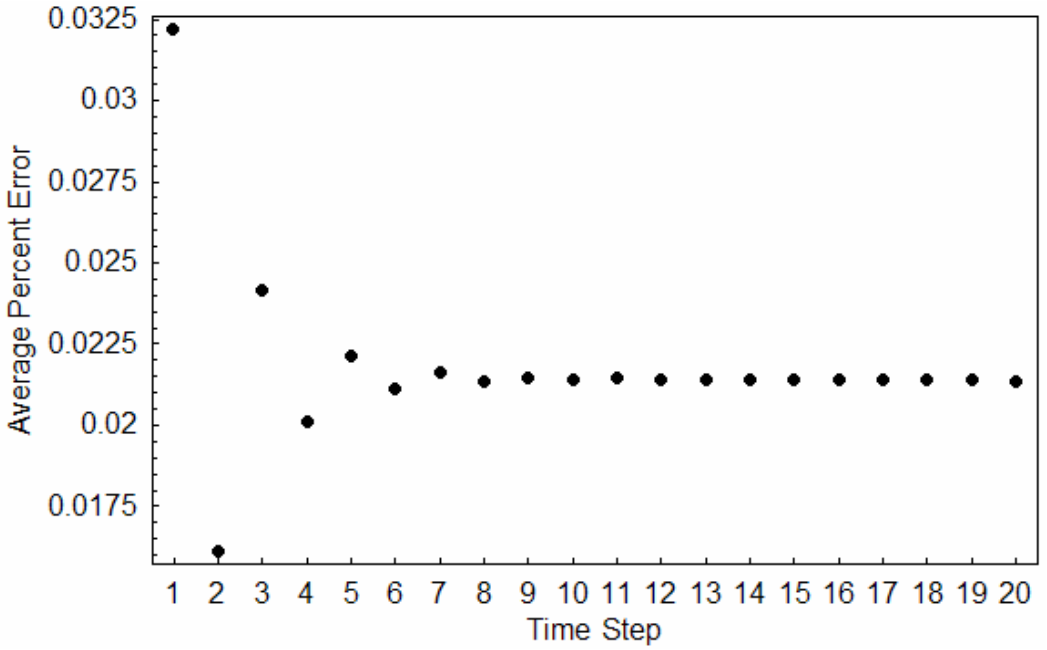

Figure 14. Plot of the average percent errors over the entire lattice for the first 20 time steps. Oscillations in these errors are apparent at the beginning of the simulation but disappear after about ten time steps. After this, the errors decrease as shown in Figure 16.

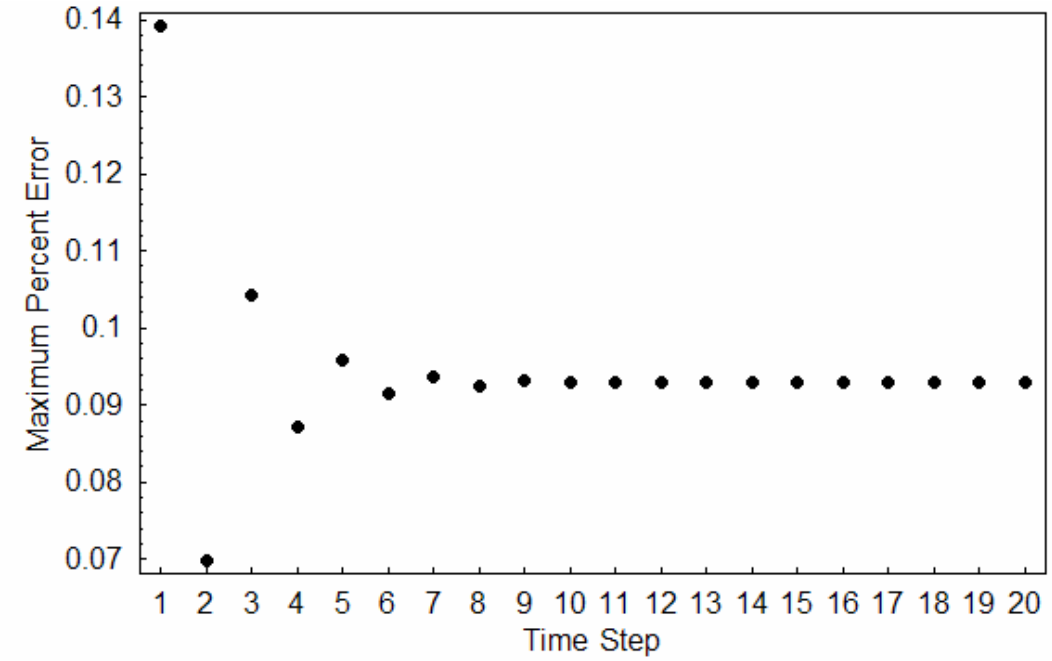

Figure 15. Plot of the maximum percent errors in the lattice for the first 20 time steps. Oscillations in the maximum percent errors are also apparent at the beginning of the simulation. The largest percent error during the simulation occurs after the first time step, and is ~ 0.14%.

