Thermodynamic Effects of a Local Bell State Projection Interaction in a One-Dimensional Dynamic Spin System

A Senior Project

By

Nickolas H. Pilgram

Advisor, Dr. Thomas Gutierrez

Department of Physics, California Polytechnic University SLO

September 9, 2015

Approval Page

Title: Thermodynamic Effects of a Local Bell State Projection Interaction in a One-Dimensional Dynamic Spin System

Author: Nickolas H. Pilgram

Date Submitted: Month XX, 201X

Senior Project Advisor: Dr. Thomas Gutierrez

Signature

Date

Contents

List of Figures

1 Introduction

The development of quantum mechanics has forever changed and revolutionized the way in which we view and understand the world in which we live. However, the development of quantum mechanics launched a still ongoing debate about whether the resulting quantum theory is in fact a realistic description of what physically happens (orthodox viewpoint) or just an extremely accurate mathematical model for predicting and understanding the world and the statistically dependent measurements we obtain (realist viewpoint) [1]. To theoretically prove the plausibility of the realist view, Einstein, Podolsky, and Rosen proposed what is now known as the EPR (Einstein Podolsky Rosen) paradox, a paradox where information transfered between two entangled particles travels faster than the speed of light (violating relativity) [2]. The EPR paradox introduced the idea of entangled quantum particles, particles who's quantum states are correlated or entangled (the measured state of one particle is dependent on the measured state of the other). Through experimentation, it was shown that the assumption of locality used in the EPR paradox was in fact incorrect and therefore the information transfer between entangled particles did not violate any fundamental laws of nature [1]. Thus, the EPR paradox did not prove the realist view, allowing the debate of the true nature of quantum mechanics to continue to this day. However, the EPR paradox did introduce the extraordinary idea of quantum entanglement and the non-classical, non-local correlation of physical states.

One extraordinary result of non-local quantum entanglement is quantum teleportation, a non-local cut and paste operation on quantum states. The process of quantum teleportation allows quantum states to be transfered over distances without physically exchanging the quantum particles themselves. Quantum teleportation has been verified and successfully preformed in the laboratory. While quantum teleportation has very interesting applications in quantum information and quantum computing, it may affect the thermal properties of a system as well. The model we developed specifically investigates how the non-local effects of quantum teleportation effect and change the thermodynamical quantities of a dynamic one-dimensional spin system. The model is based on the Ising model of a ferromagnet and is investigated using the Metropolis algorithm.

2 Theory

2.1 Quantum Mechanics

2.1.1 EPR Pairs and the Bell States

Within our model, we are only concerned with two state quantum systems such as spin $\frac{1}{2}$ particles. Now consider a system where two spin $\frac{1}{2}$ particles are configured in the singlet state:

$$\frac{1}{\sqrt{2}}\left(|\uparrow_1\downarrow_2\rangle - |\downarrow_1\uparrow_2\rangle\right) \tag{1}$$

where each index refers to the individual particle. Now, if a measurement was preformed on particle 1 there would be a 50% chance that the result would be spin up and a 50% chance the result would be spin down. However, if particle 1 is measured in the spin up state, it is automatically known that particle 2 is in the spin down state, and vice versa. If particle 1 and 2 are separated by a large distance (several kilometers or several light years it does not matter) the same result will occur. In this separated situation, if particle 1 is measured in a spin up state, it is immediately known that particle 2 (several kilometers or several light years away) is in a spin down state [1]. This interaction over large distances is known as the EPR Paradox, named after Einstein, Podolsky, and Rosen who published their famous paper in 1935 [2].

Two particles in an EPR pair, such as the singlet state in Eq. 1, reside in what is known as an entangled state. An entangled two-particle state cannot be described by the product of two one-particle states [1]. Therefore, when two particles are in an entangled state, you cannot identify the specific state of one of the particles, as its state is dependent, or entangled, with the other particle's state. The previously mentioned EPR singlet state is one of four entangled states known as the Bell states or Bell Basis. The four Bell states are:

$$\begin{aligned}
\left|\Psi_{12}^{(-)}\right\rangle &= \frac{1}{\sqrt{2}}\left(|\uparrow_1\downarrow_2\rangle - |\downarrow_1\uparrow_2\rangle\right)\\
\left|\Psi_{12}^{(+)}\right\rangle &= \frac{1}{\sqrt{2}}\left(|\uparrow_1\downarrow_2\rangle + |\downarrow_1\uparrow_2\rangle\right)\\
\left|\Phi_{12}^{(-)}\right\rangle &= \frac{1}{\sqrt{2}}\left(|\uparrow_1\uparrow_2\rangle - |\downarrow_1\downarrow_2\rangle\right)\\
\left|\Phi_{12}^{(+)}\right\rangle &= \frac{1}{\sqrt{2}}\left(|\uparrow_1\uparrow_2\rangle + |\downarrow_1\downarrow_2\rangle\right)
\end{aligned} \tag{2}$$

It is important to note that the Bell states form a complete orthonormal basis for a set of two entangled particles [3].

2.1.2 Quantum Teleportation

One remarkable result of quantum entanglement is quantum teleportation, the act of transmitting quantum information (quantum states) over distances using classical channels. By utilizing the effects of quantum entanglement, quantum states can be transmitted over distances without physically transferring the particles themselves. In addition, quantum teleportation does not require the sender to have any knowledge of the quantum state to be transmitted or the location of the intended receiving observer [3]. The process of quantum teleportation is most easily illustrated using an example. Consider a system of three spin $\frac{1}{2}$ particles, where the first particle, particle 1, resides in an unknown state, and the other two particles, particle 2 and 3, are prepared in an entangled singlet state. Therefore, the unknown particle can be described by the wave function:

$$|\phi_1\rangle = a\,|\uparrow_1\rangle + b\,|\downarrow_1\rangle \tag{3}$$

where a and b are unknown coefficients, which can be complex, that satisfy the normalization condition $|a|^2 + |b|^2 = 1$. The other two particles are described by the wave function:

$$\left|\Psi_{23}^{(-)}\right\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow_2\downarrow_3\rangle - |\downarrow_2\uparrow_3\rangle\right) \tag{4}$$

Therefore, the total wave function of this three particle system is:

$$|\Psi_{123}\rangle = \frac{a}{\sqrt{2}}\left(|\uparrow_1\rangle\,|\uparrow_2\rangle\,|\downarrow_3\rangle - |\uparrow_1\rangle\,|\downarrow_2\rangle\,|\uparrow_3\rangle\right) + \frac{b}{\sqrt{2}}\left(|\downarrow_1\rangle\,|\uparrow_2\rangle\,|\downarrow_3\rangle - |\downarrow_1\rangle\,|\downarrow_2\rangle\,|\uparrow_3\rangle\right) \tag{5}$$

where a and b are the coefficients in Eq. 3[3] Now consider a situation where one observer, called Alice, wished to transfer the unknown quantum state of particle 1 to another observer, called Bob, in another location (the names of the observers are those used in the original paper [3]). Now this can be accomplished using the other two entangled particles, particle 2 and 3. Alice is given one of these entangled particles, say particle 2, and Bob is given the other, particle 3. Though there are nonclassical correlations between particles 2 and 3, and therefore Alice and Bob, the entire system still resides in the product state given by Eq. 5. Therefore, particles 2 and 3 contain no information about particle 1 [3]. Now Alice can entangle the two systems (particle 1 and the EPR pair) by preforming a measurement on particle 1 and particle 2 in the Bell operator basis given by Eq. 2. This measurement will project particle 1 and particle 2 into one of the four Bell states. The resulting state of particle 3 is dependent on the result of the bell basis measurement on particles 1 and 2, the Bell state in which particles 1 and 2 are projected. This is illustrated by rewriting the wave function of the total system in the Bell basis of particles 1 and 2:

$$\begin{aligned} |\Psi_{123}\rangle = \frac{1}{2}\left[\left|\Psi_{12}^{(-)}\right\rangle(-a|\uparrow_3\rangle - b|\downarrow_3\rangle) + \left|\Psi_{12}^{(+)}\right\rangle(-a|\uparrow_3\rangle + b|\downarrow_3\rangle)\right] + \\ \frac{1}{2}\left[\left|\Phi_{12}^{(-)}\right\rangle(a|\downarrow_3\rangle + b|\uparrow_3\rangle) + \left|\Phi_{12}^{(+)}\right\rangle(a|\downarrow_3\rangle - b|\uparrow_3\rangle)\right] \end{aligned} \tag{6}$$

where the bell states of particles 1 and 2 are given by Eq. 2 [3]. Eq. 6 shows that particles 1 and 2 have an equal probability (25%) of being projected into each of the four Bell states following the Bell basis measurement. Eq. 6 also illustrates that the resulting state of particle 3 is dependent on the Bell state into which particles 1 and 2 are projected. For example, if particles one and two are projected into the $\left|\Psi_{12}^{(-)}\right\rangle$ state then particle 3 will reside in the state described by $|\phi_3\rangle = -a|\uparrow_3\rangle - b|\downarrow_3\rangle$. Therefore, the four possible resulting states for particle 3 are given by:

$$\begin{aligned} |\phi_3\rangle_1 &= -a|\uparrow_3\rangle - b|\downarrow_3\rangle \\ |\phi_3\rangle_2 &= -a|\uparrow_3\rangle + b|\downarrow_3\rangle \\ |\phi_3\rangle_3 &= a|\downarrow_3\rangle + b|\uparrow_3\rangle \\ |\phi_3\rangle_4 &= a|\downarrow_3\rangle - b|\uparrow_3\rangle \end{aligned} \tag{7}$$

where the resulting state corresponds directly to the result of the Bell state measurement as seen in Eq. 6. These states can also be expressed as 180° rotations of the original unknown state $|\phi_1\rangle$ about the x, y, and z axes by applying the appropriate unitary operators to $|\phi_1\rangle$. Thus, the original unknown state of particle 1 and the resulting states of particle 3 can also be expressed in matrix notation as:

$$\begin{aligned} |\phi_1\rangle &= \begin{pmatrix} a \\ b \end{pmatrix} \\ |\phi_3\rangle_1 &= -|\phi_1\rangle \\ |\phi_3\rangle_2 &= \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}|\phi_1\rangle \\ |\phi_3\rangle_3 &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}|\phi_1\rangle \\ |\phi_3\rangle_4 &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}|\phi_1\rangle \end{aligned} \tag{8}$$

Therefore, the final state of particle 3 is directly related to the initial unknown state which Alice wished to transfer to Bob. Since the final state of particle 3 is related to the initial unknown state of particle 1, all Bob must do to reconstruct the initial unknown state is apply the correct unitary operator (if any at all), indicated by Eq. 8, to particle 3 [3]. However, the final state of particle 3, and therefore the unitary operator that Bob needs to apply, is dependent on the result of Alice's measurement of particles 1 and 2. Therefore, for Bob to know what unitary operator to apply to particle 3, Alice must relay the result of her measurement to Bob through a classical channel [3]. (The information relayed by the classical channel can not travel faster than the speed of light so quantum teleportation does not violate special relativity.) It is important to note that 25% of the time particle 3's state will be $|\phi_3\rangle_1$ which is only a 180° shift of $|\phi_1\rangle$ and therefore the same as the initial unknown state. Thus, there is a 25% chance that a quantum teleportation will occur without Bob applying a unitary operator to particle 3. The interaction in our model is the same interaction Alice must apply to perform a Bell basis measurement of particles 1 and 2. Therefore, quantum teleportations can occur within our model.

2.1.3 Pair Swapping

Pair swapping is a special case of quantum teleportation where the two particles projected into a Bell state are both part of separate EPR pairs. When two entangled particles, each in a separate EPR pair, are projected into a Bell state, the particles with which they were previously entangled are also projected into a Bell state. This results in the particles swapping entanglement partners. To illustrate this consider the system of four particles consisting of two separate EPR pairs described by:

$$|\Psi_{1234}\rangle = \left|\Phi_{12}^{(+)}\right\rangle \otimes \left|\Phi_{34}^{(+)}\right\rangle = |{\uparrow_1},{\uparrow_2},{\uparrow_3},{\uparrow_4}\rangle + |{\downarrow_1},{\downarrow_2},{\uparrow_3},{\uparrow_4}\rangle + |{\uparrow_1},{\uparrow_2},{\downarrow_3},{\downarrow_4}\rangle + |{\downarrow_1},{\downarrow_2},{\downarrow_3},{\downarrow_4}\rangle \quad (9)$$

where particles 1 and 2 are in an entangled Bell state and particles 3 and 4 are in a separate entangled Bell state. Now, if a Bell basis measurement is performed on particles 2 and 3, projecting particles 2 and 3 into a Bell state, the resulting joint state of the four particles will be one of the following four:

$$\begin{aligned}
|\Psi_{1234}\rangle &= \left|\Phi_{23}^{(+)}\right\rangle \otimes \left|\Phi_{14}^{(+)}\right\rangle = (|{\uparrow_2},{\uparrow_3}\rangle + |{\downarrow_2},{\downarrow_3}\rangle) \otimes (|{\uparrow_1},{\uparrow_4}\rangle + |{\downarrow_1},{\downarrow_4}\rangle) \\
|\Psi_{1234}\rangle &= \left|\Phi_{23}^{(-)}\right\rangle \otimes \left|\Phi_{14}^{(-)}\right\rangle = (|{\uparrow_2},{\uparrow_3}\rangle - |{\downarrow_2},{\downarrow_3}\rangle) \otimes (|{\uparrow_1},{\uparrow_4}\rangle - |{\downarrow_1},{\downarrow_4}\rangle) \\
|\Psi_{1234}\rangle &= \left|\Psi_{23}^{(+)}\right\rangle \otimes \left|\Psi_{14}^{(+)}\right\rangle = (|{\uparrow_2},{\downarrow_3}\rangle + |{\downarrow_2},{\uparrow_3}\rangle) \otimes (|{\uparrow_1},{\downarrow_4}\rangle + |{\downarrow_1},{\uparrow_4}\rangle) \\
|\Psi_{1234}\rangle &= \left|\Psi_{23}^{(-)}\right\rangle \otimes \left|\Psi_{14}^{(-)}\right\rangle = (|{\uparrow_2},{\downarrow_3}\rangle - |{\downarrow_2},{\uparrow_3}\rangle) \otimes (|{\uparrow_1},{\downarrow_4}\rangle - |{\downarrow_1},{\uparrow_4}\rangle) \text{ [4].}
\end{aligned} \quad (10)$$

There is an equal probability (25%) for the system to be projected into any of the four states given in Eq. 10. Eq. 10 also reveals that the Bell state into which particles 2 and 3 are projected is the same

Bell state which particles 1 and 4 will also be projected. It can be shown that the same result given in Eq. 10 will result no matter which initial Bell states particles 1 and 2 and particles 3 and 4 initially reside [4]. The Bell state interaction incorporated within our model can result is pair swapping in addition to quantum teleportation.

