

Investigating Electron Tunnelling Using Path Integral Monte Carlo

Samuel Leigh Stone
Supervisor: Dr Massimo Mella

December 2, 2011



Acknowledgements

With thanks to:

Dr Massimo Mella For his expert and insightful supervision, for his continuously useful suggestions throughout the project, and for being incredibly patient each time I made a mistake, and maintaining that patience even when the completion of the project took much longer than it probably should have

Dr Pär Hakansson For much help and advice during the first half of the project

Cardiff University School of Chemistry and the Theoretical Chemistry Group
For funding and support

ARCCA For the use of the Cardiff University Merlin computing cluster, without which this work would not have been possible

My wife, Karen For supporting me in every way during my year of writing, throughout the duration of the project, and also for pushing me to continue working when I needed a push

Contents

I	Abstract	8
II	Introduction	9
1	Introduction	9
1.1	Electron Transfer	9
1.2	Quantum–Mechanical Tunnelling	9
1.3	Examples of Electron Transfer	10
1.4	Previous Methods for Modelling Tunnelling	12
1.4.1	WKB Method	12
1.4.2	Path Integral Instanton Approach	13
1.4.3	Absorbing Boundary Condition Green’s Functions	14
1.5	Proposed Method	15
1.6	Monte Carlo over Molecular Dynamics	16
III	Theory and Methods	17
2	Path Integral Formulation	17
2.1	Feynman’s Postulates	17
2.2	Action	18
2.3	Transforming from Quantum Mechanics to Statistical Mechanics .	19
2.4	Practicalities	20
3	Trotter Splitting	21
3.1	Partition Function	21
3.2	Quantum Mechanical Partition Function	21
3.3	Discussion	23
4	Monte Carlo Methods	26
4.1	History	26
4.2	Overview	26
4.3	Markov Chain Methods	27
4.4	Metropolis Monte Carlo	27

5	Single-Slice Path Integral Monte Carlo	30
5.1	Single-Slice Basic Method	30
5.2	Implementation	32
5.3	Sampling	32
5.4	Single-Slice Normal Method	33
6	Force-Bias Scheme	34
6.1	Inefficiency	34
6.2	Force-Bias	34
6.3	Time	35
6.4	Algorithm	35
6.5	Discussion	38
7	Multi-Slice Methods	40
7.1	Slow Convergence	40
7.2	Brownian Bridge	40
7.3	Lévy Construction	42
7.4	Multi-level Bisection Method	42
7.5	Section Regrowth Method	44
8	The Method of Expanded Ensembles	46
8.1	Free Energy	46
8.2	Expanded Ensembles	46
8.3	Free Energy Difference	49
8.4	Optimisation of Weighting Parameter	49
8.5	Compute Time	50
9	Tunnelling Time	51
9.1	Tunnelling Splitting	51
9.2	Kink Free Energy	53
10	Other Considerations	56
10.1	Electron Correlation	56
10.2	Temperature	57
10.3	Parallelism	57
IV	Testing and Results	58

11 Testing the Basic Methodology	58
11.1 Histograms	58
11.2 Simple Tests	58
11.3 Temperatures	60
11.4 Asymmetries	61
11.4.1 Convergent Trajectories	61
11.4.2 Instability	62
11.4.3 Ergodicity	62
11.5 Optimum Bead Numbers	63
11.6 Parallel Speed Tests	64
12 Autocorrelation	73
12.1 Autocorrelation	73
12.2 Chain Correlation	75
13 Testing the Method of Expanded Ensembles	79
13.1 Flat Histogram Approach	79
13.2 Converging the Free Energy	80
13.3 Free Energy with Number of Slices	80
13.4 Optimising the Decorrelation Time	80
13.5 Free Energy with Number of States	82
14 Square Well Systems	84
14.1 Smoothing the Potential	84
14.2 Double Well in 1-D	84
14.3 Double Well in 2-D	87
14.4 Triple Well in 3D	88
14.5 Discussion	89
15 The Argon Atom Model	95
15.1 Argon Barrier	95
15.2 Results	95
16 Nanotubes	99
16.1 Nanotube Conductivity	99
16.2 Building the Potential	99
16.3 Testing the Potential	102

16.4	Test Cases	107
16.5	Results and Discussion	107
V	Conclusion	110
17	Conclusion	110
17.1	Advantages	110
17.2	Drawbacks	111
17.3	Further Work	112
17.3.1	Fourth-Order Propagator	112
17.4	Further Applications	113
A	WKB Method	116
B	Pseudo-Random Numbers	119
B.1	Pseudo-Random Number Generators	119
B.1.1	Linear-Congruential Generators	119
B.1.2	Lagged-Fibonacci Generators	120
B.1.3	Linear-Feedback Shift Register Generators	120
B.2	Marsaglia's Diehard Suite	121
B.3	Non-Uniform Distributions	121
B.3.1	Inverse Cumulative Distribution Function	121
B.3.2	Box-Muller Transformation	122
B.3.3	Ziggurat Algorithm	123
B.4	Quasi-Random Numbers	124
C	Parallelism	125
C.1	Parallel Architecture	125
C.2	Shared Memory and OpenMP	126
C.3	Distributed Memory and MPI	128
D	Unit Conversions	131
D.1	Energy	131
D.2	Distance	131
D.3	Time	131
D.4	Electric Field	131

E	Usage Guide to Code	132
E.1	Parameters File	132
E.2	Command Line Options	134
E.3	External Files	136

Part I

Abstract

The problem of electron tunnelling was tackled using a path integral Monte Carlo approach, based upon the established technique of using the Trotter theorem to transform Feynman's path integral approach into a system able to be modelled classically. The multi-slice approach to Metropolis Monte Carlo path integral simulation was coupled with the method of expanded ensembles, a technique for calculating the free energy of a system. This enabled the simple calculation of kink free energy differences for complicated potentials, allowing the tunnelling splitting and thus the electron's quantum mechanical tunnelling time to be obtained. Code was developed that allows complex electron tunnelling systems to be studied.

Part II

Introduction

1 Introduction

1.1 Electron Transfer

Understanding the transfer of electrons from one site to another is crucial to a deeper understanding of much of the subject of chemistry. Redox reactions in chemistry involve electron transfer, and key biological reactions such as respiration and photosynthesis rely on electron transfer, often through water or other barriers. For this reason many transfer reactions also imply that quantum mechanical tunnelling has taken place; particularly in biological systems the electron may be required to surmount a classically impossible barrier in order for a reaction to take place.

Many physical processes in electronics are enabled by the transfer or tunnelling of electrons, although tunnelling can also cause problems in microscopic electronic components.

1.2 Quantum–Mechanical Tunnelling

Tunnelling is the quantum–mechanical effect that allows electrons (and other particles) to pass through barriers that ostensibly are too high to allow them to pass. An electron may, for a short time, “borrow” the energy required to pass through a potential barrier, allowing it to emerge from the other side. This process is utilised in many devices such as tunnelling microscopes, and it is a crucial enabler in allowing nuclei to approach each other close enough to fuse together in the cores of stars.

Tunnelling is a phenomenon deriving from the Heisenberg uncertainty principle,

$$\Delta x \Delta p \geq \frac{\hbar}{2} \tag{1}$$

It can occur in systems whereby a narrow “barrier”, consisting of a medium in

which the potential energy of the particle under consideration is higher than its total energy, is sandwiched between two other regions in which the particle's potential energy is lower than its total energy (assuming we are treating the particle as a localised object). It is an effect of the wave nature of particles, and is impossible in classical theory. The total energy of the particle is given by

$$E = \frac{1}{2m}p^2 + V(x). \quad (2)$$

Thus, apparently, classically, in the region where $V(x) > E$ the particle's kinetic energy is negative and its momentum p is imaginary. However, the uncertainty principle (Eqn 1) tells us we can't know both the position (and, therefore, the potential energy) and the momentum (leading to the kinetic energy) exactly. If we know the particle to be in a position x then the uncertainty in p tells us that the kinetic energy must be greater than the difference between the height of the barrier and the total energy.

Figure 1 shows a representation of the wavefunction of an electron tunnelling through a barrier. In the regions where the electron's kinetic energy is greater than its potential the wavefunction behaves in an oscillatory manner, as normal. However, in the barrier region, where the electron classically would not have enough energy to venture, the wavefunction shows an exponential decay. Hence, the wavefunction on the opposite side of the barrier is much reduced in amplitude, which reflects the probability of the electron appearing on that side. The form of the wavefunction outside of the barrier depends on whether the electron is confined to a well, and if so, its width and depth.

Analytical modelling of tunnelling through simple barriers is inherently possible by various methods, outlined later in this chapter; however, more complex potentials result in difficult differential equations that may be impossible to solve even in principle. For this reason quick and reliable methods for numerically solving for tunnelling probability are required.

1.3 Examples of Electron Transfer

The scanning tunnelling microscope makes use of the phenomenon of quantum-mechanical electron tunnelling in order to operate. The sharp tip of the microscope is placed near the sample being examined, but with a gap consisting of

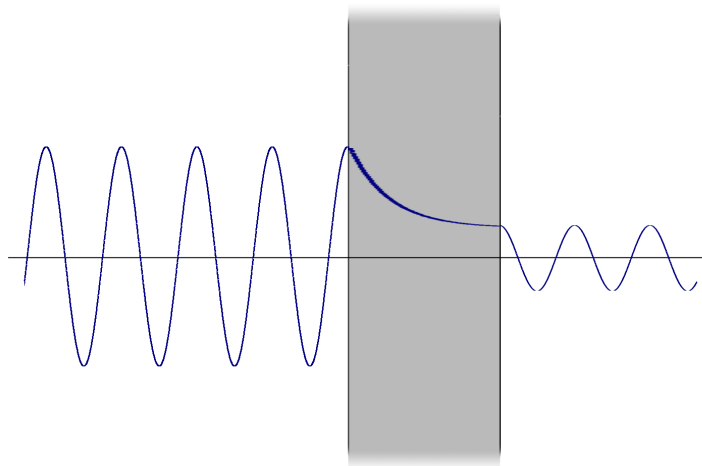


Figure 1: Representation of the wavefunction of an electron showing tunnelling through a potential barrier of energy greater than the electron. The wavefunction displays exponential decay within the barrier, and the usual form outside.

liquid, gas or vacuum. Applying a voltage across the gap can cause electrons to tunnel between the tip and the sample, dependent on the width of the gap (and other factors such as tunnelling medium and material of the sample, in particular the density of states). The current which flows is dependent entirely on the tunnelling probability, and thus upon the tip–surface distance, allowing an image to be produced as the tip scans across the surface. Whilst vacuum gaps can be modelled mathematically, more complex media such as gases or layers of water molecules require numerical modelling, due to the complex electrical potential environments.

In recently produced silicon semiconductor microchips current leakage has become a major problem, that is threatening to hold back the development of ever–faster electronics. As the transistors become smaller, the distances between them shrink concomitantly—*i.e.* the dielectric barrier within transistor components is reduced. This increases the risk of electrons tunnelling through components. Thus, being able to model the tunnelling of electrons from one material through another of arbitrary shape is of great utility in planning integrated circuit designs in order to minimise this kind of behaviour.

In many biological reactions electron tunnelling [1] (and often proton or some-

times even nuclear tunnelling) plays a part. In particular, the role of tunnelling has been well researched in enzymes, the protein catalysts that allow biological reactions to happen at useful rates. The paper by Dutton *et al.* [2] lists many recent articles in which the research into quantum tunnelling of electrons and nuclei is well documented and explored. For example, in the electron transport chain in mitochondria, electrons are transferred from site to site to allow protons to travel through the inner mitochondrial membrane, ultimately producing ATP. At each transfer, the electron must tunnel some distance to allow the transport to continue. The distances and barrier potentials vary at each site [3].

Water is a crucial solvent in biological reactions. Lin *et al.* [4] describe how layers of water sandwiched between proteins can provide a transient boost in the electron tunnelling probability. By assuming the *static medium approximation*—*i.e.* that the tunnelling process will complete in less time than it takes for the water molecules to move to a less favourable configuration—we could feasibly use path integral Monte Carlo methods to study such situations. While the methods employed in this thesis cannot provide a model of the potential environment in such a system, they should allow straightforward study of these systems if the potential can be mapped by another method and then approximated by a model. Usually such potentials will either be assembled using basic geometry to approximate the system under study, or sampled from equilibrium molecular dynamics simulations.

1.4 Previous Methods for Modelling Tunnelling

1.4.1 WKB Method

The Wentzel–Kramers–Brillouin (WKB or WKBJ) method was developed by Brillouin [5], Kramers [6] and Wentzel [7] in 1926, based on work by Jeffreys in 1924 [8]. It is a quasi-classical method that can be applied to find an approximate solution of the time-independent Schrödinger equation

$$-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} \Psi(x) + V(x) \Psi(x) = E \Psi(x) \quad (3)$$

by re-writing the wavefunction as the exponential of another function, that allows us to separate the Schrödinger equation into real and imaginary parts. From there

the functions relating to the real and imaginary parts are expanded as a power series in \hbar —which is known as the semiclassical approximation—leading to two separate cases: the amplitude varies slowly compared to the phase, or the phase varies slowly compared to the amplitude.

$$B_0(x) = \pm \sqrt{2m(E - V(x))} \quad (4)$$

$$A_0(x) = \pm \sqrt{2m(V(x) - E)} \quad (5)$$

The first case (Equation 4) corresponds to the region in which the total energy of the particle is greater than the potential energy (the classically allowed region) in which we observe the oscillatory wavefunction. The second case (Equation 5) corresponds to the opposite: the potential energy is greater than the total energy of the particle. This is the quantum tunnelling regime, in which we see an exponential decay of the wavefunction. A fuller treatment is given in Appendix A.

The WKB solution is approximate rather than exact, and furthermore the method does not lend itself to complex potentials—regions are either “classical” or “quantum” with regards to the magnitude of the potential energy compared to the total energy, and these regions are joined with boundary conditions. In real systems the boundaries between the regions may not be well-defined. The method does not allow for regions in which the phase and amplitude vary in roughly equal proportion.

1.4.2 Path Integral Instanton Approach

Instantons in quantum mechanics are classical solutions to equations of motion with a finite, non-zero action—the solutions can be thought of as critical points of the action which can be local maxima, local minima, or saddle points. They can be used to calculate the transition probability of a particle tunnelling through a potential barrier. Using path integrals, we can engage an instanton approach to find the dominant tunnelling path, which leads to an approximate result. In the path integral formulation, the transition amplitude is

$$K(a, b; t) = \langle a | e^{-\frac{iHt}{\hbar}} | b \rangle = \int d[x(t)] e^{\frac{iS[x(t)]}{\hbar}} \quad (6)$$

with action

$$S = \int_{t_a}^{t_b} dt \left(\frac{m}{2} \left(\frac{dx}{d\tau} \right)^2 - V(x) \right) \quad (7)$$

By applying a Wick rotation, $it \rightarrow \tau$, in order to make the integral convergent, one gets

$$K_E(a, b; \tau) = \langle a | e^{-\frac{H\tau}{\hbar}} | b \rangle = \int d[x(\tau)] e^{-\frac{S_E[x(\tau)]}{\hbar}} \quad (8)$$

with the Euclidean action

$$S_E = \int_{\tau_a}^{\tau_b} d\tau \left(\frac{m}{2} \left(\frac{dx}{d\tau} \right)^2 + V(x) \right) \quad (9)$$

Due to the Wick rotation the potential energy changes sign $V(x) \rightarrow -V(x)$, and therefore the potential minima transform to potential maxima. This means that a potential double-well system turns into a “double-peak” system instead.

1.4.3 Absorbing Boundary Condition Green’s Functions

In the past absorbing boundary condition Green’s function (ABCGF) methodologies have been used. A Green’s function is a type of function used to solve partial differential equations with boundary conditions or initial conditions. They can be used as propagators in quantum mechanics, where the Green’s function of the Hamiltonian is a useful concept.

Absorbing boundary conditions in general are boundary conditions that are applied to a system of differential equations in order to limit the physical size of the computation. Ideally the boundary conditions applied would minimise the amplitude of wave reflection while closely approximating the free-space (no boundary) solution.

The ABCGF approach involves propagating the wavefunction through the barrier. The approach is grid-based—hence the boundary conditions—and relies on evaluating Green’s functions efficiently and accurately. Mosyak and Nitzan [9] describe the process in some detail. The approach is mathematically complex and far from intuitive. Since it is grid-based it requires a carefully chosen grid spacing that can be affected by complex or rapidly fluctuating potentials.ki

1.5 Proposed Method

The method proposed here has been developed to be accurate and intuitive to understand, while being versatile.

A particle’s path in the path integral formulation can be represented as a sum of all possible histories for a particle. The Trotter theorem shows that a particle history can be represented by a series of jumps or slices, with the solution becoming exact as the number of slices approaches infinity. These can in turn be modelled by a classical system of beads joined by Newtonian springs. A statistical collection of many paths provides the quantum amplitude of the particle for a particular scenario. Monte Carlo simulations are used to generate possible paths, with various methodologies tested.

Using this model, not only can the particle’s quantum amplitude (or most likely path) be ascertained, but by application of the method of expanded ensembles the free energy associated with the path can be calculated. The free energy is required in order to calculate the tunnelling splitting, which in turn allows us to calculate the tunnelling probability. Values of tunnelling probability can be used to estimate reaction rates for a system. In order to be able to compute enough decorrelated paths quickly enough to find an accurate value for free energy, a multislice method has been used, in which an entire path is generated in one pass and tested for acceptance or rejection, rather than generating paths based upon previous configurations. Fourth-order methods have been explored in a bid to further reduce compute time, and the code has been written to use the MPI parallel computing environment, with extremely favourable scalability over multiple CPU cores.

If a potential environment can be cheaply modelled, then the method should remain competitive with other approaches, especially given access to a high-powered parallel computing resource. The statistical nature of the approach lends itself particularly well to parallel or distributed computing resources. Precision can be improved to an arbitrary level by devoting more time to the computation.

1.6 Monte Carlo over Molecular Dynamics

There are fundamental reasons why path integral Monte Carlo (PIMC) might be preferred over path integral molecular dynamics (PIMD). Firstly, the only approximation introduced in the PIMC approach is the factorisation described in Chapter 3. The act of slicing the path into small sections like a polymer is an approximation, but it is one that becomes exact as the number of slices approaches infinity. Therefore it is an approximation that is controllable, and thus we can choose a number of slices in order to obtain an arbitrarily accurate reconstruction. Whatever the choice regarding splitting, the sampling is nonetheless exact (assuming it is ergodic) and does not rely on numerical approximation. Contrary to this, the equations of motion used in PIMD rely on numerical approximation, and so can not be exact.

In PIMD we need to allow control over the temperature of the simulation (the canonical ensemble) rather than the energy (the microcanonical ensemble). For this various methods can be used. The Berendsen thermostat [10] can be used, but it does not generate a correct canonical ensemble. A popular method is to employ the Nosé–Hoover thermostat, which generates the correct canonical ensemble, but it is known that PIMD with this approach can produce non-ergodic trajectories due to the presence of stiff nearest-neighbour harmonic forces, and therefore the available phase space is not fully sampled [11]. Martyna, Klein and Tuckerman [12] showed how to use a chain of Nosé–Hoover thermostats for each degree of freedom. Unfortunately this requires $3N$ degrees of freedom over the PIMC method. By contrast, the temperature is directly controlled as a feature of the PIMC method, without recourse to thermostat methods or extra degrees of freedom.

The PIMC method also allows for an easy approach to ensuring ergodicity, since the chains can be generated in one pass using the Lévy bisection method, as outlined in Chapter 7. With exact sampling and direct control over the temperature the PIMC approach holds advantages over the PIMD approach that make furthers development in this area attractive.

Part III

Theory and Methods

2 Path Integral Formulation

The path integral formulation of quantum mechanics was developed by Richard Feynman in 1948 [13] [14], building on previous work by Paul Dirac [15]. Previous classical methodologies defined one unique history, or ‘path’ for a particle, with a well-defined action which was based upon a minimised action. Feynman showed how to replace this approach with one in which an infinite number of paths is integrated to produce a probabilistic quantum amplitude. This quantum amplitude represents the possible motion of the particle through space and over time.

2.1 Feynman’s Postulates

Feynman defined three postulates from which he attempted to recover all of quantum mechanics:

- The probability for an event is given by the squared length of a complex number called the “amplitude”.
- The amplitude is given by adding together the contributions of all the histories in configuration space.
- The contribution of a history to the amplitude is proportional to $e^{\frac{iS}{\hbar}}$, where S is the action of that history (\hbar is set equal to 1 when we use atomic units).

In other words: when a particle moves through space, it visits every point in space with some probability. Every path conceivable is possible (Figure 2), but some are more probable than others. When we include all possible paths (weighted by the likelihood of that particular history occurring), both the direct ones and the convoluted and massively indirect ones, we build up a probability map which shows us the most likely locations of the particle. Although in

the original formulation the weightings can be both positive and negative, cancellation effects mean that eventually only the highly probable (positive) paths eventually contribute to the total, so long as we include even the paths that would be impossible by classical standards.

2.2 Action

The action is a function of the path or trajectory of a particle, that gives a real number with dimension $E \times t$. In general in classical physics the path taken by the particle is the one for which the action is stationary (*i.e.* minimised). The action in classical mechanics is given by

$$S = \int dt L \quad (10)$$

where L is the Lagrangian. In classical mechanics, the Lagrangian is given by

$$L = T - V \quad (11)$$

where T is the kinetic energy and V is the potential energy. Thus, more probable paths will have a lower action, and less probable paths will have a higher action. Paths that intersect high-value regions of the potential energy surface and paths that cover an unnecessarily large spatial distance will have a high action.

The similarity, in essence, of the Hamiltonian

$$H = T + V \quad (12)$$

to the Lagrangian is obvious (Hamiltonian mechanics was originally a re-formulation of Lagrangian mechanics and the Hamiltonian is the Legendre transform of the Lagrangian) and we can write the action, using the Hamiltonian, as

$$S = \int dt H \quad (13)$$

Thus, we are aiming to develop a system whereby we can integrate the Hamiltonian between two times t_1 and t_2 along a possible path in order to find the action. Once we can do this, the path integral methodology becomes practical. In this approach to generating paths, rather than assigning a probability to each and every possible path, we instead generate preferentially the more probable paths; the weighting occurs naturally as we include more probable sections of path more frequently than less probable ones.

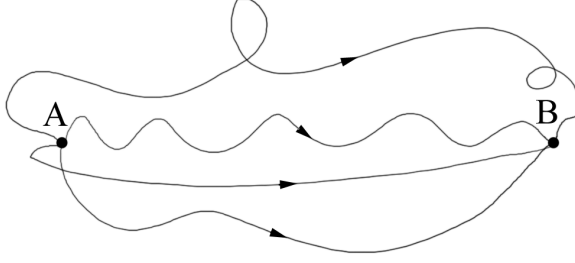


Figure 2: An example of particle paths from A to B. The paths can follow any trajectory; we assign a probability to each one before summing over all paths. Note that physically there is no reason to include curved paths in free space. Curved paths reflect possible interactions with surrounding particles by way of the potential energy surface.

2.3 Transforming from Quantum Mechanics to Statistical Mechanics

If we take the Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} |\psi(t)\rangle = \hat{H} |\psi(t)\rangle \quad (14)$$

then given the state of the system at some initial time, say $t = 0$, we can integrate to find the state at some subsequent time t . If the Hamiltonian is independent of time then

$$|\psi(t)\rangle = \exp\left(\frac{-i\hat{H}t}{\hbar}\right) |\psi(0)\rangle \quad (15)$$

The operator

$$U = \exp\left(\frac{-i\hat{H}t}{\hbar}\right) \quad (16)$$

is called the time evolution operator. We can perform a Wick rotation by substituting the inverse temperature β for the imaginary time:

$$\beta = \frac{1}{k_B T} = \frac{it}{\hbar} \quad (17)$$

Thus, the time evolution operator becomes

$$e^{-\beta \hat{H}} \quad (18)$$

The quantum mechanical problem has become one of statistical mechanics—the time evolution operator now resembles the canonical partition function

$$Z = \sum_i e^{-\beta E} \quad (19)$$

Now if we can calculate the Hamiltonian to find the energy of a path, we can calculate the relative weights and sum over paths to find the probability amplitude of the system.

2.4 Practicalities

This formulation can be applied to the problem of electron transfer through various potentials—for example, an electron moving from one part of a protein to another through a thin layer of water can be simplified to the problem of generating an appropriate selection of possible paths and summing to find transition probabilities. This entails being able to model the external potential through which the particle tunnels, and generate and select paths by their action.

Possible paths can include any motion of the electron between the start and end positions, subject to certain constraints that we may wish to apply (*e.g.* we may define some regions of infinite potential outside of the simulation that the electron cannot visit).

Obviously, we cannot generate the infinity of paths that the path integral approach requires, but we can generate an arbitrarily large number—the answer converges towards the exact result with an increasing number of paths.

3 Trotter Splitting

In Section 2 it was shown that the quantum-mechanical time evolution operator can be made to resemble the canonical partition function by performing a Wick rotation (Equation (14) to Equation (19)). To continue with the problem we have to integrate over all possible paths. Generating and selecting possible quantum mechanical paths for an electron could be a rather difficult task, since we need to take into account all possible paths and weight them; fortunately there is a theorem called the Lie–Trotter product formula [16] that makes the problem somewhat simpler by allowing the propagator to be factorised, enabling us to calculate the energy of a given path.

3.1 Partition Function

The classical partition function can be written as

$$Z(\beta) = \int e^{-\beta H(\mathbf{p}, \mathbf{q})} d\mathbf{p} d\mathbf{q} \quad (20)$$

where the vectors \mathbf{p} and \mathbf{q} are momentum and position of the particle, and H , the Hamiltonian, is

$$H(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{3N} \frac{\mathbf{p}_i^2}{2m_i} + V(\mathbf{q}) \quad (21)$$

Note that the Hamiltonian consists of two parts: the first term on the right-hand side of Equation (21) is the *kinetic* part, and the second term is the *potential* part. This will become significant shortly.

The partition function is an extremely important quantity as it determines the way that the overall energy is shared, or partitioned, between the numerous states of the system. From the partition function for a classical system, many thermodynamic quantities can be calculated. As we have seen, the path integral can be made to resemble the canonical partition function by performing a Wick rotation—*i.e.* switching between real and imaginary time.

3.2 Quantum Mechanical Partition Function

Quantum mechanically the continuous integral is replaced by a sum over all states, with a degeneracy factor to take account of the identical energies of some

levels. The degeneracy factor g_i can be ignored as we are counting up all levels individually. The quantum analogue of Equation (20) can be obtained by recalling that the probability of a given state being occupied is $e^{-\beta E_i}$ (at thermal equilibrium). Writing the quantum partition function out in full, and translating into Dirac notation we have:

$$Z(\beta) = \sum_i g_i e^{-\beta E_i} \quad (22)$$

$$Z(\beta) = \sum_i \langle i | e^{-\beta \hat{H}} | i \rangle = \text{Tr}(e^{-\beta \hat{H}}) \quad (23)$$

This is the trace of the thermal density matrix, and is an important quantity to compute in path integral calculations. In principle, all the properties of a quantum system in equilibrium can be obtained from the density matrix. For a free particle in continuous space, the above can be re-written as

$$\int d\mathbf{R} \langle \mathbf{R} | e^{-\beta \hat{H}} | \mathbf{R} \rangle \quad (24)$$

At this point we introduce the Trotter formula, a very important equation that states:

$$e^{-\beta(T+V)} \approx \lim_{M \rightarrow \infty} [e^{-\tau T} e^{-\tau V}]^M \quad (25)$$

where

$$\tau = \frac{\beta}{M} \quad (26)$$

This is essentially the basis of the path integral Monte Carlo approach. More specifically, it can be shown that

$$\int d\mathbf{R} \langle \mathbf{R} | e^{-\beta \hat{H}} | \mathbf{R}' \rangle \langle \mathbf{R}' | e^{-\beta \hat{H}} | \mathbf{R} \rangle = \langle \mathbf{R} | e^{-(\beta_1 + \beta_2) \hat{H}} | \mathbf{R} \rangle \quad (27)$$

and from this result

$$\begin{aligned} \langle \mathbf{R} | e^{-\beta \hat{H}} | \mathbf{R} \rangle = \\ \int d\mathbf{R} d\mathbf{R}'' d\mathbf{R}''' \dots \langle \mathbf{R} | e^{-\frac{\beta \hat{H}}{M}} | \mathbf{R}' \rangle \langle \mathbf{R}' | e^{-\frac{\beta \hat{H}}{M}} | \mathbf{R}'' \rangle \langle \mathbf{R}'' | e^{-\frac{\beta \hat{H}}{M}} | \mathbf{R}''' \rangle \end{aligned} \quad (28)$$

which represents a splitting of the free particle path into a number M of shorter, connected paths, or time slices. This is known as Trotter splitting. It can also be shown that

$$\langle \mathbf{R} | e^{-\beta \hat{H}} | \mathbf{R}' \rangle = e^{-\beta(\mathbf{R}-\mathbf{R}')^2} e^{-\frac{\beta}{2}(V(\mathbf{R}')+V(\mathbf{R}))} + \mathcal{O}(\beta^3) \quad (29)$$

which is a separation or *decomposition* of the kinetic and potential components of each time slice. This object is known as the propagator and it defines the degree and quality of splitting of the path in imaginary time. The consequence of the Trotter splitting is that the entire path—consisting of a series of time slices—can be represented by a series of points or ‘beads’ joined by classical harmonic springs, a little bit like a polymer, sitting in a potential $V(\mathbf{R})$. This transforms the task of generating paths into a problem of classical dynamics, statistics and ergodicity. The density matrix for a path or chain consisting of connected beads can be easily computed as the product of all the energy terms,

$$e^{-\tau(\mathbf{R}-\mathbf{R}')^2} e^{-\frac{\tau}{2}[V(\mathbf{R})+V(\mathbf{R}')] } e^{-\tau(\mathbf{R}'-\mathbf{R}'')^2} e^{-\frac{\tau}{2}[V(\mathbf{R}')+V(\mathbf{R}'')] } e^{-\tau(\mathbf{R}''-\mathbf{R}''')^2} e^{-\frac{\tau}{2}[V(\mathbf{R}'')+V(\mathbf{R}''')] } \dots (30)$$

where $\tau = \frac{\beta}{M}$ and $\beta = \frac{1}{kT}$ as usual.

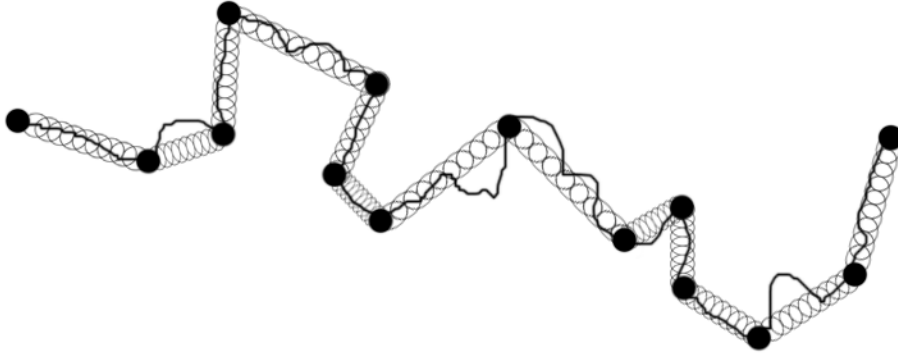


Figure 3: The Trotter theorem allows us to split the path into “time slices”—we can represent the electron path by modelling beads connected by springs.