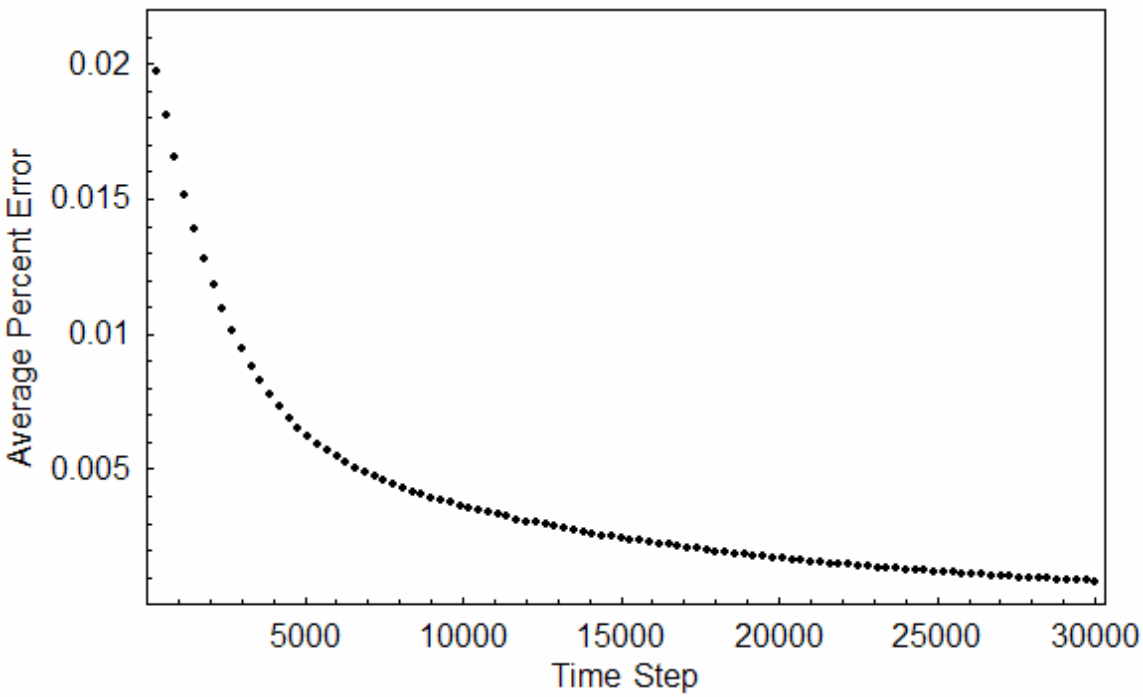

Figure 16. Plot of the average percent errors every 300 time steps. The errors decrease as the simulation moves closer to equilibrium.

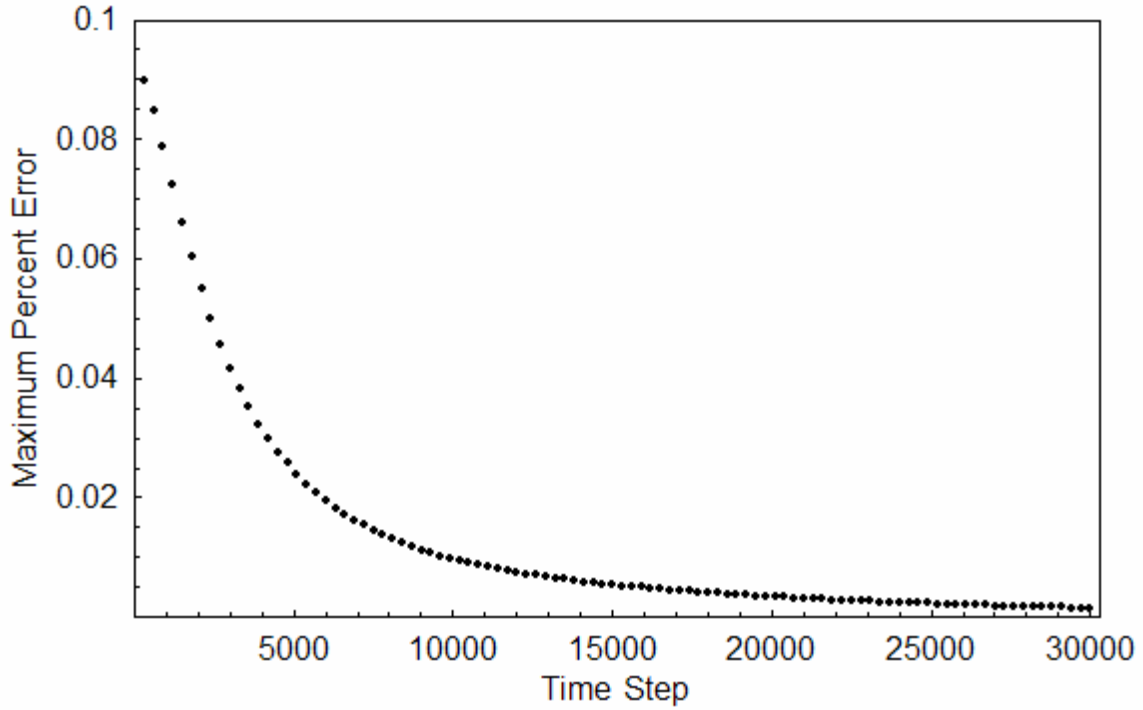

Figure 17. Plot of the maximum percent error every 300 time steps.

From these plots we see that the maximum percent error is only about a tenth of a percent, while the largest average percent error is .03 %, indicating that the simulation is a very accurate model of the diffusion equation at this lattice resolution. Increasing the resolution (that is, increasing the number of lattice sites) only improves the average percent error, as is clear from Figure 18. This is a log-log plot of the lattice resolution verses the average percent error. The plot was made by running identical simulations for 15 time steps with lattice lengths ranging from 50ℓ to 12800ℓ. The error decreases as $percent\ error \sim \delta x^{2.01}$ where $\delta x \equiv \ell / L$, indicating second order convergence in space.