2.1.4 Quantum Decoherence

This section provides a very brief, non-technical, non-mathematical description of quantum decoherence. The topic of quantum decoherence is extremely complicated and includes aspects of density matrix theory, which is beyond the scope of the material integrated within our model. This introduction of the ideas of quantum decoherence is to provide a basic understanding of the subject so that the basic method in which decoherence is incorporated within the model can be understood. A technical description of quantum decoherence is given in [5].

Quantum Decoherence is the process by which a quantum state loses its quantum coherence and devolves into a semi-classical or classical state. This loss of quantum information is necessary if any observer is to obtain a classical (and therefore useful and realistic) result when observing the system. The dispersion of this quantum information occurs through and interaction between the quantum system in question and the external environment (the environment is also a quantum system). Quantum correlations between the systems allow the quantum information to be dispersed throughout degrees of freedom external to the system (not available to the observer) [5]. These external degrees of freedom are generally referred to as the environment. This effective loss of information is desirable since it increases the entropy of the system [5]. This loss of quantum information to the environment causes the quantum state to devolve, or decohere, into a semi-classical or classical state. The manner in which a detector measures, or observes, a specific property of a quantum system is much the same, except now there is an interaction between the quantum state, the environment, and the detector (the detector must also be considered as a quantum system). The interaction between the quantum state detector system and the environment decoheres the quantum state detector system into the specific basis in which the detector is to measure the system. This specific desired basis of the detector is known as the pointer basis of the detector [5]. The quantum state detector interaction then further decoheres the quantum state into the classical state that is measured [5]. Our model incorporates a very simplified model of quantum decoherence, in which the environment interacts with the particles in the system in a way which causes the entangled Bell states to decohere into the classical spin up or spin down states.

2.2 Thermal Physics and Statistical Mechanics

2.2.1 Boltzmann Statistics: Boltzmann Factors and the Partition Function

When analyzing a thermodynamical system (especially those containing a large number of particles), it is often (if not always) more convenient to analyze the system through statistical means. An excellent way in which to analyze a system statistically is through Boltzmann statistics. Boltzmann statistics are especially convenient for quantum systems as they are concerned with the energy states of the system, specifically occupation. Boltzmann statistics provide a means of determining the probability of finding a system in a specific state, whether the system is a single atom or a gas. However, first it is convenient to describe the ratio of the probabilities of the system occupying two separate energy states. Boltzmann statistics states that this ratio of probabilities can be expressed as a ratio of two exponential factors dependent of each states energy and the temperature of the system. Thus, if the two states considered are represented by s_1 and s_2, the ratio of their occupation probabilities can be expressed as:

$$\frac{P\left(s_2\right)}{P\left(s_1\right)} = \frac{e^{-E(s_2)/k_BT}}{e^{-E(s_1)/k_BT}} = e^{-[E(s_2)-E(s_1)]/k_BT} = e^{-\Delta E/k_BT} \tag{11}$$

where $E\left(s_1\right)$ and $E\left(s_2\right)$ are the energies of the states s_1 and s_2, k_B is Boltzmann's constant, T is the temperature of the system, and ΔE is the energy difference between the two states [6]. These exponential factors are known as Boltzmann factors. Thus, the Boltzmann factor for a state s is defined as:

$$\text{Boltzmann Factor} = e^{-E(s)/k_BT} \tag{12}$$

In addition to knowing the ratio of occupation probabilities, it is also important to know the occupation probability of each state itself. In order to use Boltzmann factors to determine the probability that the system will be in a specific energy state, a new statistical quantity must be defined, the partition function. The partition function of a system is defined as the sum of all the Boltzmann factors of the system. Thus, the partition function is a sum of the Boltzmann factors over all the states of the system:

$$Z = \sum_s e^{-E(s)/k_BT} \tag{13}$$

where Z denotes the partition function and s is the energy state [6]. With the partition function defined, the probability of the system residing in a specific state s is given by:

$$P\left(s\right) = \frac{1}{Z} e^{-E(s)/k_B T} \tag{14}$$

where Z is the partition function of the system given by Eq. 13. Taking Eq. 14 and the fact that the total probability of finding the system in a particular state into account, results in the following:

$$\sum_s P\left(s\right) = \sum_s \frac{1}{Z} e^{-E(s)/k_B T} = 1 \text{ [6]}. \tag{15}$$

The partition function is an important statistical quantity since it provides far more insight into the thermodynamics of the system than just the probability that the system resides in a specific state. One of these thermodynamical quantities is the average energy of the system. The average energy of the system can be expressed as,

$$\overline{E} = \frac{1}{Z} \sum_s E\left(s\right) e^{-\beta E(s)} \tag{16}$$

where the quantity $\beta = \frac{1}{k_B T}$ and is used for convenience. Now using β in place of $\frac{1}{k_B T}$ the partition function (defined in Eq. 13) can be expressed as:

$$Z = \sum_s e^{-\beta E(s)}. \tag{17}$$

By utilizing Eq. 16 and Eq. 17 it can be shown that the average energy is proportional to a derivative of the partition function with respect to β. From this, it can be shown that the average energy of the system is given by:

$$\overline{E} = -\frac{1}{Z} \frac{\partial Z}{\partial \beta} = -\frac{\partial}{\partial \beta} \ln Z \tag{18}$$

where $\ln Z$ is the natural logarithm of the partition function [6].

The average energy of the system provides additional thermal information about the system. One important quantity that can be determined from the average energy of the system is the heat capacity at constant volume. The heat capacity at constant volume is given by the derivative of the average energy with respect to temperature,

$$C_V = \frac{\partial \overline{E}}{\partial T} \text{ [6]}. \tag{19}$$

The heat capacity provides additional insights into the thermodynamics of the system including the location nature of phase transitions.

2.2.2 The Ising Model of a Ferromagnet

The Ising model of a ferromagnet is a simplified (yet still complicated) model used to study the behavior of a ferromagnet. It is named after Ernst Ising who studied it in the 1920's [6]. A ferromagnet is composed of magnetic dipoles which have a tendency to align parallel to each other. In a ferromagnet the parallel alignment of dipoles results in a nonzero net magnetization of the ferromagnet. However, at higher temperatures, random fluctuations of the dipoles causes some to align in an antiparallel manner, decreasing the net magnetization. All ferromagnets have a critical temperature, the Curie temperature, at which their net magnetization becomes zero [6]. The Ising model of a ferromagnet seeks to model this behavior.

The Ising model of a ferromagnet operates upon two simplifying assumptions about the way in which the dipoles behave. First, the tendency of neighboring dipoles to align is accounted for, but any long-range effects between the dipoles are neglected. Secondly, it is assumed that the dipoles have a preferred axis of magnetization, thus each dipole can only point parallel or antiparallel to this axis. Therefore, the state of each dipole is either pointing up, aligned with the preferred axis (denoted by +1) or pointing down, antialigned with the preferred axis (denoted by -1). Thus, if s_i is the state of the ith dipole, s_i=1 if the dipole is pointing up and s_i=-1 if the dipole is pointing down. The energy due to the interaction between two neighboring dipoles is given by $-\epsilon s_i s_j$, where dipoles i and j are neighbors. Thus, the energy due to two neighboring parallel dipoles is $-\epsilon$, while the energy due to two neighboring antiparallel dipoles is $+\epsilon$ [6]. The total energy of the system resulting from all the dipole nearest-neighbor interactions is

$$U = -\epsilon \sum_{<i,j>} s_i s_j \tag{20}$$

where the sum over $< i, j >$ is carried over all neighboring pairs of dipoles in the system [6]. It is also convenient to determine the partition function of the system, which is given by:

$$Z = \sum_{[s_i]} e^{-\beta U} \tag{21}$$

where the sum is over all the possible sets of dipole alignments [6]. If the system contains N dipoles each with two possible alignments, then the partition function given by Eq. 21 will contain 2^N terms.

Directly calculating the partition function for the Ising model can become quite complicated if not impossible. However, for the one-dimensional case the partition function can be calculated exactly. In the one-dimensional case of the Ising model, the dipoles are arranged as a simple string (one-dimensional lattice) so that each dipole only has two nearest-neighbors. In this one-dimensional case the total energy of the system is given by

$$U = -\epsilon \left(s_1 s_2 + s_2 s_3 + s_3 s_4 + ... + s_{N-1} s_N \right) \tag{22}$$

where N is the number of dipoles in the system. With N particles in the system, the partition function, using Eq. 21, can be written as

$$Z = \sum_{s_1} \sum_{s_2} \sum_{s_3} ... \sum_{s_{n-1}} \sum_{s_N} e^{\beta \epsilon s_1 s_2} e^{\beta \epsilon s_2 s_3} ... e^{\beta \epsilon s_{N-2} s_{N-1}} e^{\beta \epsilon s_{N-1} s_N} \tag{23}$$

where each sum is executed over the possible values of s_i which are +1 and -1 [6]. Now the last sum over s_N can be expressed as

$$\sum_{s_N} e^{\beta \epsilon s_{N-1} s_N} = e^{\beta \epsilon} + e^{-\beta \epsilon} = 2 \cosh \beta \epsilon. \tag{24}$$

The result of Eq. 24 holds regardless of the state of the s_{N-1} dipole (whether s_{N-1} is +1 or -1) [6]. The next sum, over s_{N-1}, can be calculated in the same way and will yield the same result. This process can be followed over all sums excluding the first resulting in $N-1$ factors of $2 \cosh \beta \epsilon$. The first sum, over s_1 simply results in a 2. Combining these results yields the partition function of the system, which is

$$Z = 2^N \left(\cosh \beta \epsilon \right)^{N-1} \approx \left(2 \cosh \beta \epsilon \right)^N \tag{25}$$

where the final simplifying approximation is valid if N is large [6]. With the partition function the average energy of the system can be calculated. Using Eq. 18, the average energy of the system is

$$\overline{U} = -\frac{\partial}{\partial \beta} \ln Z = -N \epsilon \tanh \beta \epsilon \tag{26}$$

where Z is given by Eq. 25 [6]. As Eq. 26 indicates, $\overline{U} \to -N\epsilon$ as $T \to 0$ and $\overline{U} \to 0$ as $T \to \infty$. Thus, the dipoles are aligned (completely parallel) at T=0 and become less aligned as temperature increases, reaching a randomly aligned state at high temperature. Though the one-dimensional model produces exact results, $\overline{U}$ (Eq. 26) is perfectly smooth which indicates that there is no nonzero critical temperature at which the system makes an abrupt transition. This is inconsistent with actual ferromagnets as they do have a critical temperature [6].

The next step would be to evaluate the Ising model in higher dimensions. However, though a closed form result for the two dimensional case has been obtained, it is extremely complicated and will not be discussed. It is important to note that this closed form solution in two dimensions does have a critical temperature. No closed form solution has been obtained in three dimensions [6]. Another method (though somewhat crude) of investigating the Ising model in higher dimensions is the mean field approximation. The mean field approximation is not very accurate but does give a good insight into the behavior of a ferromagnet.

When considering a single dipole, s_i within the lattice, let n represent the number of nearest neighbors to that dipole. Therefore, n=2 in one dimension, n=4 in two dimensions, n=6 in a three dimensional simple cubic lattice, etc. Thus, the energy that results from the s_i dipole interacting with its neighbors is

$$E_i = -\epsilon \sum_{neighbors} s_{neighbor} = \begin{cases} -\epsilon n\overline{s} & : s_i = 1 \text{ (points up)} \\ +\epsilon n\overline{s} & : s_i = -1 \text{ (points down)} \end{cases} \tag{27}$$

where $\overline{s}$ is the average alignment of the nearest-neighbors [6]. The partition function for this dipole is

$$Z_i = e^{\beta\epsilon n\overline{s}} + e^{-\beta\epsilon n\overline{s}} = 2\cosh\beta\epsilon n\overline{s}. \tag{28}$$

Using the partition function the average expected value of the s_i dipole's spin alignment is

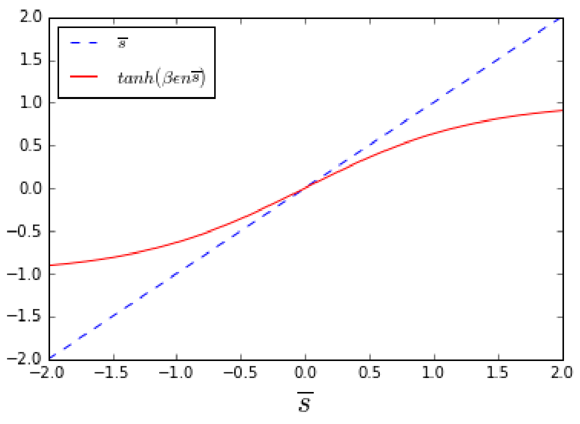
$$\overline{s_i} = \frac{1}{Z_i}\left[(1)\,e^{\beta\epsilon n\overline{s}} + (-1)\,e^{-\beta\epsilon n\overline{s}}\right] = \frac{2\sinh\beta\epsilon n\overline{s}}{2\cosh\beta\epsilon n\overline{s}} = \tanh\beta\epsilon n\overline{s} \tag{29}$$

Now, this is where the mean field approximation is utilized. The mean field approximation is $\overline{s_i} = \overline{s}$, that the dipole alignments in all neighborhoods within the ferromagnet are typical and that they do not fluctuate away from this thermal average [6]. Applying the mean field approximation results in a transcendental equation:

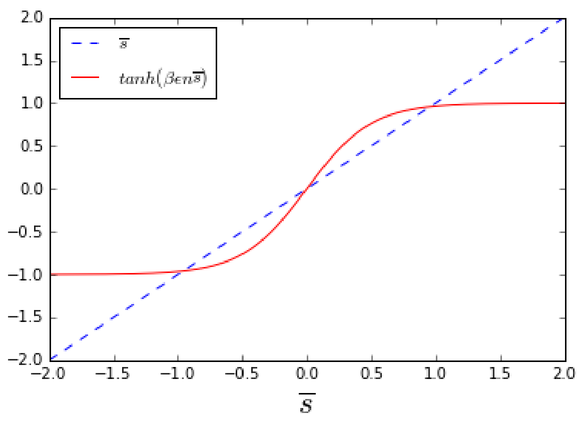
$$\overline{s} = \tanh\beta\epsilon n\overline{s} \tag{30}$$

where $\overline{s}$ is now the average dipole alignment for the entire system [6].

