Computational Approach to Bohm's Quantum Mechanics

Computational Approach to Bohm's Quantum Mechanics

by

PAULO ALEXANDRE MACHADO

B.Sc. (Oporto), M.Sc. (McMaster)

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2007) McMaster University

(Department of Physics and Astronomy) Hamilton, Ontario

TITLE: Computational Approach to Bohm's Quantum Mechanics

AUTHOR: Paulo Alexandre Machado

SUPERVISOR: Prof. D. W. L. Sprung

NUMBER OF PAGES: xiv, 146

Abstract

Bohmian mechanics is an alternative formulation of quantum mechanics that incorporates the familiar and intuitive picture of particles moving along trajectories and yet predicts the same results as the more widely accepted Copenhagen interpretation.

In recent years there has been renewed interest in this Bohmian view, in part for the novel approach that it suggests to certain problems, such as decay processes, both from a theoretical and computational stand point. In this thesis we focus on using the concepts introduced by the Bohmian framework as a practical computational tool.

I evaluate a number of implementations of the Bohmian method, get a sense of their strengths and weaknesses and attempt to overcome some stability issues that arise. For problems in one-dimension (1D), accurate solutions of the time-dependent Schrödinger equation produce a wave function from which Bohmian trajectories can be computed by integrating along flux lines. For direct integration of the quantum Hamilton-Jacobi equations, the main problems that arise are related to evauating the quantum potential (QP), especially in regions of low probability density. Sufficient accuracy is required to avoid unphysical trajectory crossings. A number of interpolation schemes were investigated, and smoothed splines with special treatment of edge effects gave the best results.

For problems in 2D the alternating direction implicit (ADI) method was employed to produce the wave function. Ways of dealing with unphysical reflections from the boundaries of a finite size domain were studied.

The use of cellular automata, especially the lattice-Boltzmann method (LBM) were also considered. Here Bohm trajectories would be propagated by following a small set of rules. The main problem identified is that, unless a scheme can be found in which the quantum potential is self-generating from an equation of continuity, the overhead of computing the QP at each time step, is prohibitive.

Acknowledgements

I would like to express my gratitude to my supervisor D. W. L. Sprung for the thoughtful guidance, kindness, patience and constructive criticism that I enjoyed over the years under his supervision.

I also thank Y. Nogami and D. Pelinovsky for their helpful suggestions and interest in the work as well as the encouragement given by R. Dumont.

This work would not be possible without the positive learning environment and financial support provided by McMaster University.

I would also like to thank Hannah Sprung, Rajat and Manju Bhaduri as well as Natalia and Tanya Wolf, Walter Phillips and Errol Rennie for keeping me occupied and making sure I didn't starve throughout the year.

Finally I would like to thank Jen and my Mom who helped me through the more challenging moments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "Things on a very small scale behave like nothing that you have any direct experience about. They do not behave like waves, they do not behave like particles, they do not behave like clouds, or billiard balls, or weights on springs, or like anything that you have ever seen."
>
> Richard Feynman

Almost since the inception of quantum mechanics, alternative interpretations have existed. One of these, developed by de Broglie and later systematized by Bohm, makes use of the familiar and intuitive picture of particles moving along trajectories to describe the unfamiliar realm of quantum mechanics.

Paradoxically Bohmian mechanics (as it came to be known), was met with little enthusiasm, even outright hostility in the early days of quantum mechanics and as a result proceeded to lay dormant for a few decades.

Recently there has been renewed interest in this interpretation, in part for the novel approach that it suggests to certain problems, such as decay processes, both from a theoretical and computational stand point.

1.1 Motivation and Objectives

In this thesis I will further explore the Bohmian perspective as an alternative framework for numerical calculations on a given quantum system.

The motivation for this study is the promise of better CPU scaling with the dimensionality of the problem at hand. Common grid based methods often become impractical in high dimensionality problems because of their exponential scaling.

It is said that there is no such thing as a free lunch on Wall Street and the same can be said of Computing Street. The performance increase offered by Bohmian methods, due to the sparse grids used, normally has a big trade off in the form of inconsistent results wherever nodes or interference phenomena develop.

Since the long time evolution of a quantum system gives rise precisely to such interference phenomena, Bohmian methods are usually riddled with stability issues.

I will investigate the causes of these stability issues, and how to minimize them. I will study how different approaches and implementations of the particle method fare, in terms of speed and long time stability, the latter usually enhanced at the cost of additional computational complexity. The aim is to improve where possible on these methods and recognize their strengths and weaknesses.

1.2 Quantum Mechanics and its Interpretations

That quantum theory "works" has long been settled by an impressive succession of successful experimental predictions, and explanations, so its validity is no longer in question. However the thorny issue of how the mathematical formulation of quantum theory is to be interpreted is quite another matter...

As a clear sign of its inherent richness, the quantum theory accommodates very

many, sometimes disconcerting, interpretations. To paraphrase Richard Feynman in one of his series of Auckland lectures on QED:

> "If you think you understand quantum mechanics, you don't understand quantum mechanics!"

We will not exhaustively describe all proposed formulations/interpretations of quantum mechanics, which at last count numbered at least nine, and rapidly increasing. Instead we will refer you to the excellent introductory paper "Nine formulations of quantum mechanics" [1] for details .

The list includes some of the more debatable and trendy ones such as "Many Worlds" and "Transactional" interpretations of quantum mechanics. For curiosity's sake though, I will enumerate from the aforementioned paper [1] the nine formulations and two extra interpretations of quantum mechanics:

- Matrix Formulation (Heisenberg)
- Wavefunction Formulation (Schrödinger)
- Path Integral Formulation (Feynman)
- Phase Space Formulation (Wigner)
- Density Matrix Formulation
- Second Quantization Formulation
- Hamilton-Jacobi Formulation
- Variational Formulation
- Many-Worlds interpretation (Everett)

- Transactional interpretation (Cramer)
- Last but not least: The Pilot Wave Formulation (de Broglie-Bohm)

To some, the Pilot Wave or Bohmian theory is one of those disconcerting formulations, regressing as it were to classical concepts such as the moving point particle with a definite momentum and position. One must not be prejudiced towards the theory because of this, but use it as a tool, and like any tool use it where it works best.

The two formulations with which we will be concerned are the usual Schrödinger representation or wavefunction formulation as it is the basis for common grid based computational methods; and the Bohmian formulation as an alternative computational route.

1.2.1 Schrödinger Picture: Wave Mechanics

Erwin Schrödinger had hoped that this formulation would cast quantum mechanics in a "congenial" and "intuitive" form but was distressed when he found that his wave functions were set in configuration space and not in ordinary three-dimensional space ([1]).

In the Schrödinger representation of quantum mechanics we deal with a complex wavefunction $\Psi(x,t) =< x|\Psi(t) >$ where the ket is in Hilbert space representing the state of the system.

The time evolution of the wavefunction is obtained by solving the Time-Dependent Schrödinger Equation (TDSE):

$$i\hbar\frac{\partial\Psi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\Psi + V\Psi \tag{1.1}$$

Observations of physical quantities (observables) like position, momentum, angular momentum, etc are associated with the "collapse" of the wavefunction onto eigenstates of the corresponding observable, i.e. a quantum measurement acts as a filter. Those systems that pass through the filter corresponding to some value of an observable emerge in the associated eigenstate. The apparatus is described by a sum of projectors onto the eigenstates of the operator associated with the particular physical quantity. The probability amplitude for a particular result is the scalar product of the initial wave function with the particular eigenket. In the Schrödinger representation, the scalar product is carried out by integration over all space coordinates, and summation over spin indices.

1.2.2 Bohmian Formulation

The Bohmian formulation of quantum mechanics is also known as the Pilot-Wave theory or causal interpretation of quantum mechanics.

In the practical implementation of this method use is made of an ensemble of sampling points representative of the state of the system at time zero. Usually these points are chosen to be more dense in regions where the particle has a high occupation probability in the initial state.

In Bohmian mechanics the analogy to classical mechanics is so compelling that we will find ourselves sometimes referring to these sampling points very loosely as "particles" whose "trajectories" develop in time. It should be understood that we are not dealing with particles in the usual sense of the word , but instead when we refer to particles in a Bohmian context we are actually referring to the sampling points carrying with them some probability volume, which taken as a whole may represent the evolution of some "real" particle (such as an electron).

It is only in the initial state when we assign these sampling points or "particles" that probabilistic arguments are invoked and the rest of the theory is completely deterministic thus earning the denomination "causal interpretation".

Following the initial setup, these point particles evolve in time according to forces derived from the sum of the classical and a newly introduced quantum potential. The quantum potential in turn depends on the particular distribution of the point particles at a given time.

It is this circular self consistency of the theory that is responsible for all the complex quantum behaviour it is able to describe (and also for some computing headaches we may add).

1.3 Historical Perspective

The historical foundations of Bohmian mechanics can be traced back to the hydrodynamical formulation of quantum mechanics due to de Broglie[2] [3] and Madelung [4] in the early days of quantum theory (i.e. late 1920's to early 1930's).

Some basic aspects of pilot-wave theory were already anticipated in de Broglie's thesis of 1924. His talk at the 5'th Solvay conference in 1927 included an almost complete exposition of the theory (see [5]).

Probably because of immediate criticism of the theory by Pauli and others (as early as the 1927 Solvay conference [5]) the pilot-wave theory of quantum mechanics was promptly cast into oblivion until David Bohm independently developed and presented it in the early 50's (see [6]).

David Bohm (from whom Bohmian mechanics takes the name) was incidentally a man of many and varied contributions and not only to physics. We pay tribute to his memory by quoting Dürr, Goldstein and Zanghi [7] celebrating the life of the then

recently deceased David Bohm:

> "... We focus here on Bohm's contributions to physics. However, he also made profound contributions to many other disciplines: to the philosophy of science and the philosophy of mind, to ethics and moral philosophy. And Bohm labored long for peace and disarmament, for dialog and mutual understanding."
>
> "Richard Feynman once declared Bohm the smartest man he had ever met. He was certainly a man of extraordinary commitment to principle, both scientific - as witnessed by his often lonely pursuit of scientific truth, without regard for prevailing fashion - and moral - as witnessed by his refusal in 1951 to testify against colleagues before the House of Un-American Activities Committee. This act led to his indictment for contempt of Congress and his banishment from Princeton and, indeed from all of American academia. "

Bohm's seminal work was followed by another period of relative disinterest in the theory until the 1970's, when John Bell became one of the most celebrated physicists to back the Bohmian view. In fact the theory is even closely related to one of his most celebrated discoveries "Bell's Inequality". According to Bell it was the non-locality present in the Bohmian theory that inspired him to develop that result.

Included in the same paper by Dürr, Goldstein and Zanghi [7] is a private communication by Bell to the Philosopher Renée Webber where the ironic connection between David Bohm's Theory and Bell's theorem (which is widely cited in support of quantum orthodoxy) is exposed in the words of J. Bell:

> "When I first realized that [Bohm's theory is nonlocal], I asked: "Is that inevitable or could somebody smarter than Bohm have done it differently

> and avoided this nonlocality?" That is the problem that [Bell's] Theorem is addressed to. The theorem says: "No! Even if you are smarter than Bohm, you will not get rid of nonlocality", that any sharp mathematical formulation of what is going on will have that nonlocality..."
>
> "In my opinion the picture which Bohm proposed then completely disposes of all the arguments that you will find among the great founding fathers of the subject- that in some way, quantum mechanics was a new departure of human thought which necessitated the introduction of the observer, which necessitated speculations about the role of consciousness, and so on."
>
> "All those are simply refuted by Bohm's 1952 theory... So I think that it is somewhat scandalous that this theory is so largely ignored in textbooks and is simply ignored by most physicists. They don't know about it."

The historical record on the computational front, where Bohmian mechanics is used in a more practical way as a basis for direct calculation of trajectories, is more recent. Early works can be traced to the 1970's in the works of Weiner et al. [8]. Then after another long hiatus, the computational side of Bohmian mechanics was picked up again in the nineties in works by Hua Wu and Sprung[9], Wyatt and Lopreore [10], and by Sales Mayor, Askar and Rabitz [11] which set in motion another wave of interest in the theory this time from a numerical perspective. A series of works then followed over the following years up to the present day by many authors, including those aforementioned and, amongst others, E. Bittner , B. Poirier , S. Garashchuk, V. Rassolov, S. Goldstein and R.E. Wyatt.

1.4 Objections to Bohmian Mechanics

Bohmian mechanics has never been a short of critics. At the 1927 Solvay conference Pauli was the first of a long list of eminent physicists to criticize de Broglie's first proposal of a Pilot-Wave theory. In 1952, Pauli again leveled criticism on the now more developed de Broglie-Bohm theory in his contribution to de Broglie's 60th birthday volume deeming it "artificial metaphysics" [12, 13] (Pauli's objections revolved around the break in the correspondence between classical and Bohmian mechanics pertaining to the symmetric treatment of canonically conjugate variables such as position and momentum. For other historical criticism a good source is Myrvold's paper [14]). Concerning other less damaging accusations, Peter Holland's book ([15]) does a good job of summarizing and then shortly addressing them:

- Cannot prove trajectories are real - Cannot prove empirically the completeness postulate either.
- Predicts nothing new - Does permit more detailed predictions pertaining to individual processes.
- Regression to classical physics - It does depend on the "state" of the whole system, represented by the guiding wave (a very non-classical concept)
- Non-locality - Non-locality seems to be a small price to pay if the alternative is to forego any account of objective processes.
- More complicated than quantum mechanics - It is just a reformulation of quantum mechanics
- Counter intuitive - Quantum phenomena require quantum intuition

- No reciprocal action of the particle on the wave - in fact the wave depends on the positions of the representative particle's trajectories.

Chapter 2

Bohmian Mechanics

For a more exhaustive and complete theoretical treatment of Bohmian mechanics the definitive reference is Peter Holland's book " The Quantum Theory of motion" [15]. In this chapter we will limit ourselves to aspects of the theory that are relevant to calculations in this thesis.

2.1 Hamilton-Jacobi Theory

We take the classical Hamilton-Jacobi equation as our starting point since Hamilton Jacobi theory in the shape of the Quantum Hamilton-Jacobi equation (QHJE) figures so prominently in Bohmian mechanics.

$$\frac{\partial S}{\partial t} = -\frac{(\nabla S)^2}{2m} - V \tag{2.1}$$

In fact it can be argued (see [16]) that the seed of wave mechanics and Bohmian mechanics is contained in the relationship between classical Hamilton-Jacobi theory and Geometrical Optics.

Goldstein argued along the following line:

If we consider that the surfaces of constant S (Hamilton's principal function) correspond to wavefronts propagating in configuration space we can assign a wave velocity to them $u = \frac{\nabla S}{m}$.

Taking the scalar wave equation of optics ,

$$\nabla^2\phi - \frac{n^2}{c^2}\frac{d^2\phi}{dt^2} = 0 \tag{2.2}$$

which for slowly changing index of refraction n, has a solution of the form $\phi = A(r)\exp(ik_0(L(r) - ct))$ where A is the amplitude of the wavefunction and L is called the optical path length or eikonal of the wave. Substituting ϕ into eq. 2.2 results, in the short wavelength approximation, in the eikonal equation of geometrical optics $(\nabla L)^2 = n^2$. The surfaces of constant L are surfaces of constant optical phase that define wave fronts just as in the case of S from the classical Hamilton Jacobi equation.

Classical mechanics corresponds then to the geometrical optics limit of wave motion. One might ask then what would happen to classical mechanics if we do not take the short wavelength approximation. The answer would be the same as in the case of wave motion: you would get the full wave equation. In the context of mechanics the analog of the full wave equation is of course the celebrated the Schrödinger equation.

So could Hamilton or his peers have stumbled upon wave mechanics nearly 100 years before its actual discovery, with all the associated tantalizing scientific and technological advances that such an early leap in human knowledge would carry? Probably not... there was no reason to suspect that h in $\lambda = \frac{h}{p}$ was anything other than zero, and classical mechanics was deemed to be rigorously true. Experimental evidence suggesting otherwise would not be available until the beginning of the XXth century in experiments such as those of Davisson and Germer, and others.

2.2 Derivation from the Schrödinger Equation

The central element in the Bohmian picture is the concept of particle with an associated (definite) position and momentum. One can obtain the Bohmian equations of motion by writing the wavefunction in polar form: $\Psi(x,t) = R(x,t)e^{iS(x,t)}$, where R and S are respectively the amplitude and phase.

Inserting the polar form of $\Psi(x,t)$ into the Schrödinger equation:

$$i\hbar\frac{\partial\Psi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\Psi + V\Psi$$

After some manipulations leads to two equations arising from the real and imaginary parts respectively ([15])

$$\begin{cases} \frac{\partial R^2}{\partial t} + \nabla(R^2\frac{\nabla S}{m}) = 0 \\ \frac{\partial S}{\partial t} = -\frac{(\nabla S)^2}{2m} - V + \frac{\hbar^2}{2m}\frac{\nabla^2 R}{R} \end{cases} \tag{2.3}$$

The first equation in 2.3 can be interpreted as a continuity equation relative to the probability flow $R^2\frac{\nabla S}{m}$.

The second equation is an analog of the Hamilton-Jacobi equation with an extra term $Q = -\frac{\hbar^2}{2m}\frac{\nabla^2 R}{R}$, the quantum potential. This extra term depends on the whole of the wavefunction, thus the name pilot wave theory, as Bohmian mechanics is sometimes referred to (the wave guides the particle).

This is the usual way of interpreting that extra term as an non-classical potential. Alternatively (and equivalently) we can consider the $-\frac{\hbar^2}{2m}\frac{\nabla^2 R}{R}$ as being another contribution to the kinetic energy $\frac{(\nabla S)^2}{2m}$, a form of a shape induced internal energy akin to the role that the internal stress tensor plays in a classical fluid dynamics context.

Quickly glancing at the quantum term, already we can make a few preliminary

observations. We note that as $\hbar$ tends to zero the quantum potential disappears and we should expect to get classical behaviour from the system. Because R appears both in the denominator and in the numerator (as its gradient) Q will not depend on the magnitude of R but on its curvature. When evaluating it numerically, however, we can anticipate some difficulties in places where R is zero or very small.

2.2.1 Eulerian/Lagrangian Frames

To further clarify on the equations of motion we can, as is done in Hamilton Jacobi theory of classical mechanics, associate a velocity field given by $v = \frac{\nabla S}{m}$, parallel to the gradient of S which therefore will be perpendicular to a wavefront (a level surface) of S.

At this point we need to discuss the concepts of Lagrangian and Eulerian frames of reference. In classical fluid dynamics if we take a "moving with the fluid" perspective, then the change of a physical quantity at each fluid "particle" is given by its material derivative (also called substantive derivative), its intrinsic change plus its variation in space: $\frac{d}{dt} = \frac{\partial}{\partial t} + \nabla v$. This is the so called Lagrangian frame where the grid evolves in time with the moving fluid particles as opposed to the Eulerian view where the grid is fixed in space.

Taking the gradient of the Quantum Hamilton Jacobi equation (QHJE) eq 2.3, with $\rho = R^2$ and working in the Lagrangian frame of reference, we obtain a more familiar set of equations([15]):

$$\begin{cases} \frac{d}{dt}\rho + \rho\nabla v = 0 \\ \frac{dv}{dt} = -\frac{1}{m}\nabla(V+Q) \\ Q = -\frac{\hbar^2}{2m}\frac{\nabla^2 R}{R} \end{cases} \tag{2.4}$$

Here we can still recognize the first as the continuity equation, but now the second equation reveals itself as just a statement of Newton's second law $F = ma$ with the now familiar extra quantum force term.

2.3 The Quantum Potential

That last term in eq. 2.4 is the quantum potential, the essence of Bohmian mechanics. It is Q which is solely responsible for all the quantum behaviour of the system and it incorporates both properties of the particle such as momentum and position, and properties of the whole system such as potential and observing apparatus.

$$Q = -\frac{\hbar^2}{2m}\frac{\nabla^2 R}{R} \tag{2.5}$$

It is quite peculiar in that it can vary rapidly in places where V, the classical potential, is almost constant. Even more strange is that since we have a normalizing factor in the denominator, it does not depend on the magnitude of the density ρ itself, but rather on its curvature (or more appropriately on the curvature of its square root R), so we could have a very large quantum potential in an area where you have a negligible probability density (as is the case for instance of interference between two Gaussians as we shall see in later chapters). This can be seen by scaling R by an arbitrary factor λ: $Q = -\frac{\hbar^2}{2m}\frac{\nabla^2 \lambda R}{\lambda R} = -\frac{\hbar^2}{2m}\frac{\nabla^2 R}{R}$

Numerically speaking, given that in general ρ can vary exponentially, and thus has an enormous dynamic range, computationally it will be beneficial to represent it instead by its logarithm $C = \log \rho$. Not only are we compressing the range of ρ but also for instance in the case of Gaussian wavepackets we are going from $e^{a_2 x^2 + a_1 x + a_0}$

to $a_2x^2 + a_1x + a_0$ obviously a form more amenable to polynomial interpolation.