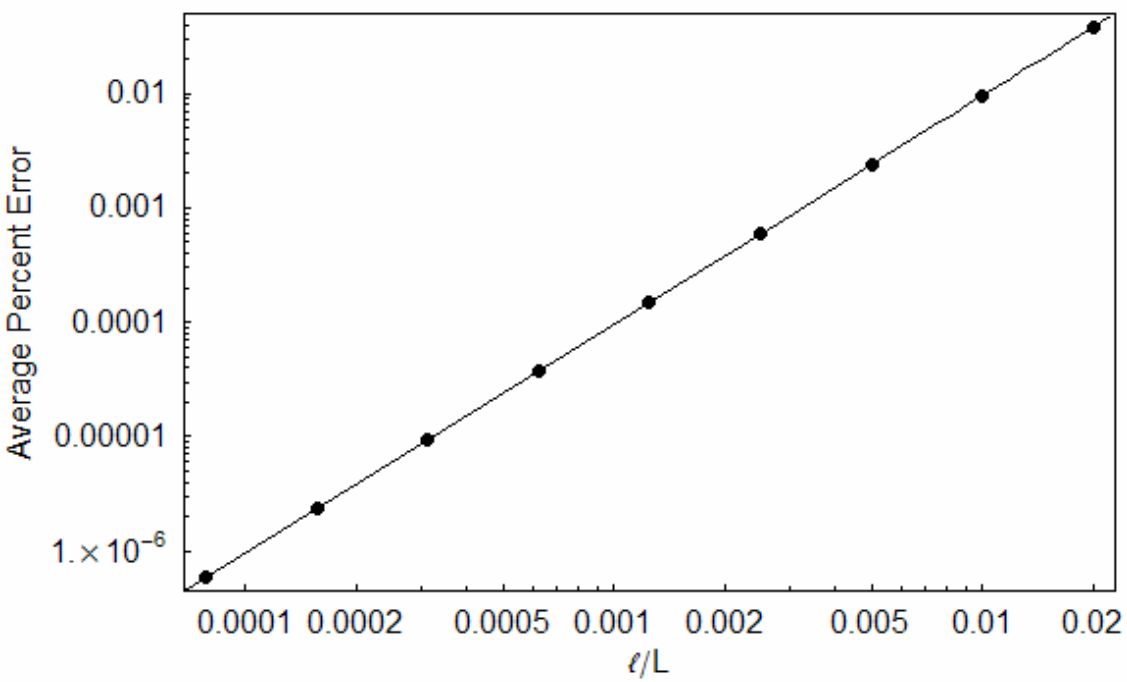

Figure 18. Log-log plot of the average percent error verses the lattice resolution. The same simulation was run with lattice lengths ranging from 50ℓ to 12800ℓ, each evaluated at 15τ. The slope of the best fit solid line is 2.01, indicating second order convergence in space.

6.2.2 Delta function initial condition

A two qubit FQLGA simulation of the diffusion equation was developed by Yepez [8]. This algorithm has the unfortunate property that the lattice consists of two

interpenetrating but noninteracting lattices. This is due to the fact that when qubit one from lattice site *m* streams right, qubit two of lattice site *m*+1 streams left. Thus, the left streaming qubit reaches lattice site *m* without ever having collided with the right streaming qubit, which is now on lattice site *m*+1. This occurs everywhere in the lattice so that qubits will only interact with those from every other lattice site. This can be demonstrated if, for example, the simulation's initial condition is a delta function[1] as shown in the left column of Figure 19.

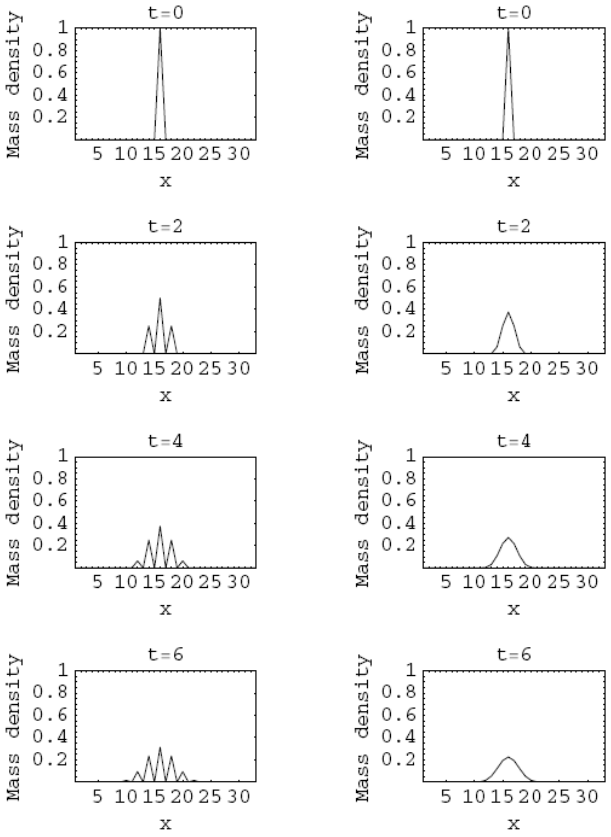

Figure 19. Results of Yepez's two qubit diffusion equation simulation with a delta function initial condition. The left column shows the results of the unmodified algorithm with qubits one and two streaming every time step. The sharp spikes in these pictures are due to the noninteracting nature of all the qubits. The right column shows a modified simulation with the left moving qubits streaming every even time step and the right moving qubits streaming every odd time step. The modified algorithm corrects the deficiency in the algorithm. This figure was used with permission from [9].

[1] This is, of course, a Dirac delta function with height equal to one.

There are two methods of correcting this deficiency. The first, taken by Yepez, is to stream the left moving qubits every even time step and the right moving qubits every odd time step. The results of this process are shown in the right column in Figure 19. An alternate solution to this problem is to use the three qubit algorithm I have developed.

The results of a simulation run with a delta function initial condition are presented in Figure 20. In this figure, the time evolution of this function is compared to the analytic time evolution of a piecewise defined function. This piecewise function is equal to zero everywhere except between the lattice sites adjacent to the delta function, where it is triangular in shape with height equal to one. This ensures that the total integrated area of the piecewise function is equal to the total density of the lattice, so that at infinite time, when the evolved lattice and piecewise function have reached equilibrium, the constant lattice density will equal the height of the evolved piecewise function.

The three qubit algorithm does not display the unusual pattern present in the unmodified version of the two qubit algorithm, since the streaming qubits interact via the stationary qubits. However, it does not seem that this method is superior to the modified two qubit algorithm for two reasons. The first is that this algorithm can be much more difficult to implement experimentally on a NMR computer because increasing the number of qubits per node increases the complexity of the system. In addition, the diffusion coefficient for the three qubit algorithm is $\ell^2/9\tau$, whereas it is $\ell^2/2\tau$ for the two qubit algorithm. This means that for equal lattice lengths and time steps, the two qubit algorithm will evolve 4.5 times faster than the three qubit algorithm.