Eq. 30 can not be solved analytically, however it can be solved graphically. The graphical solution to Eq. 30 is shown in Figure 1. As can be seen, for high temperatures ($\beta\epsilon n < 1$) there is only one solution



(a) $\beta\epsilon n < 1$



(b) $\beta\epsilon n > 1$

Figure 1: The graphical solution to Eq. 30. When $\beta\epsilon n\overline{s}$ is less than 1 there is only one stable solution at $\overline{s}$=0. When $\beta\epsilon n\overline{s}$ is greater than 1, there are three solutions, one unstable at $\overline{s}$=0 and two stable nontrivial, nonzero solutions [6].

of $\overline{s}$=0, which means that the system has no net magnetization at high temperatures as expected. For low temperatures ($\beta\epsilon n > 1$), there are three solutions, an unstable solution of $\overline{s}$=0 (of which the system can easily be perturbed from), and two stable nonzero solutions. These two stable solutions correspond to a net magnetization of the system at lower temperatures, one solution corresponds to the dipoles pointing up and the other corresponds to the dipoles pointing down (the system favors either equally).

The graphical solutions also specifies a critical temperature at which the system moves from one to three solutions. The critical temperature occurs when $\beta\epsilon n$=1. Thus the critical temperature, T_C is

$$T_C = \frac{n\epsilon}{k_B} \text{ [6].} \tag{31}$$

The mean field approximation is not terribly accurate, however it does provide a fairly consistent quantitative description of how the system transitions states when moved from high to low temperatures, or vice versa.

There is another method in which the Ising model of a ferromagnet can be investigated, Monte Carlo simulation. In this procedure, a random sample of possible states is generated and these random sample states are used to compute energy, magnetization, and other thermodynamical properties [6]. However, since there are an immense number of possible states it is better to let Boltzmann factors act as a guide in choosing the random states. A Monte Carlo algorithm utilizing importance sampling (the Boltzmann factors help choose states of importance) is known as the Metropolis algorithm [6]. This method of investigating the Ising model is the method on which our model is based. The Metropolis algorithm is described in more detail in section 3.1. The baseline interaction for our model is dynamic Ising model investigated under this Metropolis algorithm. The baseline interaction is discussed in section 3.5.1.

3 Ising Inspired Model of One-Dimensional Dynamic Spin System Operating Under a Local Bell State Interaction

To investigate the thermal properties of a one-dimensional dynamic spin system operating under a Bell-state projection interaction, we created an Ising inspired model in the Ipython notebook. The model which we created allows for each particle in the system to be individually identified, the particles to dynamically diffuse in a temperature dependent matter, Bell state projection interactions to occur on individual pairs of particles within the system, and decoherence of the quantum Bell states to more classical states to occur. The way in which the model operates is inspired by the Ising Model of a Ferromagnet. Though the model is based on the Ising model, it explores an entirely different interaction and therefore stands in its own right. This section gives a detailed description of the model, its properties, and the major theoretical ideals. The methodologies and execution of the model are also described. Blocks of pseudocode have been included to better illustrate how the model is executed and to allow easy reproduction of the model.

3.1 Model's Approach: Metropolis Algorithm

In order to examine the thermal effects of a local bell state projection, we decided to employ a random sampling method similar to that which can be used to investigate the Ising model. This random sampling method is known as the Metropolis algorithm or Monte Carlo summation with importance sampling [6]. Under this algorithm, random system states are generated based on thermal importance. Boltzmann statistics guide the random state generation allowing the more likely thermal state configurations to be generated. These generated states can then be used to calculate average energy and other thermodynamic quantities. A general outline of how the Metropolis algorithm is executed is as follows: Start with any random system state. Then choose a random particle/dipole and consider how executing the desired interaction (Bell state projection or Ising spin flip for example) on that particle will change the systems energy. If executing the interaction will lower the energy or the energy will remain unchanged, then execute it. However, if executing the interaction will raise the energy, randomly decide if the interaction will be executed, were the random decision is weighted by the Boltzmann probability of:

$$e^{\frac{-\Delta U}{kt}} \tag{32}$$

where ΔU is the change in energy, k is Boltzmann's constant, and T is the temperature. Then choose another random particle/dipole and repeat [6]. Repeated execution of the Metropolis algorithm allows the most likely states to be generated. Taking the average of a thermodynamical quantity (e.g. energy) over all iterations at a specific temperature results in the value of the thermodynamical quantity at that temperature.

3.2 The Model Basics: Defining the Particles and the System

3.2.1 The Particles

At the core of the model are the quantum particles of which the system is composed. Specifically spin $\frac{1}{2}$ particles. Initially, all the particles reside in either the spin up or spin down state, none will reside in a superposition of the two. Now, if two of the particles interact under the local Bell state projection interaction, they will be projected into one of the four entangled Bell states. Therefore, the particles will be allowed to reside in one of three possible states, spin up (denoted by +1), spin down (denoted by -1), or one of the four bell states (denoted by 0). The choice of the numeric values denoting the states is for energy calculation purposes, which is discussed in section 3.4.1.

In addition to the state of the particle, there are five other qualities, or properties, of each particle that must be defined: index, position, pairing, Bell state, and Bell state time. The index of the particle is simply a number by which each particle can be individually identified. Position simply just indicates the specific position of the particle within the system at any specific time. Paring identifies the particle's EPR partner if the particle is part of an EPR pair. The paring information becomes important when a particle that is part of an EPR pair undergoes a Bell state projection allowing a quantum teleportation or pair swapping to occur. The Bell state indicates which Bell state, or lack thereof, the particle is in. For simplicity, each Bell state is denoted by an integer, 1 corresponds to $\left|\Psi^{(-)}\right\rangle$, 2 corresponds to $\left|\Psi^{(+)}\right\rangle$, 3 corresponds to $\left|\Phi^{(-)}\right\rangle$, and 4 corresponds to $\left|\Phi^{(+)}\right\rangle$. Lastly, the Bell state time indicates the time that a particle has been entangled in a Bell state. The time which a particle has remained in a Bell state is needed to apply decoherence effects.

Since each particle is represented by six pieces of information, it is extremely difficult to represent each particle as information in a list or array. Instead, we elected to represent each particle as an object. Defining each particle as an object presents several advantages. First, the information of each particle (index, position, paring, state, Bell state, and Bell state time) can be defined as properties of the particle which can easily be accessed. An example of this advantage would be calling a particle's pairing information. Secondly, since defining the particles as objects with specific properties wraps all the particle's information within one object, the one dimensional system can now be easily represented as a list of these particle objects. Lastly, the properties and methods (functions) of an object can easily be changed or expanded allowing new information and interactions (such as particle phase and phase shifts) to easily be integrated later. A pseudo-code algorithm for generating the particle objects is given below.

Algorithm 1 Defining Particle Objects

```
procedure CREATE CLASS PARTICLE(inherit from object)
    function CLASS CONSTRUCTOR(self, index, position, state, pairing=None, bell_state=None,
bell_state_time=None)
        self.index=index                          ▷ index given by integer, unique for each particle
        self.position=position                                        ▷ position given by integer
        self.state=state                                              ▷ state given by +1, -1, or 0
        self.paring=pairing              ▷ pairing is set to paired particle object, default of None
        self.bell_state=bell_state          ▷ bell state indicated by 1, 2, 3, or 4, default of None
        self.bell_state_time=bell_state_time   ▷ bell state time given as integer, default of None
    end function
end procedure
```

With a new particle object class defined, particle objects can now be created and manipulated. Some pseudo-code for generating, manipulating, and calling particle information would look something like this:

Algorithm 2 Generating and Manipulating Particles

```
Particle1=PARTICLE(1,5,1)      ▷ Creates Particle1 with index=1, position=5, and state=+1 (spin up)
Particle2=PARTICLE(2,9,-1)     ▷ Creates Particle2 with index=2, position=9, and state=-1 (spin down)
PRINT(Particle1.index)
1
PRINT(particle2.state)
-1
Particle1.paring=Particle2                              ▷ Pairs Particle1 to Particle2
Particle1.state=0                                ▷ Sets Particle1's state to 0 (bell state)
Particle1.bell_state=2                      ▷ Sets Particle1's bell state to 2 ($|\Psi^{(+)}\rangle$)
Particle2.paring=Particle1                              ▷ Pairs Particle2 to Particle1
Particle2.state=0                                ▷ Sets Particle2's state to 0 (bell state)
Particle2.bell_state=2                      ▷ Sets Particle2's bell state to 2 ($|\Psi^{(+)}\rangle$)
PRINT(Particle1.pairing.position)          ▷ Prints position of particle paired to Particle1
9
PRINT(Particle2.pairing.index)                ▷ Print index of particle paired to Particle2
```

3.2.2 The System: A One-Dimensional Gas

However, the model is not just concerned with spin $\frac{1}{2}$ particles, but a one dimensional system composed of them. The system is modeled by a one-dimensional gas organized as a discrete lattice with periodic boundary conditions. The periodic boundary conditions treat the gas as if it were bent into a circle, so that the first space in the lattice is adjacent to the last. Each point in the lattice is occupied by a particle. Only one particle is allowed to occupy any one position (space) in the lattice at any given time. Therefore, since particles can't occupy the same position, all positions in the gas will be occupied at all times.

Since the particles are defined as objects, self-containing all their information, the one dimensional gas can simply be defined as a one dimensional list. However, in the interest of optimizing and allowing the model to be flexible and easily expanded, the gas was defined as a new class inheriting from the list class. Defining the gas as a new class offered several advantages. First, this new Gas class is specifically defined as a list of individual and separate particle objects. Secondly, defining a new class allowed us to give the gas additional properties, such as temperature, and allows these additional properties to be easily changed or expanded. Lastly, the particle movement algorithm, energy calculation algorithm, decoherence algorithm, and other gas property algorithms can be defined as methods (functions) of this new class. Some pseudo-code for defining this new gas class is shown below.

Algorithm 3 Defining Gas Class

```
procedure CREATE CLASS GAS(inherits from list)
    function CLASS CONSTRUCTOR(self, length, Temperature)
        self=LIST(length)
        for i in self
            state=RANDOM NUMBER(-1 or 1)
            self[i]=PARTICLE(i,i,state)
        end for
        self.Temperature=Temperature                ▷ Defines temperature of gas
    end function
    function INDEXES(self)                          ▷ Returns list of particle indexes
        for p in self return p.index
        end for
    end function
    function POSITIONS(self)                        ▷ Returns list of particle Positions
        for p in self return p.positions
        end for
    end function
    function STATES(self)                           ▷ Returns list of particle states
        for p in self return p.state
        end for
    end function
    function PAIRINGS(self)                         ▷ Returns list of particle pairings
        for p in self return p.pairing
        end for
    end function
    function BELL STATES(self)                      ▷ Returns list of particle bell states
        for p in self return p.bell_state
        end for
    end function
    function BELL STATE TIMES(self)                 ▷ Returns list of particle bell state times
        for p in self return p.bell_state_times
        end for
    end function
end procedure
```

3.3 Movement Algorithm: Temperature Dependent Particle Diffusion

An important part of our model is the dynamic movement or diffusion of the particles in the gas. It is integral that the particles move within the gas in a manner consistent with temperature and natural gas diffusion. Therefore, at higher temperatures the particles must have higher average velocities and will thus move further on average. To accomplish this we generate the velocity of each particle with a Gaussian random number generator. The mean of this Gaussian random number generator is produced by the function:

$$\mu = v_{max}\left(1 - e^{\frac{-T}{T_m}}\right) \tag{33}$$

where $vmax$ is the length of the one dimensional gas, T is the temperature of the gas, and T_m is the temperature at which the mean will reach $(1 - \frac{1}{e})v_{max}$. To also allow for the standard deviation of the Gaussian random number generator to increase with temperature, the standard deviation is given by:

$$\sigma = \frac{1}{2}\mu \tag{34}$$

The random diffusion of particles is executed as follows. First, a new empty gas is created. Because the particle velocities are random, several particles can try to move to the same position in the gas at once, which is not allowed. Therefore, each particle must be moved one at a time. Thus, to actually move the particles, the original gas is iterated through, from one end to the other. Each particle in the original gas is given a random velocity, generated by the Gaussian random number generator, and a random direction, 50% chance to move left or right. Then, considering periodic boundary conditions (if the particles velocity and direction would move it passed the end of the gas, it moves back around to the other side), the position in the gas which the particle is to move is examined. If the position is empty, the particle is moved there (the position in the new gas is filled by the particle). If the position is filled, then the next closest position is examined (as if the particle now had a velocity one less than the one randomly assigned). This process is continued, moving one step closer to the particles original position, until the particle is moved to a new position. If all positions between the original desired movement position and the particle's original position are full, the particle's original position is examined. If another particle has not moved into the particle's original position, the particle stays there, not moving for this iteration. However, if a another particle has moved into the particle's original position, the gas is searched for an empty position and the particle is moved there instead. Finally, once all particles are moved, the original gas is reset so that it is the same as the new gas. The pseudo-code describing the movement algorithm is quite long. Therefore, in interest of preserving continuity in the text, the pseudo-code for the movement algorithm is given in Appendix A.1. The movement function was written as a method (function) of the gas class.