So with $\rho = e^C$and $R = e^{\frac{1}{2}C}$, inserted into equation 2.5 the quantum potential then takes the simplified form:

$$Q = -\frac{\hbar^2}{2m}\frac{1}{2}(\nabla^2 C + \frac{1}{2}(\nabla C)^2) \tag{2.6}$$

We can see how, as a consequence of the presence of the differential operators $\nabla$and ∇^2, the quantum potential encodes at each point information about its neighbours and ultimately about the state of the whole system.

This is essentially the reason for the non-locality observed in quantum phenomena such as entangled states where two potentially very distant parts of the system maintain high correlation among themselves, as for example in Einstein's EPR paradox (see [17]).

2.4 Bohmian Trajectories

2.4.1 Definition

The Bohmian trajectories are determined, just as in classical mechanics, from the velocity field generated from S according to the definition $v = \frac{\nabla S}{m}$.

At each point the particle moves perpendicular (parallel to the gradient) to the wavefront of S.

2.4.2 Important Properties

In Bohmian mechanics trajectories may not cross or even touch each other. If they were to cross, at the crossing point two "particles" with distinct momenta would share

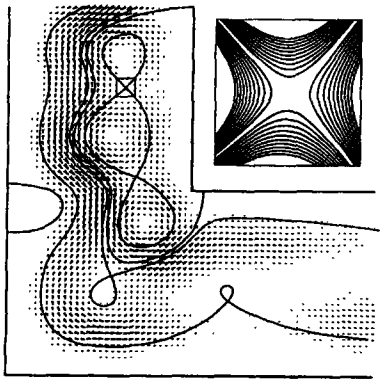


Figure 2.1: Node avoidance property
(From Hua Wu and D.W.L. Sprung [9])

the same space-time point which would imply that the underlying wavefunction should have two distinct values at the same point in space. Since the wavefunction is single valued, trajectory crossings are forbidden (see [15] and [9]). So if we see trajectories crossing we immediately know that something is wrong in our calculations.

Furthermore at nodes of the pilot wavefunction Ψ, the phase S is undefined and there will be no particle paths going through those points , instead we may have avoided crossings and or vortices (see [9]) develop around these points as illustrated in Fig. 2.1. Fig. 2.1 represents the steady state of a 2D L-shaped quantum wire at a particular energy (in this case $|t|^2 \approx 0.25$). Solid lines enclose regions containing vortices and the inset shows behaviour near a stationary point of the velocity, where the flow divides. At nodes S is no longer single valued and may undergo discrete jumps: $S_n = S + 2\pi\hbar n$ where n is an integer. The contour integral of v around a node will be $\oint v.dl = \oint \frac{\nabla S}{m} dl = \oint dS = \frac{2\pi\hbar}{m} n$ and will define a vortex when $n \neq 0$ (see [9])

These sudden jumps in S will give rise to big gradients and can cause numerical instability in algorithms, commonly referred to the node problem.

To illustrate the above, we use a compact Gaussian as the envelope of our ini-

tial wavepacket. It is positive definite everywhere thereby making the node problem aforementioned less likely to occur. The reason for that resides with the continuity equation which determines evolution of $\rho(x,t) = \rho(x_0,0)\exp(-\int_0^t \nabla v(x(t),t)dt)$ therefore if $\rho(x_0,0) > 0$ then $\rho(x,t) > 0$ for all time.

We can also make use of Ehrenfest's theorem which gives the evolution of the expectation value of an Hermitian operator to calculate the average values for the position and momentum of the system. For an arbitrary operator A,

$$\frac{d}{dt} < A >=< \frac{\partial A}{\partial t} > + \frac{i}{\hbar} < [H, A] > \tag{2.7}$$

here the expectation value $< A >$ stands for the average value of measurements of the observable A in a state $|\psi >$. By taking A to be the position operator x or momentum operator p, we obtain Ehrenfest's theorem, the equations of motion for their mean values.(eq. 2.8)

$$\begin{cases} \frac{d}{dt} < x >= \frac{<p>}{m} \\ \frac{d}{dt} < p >= - < \nabla V(x) > \end{cases} \tag{2.8}$$

We can then use eqs. 2.8 which agree with the classical equations of motion, to monitor the quality of our numerical solution by monitoring the average position and momentum along the trajectories.

2.5 Compatibility with the Copenhagen Interpretation

The ultimate test for any physical theory is agreement with experiment so we duly note that both standard quantum theory and Bohmian mechanics predict the same

experimental results, at least for single-particle problems.

One can always recover the wavefunction having solved the Bohmian trajectories by integrating along them the density and phase given by respectively the continuity equation and the Quantum Hamilton Jacobi equation.

But how does a totally deterministic theory with point particles and trajectories accommodate key quantum concepts such as uncertainty in results of measurement? The reason we "recover" these uncertainties and thus converge to the same results as conventional quantum mechanics, is that the initial conditions are not perfectly known (by design?) . We are given a statistical ensemble for the initial conditions corresponding to the wavefunction at that time, say $\rho(r_0, t_0) = |\Psi(r_0, t_0)|^2$ (where t_0 is the initial time and r_0 is some surface on which the boundary conditions are applied). From the solution of the TDSE, everything else will follow and the trajectories will retain their statistical weight through time, yielding measurements that are also probabilistic in nature.

2.6 Practical Advantages of the Trajectory Formulation

Not counting the supposed practical computing advantages, sometimes it is even advantageous to recalculate Bohmian trajectories from pre-existing time-dependent wavefunction solutions, since the former will give us a clearer understanding of some underlying physical concept(such as arrival time).

In some problems , when we are trying to answer kinds of questions that are better posed in a classical domain, the trajectory formulation makes it possible to give intuitive yet quantum mechanically correct answers. Such is the case for instance

when calculating lifetimes of particles when we want to know the probability a particular wavefunction will be contained by some potential. In the trajectory view one simply counts the number of particles/trajectories that have crossed a predetermined boundary.

An example of this is illustrated in figure 2.2, the deterministic individual trajectories are perfectly suited to explore questions of time of flight, lifetime etc. which are present in time-dependent quantum problems such as α decay. We can see that in the second case this probability is actually an oscillatory function of time [18], as we have trajectories that for a while tunnel in and out of the boundary .

Note that when using Bohmian trajectories we can determine with no ambiguity which part of the wavepacket passes and which part remains within the barrier. For instance we can also see in Fig. 2.2 that most of the transmittance is due to the front of the wavepacket and not the rear. This is something unthinkable in conventional quantum theory where the wavefunction is taken as a whole, and such information would be unavailable.

Besides these practical advantages of Bohmian mechanics, there is an important conceptual continuity between classical and quantum physics afforded by Bohmian theory[12] (at the cost of non-locality).

The emergence of classical phenomena out of a quantum world continues to be a major unsolved problem in Physics . The fact that Bohmian mechanics preserves the concepts of trajectories and particles from classical physics is a hint that Bohmian mechanics may have a role to play and be a more appropriate tool to approach this problem.

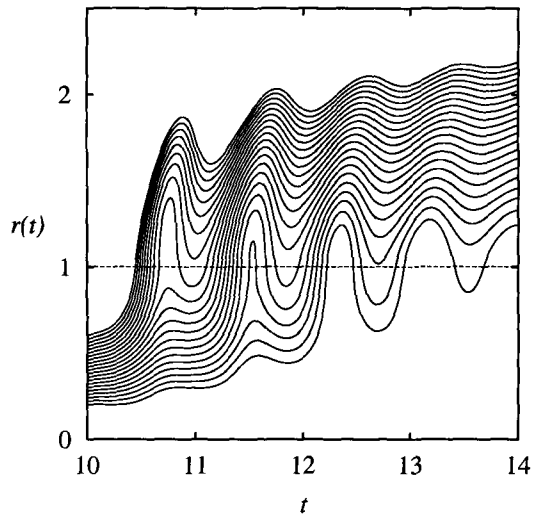


Figure 2.2: Evolution of Bohmian Trajectories in an escape problem
(From Nogami et al.[18])

2.7 Calculating Trajectories

2.7.1 From Wavefunction

Calculating trajectories from the wavefunction $\Psi(x,t)$ is really a two step process:

1. Solve the time dependent Schrödinger equation (TDSE).

2. Integrate the velocity field determined by the phase of the newly calculated Ψ.

We can solve the TDSE analytically for a relatively small number of simple systems (see [15, 19]).

For the vast majority of cases that cannot be solved analytically, with the advent of cheap and powerful computing, a suitable time propagation method can be used to numerically solve the TDSE in low dimensional systems.

There are many methods that can be used to solve the TDSE. We can for instance project the initial wavefunction onto stationary states of the system's Hamiltonian, and then evolve these according to their eigen-energies (spectral methods).

We can discretize the TDSE on a fixed space-time grid. This can be done using FFT methods, Feynman Path integrals, Monte Carlo techniques, or operator splitting methods, etc. We use the latter in a Crank-Nicholson scheme as represented in eq. 2.10 in conjunction with transparent boundary conditions as described in Moyer's paper ([20]) and in appendix A to setup a "golden" standard to which we can compare the various particle method implementations.

$$\Psi(r, t+\delta t) = \exp(-iH\delta t/\hbar) \times \Psi(r,t) \tag{2.9}$$

$$e^{\frac{-iH\delta t}{\hbar}} \approx \frac{1 - iH\frac{\delta t}{2\hbar}}{1 + iH\frac{\delta t}{2\hbar}} \tag{2.10}$$

In fact, we should note that this simple Crank-Nicholson method for 1D problems, has recently been generalized to higher orders described recently by W. van Dijk and F. M. Toyama (see [21]). They have obtained a dramatic improvement in attainable precision, for a given amount of computation.

2.7.2 Direct Trajectory Integration

In this method we directly use the Bohmian framework to make numerical calculations. The trajectories come out naturally as a constituent of the calculating procedure.

The direct Bohmian method is a more efficient way of calculating trajectories and because we use grids that are much sparser than those used in wave based methods, it usually is orders of magnitude faster. However it is vulnerable to numerical instabilities when interference effects are dominant. We will give more details on this in the next chapter.

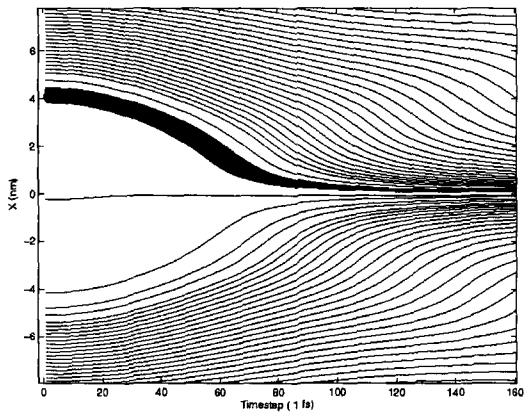


Figure 2.3: Bohmian trajectories $x(t)$ for a two-gaussian initial state.

2.8 Conservation of Probability

After we discretize the initial wavepacket into a representative statistical ensemble of particles, each of them can be thought of as carrying an associated probability volume. This probability volume will evolve in time according to $\frac{(d\rho J)}{dt} = \rho\frac{dJ}{dt} + J\frac{d\rho}{dt}$, which using the continuity equation can be put in the form $\rho J.\nabla v + J(-\rho\nabla v) = 0$ which is always zero. So it will remain constant in time, that is to say that each particle will have the volume "assigned to it" expand and compress as the probability density decreases or increases respectively, but it always represents the same probability volume. This is illustrated on Fig. 2.3, the inital state has two Gaussians placed symmetrically about the origin. As the wave packets spread they interfere but the trajectories do not cross. The shaded volume represents the same probability volume 1/N at all times and initially experiences expansion as the probability density decreases while the Gaussian packet is expanding, but then experiences compression when the Gaussian packet "meets" trajectories from the neighbouring packet, giving rise to an increased probability density.

This probability volume can be used to effectively label each particle (as is done

in [22]) since each one can easily (at least in 1D) be assigned a cumulative probability that will remain constant throughout time: $P(x_i, t) = \int_{-\infty}^{x} \rho(x', t)dx'$

In fact, in the Newtonian code of Sec. 3.4.2, which is how we will select our particle ensemble from $\Psi(x, 0)$: we will place a particle whenever the cumulative probability has increased by 1/N where N is the chosen number of particles. Thus between two trajectories there will always be the same amount of probability $\frac{1}{N}$ with the possible exception of the two particles at the extreme boundaries.

Chapter 3

Calculations

3.1 Prelude: Units

Quantum mechanical phenomena are mainly observed in the atomic domain, although increasingly they are relevant in meso and nanoscopic systems. When studying such systems, physicists and chemists find it convenient to work in the atomic system of units where $\hbar = m = 1$. The resulting Schrödinger equation for an electron simplifies to: $i\frac{\partial \Psi}{\partial t} = -\frac{1}{2}\nabla^2\Psi + V\Psi$

On Table 3.1 for convenience, we include the conversion factors between the atomic system of units and the SI system.

Physical quantity	Atomic Units	Description	SI units	
Action	$\hbar = 1$	$\frac{1}{2\pi}$ Planck's constant	$1.054571 * 10^{-34} Js$	
length	$a_0 = 1$	Bohr Radius	$5.2917 * 10^{-11} m$	
mass	$m_e = 1$	electron's mass	$9.109382 * 10^{-31} kg$	
Energy	$E_h = 1$	Hartree energy	$4.35974 * 10^{-18} J$	

Table 3.1: Atomic units ([23])

3.2 Relevance of Time-Dependent Methods

Why use a time dependent method? Since inception of quantum theory, time independent methods have always been preferred as being more tractable and efficient, and in most cases synthesizing the relevant physical processes taking place in a system.

Obviously if we are dealing with a time dependent Hamiltonian we have no choice but to use a time dependent method. However even in the case of a time independent Hamiltonian it can sometimes be advantageous to use a time dependent method.

To illustrate this let us consider spectral methods. The standard way these work is by projecting the initial wavefunction onto the eigenvector basis (stationary states) of the system, and then evolve in time a linear combination of these projections.

$$\Psi(t) = \Sigma a_k \phi_k e^{-\imath E_k t/\hbar} \tag{3.1}$$

with $a_k =< \phi_k|\Psi(0) >$and $H\phi_k = E_k\phi_k$

This is often the preferred route since initially we deal with space only and not space and time. However in some cases (i.e. scattering) the eigenstates become continuum functions making the discrete sum in equation 3.1 an integral over continuum states and if the calculation is required at very large times, it may be difficult to perform the Fourier integrals with sufficient accuracy. In such cases a time dependent method may be advantageous, an early example of which is in Heller's work ([24]).

3.3 Wavefunction Based Calculations

As we saw at the end of the previous chapter, Bohmian trajectories can be calculated a posteriori, after the full time evolution of the wavefunction $\Psi(x,t)$ is known. Since, apart from a handful of cases, analytical solutions are generally not available we

must turn to calculating the time evolution of the system via some standard method that is known to work. One such method, augmented by transparent boundary conditions, is described in the next sections.

3.3.1 Time Propagation

If we wish to compare the efficiency and accuracy of a Bohmian method, we need some standard to which it can be compared. Towards that objective we implemented a fixed grid split operator method that will serve as a "gold standard" in later sections.

The main reason we choose a particular method (see [20]) over other standard methods is that for this one, transparent boundary conditions have been devised and it is quite efficient. As for the method specifically, it uses the Crank-Nicholson scheme to propagate the system in time by symmetrizing the propagator : $e^{-\imath H\delta t} \approx 1 + iH\delta t - \frac{1}{2}H^2\delta t^2$ into the Cayley form: $e^{-iH\delta t} = \frac{1-iH\delta t/2}{1+iH\delta t/2} + O(\delta t^3)$, which is second order accurate, and more importantly unitary. The Numerov algorithm is used to extend the accuracy in the spatial domain to fifth order.

3.3.2 Boundary conditions

Transparent Boundary Conditions (TBC) are very convenient in that they allow us to concentrate on a small volume with great detail where the "interesting" physics may be happening, without worrying about unwanted wave reflections from artificial boundaries. Complex potentials are sometimes used as an absorbing layer on the boundaries to avoid having to enlarge the computing domain, but they never work perfectly.

In our case TBCs are also essential for easy comparisons between wave and particle methods, as TBCs are inherent to the Bohmian method (that is one of its advantages).

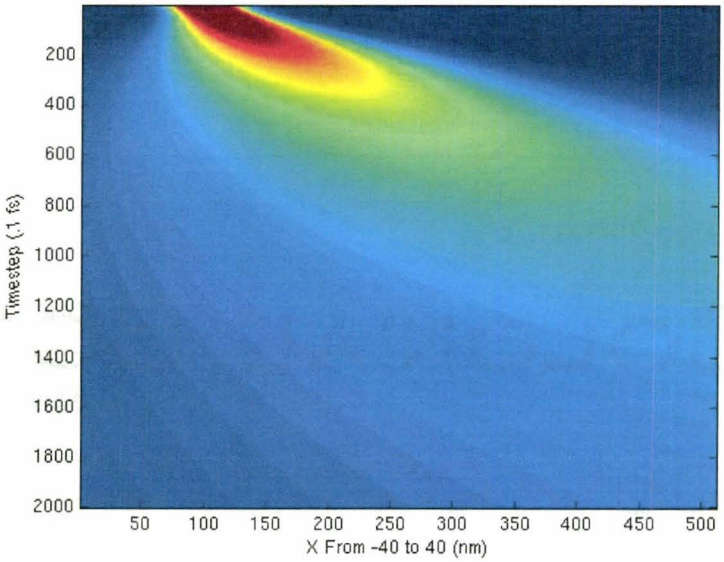


Figure 3.1: Time evolution of a free wavepacket in 1D
(no reflection is apparent at the domain boundaries, TBCs are working)

TBCs for different kinds of differential equations are discussed by Matthias Ehrhardt in [25]. In particular discrete TBCs in one dimension for the Schrödinger equation were derived in [26] and an adaptation to the Numerov method is presented by Moyer ([20]).

We can observe the time evolution of a free Gaussian with some initial momentum to the right, in Fig. 3.1; the packet spreads in time but no reflection occurs when it hits the domain walls because TBCs are enabled.

We should note that implementation of TBCs comes at a cost of the minor inconvenience that we need to record the wavefunction at the boundaries at each time step, and store this history. This has an increasing performance cost as time progresses. The reason these values are needed, is that at the system boundaries, there are fluxes of probability in both directions, and it is not valid to suppose purely outgoing flux, even when the net flux is to the right.

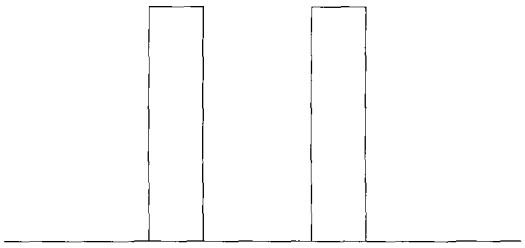


Figure 3.2: Schematic of a double barrier potential

A more detailed description of the TBC used is presented in appendix A.

3.3.3 Calculation of Trajectories

Let us take a more interesting example of a Gaussian packet hitting a double barrier, as illustrated by the potential in Fig. 3.2. We can observe the resulting time evolution in picture 3.3. Some reflection to the left from the first barrier, some transmission to the right past the second barrier and indeed some metastable states that become temporarily trapped in between the barriers (and eventually leak out) are seen.