3.3 Discussion

We have now completely mapped the quantum system onto a classical system. This is known as a quantum-classical isomorphism, and it is the basis of the path

integral Monte Carlo (PIMC) approach. By fixing the first and last beads in the chain at R_A and R_B , we can represent the path of an electron from R_A to R_B through some potential $V(\mathbf{R})$. The individual beads are replicas of the simulated particle. By splitting the path into a higher number of slices, we obtain a better representation of the path taken. The propagator above is second-order (first- and second-order propagators amount to the same thing for chains of beads), but higher-order propagators exist and will be discussed at length. A higher-order propagator should allow fewer beads to be used to attain a particular level of accuracy, thus reducing the computer time required.

Rather than generating all possible paths—an infinite number, as detailed in the path integral formulation—we can now simply generate a large number of paths. Instead of having to rely on interference to cancel out the less likely paths, leaving the relatively highly positive paths to contribute to the final total, we can just generate paths with a probability that is related to the weighting that the history would have in the full formulation.

The physical model of beads connected with springs is directly related to the weighting of possible histories. The most likely paths are those in which the action is minimised—which are the very same bead configurations in which the springs are least extended. Most generated paths will be of minimal energy, but a very few will consist of convoluted paths in which the springs are stretched into a high-energy state. This reflects the likelihood of these paths occurring in the history of the particle.

The temperature of the simulation is set by changing the value of $\beta = \frac{1}{kT}$. For a simulation at $300K$, β should be around 1052.6. The exact temperature is adjustable, but the temperature range is limited by the tunnelling splitting calculation (discussed in Chapter 9). Note that τ is directly proportional to β . The value of λ is related to the strength of the springs holding the beads together and is given by

$$\lambda = \frac{\hbar^2}{2m_e} \quad (31)$$

where m_e is the electron mass. A value of $\frac{1}{2}$ is appropriate for an electron in atomic units, and this value is used in the code.

Extracting information about the particle such as free energy and probability amplitude requires the application of some operation over all the beads in the

chain. For example, the probability amplitude is obtained by generating many chains, and creating a normalised histogram of all of the bead positions in all generated chains.

4 Monte Carlo Methods

4.1 History

Monte Carlo methods for solving problems have been around for far longer than electronic computers have existed. The essence of Monte Carlo is to randomly sample many possible states and integrate them to find the most probable states of a system. Often some rejection approach is used whereby sampled states that fail to meet specific criteria are ignored. While this lends itself to the rapid processing of digital computers, the methods have been used in some form for many centuries.

Primitive versions of random sampling were employed to some extent long before the advent of electronic computing machines, with stochastic sampling methods being performed mechanically or by hand (*e.g.* Buffon's needle, a technique for calculating an approximation to the value of π), but the technique only really came into its own with the development of the electronic computer and its ability to perform repetitive tasks extremely rapidly. Stochastic methods were first used seriously to calculate random neutron diffusion, with the calculations performed by a combination of humans and machines, and later on, solely by machines.

Von Neumann helped to lay the foundations for Monte Carlo methods when he established the mathematical basis for probability density functions (PDFs), inverse cumulative distribution functions (CDFs), and pseudorandom number generators. The methods were used extensively on early versions of electronic computers, such as those used in the Manhattan Project and later on at Los Alamos in the race to build the hydrogen bomb. Nicholas Metropolis did much work on Monte Carlo algorithms during this time, including development of the eponymous Metropolis (generalised to Metropolis-Hastings) algorithm.

4.2 Overview

The essence of Monte Carlo simulation is to avoid the necessity of having to calculate the probability of every possible state (or event) of the system (of which there could be a vast number) by instead randomly choosing a small selection from all the possible states, which will contribute towards a representative sum.

The key to an efficient simulation is to have the same probability of selecting a state or class of states as the probability of that state (or class of states) occurring. In other words, the frequency at which that state is included in the sum must reflect the probability of that state occurring.

4.3 Markov Chain Methods

A Markov chain, named after Andrei Andreyevich Markov (1856—1922), is a process in which the future state depends only on the current state, and not on how the current state was arrived at. The transition from the current state s_i to a possible future state s_j holds a probability p_{ij} , regardless of any previous jumps or probabilities. Any system in which the future state depends upon how the current state was arrived at cannot be a Markov chain.

Note that the ‘chains of beads’ idea explored in the chapter on Trotter splitting is not directly connected to the ‘Markov chain’ concept, and the similarity in name is coincidental.

4.4 Metropolis Monte Carlo

The Metropolis rejection algorithm, as initially devised by Metropolis, is a way to draw random samples from a distribution that might otherwise be difficult to sample. It is essentially a random walk, visiting individual states along the way. It is based upon a Markov process (*i.e.* the transition to the next state depends on the probability of a jump from the current state, but not how the current state was arrived at). The step from one state to the next depends on a transition rule or probability $P(s_i \rightarrow s_j)$, chosen so that the distribution of states matches a required distribution $\pi(s)$. The transition rule should be ergodic (*i.e.* given enough time, any given state can be visited more than once) and it should satisfy detailed balance

$$\pi(s_i)P(s_i \rightarrow s_j) = \pi(s_j)P(s_j \rightarrow s_i) \quad (32)$$

In other words, the transition probability $P(s_i \rightarrow s_j)$ represents how likely a transition is from one state to another, while the distribution $\pi(s)$ represents how likely the system is to be in that particular state (at equilibrium). In a system

with two states, A and B , $\pi(A)$ might be 0.3 while $\pi(B)$ might be 0.7. In this case, the values $P(A \rightarrow B) = 0.7$ and $P(B \rightarrow A) = 0.3$ would satisfy detailed balance. Of course, since *something* must happen each step, the transition probabilities should also satisfy

$$\sum_j P(s_i \rightarrow s_j) = 1 \quad (33)$$

The transition probability $P(s_i \rightarrow s_j)$ can be split into two parts

$$P(s_i \rightarrow s_j) = A(s_i \rightarrow s_j)T(s_i \rightarrow s_j) \quad (34)$$

$A(s_i \rightarrow s_j)$ is the probability that the step will be accepted, and $T(s_i \rightarrow s_j)$ is an arbitrary transition; it defines how the next state s_j will be selected at state s_i .

Re-writing (32) using (34), we get

$$\pi(s_i)A(s_i \rightarrow s_j)T(s_i \rightarrow s_j) = \pi(s_j)A(s_j \rightarrow s_i)T(s_j \rightarrow s_i) \quad (35)$$

$$\frac{A(s_i \rightarrow s_j)}{A(s_j \rightarrow s_i)} = \frac{\pi(s_j)T(s_j \rightarrow s_i)}{\pi(s_i)T(s_i \rightarrow s_j)} \quad (36)$$

Usually, $T()$ is assumed to be symmetric, so that $T(s_i \rightarrow s_j) = T(s_j \rightarrow s_i)$ and

$$\frac{A(s_i \rightarrow s_j)}{A(s_j \rightarrow s_i)} = \frac{\pi(s_j)}{\pi(s_i)} \quad (37)$$

In the canonical ensemble

$$\frac{\pi(s_j)}{\pi(s_i)} = \frac{e^{-\beta V(s_j)}}{e^{-\beta V(s_i)}} = e^{-\beta[V(s_j) - V(s_i)]} \quad (38)$$

This, however, does not give us any detail on the exact form of $A(s_i \rightarrow s_j)$, only the ratio of acceptance probabilities. We can use the fact that we may choose how the overall probability is split. Metropolis and co-workers proposed

$$A(s_i \rightarrow s_j) = e^{-\beta[V(s_j) - V(s_i)]} \quad \text{if } V(s_j) > V(s_i) \quad (39)$$

$$A(s_i \rightarrow s_j) = 1 \quad \text{if } V(s_j) \leq V(s_i) \quad (40)$$

In words: if the energy of the new state is less than the energy of the old state, the move is always accepted, otherwise the acceptance probability is equal to the ratio of the two energies.

Now, for our system the system states are defined by each individual bead position, which is a continuous variable in space (as we are working with essentially free particles). Therefore, we need to recast (34) with continuous variables:

$$P(s_i \rightarrow s_j) = A(s_i \rightarrow s_j)T(s_i \rightarrow s_j) \Rightarrow P(R \rightarrow R') = A(R \rightarrow R')T(R \rightarrow R')(41)$$

The acceptance probability is not changed by this switch, since we have already written it as a function of potential, which can in turn be a function of position. A common choice for $T()$ is to allow any movement within a pre-defined step size, S . Thus,

$$T(R \rightarrow R') = \frac{1}{V} \text{ if } |R - R'| \leq S \quad (42)$$

$$T(R \rightarrow R') = 0 \text{ if } |R - R'| > S \quad (43)$$

In this scheme, particle movements are made randomly within a box of side length $2S$. The details of this are covered in section 5. There are other schemes for moving beads, including using movements drawn from a normal distribution, using the potential landscape to plot moves, and placing multiple beads using a chain bisection or Brownian bridge algorithm.

5 Single-Slice Path Integral Monte Carlo

The simplest method for moving beads in the chain representing the electron path is to adjust slices individually in sequence. There are several approaches—some more complex—to moving beads, which will be covered later, but for testing purposes it is advantageous to have a “simplest case” implementation for comparison. Single bead movement [17], with individual acceptance/rejection testing, is the simplest way to implement the Metropolis Monte Carlo algorithm in this case, and it always converges to the correct distribution. While there are issues with sampling efficiency, such as unwanted correlation between successively generated chains, and possible high rejection rates, it stands well as a baseline case.

5.1 Single-Slice Basic Method

Each move, a bead is chosen, and its current position and the total chain energy is stored. The chosen bead is moved by a vector drawn from a uniform distribution between $-S$ and S in each dimension, where S is the pre-defined maximum step size. This means that the bead is free to move anywhere within a cube of side length $2S$ with equal probability. Once the move has been applied, the total chain energy is calculated using

$$E = \prod_{i=0}^M \left[\exp \left(-\frac{(r_{i+1} - r_i)^2}{4\lambda\tau} \right) \exp(-\tau V(r_i)) \right] \quad (44)$$

which can be re-written

$$E = \exp \left[-\sum_{i=0}^M \left(\frac{(r_{i+1} - r_i)^2}{4\lambda\tau} + \tau V(r_i) \right) \right] \quad (45)$$

where

$$\tau = \frac{\beta}{M} \quad (46)$$

$$\lambda = \frac{\hbar^2}{2m} \quad (47)$$

and compared to the stored value of the chain energy from before the bead was moved. The acceptance/rejection step is performed using

$$P_{acc} = \min \left[1, \frac{E_{old}}{E_{new}} \right] \quad (48)$$

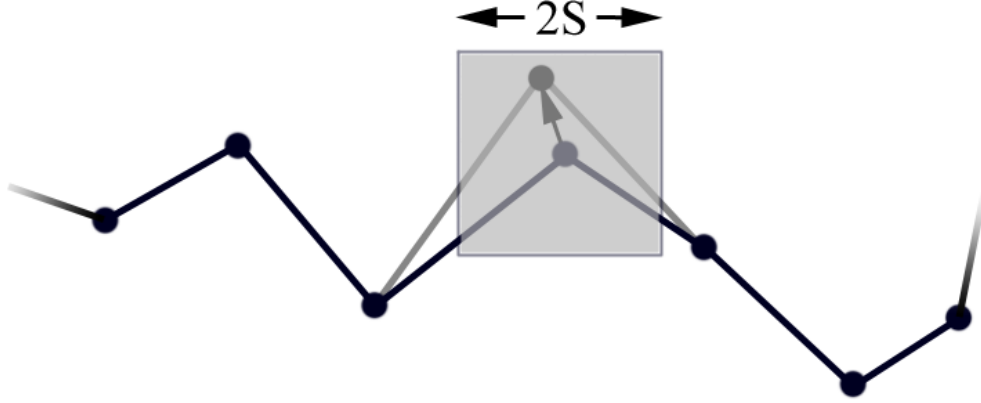


Figure 4: The single-slice basic method. Movements are made within a box of side length $2S$ where S is a user-defined step-size.

If the new energy is lower than the old energy, then the ratio is greater than one and the movement is accepted with certainty, otherwise it is accepted with probability P_{acc} . The step size S can be adjusted in order to bring the average acceptance rate to an optimal value of around 50%, as suggested by Metropolis *et al.*. A larger step size will reduce the acceptance rate but increase the speed at which chain configurations are explored, while a smaller step size will have the opposite effect, increasing the acceptance rate while slowing the progression of the random walk through configuration space. When a move is rejected the bead remains in its current position, but it still contributes to the overall averages in the same way as bead moves that are accepted.

The simplest way to select beads is to start at the first (lowest index) bead in the chain and move all of the subsequent beads sequentially until the last free bead has been moved. In our system, the first and last beads are fixed so these are omitted from the movement algorithm, although their positions play a part in calculating the kinetic component of the energy equation for adjacent beads.

5.2 Implementation

The data for a single bead is stored in a C language *struct*, so that a chain can be easily constructed and iterated with a simple array of *structs*. It was decided early on that the most straightforward way to factor the code was to place all the code required to generate a new chain using a single method into one function. Hence, the single-slice basic method can be used to generate a new chain with a single function call. In the same manner, a chain can be generated using any of the other methods discussed later with a single function call, since a chain is the single smallest useful amount of data that can be generated.

Since the energy can be computed with a summation, there is in fact no need to calculate the total chain energy every move. Instead, the energy contribution (kinetic and potential) due to the current bead being moved can be subtracted from the total chain action. Calculating the bead movement vector is a simple matter of scaling a random number drawn from a uniform distribution using a good quality pseudo-random number generator. Once the bead has been moved, the kinetic component of the energy is easily calculated, while the potential component must be calculated by calling a separate predefined function of position. The energy contribution due to the bead's new position can now be added on to the total chain energy. The acceptance probability is computed using a ratio of energies, which are both calculated using exponentials, although one exponential operation can be saved by using the fact that

$$\frac{e^A}{e^B} \equiv e^{A-B} \quad (49)$$

The acceptance/rejection step is actually performed by comparing P_{acc} to another random number R drawn from a uniform distribution. If $P_{acc} > R$ then the move is accepted—in which case, nothing need be done until the next bead move. If the test fails, then the old bead position and chain energy is restored from the previously saved value.

5.3 Sampling

The single-slice method should converge towards sampling the required distribution exactly, as the number of beads used is increased. If only one free bead is used in zero external potential, the histogram of visited co-ordinates will be a Gaussian

curve, which is useful for testing purposes. The basic single-slice method copes reasonably well with discontinuities in the external potential as the movement depends only on a pseudo-random number and the acceptance/rejection step depends only on the scalar value of the effective potential field. It's important to note that the *effective* potential includes a contribution from the springs joining adjacent beads, and therefore even with zero external potential many moves can still be rejected when beads stray too far from the endpoints of the chain.

5.4 Single-Slice Normal Method

A variation on the basic single-slice method is to draw the offset vectors from a normal distribution rather than a uniform distribution. This should allow for high acceptance rates with many smaller moves, but with the occasional longer step being accepted in order to help to decorrelate the system faster. The Lévy bisection method described in Chapter 7 reduces to the normal method when single beads are selected for movement.

6 Force-Bias Scheme

6.1 Inefficiency

Moving beads randomly around and testing after the fact whether the move was likely can be extremely inefficient. A bead sitting in a large potential gradient is in general just as likely to be moved into an area of higher potential as into an area of lower potential. Since moves from low into high potential areas have a high probability of being rejected, a lot of compute time can be wasted on moves that have little hope of contributing to the overall result. For this reason, it is very helpful to have some method that either predicts in some way whether a move will be valuable, or directs a move in progress so that it reflects the potential environment, preventing wasted time. The force-bias algorithm is one such method.

6.2 Force-Bias

The force-bias method was suggested by Pangali *et al.* in 1978 [18]. It extends the idea of beads being classical objects to include physical 'forces' exerted upon them by the electrostatic potential. The idea is to persuade the beads to slide down the potential gradient in addition to the usual random walk, in order to greatly increase the acceptance rate, and thus decrease convergence time. It is still, however, a single-slice sampling method.

Since force can be written

$$F = -\frac{dV(x)}{dx} \quad (50)$$

where $V(x)$ is the potential and F is the negative of the potential gradient, we can calculate the distance moved by a bead in a potential $V(x)$ in time Δt as

$$\Delta \mathbf{x} = -\Delta t \frac{dV_{eff}(\mathbf{x})}{dx} \quad (51)$$

The effective potential V_{eff} is the overall potential resulting from both the electrical potential field and the 'spring' force between the beads that is the result of the Trotter splitting.

$$V_{eff}(\mathbf{x}) = V_{field}(\mathbf{x}) + V_{spring}(\mathbf{x}) \quad (52)$$

Since the interaction energy of a bead at \mathbf{x}_i with the neighbouring bead at \mathbf{x}_{i+1} is given by

$$V_{spring}(\mathbf{x}_i, \mathbf{x}_{i+1}) = \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i)^2}{4\lambda\tau} \quad (53)$$

$$\frac{dV_{spring}(\mathbf{x}_i)}{dx} = \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i) + (\mathbf{x}_{i-1} - \mathbf{x}_i)}{4\lambda\tau} \quad (54)$$

The factor of two that results from differentiating (53) disappears when we write the interaction potential for a bead in the chain, as we need to divide by two to account for the fact that each bead has two nearest neighbours.

Thus:

$$\frac{dV_{eff}(\mathbf{x}_i)}{dx} = \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i) + (\mathbf{x}_{i-1} - \mathbf{x}_i)}{4\lambda\tau} - \frac{\tau}{2} \frac{dV_{field}(\mathbf{x}_i)}{dx} \quad (55)$$

$$\Delta\mathbf{x} = \Delta t \left[\frac{(\mathbf{x}_{i+1} - \mathbf{x}_i) + (\mathbf{x}_{i-1} - \mathbf{x}_i)}{4\lambda\tau} - \frac{\tau}{2} \frac{dV_{field}(\mathbf{x}_i)}{dx} \right] \quad (56)$$

6.3 Time

As an aside, note that we now have two different dimensions that both represent time in some sense; the Trotter splitting defines intervals of imaginary time τ , while the time step Δt facilitates the movement of the beads through configuration space. Neither of these are measures of time in the classical sense, both are simply tools to enable the propagation of some quantity. We will continue to use τ to represent Trotter time, and t to represent Monte Carlo time step time. Later we will see the introduction of other times, such as the tunnelling time, which will be an actual time in the physical sense, and for each of these we shall adopt a clearly different notation to avoid confusion.

6.4 Algorithm

The force bias scheme operates as follows:

- The potential $V_{eff}(\mathbf{x}_i)$ and gradient $\frac{dV_{eff}(\mathbf{x}_i)}{dx}$ at the chosen bead's current position is calculated

- A random vector \mathbf{w} is chosen from a normal distribution, with a variance defined by $\sigma = \sqrt{\Delta t}$
- The bead is moved by a vector \mathbf{r} which is the sum of the random vector and the distance moved due to the force

$$\mathbf{r} = \sqrt{\Delta t} \mathbf{w} + \Delta \mathbf{x} \quad (57)$$

Essentially, we have three components to a bead's movement: a 'spring step' due to the interaction forces between the beads; a 'gradient step' due to the gradient of the external potential; and a 'random step' to facilitate exploration of the chain configurations

- After the bead has been moved, the new energy is calculated according to

$$\exp \left[- \sum_{i=1}^M \left\{ \frac{(r_{i-1} - r_i)^2}{4\lambda\tau} + \tau V(r_i) \right\} \right] \quad (58)$$

- From this the acceptance ratio is determined from the standard equation for calculating acceptance ratios in Metropolis Monte Carlo

$$p = \min \left[1, \frac{T(R' \rightarrow R) \exp(-\beta V(R'))}{T(R \rightarrow R') \exp(-\beta V(R))} \right] \quad (59)$$

The transition ratios are no longer necessarily equal and are given by [19]

$$T(\mathbf{x} \rightarrow \mathbf{x}') = \frac{1}{(2\pi\Delta t)^{3/2}} \exp \left[- \frac{|\mathbf{x}' - \mathbf{x} - \Delta t \nabla V(\mathbf{x})|^2}{2\Delta t} \right] \quad (60)$$

$$T(\mathbf{x}' \rightarrow \mathbf{x}) = \frac{1}{(2\pi\Delta t)^{3/2}} \exp \left[- \frac{|\mathbf{x} - \mathbf{x}' - \Delta t \nabla V(\mathbf{x}')|^2}{2\Delta t} \right] \quad (61)$$

Clearly, the constant $\frac{1}{(2\pi\Delta t)^{3/2}}$ cancels out. We can write equation (59) in a simpler form as

$$p = \min \left[1, \frac{T(R' \rightarrow R) E_\nu}{T(R \rightarrow R') E_\mu} \right] \quad (62)$$

where $E_\nu = E_{new}$ and $E_\mu = E_{old}$. If we define

$$\mu = \frac{|\mathbf{x} - \mathbf{x}' - \Delta t \nabla V(\mathbf{x}')|^2}{2\Delta t} \quad (63)$$

$$\nu = \frac{|\mathbf{x}' - \mathbf{x} - \Delta t \nabla V(\mathbf{x})|^2}{2\Delta t} \quad (64)$$

then

$$p = \min [1, \exp\{(\nu - \mu) + (E_\mu - E_\nu)\}] \quad (65)$$

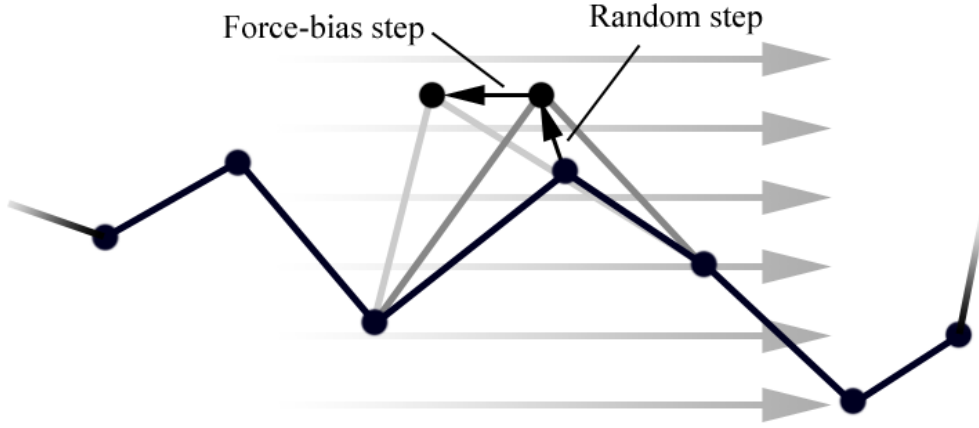


Figure 5: The operation of the force-bias algorithm. It enables the single-slice method to perform much more efficiently. The grey arrows show the direction of increasing external potential.

As in the basic single-slice method, the move is either accepted or rejected by comparison of the acceptance ratio with a uniformly distributed random number. In this case, however, if the time step is small enough, the Metropolis algorithm is not required and the distribution will converge to the correct one without rejecting any moves. For this to be valid the time step must be such that the acceptance rate would be almost 100% if the Metropolis step were to be used, and as a result the movement of the beads is extremely slow. Even an equivalent acceptance rate of 98% is too small; the 2% of moves that are rejected represent the large, unlikely moves, which have the capacity to significantly alter the result. The time step size is under user control. As before, for the Metropolis algorithm, it is advised that the time step is adjusted such that the acceptance rate is around 50%.

6.5 Discussion

The force-bias method should result in a higher acceptance rate than the uniform sampling method for a equivalent time step size; hence, it should be possible to increase the time step and keep the acceptance rate at a reasonable value, meaning that the configuration space will be explored faster, and the simulation will converge to the correct answer more quickly.

There are plenty of opportunities for optimisation in the implementation of the force-bias scheme: the gradient can be re-used in the calculation of transition ratios; many square root operations can be saved because the transition ratios require the square of a vector length; we can avoid the computationally expensive exponential in the calculation of the acceptance ratio in many cases by noting the fact that if $(\nu - \mu) + (E_\mu - E_\nu) > 0$ then $\exp\{(\nu - \mu) + (E_\mu - E_\nu)\} > 1$ and accepting the move immediately; many of the constant terms can be precomputed into one number.

One notable caveat to using the force-bias method is that potentials must be smooth and continuous, and the gradient of the potential must also be continuous—particularly if gradients are to be evaluated numerically. The basic uniform method can cope with discontinuous walls of high potential, as each bead is a single point, and the potential at its location is therefore a single scalar. With the force-bias method, we are computing the gradient of a potential field and so discontinuities will create unphysical forces. Since gradients are computed numerically using the common formula

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \quad (66)$$

we can see that if the positions $x+h$ and $x-h$ are in regions of vastly different potential, $f'(x)$ will blow up and cause the bead to be moved by a large distance in the direction of decreasing potential. While the resulting large change in spring potential will moderate this effect somewhat by preventing the acceptance of unfeasibly large moves, this means that beads that stray to within a distance h of the boundary will suddenly start to behave very differently to nearby beads that are further than h from the discontinuity.

On the other hand, if the gradient of the potential is defined analytically, the discontinuity in the potential may not show up at all, as it would be an infinitely

thin, infinitely tall spike. This would mean that the beads would behave no differently in the region of the discontinuity, and the force-bias algorithm would be of no benefit (beads moving into regions of high potential would still be more likely to be rejected, but the guiding influence of the potential gradient would be lost).

For this reason we can have no discontinuities in our potential, and complex potentials must be defined with care and smooth edges. This can make for more work when it comes to designing even simple potentials such as square wells—although the benefits of the force-bias method over the basic method should outweigh this slight disadvantage. Later chapters will cover some of the specifics of designing potentials in this manner.

The computational complexity of the force-bias method is higher than that of the basic single-slice method, and therefore the real computer time required to move one bead is greater. However, in a later chapter the force-bias method will be tested against the basic single-slice method for overall performance, where it is shown that the faster convergence of the force-bias method outweighs the increased number of operations.

7 Multi-Slice Methods

7.1 Slow Convergence

A clear disadvantage to the single-slice sampling method previously outlined—aside from the fact that it is computationally expensive to call the exponential function for every single bead move—is that it is a random walk method and each chain still depends on the positioning of the previous chain. This means that, especially with many rejections, the random walk will progress slowly and convergence will be slow. Many chains can end up being very similar to the previous chain, which is not a desirable situation—and this is why we try to choose a step-size as large as possible. Furthermore, it can be hard for chains generated in a stepwise manner to break free from deep potential wells.

A suggested solution to this problem (covered in detail by Ceperley [17]) is to place many beads at one time in a single chain configuration. There are quick methods to do this which allow us to sample the correct distribution. With a new chain generated every step, the system will decorrelate very quickly, and the configuration space should be sampled rapidly—both extremely desirable outcomes. Additionally, ergodicity is preserved, as the position of a new chain is much less dependent on the position of the current chain and deep, narrow potential wells will not hold the chain trapped for any length of time—the chain can essentially “jump” over hills in the potential energy surface.

7.2 Brownian Bridge

The Brownian bridge is a method for generating Brownian motion that is restricted to zero at two nodes positioned at the start- and end-points. This resembles our chains of beads (if approached in the correct manner). In the general case, when the start-point $W(t_1) = a$ and the end-point $W(t_2) = b$ for a time $t \in (t_1, t_2)$ the mean is

$$\mu = a + \frac{t - t_1}{t_2 - t_1}(b - a) \tag{67}$$

and the variance is

$$\text{var} = \frac{(t - t_1)(t_2 - t)}{t_2 - t_1} \tag{68}$$

With these values we can place any bead, so long as we know its relative position in the chain, using normally-distributed random numbers with the specified mean and variance.

This means that we can generate a chain in one pass using a Brownian bridge (or equivalent method, see Section 7.3), place it into our potential environment, and test it for acceptance or rejection. The acceptance test works in the same fashion as for single-slice methods, albeit modified to account for the entire chain: the chain action is computed, as before, and compared to the total action of the *previous chain*. The same criterion applies; if the action is lower, the chain is accepted, and if it is higher it is accepted with a specific probability, *i.e.*

$$P_{acc} = \min \left[1, \frac{E_{old}}{E_{new}} \right] \quad (69)$$

This wholesale generation and acceptance of chains can speed convergence by a large amount over the single-slice method—but only if the acceptance rate remains high. This may be the case for simple scenarios—situations with low or zero potential, and with plenty of slices—but if the potential is large, or complex, or there are few beads and a large distance between the end-points, then the method can fail to sustain high acceptance rates. The problem is that we are generating an entire chain without any regard for the simulation environment. If we place an unsuitable chain directly into an area of high potential, it is likely to be rejected immediately, and all the time spent creating it is then wasted. The same applies for over-stretched chains—*i.e.* those with beads separated by large distances. The previous chain will be used for the current cycle—reducing convergence speed—and we have to wait until the next chain for a chance to change the bead positions. With the single-slice method, even if many beads moves along the chain are rejected, it is likely that some will have been accepted, which advances the simulation to some degree. With the multi-slice Brownian Bridge method, if the chain is rejected, then the simulation has not been advanced at all. If the more statistically likely chains are all cutting through areas of high potential, then the rejection rate will soar and the convergence of the simulation will grind to a halt.

7.3 Lévy Construction

An alternative to the previous method of sampling a Brownian bridge is the Lévy construction. For this method to be applied, the number of free beads in the chain must be $2^n - 1$. The idea is to place the middle bead in the chain first, bisecting the chain into two regions. The beads that lie at the midpoint of the two remaining regions are placed next, further sub-dividing the chain. At each further stage, the mid-point beads are placed, until every bead has been moved, as in Figure 6. Each bisection is known as a *level*.

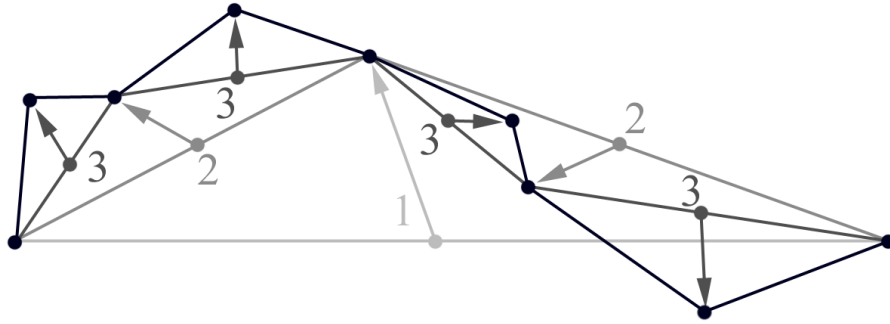


Figure 6: The Lévy construction. At each level (1—3), the chain is further bisected, until all beads have moved.

Every time a bead is placed at the centre of a bisection, it is offset by a random vector drawn from a normal distribution with variance

$$\sigma = \sqrt{\lambda(t_2 - t_1)} \quad (70)$$

where $\lambda = \frac{\hbar^2}{2m} = \frac{1}{2}$ as before, and $t_2 - t_1$ is the imaginary time interval at this level. Hence, at the highest level, $t_2 - t_1 = \tau$. The action is calculated for the entire chain; however, the action need not include the interaction energy of the beads; the correct sampling is taken care of by the bisection algorithm, and so only the external potential has any effect on the acceptance rates of the chain.