3.4 Thermodynamical Quantities of the Gas: Energy, Net Spin, Entanglement Density, and Correlation Function

3.4.1 Energy

To determine how the local bell state projection will effect the thermodynamical properties of the gas, energy and other thermodynamical properties must be defined and calculated. As for the energy of the gas, it depends on the states of the particles within it, specifically how the states of adjacent particles are arranged. In a similar manner to the Ising Model, we defined adjacent particles in aligned states (both spin up or spin down) to be favorable and adjacent particles in anti-aligned states (one spin up and the other spin down) to be unfavorable. Therefore, adjacent aligned states will lower the energy by one unit and adjacent anti-aligned states will raise the energy by one unit. This energy model only accounts for particles in either spin up or spin down states however, but particles can also exist in Bell states. Since particles in Bell states exist in a perfect 50/50 superposition of a spin up and spin down states, a particle in a Bell state is neither aligned or anti-aligned with an adjacent particle, no matter if the particle is in a spin up, spin down, or Bell state. Therefore, we defined a particle in a bell state to not contribute to the energy of the system (has an energy contribution of 0). Since the energy of the system is defined by the alignment or anti-alignment of adjacent particles, we chose to represent the states as follows: spin up as +1, spin down as -1, and any bell state as 0. Taking this into account, the energy of the entire system can be defined as:

$$U = -\epsilon \sum_{i=1}^{N} s_i s_{i+1} \tag{35}$$

where ϵ is the unit of energy (ϵ=1 for simplicity), N is the length of the gas, and s_i is the state of the ith particle. It is important to note that because of periodic boundry conditions the last term in the sum, $s_N s_{N+1} = s_N s_0$. Pseudo-code utilizing Eq. 35 to calculate the energy of the gas is shown below. We wrote the energy calculation as a method (function) of the gas class.

Algorithm 4 Energy Calculation

```
function ENERGY(self)                                        ▷ self is the gas
    E=0                                                      ▷ Start energy sum at zero
    for i in self do
        E=0
        if i=LENGTH(self) then        ▷ takes periodic boundary conditions into account
            E=E+self[0].state*self[i].state                  ▷ Add last term to energy sum
        else
            E=E+self[i].state*self[i+1].state                ▷ Add ith term to running sum
        end if
    end for
     return (-1)*E
end function
```

3.4.2 Net Spin

In addition to the energy of the system, we also wished to calculate the net spin or total spin of the system. The net spin of the system is simply defined as a sum of the spin states of the system. Therefore, a positive net spin indicates that the gas contains a majority of particles in a spin up state while a negative net spin indicates that the gas contains a majority of particles in a spin down state. As in the energy, particles in a Bell state do not contribute to the net spin. Therefore, the net spin, NS, of the system can be defined as:

$$NS = \sum_{i=0}^{N} s_i \tag{36}$$

Pseudo-code for calculating the net spin of the gas according to Eq. 36 is shown below. Just as energy, the net spin calculation was defined as a method of the Gas class.

Algorithm 5 Net Spin Calculation

```
function NET SPIN(self)                                      ▷ self is the gas
    net_spin=0                                               ▷ Begin net spin sum at zero
    for i in self do
        net_spin=net_spin+self[i].state                      ▷ Add ith term to net spin sum
    end for
     return net_spin
end function
```

3.4.3 Entanglement Density

Another quantity of the gas which we wished to calculate and examine was the entanglement density. The entanglement density is defined as the density of entangled states (bell states) in the gas. Therefore, the entanglement density, ED, of the gas can be defined as:

$$ED = \frac{\sum s_{bs}}{N} \tag{37}$$

where $s_{bs} = 1$ if a particle is in a Bell state and $s_{bs} = 0$ if a particle is in either a spin up or spin down state. N is the number of particles (length) of the gas. Pseudo-code for calculating the entanglement density of the gas is shown below. The function was also written as a method of the Gas class.

Algorithm 6 Calculating Entanglement Density

```
function ENTANGLEMENT DENSITY(self)                              ▷ self is the gas
    count=0                                                      ▷ Bell state count, initially zero
    for i in self do
        if self[i].pairing does not equal None then              ▷ Check if particle is in Bell state
            count=count+1                                        ▷ If in Bell state add 1 to Bell state count
        end if
    end for
     return count/LENGTH(self)                                   ▷ Return Bell state sum divided by number of particles
end function
```

3.4.4 Correlation Function

The last thermodynamical quantity which we wished to investigate was the correlation function of the gas. The correlation function measures how correlated two particles states are when separated by a distance r. Therefore, the correlation function is a function of particle separation. The value of the correlation function at a specific particle separation distance, r, is given by:

$$c(r) = \frac{\sum s_i s_j}{2N} \text{ where, } s_i s_j = \begin{cases} 1 & : i = j \\ -1 & : i \neq j \end{cases} \tag{38}$$

where the particles with states s_i and s_j are separated by distance r and N is the number of particles in the gas. In the model, the correlation function was calculated for separation distances from 1 to half the length of the gas and was defined as a method (function) of the Gas class. The pseudo-code for calculating the correlation function is somewhat long so it has been included in Appendix A.2.

3.5 The Interactions

Investigating the thermodynamical effects of a local Bell state projection interaction in a one dimensional spin gas is the main purpose of this model. However, to quantify the nature of these effects a baseline interaction was included. This baseline interaction is the same Ising spin flip interaction used in the Metropolis algorithm investigation of the Ising Model described in [6].

3.5.1 Baseline Interaction

The baseline interaction is the same as that used to simulate the Ising model using the Metropolis algorithm [6]. Since no particles in the gas are in a Bell state initially, an interaction that is only concerned with spin flips can be applied. The only difference between the baseline and Ising methods is that particles move in the baseline model in contrast to the stationary lattice of the Ising model. The interaction works as follows: Pick a random particle. Calculate the energy difference that would result if the particle's spin state is flipped. If flipping the particles spin results in a lower energy or no energy change, then flip the particle's spin state. If flipping the particle's spin state would result in an increase in energy, then randomly decide to flip the particle's spin state based on the thermal probability given by Eq. 32. Pseudo-code for the baseline interaction is given below. The baseline interaction was written as a method of the Gas class.

Algorithm 7 Baseline Interaction

```
function BASELINE INTERACTION(self)                                  ▷ self is the gas
    T=self.Temperature                                   ▷ set temperature to that of gas
    length=LENGTH(self)                                               ▷ length of gas
    i=RANDOM INTEGER(0 to length)               ▷ pick random particle (by index in gas)
    if i=length then                 ▷ Take periodic boundary conditions on right end into account
        Ei=(-1)*((self[i-1].state*self[i].state)+(self[i].state*self[0].state))  ▷ Initial local energy before spin flip
        Eflip=(-1)*((self[i-1].state*((-1)*self[i].state))+(((-1)*self[i].state)*self[0].state))  ▷ Local energy after spin flip
        dE = Eflip - Ei                                  ▷ Energy difference of spin flip.
    **else if** i=0 **then**    ▷ Take periodic boundary condition on left end into account
        Ei=(-1)*((self[length].state*self[i].state)+(self[i].state*self[i+1].state))    ▷ Initial local energy before spin flip
        Eflip=(-1)*((self[length].state*((-1)*self[i].state))+(((-1)*self[i].state)*self[i+1].state))    ▷ Local energy after spin flip
        dE = Eflip - Ei    ▷ Energy difference of spin flip.
    **else**
        Ei=(-1)*((self[i-1].state*self[i].state)+(self[i].state*self[i+1].state))    ▷ Initial local energy before spin flip
        Eflip=(-1)*((self[i-1].state*((-1)*self[i].state))+(((-1)*self[i].state)*self[i+1].state))    ▷ Local energy after spin flip
        dE = Eflip - Ei    ▷ Energy difference of spin flip.
    **end if**
    probability=RANDOM NUMBER(between 0 and 1)
    **if** dE $\leq$ 0 **then**    ▷ spin flip lowers or does not change energy
        self[i].state=self[i].state*(-1)
    **else if** probability $<$ EXP((-1*dE)/T) **then**    ▷ If spin flip increases energy flip with probability given by Eq. 32
        self[i].state=self[i].state*(-1)
    **end if**
**end function**
```

3.5.2 The Bell State Projection Interaction

The model is designed to study the thermodynamical effects of a local Bell state projection interaction (the same interaction involved in quantum teleportation and pair swapping). A local Bell state projection interaction is especially interesting because it is a local interaction which can have long range effects (quantum teleportation). Thus, the Bell state projection interaction is much more complicated than the baseline interaction previously described. When two particles are involved in a local Bell state interaction, three separate situations can occur. The first occurs when the two particles are in a simple spin up or spin down state. In this case, the two particles will be projected into one of the four Bell states given in Eq. 2. Since neither of the particles is part of an EPR pair, no teleportations will occur. The second situation occurs when one of the two particles involved in the interaction is part of an EPR pair. In this case, the two particles involved in the interaction are projected into one of the four Bell states and the state of the non-paired particle (or a phase shift of the state) is transfered to the particle previously in an EPR pair (this particle is not involved in the local interaction and can reside at any location within the gas). This transfer of quantum state follows the quantum teleportation scheme described in section 2.1.2. The third situation occurs when both particles involved in the Bell state projection are part of separate EPR pairs. In this case, a pair swapping will occur in the same manner described in section 2.1.3. Each specific case must be addressed separately within the model. The interaction is executed within the model by first randomly selecting a particle within the gas. Then the interaction is executed on this random (ith) particle and the particle to the right (i+1). The states in which the two particles reside determine which case will be addressed and executed.

The first case is the simplest. In this case, both particles involved in the interaction reside in a simple spin up or spin down state (neither particle is part of an entanged pair). Thus, the Bell state projection interaction will project the two particles into one of the four Bell states given in Eq. 2, resulting in the entanglement of the particles in an EPR pair. The specific Bell state into which the particles are projected is random and equally likely (25% chance for each Bell state). However, the occurrence of the interaction is dependent on the energy change it will create. If the interaction will lower the energy of the gas or leave it unchanged, then the interaction will occur. However, if the energy will be increased by the interaction, the interaction will randomly occur, where the occurrence probability is weighted by the Boltzmann factor given in Eq. 32. Therefore, the change in energy evoked by the interaction must first be calculated before the interaction can be executed. Since a particle in a Bell state does not contribute to the energy of the system (energy of zero), the local energy contribution of the particles to the system will be zero. Thus, the change in energy is given by the local energy contribution of the particles before the interaction. If the position of the randomly selected particle is given by i (thus the interaction will occur on particles i and i+1), the energy difference created when two non entangled particles are projected into a bell state is given by,

$$\Delta E = s_{i-1}s_i + s_i s_{i+1} + s_{i+1}s_{i+2} \tag{39}$$

where s_j is the state of the jth particle in the gas. Some pseudo-code for calculating the energy difference according to Eq. 39 is given below.

Therefore, if ΔE, given by Eq. 39 or Algorithm 8, is less than or equal to zero the interaction will be executed and the particles will be projected into an entangled Bell state. If ΔE is greater than zero the interaction will randomly occur weighted by a Boltzmann factor for the transition, Eq. 32.

Algorithm 8 Energy Difference

```
function ENERGY DIFFERENCE(self, i)    ▷ self is the gas and i is the index of the randomly chosen particle
    Ediff=self[i-1].state*self[i].state+self[i].state*self[i+1].state+self[i+1].state*self[i+2].state
    return Ediff
end function
```

The second case, where one particle is part of an EPR pair and the other is in a spin up or down state, is the most complicated. This case is the most complicated because the state (or phase shift of the state) of the un-entangled particle can be transmitted across the gas. Since the Bell state projection interaction acting on the gas is the same interaction involved in quantum teleportation, the transmission of states across the gas, which occurs in this second case, will follow the quantum teleportation scheme described in section 2.1.2. However, since the state to be transmitted is a simple spin up or spin down state instead of a superposition of the two, the teleportation interaction will be simplified. Consider the situation where one of the particles (particle 2) involved in the interaction is part of the singlet state, Eq. 1, and the other (particle 1) is in a spin up state. The total wave function for the three particles is a product state which can be expressed in terms of the Bell basis of particles 1 and 2. The expression of this state can be accomplished by simply setting a=1 and b=0 in Eq. 6. Therefore, the total wave function for these three particles is,

$$\left|\Psi_{23}^{(-)}\right\rangle |{\uparrow_1}\rangle = \frac{1}{2}\left[-\left|\Psi_{12}^{(-)}\right\rangle |{\uparrow_3}\rangle - \left|\Psi_{12}^{(+)}\right\rangle |{\uparrow_3}\rangle + \left|\Phi_{12}^{(-)}\right\rangle |{\downarrow_3}\rangle + \left|\Phi_{12}^{(+)}\right\rangle |{\downarrow_3}\rangle\right] \quad (40)$$

where particle 3 is the particle paired to particle 2 and not involved in the interaction. It can be shown that the other seven possible situations (all other possible combinations of the four Bell states and a spin up or spin down particle) that can occur are,

$$\begin{aligned}
\left|\Psi_{23}^{(-)}\right\rangle|\downarrow_1\rangle &= \frac{1}{2}\left[-\left|\Psi_{12}^{(-)}\right\rangle|\downarrow_3\rangle-\left|\Psi_{12}^{(+)}\right\rangle|\downarrow_3\rangle+\left|\Phi_{12}^{(-)}\right\rangle|\uparrow_3\rangle+\left|\Phi_{12}^{(+)}\right\rangle|\uparrow_3\rangle\right]\\
\left|\Psi_{23}^{(+)}\right\rangle|\uparrow_1\rangle &= \frac{1}{2}\left[\left|\Psi_{12}^{(-)}\right\rangle|\uparrow_3\rangle+\left|\Psi_{12}^{(+)}\right\rangle|\uparrow_3\rangle+\left|\Phi_{12}^{(-)}\right\rangle|\downarrow_3\rangle+\left|\Phi_{12}^{(+)}\right\rangle|\downarrow_3\rangle\right]\\
\left|\Psi_{23}^{(+)}\right\rangle|\downarrow_1\rangle &= \frac{1}{2}\left[-\left|\Psi_{12}^{(-)}\right\rangle|\downarrow_3\rangle+\left|\Psi_{12}^{(+)}\right\rangle|\downarrow_3\rangle-\left|\Phi_{12}^{(-)}\right\rangle|\uparrow_3\rangle+\left|\Phi_{12}^{(+)}\right\rangle|\uparrow_3\rangle\right]\\
\left|\Phi_{23}^{(-)}\right\rangle|\uparrow_1\rangle &= \frac{1}{2}\left[-\left|\Psi_{12}^{(-)}\right\rangle|\downarrow_3\rangle-\left|\Psi_{12}^{(+)}\right\rangle|\downarrow_3\rangle+\left|\Phi_{12}^{(-)}\right\rangle|\uparrow_3\rangle+\left|\Phi_{12}^{(+)}\right\rangle|\uparrow_3\rangle\right]\\
\left|\Phi_{23}^{(-)}\right\rangle|\downarrow_1\rangle &= \frac{1}{2}\left[-\left|\Psi_{12}^{(-)}\right\rangle|\uparrow_3\rangle+\left|\Psi_{12}^{(+)}\right\rangle|\uparrow_3\rangle+\left|\Phi_{12}^{(-)}\right\rangle|\downarrow_3\rangle-\left|\Phi_{12}^{(+)}\right\rangle|\downarrow_3\rangle\right]\\
\left|\Phi_{23}^{(+)}\right\rangle|\uparrow_1\rangle &= \frac{1}{2}\left[\left|\Psi_{12}^{(-)}\right\rangle|\downarrow_3\rangle+\left|\Psi_{12}^{(+)}\right\rangle|\downarrow_3\rangle+\left|\Phi_{12}^{(-)}\right\rangle|\uparrow_3\rangle+\left|\Phi_{12}^{(+)}\right\rangle|\uparrow_3\rangle\right]\\
\left|\Phi_{23}^{(+)}\right\rangle|\downarrow_1\rangle &= \frac{1}{2}\left[-\left|\Psi_{12}^{(-)}\right\rangle|\uparrow_3\rangle+\left|\Psi_{12}^{(+)}\right\rangle|\uparrow_3\rangle-\left|\Phi_{12}^{(-)}\right\rangle|\downarrow_3\rangle+\left|\Phi_{12}^{(+)}\right\rangle|\downarrow_3\rangle\right].
\end{aligned} \tag{41}$$

Eq. 40 and Eq. 41 show that no matter what combination of EPR state and spin state occurs, the end result of the Bell state projection will be a Bell state and a spin up or down particle. Eq. 40 and Eq. 41 also show that there is an equal probability (25%) for the two particles in the interaction (particles 1 and 2) to be projected into any of the four Bell states. Lastly, these equations indicate that the final state (spin up or spin down) of the entangled particle not involved in the interaction is dependent on the initial Bell state of the EPR pair and the other non-entangled particle, as well as the final Bell state into which the two interacting particles are projected. Thus, when the interacting particles are projected into one of the four Bell states, the state or the flipped state of the non-entangled particle is teleported to the noninteracting particle.