We were faced with an interesting problem when trying to calculate trajectories from the wavefunction results. Remembering from Sec. 2.4 that the trajectories are determined by the particle velocities $v = \nabla S/m$, we then need a way to extract the phase information from the wavefunction, which is represented by a set of complex numbers. Unfortunately the standard way of simply taking the arctan(Imag/Real) will only give results in the $\pm\pi$ domain, so we need to monitor the function for points where sudden jumps of 2π are required for continuity. These are indicate a crossing of the ordinate axis as illustrated by Fig. 3.4 . If we were to take just the gradient of the phase provided by the computer code, we would be presented with unphysical velocity jumps at those points, resulting in erroneous trajectories.

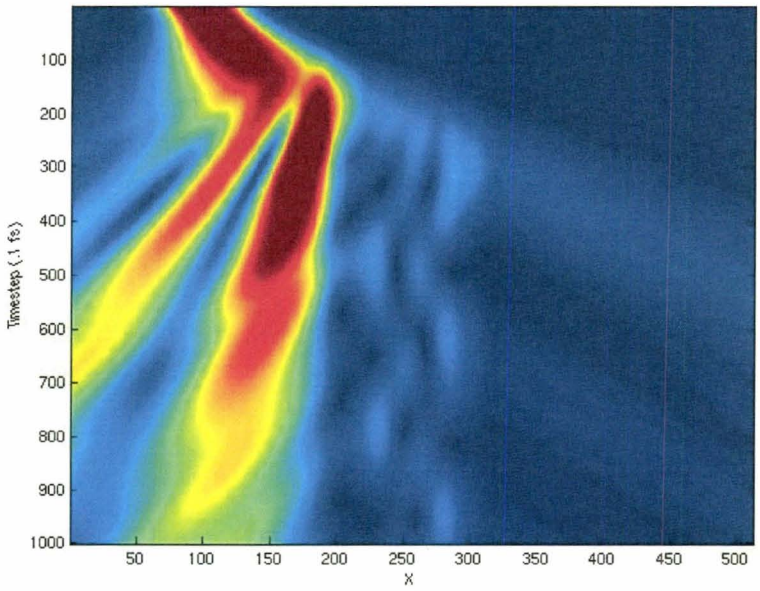
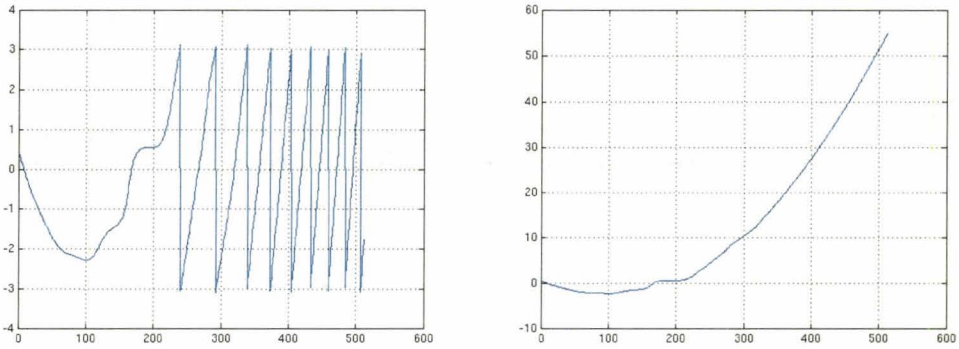


Figure 3.3: Time evolution of the wavepacket incident on a double barrier potential, from the left.

(Notice the transmitted and metastable states to the right and in the middle respectively)



(a) computed phase showing jumps of π (b) same, but with discontinuities resolved.

Figure 3.4: Example of phase wrapping around $\pm\pi$ and its solution, in 1D

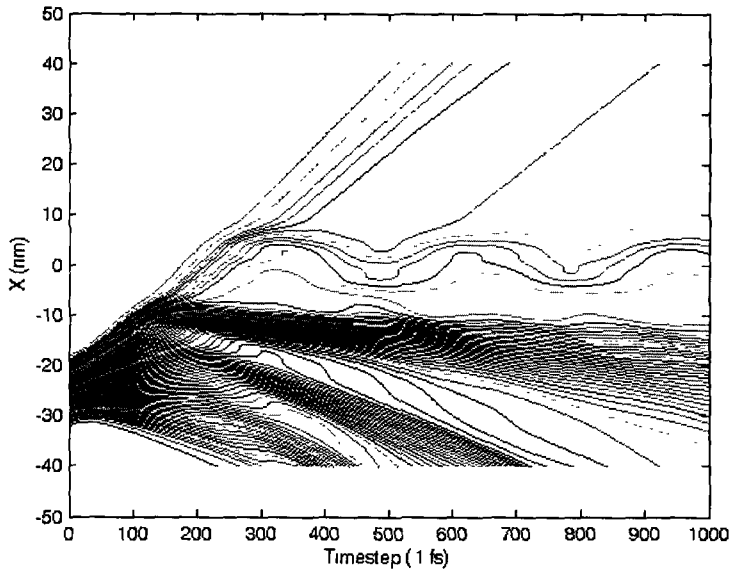


Figure 3.5: Bohmian trajectories corresponding to the wave function of Fig. 3.3 (viewing angle is rotated by 90 degrees, time is now x axis)

Once we have the correctly calculated velocities we can proceed to integrate the position $x = \int v dt$ from $v = \frac{dx}{dt}$ to get the updated positions of the particles and thereby define the trajectories.

An example of trajectories calculated from the wavefunction can be seen in Fig. 3.5. The metastable orbits inside the double barrier can more easily be seen than in Fig. 3.3 (as can their eventual tunneling to escape to the right or left side of the barrier).

3.3.4 Web Interface

We should add that, for convenience, at the time we developed a website that took as inputs the various parameters of a calculation such as dimensions, time steps, potential , initial wavepacket, etc and then fed these to a background running program. After a calculation related delay, a webpage with both the wavefunction solution and

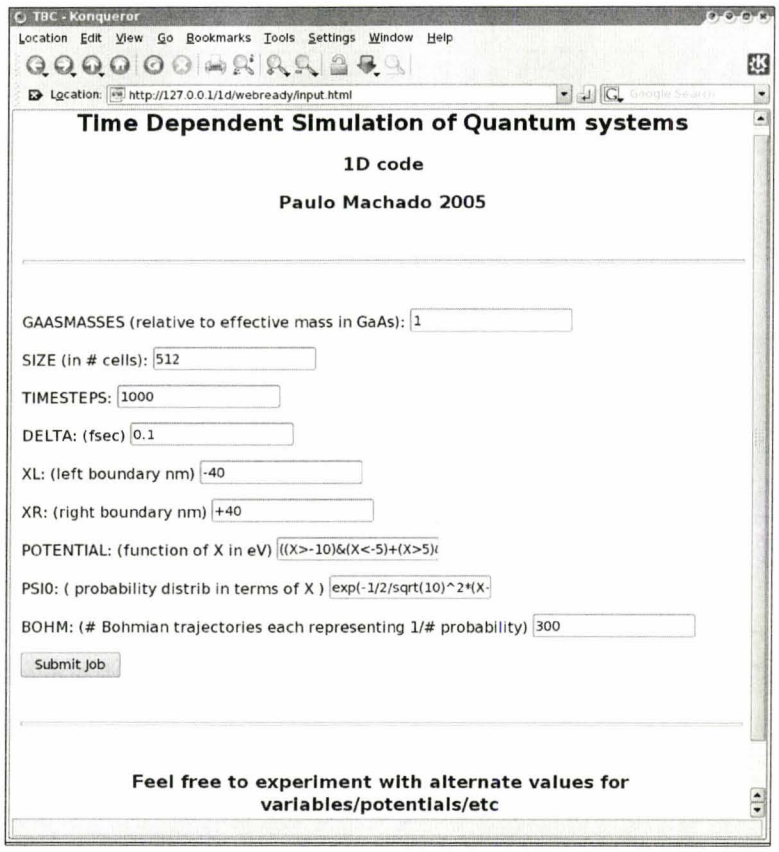


Figure 3.6: Typical run of the code via web interface

the corresponding calculated trajectories is output by the program.

A screenshot of a typical screen is shown in figure 3.6, the resulting page will present similar contents to figs 3.3 and 3.5.

We should also mention that recently, an improvement to the Crank-Nicholson scheme was introduced by W. van Dijk and F. M. Toyama (see [21]) that can be applied to further increase the accuracy of these wavefunction based calculations. Their method uses high order formulae in both space and time domains, typically of order h^{20}, where h is the step length. The extra labour per step is greatly overcompensated by using large steps in both space and time, reducing the overall computing time by several orders of magnitude. However, they have not yet incorporated TBC into their

methods.

3.4 Bohmian Calculations

First a word about boundary conditions. In trajectory based calculations the system is generally unbounded, as we simply keep track of the numerical value of the current position of each particle. In fact we can easily incorporate any kind of boundary conditions in the potential calculation.

This is in a marked contrast to grid based methods as illustrated in appendix A where we note that to enforce transparent boundary conditions (zero outside potential) an elaborate procedure is necessary that becomes more and more computationally intensive as time progresses (we need to keep a history of boundary values of the wavefunction).

3.4.1 Particle Approaches

As we noted in Sec. 2.4, the cumulative probability can be used as a convenient label for each particle, i.e. each particle carries with it a certain volume of probability volume through time that can be used to calculate the probability density. We can then choose one of two courses:

We can work with probability densities which we will have to update at each step via $\frac{d\rho}{dt} = \rho_0 e^{-\nabla \cdot v}$ and then evaluate ρ' and ρ'' to compute the quantum potential.

Or we can label each particle with its cumulative probability $P(x)$ at each spatial point which will then be constant in time at the cost of when ρ is required we use $\frac{dP}{dx}$ (from $P(x) = \int_{-\infty}^{x} \rho(y)dy$) , thus increasing by one order the derivatives required for the Q calculation. We will implement both methods of keeping track of the probability

density and compare them in terms of stability.

3.4.2 Algorithms - Newtonian Equations of Motion

Let us recall the Bohmian Equations of motion of Sec. 2.2.1:

$$\begin{cases} \frac{d}{dt}\rho + \rho\nabla v = 0 \\ \frac{dv}{dt} = -\frac{1}{m}\nabla(V+Q) \\ Q = -\frac{\hbar^2}{2m}\frac{\nabla^2 R}{R} \end{cases} \tag{3.2}$$

Using this set of equations the procedure to follow is this:

- Start with a statistical ensemble representative of the initial state of the system $\{x_i, v_i, P_i\}$ (we shall use a linked list for storage).
- If using P, from $\{x_i, P_i\}$ calculate ρ.
- Calculate Q from $\{x_i, P_i\}$ or $\{x_i, \rho_i\}$ depending on the labeling method used (more on this later).
- Update v_i via the gradient of $V+Q$ ($\frac{dv}{dt} = -\frac{1}{m}\nabla(V+Q)$).
- Update x_i from v_i ($x = \frac{dv}{dt}$).
- If we are not using P we need to update ρ_i according to $\frac{d\rho}{dt} = \rho_0 e^{-\nabla.v}$
- At this point we have advanced by δt in time and the whole process is repeated.

If we want to avoid dealing with P (and its 3rd order derivatives for Q) we can use ρ instead, provided we update it at each time step using the continuity equation as illustrated in the following section.

3.4.3 Algorithms - QHJ Equations of Motion

Also we can minimize the numerically troublesome task of taking gradients by using the Quantum Hamilton-Jacobi version of the equations of motion from 2.2, this time taking care to update the velocity of each particle as $v = \frac{\nabla S}{m}$:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla(\rho \frac{\nabla S}{m}) = 0 \\ \frac{\partial S}{\partial t} = -\frac{(\nabla S)^2}{2m} - (V + Q) \\ Q = -\frac{\hbar^2}{2m} \frac{\nabla^2 R}{R} \end{cases} \tag{3.3}$$

Using this set of equations the procedure to follow is this:

- Start with a statistical ensemble representative of the initial state of the system $\{x_i, v_i, \rho_i\}$
- Calculate Q from $\{x_i, \rho_i\}$
- Update S via the Quantum Hamilton-Jacobi Equation $\frac{\partial S}{\partial t} = -\frac{(\nabla S)^2}{2m} - (V + Q)$ (i.e. in an explicit central differences implementation, time evolution will take the form: $S(t + \delta t) = S(t - \delta t) - 2\delta t \left[\frac{(\nabla S(t))^2}{2m} + (V + Q) \right]$)
- Calculate v_i from $v = \frac{\nabla S}{m}$
- Update x_i from v_i
- Update ρ_i according to $\frac{d\rho}{dt} = \rho_0 e^{-\nabla . v}$ or equivalently $\frac{d\rho}{dt} = \rho_0 e^{-(\nabla^2 S)/m}$
- At this point we have advanced by δt in time and the whole process can be repeated.

3.4.4 Construction of the Wavefunction

At any time we can recover the standard wavefunction view of the system ([8]) by integrating both phase and density along a trajectory .

The evolved density is determined by integrating the equation of continuity:

$$\rho(x,t) = \rho(x,0)e^{-\int_0^t(\nabla.v(x,t))dt}$$

Evolution of the phase is obtained by integrating the quantum Hamilton-Jacobi Equation:

$$S(x,t) = S(x,0) + \int(\frac{dS}{dt})dt = S(x,0) + \int(\frac{(\nabla S)^2}{2m} - (V+Q))dt$$

Or combining both terms and referring to the original wavefunction we get:

$$\Psi(x,t) = \Psi(x,0)e^{\int_0^t(\nabla.v)dt}e^{\frac{i}{\hbar}\int_0^t(\frac{(\nabla S)^2}{2m}-(V+Q))dt} \tag{3.4}$$

3.4.5 Choice of Integrators

Given the equations of motion the question arises, what kind of integrator to use to evolve these in time. Since the time integration will be at the core of our procedure, special care in terms of accuracy and efficiency is required in picking the numerical integrator to use.

One other consideration is that since, as we can see from Sec. 3.4.2, we are integrating an Hamiltonian system, we would like certain constants of motion such as energy for a classical system and normalization, in a quantum system, to be conserved. The symplectic class of integrators, widely used in the fields of celestial mechanics and molecular dynamics, are designed with those properties in mind, so its remains

Coefficient	k=1	k=2	k=3	k=4
a_k	$\frac{2+2^{\frac{1}{3}}+2^{-\frac{1}{3}}}{6}$	$\frac{1-2^{\frac{1}{3}}-2^{-\frac{1}{3}}}{6}$	$\frac{1-2^{\frac{1}{3}}-2^{-\frac{1}{3}}}{6}$	$\frac{2+2^{\frac{1}{3}}+2^{-\frac{1}{3}}}{6}$
b_k	0	$\frac{1}{2-2^{\frac{1}{3}}}$	$\frac{1}{1-2^{\frac{2}{3}}}$	$\frac{1}{2-2^{\frac{1}{3}}}$

Table 3.2: Coefficient values for method S4a at each intermediate step k (reproduced from [28][29])

to choose one from that class.

To help us do that we refer to the paper by Gray (1994) [28] where he compares a range of symplectic integrators as well as more standard ones such as fourth order Runge-Kutta, in terms of their speed and conservation properties in a molecular dynamics context.

We selected the fourth order method S4a (which was originally introduced in [29]) because from all the methods examined it provided the best balance between speed and accuracy. The coefficients used, a_k and b_k, are taken from [28] and listed in table 3.2. They are used to integrate the equations of motion as directed by eq. 3.5.

$$\begin{cases} \dot{p} = -\frac{\partial H}{\partial q} \\ \dot{q} = \frac{\partial H}{\partial p} \end{cases} \rightarrow \begin{cases} p_{k+1} = p_k + b_k \tau F(q_k) \\ q_{k+1} = q_k + a_k \tau G(p_k) \end{cases} \tag{3.5}$$

where $F = -\frac{\partial V}{\partial q}$ and $G = \frac{\partial T}{\partial p} = \frac{p}{m} = v$ and you take four steps $k = 1..4$ to update the values of p and q.

3.5 Evaluation of the Quantum Potential

We have seen in Sec. 2.4 that each particle carries with it a predetermined amount of probability . We usually pick $\frac{1}{N}$ for each trajectory when using the Newtonian form of the Bohmian equations. In the QHJE case we often use an initially uniform grid

in which each probability volume will be set but different for each trajectory. In either case the cumulative probability that it carries (at least in 1D) unequivocally identifies the particle and can be used to label it. When required we can calculate the probability density as $\rho = \frac{dP}{dx}$ which follows from the definition of $P(x) = \int_{-\infty}^{x} \rho(y)dy$

Depending then on which variables we choose to work with (they are going to vary as with different methods of interpolation), Q the quantum potential and F_Q the quantum force will be given by:

$$Q = -\frac{\hbar^2}{2m}\frac{\nabla^2 R}{R}$$

or, with $R = \sqrt{\rho}$,

$$Q = -\frac{\hbar^2}{2m}\frac{1}{4}(2\frac{\nabla^2\rho}{\rho} - \frac{(\nabla\rho)^2}{\rho^2}) \tag{3.6}$$

$$F_Q = -\nabla Q = \frac{\hbar^2}{2m}\frac{1}{2}(\frac{\nabla^3\rho}{\rho} + (\frac{\nabla\rho}{\rho})^3 - 2\frac{\nabla\rho\nabla^2\rho}{\rho^2}) \tag{3.7}$$

In terms of P,

$$Q = -\frac{\hbar^2}{2m}\frac{1}{4}(2\frac{\nabla^3 P}{\nabla P} - \frac{(\nabla^2 P)^2}{(\nabla P)^2}) \tag{3.8}$$

$$F_Q = -\nabla Q = \frac{\hbar^2}{2m}\frac{1}{2}(\frac{\nabla^4 P}{\nabla P} + (\frac{\nabla^2 P}{\nabla P})^3 - 2\frac{\nabla^2 P\nabla^3 P}{(\nabla P)^2}) \tag{3.9}$$

Or instead, using ρ and determining its logarithm we obtain with $C = \log R = \frac{1}{2}\log\rho$:

$$Q = -\frac{\hbar^2}{2m}(\nabla^2 C + (\nabla C)^2) \tag{3.10}$$

$$F_Q = \frac{\hbar^2}{2m}(\nabla^3 C + 2\nabla^2 C \nabla C) \tag{3.11}$$

This logarithmic form has a number of advantages:

- It more naturally represents an exponentially decaying probability.
- It avoids the problem of dividing by a possibly very small number .
- Last but not least, C lends itself more naturally to a polynomial interpolation than ρ (i.e. for a Gaussian C is a quadratic polynomial).

The non-locality of Q is patent in the ∇ and ∇^2 terms and presents a significant numerical challenge. In the next few sections we will present some ways of extracting their values from our discrete set of trajectories.

3.6 Interpolation Schemes

3.6.1 Quantum Potential by Polynomial Interpolation

We are given a set of discrete points and are asked to calculate gradients, the simplest and more intuitive choice is to approximate these points by continuous polynomial functions which can easily be differentiated.

This interpolation scheme yields the most local approximations to Q and in fact for the case of 1 or 2 neighbouring points it will reduce to the common two and three point formulas for the first derivative, respectively.

Given a particle and its N-1 nearest neighbours at distances Δx_n with values y_n we want to determine the coefficients of the interpolation polynomial $y = c_1 + c_2 x + c_3 x^2 + ...$ where $c_n = \frac{1}{n!}\frac{d^n y}{dx^n}|_{x=x_0}$

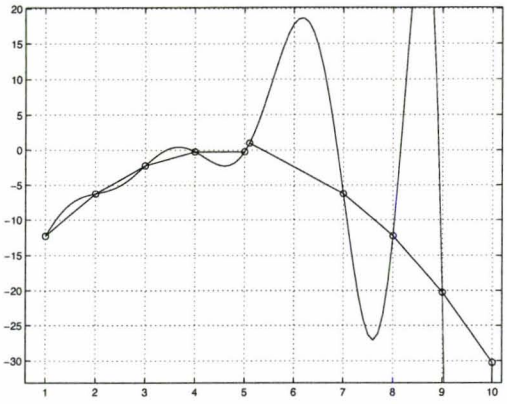


Figure 3.7: Induced oscillations in a high degree polynomial fit
(the underlying is a simple parabola with only one offset point near x=5)

They are determined by solution of the linear system:

$$\begin{bmatrix} 1 & \Delta x_1 & (\Delta x_1)^2 & (\Delta x_1)^3 & \dots \\ 1 & \Delta x_2 & (\Delta x_2)^2 & (\Delta x_2)^3 & \dots\dots \\ 1 & \Delta x_3 & (\Delta x_3)^2 & (\Delta x_3)^3 & \dots\dots \\ 1 & \Delta x_4 & (\Delta x_4)^2 & (\Delta x_4)^3 & \dots\dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ \dots \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \dots \end{bmatrix} \tag{3.12}$$

As the book Numerical Recipes [31] reminds us, this is a Vandermonde Matrix which can be quite ill-conditioned, so we can expect the coefficients thus obtained to be undesirably sensitive to small variations in the data points.