7.4 Multi-level Bisection Method

The standard Lévy construction can be further accelerated by performing early rejection on non-accepted chains. In the multi-level bisection method (also covered by Ceperley [17]), a coarse predictive chain action is calculated at each level

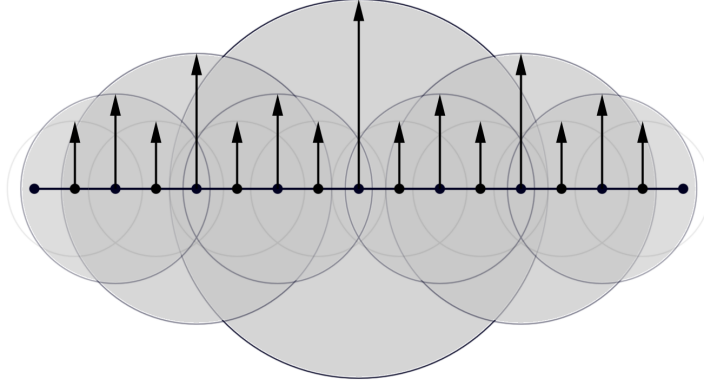


Figure 7: The Lévy construction method. The image shows the relative variances of the offset vector of each bead in a 17-bead chain. Every variance is also scaled by a parameter λ .

of the Lévy construction and compared to the previous fully accepted chain. If the action is unfavourable during the first few levels, the chain can be discarded early, avoiding wasting computational time placing beads that are not going to be accepted. Since the first (and ostensibly largest) bead move can decide the shape and therefore the energy of the entire chain, the multi-level approach can cut wasted chain generating time significantly, especially for high-potential systems.

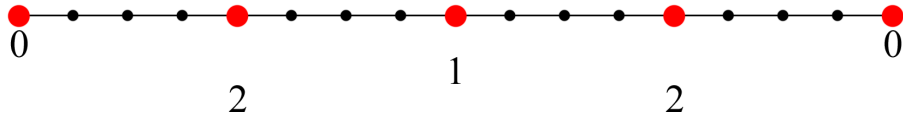


Figure 8: In the multi-level bisection method, the beads that have been moved up to and including the current level are used to calculate a coarse predictive level action. In the image the bisection has reached level two, and thus only the highlighted beads are used to calculate the action at this level.

The chain sampling proceeds past level k with probability

$$P_{acc} = \min \left[1, \frac{T_k(s_k \rightarrow s'_k) \pi_k(s') \pi_{k-1}(s)}{T_k(s'_k \rightarrow s_k) \pi_k(s) \pi_{k-1}(s')} \right] \quad (71)$$

where π_k is the action at level k [17]—although strictly speaking, the action $S = -\ln(\pi_k)$, and thus $\pi_k = e^{-S}$. Recall that k indicates the level, and s

indicates the state—the previous state being the coordinates of the slice during the last chain. Furthermore, the transition probabilities cancel, since we are not employing any force-biasing technique and so the probability of making the transition $T(s \rightarrow s')$ is the same as the probability of moving $T(s' \rightarrow s)$. In more definite terms,

$$P_{acc} = \min \left[1, \frac{E_k(s')E_{k-1}(s)}{E_k(s)E_{k-1}(s')} \right] \quad (72)$$

That is, the acceptance probability takes into account the action (exponential of the chain ‘energy’) of the previous and current levels of the current state as well as the actions of the previous and current levels of the previous state. This requires that the level actions are stored for each successfully completed chain in order that they can be used for comparison at the next level. For the level action, we can use

$$\pi_k = \sum_{i=0}^k \pi_i(s) \quad (73)$$

or, in words, the action is calculated using the beads in all levels up to and including the current one. The actual calculation is simply a sum of exponentials as per the usual. Figure 8 illustrates the beads that are included in the calculation. In practical terms this means that the level action must be stored at each level for a completed chain, and we must calculate a different value of τ (where $\tau = \frac{\beta}{M}$) taking into account the different value of M at each level (otherwise the interaction energy will be too high for the lower levels).

7.5 Section Regrowth Method

Since, when using the Lévy bisection method, entire chains are placed in one move and then tested for acceptance, rejections will result in the “new” chain being identical to the old one. This will cause a high degree of correlation, that will prevent the system from equilibrating quickly. Lower temperatures or higher potentials cause this to happen more often, reducing the effectiveness of the multi-slice method. The section regrowth method is a partial solution to this problem. This approach works by considering a sub-set of the beads in the chain as a complete chain in itself, to be placed according to the same Lévy

bisection rules as used for the entire chain. Fewer beads are placed at one time and because of this the energy change is lower and the probability of acceptance is increased. As implemented here, the code allows the user to choose the length of the section to be regrown, and then automatically selects sections of the chain in order to ensure that every bead has the chance to move at least once during the generation of a complete chain. This is more computationally expensive than generating whole chains using the Lévy bisection method, and in fact can be more expensive than even the single-slice method, but since the decorrelation is still much faster than the single-slice methods the trade-off against speed is acceptable. Choosing longer sections to regrow reduces the acceptance rate but improves the speed; the user must balance these considerations. As with many Metropolis Monte Carlo applications, the best acceptance rate to aim for may be around 50%.

8 The Method of Expanded Ensembles

8.1 Free Energy

The free energy of a system—the amount of energy available to perform work, or in other words the usable energy of a system—is, of course, a useful quantity to compute. Much of physics and chemistry is based upon finding and working with the energy of a system. The free energy is of interest in itself, and also because it can be used to calculate the tunnelling splitting, and hence the tunnelling probability, in our particular system of study.

There are two common definitions of free energy; the Helmholtz free energy given by

$$A = U - TS = -k_B T \ln Z \quad (74)$$

and the Gibbs free energy, given by

$$G = U + pV - TS \quad (75)$$

where U is the internal energy, T is the temperature, S is the entropy, p is the pressure and V is the volume of the system. Z is the partition function of the system. The Helmholtz free energy is the more useful quantity for dealing with a microscopic or quantum system, and this is the definition that should be assumed in each mention of free energy from here on. Since we cannot compute the partition function directly, there is no direct measure of entropy available to us, so we must search for an alternative method for obtaining a value for the free energy. Shirts *et al.* discuss various methods for finding free energies [20, 21].

8.2 Expanded Ensembles

The method of expanded ensembles was introduced by Lyubartsev *et al.* in 1991 [22]. It is often used to gradually insert particles into a system by way of slowly stepping up the coupling between the particle and the system, and by this means finding the free energy difference associated with the particle insertion. It allows the system to equilibrate and adjust to the changes, so that the insertion does not cause a sudden dramatic change in the system—preventing the system

from straying too far from equilibrium. Escobedo and Martínez–Veracoechea [23] provide a straightforward and comprehensive overview of the implementational detail, as described below.

In our case, the system is simulated as normal, except that the potential barrier height is multiplied by a parameter Λ_λ which ranges over M stages from 0 to 1, with λ being a label for each state. In the state $\lambda = 0$, $\Lambda_0 = 0$ and in the state $\lambda = M$, $\Lambda_M = 1.0$. Insertion or removal of the barrier by stages is analogous to the insertion or removal of a particle by stages.

The system is free to randomly walk between adjacent states (after suitable equilibration intervals), subject to an acceptance step, with the probability of acceptance of a jump from i to j given by

$$P_{i \rightarrow j}^{acc} = \min \left[1, \frac{\alpha_{ji}}{\alpha_{ij}} e^\xi \right] \quad (76)$$

where the ratio $\frac{\alpha_{ji}}{\alpha_{ij}}$ is 1 except when the system is at 0 or M , in which case it equals $\frac{1}{2}$. Likewise, the change in state, Δ , for most of the states is ± 1 , whereas for the state 0 it is 0 or $+1$, and for state M it is 0 or -1 . The argument of the exponential, ξ , is given by

$$\xi = -\beta(U_j - U_i) + \xi^* \quad (77)$$

where U_i and U_j are the total energies of the system in states i and j respectively. The total energy U_i of the system in the current state i should already be known as part of the Metropolis Monte Carlo algorithm; the energy U_j of the prospective future state j can be calculated by performing a normal chain energy calculation, but substituting the value of the potential barrier in state j for the current potential barrier value. Since the total energy of a specific chain will inevitably be increased in higher states compared to lower ones, this means that jumps where $\Delta > 0$ (*i.e.* where we are moving up towards $\lambda = M$) are less likely to be accepted, and the simulation will spend most of its time in lower-energy states. With this behaviour, we would not be able to determine a reliable value for the free energy of the system.

This situation is remedied by including the parameter ξ^* , which is a weighting parameter given by

$$\xi^* = \psi_j - \psi_i \quad (78)$$

where ψ_i and ψ_j are weights assigned to each state. The assignment of these weights is discussed shortly; with carefully chosen weights the natural and undesirable tendency of the simulation to reside in low-energy states can be ameliorated.

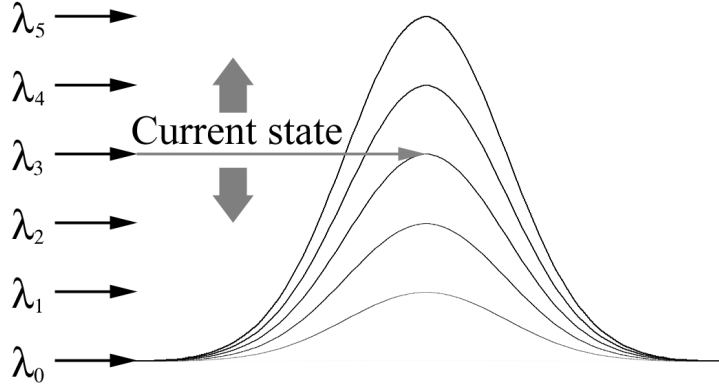


Figure 9: The method of expanded ensembles: the system jumps between energy states, with each state corresponding to a scaled external potential.

The system is allowed to equilibrate between each jump—a reasonable equilibration time can be estimated by measuring the decorrelation time, obtained from the autocorrelation of the system, as discussed earlier. Obviously, an equilibration time that is too short will not allow the system time to adapt to its new configuration, affecting the transition probability of the next jump; while one that is too long will waste time, since we need to achieve as many state-changes as possible in order to obtain a well-converged transition probability distribution.

The method of expanded ensembles as presented here has two main advantages. Firstly, the energy difference between two states can be found, and hence the energy difference between a system with a large potential barrier and one with no barrier can be found. Secondly, problems with ergodicity are alleviated somewhat, as the chain of beads can relax and spread out when the barrier is at low energies. In fact there will always be a certain portion of time during which the chain of beads feels no external force at all, and we are simulating a free particle with no potential field. There is likely to be a much higher range of possible chain configurations at low or zero energy.

8.3 Free Energy Difference

The free energy difference between two states can be calculated from the transition probabilities of each state. As the code runs, the calculated transition probability to and from each state is stored as a running average. The free energy difference between two adjacent states is given by

$$\beta [F(\lambda_j) - F(\lambda_i)] = -\ln \left[\frac{\Pi(\lambda_j)}{\Pi(\lambda_i)} \right] = -\ln \left[\frac{\langle P_{i,j}^{acc} \rangle_i}{\langle P_{j,i}^{acc} \rangle_j} \right] \quad (79)$$

where $\Pi(\lambda)$ is the transition probability distribution over all states, and $\langle \rangle$ denote the average over all attempted transitions of a given type. Due to this requirement, the code needs to switch between states as often as possible, hence the need for a random walk between states. Note that the transition probability that is stored in order to calculate the free energy difference is the unweighted transition probability, whereas the weighted transition probability is used to actually regulate the movement between states. This means that the implementation needs to be carefully checked in order not to confuse the two distributions. In fact, the transitions to lower energies will always be accepted, therefore transitions in a downwards direction will always have an acceptance probability of one. This means that we can choose the direction in which we calculate the free energy difference in order to eliminate the division in Equation (79).

The free energy difference between the extreme states is a simple summation of differences over all states. Since we are allowing Λ to range from 0 to 1, the free energy difference between $\lambda = 0$ and $\lambda = M$ is the total free energy due to the electron sitting in the potential field. The free energy of the electron without the potential (calculated by Equation (89)) must be added to this to give the total free energy.

8.4 Optimisation of Weighting Parameter

The weighting for each state can be adjusted to improve the rate of transitions between states, and, moreover, to prevent the system spending all its time at the lower-energy states. The approach chosen (for simplicity's sake) is the flat-histogram approach, from Trebst *et al.*, in which the weighting difference between states is the same as the free energy difference between states, aside from a factor

of β :

$$\psi_j - \psi_i = \beta [F(\lambda_j) - F(\lambda_i)] = -\ln \left[\frac{\Pi(\lambda_j)}{\Pi(\lambda_i)} \right] \quad (80)$$

This should produce a flat histogram of states visited—in other words, $\Pi(\lambda_i)$ is a uniform distribution. The implementation starts with the weight for each state set to zero, and no knowledge of the free energy difference. As the simulation progresses the average transition probability to each state in each direction is calculated, and a running estimate of the free energy for each pair of states is produced, which is then used as a weighting factor for each state. As the simulation runs these quantities are updated, so the free energies and weights converge towards an accurate value.

The weights only affect the efficiency of the simulation, so even if the weights are not used a value for the free energy can be calculated, but with a much greater convergence time. With larger barrier heights the flat histogram approach appears to work less well. It is suggested that the reason that large barrier energies cause the flat-histogram approach to break down is that the estimation of the weights requires the simulation to run up and down through all of the states a number of times initially. With huge energies, the system will drop straight into its lowest energy states and rarely venture into high energy states at all, meaning that the weighting factors never get a chance to converge. However, the energies that are sufficient to impact the efficiency significantly are somewhat unphysical ($\sim 1\text{keV}$), and for realistic test applications this should not be a cause for concern.

8.5 Compute Time

One drawback to the method of expanded ensembles is that the system needs to be allowed to equilibrate in between state jumps—a time that is of the order of the decorrelation time of the system. This can be a large interval for complex systems with a high number of slices, causing the free energy to converge slowly. The main cause of the correlation is the fact that with single-slice moves the chain does not change greatly from one to the next. For this reason, the multi-level multi-slice moves discussed in Chapter 7 should provide a massive advantage. With the multi-level approach, as soon as a new chain is accepted the system has become almost completely decorrelated.

9 Tunnelling Time

9.1 Tunnelling Splitting

The tunnelling probability is closely related to the the tunnelling path taken and the tunnelling time, which is in turn directly inversely proportional to a quantity known as the *tunnelling splitting*. If we consider the wavefunction of the electron in the simplest two-well problem we may study, then there are two accessible states in the two-well problem that correspond to the even and odd parity eigenfunctions of the Hamiltonian operator in the Schrödinger equation. Due to symmetry the wavefunction may split into two states, which are the even and odd parity solutions to the Schrödinger equation, see Figure 10.

$$|\psi\rangle = c_1|1\rangle + c_2|2\rangle \quad (81)$$

Here, c_1 and c_2 are the two possible states of the electron—*i.e.* in well one or well two. There are two possibilities:

$$|+\rangle = |1\rangle + |2\rangle \rightarrow \lambda_+ \quad (82)$$

$$|-\rangle = |1\rangle - |2\rangle \rightarrow \lambda_- \quad (83)$$

where λ is the energy of each solution.

The first, positive or “even”, wavefunction $|+\rangle$ never crosses zero, while the second, odd, wavefunction $|-\rangle$ does cross the zero value. This causes an energy difference between the even and odd wavefunctions—although the probability of the electron being in that well is unaffected because the square of the wavefunction $|\psi|^2$ is unchanged. The energy difference arises due to the difference in the kinetic part of the contribution to the energy, and will generally be small due to the relatively large distance between wells. The coupling of the energy levels is provided by the tunnelling splitting term

$$\Delta\lambda = \lambda_+ - \lambda_- \quad (84)$$

The tunnelling splitting can be found from the free energy difference between a path that begins and ends in one well, and a path that is stretched from one well to another—known as a *kink*. Hence, in order to calculate the tunnelling

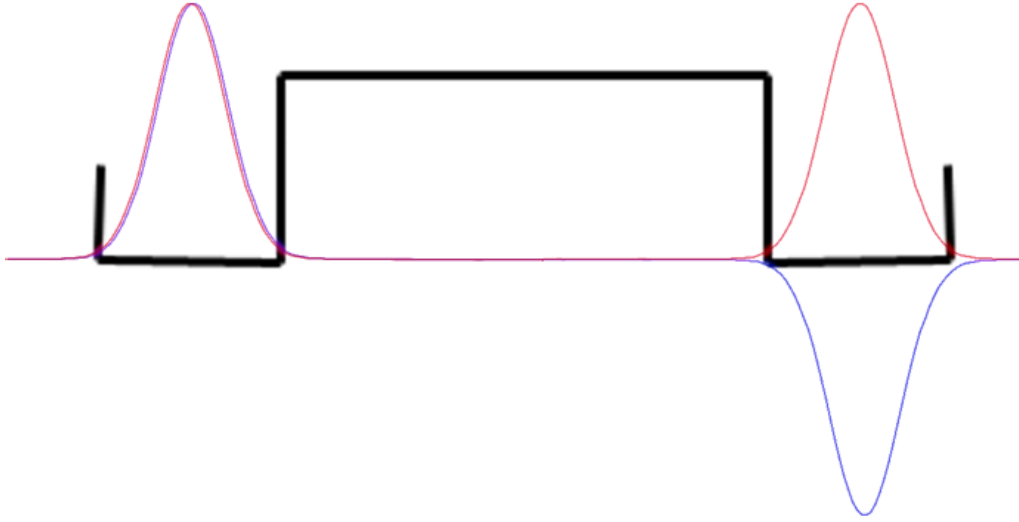


Figure 10: The two possible wavefunctions of an electron in a two-well system. The two states of the electron are $|c_1\rangle$ and $|c_2\rangle$. There is an energy difference between the two wavefunctions, while the square of the wavefunction remains the same.

splitting, an efficient method for calculating the free energy must be known (see Chapter 8).

The tunnelling splitting is given by Benjamin and Nitzan [24] (note that due to a mistake in their derivation their equation is incorrect by a factor of two, corrected here):

$$\Delta\lambda = 2 \frac{e^{-\beta F_K}}{\beta} \quad (85)$$

where F_K is the free energy of forming a kink. The value of β must be set so that

$$\Delta\lambda \ll \beta^{-1} \ll \Delta E \quad (86)$$

In words, the inverse β must be much greater than the tunnelling splitting and much less than the distance to the next energy level. This means that the simulation can only be carried out within a fairly narrow range of temperatures, and the user must have a reasonable idea of where this energy lies to start the calculation.

The tunnelling time is given by

$$t_t = \frac{1}{\Delta\lambda} \quad (87)$$

9.2 Kink Free Energy

The kink free energy, F_K , can be easily computed providing a method exists for computing the free energy of a particular chain configuration. Two computations are required: the free energy of a system in which a chain begins and ends in the same well; and the free energy of a system in which the chain begins and ends in different wells—*i.e.* the chain is stretched through a potential barrier, as in Figure 11. Calculating the inverse of the tunnelling splitting to give the tunnelling time is then trivial¹.

F_K can be decomposed (see Figure 12):

$$F_K = F_K^{(1)} + F_K^{(2)} - F_K^{(3)} \quad (88)$$

where $F_K^{(1)}$ is the free energy of stretching the chain from a position where both ends are fixed at the same point to a position where one end is fixed at the start point and one end is fixed at the end point without an external potential; $F_K^{(2)}$ is the free energy associated with introducing a potential to the fixed stretched chain; $F_K^{(3)}$ is the free energy associated with introducing a potential to the fixed unstretched chain. The free energy of a chain without the external potential $F_K^{(1)}$ can be calculated as

$$F_K^{(1)} = (k_B T)^2 \frac{m d^2}{2 \hbar^2} \quad (89)$$

where $k_B T = \beta^{-1}$, $m/2\hbar^2 = 1/2$ for an electron in atomic units, and d is the separation between the start and end of a chain, therefore

$$F_K^{(1)} = \frac{1}{2} \left(\frac{d}{\beta} \right)^2 \quad (90)$$

This energy is independent of the number of slices. Since we will be calculating by means of the path integral approach only $F_K^{(2)}$ and $F_K^{(3)}$ —the respective free

¹In general multiplication by the reduced Planck constant \hbar is required in order to attain the correct units, but this is of course equal to one in atomic units

energy difference between each configuration of the chain in zero potential and in full potential—we will need to add this $F_K^{(1)}$ value to the $F_K^{(2)}$ result to obtain the full free energy of the stretched chain. The PIMC code does this automatically and returns both values.

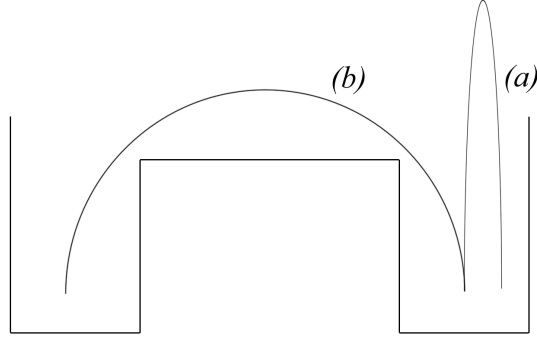


Figure 11: The two systems that are needed to compute the tunnelling splitting, and hence the tunnelling probability. *(a)* is a system in which the chain starts and ends in the same well; *(b)* is a system in which the chain is stretched across the potential barrier. The free energy difference between the two leads to the tunnelling splitting.

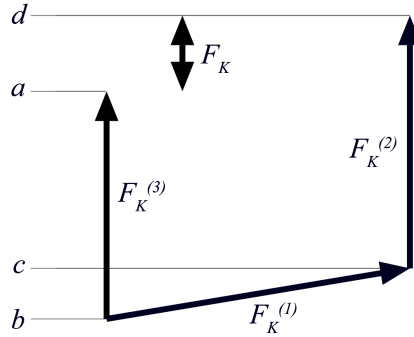


Figure 12: *(a)* is the energy of a system in which the chain starts and ends in the same well; *(b)* is the energy of a system in which the chain starts and ends at the same point with no external potential; *(c)* is the energy of a system in which the chain is stretched but no barrier exists; *(d)* is the energy of a system in which the chain is stretched across the potential barrier. The free energy difference F_K leads to the tunnelling splitting.

10 Other Considerations

10.1 Electron Correlation

In many quantum chemical calculations, particularly with regards to multi-atomic interactions, the effects of electron correlation are important to consider. In quantum calculations performed with Hartree–Fock, the electron–electron repulsion of energy r_{ij}^{-1} is replaced by the interaction of each electron with an average electron cloud, which introduces an error in the energy—called the correlation energy. This error is essentially caused by the assumption that the electrons’ movements are uncorrelated, when in fact there will always be some degree of correlation between same–spin electrons.

In path integral calculations of the nature of those performed here we tend to treat the external potential energy surface $V(x)$ as an unchanging background pseudopotential over which the particle exists independently—by which action we are reducing the many–electron problem of tunnelling to a one–electron problem. However, we must engage in some discussion as to whether this is indeed a valid assumption to make. Certainly we could reasonably expect the tunnelling electron to have some effect upon the valence electrons of the atoms within the barrier as it passes through the material of which the barrier is made, and vice–versa.

However, there is some significant difference in the speed of the tunnelling electrons compared to that of the electrons in atomic orbitals and therefore in the response times of the atomic electrons compared to the free electron, and because of this the non–local electron exchange pseudopotential V_{ex}^{NL} can be transformed into a local potential V_{ex}^L (outlined in Kuki and Wolynes [25]):

$$V_{ex}^L = \frac{V_{ex}^{NL}\psi_L}{\psi_L} = \frac{1}{2} \frac{\psi_c(r)}{\psi_L(r)} \int \frac{\psi_c(r')\psi_L(r')}{|r - r'|} dr' \quad (91)$$

where

$$\psi_L = \frac{\sinh kr}{kr} \quad (92)$$

is the spherically symmetric wave function describing the penetration of the tunnelling electron into the closed shell of an atom. When carrying out calculations on atom–based pseudopotentials, this additional pseudopotential term will correct for the electron exchange energy.

10.2 Temperature

With PIMC the temperature of the system is directly under our control as a parameter of the system, via the parameter β . This is an important factor, as control of the temperature allows us to make sure we are calculating the energy of the state we are interested in (generally the ground state). As mentioned earlier, the value of β must be set so that

$$\Delta\lambda \ll \beta^{-1} \ll \Delta E \quad (93)$$

—the inverse β must be much greater than the tunnelling splitting and much less than the distance to the next energy level.

Although this appears to limit the temperature range of the simulation, in fact the selection of quantum state is more important than choosing the temperature precisely. The external temperature of the surroundings acts only to promote the electron to excited states or reduce it to lower energy states, since we are assuming the Born–Oppenheimer approximation and thus nuclear motion is frozen. The thermal contribution of the temperature of the simulation cannot be separated from the quantum temperature, and thus the range of temperatures to which we are restricted by the requirement put forth in Equation (86) is not a limitation of the method. If we know the excitation state of the electron we are interested in we must choose a simulation temperature to achieve this state. If the temperature does not sufficiently satisfy the condition above, we may see a mixing of states rather than seeing the system in one state only.

10.3 Parallelism

Parallelisation of the code for use on independent computing cores is discussed fully in Appendix C.

Part IV

Testing and Results

11 Testing the Basic Methodology

11.1 Histograms

By dividing the space around the electron into a series of histogram bins, and collecting the positions of each of the beads as they move around, we can build a history of visited sites. Fully converged and normalised histograms are equivalent to the probabilistic amplitude of the particle's path. By examining a plot of the histogram we can usually see the approximate location of the least-action path or paths, by the areas of maximum probability amplitude. The locations of the areas of high external potential show up as depressions in the surface of the plot since the particle avoids these areas (Figures 13, 14 & 15). The histogram serves to convince us (in a fairly non-quantitative manner) that the methodologies used are working correctly, and allows us to check and compare two algorithms against each other.

Since visually plotting a three-dimensional field is difficult (generally one spatial dimension needs to be mapped to the data value), three-dimensional systems can be visualised by taking a two-dimensional histogram slice in the location and direction required.

11.2 Simple Tests

The simplest test that can be performed is to reduce the number of slices so that we have just one free bead. With one free bead and no external potential the shape of the histogram should be a Gaussian curve—a single free bead is the equivalent to a basic random walker. The width of the Gaussian for a given separation will depend on both the spring strength (controlled by the parameter λ) and the temperature of the simulation. Comparing single bead histograms proved invaluable during the implementation phase of the various methodologies, since any errors in the coding produced differing histograms from what would be

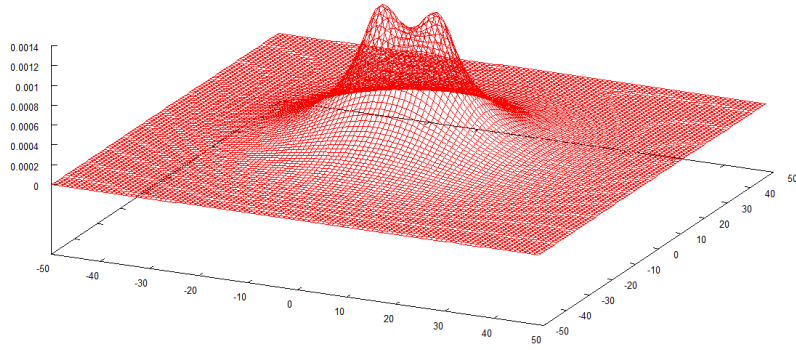


Figure 13: Histogram of free electron with no potential barrier. 100 beads at 300K, beads fixed at $-5.25x$ and $5.25x$.

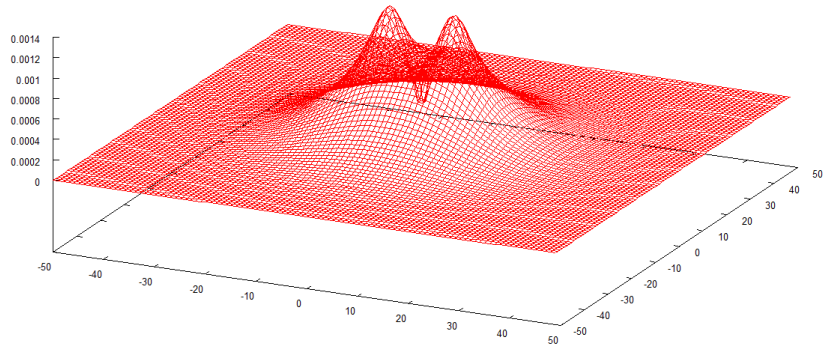


Figure 14: Histogram of free electron with Gaussian-shaped barrier, FWHM 0.415, height 0.1 Hartrees. 100 beads at 300K, beads fixed at $-5.25x$ and $5.25x$.

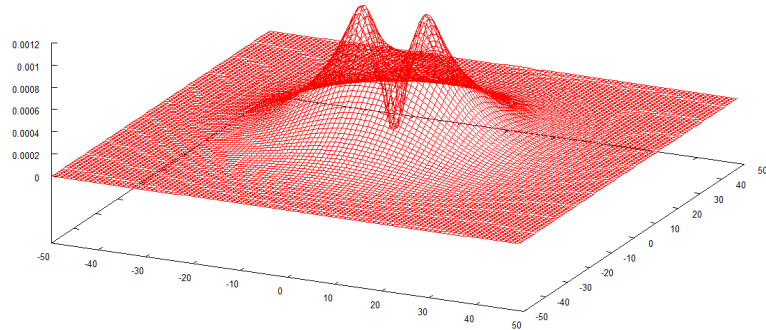


Figure 15: Histogram of free electron with Gaussian-shaped barrier, FWHM 0.415, height 0.25 Hartrees. 100 beads at 300K, beads fixed at $-5.25x$ and $5.25x$.

expected.

Figure 16 shows the plot of bead position for a single free bead in zero potential using three methods—the basic method, the force-bias method and the multi-slice bisection method. Figure 17 shows the histograms for three methods: the basic method, the multi-slice bisection method and the section regrowth method. The single-slice histogram is suffering from ergodicity problems (see sub-Section 11.4.3), but otherwise the match is nearly perfect—see the right-hand peak for evidence of this.

11.3 Temperatures

The simulation allows the inverse temperature β to be altered. Usually for electron transfer problems we will be running at $\beta = 39.47 \approx 8000K$ but by varying this number it is possible to choose the energy of the electron we are simulating. 8000K allows us to simulate an electron in the ground state in most cases. Higher temperatures will cause mixing with the excited states of the electron. Note that the “temperature” does not correspond to the physical temperature of the system’s surroundings (*i.e.* atomic vibrational energy), but rather to the energy of the electron itself. Figures 18 and 19 shows the histogram plots for a range of temperatures. It can be seen that lower temperatures cause greater spread in the histogram. Lower temperatures cause a stronger interaction of the electron with the potential energy surface, and reduce its ability to tunnel through barriers. Recall that the chain action is calculated using Equation (45) in Section 5.1:

$$E = \exp \left[- \sum_{i=0}^M \left(\frac{(r_{i+1} - r_i)^2}{4\lambda\tau} + \tau V(r_i) \right) \right]$$

Thus, with a large temperature (*i.e.* small τ) the kinetic part will dominate the chain action. However, as temperature drops and τ increases, the potential part will begin to dominate. Hence, one would expect to see the strength of the interaction between slices fall as the temperature drops, causing the low-temperature spreading seen in Figure 18.

11.4 Asymmetries

Initially some of the histograms of bead position displayed marked asymmetries when external potentials were used. As the barrier being used was a Gaussian-shaped barrier centred on the middle position between the two fixed beads, the resulting histogram tended to be a double-peaked curve. It was noticed that often one of these peaks was higher than the other (with no direction consistently preferred), when in fact there should have been no reason for this to be so (Figure 20). The code was carefully checked for bugs; none were found that could possibly have produced this anomaly. A set of tests was run using the same random number seed, and therefore the same sequence of random numbers, for different starting conditions, with the hope of finding out the mechanism which led to the bias towards one side or the other.