One should note that neither diffusion equation algorithm effectively models the evolution of a large gradient function with poor lattice resolution, such as a delta

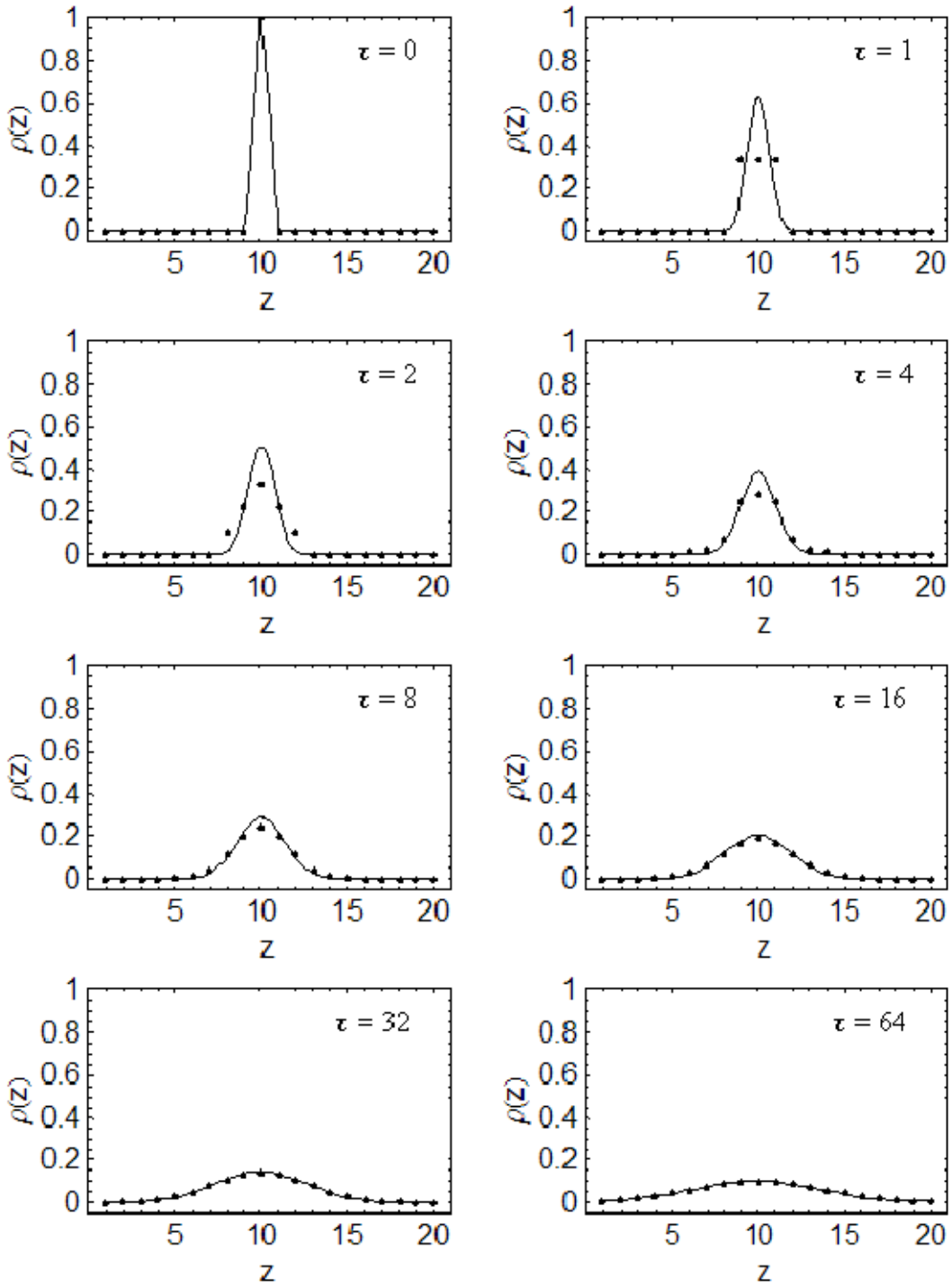

Figure 20. Results of the three qubit FQLGA (black data points) with the initial condition equal to zero everywhere except at the center lattice site where it is equal to one. The time evolution of this function is compared to the analytic time evolution of a piecewise defined function (solid line), with an integrated area equal to the total density of the FQLGA simulation.

function. This is because the "smallness parameter" ε, which was defined to be on the order of the Knudsen number, is no longer small. The Knudsen number is equal to the mean free path (the lattice spacing) over the characteristic length (the width of the delta function). Since these two numbers are roughly equal, it is no longer appropriate to neglect the terms of order ε^3 and higher in the equation

$$\frac{\partial \rho}{\partial t} = \frac{\ell^2}{9\tau}\frac{\partial^2 \rho}{\partial x^2} + \mathrm{O}(\varepsilon^3), \tag{6.17}$$

and the diffusion equation is no longer a good approximation of the governing equation of the lattice. It is not surprising, then, that the agreement between the algorithm and the analytic solution to the diffusion equation is poor at the beginning of the simulation. Nevertheless, at times after about 16τ the Knudsen number becomes smaller and the algorithm begins to converge to the analytic solution of the diffusion equation.

Another case where the FQLGA can be expected to perform poorly when modeling the diffusion equation is when the assumption

$$f_m = d_m + \varepsilon\delta f_m^{(0)} + \varepsilon^2 \delta f_m^{(1)} + \mathrm{O}(\varepsilon^3) \tag{6.18}$$

is no longer valid—i.e. if $\delta f_m^{(0)} / d_m$ is no longer on the order of $\varepsilon \sim$ Kn. In this case, the Chapman-Enskog expansion will fail to produce an equation which accurately models the lattice in the continuum limit. For example, if one were to naively choose the initial conditions of the lattice so that the density was not distributed near local equilibrium to begin with, then $\delta f_m^{(0)} / d_m$ will be large and the lattice will not behave in a manner which mimics the diffusion equation. This would have been the case, for instance, if the density distribution shown in Figure 13 had initially been distributed so that the total particle

density at every lattice site was located in a single qubit, instead of being spread equally between qubits one, two, and three. Therefore, to model fluid dynamics with a FQLGA it is essential to distribute the qubit probabilities near local equilibrium when a lattice is initialized.