Just as in the first case, the occurrence of the interaction is dependent on the energy change it will cause. However, only the local energy change is considered in the second case, not the energy change that will result from the teleported state. The energy change due to the teleported state is not considered because the resulting teleported state is random and depends on the result of the Bell state projection interaction itself. Therefore, only the local energy change is considered, allowing ΔE to also be calculated using Eq. 39 and Algorithm 3, however s_i or s_{i+1} now equals zero instead of ± 1. Just as in the first case the interaction will occur if $\Delta E \leq 0$, otherwise the the interaction will randomly occur with a probability given by Eq. 32. However, in the second case the teleportation of a random spin state will also occur. Therefore, even if the Bell state projection interaction lowers the local energy, the random teleported state may actually cause the total energy of the system to increase. Conversely, the teleported state may cause the total energy of the system to decrease even though the Bell state projection interaction causes a local energy increase. In addition to this, the teleported state may supplement the net energy increase or decrease resulting from the Bell state projection interaction.

The third and last case that can occur with the Bell state projection interaction is when both interacting particles are in separate EPR pairs. In this case, the Bell state projection of the two particles will cause a pair swapping to occur in the same manner described in section 2.1.3. As Eq. 10 shows, there is an equal probability (25%) of the particles being projected into any of the four Bell States. Also, Eq. 10 shows that the Bell state into which the noninteracting particles are entangled is the same as the Bell state into which the interacting particles are projected by the interaction. Lastly, since all four particles effected by the interaction will remain in Bell states (though different ones), the pair swapping will not change the energy of the gas. Therefore, the interaction will always occur in this case. Since all three of the above cases must be addressed separately in the interaction code, the code for the interaction is quite involved and lengthy. Therefore, the pseudo-code for the Bell state projection interaction has been included in Appendix A.3.

3.5.3 Quantum Decoherence Interaction

Since the central interaction of the model is quantum mechanical in nature, the model also includes a simplified version of a quantum decoherence interaction as well. This quantum decoherence simulates an interaction between the gas and the environment which causes the quantum Bell states in the gas to decohere to the more classical spin up and spin down states. This quantum dechoerence also mitigates an inherent problem of the interaction, over time the Bell state projection interaction will project all the particles within the gas into Bell states. The decoherence interaction mitigates this effect by devolving some of the Bell states within the gas back to spin up and spin down states. The decoherence interaction operates on the amount of time a particle has resided in a Bell state. The probabiliy that the environmental interaction will decohere a Bell state into spin up and spin down states is dependent on the time the particles have been in the Bell state. The decoherence probability is given by,

$$P_d = 1 - e^{-t/\tau} \tag{42}$$

where t is the time in which the particles have been in the Bell state, and τ is the characteristic decoherence time. When $t = \tau$ there is a $1 - e^{-1} \approx 63\%$ probability that the environment interaction will cause the bell state to decohere. For small τ values the Bell states will almost immediately decohere back to spin up or down states, while large τ values will cause almost no decoherence to occur. When Bell states do decohere, they will decohere into the spin up and spin down states consistent with the Bell state in which the particles reside. For example, if two particles residing in the $\left|\Psi_{12}^{(+)}\right\rangle$ state, given by

$$\left|\Psi_{12}^{(+)}\right\rangle = \frac{1}{\sqrt{2}} \left(\left|\uparrow_1\downarrow_2\right\rangle + \left|\downarrow_1\uparrow_2\right\rangle\right),$$

decohere into spin up and spin down states, the possible results are

$$\left|\psi_1\right\rangle = \left|\uparrow_1\right\rangle \text{ and } \left|\psi_2\right\rangle = \left|\downarrow_2\right\rangle$$

or

$$\left|\psi_1\right\rangle = \left|\downarrow_1\right\rangle \text{ and } \left|\psi_2\right\rangle = \left|\uparrow_2\right\rangle .$$

A similar result will follow for the $\left|\Psi_{12}^{(-)}\right\rangle$ state. The result for the $\left|\Phi_{12}^{(+)}\right\rangle$ and $\left|\Phi_{12}^{(-)}\right\rangle$ states would be both spin up or both spin down instead of a combination of the two. The pseudo-code for the decoherence interaction is somewhat long so it has been included in Appendix A.4 for convenience. The decoherence function was defined as a method of the Gas class.

3.6 Model Execution: Temperature Iteration

In order to adequately examine the effects of a Bell state projection interaction on the thermodynamical quantities of the gas, the model must iterate over temperature. To iterate over a temperature range, the gas is iterated for a defined number of iteration steps at each temperature value within the range. This iteration, at each temperature value, allows the most likely states for each temperature to be generated. The thermodynamical quantities (energy, net spin, and entanglement density) are calculated after each iteration step and then averaged to determine a value for each temperature. The resulting gas from each iteration step is passed as the initial gas for the next iteration step. Similarly, the resulting gas from each temperature step is passed as the initial gas to the next temperature value by changing the resulting gas's temperature to the next value and repeating the iteration process at the new temperature. Passing the resulting gas from one temperature value to the next gives the calculated thermodynamical quantities more continuity with respect to temperature as the initial gas will reside in a very likely state configuration. The iteration functions at constant temperature for both the baseline and Bell state projection interaction are shown below. The only difference between the baseline and Bell state projection iteration algorithms is the specific interactions that are executed.

Algorithm 9 Constant Temperature Iteration

```
function INTERACTION ITERATION(N, gas, T_m, tau)    ▷ N is the number of iterations, T_m is the $T_m$ value from Eq. 33 needed for movement algorithm, tau is the characteristic time of the decoherence interaction
    I=LIST(length N)                                 ▷ create time step array
    E=LIST(length N)                                 ▷ create energy array
    N_S=LIST(length N)                               ▷ create net spin array
    Ent_Dens=LIST(length N)                          ▷ create entanglement density array
    for i in RANGE(0 to N) do
        gas.interaction()                            ▷ execute interaction, $gas.baseline()$ for baseline or $gas.bell\_state\_projection()$ and $gas.decoherence()$ for Bell state projection interaction
        gas.move(T_m)                                ▷ execute movement algorithm
        I[i]=i
        E[i]=gas.Energy()                            ▷ calculate energy using energy function
        N_S[i]=gas.Net_Spin()                        ▷ calculate net spin using net spin function
        Ent_Dens[i]=gas.Entanglement_Density()       ▷ calculate entanglement density using entanglement density function
    end for
     return gas, I, E, N_S, Ent_Dens
end function
```

Constant temperature iteration is also useful for examining the correlation function of the gas at various temperatures. Therefore, to observe the effects of both the baseline and Bell state projection interactions on the correlation function, the gas is iterated over a defined number of steps and then the correlation function is calculated. This can then be repeated for several different temperatures to observe the temperature effects on the correlation function.

The temperature iteration allows the temperature dependence of the thermodynamical quantities to be observed for both interactions, baseline and Bell state projection. Pseudo-code for the Bell state projection and quantum decoherence temperature iteration is shown below.

Algorithm 10 Bell State Projection and Quantum Decoherence Temperature Iteration

```
function TEMPERATURE ITERATION(T_i, T_f, TN, N, size, T_m, tau)    ▷ T_i is initial temperature, T_f is final temperature, TN is the number of temperature values between T_i and T_f, N is number of time iteration steps, size is size of gas, T_m is $T_m$ for movement algorithm, and tau is $\tau$ for decoherence interaction
    Temp=LIST(start=T_i, stop=T_f, step=step)    ▷ create temperature array
    Energy=LIST(length TN)    ▷ create energy array
    Net_Spin=LIST(length TN)    ▷ create net spin array
    Ent_Dens=LIST(length TN)    ▷ create entanglement density array
    gas_in=GAS(size, Temp[0])    ▷ create gas
    for i in Temp do
        gas_in.Temperature=Temp[i]    ▷ set new temperature value
        gas_out,I,E,NS,ED=INTERACTION ITERATION(N, gas_in, T_m, tau)    ▷ run iteration function
        Energy[i]=AVERAGE(E)
        Net_Spin[i]=AVERAGE(NS)
        Ent_Dens[i]=AVERAGE(ED)
        gas_in=gas_out
    end for
      return Temp, Energy, Net_Spin, Ent_Dens
end function
```

The baseline temperature iteration algorithm has not been included because it is essentially the same as the bell state projection temperature iteration shown above. The only difference between the two temperature iteration functions is that the Baseline Iteration function is used instead of the Bell state projection interaction Iteration function.

4 Results and Analysis

The Ising inspired model of a one-dimensional dynamic spin system was utilized to investigate the effects of a local Bell state projection interaction. The model specifically investigates how the long range effects of a local Bell state projection interaction changes the thermodynamical properties of a one dimensional gas. The results from the Bell state projection interaction are compared with a baseline dynamic Ising interaction from which the model was based. This baseline comparison allows the specific effects of the Bell state projection interaction to be differentiated from the basic effects of the Ising model. The model investigates the gas's energy, net spin, entanglement density, and correlation function with respect to temperature. Since $k_B = 1$ and $\epsilon = 1$ in the model, all temperature values are given in units of ϵ, the neighboring particle interaction energy specified in Eq. 35.

4.1 Temperature Iteration Results

The thermodynamical quantities of energy, net spin, and entanglement density were investigated through temperature iteration. Temperature iteration allows each thermodynamical quantity's functional dependence on temperature to be determined. We examined how this temperature dependence differed between the baseline and Bell state projection/decoherence interactions. the high temperature (T $\gtrsim$ 5) investigation yielded smooth predictably near zero results for energy and net spin when executed for both interactions. Therefore, the high temperature iteration results have been omitted as they do not contribute to the discussion or add anymore relevant information. The temperature iterations which do yield relevant results were conducted over a temperature range from $T = 5\epsilon$ to $T = 2\epsilon$. The temperature range was divided into 1,150 evenly spaced temperature steps, where the model gas was iterated 10,000 times at each individual temperature step. A gas of length 100 was used as well as a $T_m = 1000\epsilon$ value for the movement algorithm and a $\tau = 200$ value for the decoherence interaction. The energy results for both the baseline and Bell state projection interactions are shown in Figure 2.

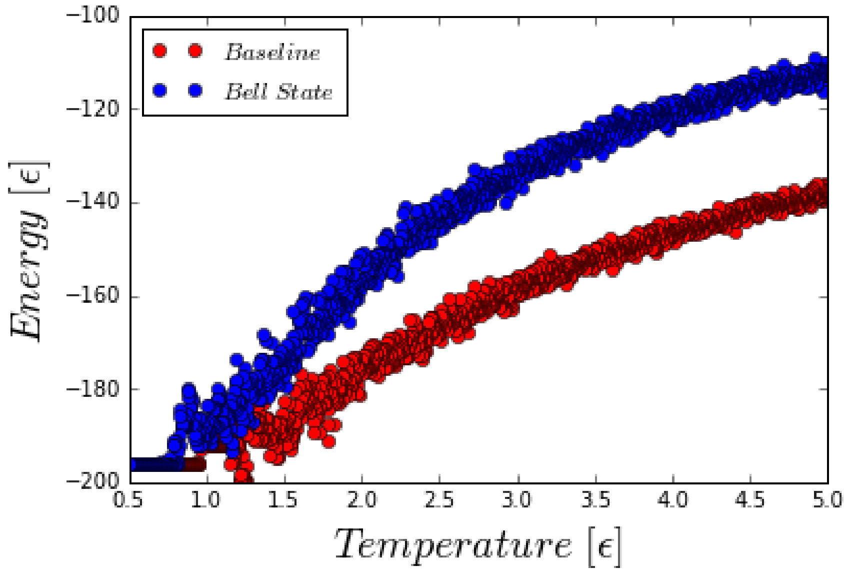


Figure 2: Energy vs Temperature Results.

As can be seen in Figure 2, there is a difference between the baseline and Bell state projection Energies. Both energy functions are smooth and seem to follow a functionality similar to the $(2cosh(\beta\epsilon))^N$ functionality of the one-dimensional Ising model. However, the Bell state projection energy has a steeper curvature and reaches a higher energy. The difference between the energy functions is the result of the Bell state projection interaction.