Instability is especially obvious when a large number of points is involved in the polynomial interpolation. This is illustrated in Fig. 3.7 where we start with a perfect parabolic distribution of points and slightly shift only one point (near 5). We can observe how a high degree polynomial fit tends to oscillate wildly around the interpolated points as it tries to pass exactly through each and every one of them (Runge's Phenomenon).

Lagrange interpolation thus has some desirable characteristics such as simplicity

and the fact that it hits each point exactly but as one makes it less local (by including more neighbouring points in the interpolation) it becomes less and less stable.

One should note as well that given the form of eq. 3.11 and in particular its $\nabla^3 C$ term that the polynomial used should be of at least fourth degree in order to contribute to that term in the quantum force. This observation holds for other types of interpolation that depend on polynomials such as the one discussed in the next section.

3.6.2 Least Squares Approximation

We noted before that Q is non-local, so one can attempt to capture more of this non-locality by including a larger number of nearest neighbours in the polynomial interpolation. Unfortunately as we saw in Sec. 3.6.1 and illustrated by Fig. 3.7 the polynomial interpolation does not do well with many points, becoming hypersensitive to noise.

If we increase the number of points in the interpolation without increasing the degree of the polynomial we get an overdetermined system of equations that we can solve optimally using the least squares method.

We wish to fit a polynomial function of the type:

$$y = c_0 + c_1 x + c_2 x^2 + ... + c_N x^N$$

Again we get a linear system similar to eq. 3.12 but now for M points, we have a MxN matrix with $M > N$ which we can solve using for instance the SVD method (Singular Value Decomposition, see [56]). This is now a semi-local evaluation of the derivatives since we are taking information from more and more distant neighbouring particles when evaluating gradients.

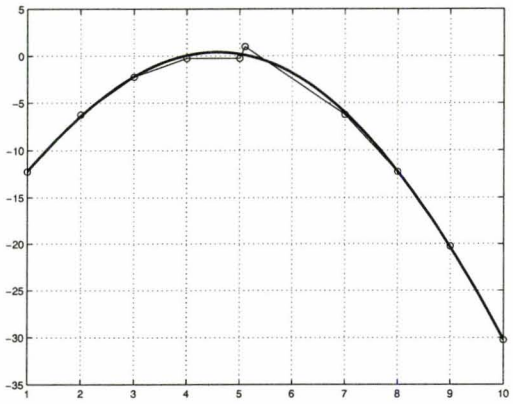


Figure 3.8: Least squares approximation to the same points as in Fig. 3.7

The difference is dramatic in Fig. 3.8, specially when we compare it with Fig. 3.7 of the polynomial interpolation. The interpolating points are the same for both cases, yet we can see how the least squares approximation reproduces almost exactly the original generating quadratic form.

Overall the least squares method results in a smoother approximation to the underlying distribution and therefore should lead to more physically valid results when evaluating its gradients (especially if we have some previous knowledge of the underlying distribution). We note however that because of this weighted approach the interpolating function will not, in general, pass through each point. In fact in Fig. 3.8 we can see that it basically ignores the sixth point from the left. This averaging property can lead to trajectory crossings down the line (more on these in the next chapter).

3.6.3 Other Approaches

The interpolation methods described in this chapter tend to be unsatisfactory in all but the simplest situations. In the next chapter we will explore alternative ways of evaluating Q with better stability properties.

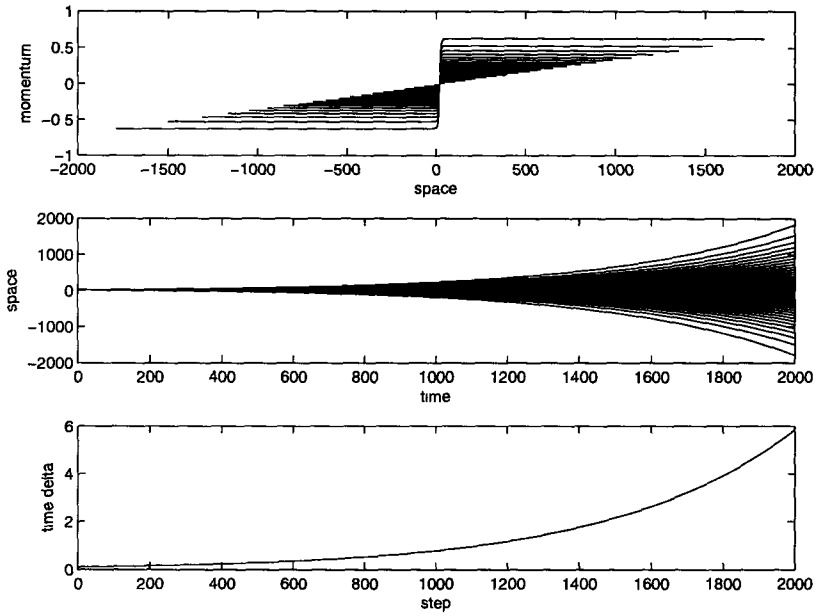


Figure 3.9: Example of free packet evolution using least squares interpolation
Top: Phase diagram
Middle: Space-time diagram
Bottom: Dynamical timescale (here it is always increasing)

As a testimony to the computational versatility of the Bohmian interpretation, there are a plethora of numerical implementations of the basic Bohmian equations. Some of the methods we didn't mention here include the derivative propagation method, covering function , Wigner function approach, etc. (see [8] for an overview).

Chapter 4

Numerical Breakdown

The particle method works quite well and is known to be stable for Gaussian packets in free space, linear and quadratic potentials, where the time evolution of the wavefunction is shape preserving (these are incidentally, cases where analytical solutions also exist).

However, if we introduce a minute cubic or quartic term in the potential or a barrier, the system will, given enough time steps, become unstable, eventually leading to nonsensical conditions such as trajectory crossings, the telltale sign that something has gone wrong.

In this chapter we will explore the causes and conditions that affect the stability of the particle method. We will be very interested in what role numerical instabilities caused by interference play in this, and finally we will introduce some alternate ways of evaluating the quantum potential that may have better stability and fidelity properties.

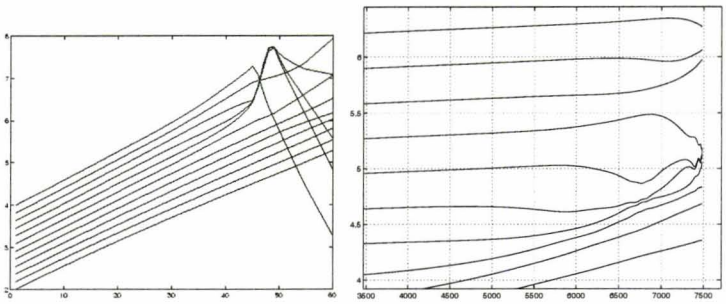


Figure 4.1: Trajectory crossings caused by numerical errors

4.1 Anatomy of a Trajectory Crossing

Referring back to Sec. 2.4.2, we recall that one of the important properties mentioned was that particle trajectories should not cross; this is a consequence of the single-valuedness of the underlying wavefunction. If we do observe crossings, we can be assured that something has gone wrong with the calculations that led to those crossings (see Fig. 4.1).

When two trajectories approach each other in a sudden manner, it becomes difficult to reliably estimate ρ from $\frac{dP}{dx}$. Furthermore because of the denominator term in Q, when ρ tends to 0 , we can expect to see big swings in the value of Q (and high gradients) which in turn lead to high particle velocities which may then catapult a particle towards another. In fact because $\rho = \frac{dP}{dx}$ when they cross, dx tends to zero and thus ρ becomes infinite, or in numerical terms "diverges", that can be seen by the sudden scattering of particles in Fig. 4.1.

In the real world there has to be a force that prevents the trajectories from getting too close. This is the quantum force term (Sec. 3.5), and a trajectory crossing is an indication of too big a time step or a badly calculated quantum force (i.e. insufficient magnitude or pointing in the wrong direction).

To illustrate how this could happen, let us take the example of the least squares

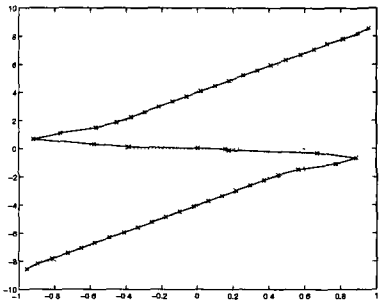


Figure 4.2: Rotated view of ∇S
(points near 0.2 are about to cause a trajectory crossing)

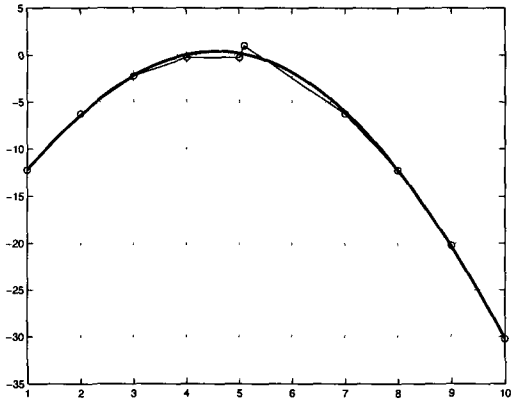


Figure 4.3: The smoothing property of Least Squares
(it largely ignores the particle shift near 5)

method of Sec. 3.6.2.

We start with an uniform sampling of points taken from a simple quadratic form. Now we artificially move particle 6 to the left (see Fig. 4.3). We can see that the characteristic smoothing responsible for the stabilization of the derivatives here works against us and smoothes out the crucial area between the now almost overlapping points 5 and 6. This smoothing prevents the particles from "seeing" each other when calculating Q with the least squares interpolation and thus we don't have a suitable quantum force to "repel" the particles and thereby prevent a crossover.

There are then two issues that should be addressed in order to prevent particles

from getting excessively close to each other: The proper interpolation of Q and a time step small enough to enable the particles to respond to Q and decelerate. The time step issue will be addressed in the next Sec. 4.3.2.

4.2 Remedies

4.2.1 Artificial Friction Term

By introducing a term of the form $V_f = -f_c \frac{dx}{dt}$ in the Hamiltonian ([44][45]) where f_c is the friction coefficient, we artificially introduce dissipation into the system. We will obviously destroy any energy conservation properties so it is certainly not a way to get the proper time evolution of the system. However a slight amount of artificial friction can be beneficial to the stability of the simulation and may be introduced so as to dampen spurious numerical oscillations that typically occur in the evaluation of gradients. If the friction is low enough, we can do this without completely changing the character of the system.

If one takes this to an extreme, and increases the friction too much, the system will evolve into its ground state. The ground state in a Bohmian context is peculiar in that all movement stops as the particles remain static at the bottom of the potential. In Fig. 4.4 we can see the system relaxing towards the ground state of a potential well.

4.2.2 Fourier Residues

This is an alternate interpolation scheme that combines a polynomial least squares interpolation with a Fourier expansion to approximate the log density.

The rough rationale for this combination is that the polynomial form arises nat-

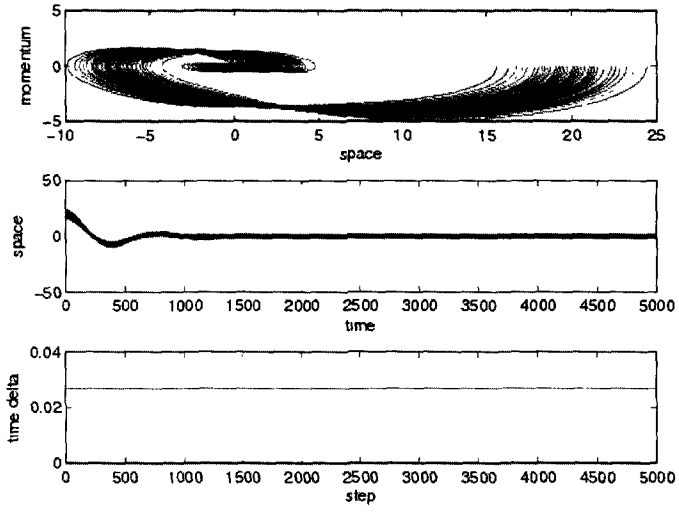


Figure 4.4: Wavepacket evolution with a large friction coefficient
From top to bottom: Phase space, time-evolution and dynamic time scale

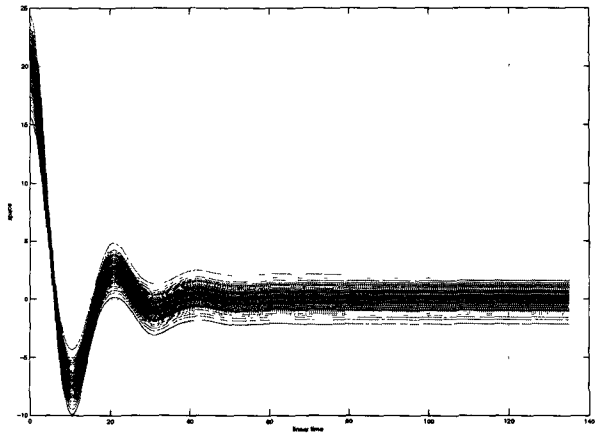


Figure 4.5: Resulting linear time-evolution of the system of Fig. 4.4
(note the convergence to the ground state, where the Bohmian particles remain motionless)

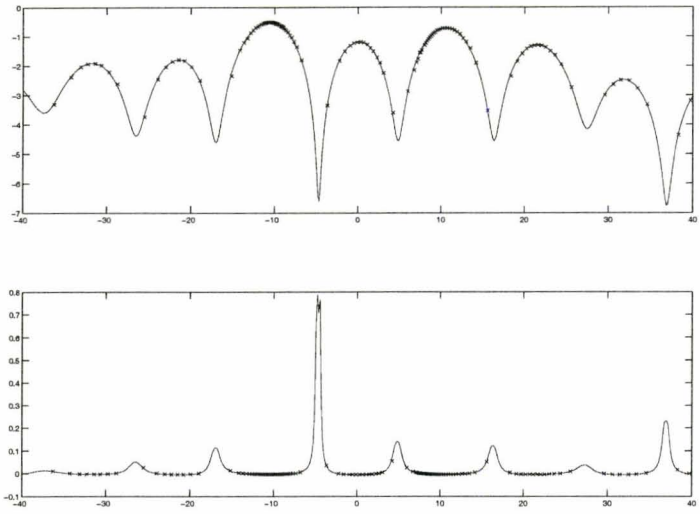


Figure 4.6: Example of interference
Top: log density profile
Bottom: Corresponding -Q
(Sampling particles spaced by 1% probability)

urally from the use of Gaussian wavepackets (where the log density is a quadratic form) and the fact that when we have potential reflection we should expect interference terms to start developing in the density function (see Fig. 4.6).

These interference terms are characteristically oscillatory in nature and thus favorably inclined to be represented by a Fourier expansion.

Mathematically the approximation to the density with an oscillatory residue is:

$$\rho \approx \exp(\alpha_0 + \alpha_1 x + \alpha_2 x^2 + ... + \alpha_m x^m + \mathrm{Res}(x)).$$

$$C \approx \alpha_0 + \alpha_1 x + \alpha_2 x^2 + ... + m x^m + \mathrm{Res}(x) \equiv \mathrm{Poly_m}(x) + \mathrm{Res}(x).$$

The polynomial (of order m) part of C, the log density, can be determined for

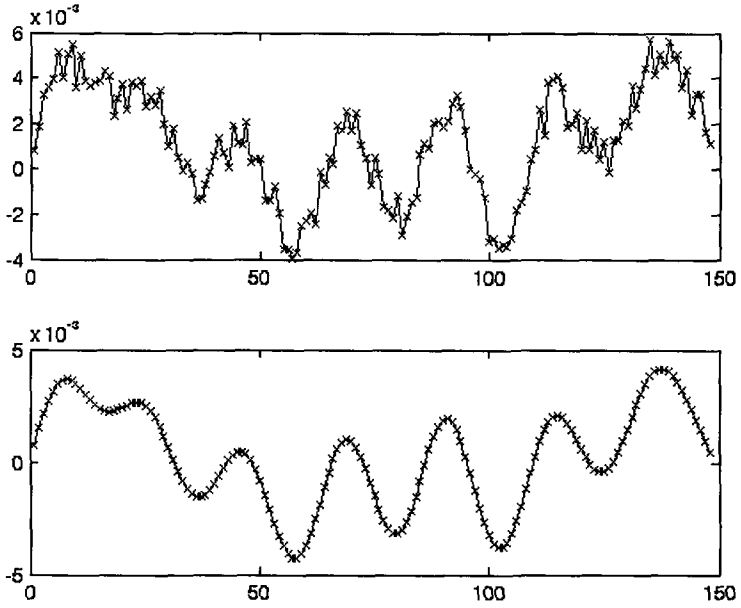


Figure 4.7: The residue of a particular wavefunction and the result of low pass filtering

instance in a least squares way. Once that is determined, the residue $\mathrm{Res}(x) = C(x) - \mathrm{Poly_m}(x)$ is calculated and can be expanded in a (discrete) sine series (DST) as in equation 4.1.

$$y(k) = \Sigma_{n=1}^{N} x(n) \sin(\pi \frac{kn}{N+1}), k = 1, ..., N \tag{4.1}$$

Recalling equation 3.11, in order to calculate the quantum force we will require C',$C''$and C''':

$$F_Q = \frac{\hbar^2}{2m}(\nabla^3 C + 2\nabla^2 C \nabla C)$$

The calculation of derivatives for the polynomial part of C is straightforward. To determine the derivatives for the residue part we should differentiate in Fourier space.

Since we are now working in Fourier space, at this point if desired, we can low-pass filter the residue in momentum space (Fig. 4.7) by adjusting the coefficients of Res(k), and then make use of the differentiating property of Fourier transforms $F(\frac{df}{dx}) = ikF(f)$ to calculate the needed derivatives of the residue as shown on the right side of eqs. 4.2

$$\begin{aligned} C' &= \text{Poly}' + F^{-1}(ik.\text{Res}(k)) \\ C'' &= \text{Poly}'' + F^{-1}(-k^2.\text{Res}(k)) \\ C''' &= \text{Poly}''' + F^{-1}(-ik^3.\text{Res}(k)) \end{aligned} \tag{4.2}$$

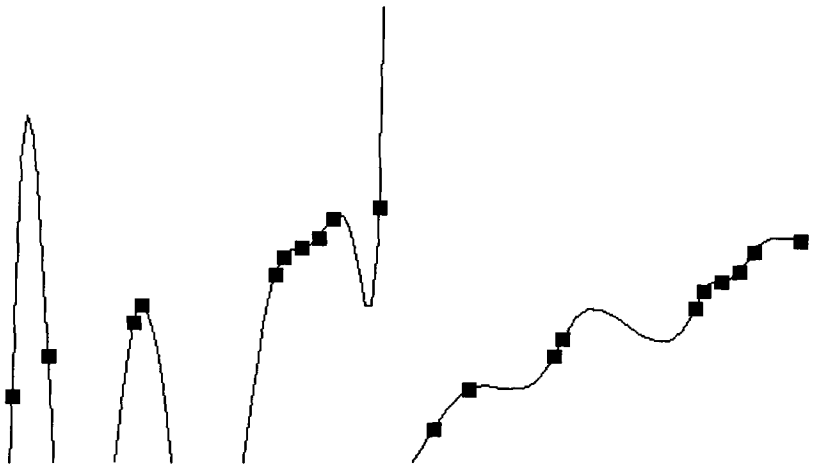
Here F^{-1} is the inverse discrete sine transform (IDST) that takes us back to position space and is the reverse of eq. 4.1:

$$x(n) = \tfrac{2}{K+1} \ \Sigma_{k=1}^{K} y(k) \sin(\pi \frac{kn}{N+1}), n = 1, ..., K \tag{4.3}$$

This is the ultimate non-local approximation to Q since we take information from every other point to evaluate the gradients $C^{(n)}$ at each location. Practically, although the estimates for $C^{(0)}$ correlate very well with C , the estimated gradients tend to oscillate for points near the edge, giving rise to problems in future time steps.