Returning to the delta function simulation, it is worth pointing out that this simulation contains the most extreme gradient possible in the density function ρ. For this reason, it represents a strong test of the numerical stability of both the two and three qubit FQLGAs. Both algorithms perform well under these conditions due to the unitary nature and structure of the collision operators. Since the operators are unitary and block diagonal to preserve the probability of measuring qubits in the state $|1\rangle$, the total probability of finding all the qubits in the simulation in the state $|1\rangle$ is preserved. Thus, the values of particle density will remain bound and not "blow up," and are in this sense numerically stable.

Another common definition of numerical stability concerns the accuracy with which an algorithm models an equation [32]. It is clear from the results of this thesis that the three qubit Factorized Quantum Lattice Gas Algorithm is, in this sense, a numerically stable model of the diffusion equation when the Knudsen is much less than one. In addition, the results presented in Figure 20 suggest that even if the Knudsen number is initially large, the algorithm can be numerically accurate at later times as long as the initial conditions of the lattice and the simulated function are the same, and the integrated area of the simulated function is equal to the average density of the FQLGA.

7 Conclusion

This thesis was written in support of AFRL's and AFOSR's Quantum Computation for Physical Modeling basic research theme, by exploring and extending a quantum algorithm designed to model fluid dynamics using a practical quantum computer. To date, the most advanced type II quantum computer prototype uses NMR technology. Two properties of the algorithm presented in this paper make it an ideal test case for the modeling utility of a three qubit per node type II NMR quantum computer. The first is that there is a maximum of three qubits that the algorithm requires to be coherent at a given time. This allows the algorithm to be run on a set of three qubit parallel quantum computers connected by classical communications channels: a type II quantum computer. The second property that makes the algorithm ideal for a NMR quantum computer is that information is stored in the probability coefficients of the qubit basis states. Obtaining these probabilities requires an ensemble measurement of identical quantum computers running the same algorithm, which is precisely the sort of measurement used in a NMR machine. Therefore, this algorithm represents a good test of the computational capabilities of a NMR quantum computer.

This thesis extended the Factorized Quantum Lattice Gas Algorithm from two qubits to three, in an effort to improve the algorithm and possibly obtain a new macroscopic governing equation. The most general three qubit collision operator that preserves particle number was derived, along with the Quantum Lattice Boltzmann Equation for this operator. A partial derivation of the governing macroscopic equation for the algorithm in the continuum limit was presented.

Difficulties in deriving the qubit local equilibrium values lead me to consider a more specific collision operator. For this operator, the governing macroscopic equation for the lattice in the continuum limit was the diffusion equation. A numerical simulation of this algorithm carried out on a conventional desktop computer with a lattice length $L = 250\ell$ was then presented and compared to the analytic solution of the one dimensional diffusion equation with circular boundary conditions. The simulation and analytic solution matched very well—the largest average percent error occurred after the first time step and was 0.03%. Thereafter the error decreased as the simulation progressed. Repeated simulations with identical initial conditions but varying lattice resolutions revealed that the simulation possesses second order convergence in space.

Simulation of a severely under-resolved gradient (a Dirac delta function) was also presented to check for numerical instabilities. No numerical overflows occurred and the model remained stable owing to the collision operator being unitary and block diagonal, preserving particle number. A comparison of the delta function evolution using the FQLGA, to the analytic time evolution of a piecewise defined function revealed that the diffusion equation remained an accurate governing equation for the FQLGA in the continuum limit, as long as the Knudsen number is small. In addition, it was observed that the Chapman-Enskog expansion will not result in a valid governing PDE unless the ratio of the lattice deviation from local equilibrium to the local equilibrium density is on the order of the Knudsen number.

Appendix A. SU(3) Matrix

A unitary matrix may be generated by exponentiating a Hermitian matrix. Exponentiating a Hermitian quantum mechanical Hamiltonian to generate a unitary time evolution operator is a well known example of this.

The collection of all n by n unitary matrices forms a closed group known as U(n). The closed subgroup of all unitary n by n matrices with a determinant equal to one is called "special" and is denoted SU(n). Obviously, it is possible to create members of these groups by exponentiating Hermitian matrices. An element **R** of the unitary group G may be written

$$\mathbf{R} = \exp[i\theta\mathbf{S}] \tag{A.1}$$

where the Hermitian **S** is called a generator of G [31]. In addition, since special unitary matrices have a determinant equal to one, the generators of a special unitary matrix must be traceless [31].

$$\det(\mathbf{R}) = \exp(\mathrm{tr}(\ln(\mathbf{R}))) = \exp(i\theta\,\mathrm{tr}(\mathbf{S})) = 1 \Rightarrow \mathrm{tr}(\mathbf{S}) = 0 \tag{A.2}$$

Since SU(n) is a closed group, multiplying two or more elements within the group will produce another element of that group. Multiplying all the exponentiated generators of a group is one way to create the most general element of that group. For instance, a possible choice for the generators of SU(3) are the Gell-Mann matrices:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}. \tag{A.3}$$

Notice that by multiplying each of these matrices by a free parameter and summing them together one will obtain the most general three by three traceless Hermitian matrix.

The exponentiated Gell-Mann matrices are

$$\mathbf{R}_1 = \begin{pmatrix} \cos(\alpha) & i\sin(\alpha) & 0 \\ i\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{R}_2 = \begin{pmatrix} \cos(\beta) & \sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_3 = \begin{pmatrix} \cos(\gamma) & 0 & i\sin(\gamma) \\ 0 & 1 & 0 \\ i\sin(\gamma) & 0 & \cos(\gamma) \end{pmatrix}, \quad \mathbf{R}_4 = \begin{pmatrix} \cos(\delta) & 0 & \sin(\delta) \\ 0 & 1 & 0 \\ -\sin(\delta) & 0 & \cos(\delta) \end{pmatrix}$$

$$\mathbf{R}_5 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varepsilon) & i\sin(\varepsilon) \\ 0 & i\sin(\varepsilon) & \cos(\varepsilon) \end{pmatrix}, \quad \mathbf{R}_6 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \tag{A.4}$$

$$\mathbf{R}_7 = \begin{pmatrix} e^{i\eta} & 0 & 0 \\ 0 & e^{-i\eta} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{R}_8 = \begin{pmatrix} e^{i\xi} & 0 & 0 \\ 0 & e^{i\xi} & 0 \\ 0 & 0 & e^{-i2\xi} \end{pmatrix}$$

Multiplying these matrices together is one method of generating the most general SU(3) matrix. This matrix is very complicated and not worth explicitly writing here.