11.4.1 Convergent Trajectories

During the course of the tests, it was observed that entirely different starting positions—though with the same external conditions (*i.e.* potential, temperature, number of beads)—run with the same sequence of random numbers eventually led to identical outcomes. Further investigation revealed that all trajectories, given the same parameters and run with the same sequence of random numbers, converge to one trajectory. The phenomenon is discussed in a paper by B. Uberuaga *et al.*[26]. This is an unfortunate finding, as it suggests that the ultimate outcome of a simulation in a particular set of conditions is determined solely by the choice of random number seed. It was initially hoped that a suitable period of equilibration would help to ease the problem or that running the simulation for longer or starting the beads in different positions would produce a better histogram with less asymmetry. However, this finding demonstrates that none of these measures will help; the final shape of the histogram is inextricably linked to the choice of random number seed (and the geometry of the system). The shape of the histogram emerges quite early on in a simulation run, and does not alter with longer runs.

11.4.2 Instability

It is suggested that one cause of the asymmetry may be that when using the single-slice methods the chain begins stretched evenly through the barrier. The bead moves occur from start to end in sequence, so the first bead to move will have an effect (through the kinetic part of the total energy) on the direction of the second bead, which will in turn affect the third bead, *etc.* Since the chain begins in an energetically unfavourable state (lying within the potential barrier), it takes little influence to cause many of the beads to slide off to one side together. The influence of the first move could be enough to pull a majority of the chain into one well over the course of a number of passes. Once the chain is within one well, the barrier acts to keep it there.

11.4.3 Ergodicity

In an attempt to alleviate the asymmetrical effect, the same simulation was run repeatedly using an evenly spaced range of random number seeds and the results combined into one averaged histogram. It is clear that this attempt has been unsuccessful, as the graphs display asymmetries. The first graph contains 24 different runs combined, with a potential barrier height of 0.3 Hartrees [8.2eV]. The second contains 135 runs, with a barrier height of 30 Hartrees [816.3eV] (it should be noted that the value of 30 Hartrees is far in excess of the potential barrier heights that might be encountered in most real molecular environments).

The code starts from the zero position and moves the beads in sequence until the far end of the chain is reached. Initially, it was suspected that this fact was the cause of the problem, as the initial movement at one side of the barrier produced a force towards that side that cascaded down the chain so that the total initial movement was towards the starting position; however, this was shown to be false as the asymmetry does not consistently bias the result towards one side only.

This is essentially a problem of ergodicity that only manifests in the one-dimensional case for this particular barrier shape. In higher dimensions, a circularly- or spherically-shaped potential would not provide a definite barrier to the electron—in other words, the path would not be forced to tunnel through the barrier, but could instead go around it. For a two-dimensional Gaussian-shaped barrier the

path can go either side of the potential peak and might still show become trapped for one side or the other, but for a three-dimensional (spherically-shaped) barrier the chain is free to wander in space and should show an even distribution in all directions around the barrier.

Since all physical calculations are to be carried out in three dimensions, the chance for ergodicity-related problems is reduced, but it is still possible to conceive of three-dimensional situations in which the problem manifests itself, especially with single-slice methods. For example, when using enclosed wells, it might be hard for the beads to switch from one well to another. However, by using the method of expanded ensembles (Chapter 8) and/or a multi-slice method (Chapter 7) we can reduce or resolve this problem entirely. The method of expanded ensembles periodically and stochastically reduces the potential to zero, allowing the chain to explore space freely, and the multi-slice methods place new chains almost without regard to previous chain positions.

11.5 Optimum Bead Numbers

The quality of the result obtained depends on the number of time slices used. The computational effort goes up with the number of beads, so it is important to select an appropriate number of beads for each calculation to allow the simulation to converge properly within a reasonable time frame. By plotting the value of a function over beads as the number of beads is increased for a fixed number of chains, we can see the convergence of the simulation with number of beads. The optimum number of beads can be chosen by estimation from such a plot. For example, in Figure 21 the plot shows that 128 beads is too few, but there is no need to go to 512 beads since the simulation has already converged by this point. Somewhere in between these numbers is the optimum. Using 256 beads appears to be a reasonable choice, with little variation in value either side of the data point.

In the histogram plot of Figure 22 the manner in which the simulation converges as the number of beads is increased is clear. With lower bead numbers, the restrictive springs keep the beads oscillating in one location. As more beads are added, the chain moves more freely, and at 512 beads there is not much more room for improvement (other than by running for longer to converge the sim-

ulation better). However, this value will be different for different systems, and will vary especially when complex or large potentials are used. For very long distances, or very complex systems, many more beads may be required.

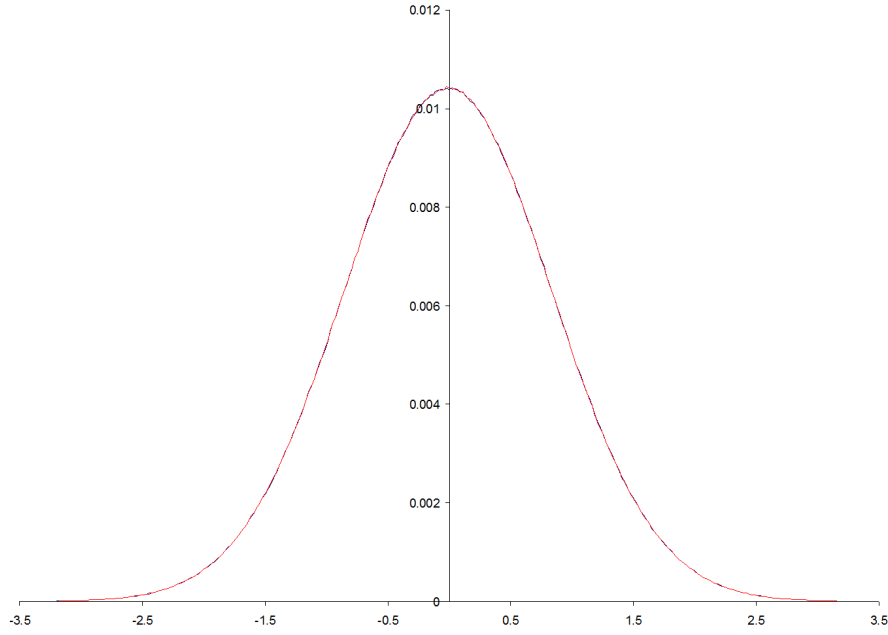
11.6 Parallel Speed Tests

The parallel speedup was tested to ensure that the gain in parallelising the algorithm is sufficient at high processor numbers. With any parallelised system there is always a chance that the overhead of parallelisation will counter the gain achieved by adding processors—in some cases this may not happen at low processor counts but might become apparent as the number of cores is increased. The passing of data between cores carries an overhead that naturally increases with the number of cores. Since the parallelisation method employed here does not pass any data during the simulation run—the data is passed between cores only at the beginning and end of the program—and the amount of data passed increases linearly with the number of cores, we would expect the time taken to generate N total chains to be inversely proportional to the number of processors P .

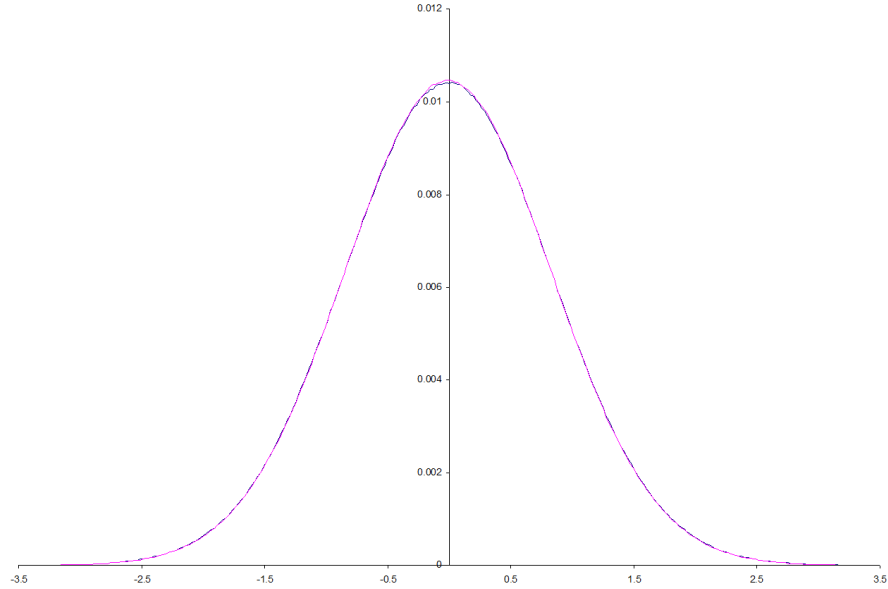
A test was run, generating 6.4×10^6 chains total, using between 1 and 128 cores, at 3000K with a chain of length 4.0 Bohr stretched between two square wells of size 1.5 Bohr with a barrier height of 0.1 Hartree, using the method of expanded ensembles to calculate a free energy. The results in seconds for the entire process from start to finish, including MPI message passing and finalising of data, are given in Table 1. The data when plotted (Figure 23) show an almost perfect inverse relationship between the time taken t and the number of cores P , up to 128 cores. It was impractical to test with a higher number of cores due to heavy usage of the Merlin cluster, but since the number of chains generated per core at 128 total cores is only 50,000, it is fair to conclude that even for small numbers of chains at high core counts the code performs with high parallel efficiency. While there will almost certainly be a core count that breaks from the linear scaling observed here, it is expected that this will only happen at a number of cores beyond the typical number used with this code, and beyond the typical number of cores available in most situations at present.

Table 1: Generation time for 6.4×10^6 chains using between 4 and 128 cores.

Cores	Time (s)
4	6886.11
8	3391.09
16	1656.26
32	904.41
64	447.11
128	228.15



(a) Basic single-slice compared to the force-bias method, single free bead attached at -1.0 and 1.0, no potential. The curves cannot be distinguished at this scale.



(b) Basic single-slice compared to the multi-slice Lévy bisection method, single free bead attached at -1.0 and 1.0, no potential. The curves cannot be distinguished at this scale.

Figure 16: The histogram of bead positions for a single free bead is a Gaussian. This plot shows the one-dimensional histograms for a single free bead in one dimension. The curves are so close they are virtually indistinguishable. The fixed start and end beads are located at -0.5 and $+0.5$. Distances in Bohr.

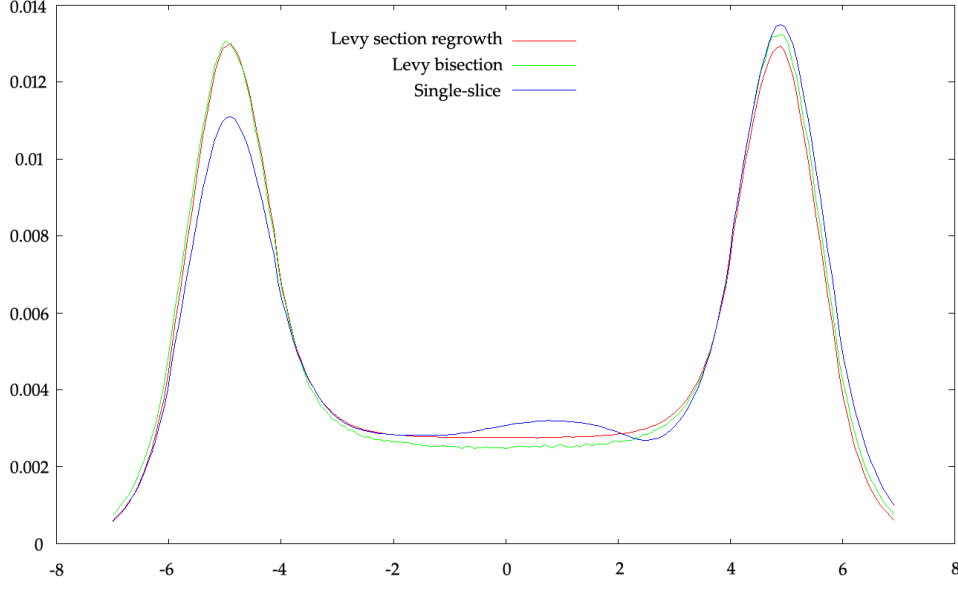


Figure 17: The histograms of three different methods: single-slice, Lévy bisection and section regrowth (129-bead sections) using Lévy bisection. Generated with 513 beads at 8000K ($\beta = 39.5 \text{ H}^{-1}$) in a double-well potential of the form $f(x) = (x + a)^2(x - a)^2/a^4$ for $a = 5$. End beads fixed at -5 and 5. Distances (x -axis) in Bohr.

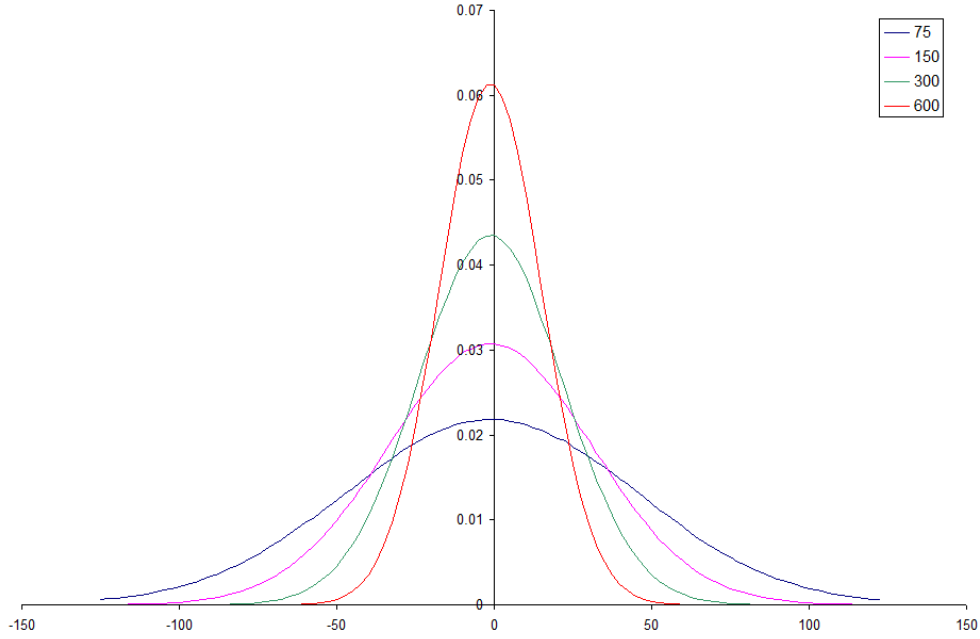


Figure 18: The histogram of a single free bead (chain fixed at -1.0 and 1.0) in one dimension with no external potential at different temperatures. Distances (x -axis) in Bohr. Single-slice basic method used.

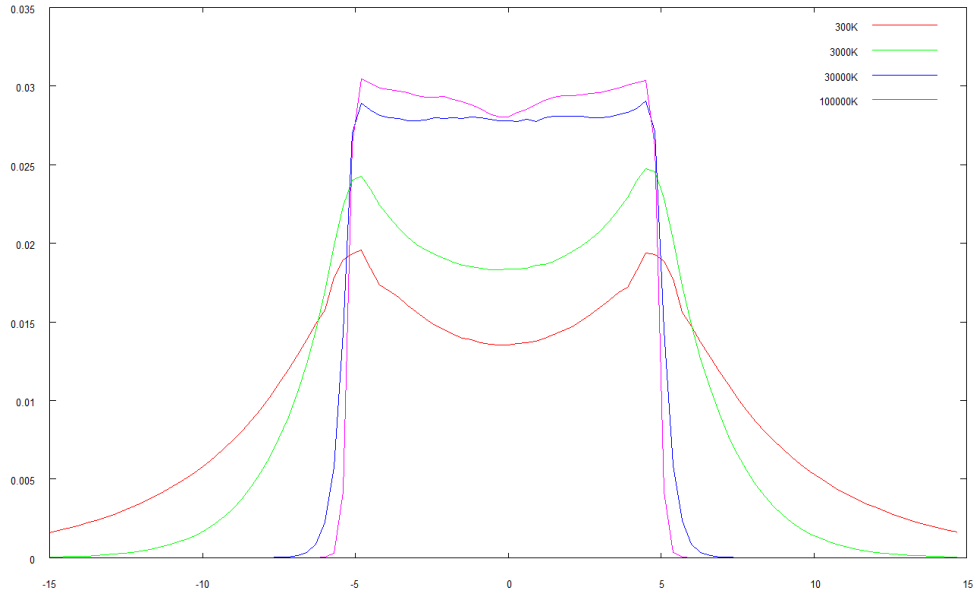


Figure 19: The histogram of an electron (chain fixed at -5.0 and 5.0) in one dimension in a weak (0.05 Hartree) double square well potential at different temperatures. Distances (x -axis) in Bohr. The reduced effect of the potential barrier on the electron as the temperature is increased is apparent.

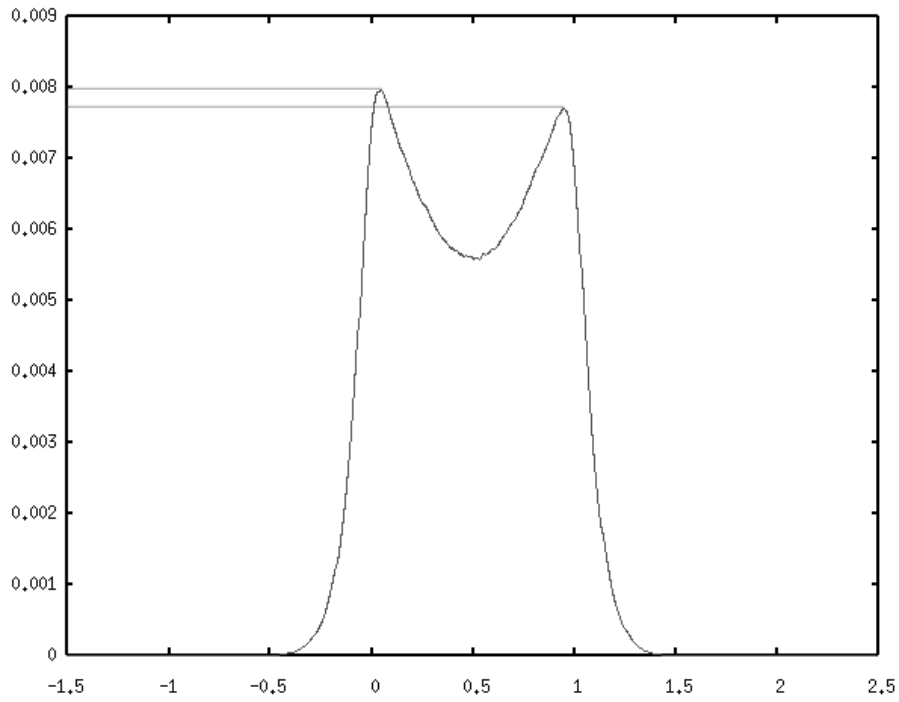
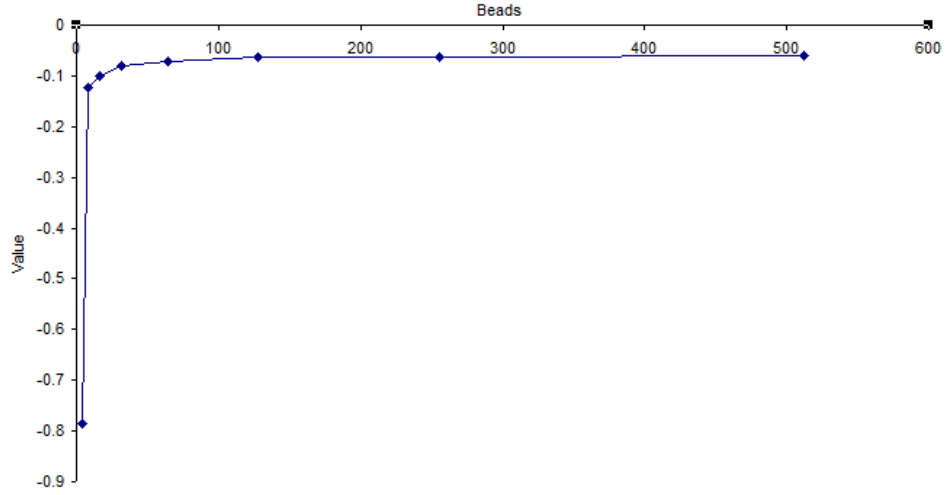
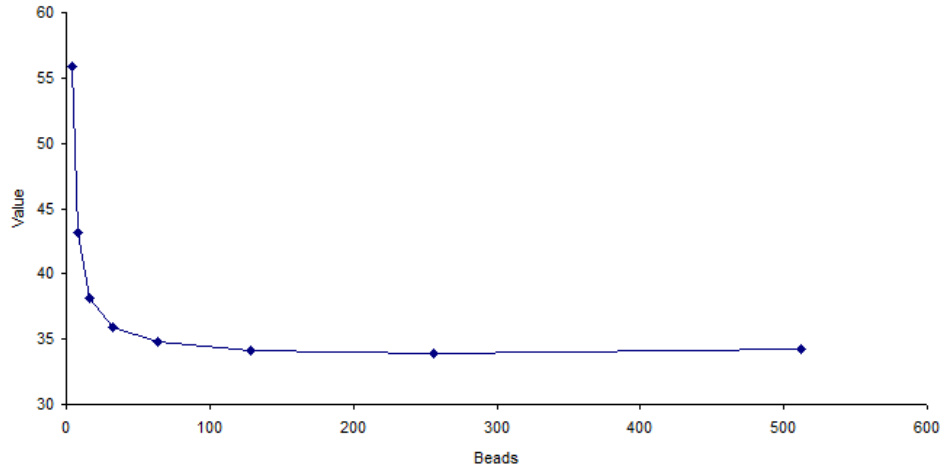


Figure 20: In this histogram the asymmetry of the plot is clearly visible (marked by light grey lines). Since this is a one-dimensional system using the single-slice method, the beads can become trapped to one side or the other of the potential peak in the middle. Used basic single-slice method, Gaussian-shaped barrier of height 5.0 Hartrees, 33 beads. Distances (x -axis) in Bohr.



(a) Convergence of e^{-x} with number of beads (chain at -0.5 to 0.5, in zero external potential).



(b) Convergence of \sqrt{x} with number of beads (chain at -0.5 to 0.5, in zero external potential).

Figure 21: Plotting the final average value of a quantity over all beads against the number of beads used gives us an idea of how many beads is necessary for a particular simulation. The simulation has converged as the bead number passes around 200.

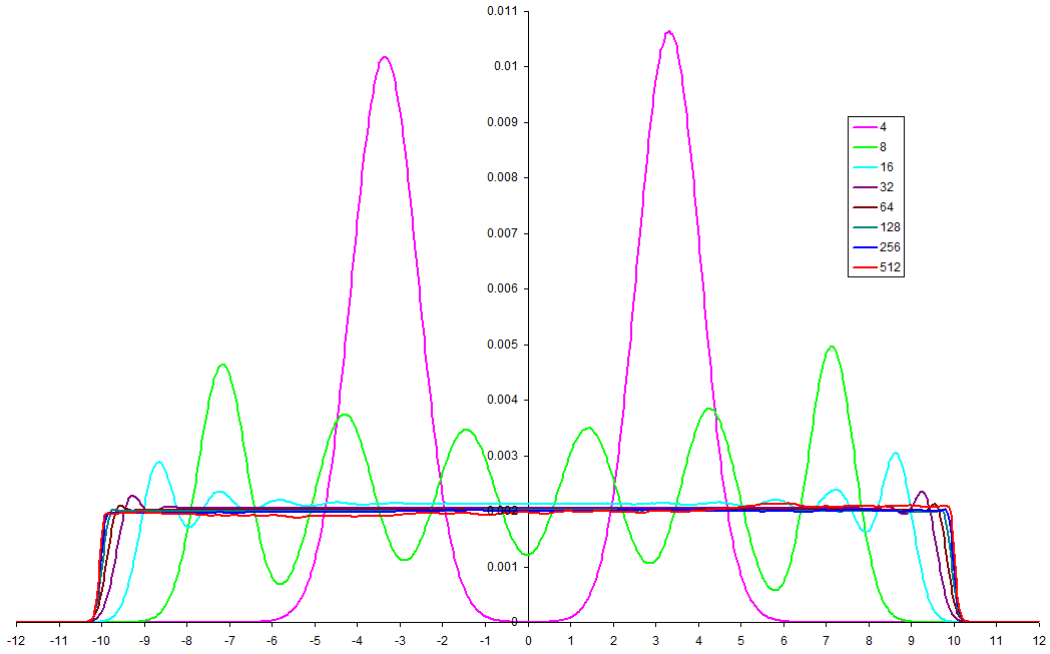


Figure 22: A plot of normalised histograms with number of beads, single-slice basic method—no external potential has been applied. The start and end beads are located at -10.0 and 10.0 . This plot shows the convergence towards the exact probability density function as the number of beads is increased, and the chain is allowed to relax into its surroundings.

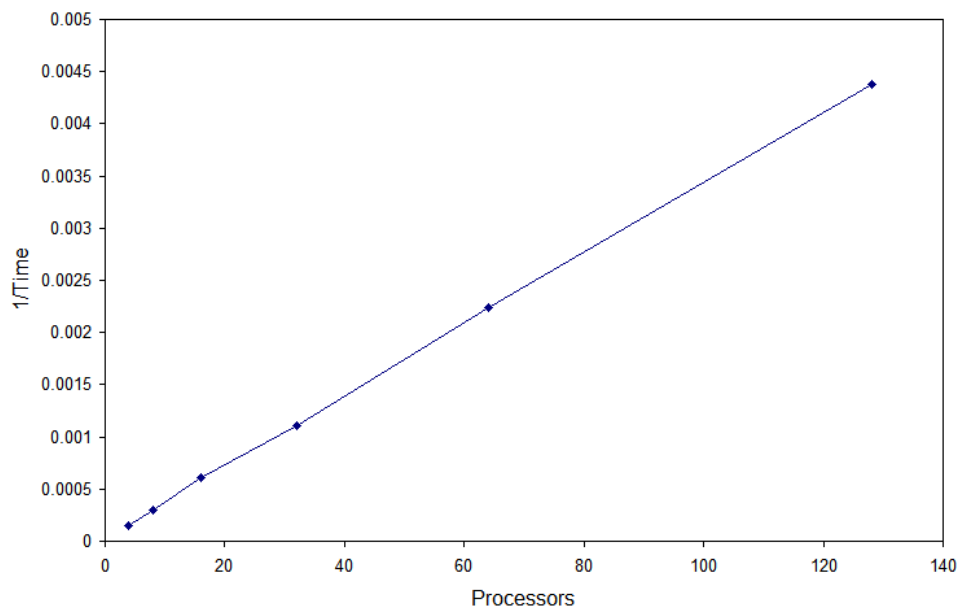


Figure 23: The plot of $1/\text{Time}$ for the same amount of work distributed over N processors.

12 Autocorrelation

12.1 Autocorrelation

It is critical to have a simple (preferably scalar) measure of how well one algorithm performs against another. For example, we may wish to compare the force-bias algorithm against the multi-level bisection algorithm. The idea of “convergence” has been discussed in the sections covering the various algorithms (sections 5, 6 and 7), but has not yet been defined.

To measure the speed of convergence of an algorithm, we introduce the autocorrelation. The autocorrelation function of a quantity is a measure of self-correlation—that is, the correlation of a quantity with itself over time. The autocorrelation χ of a measurable m from time t to dt is given by

$$\chi(dt) = [m(t)m(t + dt)] \quad (94)$$

That is, the value of the quantity is stored at an initial time t , and at each further step dt the autocorrelation function is given by multiplying this stored value by the current value of m . The function can be fully normalised, if the time average and the time average of the square of m is available:

$$\chi(t) = \int dt \frac{m(t)m(t + dt) - \langle m \rangle^2}{\langle m^2 \rangle - \langle m \rangle^2} \quad (95)$$

For this particular system (of beads connected by harmonic springs), obvious choices for the measurable quantity include the position of a single selected bead, the average position over all beads, and the average potential, all of which are calculated as a matter of course during the execution of the program. As the beads move around considerably it might be expected that the average potential remains more constant than the bead positions, and thus is a more stable quantity to use. The code allows the user to choose which of these to use.

Building the autocorrelation function could be a slow process; if we store a value m at time t and then calculate the autocorrelation at time $t + dt$ for one run of the simulation, the resultant autocorrelation function will consist largely of noise. In order to accelerate the definition of this function, a system was implemented whereby several separate “windows” are used to give numerous sets of results simultaneously, which are then averaged to a single correlation decay

curve. Initial positions are stored as the current time reaches the beginning of each window, and as the simulation proceeds the correlation of the current data with each stored position is calculated. This data is added to the overall autocorrelation function, with the correct offset, as illustrated in the graphic of Figure 24.

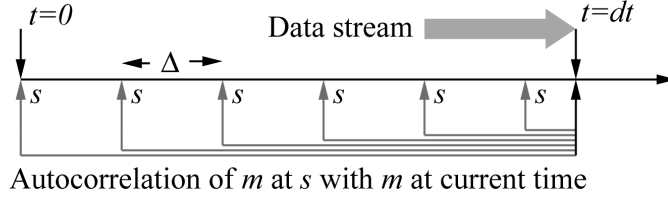


Figure 24: The measurable m is stored at intervals of Δ . The autocorrelation of m at $t = dt$ with m at $t = s$ is calculated, and averaged into one function.

In order to obtain a complete function of length δ for each window, the last store point s must occur at least δ chains before the end of the simulation. In the code, the user can choose how many windows to use and the length of one window. The length of a window should be larger than the decorrelation time—some trial and error may be required in order to choose a suitable window length δ . The windowing method produces some regular high-frequency oscillations, but the overall shape of the curve is unaffected. Running the simulation for as long as possible minimises noise at all frequencies and allows the shape of the data to be resolved. Figure 25 shows a decorrelation comparison between two methods.

We can obtain an estimate of the decorrelation time τ_d by fitting a curve of the form

$$y = Ae^{-\frac{x}{k_1}} + (1 - A)e^{-\frac{x}{k_2}} \quad (96)$$

as in Figure 26, and then the decorrelation time τ_d is given by

$$\tau_d = Ak_1 + (1 - A)k_2 \quad (97)$$

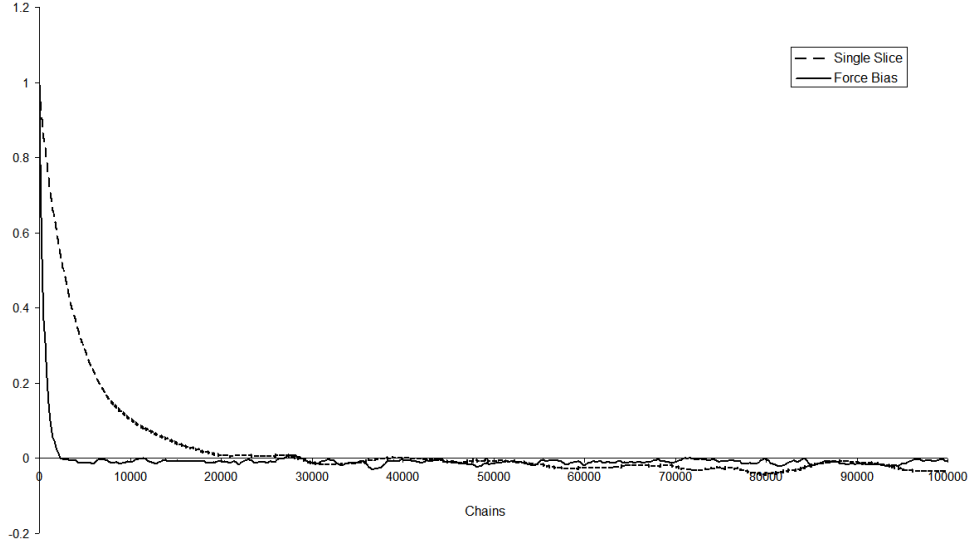


Figure 25: The plot shows the autocorrelation curves for the basic and force-bias methods. The force-bias method clearly out-performs the basic method (5.0×10^7 chains at 1000K, 33 beads over a gaussian-shaped 1-D well height 10.0 Hartrees).