4.2.3 Spline Interpolation

We have seen in Sec. 3.6.2 that the least squares method, although less local, smoother and more reasonable looking than the polynomial interpolation, will not prevent trajectory crossings because it doesn't pass through every point.



(a) Polynomial interpolation (b) Spline interpolation

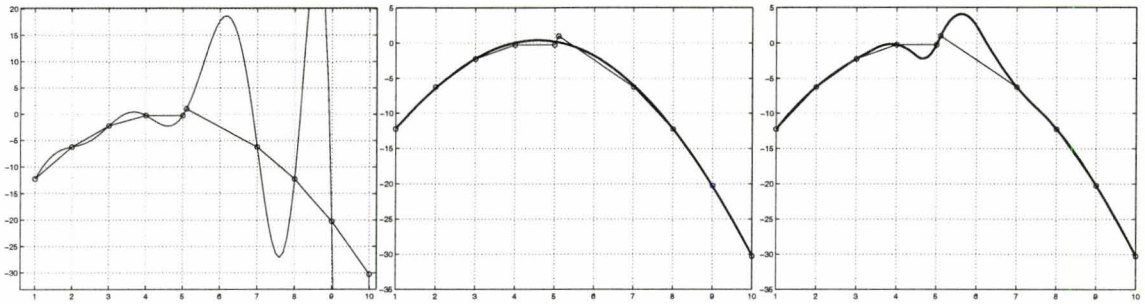
Figure 4.8: Comparison of two interpolation methods with same arbitrarily distributed points
(The spline method is distinctly more physical)

Ideally we would like to have an interpolation of C or ρ, that although smooth and taking information from as many points as possible, still manages to pass through each point exactly so that trajectory crossings may be avoided.

This should be in the spirit of the polynomial interpolation but without the stability issues that riddle the former, and somewhat in the spirit of the least squares approximation which takes information from as many points as possible, but being certain to pass through each fitted point, exactly. A method that satisfies both requirements is the spline approximation, where you approximate piecewise with polynomials.

In Fig. 4.8 illustrate the difference between the polynomial and the spline methods, and in particular the fact that the splines still pass through every point, while the overall function looks much more smooth(and physical) than the polynomial.

In Fig. 4.9 we see the effect of shifting that sixth point towards the left in the least squares and splines method, the latter passes through each point and doesn't



(a) Least Squares interpolation (b) Least Squares interpolation (c) Spline interpolation

Figure 4.9: Interpolation methods compared

seem to suffer from big oscillations as does Lagrange interpolation.

To calculate the quantum force, we will use C instead of ρ, as its predominantly polynomial shape is better suited to being interpolated by splines. Because we need terms of order $\nabla^3 C$ in eq. 3.11, if we are using P , the cumulative probability, the degree of splines we should use is four to make the quantum force term continuous . Therefore we want to use at a minimum quartic splines instead of the more commonly used cubic splines (see [31]) to interpolate C (a good guide on the method of splines is Carl De Boor's book[38]).

4.2.4 Smoothed Splines

Unfortunately the spline interpolation is liable to develop oscillations in Q at the edge of the interpolating domain. These propagate in time and eventually compromise the whole simulation. An example of this phenomenon is presented in Fig. 4.10.

As we saw in the previous section the spline method has so many other desirable properties, that we should try to fix this edge effect. One way of doing that is to use a smoothing spline with a site dependent parameter p that will make the interpolation smoother at the edges, and less sensitive to numerical oscillations.

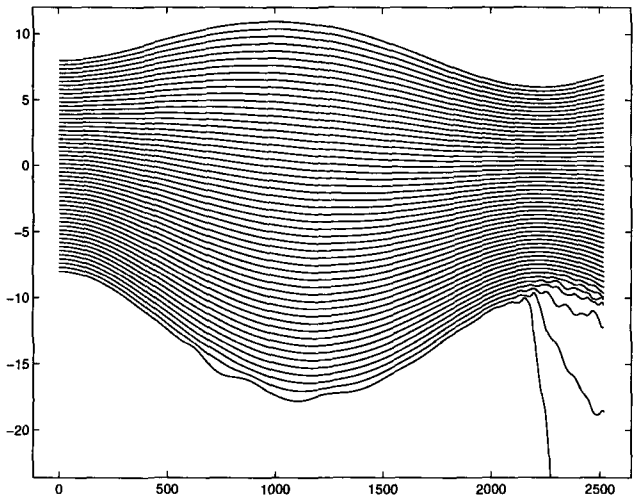


Figure 4.10: Trajectories $x(t)$ where edge instabilities develop

The smoothing spline SS will minimize:

$$p\sum_{j=1}^{n} w(j)(C - SS(x))^2 + (1-p)\int \lambda(x)(\nabla^2 SS)^2 dx$$

The parameter p will control how smooth the spline will be by emphasizing the point distance ($p \rightarrow 1$) or the low gradients ($p \rightarrow 0$). C is the usual log-density known at sites x_j and w and λ are site dependent weights. We can, by adjusting this local parameter p, generate a smoother interpolation of C near the edges while remaining close to the original splines in the middle points.

4.2.5 Train of Gaussians

This alternate interpolation scheme takes its inspiration from the field of digital signal processing, in particular in the Shannon sampling of a discrete signal, and specifically of an irregularly sampled discrete signal as our Bohmian points really are.

Because the density is at all times positive definite we should enforce this by substituting the sinc function from Shannon sampling which allows for negative values

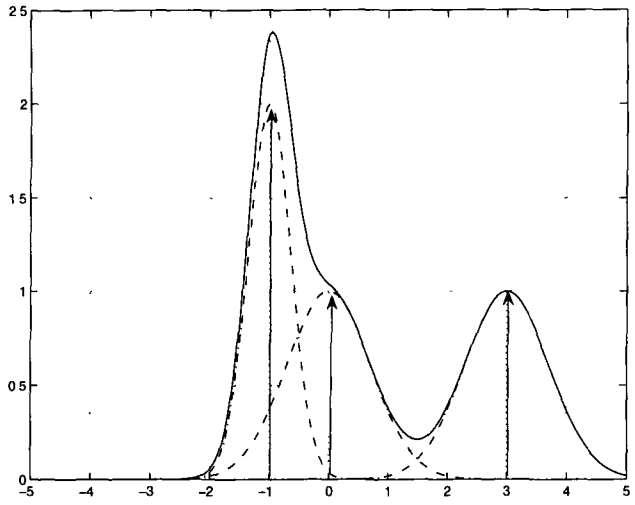


Figure 4.11: ρ representation by a train of Gaussians
(the solid line is the resulting ρ and arrows represent sampling points)

with a Gaussian shape that is convolved at each sample point represented by a delta function. This procedure is illustrated graphically in Fig. 4.11.

Normally each point represents the same probability volume, thus the Gaussians will have the same volume, resulting in sharper distributions when points are closer together and more diffuse ones when points are far apart. Mathematically what we are doing is convolving a series of delta functions (the particle locations) with Gaussians of varying standard deviation but the same area (thus the different heights in Fig. 4.11).

$$\begin{aligned} \rho(x) &= \frac{1}{N}\Sigma_{n=1}^{N}\delta(x-x_n) * G_n(x,\sigma_n) \\ \rho(x) &= \frac{1}{N}\Sigma_{n=1}^{N}G_n(x-x_n,\sigma_n) \end{aligned}$$

where $G_n(x,\sigma_n)$ represents a Gaussian centered at 0 with a standard deviation of σ_n.

$$G(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{4.4}$$

Recalling the form of the quantum force when using ρ as in eq. 3.7 we are going to need terms of the form $\frac{\nabla^3\rho}{\rho}$, $(\frac{\nabla\rho}{\rho})^3$and $\frac{\nabla\rho\nabla^2\rho}{\rho^2}$. Since we can analytically differentiate the Gaussians, it is just a (laborious) matter of calculating G_n,G'_n,G''_n and G'''_n and combining these according to eq. 3.7 to get an analytic form of Q for a given particle distribution:

$$\begin{aligned}
G(x,\sigma) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \\
G'(x,\sigma) &= -\frac{x}{\sigma^2} G(x,\sigma) \\
G''(x,\sigma) &= \frac{x^2-\sigma^2}{\sigma^4} G(x,\sigma) \\
G'''(x,\sigma) &= \frac{x(3\sigma^2-x^2)}{\sigma^6} G(x,\sigma)
\end{aligned}$$

Having the different derivatives of G we can now express the needed terms of $\rho(x)$ as:

$$\begin{aligned}
\rho(x) &= \frac{1}{N}\Sigma_{n=1}^{N} G_n(x-x_n,\sigma_n) \\
\frac{\nabla\rho(x)}{\rho(x)} &= \frac{\Sigma_{n=1}^{N} -(x-x_n) G_n(x-x_n,\sigma_n)}{\sigma^2 N\rho(x)} \\
\frac{\nabla^2\rho(x)}{\rho(x)} &= \frac{\Sigma_{n=1}^{N} \left((x-x_n)^2-\sigma^2\right) G_n(x-x_n,\sigma_n)}{\sigma^4 N\rho(x)} \\
\frac{\nabla^3\rho(x)}{\rho(x)} &= \frac{\Sigma_{n=1}^{N} \left((x-x_n)\left(3\sigma^2-(x-x_n)^2\right)\right) G_n(x-x_n,\sigma_n)}{\sigma^6 N\rho(x)}
\end{aligned}$$

Finally we can plug these values in the quantum potential or the quantum force

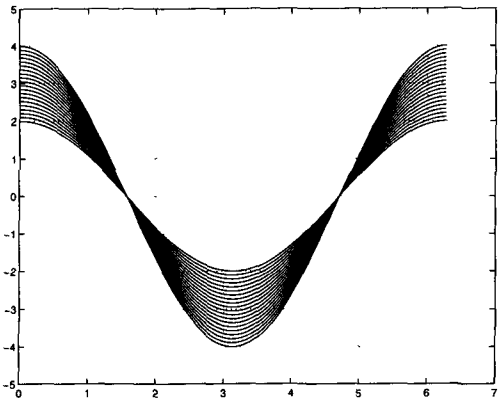


Figure 4.12: Trajectories $x(t)$ for the harmonic oscillator in the classical limit (t vs x)

as required:

$$Q = -\frac{\hbar^2}{2m}\frac{1}{4}(2\frac{\nabla^2\rho}{\rho} - \frac{(\nabla\rho)^2}{\rho^2})$$

$$F_Q = -\nabla Q = \frac{\hbar^2}{2m}\frac{1}{2}(\frac{\nabla^3\rho}{\rho} + (\frac{\nabla\rho}{\rho})^3 - 2\frac{\nabla\rho\nabla^2\rho}{\rho^2})$$

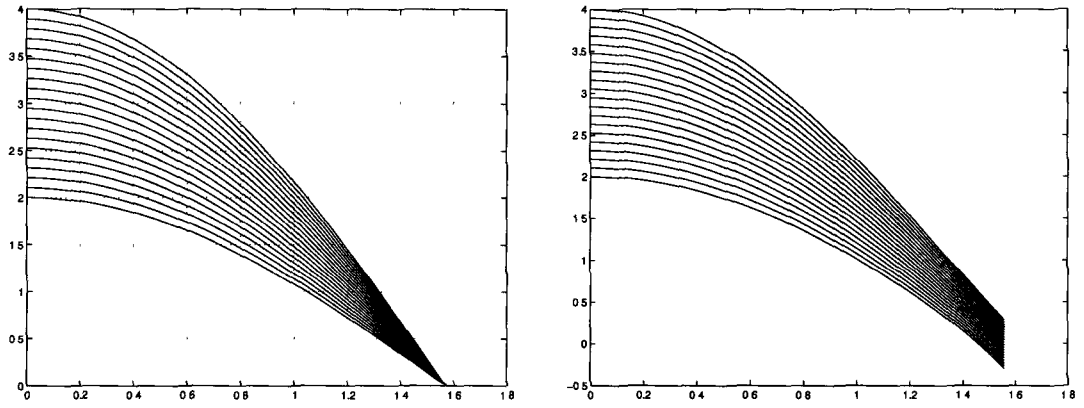
Classical limit from vanishing Q The classical limit is approached whenever the system's action is large compared to $\hbar$ or alternatively $Q \to 0$.

By turning off the quantum potential we can appreciate the differences between the classical and quantum worlds.

In Figure 4.12, which represents an harmonic oscillator, we can see that with $Q = 0$ we have trajectory crossings and every particle follows a sinusoidal path independent of the other particles.

Fig. 4.13 represents the same potential. Here we can see that the quantum trajectories are no longer independent, and they do not cross.

Finally in Fig. 4.14 we have a hint of how classical behaviour may emerge from



(a) Classical

(b) Quantum case, note the trajectories do not colapse

Figure 4.13: Detail of effect of "turning on" the quantum potential, over a quarter period of the oscillation (t vs x)

Bohmian trajectories. In this case we use the train of Gaussians interpolation method, but with somewhat sharper density distributions than usual.

Looking at Fig. 4.15 it is apparent that we don't actually have trajectory crossings but if we add a property of indistinguishability to the neighbouring particles those trajectories would appear to cross when we zoom out back to Fig. 4.14 . This is speculative but it would seem that Bohmian mechanics would, in this way, enable a natural emergence of classical behaviour.

4.3 Interference

> "The double slit experiment has in it the heart of quantum mechanics. In reality it contains the only mystery.."
>
> Richard Feynman

How does interference arise in our bohmian context? To study that, we are going to use two Gaussians freely evolving in space and eventually interfering together. To

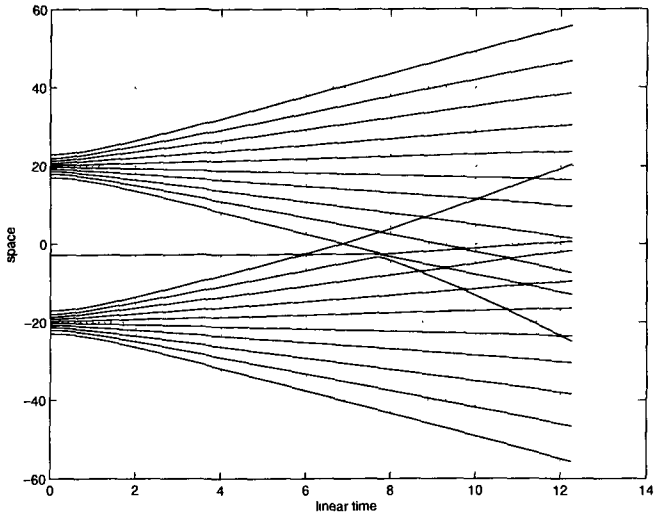


Figure 4.14: Avoided crossings between two Gaussians using sharply defined interpolation kernels

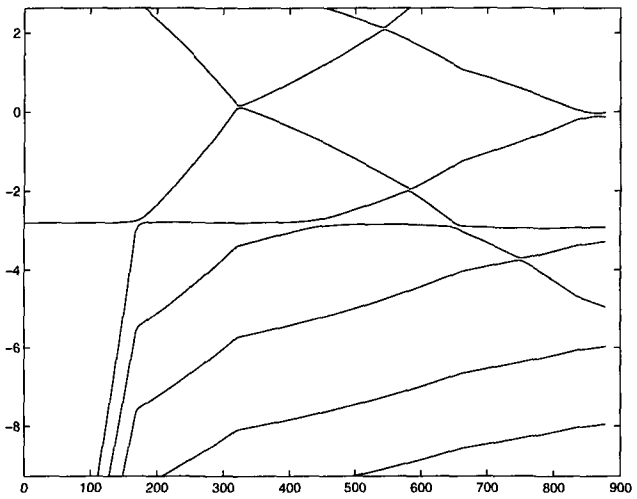


Figure 4.15: Trajectory detail of Fig. 4.14

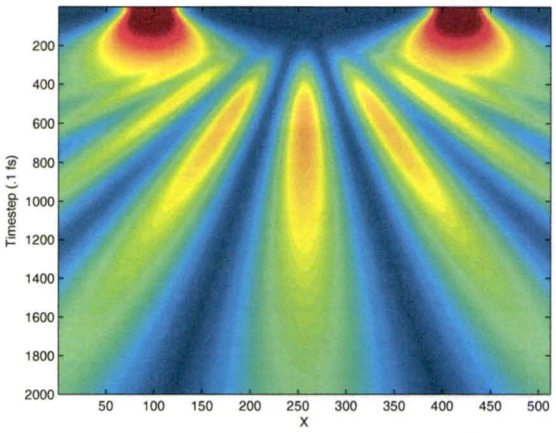


Figure 4.16: Wavefunction of two compact Gaussian packets interfering

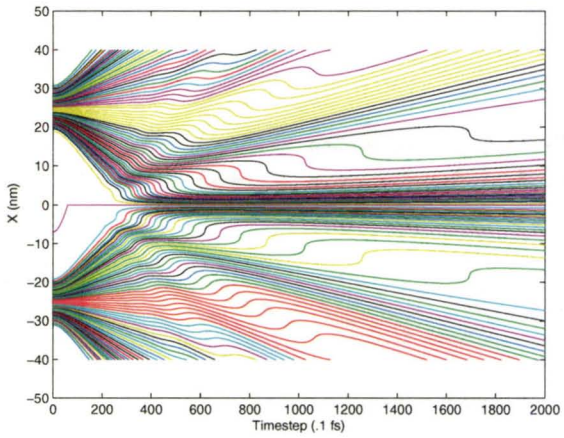


Figure 4.17: Trajectories corresponding to Fig. 4.16
(In this figure the space and time axes have been interchanged as compared to Fig. 4.16)

obtain the proper trajectories, we first calculate the wavefunction using the techniques of Sec. 3.3. In Fig. 4.16 we can see regions of compression (in red) and expansion (in blue) typical of interference patterns emerging as time passes.

In the frames of Fig. 4.19 we can readily see one reason why it may be difficult to capture this behaviour using the traditional particle treatment: insufficient sampling.

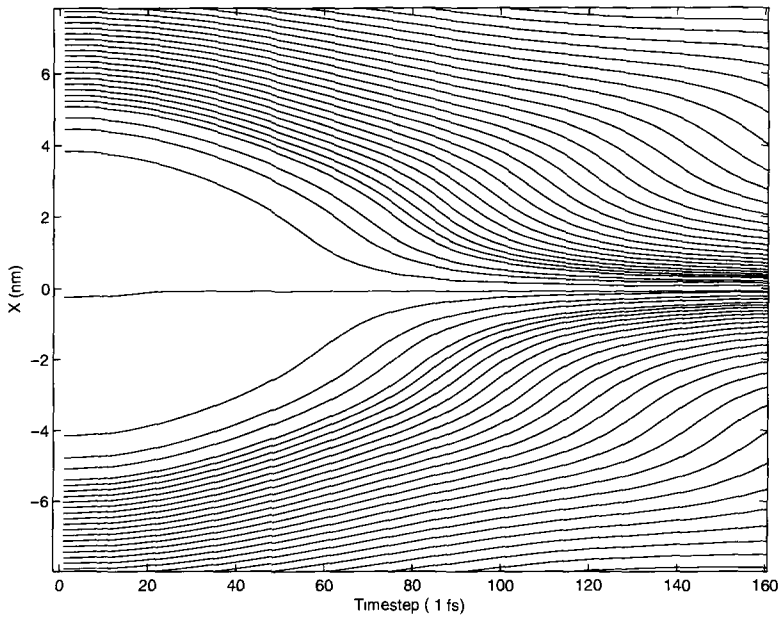


Figure 4.18: Trajectory detail of interference between two extended Gaussians

The interference emerges from a zone of negligible density where there are very few particles encoding for the log-density field. In a density weighted particle distribution for this case, the majority of the particles will initially be far away from the zone where interference develops and so will not be able to account for it, some resampling of that area will be in order if we want to capture that information earlier in the simulation.

4.3.1 Dynamic Resampling

It is that little blip near $x = 0$ in the quantum potential of Fig. 4.19 and the associated discontinuity in the velocity fields of Fig. 4.20 that are going to self feed in time and grow into the interference patterns of the density distribution.

There is no mystery as to how a trajectory on the far right "knows" that there is a trajectory on the far left, that information is carried by the quantum potential and it has its origin in that small bump in Q between the two Gaussians.