For the purposes of the FQLGA, this matrix can be simplified a bit if the order of multiplication is

$$\begin{aligned} \mathrm{SU}(3) &= \mathbf{R}_8\mathbf{R}_7\mathbf{R}_6\mathbf{R}_5\mathbf{R}_4\mathbf{R}_3\mathbf{R}_2\mathbf{R}_1 \\ &= \mathbf{R}_8\mathbf{R}_7\mathbf{M} \end{aligned} \tag{A.5}$$

where $\mathbf{M}$ is the product of $\mathbf{R}_6$ through $\mathbf{R}_1$. Inserting this notation into equation (5.4) we see that

$$\hat{C}^{\dagger}\hat{n}_m\hat{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathbf{M}^{\dagger}\mathbf{R}_7^{\dagger}\mathbf{R}_8^{\dagger}\hat{D}_{m,1}\mathbf{R}_8\mathbf{R}_7\mathbf{M} & 0 & 0 \\ 0 & 0 & \mathbf{M}^{\dagger}\mathbf{R}_7^{\dagger}\mathbf{R}_8^{\dagger}\hat{D}_{m,2}\mathbf{R}_8\mathbf{R}_7\mathbf{M} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad \text{(A.6)}$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathbf{M}^{\dagger}\hat{D}_{m,1}\mathbf{M} & 0 & 0 \\ 0 & 0 & \mathbf{M}^{\dagger}\hat{D}_{m,2}\mathbf{M} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

since the matrices $\hat{D}_{m,n}$, $\mathbf{R}_7$, and $\mathbf{R}_8$ are all diagonal. For instance, for the matrix $\hat{D}_{1,1}$ one obtains

$$\mathbf{R}_7^{\dagger}\mathbf{R}_8^{\dagger}\hat{D}_{1,1}\mathbf{R}_8\mathbf{R}_7$$
$$= \begin{pmatrix} e^{-i\eta} & 0 & 0 \\ 0 & e^{i\eta} & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} e^{-i\xi} & 0 & 0 \\ 0 & e^{-i\xi} & 0 \\ 0 & 0 & e^{i2\xi} \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} e^{i\xi} & 0 & 0 \\ 0 & e^{i\xi} & 0 \\ 0 & 0 & e^{-i2\xi} \end{pmatrix}\begin{pmatrix} e^{i\eta} & 0 & 0 \\ 0 & e^{-i\eta} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{(A.7)}$$
$$= \begin{pmatrix} e^{-i\eta}e^{-i\xi}e^{i\xi}e^{i\eta} & 0 & 0 \\ 0 & e^{i\eta}e^{-i\xi}e^{i\xi}e^{-i\eta} & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \hat{D}_{1,1}$$

Therefore, for the purposes of this FQLGA it is possible to replace the full eight parameter SU(3) matrix with the six parameter matrix **M**.

In fact, preliminary investigation suggests that it may be possible to replace **M** with a simpler four parameter matrix, which we shall label **N**. This can best be understood by considering that the purpose of the matrix **N** is to redistribute probabilities among three basis states. As an example, suppose we were interested in swapping the probabilities between the first and second states. One could then use the following swap matrix to do so

$$\mathbf{Swap}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.8}$$

If, however, one was interested in only "partially swapping" the probabilities then it would be better to use the one over α^{th} root of $\mathbf{Swap}_1$, which is

$$\begin{aligned} \mathbf{Swap}_1^{\alpha} &= \lambda_1^{\alpha} |E_1\rangle\langle E_1| + \lambda_2^{\alpha} |E_2\rangle\langle E_2| + \lambda_3^{\alpha} |E_3\rangle\langle E_3| \\ &= e^{i\pi\alpha} \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \begin{pmatrix} \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} + 1^{\alpha} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} + 1^{\alpha} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 1+e^{i\pi\alpha} & 1-e^{i\pi\alpha} & 0 \\ 1-e^{i\pi\alpha} & 1+e^{i\pi\alpha} & 0 \\ 0 & 0 & 2 \end{pmatrix}. \end{aligned} \tag{A.9}$$

There are three more ways that one can swap probabilities among qubits. They are listed below along with their associated roots.

$$\begin{aligned} \mathbf{Swap}_2 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{Swap}_2^{\beta} = \frac{1}{2} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1+e^{i\pi\beta} & 1-e^{i\pi\beta} \\ 0 & 1-e^{i\pi\beta} & 1+e^{i\pi\beta} \end{pmatrix} \\ \mathbf{Swap}_3 &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{Swap}_3^{\gamma} = \frac{1}{2} \begin{pmatrix} 1+e^{i\pi\gamma} & 0 & 1-e^{i\pi\gamma} \\ 0 & 2 & 0 \\ 1-e^{i\pi\gamma} & 0 & 1+e^{i\pi\gamma} \end{pmatrix} \\ \mathbf{Swap}_4 &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{Swap}_4^{\phi} = \frac{1}{3} \begin{pmatrix} 1+2e^{-i\pi\phi} & 1-e^{-i\pi\phi} & 1-e^{-i\pi\phi} \\ 1-e^{-i\pi\phi} & 1+2e^{-i\pi\phi} & 1-e^{-i\pi\phi} \\ 1-e^{-i\pi\phi} & 1-e^{-i\pi\phi} & 1+2e^{-i\pi\phi} \end{pmatrix} \end{aligned} \tag{A.10}$$

The determinants of all the root matrices are complex numbers with magnitude one, meaning they are all unitary as opposed to special unitary. They can easily be made a member of SU(3) by multiplying each matrix by its determinate to the $-\frac{1}{3}$ power.

However, this step complicates the matrices and has no effect on the final results of the algorithm, so it is unnecessary.