Since Temperature iteration produces the gas's energy as a function of temperature, Eq. 19 can be utilized to determine the heat capacity at constant volume of the gas. However, since the energy vs temperature functions, Figure 2, are somewhat noisy, taking derivatives yields undefinable results. So the heat capacity was calculated using an equivalent method given in Eq. 43.

$$C_V = \frac{1}{T^2}\left(\left\langle E^2 \right\rangle - \left\langle E \right\rangle^2\right) \tag{43}$$

where $\left\langle E^2 \right\rangle$ is the average of the energy squared and $\langle E \rangle$ is the average energy. The heat capacities as functions of temperature were then determined by using Eq 43. The heat capacities for both the baseline and Bell state projection interactions are shown together in Figure 3 and separately in Figure 4.

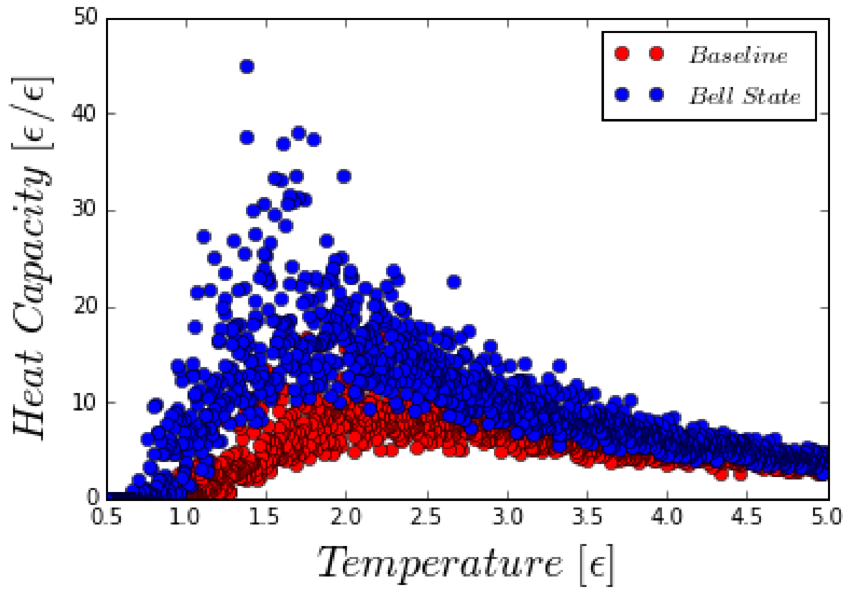
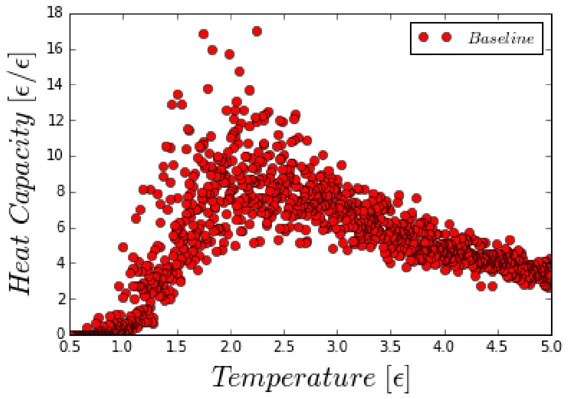
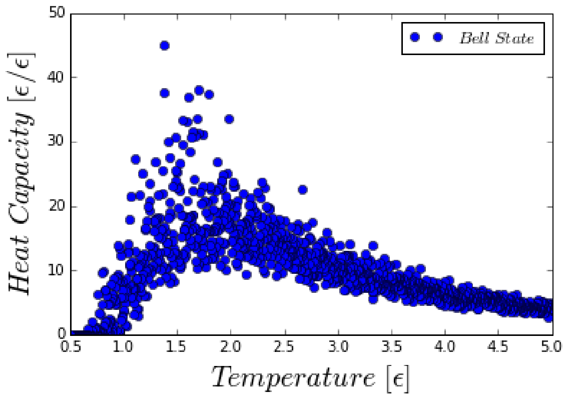


Figure 3: Heat Capacity vs Temperature Results. The units of heat capacity are given in energy(ϵ)/temperature(ϵ).



(a) Heat Capacity for Baseline Interaction



(b) Heat Capacity for Bell State Projection interaction

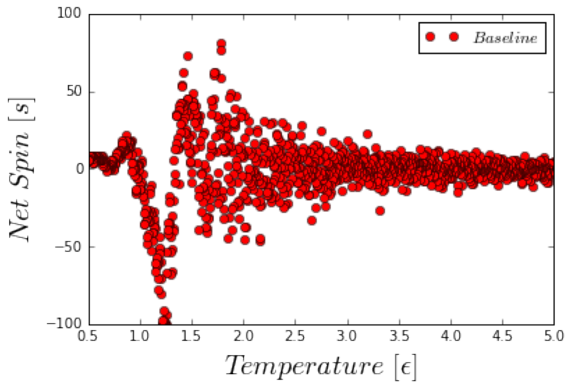
Figure 4: Heat capacity vs temperature for baseline and Bell state projection interactions. The units of heat capacity are given in energy(ϵ)/temperature(ϵ).

Just as with the energy, there is a difference between the magnitude and curvature of the baseline and Bell state projection heat capacities. As can be seen in Figure 4a, the baseline heat capacity is relatively smooth indicating that no phase change occurs in the baseline model. Just like the baseline model, the Bell state projection heat capacity, Figure 4b, is relatively smooth. However, the Bell state projection heat capacity has a greater curvature leading up to its maximum. Also, even though Figure 4b is smooth it does somewhat resemble the non-smooth cusp of universal phase transitions which follow the function,

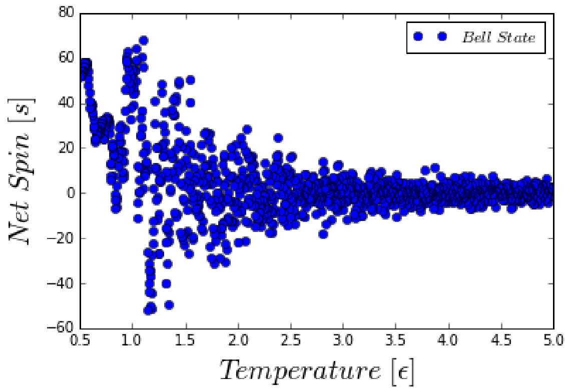
$$C_v \propto |T_c - T|^{-\alpha} \tag{44}$$

where T_c is the temperature of the phase transition, T is the temperature, and α is a universality parameter. Changing the τ value or investigating the model in two dimensions may cause the Bell state projection heat capacity to develop a phase transition following the behavior of Eq. 44.

The net spin results for both the baseline and Bell state projection interactions are shown in Figure 5. As can be seen in Figure 5, the net spin for both the baseline and Bell state projection interactions is very noisy and fluctuates greatly at lower temperatures. This indicates that the system never chooses a preferred spin direction. This lack of a dominating spin direction after a critical temperature indicates that in both interactions there is no phase change in the total spin of the system. This lack of a phase change is also apparent in the smooth heat capacity functions.



(a) Net Spin for Baseline Interaction



(b) Net Spin for Bell State Projection Interactions

Figure 5: Net spin vs temperature for baseline and Bell state projection interactions. The net spin is given in units of s, the magnitude of the spin of a single particle.

Finally the entanglement density vs temperature functions are shown in Figure 6. As can be seen in Figure 6, the entanglement density decreases with decreasing temperature until reaching a value of zero near $T = 0.7\epsilon$. This indicates that the decoherence interaction gains increasing dominance over the Bell state projection interaction as temperature decreases. Different τ values will need to be investigated to see if this general trend persists or if it is just a result of this specific τ value.

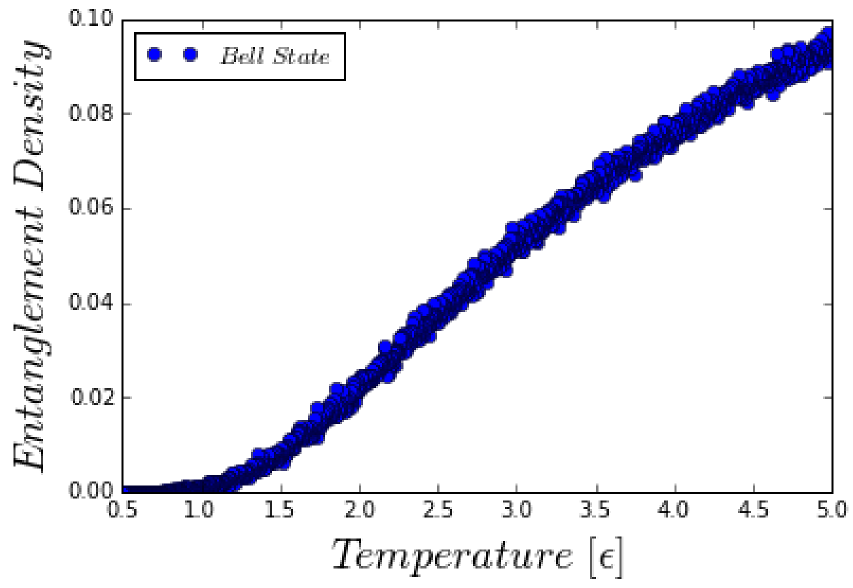
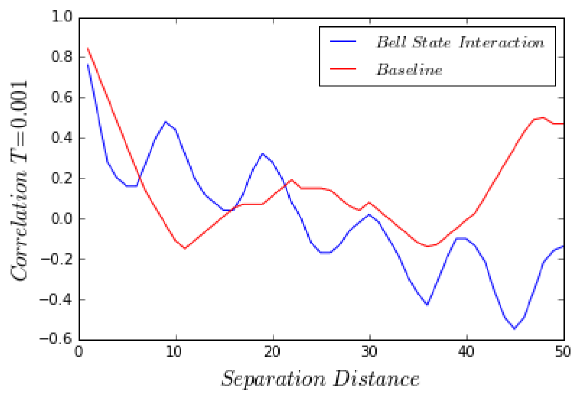


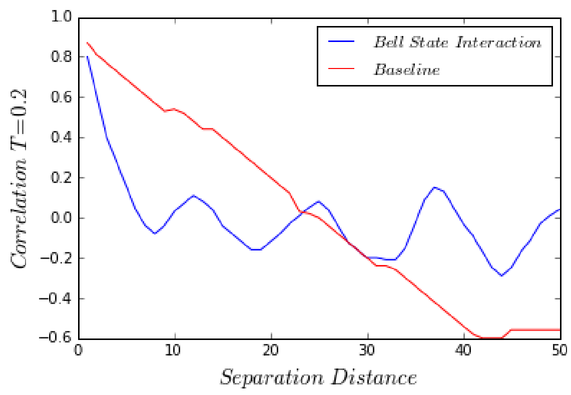
Figure 6: Entanglement Density vs Temperature Results.

4.2 Correlation Function Results

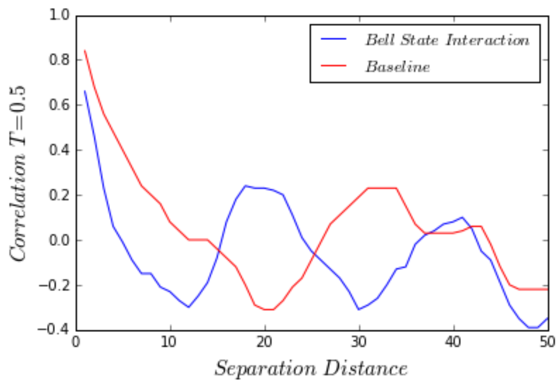
In addition to the energy, heat capacity, net spin, and entanglement density, the correlation function of the gas was also investigated. The correlation function was determined at several different temperatures for both the baseline interaction and the Bell state projection interaction. A gas of length 100 was used with values of $T_m = 1000\epsilon$ and $\tau = 200$. The correlation function was calculated after 1000 iterations. The correlation functions for temperature values of $T = 0.001\epsilon \approx 0\epsilon$, $T = 0.2\epsilon$, $T = 0.5\epsilon$, and $T = 1\epsilon$, are shown in Figure 7. Figure 7 shows some significant differences between the baseline interaction and Bell state projection interaction correlation functions. First, at all four temperatures, the Bell state correlation function has more peaks and troughs than the baseline correlation function. Second, and most importantly, these peaks and troughs seem to occur in a periodic manner. These periodic peaks in the Bell state correlation function are almost certainly a result of the Bell state projection interaction and the quantum teleportations it can cause. In the near future, we will preform a Fourier analysis of the correlation functions to determine if the periodic nature of the correlation function has functional form or is just random fluctuation. Other τ values will also be investigated in the future to see if this periodic nature persists.



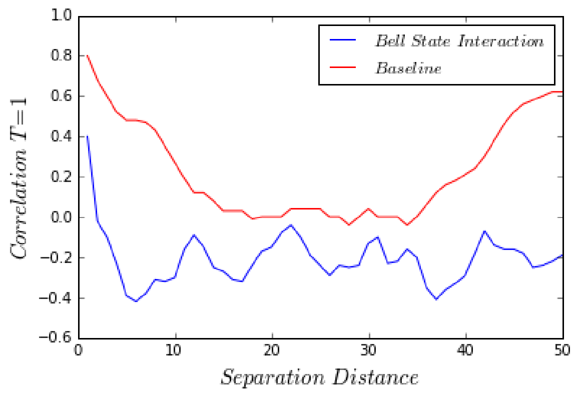
(a) T=0.001



(b) T=0.2



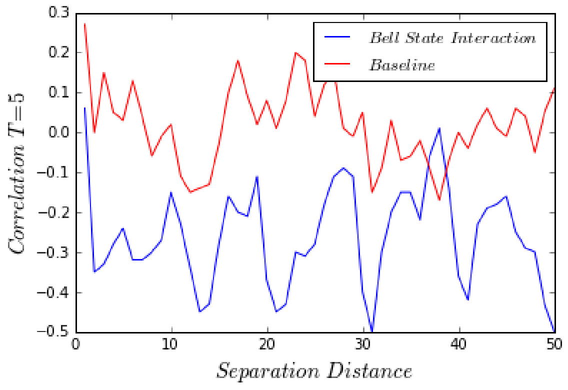
(c) T=0.5



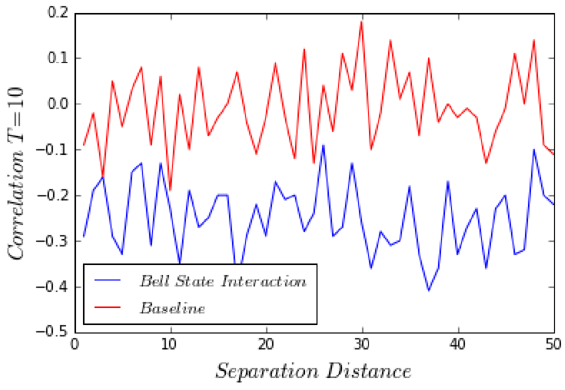
(d) T=1

Figure 7: Low temperature correlation functions for temperatures of $T = 0.001\epsilon$, $T = 0.2\epsilon$, $T = 0.5\epsilon$, and $T = 1\epsilon$. The separation distance is given by difference in lattice position.