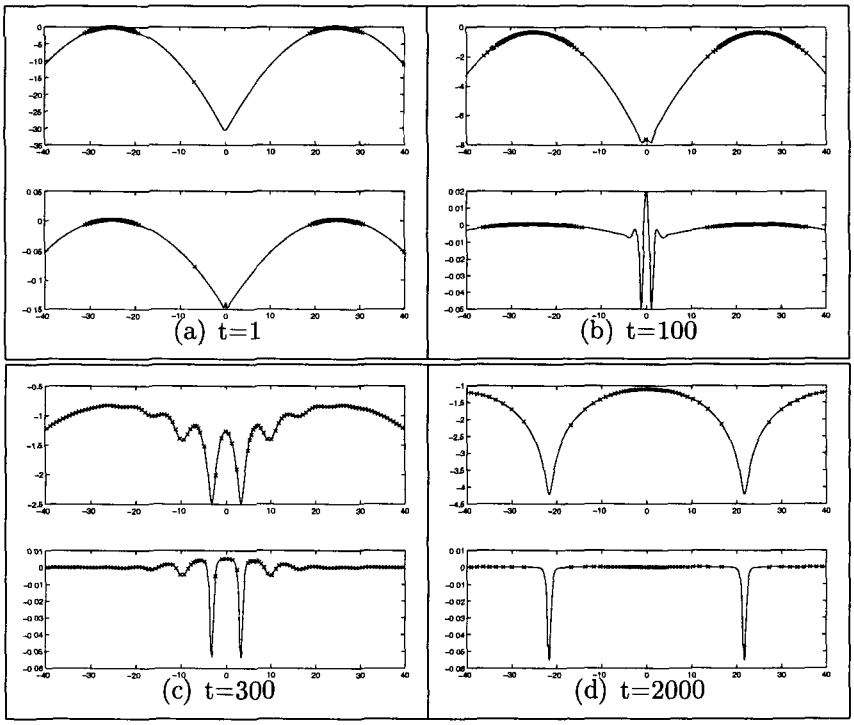


Figure 4.19: Evolution of the log-density (upper graphs) and quantum potential Q (lower graphs) at selected times.
On each panel, representing a particular timestep, the log density and corresponding Q are drawn versus space

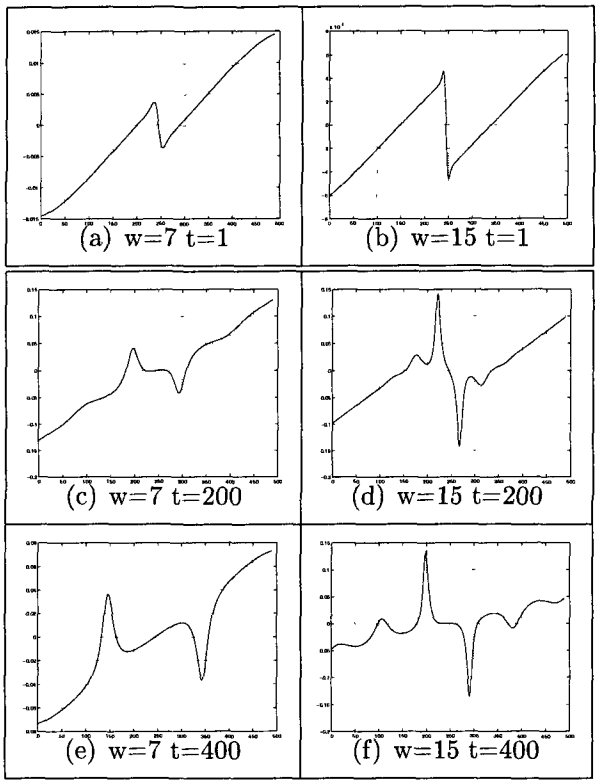
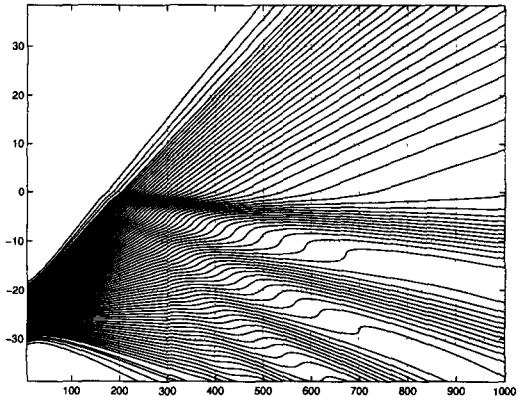


Figure 4.20: Evolution of the velocity profiles for two Gaussian packets with different separations (x vs v)
(Left panels: the Gaussians are initially centered at ± 7, and right panels, ± 15 units of distance)



(a) linear

Figure 4.21: Trajectory compression and expansion in a barrier scattering problem

How can we hope to capture this behaviour? Besides the obvious importance of having a physically acceptable interpolation method, one possibility is that we must change the premise that we need only sample regions of space where the probability density is significant, as we have already seen that the quantum potential does not depend on absolute densities.

Instead we should resample regions where the inter-particle distance is contracting and where that distance is above a certain threshold (to avoid oversampling) in order to properly capture the highly variable gradients in Q and S or v that will develop in the region. The reason for the distance criterion is related to the Shannon sampling theorem in that, if we have interparticle spacing of Δx, we can only hope to obtain momentum information of order $\leq \frac{1}{\Delta x}$. Since we will have high gradients in the interference area, we should decrease the average Δx in this region.

4.3.2 Dynamic Timescale

In order to estimate the error in our calculation at the end of each time step, we compare the results (as measured by the particle positions or more accurately by the

gradients of a field like S or C) with the errors obtained by taking two half steps. This way we can monitor the error and maintain it below some predetermined threshold by making dynamical adjustments to the time step. At the end of the process we get to keep the more accurate two half step result so the extra calculations involved are not completely wasted.

In our particular calculations we are cutting the time step in half in case of too large an error and increasing it by 1% if the error is acceptable, this enables the time step to "recover" from very low values in regions where such is possible, as illustrated in Fig. 4.22.

In the case of a Gaussian packet in a parabolic potential, we get a decreasing time step as the packet approaches the bottom of the potential where it has maximum momentum and highest density of trajectories (and therefore where we need a smaller time step).

Following this point of maximum trajectory density, circa step 750 in Fig. 4.22, the time step gradually recovers to larger values accelerating the simulation.

Furthermore if a more serious error condition is detected such as a trajectory crossing (as opposed to the error simply exceeding some threshold), the time step is again cut in half, and in addition we restore the system to the state of some predetermined number of time steps in the past, to enable it to avoid an evolution to the serious error condition.

In Fig. 4.24 we can see the reason why we have to go back a couple of time steps in order to correct a trajectory crossing. It is that usually, by the time it becomes apparent that the positions of the particles are erroneous, the background fields (such as Q and S) have been compromised for quite a while. Indeed some interpolation methods (i.e. splines) are prone to develop nasty oscillatory perturbations by the

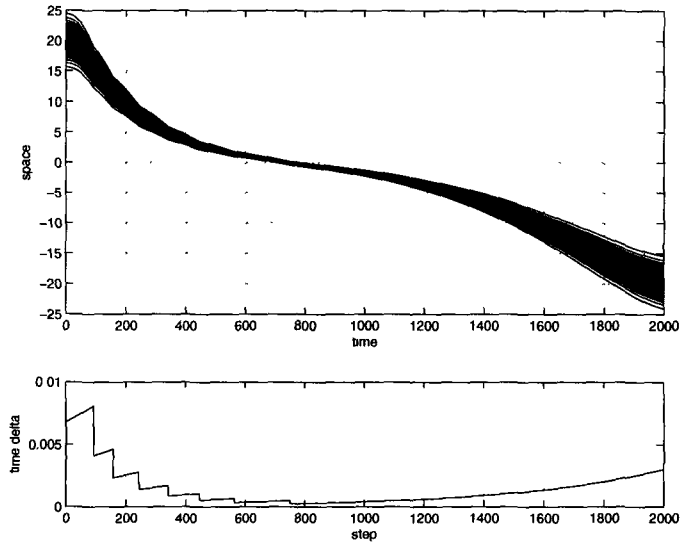


Figure 4.22: Dynamic timescale, abrupt declines represent time step adjustments caused by error conditions

Top:Space-time graph
Bottom:Time step versus time

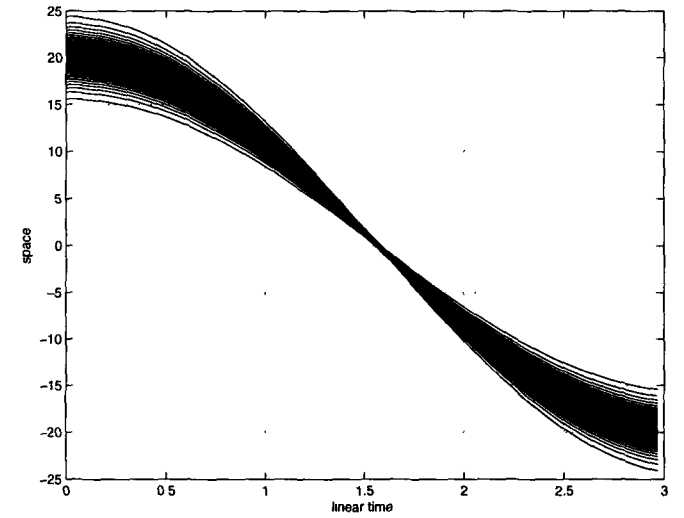


Figure 4.23: Linear time scale
(Using the recorded time steps from Fig. 4.22 we can convert the space-time graph back to a linear time scale where the Harmonic oscillator trajectories are more easily recognized)

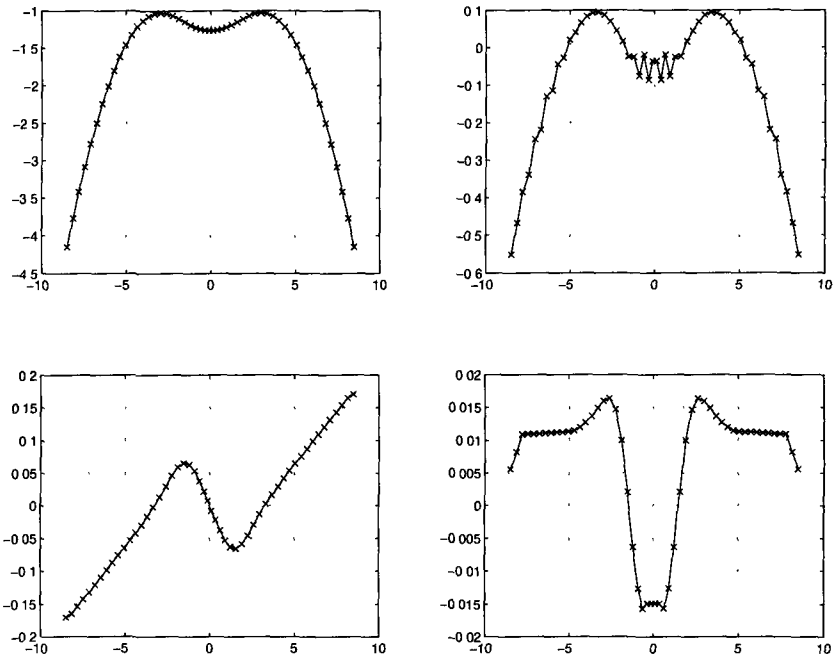


Figure 4.24: Perturbation in secondary fields not yet apparent in the log probability Clockwise from top left: log probability C , Q , $\nabla^2 S$ and ∇S

edges that will self-feed after many time steps and give rise to trajectory crossings. Because of this lag in cause and effect, we must use different monitoring functions for different methods.

Areas that are troublesome when evaluating derivatives with splines, are the compression areas, where the trajectories are coming closer together and thus the density is increasing leading to higher numerical error in the differentiation. These areas can be identified by $\nabla^2 S < 0$ or equivalently $\nabla v < 0$, that area is readily visible in the right bottom panel of Fig. 4.24.

A useful monitoring function for problematic zones that does a good job of catching unstable conditions early on, is the time evolution of the quantum potential,

$$\dot{Q} = \frac{1}{4}\nabla^4 S + \frac{1}{2}\nabla C \nabla^3 S$$

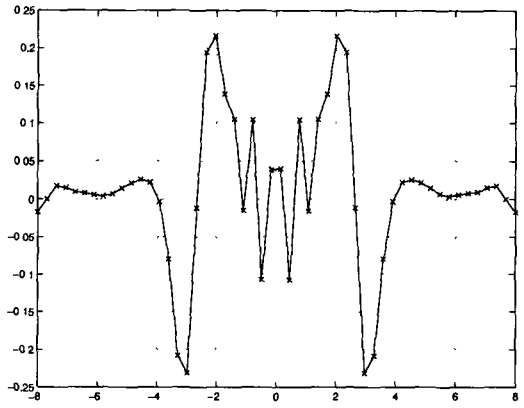


Figure 4.25: High frequency oscillations in $\nabla^4 S$ for the system of Fig. 4.24

which we can for error monitoring purposes approximate by its more problematic first term $\nabla^4 S$ that is represented in Fig. 4.25. By comparing all fields represented in that figure and the previous one, we can see an illustration of the lag in the propagation of instabilities as these are quite evident in $\nabla^4 S$ as high frequency spacial oscillations and are already somewhat developed in Q of Fig. 4.24 but they are not yet recognizable in C at that particular time step.

Quality of Fit An added bonus of using a dynamical timescale is that we are given a measure of the quality of the simulation at each point in time. The dynamical time step tends to be very small in regions where the system is encountering errors and when the time evolution is relatively error free it will increase as in Fig. 4.22.

Furthermore, we can make use of the fact that the Ehrenfest theorem should hold, as this is a quantum system, so we can take the average of observables such as position and momentum and compare their values to those expected in a purely classical system, according to eq. 2.8. At each time step we compute the average position $< x >$ of the trajectories we are following, and compare them to those of the classical trajectories arising from the mean initial values.

Chapter 5

Validation, Test Cases

To check the validity of the trajectories and to test the performance of the various interpolation methods we start by choosing a couple of (the very few) systems for which an analytical solution is known. We then apply the same tests to a propagating problem with interference similar to that of Sec. 4.3, that is known to have long time stability issues in Bohmian calculations.

5.1 Comparisons of Accuracy and Performance with Standard Techniques

The following tests were done on an Intel Dual Core Duo 3GHz machine with 4Gb RAM under MATLAB in Linux, we used single threaded code for our tests. The tests consisted of measuring the running time and flagging error conditions (if any) for wavepacket propagation in certain standard quantum potentials using our Bohmian code.

It is our experience that using the quantum Hamilton-Jacobi equations (QHJE) of

motion of Sec. 3.4.3 as opposed to the Newtonian code of 3.4.2, results in both faster execution and more stable numerics. The reason for that is probably linked with the higher order of numerical differentiations that the usage of P and the algorithms of Sec. 3.4.2 carry and the fact that ∇Q is not calculated directly but instead Q is combined with the kinetic energy and the classical potential in $\frac{dS}{dt}$.

Therefore in the following calculations, intended to compare interpolation routines, we will be using the QHJE code, modified in the following way:
So that comparisons are easier to make we use a fixed time step ($dt = 5 \times 10^{-3}$) for all trials. This fixed time step, which we get by turning off the dynamic time step machinery, will cause trajectory crossings to be more frequent than would normally happen, but the trajectory crossings will still depend on the quality of the interpolation methods used.

When the run is successful we compare the end wavepacket with the one obtained using the split operator method (see sec. 3.3). If on the other hand error conditions do occur a note is taken of the time step when that happened, giving us a rough measure of which interpolation used is more susceptible to numerical instabilities.

5.1.1 Table Description

A brief explanation of the parameters seen in the following tables 5.1, 5.2 and 5.3 follow.

The column entitled CPU(s) is self explanatory and it is simply the time in seconds that it takes to run the code. Note that unlike the Moyer code, no optimization was attempted on the Bohmian code and visualization and debugging code is run inside the main calculating loop. For our purposes of comparing interpolation methods this is not an issue though since they all run the same unoptimized code.

T_{stop} is the time step at which execution is interrupted, either because we have reached our predetermined maximum number of time steps or because an error condition such as trajectory crossing is encountered, that information is included in the next column "Traj".

In the next two columns estimates of the error of the final wavefunction are given as compared to the analytic solution of the same problem or to the Moyer solution (it was not necessary to use a numerical solution for the first two tables as analytic solutions exist for those potentials but we always compute the Moyer solution in all cases).

The error is the difference between our solution and the wavebased solution in the spatial domain common to both. In $< (\rho - \rho_m) >$ we have the error in the density while in $< (C - C_m) >$the error in the log density ($\times \frac{1}{2}$) is represented, these are presented as an average value $\pm$ the standard deviation $\sigma_\rho = \sqrt{\frac{(\rho-\rho_m)^2}{N-1}}$ or $\sigma_C = \sqrt{\frac{(C-C_m)^2}{N-1}}$.

Finally the last column gives us an estimate of the relative speed of all the methods in time steps per second of CPU time.

The Moyer grid method with a parameter 2048 $\times$ 2000 means that it was run on a 2048 point space grid for 2000 time steps .

P is the number of trajectories or particles used and in the polynomial method D stands for the polynomial degree used. At each point we find the D nearest neighbours and fit a polynomial to them to get estimates of the derivatives (for instance if D=2 we get the usual central difference formula for the curvature).

Part of the least square method is an extra parameter N, the number of nearest neighbours considered at each point. In the case N>D+1, the higher the N the more averaging occurs in the least squares estimate.

In the sines method, the LPFilter parameter represents the percentage of low pass

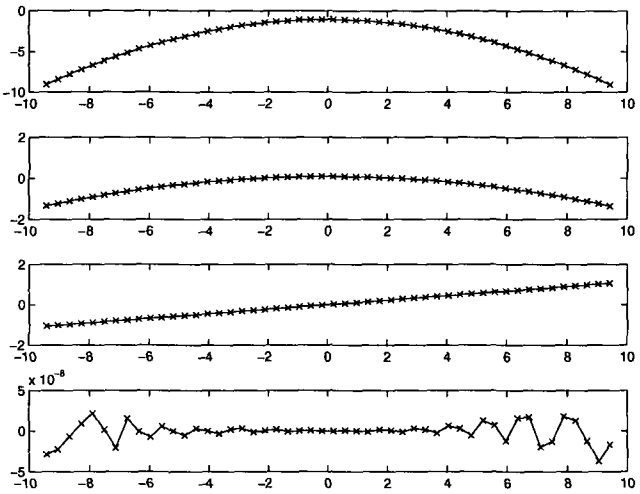


Figure 5.1: Sample run for the free wavepacket
Top to bottom panels are: C, Q, ∇S, $\nabla^4 S$

filtering done in momentum space as discussed in Sec. 4.2.2.

In the smoothed splines method B represents the number of points considered at left or right "edges", and smoothed according to the procedure of Sec. 4.2.4.

5.1.2 Free Wavepacket Propagation

This is the simplest case where there is no external potential and only one initial Gaussian packet exists. No interference effects are expected.

The free Gaussian wavepacket evolves ([15]) in time according to :

$\Psi(x,t) = (2\pi s_t)^{-\frac{1}{4}} \exp(-\frac{(x-x_o)^2}{4\sigma_0 s_t})$ with $s_t = \sigma_0(1 + i\frac{t}{2m\sigma_0})$

All interpolation methods deal quite well with this very simplest of cases of a single decaying Gaussian. The one exception is when we increase the number of points in the polynomial interpolation to 32; there the simulation doesn't quite make it to 500 timesteps and there is a trajectory crossing at 469. This is probably due to the high order of polynomial used causing oscillations as pictured in the Runge phenomenon in Fig. 3.7.