Thus, a simpler four parameter matrix which can replace the eight parameter SU(3) may be

$$\mathbf{Swap}_1^{\alpha}\mathbf{Swap}_2^{\beta}\mathbf{Swap}_3^{\gamma}\mathbf{Swap}_4^{\phi} = \begin{pmatrix} \frac{1}{3}+\frac{1}{8}e^{-i\phi}(\frac{1}{3}+e^{i\alpha}-e^{i\beta}+e^{i(\alpha+\beta)}+e^{i\gamma}(1+3e^{i\alpha}+e^{i\beta}-e^{i(\alpha+\beta)})) \\ \frac{1}{3}+\frac{1}{8}e^{-i\phi}(\frac{1}{3}-e^{i\alpha}-e^{i\beta}-e^{i(\alpha+\beta)}+e^{i\gamma}(1-3e^{i\alpha}+e^{i\beta}+e^{i(\alpha+\beta)})) \\ \frac{1}{3}+\frac{1}{4}e^{-i\phi}(\frac{-1}{3}+e^{i\beta}-e^{i\gamma}(1+e^{i\beta})) \end{pmatrix} \tag{A.11}$$

$$\begin{matrix} \frac{1}{3}+\frac{1}{4}e^{-i\phi}(\frac{-1}{3}-e^{i\alpha}+e^{i\beta}-e^{i(\alpha+\beta)}) & \frac{1}{3}+\frac{1}{8}e^{-i\phi}(\frac{1}{3}+e^{i\alpha}-e^{i\beta}+e^{i(\alpha+\beta)}-e^{i\gamma}(1+3e^{i\alpha}+e^{i\beta}-e^{i(\alpha+\beta)})) \\ \frac{1}{3}+\frac{1}{4}e^{-i\phi}(\frac{-1}{3}+e^{i\alpha}+e^{i\beta}+e^{i(\alpha+\beta)}) & \frac{1}{3}-\frac{1}{8}e^{-i\phi}(\frac{-1}{3}+e^{i\alpha}+e^{i\beta}+e^{i(\alpha+\beta)}+e^{i\gamma}(1-3e^{i\alpha}+e^{i\beta}+e^{i(\alpha+\beta)})) \\ \frac{1}{3}+\frac{1}{2}e^{-i\phi}(\frac{1}{3}-e^{i\beta}) & \frac{1}{3}+\frac{1}{4}e^{-i\phi}(\frac{-1}{3}+e^{i\beta}+e^{i\gamma}(1+e^{i\beta})) \end{matrix}$$

where the matrix is written on two lines due to its length.

Appendix B. Analytic Solution of the Diffusion Equation

The diffusion equation

$$\frac{\partial \rho}{\partial t} = d\frac{\partial^2 \rho}{\partial x^2} \tag{B.1}$$

can be solved assuming a separable solution $\rho = X(x)T(t)$. Then the diffusion equation becomes

$$\frac{1}{d}\frac{T'}{T} = \frac{X''}{X} = -k^2 \tag{B.2}$$

after dividing (B.1) by dXT. Note that since the left hand side of the equation depends only on *t*, while the right hand side depends only on *x*, both sides must be equal to a constant to be true for all *t* and *x*. The constant is labeled $-k^2$ in anticipation of what follows.

By inspection, the solution to the equation $T'/T = -dk^2$ is

$$T(t) = Ae^{-dk^2 t} + B \tag{B.3}$$

where *A* and *B* are constants. The solution to $X''/X = -k^2$ is

$$X(x) = C\sin(kx) + D\cos(kx) + E \tag{B.4}$$

where *C*, *D*, and *E* are also constants. With circular boundary conditions,

$$X(0) = D + E = C\sin(kL) + D\cos(kL) + E = X(L) \tag{B.5}$$

which can only be true if $k = 2\pi m/L$ for integer *m*. Of course, the most general solution for $X(x)$ must be the sum of (B.4) over all possible *m*, so that

$$\rho = X(x)T(t) = B + \sum_{m=1}^{\infty} e^{d(2\pi m/L)^2 t}\left(C_m \sin\left(\frac{2\pi m x}{L}\right) + D_m \cos\left(\frac{2\pi m x}{L}\right)\right). \tag{B.6}$$

The initial condition is thus

$$\rho(x,0) = B + \sum_{m=1}^{\infty}\left(C_m \sin\left(\frac{2\pi m x}{L}\right) + D_m \cos\left(\frac{2\pi m x}{L}\right)\right). \tag{B.7}$$

To solve for the constants, we use identities

$$\begin{aligned}
&\int_0^L \cos\left(\frac{2\pi n x}{L}\right) dx = 0 \\
&\int_0^L \sin\left(\frac{2\pi n x}{L}\right) dx = 0 \\
&\int_0^L \cos\left(\frac{2\pi m x}{L}\right)\cos\left(\frac{2\pi n x}{L}\right) dx = \frac{L}{2}\delta_{mn} \\
&\int_0^L \sin\left(\frac{2\pi m x}{L}\right)\sin\left(\frac{2\pi n x}{L}\right) dx = \frac{L}{2}\delta_{mn} \ .
\end{aligned} \tag{B.8}$$

Then multiplying both sides of (B.7) by $\sin(2\pi n x / L)$, integrating from zero to L, and using identities two and four gives

$$C_n = \frac{2}{L}\int_0^L \rho(x,0)\sin\left(\frac{2\pi n x}{L}\right) dx. \tag{B.9}$$

Similar steps allow one to solve for the constants

$$\begin{aligned}
B &= \frac{1}{L}\int_0^L \rho(x,0)dx \\
D_m &= \frac{2}{L}\int_0^L \rho(x,0)\cos\left(\frac{2\pi m x}{L}\right) dx \ .
\end{aligned} \tag{B.10}$$

Bibliography

1. Jeffrey Yepez, Quantum Computation for Physical Modeling, *Computer Physics Communications.* **146** (3), 277 (2002)

2. Jeffrey Yepez, Lattice-Gas Quantum Computation, *International Journal of Modern Physics C*. **9** (8), 1587 (1998)

3. Jeffrey Yepez, Quantum lattice-gas model for computational fluid dynamics, *Phyical Review E*. **63**, 046702 (2001)

4. Jeffrey Yepez, A Quantum Lattice-Gas Model for Computational Fluid Dyanmics, *Physical Review E.* Submitted (1999), published without background material (2001)