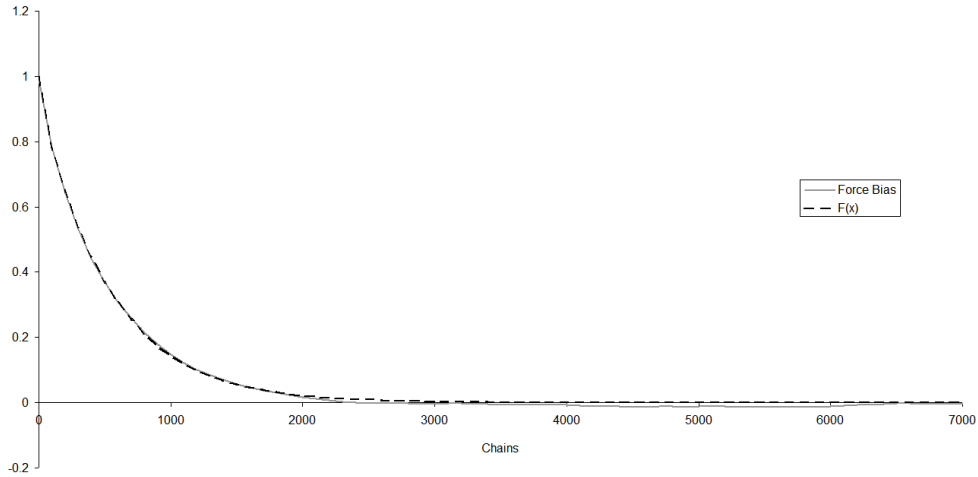


Figure 26: By fitting a curve defined in Equation (96) to the autocorrelation function we can determine the decorrelation time. In this case $A = 0.059570934$, $T1 = 2.0$, $T2 = 533.7208086$ and the decorrelation time from Equation (97) is 502.1 chains.

12.2 Chain Correlation

The main outcome of this work is to determine efficiently the tunnelling time for an electron, by calculating the free energy using the method of expanded

ensembles. The method of expanded ensembles requires the system to undergo as many state-changes as possible, in order for the value of the free energy to converge sufficiently. In order for this to happen, the system must equilibrate at each state, and the equilibration time is of the order of the time it takes the system to decorrelate—that is, the time taken for the system to have no ‘memory’ of its previous state (Section 12.1). For this reason, a methodology that decorrelates quickly is highly desirable. The plot in Figure 27 shows the autocorrelation curve of the potential averaged over all beads for three methods: the basic single-slice method, the force-bias algorithm, and the Lévy bisection method. Clearly, the Lévy bisection algorithm is far superior to the other two, since it requires a separate plot in order to show how quickly it decorrelates. The Lévy bisection algorithm is the only conceivable choice for free energy calculations using the method of expanded ensembles at high temperature; the Lévy bisection section regrowth method is usable at lower temperatures also. The section regrowth method is essentially the same method as the Lévy bisection method—in fact, since several moves are made per chain pass, it is expected to decorrelate even faster than the whole chain Lévy bisection method.

Table 2: Decorrelation times for the three methods tested. Times given in numbers of chains.

Method	31 slices	127 slices
Single-slice	4069	62282
Force-bias	501	14498
Lévy bisection	1.5	1.5

The decorrelation time is given by integrating the curve of the autocorrelation plot; values are given in Table 2. Note that the decorrelation time for the Lévy bisection method is unchanged with bead number. While this method decorrelates within a few chains, this is the value for a fixed potential. The method of expanded ensembles requires the potential to be ramped up and down as the system jumps between states. In low potential states the Lévy bisection method will equilibrate to a low-energy state, which may cause difficulties when the potential jumps back up to a higher value, since the method involves placing an entire chain. For this reason we have to allow more than two chains of equilibration time, to account for rejected chains and to give the algorithm time to find

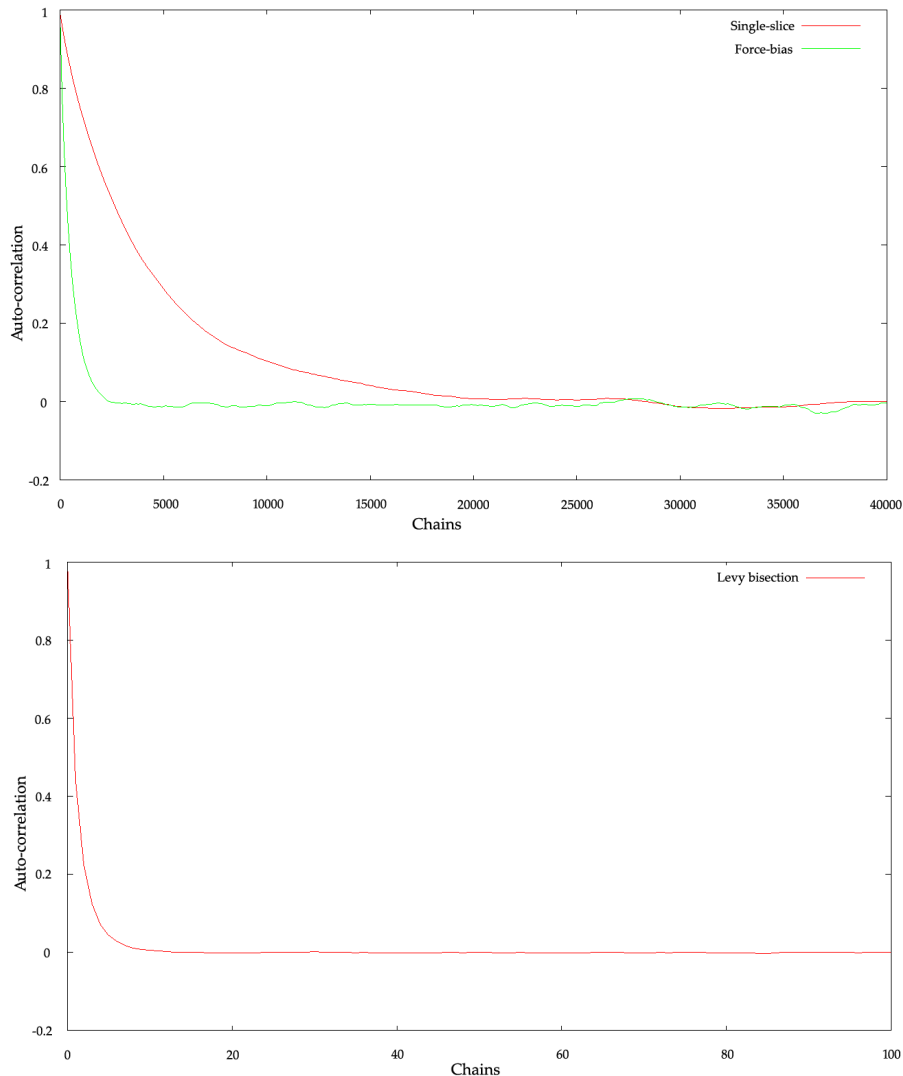


Figure 27: A plot of the normalised autocorrelation curve for three methods: single-slice, force-bias and Lévy bisection (257 beads fixed at -2.0 and 2.0, Gaussian-shaped external potential with barrier height 0.1 Hartrees). Clearly the force-bias algorithm far outperforms the basic single-slice method. However, the Lévy bisection method decorrelates within a few chains, shown in the second graph.

an acceptable low-energy chain configuration in a high-potential environment. Clearly, we want to reduce this number as much as possible, to allow for as many ensemble/state changes as possible during the course of the simulation. Furthermore, the number of states used in the expanded ensembles approach must be high enough that the jump between states is not too coarse, in order to reduce

the equilibration time as much as possible—although a higher number of states is preferable for this approach in general.

13 Testing the Method of Expanded Ensembles

13.1 Flat Histogram Approach

The flat histogram approach is a method of optimising the weights of Equation (78) that should ensure that all energy states are visited an equal number of times. A histogram of states visited should converge to being nearly flat. Although the optimisation does not affect the result of the calculation, it affects the efficiency and thus the speed at which the free energy value converges. Figure 28 shows the histogram of visited states for a particular run. The flat histogram approach shows a trend sloping downwards towards higher energies, but this is likely due to the influence of the earlier part of the simulation, before the optimising weights had time to accurately converge. The histogram is flat enough to convince us that the system has visited the high-energy states sufficiently often to ensure a fully converged answer, given an appropriate amount of time.

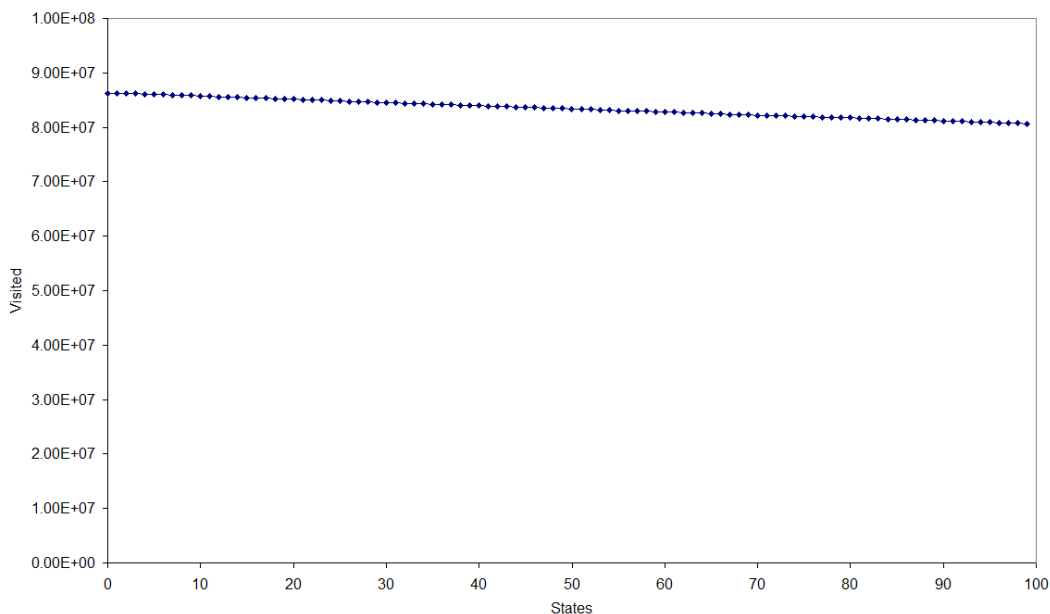


Figure 28: A plot of the histogram of visited states in the method of expanded ensembles (129 beads at -1.5 to 1.5, Gaussian barrier with a height of 1 Hartree, using the Lévy bisection method at 3000K).

13.2 Converging the Free Energy

Since the method of expanded ensembles works by stepping up and down through an ensemble of states, equilibrating at each state before moving on, the free energy can take some time to converge. When running the code with the method of expanded ensembles we need to ensure that we are using enough states, and running the system for long enough, to ensure that the calculated value of free energy has fully converged to the required precision. To this end, the value of free energy can be recorded as the system evolves, and plotted as per the convergence plots in Chapter 12. Figure 29(a) shows a plot of the convergence history of the free energy for a particular simulation run (63 free beads, Gaussian-shaped potential barrier of height 1 Hartree, 50 ensemble states). After 10^8 chains (generated at the highest state—the total number of chains generated over all states is of the order of 50×10^8), the value of free energy appears well-converged at 0.585 Hartrees—however, on plotting at a finer scale (Figure 29(b)), the simulation shows much variation. As with many statistical simulations, converging each additional significant figure can take an order of magnitude longer than the previous significant figure. The error value gives some indication of how precise the answer is.

13.3 Free Energy with Number of Slices

It is a feature of the path integral approach that any calculated property of the system converges to an exact value as $M \rightarrow \infty$, that is, as the number of time slices/beads approaches infinity. It was shown in Chapter 11 that arbitrary properties of the system converge to exact values at around a few hundred beads, and this is to be expected for the value of free energy also. Figure 30 shows a plot of the free energy with the number of beads, and with the reciprocal of the number of beads. Clearly, from the first plot, the free energy is accurate above N beads, and the second plot shows that the convergence follows a power law.

13.4 Optimising the Decorrelation Time

The Lévy bisection method allows us to use a much shorter equilibration time than any single-slice method, as one single accepted complete chain essentially

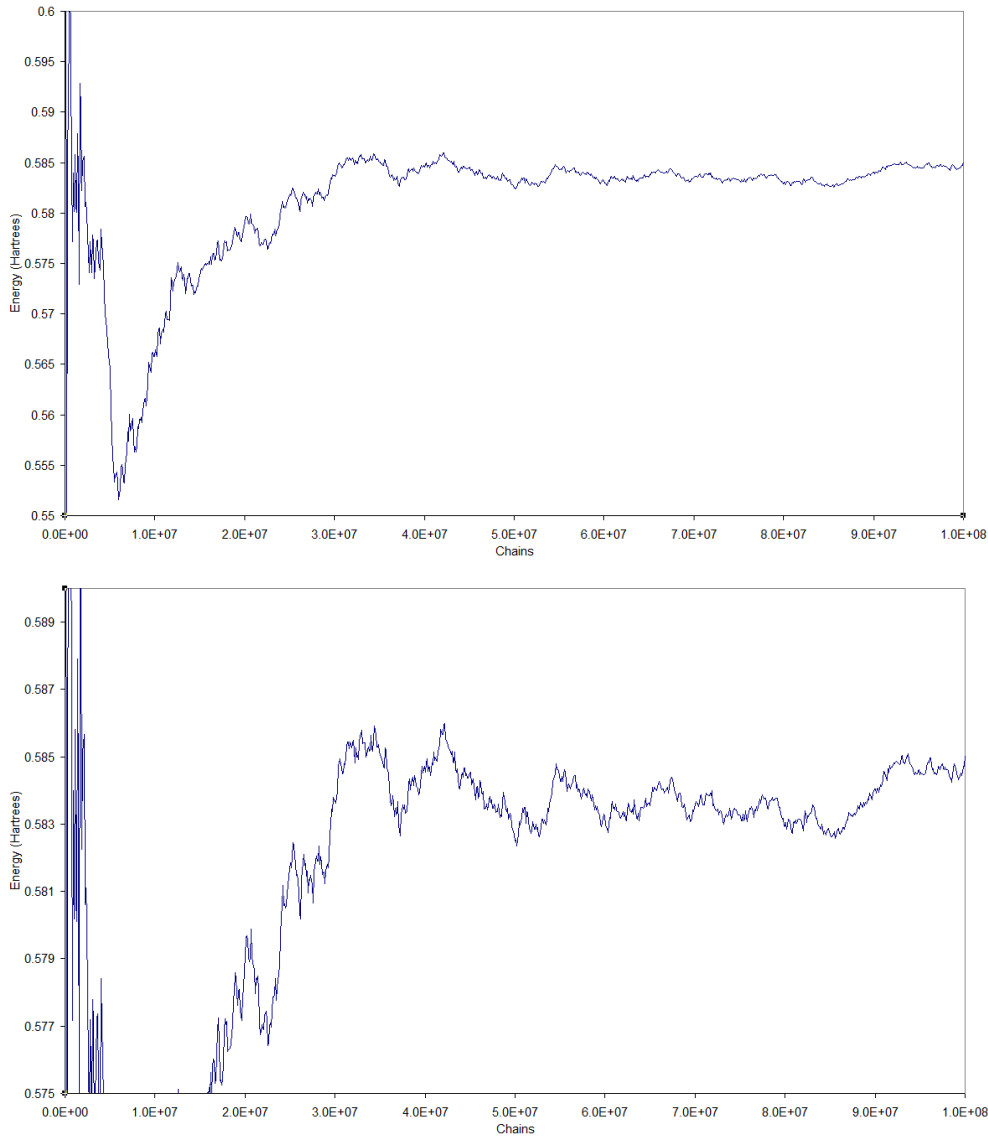


Figure 29: Plots of the convergence of the free energy at two different scales (63 beads from -2.5 to 2.5 over a 1 Hartree Gaussian-shaped barrier using 50 expanded ensemble states at 3000K).

decorrelates the system. The decorrelation time calculated in Chapter 11 of 1.5 chains should enable the equilibration time to be set to 5 chains, allowing for occasional runs of rejected chains. The section regrowth method makes more than one pass per chain, which allows even faster decorrelation, making the choice of five chains sufficient. Tests were run with the equilibration time set to 3, 5, 10, 25 and 50 chains in order to ensure that a short equilibration period still provides

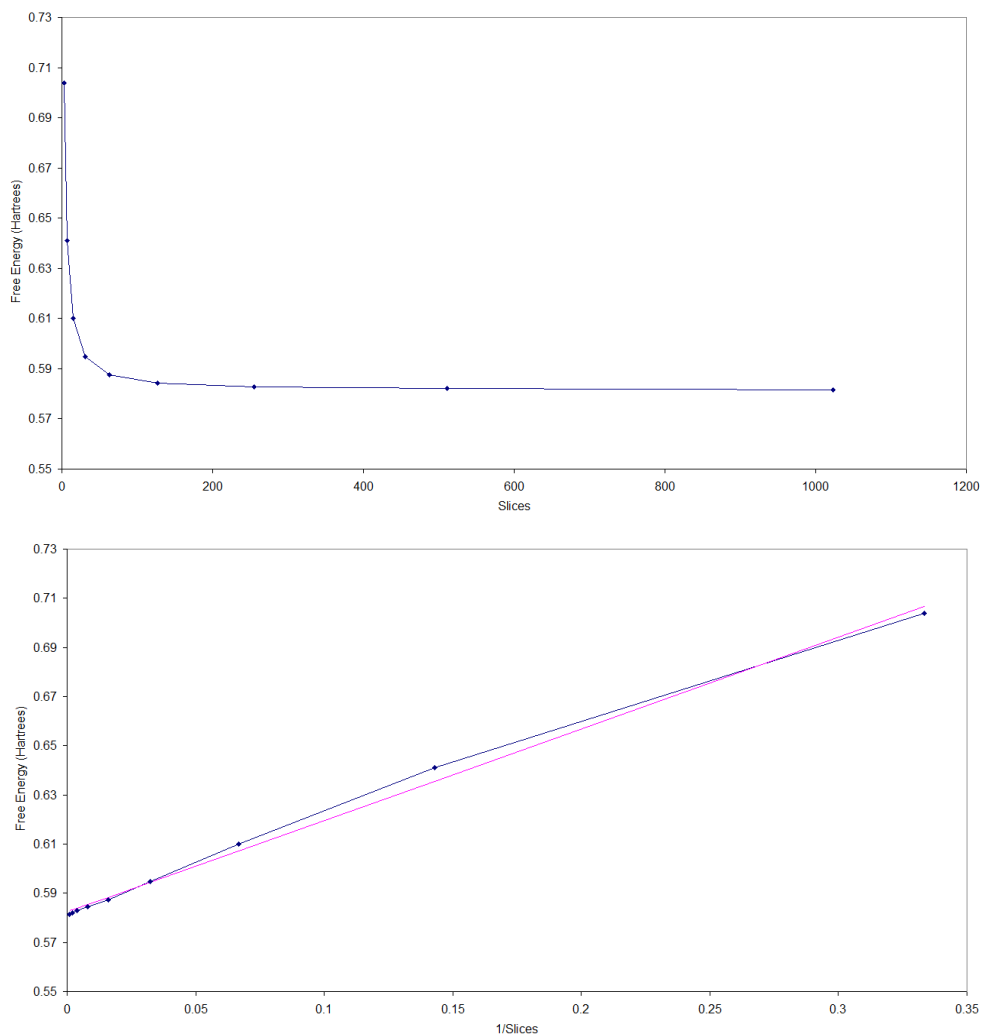


Figure 30: Plots of how the free energy changes with the number of slices (from -2.5 to 2.5 over a 1 Hartree Gaussian-shaped barrier using 50 expanded ensemble states at 3000K). The error bars are too small to show here. The first plot shows the raw curve, while the second plot is $1/E$.

an accurate result.

13.5 Free Energy with Number of States

The number of states used in the method of expanded ensembles influences the quality of the result. Too few states and the answer will not be precise; too many and the simulation will not complete in a reasonable time. Simulations were run with differing numbers of states in order to investigate the optimum quantity.

The plot below in Figure 31 shows the free energy for 10–110 states in intervals of 10 (the plots are generated using 127 free beads, with a Gaussian potential barrier of height 1 Hartree, converged to 8×10^8 chains). It is clear that around 100 states are required in order to ensure a converged answer. This value may vary with the barrier height.

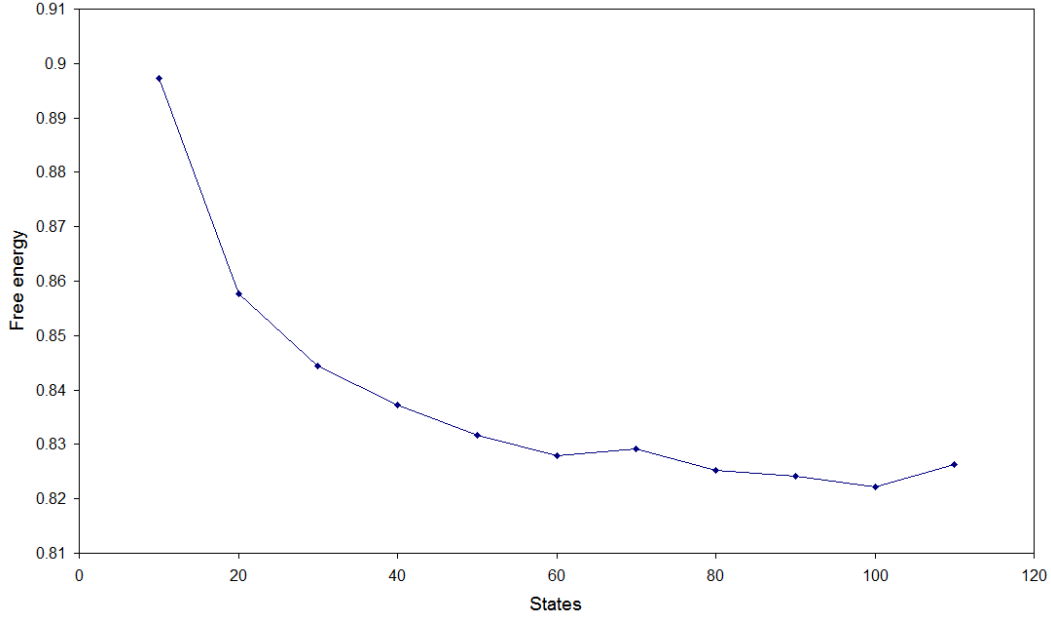


Figure 31: A plot of the free energy as the number of ensembles is increased (from -2.5 to 2.5 over a 1 Hartree Gaussian-shaped barrier using 50 expanded ensemble states at 3000K). We see that at least 100 states are required in order to fully minimise the free energy.

14 Square Well Systems

One of the simplest test-cases that can be studied is the double rectangular or square well. Square wells are simple to model both analytically and numerically, and in fact can represent some real-world situations with surprising accuracy. Tunnelling in microscopic electronic components, for example, can sometimes be modelled using a square well system, and more complex systems can be approximated by a judicious arrangement of square or rectangular wells.

The PIMC code allows for the use of a potential consisting of any number and arrangement of axis-aligned rectangular boxes in three dimensions. The barrier height (external medium) and well floor depth (well medium) can be chosen. This chapter covers various systems explored using square wells.

14.1 Smoothing the Potential

Before the square well systems themselves are discussed, it is worth quickly describing the function used to make the well edges “soft”. This is done not only to avoid discontinuities when using the force-bias or fourth-order algorithms—that both require the gradient of the potential—but also to make the system more “physical”. The same edge curve function is used for every potential model developed—*i.e.* square well, nanotube, cubic well and others.

The curve is based upon the equation

$$f(x) = 3x^2 - 2x^3 = 2x^2\left(\frac{3}{2} - x\right)$$

between the values of zero and one. The gradient is then

$$\frac{df(x)}{dx} = 6x(1 - x)$$

These can be scaled depending on the distance and height difference required. The gradient of $f(x)$ becomes zero at $x = 0$ and $x = 1$, ensuring the boundary between the smooth curve and the surrounding flat potential is continuous.

14.2 Double Well in 1-D

The double square well is perhaps the simplest useful case we can study. The electron begins in one square well and tunnels through to the other. This can

represent such systems as the scanning tunnelling microscope, and is the starting point for studies of outer-shell electron transfer in many simple reactions. An increase in the barrier potential should produce a concomitant decrease in tunnelling splitting, and thus an increase in the tunnelling time. Likewise, an increase in the barrier width should reduce the tunnelling time.

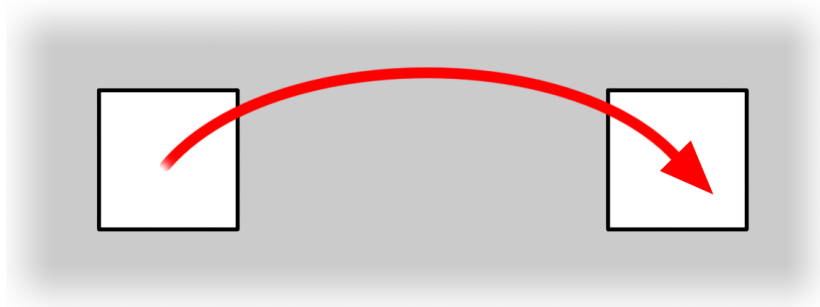


Figure 32: The double square potential well, the simplest case of electron tunnelling.

The square well is simple enough for analytical or numerical solutions to be computed, in order that we can compare results. The first such system is the one-dimensional double square well. A test was done with two wells at zero potential of width 2.0 Bohr separated by a 10.0 Bohr square barrier of height 0.5 Hartrees. The results are given below in Tables 3 and 4. The Grid Method (GM) results were calculated² using a grid-based method to solve the Schrödinger equation. The PIMC results were calculated using the approach outlined in this document. The total free energies do not agree exactly; this may be partly due to the fact that all we can compute using the path integral Monte Carlo method is the energy difference between a free electron and an electron sitting in one or both wells, rather than the total free energy. Statistical error in the PIMC approach, various systematic errors and sampling error in the grid-based approach account for most of the rest of the discrepancy, and the values calculated may benefit from longer computational runs. However, the free energy difference is in reasonable agreement between both methods, which is persuasive evidence that the path integral Monte Carlo approach is accurate and fit for purpose.

The one-dimensional double square well was then used again, and this time

²Calculated by Dr Massimo Mella using his own code.

Kink	Free Energy Grid Method	Free Energy PIMC
yes	0.407088	0.34928 ± 0.0025
no	0.284503	0.23409 ± 0.0031

Table 3: Calculated results for the one-dimensional double well system at a barrier width of 10.0 Bohr, with a barrier height of 0.5 Hartrees. Temperature is 8000K ($\beta = 39.5$). All data in atomic units. GM = Grid Method, PIMC = Path Integral Monte Carlo.

FE difference	Tunnelling Splitting
GM 0.1226	GM 0.000398
PIMC 0.1152 ± 0.004	PIMC $0.00054 \pm (2.1 \times 10^{-6})$

Table 4: Free energy differences and tunnelling splitting for the one-dimensional double well system at a barrier width of 10.0 Bohr, with a barrier height of 0.5 Hartrees. Temperature is 8000K ($\beta = 39.5$). All data in atomic units. GM = Grid Method, PIMC = Path Integral Monte Carlo.

the barrier height was varied from 0.025 Hartree to 0.1 Hartree (0.68eV to 2.72eV) with a barrier width of 2.0 Bohr, and a well each side of the barrier of width 4.0 Bohr. The results are given in Table 5. However, at these well configurations, the wavefunction of the electron is highly de-localised, as shown in Figure 33. Thus, the splitting results calculated by the grid method cannot be compared with these results, as the grid method makes the assumption of localised electrons.

Barrier Height	ΔE PIMC	Splitting
0.025	0.0114 ± 0.0142	$0.03232 \pm 4.59E - 04$
0.050	0.0115 ± 0.0148	$0.03223 \pm 4.78E - 04$
0.075	0.0119 ± 0.0150	$0.03171 \pm 4.77E - 04$
0.100	0.0125 ± 0.0152	$0.03097 \pm 4.72E - 04$

Table 5: Calculated results for the one-dimensional double well system at a barrier width of 2.0 Bohr, with a barrier height varied from 0.025 Hartree to 0.1 Hartree (0.68eV to 2.72eV), with a well width of 4.0 Bohr. Temperature is 8000K ($\beta = 39.5$). Using 5×10^7 chains, section re-growth method, 100 expanded ensembles. All data in atomic units. GM = Grid Method, PIMC = Path Integral Monte Carlo.

The one-dimensional double square well was explored with a different set of parameters, and this time the barrier height was varied from 0.025 Hartree to 0.1 Hartree (0.68eV to 2.72eV) with a well width of 6.0 Bohr, and a well separation (centre-centre) of 34.0 Bohr. The well configuration and accompanying wavefunction plot is shown in Figure 34.

Barrier Height	ΔE GM	ΔE PIMC	Splitting
0.025		0.0270 ± 0.0201	$0.01745 \pm 3.51E - 04$
0.050		0.0442 ± 0.0304	$0.00884 \pm 2.68E - 04$
0.075		0.0613 ± 0.0312	$0.00451 \pm 1.41E - 04$
0.100		0.0764 ± 0.0222	$0.00248 \pm 5.51E - 05$

Table 6: Calculated results for the one-dimensional double well system at a barrier width of 2.0 Bohr, with a barrier height varied from 0.025 Hartree to 0.1 Hartree (0.68eV to 2.72eV), with a well width of 4.0 Bohr. Temperature is 8000K ($\beta = 39.5$). Using 5×10^7 chains, section re-growth method, 100 expanded ensembles. All data in atomic units. GM = Grid Method, PIMC = Path Integral Monte Carlo.

14.3 Double Well in 2-D

The next system studied consisted of two square wells of side length 1.0 Bohr (including a continuous curve of width 0.1 Bohr at each face to avoid discontinuity), initially separated by a barrier of width 2.0 Bohr (actual barrier width rather than well centre-centre distance). The electron path starts at the very centre of well *a* and finishes at the exact centre of well *b*. The inter-well distance (actual barrier width rather than well centre-centre distance) was varied from 2.0 Bohr to 20.0 Bohr with the barrier height set at 0.05 Hartrees (1.36eV). The results are given in Table 7. All calculations were performed using the Lévy bisection method, with a total of 3.2×10^8 chains per expanded ensemble state, at a temperature of 8000K ($\beta = 39.5$). This temperature is low enough to be sure we are largely only calculating ground state energies, but high enough to keep the energy much higher than the tunnelling splitting we are trying to calculate.