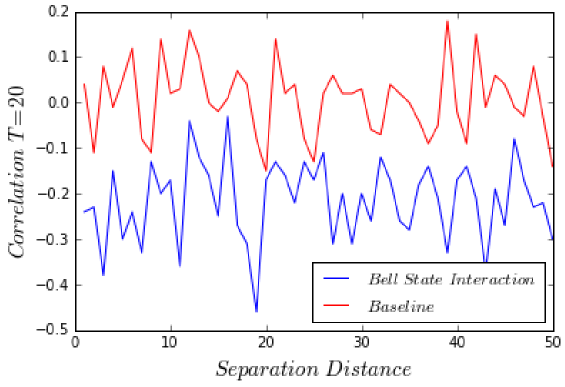
The periodic nature of the Bell state correlation function seen in Figure 7 does not seem to continue, or is at least not as apparent, at higher temperatures. The correlation functions for temperatures of $T = 5\epsilon$, $T = 10\epsilon$, $T = 20\epsilon$, and $T = 50\epsilon$ are shown in Figure 8. Though the higher temperature



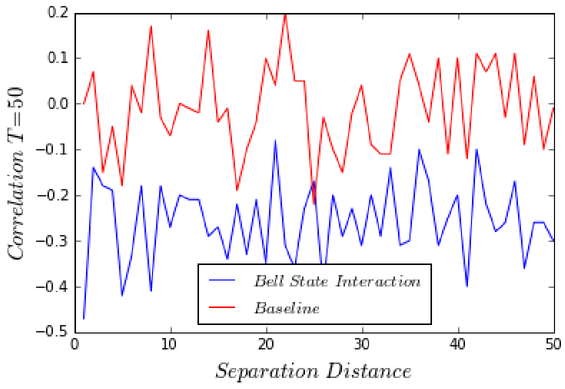
(a) T=5



(b) T=10



(c) T=20



(d) T=50

Figure 8: High temperature correlation functions for temperatures of $T = 5\epsilon$, $T = 10\epsilon$, $T = 20\epsilon$, and $T = 50\epsilon$. The separation distance is given by difference in lattice position.

correlation functions seem to have no significant periodic nature, the Bell state correlation function for $T = 5\epsilon$, Figure 8a, still seems to exhibit some of the periodic nature seen in Figure 7. Otherwise, the higher temperature correlation functions seem to exhibit a more random than periodic nature. However, there is another difference between the baseline and Bell state correlation functions at higher temperature. The Bell state correlation function in all four temperatures in Figure 8 are less correlated than the baseline correlation function. Other τ values will be investigated in the future to see if the Bell state correlation function remains less correlated in all cases.

5 Discussion

The Ising inspired model of a one dimensional dynamic spin system was used to investigate the non-local effects of a local Bell state projection interaction on the thermodynamical quantities of the system. Comparing the thermodynamical quantities that resulted from the Bell state projection interaction with those that resulted from the baseline Ising interaction allowed the non-local effects to be identified. The energy that resulted from the Bell state projection interaction generally followed the same functionality as that of the baseline interaction, however it covered a greater range and had a greater curvature than the baseline energy. This difference in the curvature and magnitude is a result of the non-local and random effects of the Bell state projection interaction. Similarly, the heat capacity results also presented a difference in magnitude and curvature with the Bell state projection heat capacity displaying a higher magnitude. Also, The Bell state projection heat capacity shares some similarities in functionality to the universal power law heat capacity proportionality seen when a phase change occurs. Even though the Bell state projection heat capacity never develops a non-smooth cusp, which would indicate a phase transition, it does resemble the tent shape followed by heat capacities proportional to Eq. 44. Therefore, even though no phase transition occurred in either of the models, we expect one to develop if the model is studied in two dimensions. Also, the differing functionality between the two heat capacities may indicate that the natures of the phase transitions in two dimensions may differ as well. A two dimensional model will be studied in the future.

Even though the Bell state projection model produced energy and heat capacity results greater in magnitude and curvature, there is also an issue with the way in which the energy is defined within the model. The Ising like energy definition of aligned spins being favorable (lower energy) and unaligned spins being unfavorable (higher energy) is adequate when the particles in the lattice do not move. However, since the particles in the gas can move, the relative alignment of adjacent particles will change every time the particles move. Thus, just executing the movement algorithm will cause the energy of the gas to change. These movement effects are suppressed at low temperatures since the particles do not move as much or at all. However, at higher temperatures the particles do move, sometimes significantly, and the energy change do to the movement may be suppressing any high temperature effects. A solution to this problem would be to examine quenched (stationary) systems or redefine how the energy of the gas is determined. The most intuitive manner in which to redefine the energy is to allow some quantum states to be more favorable (lower energy) than others. One way of modeling this is to put the gas in a magnetic field so that either the spin up or spin down state is favored more than the other.

As for the net spin results, they were too inconsistent to provide any insight on how the Bell state projection interaction, or baseline interaction, effected the net spin of the system. All of the net spin results fluctuated too randomly to allow any general trend to be determined. However, this lack of a general trend in the net spin indicates that the system never undergoes an abrupt transition in total spin or magnetization. The entanglement density results however are quantifiable, though there are no Bell states in the baseline interaction case so there are no baseline results for comparison. However, the entanglement density results do provide an insight into the interplay between the Bell state projection interaction and the decoherence interaction. The entanglement density indicates which is dominating, the Bell state if the entanglement density exponentially approaches one or decoherence if the entanglement density remains low (near zero). For the τ value used, it is apparent that the decoherence interaction dominates increasingly with lowering temperature. The effects of varying τ values on the general trend of the entanglement density still need to be investigated and will be done in the future.

The correlation function results displayed a difference between the Bell state projection interaction and the baseline interaction as well. As discussed in section 4.2, the correlation function which resulted from the Bell state interaction displayed periodic tenancies at low temperatures, where the baseline correlation function showed little to no periodic tendencies at the same temperatures. The periodic nature of the correlation function will be investigated in the future by applying a Fourier transform to the correlation function to determine the frequency components of the function and their respective magnitudes. This future Fourier analysis will be conducted for the correlation function at various temperatures and lengths to determine the frequency components dependence on both parameters. By conducting a Fourier analysis of the correlation function we will be able to determine if the apparent periodic nature is truly a result of the Bell state projection interaction or some intrinsic function of the model.

In addition to the future investigation of the correlation function and the refinement of the energy and its involvement in the interactions, we wish to expand the model in several other regards. Firstly, we will expand the model into two dimensions allowing a more realistic gas to be modeled. The two dimensional model will investigate both quenched (stationary) and dynamic systems. Secondly, we wish to include a more involved and realistic decoherence model dependent of the particle's phase. Lastly, we hope the model can come to include particles residing in a superposition of spin up and down states, not just residing in one of the two. We believe these refinements to the model will further distance the Bell state projection result from those of the baseline interaction.

6 Conclusions

The Ising inspired model of a one-dimensional dynamic spin system was utilized to investigate the thermodynamic effects of a Bell state projection interaction on the system. Since a local Bell state projection interaction can have long range effects through quantum teleportation, the model specifically worked to isolate the thermodynamical effects that would result from this local, nonlocal interaction. In order to isolate these effects, the thermodynamical results of the Bell state projection interaction were compared to a baseline Ising interaction. The thermodynamical quantiles investigated were the gas's energy, heat capacity, net spin, entanglement density, and correlation function. The resulting temperature dependent energies and heat capacities, from both the Bell state and baseline interactions, had differing curvatures and magnitudes. Also, the Bell state projection interaction's heat capacity functionally differed from the baseline heat capacity as it more closely resembled the universal phase change proportionality function of Eq. 44. These differences indicate that the Bell state projection interaction does create some change in the thermodynamic quantities of energy and heat capacity. The temperature dependent net spin results included too much random fluctuation to determine a general trend in functionality or magnitude. However, this lack of a general trend indicates that both the Bell state projection and baseline models never experienced a defined transition in the total spin of the system. The temperature dependent entanglement density results did follow a general trend however. When the decoherence interaction is neglected (very high τ values), the entanglement density approaches a value of one in an exponential manner. This is an inherent result of the Bell state projection interaction. However, when the decoherence interaction is involved, it mitigates these effects of the Bell state projection interaction and keeps the entanglement density low. Overall, the entanglement density represents the interplay of the Bell state projection and decoherence interactions. The Correlation functions resulting from the Bell state and baseline interactions displayed a significant difference. At low temperatures, the Bell state correlation function generally displayed a periodic nature which was much less present or not present at all in the baseline correlation function. At higher temperatures, the periodic nature of the Bell state correlation function becomes non-apparent , however the Bell state correlation function becomes much less correlated (lower magnitude) than the baseline correlation function. Overall, the acquired results indicate that the Bell state projection interaction does effect the thermodynamics of the one dimensional spin system.

The periodic nature of the correlation functions will be investigated in the future through Fourier analysis. We are also in the process of expanding the model into two dimensions, as well as possibly including particles which can reside in a super imposed state of spin up and down. In addition to this, varying τ values in the decoherence interaction will be investigated to determine if varying τ values also cause a variation in the thermodynamics of the system. In order to correct for the energy variation caused by particle movement, stationary systems will be studied and the way in which the energy is defined will be refined to be more complementary to particle movement. Also, the decoherence interaction will be made more rigorous and realistic.

A Extended Pseudocode Blocks

Some of the pseudocode blocks defining the algorithms and execution of the model are quite long. Therefore, in the interest of retaining fluidity in the written model description, the longer blocks of pseudocode have been included in this appendix. These algorithms may reference previous algorithms defined earlier in the paper.

A.1 The Movement Algorithm

The pseudocode for the movement algorithm is given below. The movement function was written as a method (function) of the Gas class we defined. Therefore, the self argument just refers to the gas itself.

Algorithm 11 Particle Movement

```
function MOVE(self,$T_m$)    ▷ Movement function, self is the gas and $T_m$ is same value as in Eq. 33
    new=GAS(length of self, self.Temperature)    ▷ Makes new empty gas
    for i in new do
        new[i]=PARTICLE(empty,none,none)
    end for
    vmax=length of self    ▷ Sets max velocity
```

```
    for i in self do
        moved=no    ▷ Sets variable to determined if each particle has been moved
        number=RANDOM GAUSSIAN NUMBER($\mu$,$\sigma$)    ▷ Generates random number according to
random Gaussian distribution. $\mu$ and $\sigma$ given by Eq. 33 and Eq.  34
        velocity=INTEGER(number)    ▷ Sets velocity for each particle
        if velocity $>$ vmax then velocity=vmax    ▷ Makes sure velocity is not grater than vmax
        end if
        direction=RANDOM NUMBER(between 0 and 1)
        if velocity=0 then
            if new[i].index=empty then    ▷ Particle stays in same spot if other particle is not there
                new[i]=self[i]
            else
                for k in new do    ▷ if Particles position is already taken search for empty Position
                    if new[k].index=empty
                        new[k]=self[i] then    ▷ move particle to empty position
                        Break
                    end if
                end for
            end if
        else if direction $>$ 0.5 then    ▷ move left
            if i + velocity $>$ length self then    ▷ Take periodic boundary conditions into account,
moving back to beginning of gas
                for j in (velocity + i - LENGTH(self)) to 0 do    ▷ Search for empty position in beginning
of gas
                    if new[j].index=empty then
                        new[j]=self[i]    ▷ move particle to empty position if available
                        moved=yes    ▷ note that particle has been moved
                        Break
                    end if
                end for
                if moved=no then    ▷ if particle did not move periodically to other side of gas, desired
positions in beginning of gas were not available
                    for j in LENGTH(self) to i do                               ▷ search closer positions
                        if new[j].index=empty then
                            new[j]=self[i]                ▷ move particle to empty position if available
                            moved=yes                             ▷ note that particle has been moved
                            Break
                        end if
                    end for
                    if moved=no then            ▷ Particle was not moved, no desired positions available
                        for k in new do                                    ▷ search for empty Position
                            if new[k].index=empty
                                new[k]=self[i] then                      ▷ move particle to empty position
                                Break
                            end if
                        end for
                    end if
                end if
            else                                          ▷ Periodic boundary conditions not needed
                for j in velocity + i to i do                              ▷ search for empty position
                    if new[j].index=empty then
                        new[j]=self[i]                    ▷ move particle to empty position if available
                        moved=yes                                 ▷ note that particle has been moved
                        Break
                    end if
                end for
                if moved=no then                ▷ Particle was not moved, no desired positions available
                    for k in new do                                        ▷ search for empty Position
                        if new[k].index=empty
                            new[k]=self[i] then                          ▷ move particle to empty position
                            Break
                        end if
                    end for
                end if
            end if
        else if direction <= 0.5 then                                                    ▷ move left
            if velocity > i then      ▷ Take periodic boundary conditions into account, moving particle back to end of gas
                for j in (LENGTH(self)+ i - velocity) to LENGTH(self) do   ▷ Search for empty position in end of gas
                    if new[j].index=empty then
                        new[j]=self[i]                     ▷ move particle to empty position if available
                        moved=yes                                 ▷ note that particle has been moved
                        Break
                    end if
                end for
                if moved=no then    ▷ if particle did not move periodically to other side of gas, desired positions in end of gas were not available
                    for j in 0 to i do                                          ▷ search closer positions
                        if new[j].index=empty then
                            new[j]=self[i]                 ▷ move particle to empty position if available
                            moved=yes                             ▷ note that particle has been moved
                            Break
                        end if
                    end for
                    if moved=no then              ▷ Particle was not moved, no desired positions available
                        for k in new do                                       ▷ search for empty Position
                            if new[k].index=empty
                                new[k]=self[i] then                       ▷ move particle to empty position
                                Break
                            end if
                        end for
                    end if
                end if
            else                                                 ▷ Periodic boundary conditions not needed
                for j in i - velocity to i do                                  ▷ search for empty position
                    if new[j].index=empty then
                        new[j]=self[i]                     ▷ move particle to empty position if available
                        moved=yes                                 ▷ note that particle has been moved
                        Break
                    end if
                end for
                if moved=no then                  ▷ Particle was not moved, no desired positions available
                    for k in new do                                           ▷ search for empty Position
                        if new[k].index=empty
                            new[k]=self[i] then                           ▷ move particle to empty position
                            Break
                        end if
                    end for
                end if
            end if
        end if
    end for
```

```
        for i in self do self[i]=new[i]          ▷ Set original positions to the moved positions in the new gas
    self[i].position=i                                                  ▷ Set new positions of each particle
        end for
    end function
```

A.2 The Correlation Function

The Pseudocode for the correlation function is given below. The code calculates the correlation function for separation distances from 1 to half the length of the gas. Just as the movement function, the correlation function was defined as a method of the Gas class.