5. Jeffrey Yepez, Quantum Computation of Fluid Dynamics, *Lecture Notes in Computer Science*. **1509**, 34 (1998)

6. Marco A. Pravia, Zhiying Chen, Jeffrey Yepez and David G. Cory, Towards a NMR Implementation of a Quantum Lattice Gas Algorithm, *Computer Physics Communications.* **146** (3), 339 (2001)

7. Marco A. Pravia, Zhiying Chen, Jeffrey Yepez, and David G. Cory, Experimental Demonstration of Quantum Lattice Gas Computation, *Quantum Information Processing*. **2**, 97 (2003)

8. G. P. Berman, A. A. Ezhov, D. I. Kamenev, and J. Yepez, Simulation of the diffusion equation on a type-II quantum computer, *Physics Review A.* **66**, 012310 (2002)

9. Jeffrey Yepez, Quantum Lattice-Gas Model for the Diffusion Equation, *International Journal of Modern Physics C.* **12**, 1285 (2001)

10. Jeffrey Yepez, Quantum lattice-gas model for the Burgers equation, *Journal of Statistical Physics.* **107** (1), 203 (2002)

11. Zhiying Chen, Jeffrey Yepez, and David G. Cory, Simulation of the Burgers equation by NMR quantum information processing, *Physical Review A*. to appear (2006) arXiv:quant-ph/0410198

12. Jeffrey Yepez, An efficient quantum algorithm for the one-dimensional Burgers equation, *Physical Review.* Submitted (2002) arXiv:quant-ph/0210092

13. Richard P. Feynman, Simulating physics with computers, *International Journal of Theoretical Physics*. **21** (6), 467 (1982)

14. Richard P. Feynman, Quantum mechanical computers, *Optics News*. **11** (2), 11 (1985)

15. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge (2000)

16. P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, *Proceedings, 35th Annual Symposium on Foundations of Computer Science*. IEEE Press (1994)

17. L. K. Grover, Quantum mechanics helps in searching for a needle in a haystack, *Physics Review Letters*. **79** (2), 325 (1997)

18. Peter J. Love, and Bruce M. Boghosian, Type-II Quantum Algorithms, *Physica A*. in Press (2006)

19. Maximilian Schlosshauer, Decoherence, the measurement problem, and interpretations of quantum mechanics, *Reviews of Modern Physics*. **76**, 1267 (2004)

20. L. D. Landau and E. M. Lifshitz, *Fluid Mechanics*, volume 6 of *Course of Theoretical Physics*. Pergamon Press, 2nd edition (1987)

21. J M Buick. Lattice Boltzmann Methods in Interfacial Wave Modeling, The University of Edinburgh (1997)

22. Edward R. Benton, and George W. Platzman, A table of solution of the one-dimensional Burgers equation, *Quarterly of Applied Mathematics*. **30**, 195 (1972)

23. Stephen Wolfram, Cellular automaton fluids 1: Basic theory, *Journal of Statistical Physics*. **45** (3/4), 471 (1986)

24. U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation, *Physics Review Letters*. **56**, 1505 (1986)

25. Cecile Appert, and Stephane Zaleski, Lattice gas with a liquid-gas transition. *Physical Review Letters*. **64**, 1 (1990)

26. Jeffrey Yepez, Lattice gas dynamics: Volume I viscous fluids, Technical report pl-tr-96-2122 (i), Air Force Research Laboratory (1996)

27. Donald A. Gurnett and Amitava Bhattacharjee, *Introduction to Plasma Physics*. Cambridge University Press (2005)

28. William Feller, *An Introduction to Probability Theory and Its Applications*. volume 1, Wiley (1966)

29. Eric W. Weisstein, *CRC Concise Encyclopedia on Mathematics*. Chapman & Hall/CRC (2003)

30. Gilbert Strang, *Linear Algebra and its Applications*. Thompson Learning, Inc (1988)

31. George Arfken and Hans Weber, *Mathematical Methods for Physicists*. Harcourt Academic Press, 5th edition (2001)

32. Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*. Society of Industrial and Applied Mathematics (1996)

Vita

Second Lieutenant James Scoville graduated from East View High School in Apple Valley, Minnesota. He entered undergraduate studies at the United States Air Force Academy where he was commissioned and graduated with honors with a Bachelor of Science degree in Physics in June 2004. Following this he entered the Graduate School of Engineering Physics at the Air Force Institute of Technology, Wright Patterson AFB. Upon graduation, he will be assigned to the Air Force Research Laboratory at Hanscom AFB, Massachusetts.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
23-03-2006	Master's Thesis	Jun 2005 – Mar 2006

4. TITLE AND SUBTITLE

Type II Quantum Computing Algorithm
For Computational Fluid Dynamics

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Scoville, James A., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GAP/ENP/06-17

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/VSBYA
Attn: Dr. Jeffrey Yepez
29 Randolph Rd.
Hanscom AFB, MA 01731 DSN: 377-5957

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

An algorithm is presented to simulate fluid dynamics on a three qubit type II quantum computer: a lattice of small quantum computers that communicate classical information. The algorithm presented is called a three qubit factorized quantum lattice gas algorithm. It is modeled after classical lattice gas algorithms which move virtual particles along an imaginary lattice and change the particles' momentums using collision rules when they meet at a lattice node. Instead of moving particles, the quantum algorithm presented here moves probabilities, which interact via a unitary collision operator. Probabilities are determined using ensemble measurement and are moved with classical communications channels. The lattice node spacing is defined to be a microscopic scale length. A mesoscopic governing equation for the lattice is derived for the most general three qubit collision operator which preserves particle number. In the continuum limit of the lattice, a governing macroscopic partial differential equation—the diffusion equation—is derived for a particular collision operator using a Chapman-Enskog expansion. A numerical simulation of the algorithm is carried out on a conventional desktop computer and compared to the analytic solution of the diffusion equation. The simulation agrees very well with the known solution.

15. SUBJECT TERMS
Quantum Theory, Quantum Computing, Quantum Computers, Type II Quantum Computers, NMR Quantum Computers, Fluid Dynamics, Computational Fluid Dynamics, Algorithms

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			David E. Weeks (ENP)
U	U	U	UU	94	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4561 (david.weeks@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18