The change as the width increases, as listed in Table 7, appears to reduce with increasing width. The free energy differences are plotted in Figure 35 and the tunnelling splitting is plotted in Figure 36. Clearly, width changes in a

narrow barrier have a much greater effect than width changes in a wide barrier—the response is not linear. This is to be expected and reflects the fact that the wavefunction of the electron experiences exponential decay within a barrier. Past a certain barrier width, the value of the wavefunction is small but does not decrease significantly.

Barrier Bohr	Kink	Free Energy Hartrees	Difference	Splitting	Time (a.t.u)
2.0	no	0.44886 ± 0.0032			
[1.06Å]	yes	0.45147 ± 0.0032	0.00261 ± 0.0046	0.04570 ± 0.00021	21.8810 ± 0.1
5.0	no	0.43768 ± 0.0027			
[2.65Å]	yes	0.45729 ± 0.0028	0.01961 ± 0.0043	0.02337 ± 0.0001	42.7996 ± 0.18
10.0	no	0.43799 ± 0.0032			
[5.29Å]	yes	0.50067 ± 0.0021	0.06268 ± 0.0039	$(4.269 \pm 0.017) \times 10^{-3}$	234.27 ± 0.91
15.0	no	0.43799 ± 0.0032			
[7.94Å]	yes	0.55440 ± 0.0017	0.11641 ± 0.0037	$(5.119 \pm 0.019) \times 10^{-4}$	1953.69 ± 7.1
20.0	no	0.43799 ± 0.0032			
[10.6Å]	yes	0.47294 ± 0.0014	0.18329 ± 0.0035	$(3.654 \pm 0.01) \times 10^{-5}$	27370.3 ± 96

Table 7: Results for the free energy difference versus an increase in barrier width, with a barrier height of 0.5 Hartrees.

14.4 Triple Well in 3D

The triple well system is one in which a central square well provides a temporary stopping point for an electron tunnelling through the potential barrier. Lin *et al.* [4] describe how the addition of water layers between the start and end points of an electron tunnelling path can significantly increase the transfer probability. Pockets of low potential formed by chance arrangements of layered water molecules provide “stepping stones” that the electron can utilise to increase the tunnelling probability. Many biological systems may benefit from this type of intermediate well. A square well should provide the same effect, allowing an electron to tunnel more easily between the two sites.

The system studied consisted of two square wells of side length 1.5 (including a continuous curve of width 0.25 at each face to avoid discontinuity), initially separated by a centre–centre distance of 10.0. A centre well of side length 1.5 is included, midway between each outer well. The electron path starts at the very centre of well *a* and finishes at the exact centre of well *b*. The centre well potential

was varied from 0 Hartrees through to 1.0 Hartrees in steps of 0.1. The results are given in Table 8. All calculations were performed using the Lévy bisection method, with a total of 3.2×10^8 chains per expanded ensemble state, apart from the values around 0.5 Hartrees, which were run for longer in order to reduce the error bars.

Well Floor Hartrees	Free Energy Difference
0	0.397 ± 0.0012
0.10	0.401 ± 0.0012
0.20	0.410 ± 0.0012
0.30	0.415 ± 0.0012
0.40	0.422 ± 0.0012
0.45	0.420 ± 0.00085
0.50	0.419 ± 0.0012
0.55	0.430 ± 0.00085
0.60	0.435 ± 0.0012
0.70	0.439 ± 0.0012
0.80	0.444 ± 0.0012
0.90	0.447 ± 0.0012
1.00	0.454 ± 0.0012

Table 8: Results for the triple well system, with a varied centre well depth.

There is a dip visible in the plot (Figure 38) at around 0.5 Hartrees. The free energy dips significantly here, compared to the rising trend, showing that this well potential would allow electrons to tunnel in slightly less time than for the well floor heights represented by adjacent data points. It is suggested that this is due to a mixing of the ground state of the outer wells with an excited state of the central well at around 50% well depth. It would be rather interesting, given more time, to explore how the plot of free energy against well depth changes with different centre-well to outer-well size ratios.

14.5 Discussion

Initially we should consider whether the electron tunnelling times calculated are physically reasonable. The conversion factor for atomic time units to seconds is 1 atomic time unit = 2.418885×10^{-17} seconds. This means that the times

listed in the 2-D double well section—for example, at 5.0 Bohr the time is 42.8 time units—are of the order of one femtosecond (*i.e.* 10^{-15} seconds). Firstly, let's calculate roughly whether such a speed is feasible with respect to special relativity, ignoring for a moment energy considerations and any suggestion that the electron tunnels instantaneously. The distance tunnelled is 5.0 Bohr $\simeq 0.265 \times 10^{-9}$ m. Dividing by $c = 3.0 \times 10^8 \text{ms}^{-1}$ to find the minimum time to travel this distance we get 8.9×10^{-19} seconds—so our tunnelling value is well within the possible maximum electron speed imposed by relativity (in fact it is around $0.0009c$, which is less than the speed reached by electrons in atomic orbitals in hydrogen). Taking the standard (non-relativistic) $E = \frac{1}{2}mv^2$ and using a velocity value of $v = \frac{5.0}{42.8} = 0.117$ Bohr/atu, then (recalling that for an electron in atomic units $m = 1$) $E = 0.0068$ H = 0.19eV, which is a reasonable rough value for kinetic energy for an electron.

The 1-D square well results match the results calculated by another method detailed in Section 14.2 well enough; this agreement is good evidence that the method works well and is capable of producing accurate and useful predictions about the behaviour of a system.

The 2-D and 3-D well results demonstrate that the method is physically realistic; the exponential fall-off in tunnelling splitting with increased barrier width is exactly what would be expected based upon physical reasoning. The feature shown at 0.5 Hartrees in the triple well results shows that the method is highly sensitive to free energy changes caused by mixing of ground and excited states in different well geometries.

Each of the calculations listed was performed within twenty hours using 64 CPU cores on a powerful computing cluster; hence, we conclude that the method is able to give precise, accurate answers within a reasonable time frame, given sufficient computing power. The computing power (*i.e.* number of CPU cores) can be reduced at the expense of either extended computing time or reduced precision.

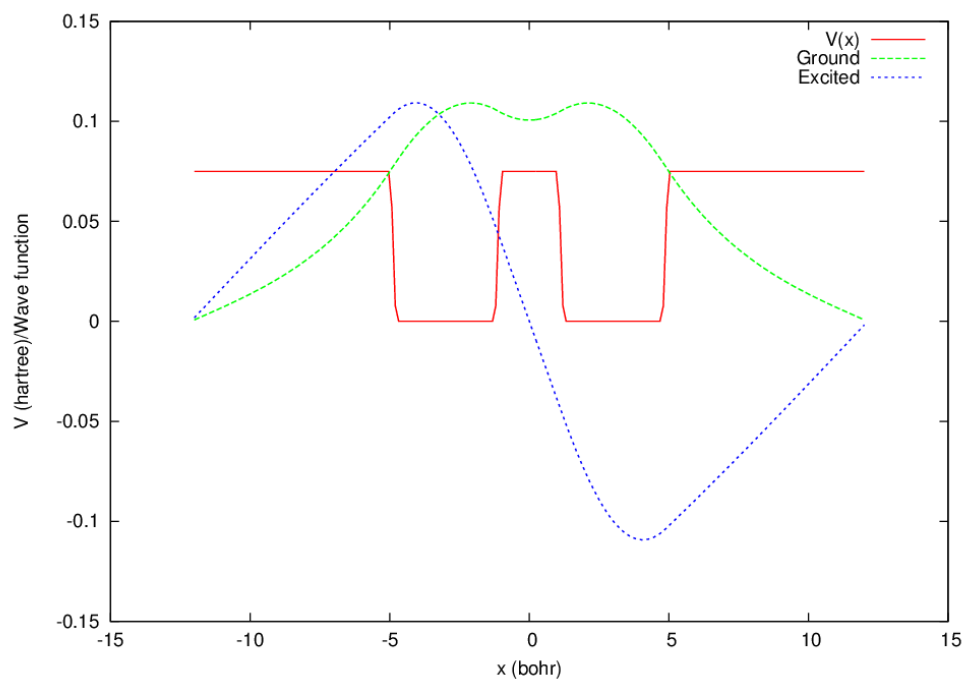


Figure 33: De-localised wavefunction in narrow, close-together potential wells.

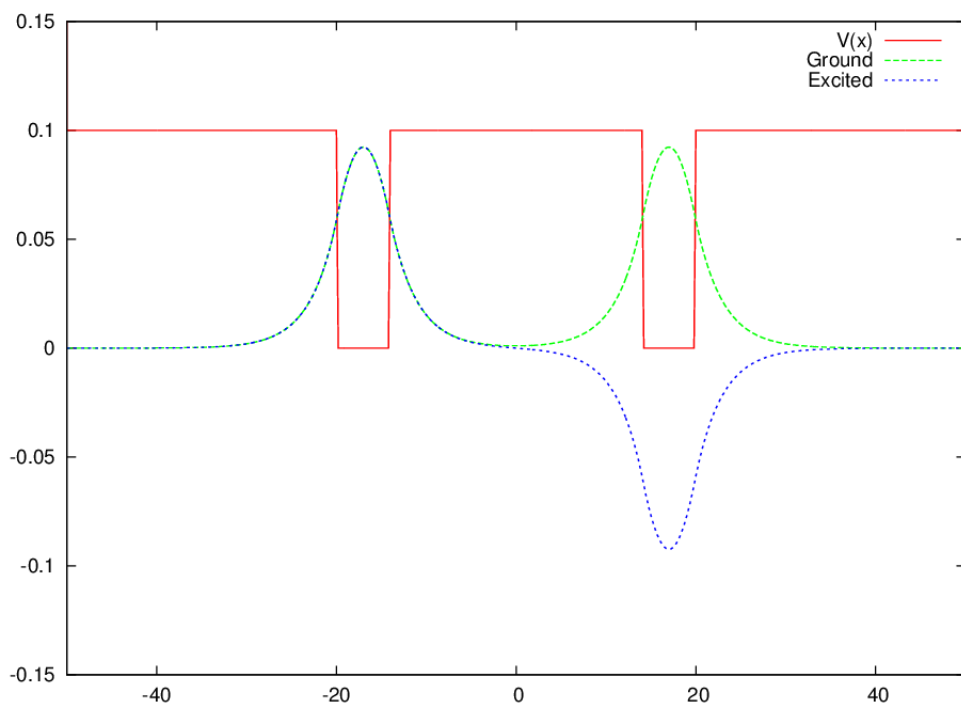


Figure 34: Wavefunction in 6.0 Bohr potential wells with 34.0 Bohr centre-centre separation.

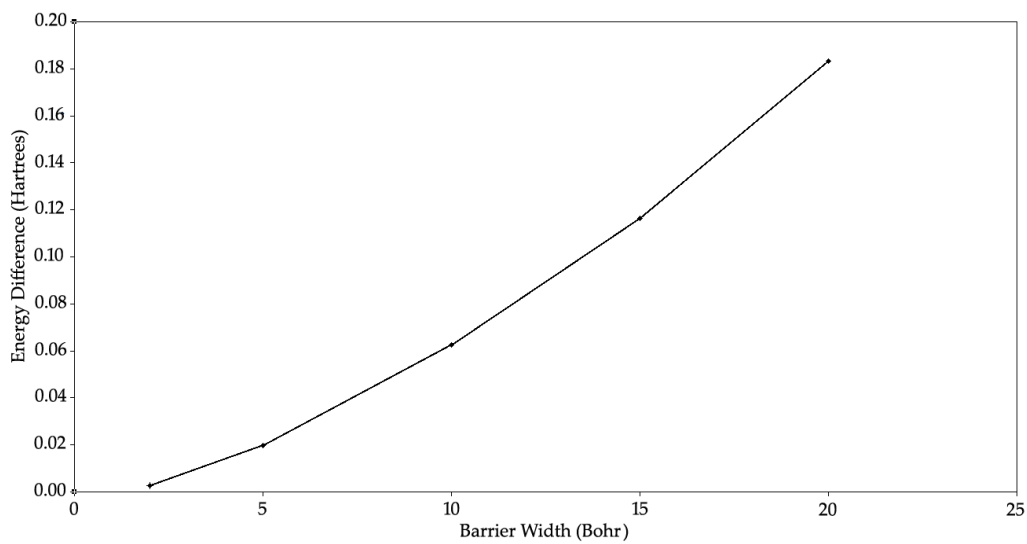


Figure 35: The free energy difference (Hartrees) with barrier width (Bohr).

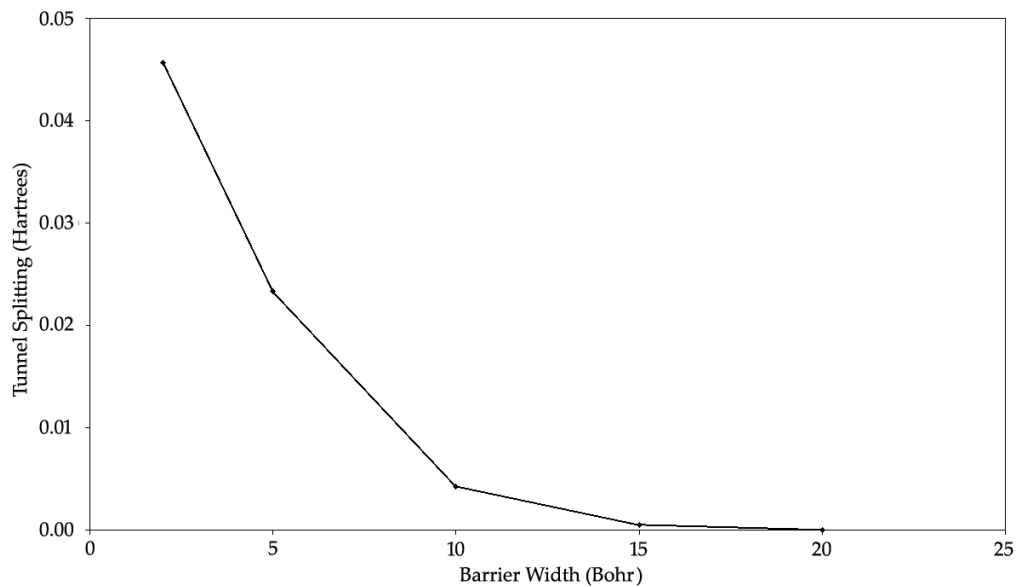


Figure 36: The tunnel splitting (Hartrees) with barrier width (Bohr).

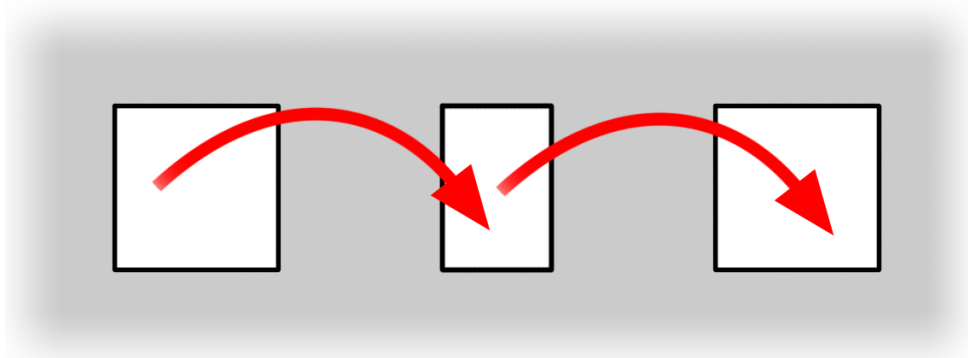


Figure 37: The triple square potential well, allowing the electron a “stepping-stone” between start and end points.

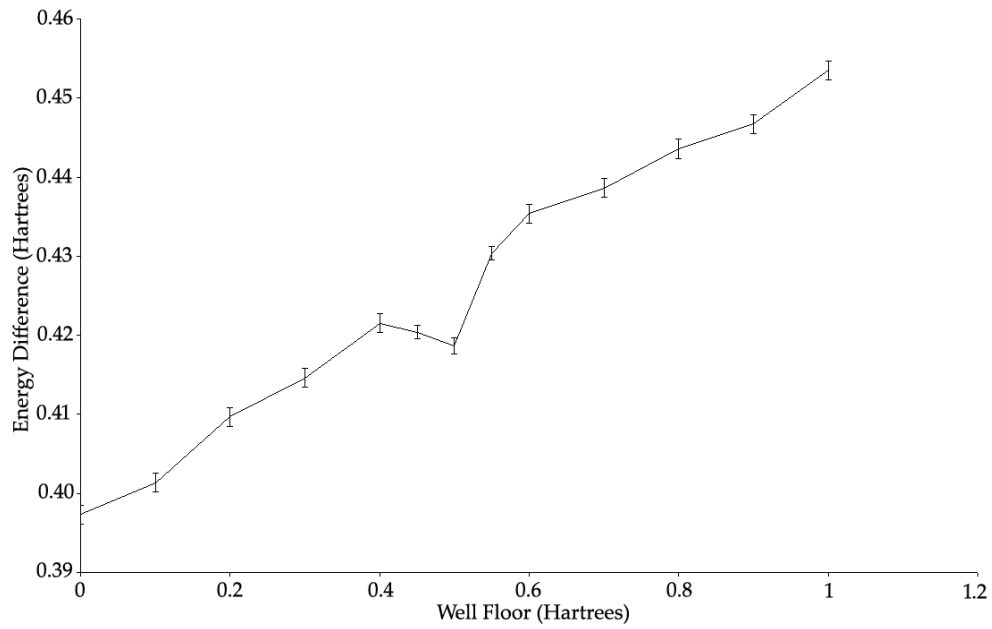


Figure 38: The tunnelling time with increased well depth for the triple square potential well, allowing the electron a “stepping-stone” between start and end points. There is a large dip at around 0.5 Hartrees depth, increasing the electron’s transmission probability.

15 The Argon Atom Model

15.1 Argon Barrier

In order to prove that the method developed here works correctly, it is important to subject it to tests beyond simple square barriers. A basic test that provides a more complex environment is an array of atoms set in a square barrier. Argon atoms were chosen due to their inert character and favourable size; however, the atom chosen can only be represented by a rough potential model consisting of the Van der Waals radius, covalent radius and equivalent potential barrier height. Quantities such as atomic mass cannot be represented, although we are assuming the Born–Oppenheimer approximation and the static medium assumption thus nuclear mass and nuclear motion can be neglected. We are also assuming the atoms are in the ground state and not ionised.

The atoms are arranged in a two dimensional hexagonal lattice, set four-deep into a square cross-section potential barrier. The distance between the atoms is 7.21 atomic units or 3.82Å. The argon atom potential height is set at 0.73, and the radius is around 1Å. Each atom is modelled by a polynomial potential shown in Figure 39 between 0 and 7 units of radial distance, and zero beyond this. The polynomial was fitted to an electron–argon pseudopotential described by Li *et al.* [27] and Lopez–Castillo *et al.* [28]. The square barrier is of height similar to the argon atom maximum potential height at 0.87 Hartrees. The total maximum barrier is therefore 1.6 Hartrees (43.5eV), at the point where where an atom centre sits within the square barrier. The entire system was studied in two dimensions in order to eliminate the need for a more computationally expensive three-dimensional lattice of atoms.

15.2 Results

To show the method is working, the electron path must avoid the argon centres whilst finding the shortest path, and we must be able to calculate a tunnelling time (Chapter 9). Figure 42 shows the histogram calculated—the electron is shown avoiding the argon centres while taking the quickest path from point to point. The contour lines use d_{in} in the plot were chosen for each element (potential and histogram) to maximise the visibility of features. The free energy calculations

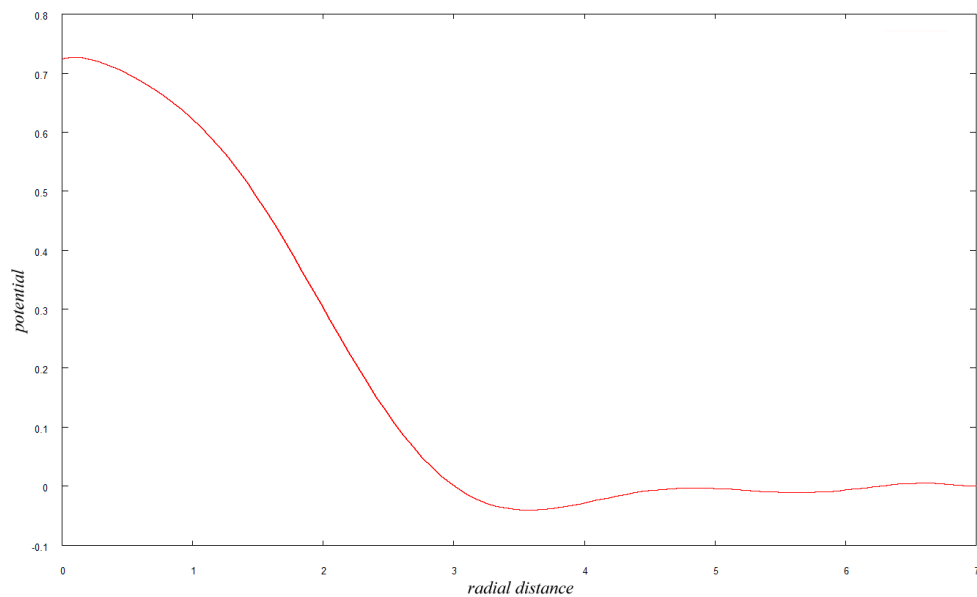


Figure 39: The polynomial used to approximate the atomic potential of argon.

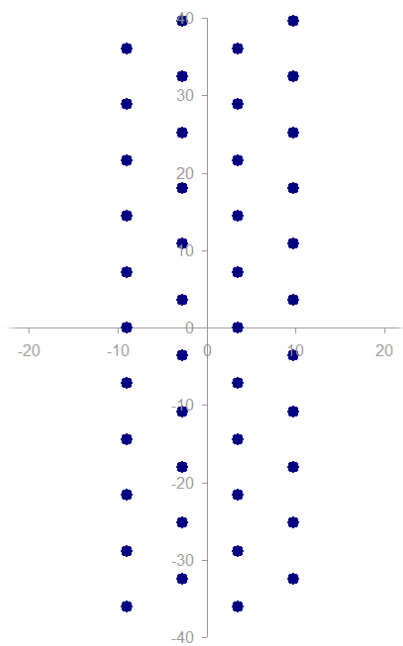


Figure 40: The arrangement of argon atoms in the potential barrier (top down view). The electron crosses in a horizontal direction.

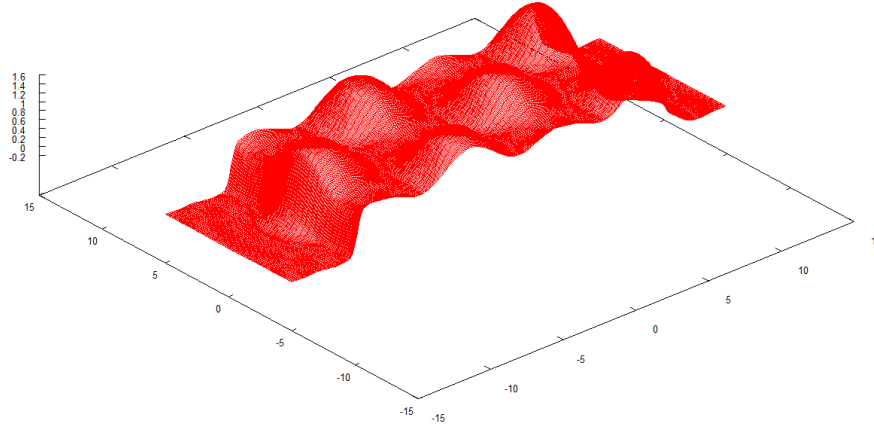


Figure 41: A slice through the potential showing the relative size and the arrangement of argon atoms in the potential barrier.

were performed at 8000K, *i.e.* with $\beta = 39.47$.

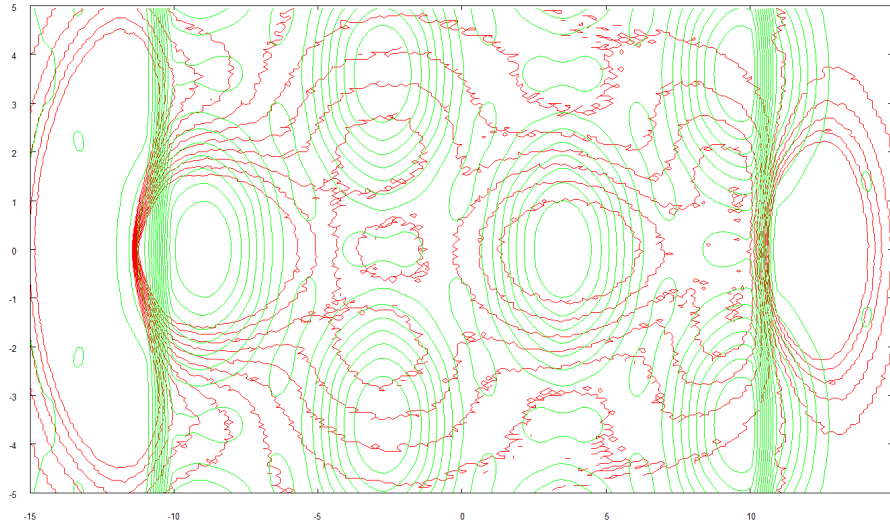


Figure 42: A contour plot of the histogram of the electron travelling through an argon barrier (red), overlaid onto the contours of the argon potential (green). The hills and valleys caused by the argon centres are clearly visible in the plot. Distances (both axes) are in Bohr. Contour lines for each plot are arbitrarily chosen to maximise visibility.

The free energies calculated are given in Table 9. The tunnelling time is a large value, at $(1.64 \pm 0.24) \times 10^{15}$ a.u. ($[3.96 \pm 0.57] \times 10^{-2}$ seconds), suggesting

that tunnelling with these parameters would not be likely to occur. The main problems are the long distance required, and more importantly, the height and width of the potential the atoms are set into.

Kink	Free Energy	Difference
yes	0.8409 ± 0.0074	—
no	0.029 ± 0.14	0.8119 ± 0.144

Tunnelling Splitting	Tunnelling Time
$(6.112 \pm 0.88) \times 10^{-16}$	$(1.64 \pm 0.24) \times 10^{15}$

Table 9: Results for the tunnelling time through the argon atom barrier.

16 Nanotubes

16.1 Nanotube Conductivity

One problem that could benefit enormously from a technique able to calculate electron transfer through complex potentials is the study of conductivity through carbon nanotubes. In particular, networks of carbon nanotubes may or may not allow easy transport of electrons depending on the orientation, organisation and concentration of the network. Previous studies, such as Wescott *et al.* [29], have approached the problem by using a single value for the conductivity along the length of the nanotube, and simulating nanotube conductivity through networks using dissipative particle dynamics, or by using an analytical model for conductivity, such as the work by Deng *et al.* [30].

The approach taken here is to model nanotubes as cylindrical regions of lower potential embedded in regions of higher potential. The path integral Monte Carlo approach can then be used to model the electron path and find the tunnelling probability, which is directly linked to the conductivity.

A model of potential was built in which cylindrical tubes of arbitrary length and radius can be placed anywhere within the system, to represent carbon nanotubes. The potential inside and outside the tubes can be specified. No checks are made as to whether nanotubes overlap, so it is the responsibility of the program user to choose non-intersecting nanotubes. When building the potential, it was important to ensure that the cylinders were compatible with the force-bias algorithm and the gradient calculation used in the fourth-order propagator by avoiding discontinuities (see discussion in Section 6.5).

16.2 Building the Potential

The system is initialised by defining the radius, length, location and orientation of every nanotube. Since a nanotube can be assumed cylindrically symmetric the total location and orientation of the tube can be represented solely by giving the coordinates of the end points of the tube axis. Some other details are also calculated at load time, such as the square of the radius and length.

To find the potential at any point given in space, the first task is to determine

whether it lies inside any of the nanotubes. The point should only lie inside one nanotube, as obviously two nanotubes cannot occupy the same physical space. This test can be performed using a simple test based on vector operations. First, the given point must be translated so that one end of the tube of interest becomes the origin. A rough check is to dot-product the vector from the given point to one end of the cylinder with the vector between both ends of the cylinder. A dot-product that is negative or greater than the length of the cylinder implies that the point is beyond the ends of the cylinder.

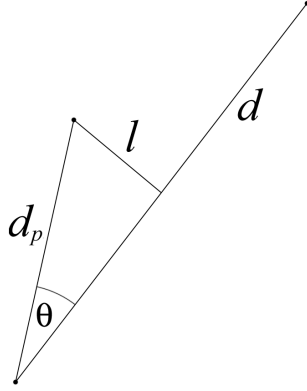


Figure 43: Finding whether a point is inside a cylinder.

We then need to find the distance between the point and the length axis of the cylinder in order to test it against the radius. Here, d is the vector from one end of the cylinder to the other and d_p is the vector from one end of the cylinder to the target point, as in Figure 43. The distance between the point and the cylinder axis is l , which is given by

$$l = |d_p| \sin(\theta) \quad (98)$$

$$l^2 = |d_p|^2 \sin^2(\theta) = |d_p|^2 [1 - \cos^2(\theta)] \quad (99)$$

since $\cos^2(\theta) + \sin^2(\theta) = 1$. Of course, the dot product is defined as

$$d \cdot d_p = \cos(\theta) |d| |d_p| \quad (100)$$

so

$$\cos(\theta) = \frac{d \cdot d_p}{|d| |d_p|} \quad (101)$$

thus

$$l^2 = |d_p|^2 \left[1 - \frac{(d \cdot d_p)^2}{|d|^2 |d_p|^2} \right] = |d_p|^2 - \frac{(d \cdot d_p)^2}{|d|^2} \quad (102)$$

Comparing l^2 against the squared radius of the cylinder tells us whether the point is inside or outside the volume of the cylinder.

Once we have a broad-phase picture of which cylinder the point is within (if any), we can then build the detailed potential model. If the point is outside of all cylinders, the potential returned is simply a single number, representing the potential of the surrounding medium. If inside a nanotube cylinder, the potential must be built up depending on which region the point is located within.

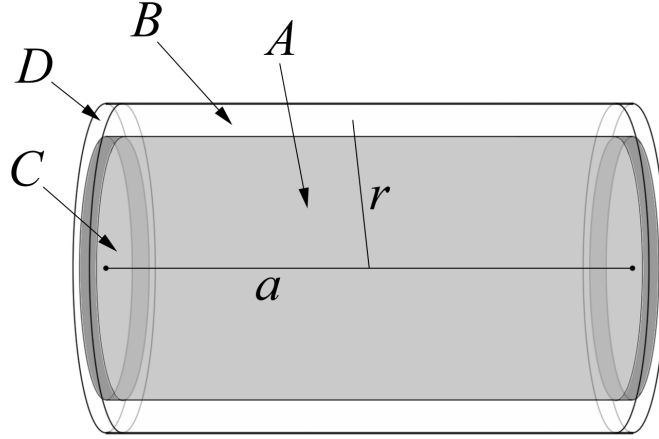


Figure 44: The cylinder can be divided into four regions: region A is a cylinder, region B is a cylindrical shell, region C is a thick disc and region D is an “annular” cylindrical shell.