Algorithm 12 Correlation Function

```
function CORRELATION(self)                                              ▷ self is the gas
    separation=ARRAY(length of half self)                               ▷ Create separation array
    for i in 1 to LENGTH(self)/2 do                                     ▷ Set values of separation array
        separation[i]=i
    end for
    correlation=ARRAY(length of separation)                             ▷ Create correlation function array
    for i in separation do
        cor=1                        ▷ Set correlation sum for each separation distance, initially zero
        for j in self do                                                ▷ Loop through gas
            if j - separation[i] < 0 then    ▷ Take periodic boundary conditions on left end into account
                if self.state=self[LENGTH(self) + j - separation[i]].state then    ▷ Check correlation for particle distance separation[i] to left
                    cor=cor+1                                ▷ Add to correlation if particles correlated
                else
                    cor=cor-1                  ▷ Subtract from correlation if particles are not correlated
                end if
                if self.state=self[j + separation[i]].state then    ▷ Check correlation for particle distance separation[i] to right
                    cor=cor+1                                ▷ Add to correlation if particles correlated
                else
                    cor=cor-1                  ▷ Subtract from correlation if particles are not correlated
                end if
            else if j + separation[i] > LENGTH(self) then▷ Take periodic boundary conditions on right end into account
                if self.state=self[j - separation[i]].state then    ▷ Check correlation for particle distance separation[i] to left
                    cor=cor+1                                ▷ Add to correlation if particles correlated
                else
                    cor=cor-1                    ▷ Subtract from correlation if particles are not correlated
                end if
                if self.state=self[j + separation[i] - LENGTH(self)].state then    ▷ Check correlation for
particle distance separation[i] to right
                    cor=cor+1                                  ▷ Add to correlation if particles correlated
                else
                    cor=cor-1                    ▷ Subtract from correlation if particles are not correlated
                end if
            else                                               ▷ No periodic boundary conditions needed
                if self.state=self[j - separation[i]].state then    ▷ Check correlation for particle distance
separation[i] to left
                    cor=cor+1                                  ▷ Add to correlation if particles correlated
                else
                    cor=cor-1                    ▷ Subtract from correlation if particles are not correlated
                end if
                if self.state=self[j + separation[i]].state then    ▷ Check correlation for particle distance
separation[i] to right
                    cor=cor+1                                  ▷ Add to correlation if particles correlated
                else
                    cor=cor-1                    ▷ Subtract from correlation if particles are not correlated
                end if
            end if
        end for
        correlation[i]=cor/(2*LENGTH(self)                      ▷ Divide each correlation term by 2N
    end for
     return separation, correlation
end function
```

A.3 The Bell State Interaction Algorithm

The pseudo-code for the Bell state projection interaction is below. The function covers all three cases described in section 3.5.2. The function was written as a method of the Gas class

Algorithm 13 Bell State Projection Interaction

```
function BELL STATE PROJECTION INTERACTION(self)                    ▷ self is the gas
    i=RANDOM NUMBER(between 0 and gas length)                ▷ random interaction particle
    particle=self[i]
    if i=length of gas then
        other=self[0]                                      ▷ periodic boundary conditions
    else
        other=self[i+1]
    end if
    DE=ENERGY DIFFERENCE(self, i)         ▷ Calculate energy difference from interaction
    if particle.pairing=None and other.pairing=None then    ▷ Case1: both particles not paired
        r=RANDOM NUMBER(between 0 and 1)                         ▷ random Bell state chance
        p=RANDOM NUMBER(between 0 and 1)                     ▷ random interaction probability
        if DE<=0 then
            particle.state=0                                           ▷ entangle particles
            particle.bell_state_time=0
            particle.pairing=other
            other.state=0
            other.bell_state_time=0
            other.pairing=other
            if r<=0.25 then
                particle.bell_state=1                                  ▷ set random Bell state
                other.bell_state=1
            else if r>0.25 and r¡=0.5 then
                particle.bell_state=2                                  ▷ set random Bell state
                other.bell_state=2
            else if r>0.5 and r¡=0.75 then
                particle.bell_state=3                                  ▷ set random Bell state
                other.bell_state=3
            else
                particle.bell_state=4                                  ▷ set random Bell state
                other.bell_state=4
            end if
        else if p<EXP(-DE/self.Temperature) then
            particle.state=0                                           ▷ entangle particles
            particle.bell_state_time=0
            particle.pairing=other
            other.state=0
            other.bell_state_time=0
            other.pairing=other
            if r<=0.25 then
                particle.bell_state=1                                    ▷ set random Bell state
                other.bell_state=1
            else if r>0.25 and r<=0.5 then
                particle.bell_state=2                                    ▷ set random Bell state
                other.bell_state=2
            else if r>0.5 and r<=0.75 then
                particle.bell_state=3                                    ▷ set random Bell state
                other.bell_state=3
            else
                particle.bell_state=4                                    ▷ set random Bell state
                other.bell_state=4
            end if
        end if
    else if (particle.paring! =None and other.pairing=None) or ((particle.paring=None and
other.pairing! =None) then                                                                  ▷ Case2
        if particle.paring! =None then                                           ▷ particle is paired
            partner=particle.pairing                                           ▷ set partner particle
            state=other.state
        else if other.paring! =None then                                            ▷ other is paired
            partner=other.pairing                                              ▷ set partner particle
            state=particle.state
        end if
        r=RANDOM NUMBER(between 0 and 1)                               ▷ random Bell state chance
        p=RANDOM NUMBER(between 0 and 1)                           ▷ random interaction probability
        if DE<=0 then
            particle.state=0                                                   ▷ entangle particles
            particle.bell_state_time=0
            particle.pairing=other
            other.state=0
            other.bell_state_time=0
            other.pairing=other
            partner.bell_state=None
            partner.bell_state_time=None
            partner.pairing=None
            if r<=0.25 then
                particle.bell_state=1                                    ▷ set random Bell state
                other.bell_state=1
                if partner.bell_state=1 or partner.bell_state=2 then  ▷ set partner state according to Bell
state result
                    partner.state=state
                else
                    partner.state=-1*state
                end if
        else if r>0.25 and r<=0.5 then
            particle.bell_state=2                                        ▷ set random Bell state
            other.bell_state=2
            if partner.bell_state=1 or partner.bell_state=2 then  ▷ set partner state according to Bell state result
                partner.state=state
            else
                partner.state=-1*state
            end if
        else if r>0.5 and r<=0.75 then
            particle.bell_state=3                                        ▷ set random Bell state
            other.bell_state=3
            if partner.bell_state=3 or partner.bell_state=4 then  ▷ set partner state according to Bell state result
                partner.state=state
            else
                partner.state=-1*state
            end if
        else
            particle.bell_state=4                                        ▷ set random Bell state
            other.bell_state=4
            if partner.bell_state=3 or partner.bell_state=4 then  ▷ set partner state according to Bell state result
                partner.state=state
            else
                partner.state=-1*state
            end if
        end if
    else if p<EXP(-DE/self.Temperature) then
        particle.state=0                                                 ▷ entangle particles
        particle.bell_state_time=0
        particle.pairing=other
        other.state=0
        other.bell_state_time=0
        other.pairing=other
        partner.bell_state=None
        partner.bell_state_time=None
        partner.pairing=None
        if r<=0.25 then
            particle.bell_state=1                                        ▷ set random Bell state
            other.bell_state=1
            if partner.bell_state=1 or partner.bell_state=2 then  ▷ set partner state according to Bell state result
                partner.state=state
                else
                    partner.state=-1*state
                end if
            else if r>0.25 and r<=0.5 then
                particle.bell_state=2                                        ▷ set random Bell state
                other.bell_state=2
                if partner.bell_state=1 or partner.bell_state=2 then  ▷ set partner state according to Bell
state result
                    partner.state=state
                else
                    partner.state=-1*state
                end if
            else if r>0.5 and r<=0.75 then
                particle.bell_state=3                                        ▷ set random Bell state
                other.bell_state=3
                if partner.bell_state=3 or partner.bell_state=4 then  ▷ set partner state according to Bell
state result
                    partner.state=state
                else
                    partner.state=-1*state
                end if
            else
                particle.bell_state=4                                        ▷ set random Bell state
                other.bell_state=4
                if partner.bell_state=3 or partner.bell_state=4 then  ▷ set partner state according to Bell
state result
                    partner.state=state
                else
                    partner.state=-1*state
                end if
            end if
        end if
    else if particle.paring! =None and other.pairing! =None then        ▷ Case3: both particles paired
        partner1=particle.pairing                                                ▷ set partner1
        partner2=other.pairing                                                   ▷ set partner2
        r=RANDOM NUMBER(between 0 and 1)                               ▷ random Bell state chance
        p=RANDOM NUMBER(between 0 and 1)                          ▷ random interaction probability
        particle.state=0                                                         ▷ entangle particles
        particle.bell_state_time=0
        particle.pairing=other
        other.state=0
        other.bell_state_time=0
        other.pairing=other
```

```
            partner1.state=0                                   ▷ entangle partners
            partner1.bell_state_time=0
            partner1.pairing=partner2
            partner2.state=0
            partner2.bell_state_time=0
            partner2.pairing=partner1
            if r<=0.25 then
                particle.bell_state=1                          ▷ set random Bell state
                other.bell_state=1
                partner1.bell_state=1
                partner2.bell_state=1
            else if r>0.25 and r<=0.5 then
                particle.bell_state=2                          ▷ set random Bell state
                other.bell_state=2
                partner1.bell_state=2
                partner2.bell_state=2
            else if r>0.5 and r<=0.75 then
                particle.bell_state=3                          ▷ set random Bell state
                other.bell_state=3
                partner1.bell_state=3
                partner2.bell_state=3
            else
                particle.bell_state=4                          ▷ set random Bell state
                other.bell_state=4
                partner1.bell_state=4
                partner2.bell_state=4
            end if
        end if
    end function
```

A.4 Quantum Decoherence Algorithm

The pseudo-code for the quantum decoherence algorithm is below. This function executes the quantum decoherence of the Bell states in the gas to spin up and spin down states according to the probability given by Eq. 42. This function was written as a method of the Gas class.

Algorithm 14 Quantum Decoherence Interaction

```
function DECOHERENCE(self, tau)                    ▷ self is the gas and tau is characteristic time
    for i in self do
        if self[i].paring does not equal None then                    ▷ if particle is in Bell state
            r=RANDOM NUMBER(between 0 and 1)                                   ▷ random chance
            t=self[i].bell_state_time                                         ▷ time in bell state
            if r<1-EXP((-1*t)/tau) then                ▷ Decohere according to probability of Eq. 42
                p=RANDOM NUMBER(between 0 and 1)                           ▷ result probability
                if self[i].bell_state=1 or self[i].bell_state=2 then     ▷ in $|\Psi^+\rangle$ or $|\Psi^-\rangle$ states
                    if p<0.5 then                                                    ▷ result 1
                        self[i].state=1                                               ▷ spin up
                        self[i].bell_state=None
                        self[i].bell_state_time=None
                        self[i].pairing.state=-1                                    ▷ spin down
                        self[i].pairing.bell_state=None
                        self[i].pairing.bell_state_time=None
                        self[i].pairing.pairing=None
                        self[i].pairing=None
                    else                                                             ▷ result 2
                        self[i].state=-1                                            ▷ spin down
                        self[i].bell_state=None
                        self[i].bell_state_time=None
                        self[i].pairing.state=1                                       ▷ spin up
                        self[i].pairing.bell_state=None
                        self[i].pairing.bell_state_time=None
                        self[i].pairing.pairing=None
                        self[i].pairing=None
                    end if
                else if self[i].bell_state=3 or self[i].bell_state=4 then      ▷ in $|\Phi^+\rangle$ or $|\Phi^-\rangle$ states
                    if p<0.5 then                                              ▷ result 1
                        self[i].state=1                                        ▷ spin up
                        self[i].bell_state=None
                        self[i].bell_state_time=None
                        self[i].pairing.state=1                                ▷ spin up
                        self[i].pairing.bell_state=None
                        self[i].pairing.bell_state_time=None
                        self[i].pairing.pairing=None
                        self[i].pairing=None
                    else                                                       ▷ result 2
                        self[i].state=-1                                       ▷ spin down
                        self[i].bell_state=None
                        self[i].bell_state_time=None
                        self[i].pairing.state=-1                               ▷ spin down
                        self[i].pairing.bell_state=None
                        self[i].pairing.bell_state_time=None
                        self[i].pairing.pairing=None
                        self[i].pairing=None
                    end if
                end if
            end if
        end if
    end for
    for j in self do
        if self[j].pairing does not equal None then                ▷ particle still in Bell state
            self[j].bell_state_time+=1                             ▷ increase bell state time by 1
        end if
    end for
end function
```

References

[1] D. J. Griffiths, *Introduction to Quantum Mechanics*, Second Edition, (Pearson Prentice Hall, Upper Saddler River, NJ, 2005.)

[2] A. Einstein, B. Podolsky, and N. Rosen, Phys. Rev. **47**, 777 (1935).

[3] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W. K. Wootters, Phys. Rev. **70**, 1895 (1993).

[4] S. Bose, V. Vedral, P. L. Knight, Phys. Rev. A. **57**, 2 (1998).

[5] W. H. Zurek, *Decoherence and the Transition from the Quantum to Classical - Revisited*, (Los Alamos Science, 2002).

[6] D. V. Schroeder, *An Introduction To Thermal Physics*, (Addison Wesley Longman, United States, 2000.)