Description/Parameters	CPU(s)	T_{stop}	Traj	$< \rho - \rho_m > 10^{-5}$	steps/s
Moyer grid 2048x2000	49	2000	N/A	0	40.8
Poly P=50 D=2	83	500	Stable	-1.54 ± 2.3	6.02
Poly P=50 D=5	137	500	Stable	-1.54 ± 2.3	3.65
Poly P=50 D=6	141	500	Stable	-1.54 ± 2.3	3.55
Poly P=50 D=8	142	500	Stable	-1.54 ± 2.3	3.52
Poly P=50 D=16	145	500	Stable	-1.54 ± 2.3	3.45
Poly P=50 D=32	161	469	Cross	N/A	3.11
LeastSq P=50 D=2 N=6	85	500	Stable	-1.54 ± 2.3	5.88
LeastSq P=50 D=6 N=8	139	500	Stable	-1.54 ± 2.3	3.60
Sines P=50 LPFilter=10%	16.4	500	Stable	-1.54 ± 2.3	30.5
Gaussians P=50	23.3	500	Stable	-1.54 ± 2.3	21.5
Splines P=50	16.8	500	Stable	-1.54 ± 2.3	30.5
SmoothSplines P=50 B=12	21	500	Stable	-1.54 ± 2.3	23.8

Table 5.1: Free Gaussian packet evolution

One can also observe where the different methods stand in terms of performance relative to each other. We see that in general the polynomial and least squares methods are much slower than the rest. In the polynomial or least squares methods, the routine has to consider each point, determine a local neighbourhood of size N, and then construct an approximation in that neighbourhood. In other methods (such as splines) we first get a global approximation to the log density and then we get updated values to every trajectory at the same time. In other words we do operations to the whole vector of positions instead of each point separately, with performance gains. This is particularly true as the number of trajectories increases.

5.1.3 Harmonic Oscillator

Another analytically solvable system, the harmonic oscillator's time evolution alternates between periods of compression and expansion as illustrated in Fig. 5.2. Here the potential used is a weak quadratic centered around $x = 0$, $V(x) = 5 \times 10^{-2}(x)^2$.

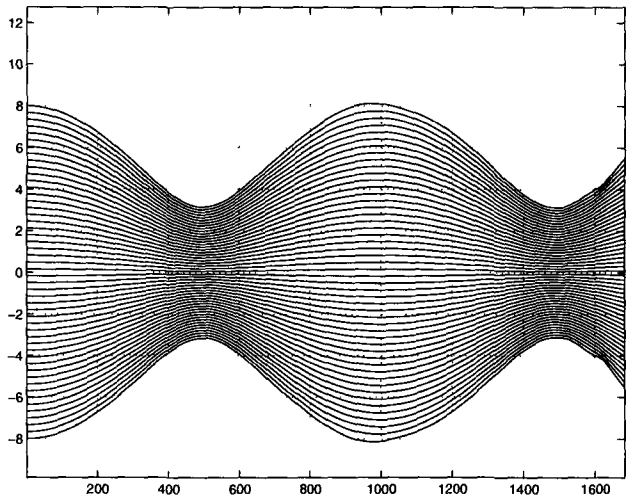


Figure 5.2: Sample run in an harmonic oscillator potential

Fig. 5.2 looks somewhat different from Fig. 4.23 because we are using an uniform distribution of initial points as opposed to distributed as $\frac{1}{N}$ in Fig. 4.23.

Description/Parameters	CPU(s)	T_{stop}	Traj	$< \rho - \rho_m > 10^{-5}$	steps/s
Moyer grid 2048x2000	49	2000	N/A	0	40.8
Poly P=50 D=2	173	1000	Stable	-9.73 ± 539	5.78
Poly P=50 D=3	170	1000	Stable	-9.73 ± 539	5.88
Poly P=50 D=6	295	1000	Stable	-9.73 ± 539	3.39
Poly P=50 D=10	295	1000	Stable	-9.73 ± 539	3.39
LeastSq P=50 D=2 N=6	164	1000	Stable	-9.73 ± 539	6.10
LeastSq P=50 D=6 N=8	277	1000	Stable	290 ± 5300	3.61
Sines P=50 LPFilter=10%	31	1000	Stable	5400 ± 15200	32.2
Gaussians P=50	45	1000	Stable	-13.7 ± 41	22.2
Splines P=50	33	1000	Stable	-1990 ± 4610	30.5
SmoothSplines P=50 B=12	40.8	1000	Stable	-9.73 ± 539	24.5
SmoothSplines P=50 B=12	361	10000	Stable	-182 ± 820	27.7

Table 5.2: Gaussian packet evolution in an harmonic potential

5.1.4 Interference of Two Gaussians

Here we apply the same tests to a system that will develop interference patterns similar to those that we discussed in Sec. 4.3. We have two Gaussians that are placed side by side and let them interfere together with no external potential applied. The evolution of the system is represented in Fig. 5.3 which was taken from a run of the smoothed splines method (last item on table 5.3).

Description/Parameters	CPU(s)	T_{stop}	Traj	$< \rho - \rho_m > 10^{-3}$	steps/s
Moyer grid=2048x2000	49	2000	N/A	0	40.8
Poly P=50 D=2	132	862	Cross	N/A	6.5
Poly P=50 D=3	135	840	Cross	N/A	6.2
Poly P=50 D=4	212	814	Cross	N/A	3.8
Poly P=50 D=5	211	737	Cross	N/A	3.5
Poly P=50 D=6	N/A	736	Cross	N/A	N/A
Poly P=50 D=10	N/A	652	Cross	N/A	N/A
Least Sq P=50 D=2 N=4	70	416	Cross	N/A	5.9
Least Sq P=50 D=2 N=5	108	640	Cross	N/A	5.9
Least Sq P=50 D=2 N=6	110	686	Cross	N/A	6.2
Sines P=50 LPFilter=10%	17.5	472	Cross	N/A	27
Gaussians P=50	N/A	332	Cross	N/A	N/A
Splines P=50	N/A	629	Cross	N/A	N/A
SmoothedSplines B=12	75	2000	Stable	-0.5 ± 1.5	26.7
SmoothedSplines B=12	392	10000	Stable	-1.3 ± 2.8	25.5

Table 5.3: Two Gaussian packets interfering

We can see from table 5.3 that most methods lead to trajectory crossings. We remind the reader that we have turned off the dynamic time scale control mechanism. The objective here is not to avoid trajectory crossings, but to evaluate different interpolation methods, and so by looking at table 5.3, and specifically at which time step trajectory crossings were observed, we can say that a certain method is more stable than another. Of course, the proper stable choice here should be the smoothed

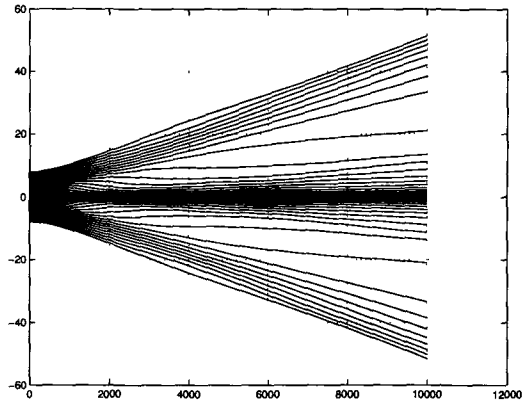


Figure 5.3: Trajectories $x(t)$ for smoothed spline calculation, 10,000 timesteps

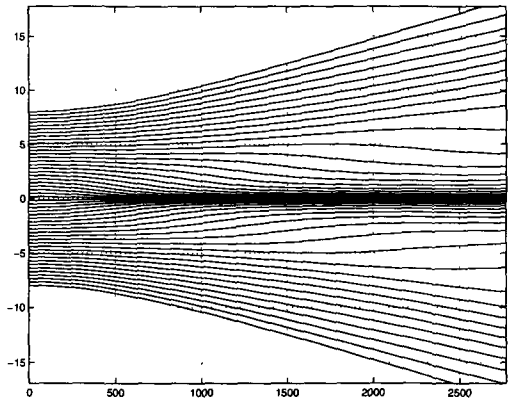


Figure 5.4: Detail of Fig. 5.3

splines method as it is the only one that doesn't result in crossed trajectories.

A surprising fact is that $D = 2$ poly, which is really nothing more than fitting parabolas at each point, (resulting in the familiar central differences formula) doesn't do too badly compared to other methods and outperforms the least square method and even the basic spline method (crosses at 862 vs 629).

5.2 Numerical Instabilities and an Alternate Criterion for Δt

As we saw in Sec. 4.3.2 and Figs. 4.24,4.25, by the time we notice that there is something wrong with the C or the S fields, usually the root cause of that problem already occured a few time steps previous, and a typical signature is high frequency oscillations in $\nabla^4 S$.

Oscillations suggest that we are integrating some trajectories too far in the direction of other particles, resulting in an excessive repulsive force and oscillations down the line. Why this tends to happen in problem areas of compression is linked to the fact that the absolute error of a derivative goes approximately as $\frac{2}{\delta x}$ times the error of the underlying field, so as particles are packed closer together and δx becomes smaller, so grows the error in the gradients estimated in that area. In fact for ∇^2 the error will be of order $4/(\delta x)^2$; therefore it is imperative, when using non-averaging (such as least square) interpolations, that we reduce the time intervals in lockstep with a minimum inter-particle distance to minimize integration errors .

Let's take a look at the QHJE equations of motion:

$$\begin{aligned} \dot{X} &= \nabla S \\ \dot{S} &= -\frac{1}{2}(\nabla S)^2 + \frac{1}{2}(\nabla^2 C + (\nabla C)^2) - V \\ \dot{C} &= -\frac{1}{2}\nabla^2 S \end{aligned}$$

where X is the position, S the phase and C the log-density. For a moment let's ignore any errors in the X field and concentrate in the fields $S \pm s$ and $C \pm c$ with absolute

errors s and c respectively.

In the case of simple numerical differentiation, every time we are evaluate ∇S we get an error of order $\pm 2s/\delta x$, for $\nabla^2 S$ the error is $\pm 4s/(\delta x)^2$ and for $(\nabla S)^2$ we get $\pm\left(4s/(\delta x)^2\right)\nabla S$. Similarly for C.

Let's see what happens in just one Euler integration over dt, of the equations of motion:

$$\begin{aligned} S_1 &= S_0 + dt\left(-\frac{1}{2}(\nabla S_0)^2 + \frac{1}{2}\nabla^2 C_0 + \frac{1}{2}(\nabla C_0)^2\right) \\ C_1 &= C_0 + dt\left(-\frac{1}{2}\nabla^2 S_0\right) \end{aligned}$$

If now we consider $S_0 = \bar{S}_0 \pm s_0$ and $C_0 = \bar{C}_0 \pm c_0$ where the bar denotes the noiseless fields, using the expressions for the errors in the gradients we have the error estimates:

$$\begin{aligned} s_1 &\approx s_0 + dt \times \frac{1}{2}\left(\left(\pm\frac{4}{\delta x}\nabla\bar{S}_0\right).s_0 \pm \frac{4}{\delta x^2}c_0 \pm \nabla C_0\frac{\bar{4}}{\delta x}c_0\right) \\ c_1 &\approx c_0 \pm dt \times \frac{1}{2}\left(\frac{4}{\delta x^2}s_0\right) \end{aligned}$$

For small δx, the δx^2 term will dominate in the first equation and we can further simplify:

$$\begin{aligned} s_1 &\approx s_0 \pm \frac{2dt}{(\delta x)^2}c_0 \\ c_1 &\approx c_0 \pm \frac{2dt}{(\delta x)^2}s_0 \end{aligned}$$

If we extend to further time steps, these equations define a recursive relation between two variables s and c for time step n:

$$\begin{aligned} s_n &\approx s_{n-1} \pm \frac{2dt_{n-1}}{(\delta x_{n-1})^2} c_{n-1} \\ c_n &\approx c_{n-1} \pm \frac{2dt_{n-1}}{(\delta x_{n-1})^2} s_{n-1} \end{aligned}$$

or with $\alpha_n = \pm \frac{2dt_{n-1}}{\delta x_{n-1}^2}$,

$$\begin{aligned} s_n &\approx s_{n-1} + \alpha_{n-1} c_{n-1} \\ c_n &\approx c_{n-1} + \alpha_{n-1} s_{n-1} \end{aligned}$$

To get a general expression for the errors we sum both equations,

$$s_n + c_n \approx (s_{n-1} + c_{n-1}) + \alpha_{n-1} (s_{n-1} + c_{n-1}) = (s_{n-1} + c_{n-1}) (1 + \alpha_{n-1})$$

So in general

$$s_n + c_n \approx (s_0 + c_0) \Pi_{n=1}^{N} (1 + \alpha_{n-1})$$

Now $\alpha_n = \pm \frac{2dt_n}{\delta x_n^2}$, so we want α_n as small as possible which implies an alternative criterion for Δt :

$$\Delta t \ll \frac{1}{2}(\delta x_{min})^2$$

Where δx_{min} is the minimum inter-particle distance. This will mean that sometimes the whole simulation will have to slow down by virtue of a small dt in order to integrate accurately zones of compression. If we are using a local interpolation method, this could probably be relaxed to integrating only the neighbourhood of small δx with a reduced timescale, in a sort of adiabatic approximation where the rest of the field would be "frozen", but we have not yet explored that possibility.

5.3 Conclusion on Interpolation Methods Tested

There is a delicate balance between the demands of having a stable estimation and having an accurate estimation of the gradients in our Bohmian calculations. As we saw in chapter 4 the accurate estimation is crucial, to make sure that particles avoid each other. However with this added sensitivity to local conditions also comes sensitivity to numerical noise and the possibility of inducing numerical oscillations in the solution. On the other hand, the low noise and stable estimation of derivatives afforded by an averaging method come at the cost of not being accurate enough when genuine local fluctuations do exist.

A compromise between these is present in the smoothing splines method where we can specify (sec. 4.2.4) zones of higher or less sharpness in the interpolation. (A more computationally costly but valid possibility is to have a least squares method with a locally adjustable number of fitting neighbours)

In any case, from the data in the previous tables, it seems reasonable to pick the smoothing splines method of Sec. 4.2.4 as it is the only method in our sample to deal

successfully with the interference of Fig. 5.3. Also, because the smoothed splines are calculated as a global approximation to all points, it is relatively fast compared to other interpolation methods, that require a local calculation in the neighbourhood of each point.

5.4 Other Existing Methods - Covering function, Ψ splitting, etc.

For completeness sake we should mention that besides variations on resampling and or interpolation methods, there are some very different numerical takes on the Bohmian interpretation that we did not explore. One such case is the derivative propagation method (see [32]). Another one is based on the Wigner function formulation (see [57]) . The arbitrary Lagrangian-Eulerian method of Wyatt and coworkers ([8]) is a way to deal with insufficient data points in zones of interest for the fields, conveniently letting us specify the grid that we wish to work with at each time step.

Finally as a way to deal with the problematic nodal zones, we mention the covering function method of Babyuk and Wyatt ([33]), where the wavefunction is split into two nodeless components (to be independently evolved), and in the work of Poirier and Trahan ([34, 35, 36]) where the wavefunction is represented by two counter-propagating traveling waves.

Chapter 6

2D and Higher Dimensions

Generalization of Bohmian methods to two and higher dimensions should be relatively straightforward. Unfortunately the same cannot be said of the wave based methods. When it comes to boundary conditions it is not so straightforward and even when it comes to integrating techniques they can look quite different in 1D, 2D and higher dimensions. In fact we do not know of an equivalent transparent boundary condition scheme in 2D to the one we used in Sec. 3.3.2 for 1D. Instead we revert to the imperfect technique of complex absorbing potentials as an approximation to TBCs.

6.1 Wave Based Calculations

As we did in Sec. 3, we begin with the wavefunction based methods to serve as a baseline for comparison to the trajectory methods.

6.1.1 ADI Method

To get the time evolution of the wavefunction we will use the ADI method (Alternating-Direction Implicit see [56]). This method is second order accurate both in space and time and it is an efficient way of solving a parabolic diffusion equation such as the Schrödinger equation.

For a general parabolic partial differential equation (assuming D to be constant):

$$\frac{\partial \psi}{\partial t} = D\frac{\partial^2 \psi}{\partial x^2}$$

the 2D ADI method in discretized form reads:

$$\begin{cases} \psi_{x,y}^{t+\frac{1}{2}} = \psi_{x,y}^{t} + D\frac{\delta t}{2\delta x^2}(\psi_{x+1,y}^{t+\frac{1}{2}} + \psi_{x-1,y}^{t+\frac{1}{2}} - 2\psi_{x,y}^{t+\frac{1}{2}} + \psi_{x,y+1}^{t} + \psi_{x,y-1}^{t} - \psi_{x,y}^{t}) \\ \psi_{x,y}^{t+1} = \psi_{x,y}^{t+\frac{1}{2}} + D\frac{\delta t}{2\delta x^2}(\psi_{x+1,y}^{t+\frac{1}{2}} + \psi_{x-1,y}^{t+\frac{1}{2}} - 2\psi_{x,y}^{t+\frac{1}{2}} + \psi_{x,y+1}^{t+1} + \psi_{x,y-1}^{t+1} - \psi_{x,y}^{t+1}) \end{cases} \tag{6.1}$$

The name ADI comes from the fact that we split the time evolution into two half steps. In the first we mix a forward and a backwards Euler scheme for the x and y directions and in the second half step those roles are reversed, thus earning the name alternating direction.

6.1.2 Boundary Conditions

As we mentioned in Sec. 6, in 2D and higher dimensions we are faced with the need to minimize unphysical reflections of the wavefunction from artificial domain walls. To achieve this we tried a couple of different approaches, with varying degrees of success.

We present in the next few sections the propagation of a free Gaussian in a box (as

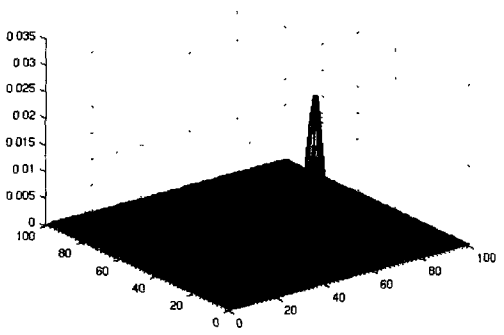


Figure 6.1: Initial wavepacket (t=0)

shown in Fig.6.1) with different boundary conditions: Hard wall (Fig. 6.2), periodic (Fig. 6.3) and complex absorbing potential (Figs. 6.5 6.4). The wavepacket is given some initial momentum in the y direction so that after a few time steps, interaction with the boundary can be more easily be seen.

Obviously the ideal result would be the analytical one where the wavepacket retains its Gaussian shape for all time and merely becomes more diffuse (larger σ) with time. While that does not happen for any of the pseudo transparent boundary condition methods tried, some do better than others in minimizing reflections and retaining the Gaussian shape, as we shall see.

Hard Wall Boundary Conditions

This is just the default type of boundary condition and we include it for comparison with the others. In this case we assume that the wavefunction suddenly encounters an infinite potential at the domain walls and thus necessarily vanishes on the boundaries. Unsurprisingly, as we can see in Fig. 6.2, reflections from the boundaries are quite obvious at t=800.

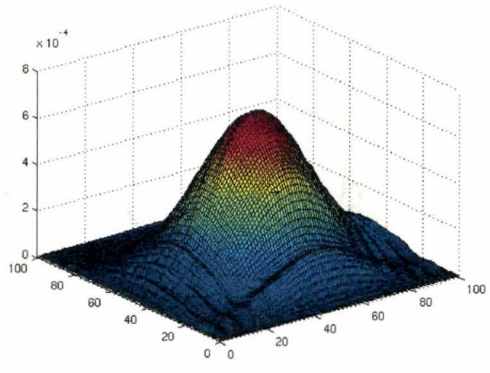
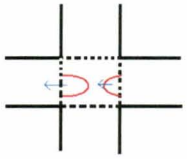
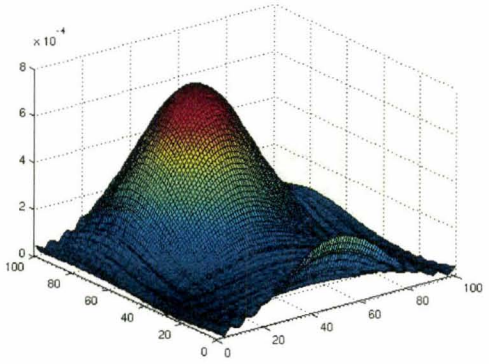


Figure 6.2: Hard Wall boundary conditions in 2D, at t=800



(a) Schematic of the boundary conditions



(b) t=800

Figure 6.3: Periodic boundary conditions in 2D, and the resulting wavefunction

Periodic Boundary Conditions

In this case the domain walls are assumed to be transparent and periodic. The wavefunction exits from one side, only to reappear on the opposite side domain wall. As we can see by comparing Fig. 6.3 to Fig. 6.2 at the same time step, the periodic system seems to have fewer interference fringes than the hard wall case, so it is marginally better suited to our purposes.