The cylinder can be divided into four distinct regions, as illustrated in Figure 44. Region A is the central cylindrical area, within which the potential will be a single number representing the potential inside a nanotube. The other three regions contain a potential gradient (see Section 14.1), allowing a smooth transition between the external and internal potentials. Region B is a cylindrical shell; within this region the potential needs to transition in a radial direction. Region C is a solid disc, within which the potential must transition in an axial direction. Region D is the most complex region. Within this region the potential gradient

must always lie within a plane formed by the radius vector r and the cylinder axis a , and it must always point towards the curved line formed by the outermost circumference of A at its end, labelled c in Figure 45.

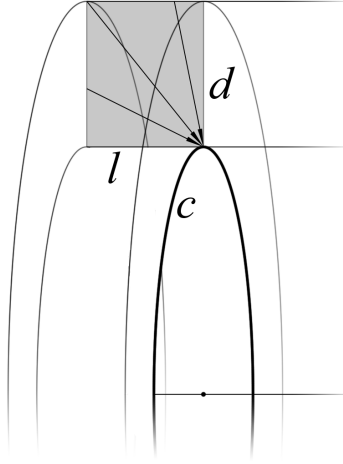


Figure 45: Region D is the most difficult potential region to calculate. The potential gradient must be towards (assuming the potential inside the tube is lower than outside) the line c and must lie in the radial-axial ($l-d$) plane.

The point must be transformed into the reference frame of the cylinder, with the cylinder's geometrical centre as the origin and the cylinder axis aligned with the primary axis of our coordinate system. Working in cylindrical coordinates (r, φ, h) , the φ coordinate is unused for calculating scalar potential (since the gradient is always within the radial-axial plane) but necessary for returning the gradient vector of the potential. While calculating the scalar potential for region D is not too hard, the force-bias algorithm requires the potential gradient, which is calculated by finding the vector that points towards the inner circle from the point specified, and scaling it by the magnitude of the gradient.

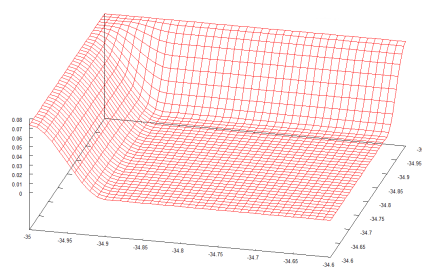
16.3 Testing the Potential

The model works for nanotubes at any alignment, although many of the simplest and most useful test systems involve nanotubes aligned in the plane. By plotting the potential as usual, we can check whether the potential is working as expected.

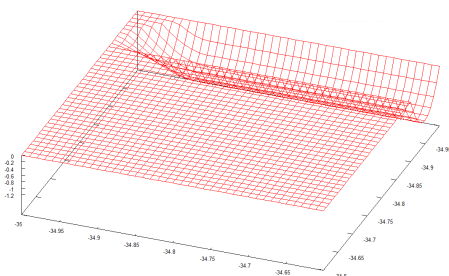
The plots in Figure 46 show the potential and gradient within the 2-D x - y plane for one corner of a nanotube, while Figure 48 shows the scalar potential for nanotubes at arbitrary alignments. Figure 47 shows the vector field of one entire end of a tube.

The first test to perform is to make sure the nanotube potential code works correctly with the *PIMC* code. To this end, a nanotube layout was created whereby two nanotubes are almost aligned axially end-to-end, except that the first is rotated slightly so that the two tubes abut radially rather than axially, as in Figure 49. With this alignment, it should be possible to see the electron path tunnelling preferentially through the angled tube rather than in a straight line through the barrier.

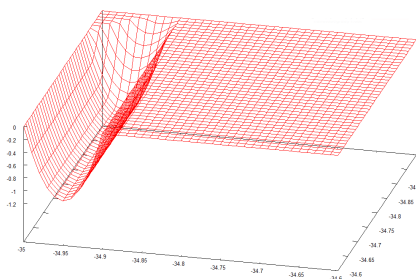
In Figure 50 the histogram is shown for an electron tunnelling from $(-3, 0, 0)$ to $(3, 0, 0)$ through the potential shown in Figure 49. The detour the electron takes in order to remain within the low-potential area is visible. In order to hop from one nanotube to the next, the electron must tunnel through the high potential barrier.



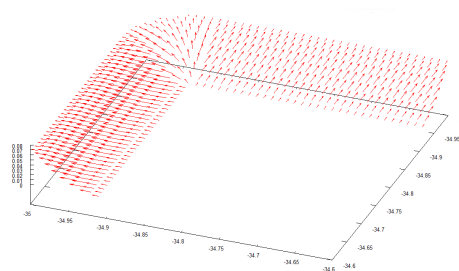
(a) Scalar potential



(b) Gradient in the x direction



(c) Gradient in the y direction



(d) Gradient shown as a vector field

Figure 46: A 2-D slice through the potential region in one corner of a nanotube, showing the smooth fall-off required between the exterior and interior of the tube.

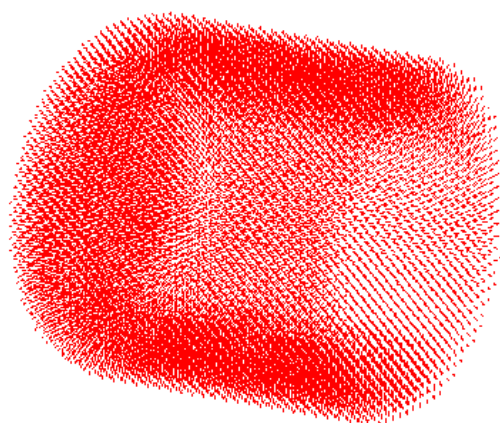


Figure 47: The entire of one end of a nanotube. The vector arrows are too small to distinguish, but it demonstrates that the overall shape of the tube is being correctly generated. The “cap” of the tube is to the left side of the image.

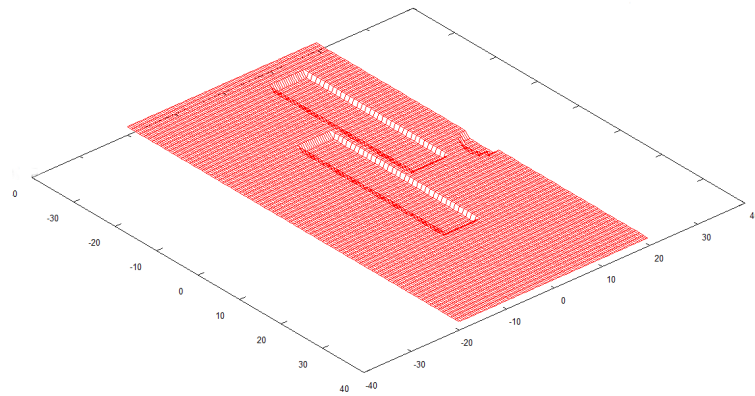
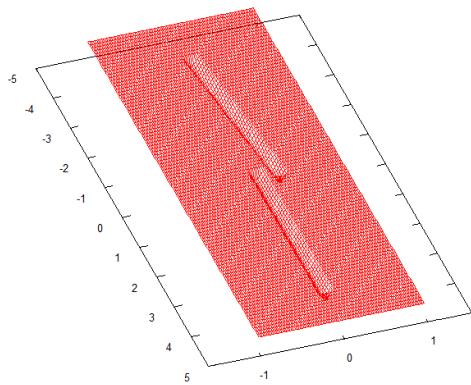
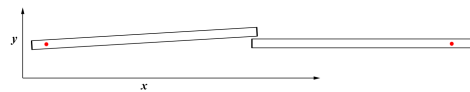


Figure 48: The outline of three nanotubes sitting in a potential field—two aligned with the x -axis, and one rotated through 90° around the y -axis.



(a) Scalar potential



(b) Outline diagram—the red points indicate the start and end points of the chain

Figure 49: The outline of two nanotubes sitting in a potential field—both roughly aligned with the x -axis, but with one rotated through 3.4° around the z -axis so that they meet edge-to-edge rather than end-to-end.

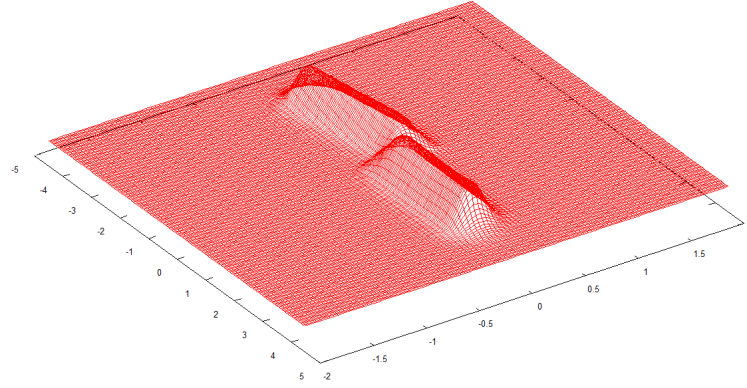


Figure 50: The histogram produced by an electron tunnelling from $(-3, 0, 0)$ to $(3, 0, 0)$ through the potential shown in Figure 49. The barrier height was set to an extreme value of 100 Hartrees (2.721keV) in order to clearly show the effect by only allowing the electron to exist (at significant probability) within the tubes. Note that the tubes used here are significantly thinner than realistic nanotubes.

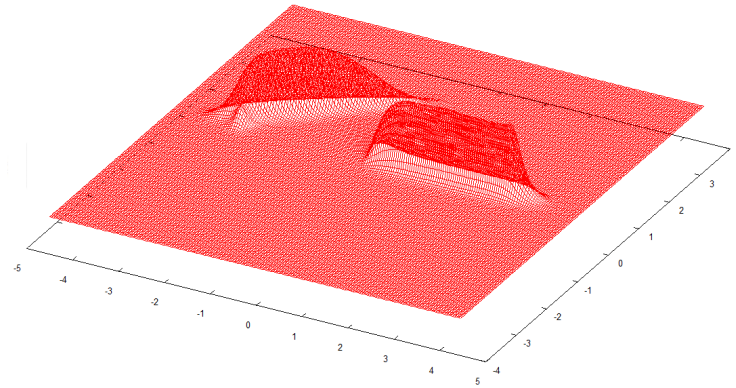


Figure 51: The histogram produced by an electron tunnelling from $(-3, 0, 0)$ to $(3, 0, 0)$ through a potential similar to that shown in Figure 50, except with the rotated tube aligned at a greater angle. Note that the tubes used here are significantly thinner than realistic nanotubes.

16.4 Test Cases

We are assuming single-walled nanotubes, which generally have a diameter of 5–10 Å. We shall assume 5 Å ($\simeq 10$ Bohr). The length of real nanotubes can be any value, sometimes extending well beyond the atomic distance scale to macroscopic lengths. The test cases studied here generally use nanotubes with an aspect ratio of up to 50—*i.e.* a length of 250 Å or around 500 Bohr. Since nanotubes conduct well along their length, the potential inside is set to 0. The external potential is set to 0.5 Hartrees.

An interesting case to study consists of the situation where three nanotubes are initially aligned end-to-end with a small barrier in between each, which we shall denote A – B – C , with the electron tunnelling from A through B to C (see Figure 52(a)). As the central tube B is rotated gradually out of alignment we shall examine the effect on electron tunnelling time. The results are given in Table 10.

Secondly, the effect of nearby nanotubes on direct tunnelling routes should be explored. Even when the tunnelling path seems obvious, another nanotube placed nearby, offset from the lowest energy path, may affect the tunnelling time, since the electron path consists of the sum over all possible histories. The tunnelling of an electron from one nanotube to another, placed end-to-end, was studied, with a third tube aligned in the same direction as the other two but positioned off to one side (see Figure 52(b)). The initial end-end nanotube gap was set to 5.0 Bohr, and the distance of the third nanotube (centre axis to centre axis) was set to 6, 8 and 12 Bohr. The results are given in Table 11.

Both systems were studied at a temperature of 8000K ($\beta = 39.5$), using the section regrowth method with 511 free beads.

16.5 Results and Discussion

The tables (Table 10 and 11) and plot (Figure 53) clearly indicate that the distance required for tunnelling makes it extremely unlikely overall; however, the numbers still show the relationship expected. The tunnelling probability falls off as the nanotube rotates in the first case. Small displacements from parallel produce large changes in free energy; the change is reduced for large angles. In

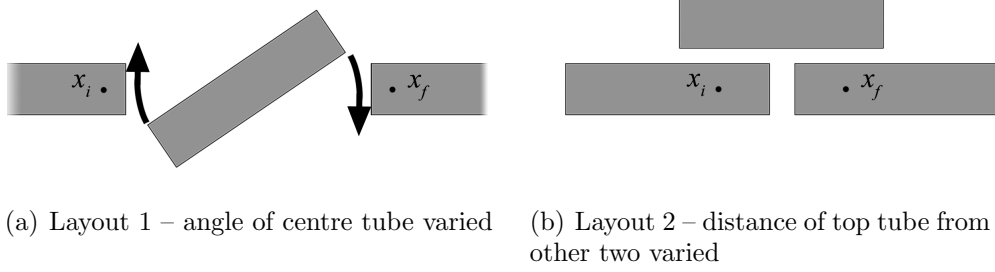


Figure 52: The two different geometries studied. In each case one parameter was varied. The chain in the stretched case runs $x_i \rightarrow x_f$.

other words, nanotube conductivity can be greatly affected by the nanotubes being slightly out of alignment rather than in parallel.

The parallel study shows that having another tube nearby but not directly lying in the line from start point to finish point can affect the tunnelling probability even though the classical electron path might not be expected to pass through it, so long as the parallel tube is physically very close. If the distance increases by even one ångström, the tunnelling falls off. Overall, due to the lower distance and reduced number of barriers, the tunnelling for this system is much more likely than for the previous nanotube system.

Note that the values calculated here correspond to the likelihood of an electron *spontaneously* tunnelling along the nanotubes. Adding a driving potential (*i.e.* a voltage, as in real electrical components) would increase the tunnelling probability, although the asymmetry introduced by this would require further terms in the calculation of the tunnelling splitting. The numbers generated here still give a feel for the relative likelihood of tunnelling over various configurations.

Angle degrees	F.E. Difference Hartrees	Tunnelling Splitting Hartrees	Tunnelling Time atomic time units
0	0.9335 ± 0.031	$(5.028 \pm 0.16) \times 10^{-18}$	$(1.989 \pm 0.062) \times 10^{17}$
15.0	1.0811 ± 0.027	$(1.483 \pm 0.04) \times 10^{-20}$	$(6.744 \pm 0.18) \times 10^{19}$
30.0	1.1840 ± 0.025	$(2.554 \pm 0.06) \times 10^{-22}$	$(3.92 \pm 0.09) \times 10^{21}$
45.0	1.2169 ± 0.0025	$(6.971 \pm 0.17) \times 10^{-23}$	$(1.435 \pm 0.036) \times 10^{22}$

Table 10: Results for the tunnelling time (in atomic time units) as the centre nanotube is rotated away from parallel.

Distance Bohr	F.E. Difference Hartrees	Tunnelling Splitting Hartrees	Tunnelling Time atomic time units
6.0	0.1982 ± 0.031	$(2.026 \pm 0.063) \times 10^{-5}$	$(4.94 \pm 0.15) \times 10^4$
8.0	0.2059 ± 0.032	$(1.4974 \pm 0.047) \times 10^{-5}$	$(6.68 \pm 0.21) \times 10^4$
12.0	0.2061 ± 0.032	$(1.4871 \pm 0.047) \times 10^{-5}$	$(6.72 \pm 0.21) \times 10^4$

Table 11: Results for the tunnelling time (in atomic time units) as the parallel nanotubes are separated.

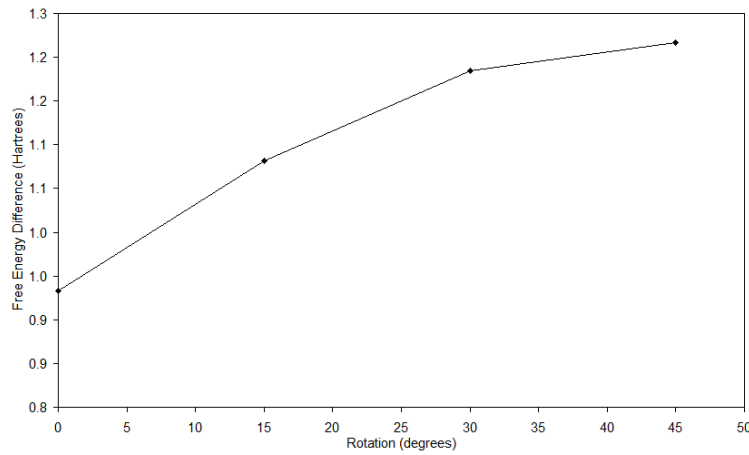


Figure 53: A plot of how the free energy difference changes with centre nanotube angle (in degrees).

Part V

Conclusion

17 Conclusion

The path integral Monte Carlo expanded ensembles method for calculating free energies and tunnelling times is sufficiently accurate and entirely feasible. It is capable of handling complicated potentials and it is inherently parallelisable and quick. The main developments outlined here are the combination of the multi-slice metropolis Monte Carlo path integral methodology with the method of expanded ensembles, which provides a straightforward and versatile methodology for computing free energy differences, and hence tunnelling splitting and tunnelling times. Due to the rapid decorrelation required by the method of expanded ensembles, the multi-slice Lévy bisection method is an essential ingredient in making the approach competitive.

17.1 Advantages

The approach outlined has several advantages. Firstly, the path integral Monte Carlo approach is based on a semi-classical approximation that becomes exact as the number of slices approaches infinity. This means in practice that only a few hundred slices are needed to achieve a precise answer. It can be used to compare with analytical answers in simple systems, or to calculate tunnelling times in complicated situations.

Secondly, the method can handle complicated potentials of any shape, so long as they can be modelled. Potentials with deep enclosed wells do not suffer from problems associated with ergodicity (*i.e.* the beads do not become trapped in wells) due to the dual benefits of the Lévy bisection method and the method of expanded ensembles.

The method is easily distributable—the interconnect speed between cores is not an issue, since little inter-process communication is needed and therefore the code could be rewritten to run on separate desktop machines with only a standard low-speed network connection or even machines connected by the Internet.

Lastly, since the method is based upon statistical sampling, a rough answer can be obtained with little expenditure of CPU time. The first couple of digits of precision are arrived fairly quickly, and further time is used only to refine the answer to the required level of precision. This means that system behaviour can be explored initially very rapidly although in a coarse manner. Furthermore, if greater accuracy is required, the data can be reloaded to allow for further refinement.

17.2 Drawbacks

While the method is useful for many situations, there are a limited number of drawbacks. Firstly, without the benefit of parallelisation the method can take a long time to run. Many of the results reported earlier were generated using 32 processor cores within ten hours; using only one core (of comparable speed) to converge to the same level of precision would therefore take around two weeks. This may not be a problem for most researchers, as access to powerful computing clusters is widespread, and the method is highly parallelisable, as detailed above.

Furthermore, the potentials used are necessarily pseudo-potentials of a relatively coarse nature. The exact potential field around an atom due to electron orbitals is more complex than the simple model used in, for example, the argon atom studies detailed in Chapter 15. Electron correlation is not accounted for, and at larger scales even modelling potentials atom-by-atom might prove costly. However, a wise choice of geometry could account for this, and atom-by-atom modelling of thousands of atomic potentials could be attempted by employing some form of space partitioning to reduce CPU load.

Due to the requirement put forth in Chapter 9, Equation 86, the temperature range is limited, in order to ensure we are primarily calculating the ground state energy of the electron. This also means that some consideration must be given to the correct temperature at which to run the simulation. Basic calculations can be performed with simple equivalent systems to determine the temperature at which the simulation should be run.

17.3 Further Work

There remain some tasks to be completed, that would improve the method as described here. First and foremost, the Lévy bisection method could be extended to use the fourth-order propagator given in Chapter ?? . The fourth-order propagator is similar to the second-order propagator in the kinetic part of the chain action—the only difference is in the calculation of the potential part. The Lévy bisection method works by discarding the kinetic part of the energy propagator (the algorithm takes care of correct sampling in this respect) and calculating acceptance or rejection based upon the potential part alone. Therefore, the potential part of the fourth-order propagator could be used to select chains, reducing the number of beads required. While the fourth-order method requires marginally more CPU time due to the extra work associated with calculating the potential gradient, it is expected that the benefit of being able to use fewer time slices would outweigh this cost. In fact, since the gradient calculations are based upon analytical gradients for all of the model potentials used here, the cost of calculating a gradient is minimal.

The method of expanded ensembles relies for efficiency upon the flat histogram approach, in which the energy spacing between states remains constant and the weighting factors are optimised in order to ensure all states are visited equally. There are other approaches to optimising the method. For example, the energy spacing between states can be optimised in order to ensure the states are all visited equally; or, the weights can be optimised to maximise the number of round trips performed from the minimum state to the maximum [31, 32]. These could all be explored for the methods and systems described here in order to find a more efficient formulation of the method of expanded ensembles.

17.3.1 Fourth-Order Propagator

The first-order propagator given in Chapter 3, Equation (29), allows a basic splitting of the path into discrete slices. A higher-order propagator would provide a better quality path splitting, and should allow for good results with fewer beads. It is somewhat analagous to using a higher-order polynomial to fit to a data set.

The fourth-order propagator is given by [33]

$$\int d\mathbf{r}'' \exp \left[-m \frac{(\mathbf{r}' - \mathbf{r}'')^2}{\tau} \right] \exp \left[-\tau \tilde{V}(\mathbf{r}'') \right] \exp \left[-m \frac{(\mathbf{r}'' - \mathbf{r})^2}{\tau} \right] \exp \left[-\tau \frac{V(\mathbf{r}) + V(\mathbf{r}')}{6} \right] \quad (103)$$

where

$$\tilde{V}(\mathbf{r}) = \frac{2}{3}V(\mathbf{r}) + \frac{\tau^2}{72}[\nabla V(\mathbf{r})]^2 \quad (104)$$

and ∇ is the usual gradient

$$\nabla = \hat{i} \frac{\partial}{\partial x} + \hat{j} \frac{\partial}{\partial y} + \hat{k} \frac{\partial}{\partial z} \quad (105)$$

Note that the fourth-order propagator behaves differently to the second-order propagator. The point \mathbf{r}'' is an intermediate point mid-way between the two beads \mathbf{r}' and \mathbf{r} —essentially we can create a virtual bead at the mid-point. This causes extra computational load, but should enable the correct sampling of the electron path using fewer time slices. The interaction contributions are the same as for the second-order propagator. Alternatively, the existing beads can be utilised such that odd-numbered beads (bead numbering begins at zero) represent the midpoints or virtual beads and even-numbered beads represent the real beads.

Combining the fourth-order propagator with the equations of the force-bias method has not been done, but it can be used with the single-slice basic method easily—the second-order calculation of the chain action is replaced by a fourth-order propagator calculation. When combined with the multi-slice algorithm the fourth-order propagator should allow for very fast convergence with fewer beads than the second-order propagator. The fact that the Lévy bisection method is based upon whole-chain moves should allow for fewer calculations than using the fourth-order propagator in a single-slice capacity since single bead moves will cause each intermediate, virtual bead gradient to be evaluated twice (once for the bead to the left of it, and again when the bead to the right of it moves).

Furthermore, more situations and systems could be studied, both simple and complex. Such systems are described below.

17.4 Further Applications

Many more studies could be done using the methodology developed here. The simple systems of nanotubes described in Chapter 16 could be extended to the

concept of networks of tubes, set into a substrate of high barrier potential. This method should be able to calculate tunnelling rates (and therefore, ultimately, conductivity) for any complicated nanotube network, that could be initialised with a simple piece of code. Wescott *et al.* [29] using a model of conductivity through nanotubes study such a system of nanotubes set into a substrate consisting of two mixed polymers, while Deng and Zheng [30] give an analytical model potential for the conductivity of nanotubes that could be used to calculate a realistic potential value.

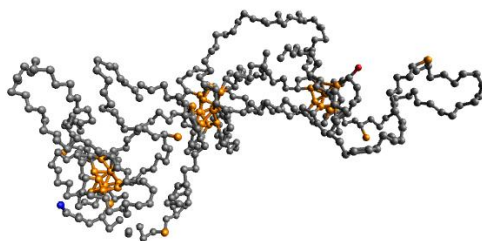


Figure 54: Multiple cores formed by a polymer in solution with hydrophobic substituents.

A system that is interesting to examine in detail is one described by Mella and Izzo [34] in which they perform Monte Carlo simulations of polymeric species in solution, with covalently linked substituents of different solubility. They studied the effect of hydrophobic substituents and found that the polymer undergoes a structural phase transition from disordered to compact, with a sufficiently high number of interacting substituents forming multiple hydrophobic cores (see Figure 54). The issue of how such polymers behave in solution is highly relevant to biological sciences, and medicine in particular, as often active drug centres are attached to polymers to facilitate delivery and circulation around the body. The formation of tightly-packed nuclei can strongly impact the efficacy of the drug—either positively, as it is shielded from damage, or negatively as it keeps the active site away from the area of importance. It is suggested that these cores form potential wells that electrons may be able to tunnel between, so that certain reactions could still happen even if most of the core site was enfolded in the polymer. This would be an interesting case; the potential could be modelled on

the output from the Monte Carlo polymer simulation.

Appendices

A WKB Method

The Wentzel–Kramers–Brillouin (WKB or WKBJ) method is a quasi–classical method that can be applied to find an approximate solution of the one dimensional, time–independent Schrödinger equation

$$-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} \Psi(x) + V(x) \Psi(x) = E \Psi(x) \quad (106)$$

that can also be written

$$\frac{d^2}{dx^2} \Psi(x) = \frac{2m}{\hbar^2} (V(x) - E) \Psi(x) \quad (107)$$

We can write the wavefunction as the exponential of another function

$$\Psi(x) = e^{\Phi(x)} \quad (108)$$

which leads us to re–write Equation (107) as

$$\Phi''(x) + [\Phi'(x)]^2 = \frac{2m}{\hbar^2} (V(x) - E) \quad (109)$$

We now separate the derivative $\phi'(x)$ into real and imaginary parts

$$\Phi'(x) = A(x) + iB(x) \quad (110)$$

where $A(x)$ and $B(x)$ are real functions, meaning that the amplitude of the wavefunction is

$$\exp \left[\int^x A(x') dx' \right] \quad (111)$$

and the phase is

$$\int^x B(x') dx' \quad (112)$$

We can separate the real and imaginary parts of the Schrödinger equation as follows:

$$A'(x) + A(x)^2 - B(x)^2 = \frac{2m}{\hbar^2} (V(x) - E) \quad (113)$$

$$B'(x) + 2A(x)B(x) = 0 \quad (114)$$

At this point, we assume the system is close to classical and expand each function $A(x)$ and $B(x)$ as a power series in \hbar —this is known as the semiclassical approximation.

$$A(x) = \frac{1}{\hbar} \sum_{n=0}^{\infty} \hbar^n A_n(x) \quad (115)$$

$$B(x) = \frac{1}{\hbar} \sum_{n=0}^{\infty} \hbar^n B_n(x) \quad (116)$$

This gives us conditions on $A(x)$ and $B(x)$ to zeroth-order:

$$A_0(x)^2 - B_0(x)^2 = 2m(V(x) - E) \quad (117)$$

$$A_0(x)B_0(x) = 0 \quad (118)$$

Clearly either $A_0(x)$ or $B_0(x)$ must be zero. Classically, the total energy of the particle must be greater than the potential energy, in which case

$$B_0(x) = \pm \sqrt{2m(E - V(x))} \quad (119)$$

In this region the amplitude varies slowly compared to the phase—we see the oscillatory wavefunction of a free particle. However, if the phase varies slowly compared to the amplitude, we get

$$A_0(x) = \pm \sqrt{2m(V(x) - E)} \quad (120)$$

in which case we are seeing the exponential decay of the wavefunction in a region in which the potential energy is greater than the total energy—the quantum tunnelling regime. These are good approximations as long as the change in wavelength is small over a distance of one wavelength. In the region where $E = V(x)$ these solutions become singular, as the term

$$\sqrt{\frac{2m}{\hbar^2} (V(x) - E)} \quad (121)$$

appears in the denominators of the expansion of the wavefunction, so the equations (119) and (120) above only apply away from the boundaries where $E \approx$

$V(x)$. Now the approximate solution near the classical turning points where $E = V(x)$ must be found. If x_1 is a classical turning point, then at x_1 the term

$$\frac{2m}{\hbar^2} (V(x) - E) \quad (122)$$

can be expanded in a power series:

$$\frac{2m}{\hbar^2} (V(x) - E) = U_1(x - x_1) + U_2(x - x_1)^2 + \dots \quad (123)$$

from which, taking the first order only, one finds

$$\frac{d^2}{dx^2} \Psi(x) = U_1(x - x_1) \Psi(x) \quad (124)$$

This now takes the form of the Airy equation [35]

$$\frac{d^2 y}{dx^2} - xy = 0 \quad (125)$$

so we can write the solution using Airy functions:

$$\Psi(x) = C_A \text{Ai} \left(\sqrt[3]{U_1}(x - x_1) \right) + C_B \text{Bi} \left(\sqrt[3]{U_1}(x - x_1) \right) \quad (126)$$

From this, the relationship between C_0 , θ , C_+ and C_- can be found:

$$C_+ = +\frac{1}{2}C_0 \cos \left(\theta - \frac{\pi}{4} \right) \quad (127)$$

$$C_- = -\frac{1}{2}C_0 \sin \left(\theta - \frac{\pi}{4} \right) \quad (128)$$

B Pseudo–Random Numbers

Von Neumann famously uttered the phrase “*Anyone who considers arithmetic methods of producing random digits is, of course, in a state of sin*” in 1951. Numbers generated algorithmically can never be truly random, as every series of operations performed on a computer is deterministic; *true* randomness can only arise from physical processes. However, there exist many methods for producing deterministic sequences of numbers which not only appear random, but pass statistical tests for randomness well enough to be useful. Because these numbers are not actually random, but can be used as such, they are known as pseudo–random numbers, and the algorithms which generate them are known as **Pseudo–Random Number Generators**, or PRNGs.

These methods generally start from a ‘seed’, either a single number or an array of numbers, which must be provided in some way by the programmer—usually by reading the current time from the system clock. If an array of random seeds is required, then the system clock is used to seed a simple random number generator, which is used to generate an array of random numbers which the more complex generator can then make use of. Of course the seed can instead be chosen by the user, which allows for identical sequences if the same seed is used twice. A series of arithmetic operations is then used to go from the seed to the next number in the sequence, and then the next. The operations used can vary from simple additions and multiplications, to bitwise operations such as bit–shifts and exclusive–ors, to modulo arithmetic.

B.1 Pseudo–Random Number Generators

B.1.1 Linear–Congruential Generators

The simplest class of PRNG is the linear–congruential generator (LCG). This takes a seed, multiplies it by a pre–set value a and then adds another value c , modulo a third value, m .

$$x_{n+1} = ax_n + c \pmod{m} \tag{129}$$

These generators suffer from severe problems, particularly with respect to Monte Carlo simulation. For example, all the points in an n -dimensional plot of sets of

n numbers taken in sequence will lie in $n - 1$ dimensional planes. Additionally, they often have rather short periods (in relation to the billions of numbers we often need to generate in Monte Carlo simulation), and the low-order bits can have an even shorter period. This type of generator is patently unsuitable for Monte Carlo simulation and will not be used during this project.

B.1.2 Lagged-Fibonacci Generators

The lagged-Fibonacci generator is intended to be an improvement on the linear-congruential generators discussed above. It takes the form

$$S_n \equiv S_{n-j} \star S_{n-k} \pmod{m} \quad (130)$$

where the \star symbol represents some form of operation (addition, subtraction, multiplication, or bitwise exclusive-or). The values j and k must of course be positive, and are chosen carefully—at least one should be odd, and the ratio between them is important. The last k values must be stored, and the maximum period is

$$(2^k - 1) * 2^{(m-1)} \quad (131)$$

though this depends on the operation used. When using this class of generator, one has to be careful with initialisation of the first k values, as poorly chosen numbers can spoil the random number sequence.

B.1.3 Linear-Feedback Shift Register Generators

Linear-feedback shift register generators are generators in which certain bits of the stored state are used to generate the next bits in the sequence. The R250 generator is an example of this type of generator that uses 250 registers. It generates numbers extremely quickly using only a few operations, and has a period of $2^{249} \approx 9 \times 10^{74}$. It is based upon the simple equation

$$I_k = I_{k-q} \oplus I_{k-p} \quad (132)$$

where $q = 103$, $p = 250$ and \oplus is the *exclusive or* operation, which is used to ‘sum’ bits in parallel. To initialise the PRNG, 250 random numbers must first

be generated to fill up the registers using another PRNG, such as a LCG. The quality of the initialising PRNG is unimportant, since such a small number of bits is needed.

The well-known Mersenne Twister PRNG is a variant of a linear-feedback shift register generator.

B.2 Marsaglia's Diehard Suite

The Diehard Battery of Tests of Randomness is a piece of software containing a suite of statistical tests for PRNGs, developed by George Marsaglia [36]. The user generates a block of random bits—around ten million bits is sufficient—using their chosen PRNG, and the software subjects these to a battery of statistical tests designed to expose any statistical anomalies. The tests performed include geometrical, numerical and Monte Carlo statistical tests.

The result of each test is returned as a p -value in the interval $(0, 1)$. Values very close to zero or one indicate a failed test—although false negatives, in which a statistically random set of data produces a p -value close to zero or one, are possible. Lots of failed tests indicates that the PRNG may be unsuitable for many applications.

The PRNG chosen for this project, KISS (Keep It Simple Stupid), developed by Marsaglia, was subjected to the Diehard suite of tests, and passed to a sufficient standard. The data are not reproduced here for reasons of space.

B.3 Non-Uniform Distributions

Sometimes the uniform distribution produced by most PRNGs is not sufficient for the intended purpose. For the Monte Carlo simulation developed here, numbers are required that follow a normal (Gaussian) distribution. There are several ways to do this.

B.3.1 Inverse Cumulative Distribution Function

The first, and most generalisable, method for producing numbers of a required distribution is to use the inverse of the cumulative distribution function. The

distribution function is given by

$$F(x) = \int_{-\infty}^x f(t)dt \quad (133)$$

If we invert this, giving us

$$F^{-1}(y) \quad (134)$$

then we have a function that will take as its input uniform random numbers in the range $[0, 1]$ and return numbers with the desired distribution. This can be used to generate pseudo-random numbers with a required distribution, but it is not generally the most efficient method for doing this. Furthermore, in general, this technique obviously can only be applied if the CDF is invertible.

In our case, we want numbers with the distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (135)$$

The integral of this function does not have an algebraic expression; therefore we must use a different approach.

B.3.2 Box–Muller Transformation

The Box–Muller transformation [37] performs the same task, by using a trick that requires it to consume and return pairs of numbers. It comes in two forms. The basic form takes pairs of numbers from a uniform distribution, in the range $(0, 1]$, and transforms them to normally distributed numbers. The polar form does the same, but it avoids using sine or cosine functions, as a matter of computational efficiency. It takes a pair of samples from the uniform range $[-1, 1]$, and transforms them to normally distributed numbers. The polar form is intended to be more efficient by replacing the trigonometric functions with rejection sampling and divisions, although on modern hardware the difference in performance may not be so great as it might have been when the algorithm was designed.

Given a pair of uniform random variables U_1 and U_2 , the standard Box-Muller transform returns two independent standard normally distributed numbers X and

Y:

$$\begin{aligned}\Theta &= 2\pi U_1 \\ X &= \sqrt{-2\ln(U_2)}\cos(\Theta) \\ Y &= \sqrt{-2\ln(U_2)}\sin(\Theta)\end{aligned}\tag{136}$$

A more detailed derivation is beyond the scope of this document, but can be found readily online or in textbooks. The argument of the trigonometric functions $\Theta = 2\pi U_1$ is clearly just the conversion of a single uniform random variate to an angle. Assuming length 1, so that every point is on the unit circle $x^2 + y^2 = 1$, the two normally distributed numbers returned are simply the components of a random unit vector multiplied by a factor of $\sqrt{-2\ln(U_2)}$.

The polar form uses a form of rejection sampling; the uniform random variables supplied are taken to be a location vector within a square of side length two. Any points that are outside the unit disc $x^2 + y^2 \leq 1$ are thrown away, until the point is inside the disc. The conversion to points on a unit circle is easily performed by normalising the vector, and then the two normal variates are generated by multiplying by the factor of $\sqrt{-2\ln(U_2)}$, as before. However, roughly 21.5% ($1 - \pi/4$) of the numbers input by the PRNG are thrown away, which means that the PRNG used to feed it must be fast for this version to be an improvement over the basic form. Moreover, rejecting numbers can cause instruction pipeline problems on modern hardware, and calculating the square root required for normalisation may not be faster than calling sine and cosine functions.

B.3.3 Ziggurat Algorithm

The Ziggurat algorithm is intended to be a much quicker way of transforming numbers to a desired distribution. It is another rejection sampling algorithm based upon stacked rectangles of equal area that resemble a ziggurat pyramid, hence the name. It is best used when large quantities of numbers are required—thus, it suits our purposes perfectly. Most of the time (typically 97.5%), the algorithm consists of generating two random numbers (one floating-point and one integer), performing one table lookup, one multiply and one comparison,

making it extremely fast (depending on the speed of the PRNG, of course).

B.4 Quasi-Random Numbers

Another class of random numbers that can be useful in Monte Carlo methods—particularly for numerical quadrature—is quasi-random sequences, also called low-discrepancy sequences. These are numbers that are unpredictable, but that fill space more evenly than true random numbers or pseudo-random numbers. In straightforward Monte Carlo numerical integration, this can lead to faster convergence—*i.e.* to lower error for a given number of samples.

A commonly-used sequence is the Sobol sequence—while the details are not relevant to this work and therefore are not given here, the algorithm is not particularly complex.

C Parallelism

C.1 Parallel Architecture

Since most computing today is moving towards parallel architectures, with even desktop machines containing two or more processor cores, parallelisation of the code is a key factor in obtaining results within a reasonable time-frame. With two main paradigms for parallelisation, supported by two major APIs, it is important to consider carefully the best approach for parallelising CPU-intensive code. The shared memory paradigm, as supported by the OpenMP API, requires a different approach to the distributed memory paradigm, as supported by the MPI API. Since the two APIs are intended for slightly different purposes, the design of any parallel software should take into account the intended target hardware. Parallelisation on desktop machines may require a different approach to that on medium- to large-sized clusters.

Whilst the choice of API to use partially depends on the hardware architecture, most computing clusters in use today are suitable for both paradigms—although the optimal choice may depend on the number of processing cores required. Since the last decade has seen a rise in the availability of multi-core CPUs, many clusters are now constructed from collections of nodes, each containing a number of multi-core processors.

The OpenMP approach is perhaps best-suited to run on either single processors with multiple cores that share a common pool of memory, or single nodes with multiple processors, closely linked by a high-speed memory bus to a single chunk of node memory. MPI is better suited when the code is to be run over several nodes, linked by high- or medium-speed interconnects, each with its own pool of memory. Of course, a third possible approach is to combine both paradigms by finding a way to split the work into separate chunks to be run on different nodes, that can then be further split into sub-tasks to be run on neighbouring CPU cores. The fourth approach could be to run entirely separate instances of the code on multiple machines, and then combine the results once every machine has finished. While such systems exist (for example, the CONDOR system), the approach will be discounted for the rest of this discussion since it really has no bearing on the final design of the code.

Amdahl’s law states that the overall speedup of making a proportion P of a program run in parallel on N processors is

$$\frac{1}{(1 - P) + \frac{P}{N}} \tag{137}$$

This shows that unless the proportion of the code that can be made to run in parallel is close to 100%, the speedup from assigning more processors to the task falls off rapidly with N . The portion of a program that cannot be made to run in parallel includes things like housekeeping (such as memory management), inter-processor communication, and the time spent waiting for data from memory, as well as the portion of each algorithm used that cannot be parallelised.

Fortunately, Monte Carlo algorithms are inherently parallelisable—so much so that they can often be described as “embarrassingly parallel”. In other words, the amount of data produced in a given time scales very nearly linearly with the number of processors working on the task. The reason for this is that Monte Carlo techniques almost exclusively involve data generation and processing, with little or no data passing—and all the data generated by each processor is unique and independent.

A further advantage to parallelisation of statistical calculations is that each process begins with a different random number seed and hence a different random number stream. This ensures that any bias that might remain in the answer produced by only one core due to short convergence times being used will be averaged out over all cores.

C.2 Shared Memory and OpenMP

OpenMP, designed primarily for shared memory systems, is perhaps the easier API. From the programmer’s point of view it is simpler to use, since the software does much of the work of assigning tasks to the multiple cores, but this simplicity comes at the cost of versatility. It allows loops to be spread over multiple processors, so long as the memory pool is shared, so that each core performs a portion of the work. Race conditions and other errors can be partially avoided by declaring a loop to be “atomic”, but this can also limit the speed at which loops operate, as it locks memory accesses to one processor at a time.

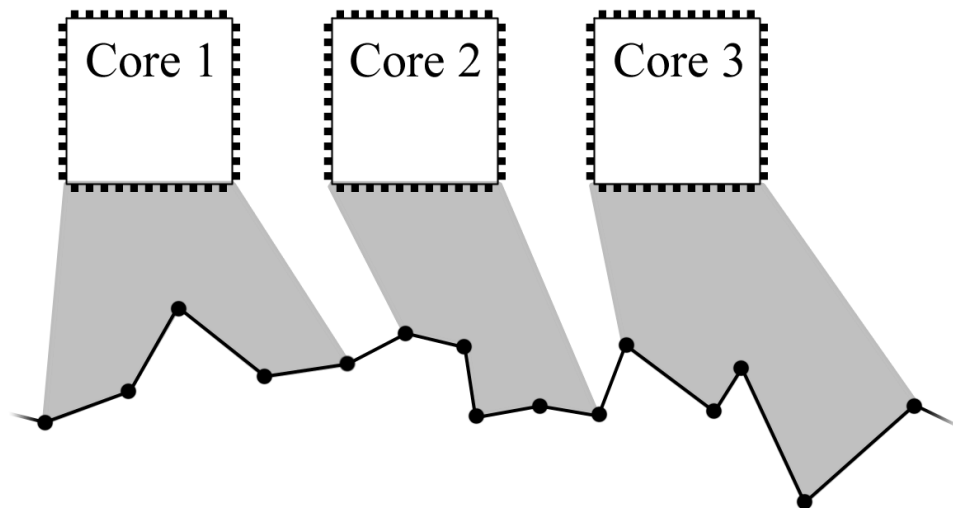


Figure 55: The OpenMP approach to generating chains of beads

Since the Path Integral Monte Carlo code is designed to generate one entire chain of beads using a loop over all beads, one obvious direction to take is to use OpenMP to parallelise the loop. Essentially this would entail moving each section of the chain on a different processor core, as per the diagram in Figure 55. With N processor cores and M beads, approximately M/N beads would be moved on each core—although there would be no way to control how many beads were moved on a particular core, or to make certain that the beads were divided into blocks and moved this way. We have to rely on the OpenMP implementation to ensure that the task division is done in a sensible fashion. The code to do this is easy to write—it simply involves adding some compiler directives directly before the loop in question. This approach is probably the only sensible way to split the work required using OpenMP, since the generation of each chain depends on the state of the previous chain for many of the algorithms used.

There are other, peripheral tasks that could also benefit from OpenMP. During execution of the Monte Carlo simulation, much of the data must be processed in order to perform tasks such as histogram binning, autocorrelation calculations, energy calculations, weighting optimisations and other operations. Any of this data processing that makes use of a for-loop over all beads can be shared between processor cores.

This kind of simple, shared memory task division is only suitable for cores in the same machine. Cores in separate cluster nodes will be unable to make good use of this paradigm due to the amount of data that would need to be shared between processes. Another disadvantage to the OpenMP approach when generating chains of beads is caused by the overlap points—where a bead controlled by one processor core is joined to a bead controlled by another core. Since the movement and final placement of beads depends to a large extent on the positions of neighbouring beads, there exists the possibility of a race condition if two adjacent beads are updated during the same period of real time by different cores. If two large moves are both accepted simultaneously then the result could be that two beads are further apart than they would otherwise be allowed to move. This can be cured by defining memory updates for the loop as “critical”—which locks an area of memory so that only one core can access it at a time—but this can kill the performance, since often a processor will have to wait for others to finish before it can continue with its work. The OpenMP approach also is somewhat unsuitable for multi-slice chain generation methods, especially when chains are generated by a bisection method, since this does not follow the simple technique of moving beads one-by-one from start to end.

C.3 Distributed Memory and MPI

MPI is perhaps a more versatile API than OpenMP, since it allows the programmer to split the tasks up as he or she sees fit; the downside to this is that the programmer must also take care of all movement of data between processor cores, and must think carefully about program design in order to avoid impacting performance or causing errors. If any data need to be passed from one processor core to another, then a message containing the data must be sent explicitly by the code. Other actions such as halting execution in order to wait for updates from other cores can also be performed. The programmer has ultimate control over when and what data is passed between cores, but the price of this is more difficult design decisions, and greater opportunity for unfortunate mistakes.

The beauty of Monte Carlo approaches when it comes to parallelisation is that sets of statistical data can be generated independently and combined afterwards—

essentially each core can run independently without requiring any inter-process communication during the simulation run itself. This is the basis of the MPI approach to parallelising the Path Integral Monte Carlo code. The model used is to assign one core or process to be the master process, which is responsible for loading and saving data whenever required and for performing collation and analysis of data once the chain generation has completed. All processes generate complete chains independently, as in Figure 56. We expect the scaling of speed with number of processors to be excellent—*i.e.* for the speedup to be almost linear with number of processors.

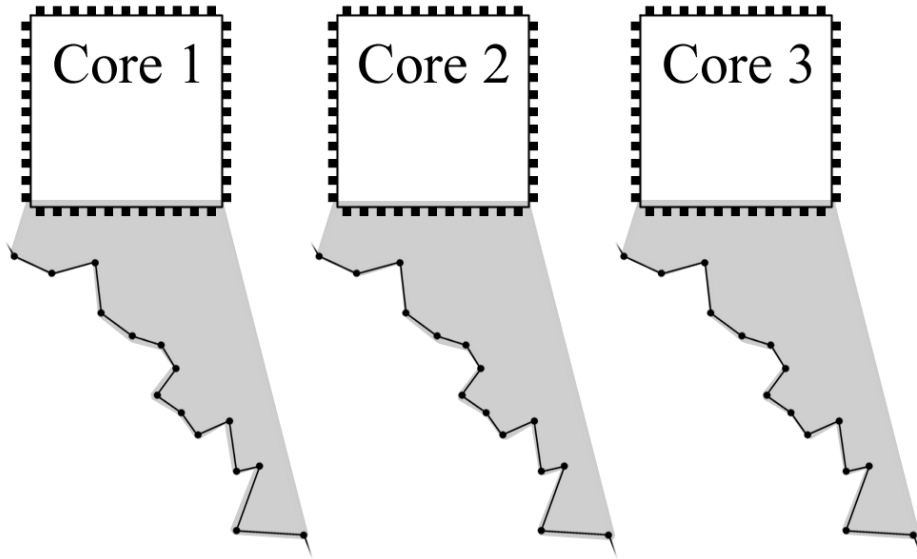


Figure 56: The MPI approach to generating chains of beads

After consideration, the MPI approach was chosen, since it scales better over a wider range of computing clusters, and requires little extra code complexity. Since collating the data only occurs at the very end of the simulation, when all processes have completed, and requires very little data transfer, the interconnect between CPU cores need not be high-bandwidth. The MPI libraries take care of the details of sending messages, regardless of whether the processes are running on the same processor core or on different nodes.

When expanded ensembles is used, the only chains that count towards the total number are the chains generated in the highest energy state only. Since the stepping between states is a random process, it is impossible to predict how long

a given process will take to generate n chains. Process 1 might remain largely in the highest state, and reach n chains fairly quickly, whilst process 2 might remain largely in lower states and take much longer. Data must be collected by the master process at the end of the simulation, so the entire run will take as long as the slowest process takes to complete. By using a short equilibration time in each state—such as with the Lévy bisection method—this problem is reduced, because the stepping between states happens much more rapidly.

D Unit Conversions

Some useful conversion factors for atomic units.

D.1 Energy

$$1 \text{ Hartree} = 219474.6314 \text{ cm}^{-1}$$

$$1 \text{ Hartree} = 27.2117 \text{ eV}$$

$$1 \text{ Hartree} = 627.5095 \text{ kcal/mol}$$

$$1 \text{ Hartree} = 4.3597482 \times 10^{-18} \text{ J}$$

D.2 Distance

$$1 \text{ Bohr} = 0.5291772083 \text{ \AA}$$

$$1 \text{ Bohr} = 0.05291772083 \text{ nm} = 52.91772083 \text{ pm}$$

D.3 Time

$$1 \text{ atomic unit of time} = 2.418885 \times 10^{-17} \text{ s}$$

D.4 Electric Field

$$1 \text{ atomic unit of electric field} = 5.14 \times 10^9 \text{ V/cm}$$

E Usage Guide to Code

This section serves as a brief guide to anyone using the PIMC code. With the use of the makefile provided, the code compiles to a pair of executable files. *pimcmulti.exe* is the MPI version, and *pimcser.mpi* is the single-core version.

E.1 Parameters File

The parameters file contains mainly numerical entries that are too numerous to include as command line parameters. These parameters must always be in the order given here, and should consist of the name of the parameter followed by a space and then the value of the parameter. The code will read the file *parameters.dat* by default, though this can be changed.

stepsize *double* The step size for the basic single-slice method.

tempK *double* The temperature of the simulation, in Kelvin.

chains *integer* The number of chains to generate.

beads The number of beads to use in the chain, including the fixed end beads.

firstbead *double* The x -position of the first (fixed) bead.

lastbead *double* The x -position of the last (fixed) bead.

barrier *double* The barrier height to use when using polynomial or function-based external potentials.

histogram_x *double* The length of the histogram in x (centred on zero).

histogram_y *double* The width of the histogram in y (centred on zero).

decorrelation *integer* The maximum time for the chain to decorrelate. This parameter sets how many chains are generated in between each expanded ensembles state switch.

section *integer* The number of beads to select to regrow when using the section regrowth method. Must be $2^n + 1$ and less than the overall number of beads.

write_freq *integer* This is the frequency (per number of chains) at which the code writes its data out to disc, and also the frequency at which the code performs some of its data analysis.

equil *integer* The number of chains to equilibrate over before the main code loop starts. Now unused, leave at 0.

autocorr *integer* The number of chains to generate the autocorrelation function over.

autointerval *integer* The space between each consecutive window of the autocorrelation function.

deltax *double* For numerical calculation of the potential gradient, this is the x -interval to use.

deltay *double* For numerical calculation of the potential gradient, this is the y -interval to use

deltaz *double* For numerical calculation of the potential gradient, this is the z -interval to use

delta_t *double* The time step to use for the force-bias and fourth-order methods.

lambda *double* The value of lambda. Should be left at 1.0.

file *string* The file name root for saving data files. For example, the root *single* might produce files *histo-single.csv*, *conv-single.csv* and *chain-single.csv*

dir *string* The directory to save data files to.

seed *integer* The random number seed to use. If this is left as zero, the PRNG will be initialised from the system time.

lowparameter *double* When using the method of expanded ensembles, this is the lowest value of the scaling parameter λ .

highparameter *double* When using the method of expanded ensembles, this is the highest value of the scaling parameter λ .

steps *integer* When using the method of expanded ensembles, this is the number of steps to use from the lowest to the highest energies.

focusstate *integer* When using the method of expanded ensembles, this is the state to begin at, and the state at which data output takes place.

potential *string* Selects the potential model to use. Choices are: *zero*, *argon*, *nanotube*, *gaussian*, *ridge*, *circlewell*, *squarewell*, *squareblock*, *doublesquarewell1d*, *doublesquarewell*, *doublewell* and *box* or *boxes*. The *argon*, *nanotube* and *box* or *boxes* options load the potential model from the appropriate file, *nuclei.dat*, *nanotubeslayout.dat* or *box.dat* (by default).

E.2 Command Line Options

The code uses command line arguments to choose basic options such as the algorithm to use and whether to write a histogram or an autocorrelation function. The command line arguments can be as follows:

Autocorrelation Use the argument *auto* followed by either *pot* or *avgpot* to take the autocorrelation of the average potential over all beads, or *bead n* to take the autocorrelation of the position of bead *n*, where *n* should be a number greater than zero and less than the number of beads. The default option is to take the autocorrelation of the average potential.

Autocorrelation normalisation Argument *anorm* to set this to *yes*. Whether to automatically normalise the autocorrelation function when the data is written to disc.

No console The argument *noconsole* will tell the code not to write any information to the console, but instead write to an output file.

Histogram Use *histo* to tell the code to save a histogram of bead positions.

Convergence Use *conv* to tell the code to calculate and save the convergence functions out to disc.

Lévy The argument *levy* tells the program to use the multi-slice Lévy bisection method.

Bisection The argument *bisec* tells the program to use the multi-slice multi-level Lévy bisection method.

Section regrowth The argument *section* tells the program to use the section regrowth multi-slice Lévy bisection method.

Force-bias The argument *bias* tells the program to use the force-bias method.

Fourth-order propagator The argument *fo* or *FO* tells the program to use the fourth-order propagator (currently not working reliably).

Expanded ensembles The argument *exe* or *stateswitch* tells the program to use the method of expanded ensembles. This will cause extra data files to be written out to the directory specified in the parameters file.

Reload The argument *load* followed by a directory name and root file name tells the program to reload data from the directory specified and continue from the last completed chunk of data.

Parameters file The argument *paramfile* or *paramsfile* or *pfile* followed by a complete file name tells the program to load that file as a parameters file, rather than the default *parameters.dat*.

Box potential file The argument *boxfile* or *boxesfile* or *bfile* followed by a complete file name tells the program to load that file as a box potential file, rather than the default *box.dat*. See Section E.3.

Write potential The argument *wripotential* or *wriepot* or *wpot* tells the code to write out a plot of the potential over the area set for the histogram, with the same dimension as the histogram.

New chains The argument *nchains* or *setchains* tells the code to override the total number of chains specified in the *parameters.dat* file. This option is used together with *load* when the code is reloading a previous calculation to restart and generate more data.

Count beads in box The argument *bib* or *beadsinbox* or *inbox* followed by an integer number *n* tells the code to keep track of the average of how many beads per full chain are within box *n*—this only makes sense when the box potential is in use.

E.3 External Files

The program looks for three external files—besides the *parameters.dat* file—that must be present in the same directory as the executable. These files are:

nuclei.dat is a list of 44 three-dimensional coordinates that specify the positions of the argon nuclei. This file should not be changed.

nanotubeslayout.dat is a list of nanotubes to use. The first line contains a single integer specifying the number of nanotubes in the file. Each nanotube is then read from three consecutive lines: the coordinates of the centre of one end of the tube; the coordinates of the centre of the other end of the tube; the radius of the tube.

box.dat is a list of box-shaped potential wells to use. The first line contains a single integer specifying the number of boxes in the file. Each box is then read from three consecutive lines: the coordinates of the centre of the box; the side lengths of the box; the potential inside the box.

References

- [1] J. M. B. Jr., “Electron relay in proteins,” *Science*, vol. 320, p. 1730, 2008.
- [2] P. L. Dutton, A. W. Munro, N. S. Scrutton, and M. J. Sutcliffe, “Introduction. Quantum catalysis in enzymes: Beyond the transition state theory paradigm,” *Phil. Trans. R. Soc. B*, vol. 361, pp. 1293–1294, 2006.
- [3] C. C. Moser, T. A. Farid, S. E. Chobot, and P. L. Dutton, “Electron tunnelling chains of mitochondria,” *Biochimica et Biophysica Acta*, vol. 1757, no. 9-10, pp. 1096–1109, 2006.
- [4] J. Lin, I. A. Balabin, and D. N. Beratan, “The nature of aqueous tunneling pathways between electron-transfer proteins,” *Science*, vol. 310, no. 5752, pp. 1311–1313, 2005.
- [5] L. Brillouin, “La mécanique ondulatoire de Schrödinger: une méthode générale de résolution par approximations successives [Schrodinger’s wave mechanics: a general method of resolution by successive approximations],” *Comptes Rendus de l’Academie des Sciences*, vol. 183, pp. 24–26, 1926.
- [6] H. A. Kramers, “Wellenmechanik und halbzahlige quantisierung [Wave mechanics and half-integer quantisation],” *Zeitschrift für Physik*, vol. 39, pp. 828–840, 1926.
- [7] G. Wentzel, “Eine verallgemeinerung der quantenbedingungen für die zwecke der wellenmechanik [A generalization of the quantization rules for the purposes of wave mechanics],” *Zeitschrift für Physik*, vol. 38, pp. 518–529, 1926.
- [8] H. Jeffries, “On certain approximate solutions of linear differential equations of the second order,” *Proceedings of the London Mathematical Society*, vol. 23, pp. 428–436, 1924.
- [9] A. Mosyak and A. Nitzan, “Numerical simulations of electron tunnelling in water,” *J. Chem. Phys.*, vol. 104, no. 4, pp. 1549–1559, 1996.
- [10] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, “Molecular-Dynamics with Coupling to an External Bath,” *J. Chem. Phys.*, vol. 81, pp. 3684–3690, 1984.

- [11] R. W. Hall and B. J. Berne, “Nonergodicity in path integral molecular dynamics,” *J. Chem. Phys.*, vol. 81, p. 3641, 1984.
- [12] G. J. Martyna, M. L. Klein, and M. Tuckerman, “Nosé-Hoover chains: The canonical ensemble via continuous dynamics,” *J. Chem. Phys.*, vol. 97, p. 2635, 1992.
- [13] R. P. Feynman, *The Principle of Least Action in Quantum Mechanics*. PhD thesis, Princeton University, 1942.
- [14] R. P. Feynman, “Space-time approach to non-relativistic quantum mechanics,” *Rev. Mod. Phys.*, vol. 20, p. 367, 1948.
- [15] P. A. M. Dirac, “The Lagrangian in quantum mechanics,” *Phys. Zeitschr. Sowjetunion*, vol. 3, p. 64, 1933.
- [16] H. F. Trotter, “On the product of semi-groups of operators,” *Proceedings of the American Mathematical Society*, vol. 10, pp. 545–551, 1959.
- [17] D. M. Ceperley, “Path integrals in the theory of condensed helium,” *Rev. Mod. Phys.*, vol. 67, no. 2, pp. 279–355, 1995.
- [18] C. Pangali, M. Rao, and B. J. Berne, “On a novel Monte Carlo scheme for simulating water and aqueous solutions,” *Chem. Phys. Lett.*, vol. 55, no. 3, p. 413, 1978.
- [19] A. Scemama, T. Lelièvre, G. Stoltz, E. Cancès, and M. Caffarel, “An efficient sampling algorithm for variational Monte Carlo,” *J. Chem. Phys.*, vol. 125, no. 11, p. 114105, 2006.
- [20] M. R. Shirts, E. Bair, G. Hooker, and V. S. Pande, “Equilibrium free energies from nonequilibrium measurements using maximum-likelihood methods,” *Phys. Rev. Lett.*, vol. 91, no. 14, p. 140601, 2003.
- [21] M. R. Shirts and V. S. Pande, “Comparison of efficiency and bias of free energies computed by exponential averaging, the Bennett acceptance ratio, and thermodynamic integration,” *J. Chem. Phys.*, vol. 122, no. 14, p. 144107, 2005.

- [22] A. P. Lyubartsev, A. A. Martinovski, S. V. Shevkunov, and P. N. Vorontsov-Velyaminov, “New approach to Monte Carlo calculation of the free energy: Method of expanded ensembles,” *J. Chem. Phys.*, vol. 96, no. 3, p. 1776, 1992.
- [23] F. A. Escobedo and F. J. Martinez-Veracoechea, “Optimized expanded ensembles for simulations involving molecular insertions and deletions. i. closed systems,” *J. Chem. Phys.*, vol. 127, no. 17, p. 174103, 2007.
- [24] I. Benjamin and A. Nitzan, “Path-integral computations of tunneling processes,” *J. Chem. Phys.*, vol. 123, no. 10, p. 104103, 2005.
- [25] A. Kuki and P. Wolynes, “Electron Tunnelling Paths in Proteins,” *Science*, vol. 236, p. 1647, 1987.
- [26] B. P. Uberuaga, M. Anghel, and A. F. Voter, “Synchronization of trajectories in canonical molecular-dynamics simulations: Observation, explanation, and exploitation,” *J. Chem. Phys.*, vol. 120, no. 14, p. 6363, 2004.
- [27] X. Li, J. Q. Broughton, and P. B. Allen, “Electroninert gas pseudopotentials for use in path integral simulations,” *J. Chem. Phys.*, vol. 85, no. 6, p. 3444, 1986.
- [28] J.-M. Lopez-Castillo, Y. Frongillo, B. Plenkiewicz, and J.-P. Jay-Gerin, “Path–integral molecular–dynamics calculation of the conduction–band energy minimum v_0 of excess electrons in fluid argon,” *J. Chem. Phys.*, vol. 96, no. 12, p. 9092, 1992.
- [29] J. T. Wescott, P. Kung, and A. Maiti, “Conductivity of carbon nanotube polymer composites,” *Applied Physics Letters*, vol. 90, p. 033116, 2007.
- [30] F. Deng and Q.-S. Zheng, “An analytical model of effective electrical conductivity of carbon nanotube composites,” *Applied Physics Letters*, vol. 92, p. 071902, 2008.
- [31] S. Trebst, D. A. Huse, and M. Troyer, “Optimizing the ensemble for equilibration in broad-histogram Monte Carlo simulations,” *Phys. Rev. E*, vol. 70, no. 4, p. 046701, 2004.

- [32] F. J. Martinez-Veracoechea and F. A. Escobedo, “Optimized expanded ensembles for simulations involving molecular insertions and deletions. i. closed systems,” *J. Chem. Phys.*, vol. 127, no. 17, p. 174103, 2007.
- [33] M. Mella, G. Morosi, and D. Bressanini, “Time step bias improvement in diffusion Monte Carlo simulations,” *Phys. Rev. E*, vol. 61, no. 2, pp. 2050–2057, 2000.
- [34] M. Mella and L. Izzo, “Structural properties of hydrophilic polymeric chains bearing covalently-linked hydrophobic substituents: Exploring the effects of chain length, fractional loading and hydrophobic interaction strength with coarse grained potentials and Monte Carlo simulations,” *Polymer*, vol. 51, pp. 3582–3589, 2010.
- [35] G. B. Airy, “On the intensity of light in the neighbourhood of a caustic,” *Transactions of the Cambridge Philosophical Society*, vol. 6, pp. 379–402, 1838.
- [36] G. Marsaglia, “Diehard battery of tests of randomness.” World Wide Web electronic publication, 2003.
- [37] G. E. P. Box and M. E. Muller, “A note on the generation of random normal deviates,” *Ann. Math. Stat.*, vol. 29, no. 2, pp. 610–611, 1958.