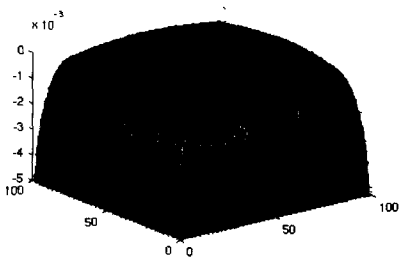


Figure 6.4: Profile of the absorbing imaginary potential

Complex Absorbing Potentials

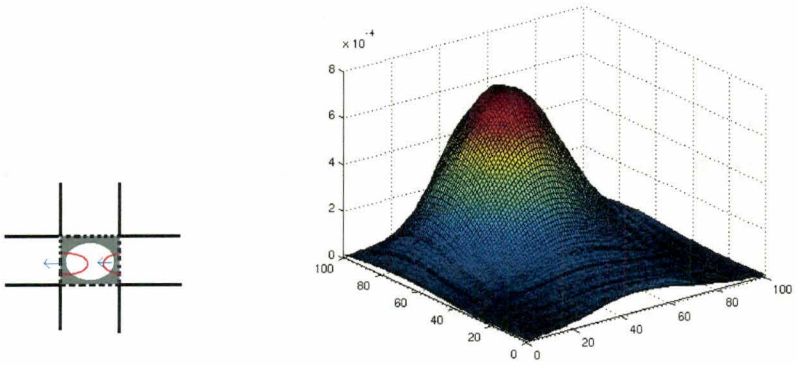
In this section, we combine the marginal benefits of the periodic boundary conditions with a more effective complex absorbing potential "coating" on the domain walls, as used by Muga *et al.* [37]. The complex part of the potential is pictured in Fig. 6.4. The top of the potential surface represents zero potential, while the sides of that surface are where absorption occurs.

This combination seems to be a more sensible approximation to 2D transparent boundary conditions as judged by the relative integrity of the Gaussian shape shown in Fig. 6.5 at the same time step, as compared to the previous figures. For longer simulation times we still get reflection and interference phenomena that eventually overcome the characteristic Gaussian shape of the original packet, because we are dealing with only approximations to 2D TBCs.

6.2 Trajectories and Phase Unwrapping in 2D

Recall that to calculate the trajectories in Sec. 3.3.3 we had to unwrap the 1D phase of the wavefunction. In 2D we have the same problem, except that now we have to monitor for jumps of 2π in any direction.

In Fig. 6.6 we display a simple case where the phase is increasing mainly in one



(a) schematic of boundary conditions

(b) t=800

Figure 6.5: Pseudo-Transparent boundary conditions (with an imaginary potential near the walls) and resulting wavefunction

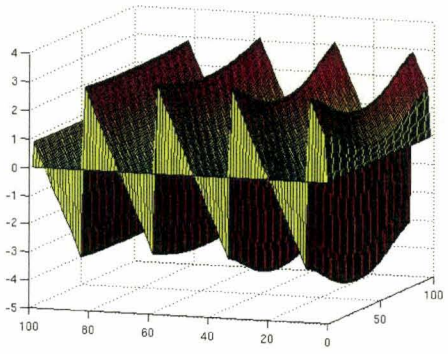
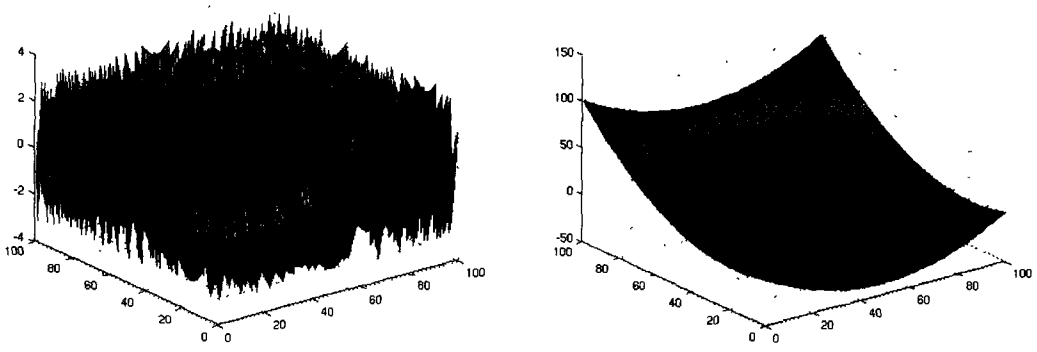


Figure 6.6: A simple phase angle problem in 2d



(a) original (b) unwrapped phase

Figure 6.7: A practical case of the phase problem and its solution

direction and in Fig. 6.7 a more realistic case is shown, as well as its resolution.

6.3 Bohmian Methods in 2D

The basic algorithms (Sec. 3.4.2 and 3.4.3) that we used to implement the Bohmian equations of motion remain the same. The only adjustments that we have to make are to account for the fact that we are now dealing with position and momentum vectors with N=2 coordinates instead of just scalars. Most interpolation techniques translate to 2-D and higher dimensions relatively straightforwardly.

6.3.1 Instance of Generalizing from 1D to 2D: Least Squares Method

As a simple example of how these methods transpose to higher dimensions, we show the time evolution of an initially resting Gaussian wavepacket in Fig. 6.8. Here we use the least squares method to calculate Q from the density of the 25 particle cloud. Note that given this particular visualization method of projecting

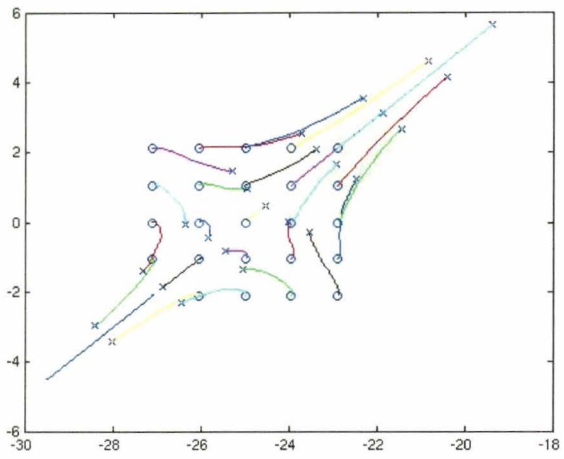


Figure 6.8: 2D Bohmian evolution of an inhomogeneous packet as viewed from the z-axis

a cross-section of what really is a 3D surface (2 space+1 time) into a 2D picture, the trajectories may appear to cross. In the actual 3D space the non-crossing of trajectories is of course still observed.

As one increases the number of particles the computational effort should scale linearly, independent of the dimensionality of the space in which we are working. This is in marked contrast to traditional wave based methods, where in going from 1D to 2D we square the number of grid points, and in general that number scales exponentially as N^D, leaving us with no choice but to use coarser grids when working in higher dimensions. Here lies one of the big advantages of Bohmian methods where the grid stays with the wavefunction.

Chapter 7

Cellular Automata and Lattice Boltzmann Methods

In this chapter we explore a possible connection between simulation of Bohmian mechanics and the world of cellular automata, of which lattice Boltzmann methods are a subset.

7.1 Cellular Automata

Since they were invented by von Neumann is the 1940s, cellular automata have been successfully used to simulate an enormous range of systems, including bacterial growth, forest wildfires, rush hour traffic in a busy city and classical fluid dynamics [39, 40]. These cellular automata operate according to a tantalizingly simple set of rules (that is supposed to capture the microphysics of the process), from which the macroscopically complex behaviour of the system gradually emerges. It is conceivable that by using a proper set of rules we may be able to reproduce the behaviour of a quantum fluid in the Bohmian sense, thus approaching the problem from a completely

different viewpoint.

Recently re-popularized by Steven Wolfram [40], Cellular Automata or CA have been with us for decades. As we mentioned they are particularly well suited to model complex and chaotic systems as well as more "classical" problems. This versatility should not surprise us since they have been demonstrated [42] to be formally equivalent to a Turing machine, so theoretically they are capable of carrying out any well posed calculation. To emphasize this point we should note that the calculations of previous chapters were done on Turing machines (standard computers).

The question remains, are there are advantages to using cellular automata in the context of time evolution of a quantum system, as opposed to more conventional techniques, and in particular can we marry the framework of Bohmian mechanics to the gear of cellular automata? That is what we explore in this section.

7.1.1 Conway's Game of Life

There are many and interesting cellular automata implementations that have been studied ([40]). The reader is probably already familiar with at least one of these. In Conway's "Game of life", a simple set of rules gives rise, totally deterministically, to quite complex behaviour of its constituents, mimicking organisms that live, die and even self replicate on the computer screen. Incidentally a Turing machine has been constructed with Conway's game of life "parts"; it is described in [43].

7.2 Cellular Automata and Fluid Mechanics

7.2.1 Lattice Gas Cellular Automata (LGCA)

Historically, these models are the original catalysts for the usage of cellular automata in fluid simulation. Originally introduced in the 1970's by Hardy, Pomeaux and de Pazzis, these simulations consist of an underlying lattice on which two basic steps are performed to advance in time :

- A free streaming step, where particles move with whatever momentum has been assigned to them.
- A collision step, where particles collide with each other in momentum and particle number conserving exchanges.

Finally if at some point in time we wish to recover the physically relevant macroscopic fields, an averaging over many lattice sites is done.

Amazingly, together with a judicious choice of underlying lattice, this is all that it takes to make a respectable simulation of a simple fluid. Following are some historically famous choices for the lattice and the corresponding collision rules.

"HPP" LCGA

This is the original model by Hardy, Pomeaux and de Pazzis (HPP). Here, the chosen lattice is square, and the simple collision rules are illustrated in Fig. 7.1. This is mostly a toy model and the simple choice of lattice ends up having consequences in terms of the resultant flow not being isotropic. In fact this lattice gives rise to fluid vortices which are square instead of circular.

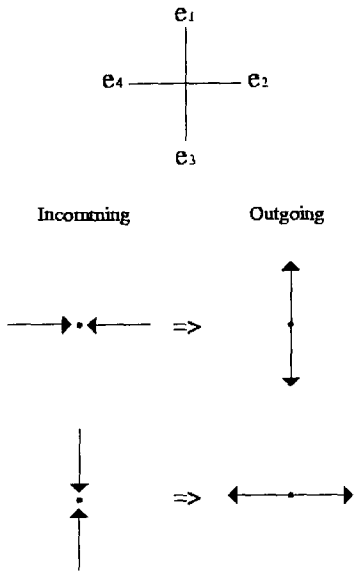


Figure 7.1: Lattice and collision rules for HPP model

"FHP" LGCA

Later introduced by Frish, Hasslacher and Poumeaux, this CA does not suffer from the anisotropic anomaly of the HPP model. This is thanks to using an hexagonal lattice and a correspondingly larger set of collision rules (see Fig. 7.2).

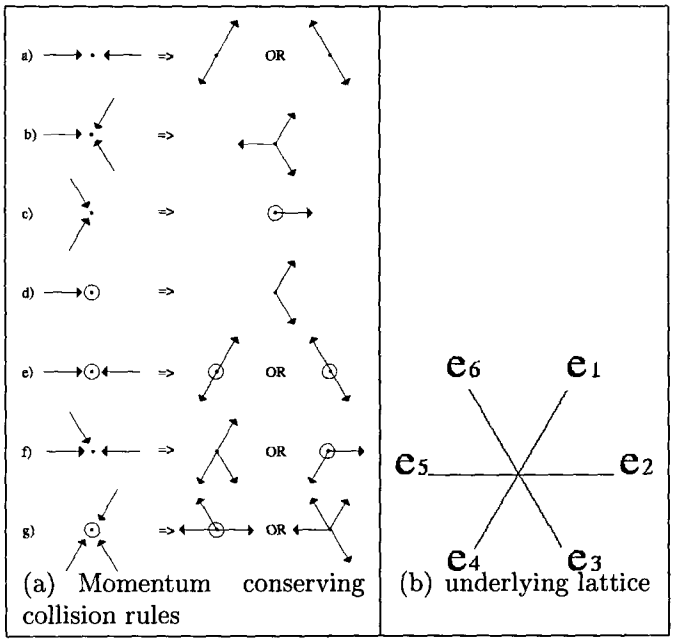


Figure 7.2: Lattice and collision rules for FHP model

Other Lattices

As you move to higher dimensions, the complexity of the lattices necessary to preserve isotropy quickly escalates and the number of collision rules that we need to track, rises exponentially.

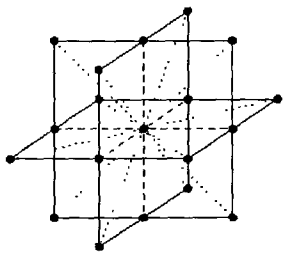


Figure 7.3: Example of 3D lattice for a LGCA model

The main advantage of LGCA methods for 2D is the small storage requirement and fast collision calculations (dealing with logical operations) given the Boolean nature of the collision operators. In the 1980's and 1990's a couple of machines were

specifically built to efficiently implement these methods in hardware. On the other hand modern computers are more and more engineered towards efficient floating point calculations.

7.3 Lattice Boltzmann Methods

One of the disadvantages of the previously mentioned LGCA methods is their high statistical noise, a consequence of their discrete nature whereby a lattice direction either is or is not occupied. A natural evolution from these ideas is presented in the Lattice Boltzmann method where one replaces the Boolean particle number by its average value resulting in a continuous density distribution function $f(\vec{r}, \vec{v}, t)$, defined in phase space . The evolution of f is governed by the Boltzmann equation

$$\partial_t f + v_x \partial_x f + F_x \partial_{v_x} f = \Omega(f) \tag{7.1}$$

Here v_x the velocity and F_x the force in the x direction. The collision operator Ω can be a very complex function that needs to be approximated. Approximation of Ω by $A_{ij}(f_j - f_j^{eq})$ constitutes the quasi-linear LBE method. Going one step further we get the fully linear approximation, the BGK (Bhatnagar,Gross, Krook [58]) collision operator:$\Omega = \frac{1}{\tau}(f_0 - f)$.

7.4 Cellular Automata and Quantum Mechanics

Now that we have established a link between cellular automata, lattice Boltzmann and fluid mechanics, we use Succi ([50]) as a guide to illustrate the link in one dimension between the Dirac equation and lattice Boltzmann methods.

7.4.1 Dirac Equation

In 3D the usual way of writting the Dirac equation is,

$$i\partial_t \Psi = (\alpha \cdot \nabla + m \cdot \beta)\Psi$$

with $\alpha^{x,y,z} = \begin{bmatrix} \sigma_{x,y,z} & 0 \\ 0 & -\sigma_{x,y,z} \end{bmatrix}$, $\beta = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$ and where $\sigma_{x,y,z}$ are the usual Pauli matrices:

$$\sigma_{x,y,z} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

7.4.2 Majorana Representation

In the Majorana representation, where $\Psi_j = (\alpha^y_{jk} + i\beta_{jk})\Psi_k$, the Dirac equation is represented by

$$(W^\mu_{jk}\partial_\mu)\Psi_k = iM_{jk}\Psi_k \tag{7.2}$$

here $\mu = 0, 1, 2, 3$ and $W^0_{jk} = \delta_{jk}$,$W^1 = \alpha^x$, $W^2 = \beta$, $W^3 = -\alpha^z$ (all real 4 by 4 matrices). The modified mass term $M_{jk} = -im\alpha^y + qV\delta_{jk}$ contains the rest mass and an external potential term. This form is reminiscent of a 4-phase fluid (the two Dirac spinors each with their 2 components) interacting via a scattering matrix M_{jk}.

Restricting ourselves to 1-D and taking $\Psi = \begin{bmatrix} u_\uparrow \\ u_\downarrow \\ d_\uparrow \\ d_\downarrow \end{bmatrix}$, where u and d represent the top and bottom Dirac spinors with components $u_1 = u_\uparrow$, $u_2 = u_\downarrow$, $d_1 = d_\uparrow$, $d_2 = d_\downarrow$,

eq. 7.2 becomes:

$$\begin{cases} \partial_t u_{1,2} - \partial_z u_{1,2} = m d_{2,1} + i g u_{2,1} \\ \partial_t d_{1,2} + \partial_z d_{1,2} = -m u_{2,1} + i g d_{1,2} \end{cases} \tag{7.3}$$

7.4.3 Formal Connection to LBE

Looking at eq. 7.3 and eq. 7.1, if we make the substitutions $f_i = d, u$, $v_i = \mp 1$, $f_1^e = \frac{i\omega_c}{g} f_2$, $f_2^e = -\frac{i\omega_c}{g} f_1$, we can make a formal mapping of the Dirac equation onto two Lattice Boltzmann equations, one for $f_i = d$ and another for $f_i = u$,

$$\partial_t f_i + v_i \partial_z f_i = ig(f_i - f_i^e)$$

That is to say that the quantum system is formally equivalent to a 1-D Lattice Boltzmann system with four channels (up and down spinors) that will intermix because f_1 depends on f_2 and vice versa .

7.5 Numerical Experiments

7.5.1 Algorithm

We use natural units as in Sec. 3, setting $\hbar = c = 1$. If we discretize the eqs. 7.3 in a Crank-Nicolson scheme we obtain:

$$\begin{cases} \hat{u_{1,2}} - u_{1,2} = \frac{m}{2}(\hat{d_{1,2}} d_{2,1}) + i\frac{g}{2}(\hat{u_{2,1}} + u_{2,1}) \\ \hat{d_{1,2}} - d_{1,2} = -\frac{m}{2}(\hat{u_{2,1}} + u_{2,1}) + i\frac{g}{2}(\hat{d_{1,2}} + d_{1,2}) \end{cases} \tag{7.4}$$

Where the hat $\hat{}$ on a variable means that time has advanced by one step. This is an implicit scheme, as can be seen by the presence of advanced terms on both sides of

the equation. To make it explicit and refer only to known values, we solve the linear system for $\hat{u}$ and $\hat{d}$, giving us the following equations:

$$\begin{pmatrix} \hat{u} \\ \hat{d} \end{pmatrix} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \begin{pmatrix} u \\ d \end{pmatrix}$$

Where the coefficients a and b are given by:

$$\begin{cases} a = \frac{1-\Omega/4}{1+\Omega/4-ig} \\ b = \frac{m}{1+\Omega/4-ig} \end{cases}$$

with $\Omega = m^2 - g^2$.

This convenient linear system will be the basis of our lattice propagation and collision calculations. Note that if desired we can easily include non-linear and/or self-consistent terms by suitably changing the potential term g (i.e. if we want it to be a function of the density we can make it a function of $(u^2 + d^2)$). In that case the coefficients a and b will no longer be constant and will have to be recomputed at each time step.

7.5.2 Quantum Systems studied

To test the algorithm we are going to use similar systems to those of Sec. 5.1, whose analytical solutions are well known.

Free Packet

We start with a Gaussian packet centered in the middle of our array and given a certain momentum p towards the right. The fact that the Gaussian disperses in time

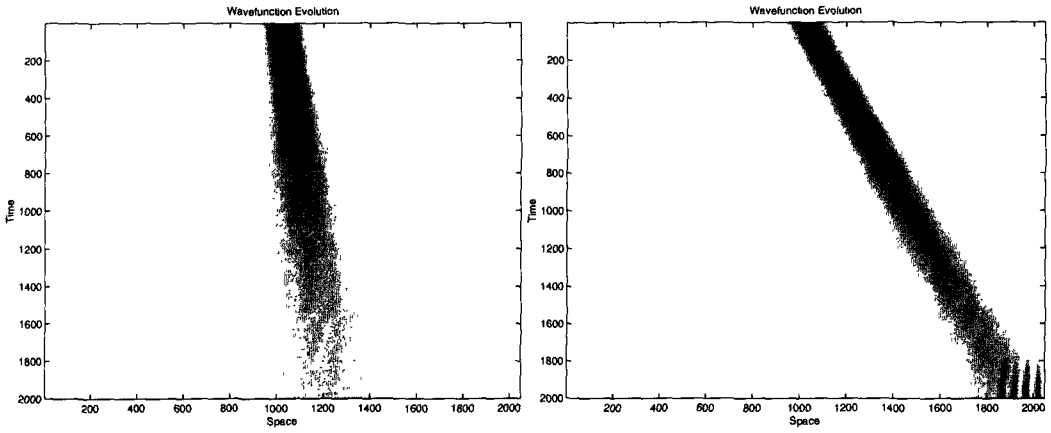


Figure 7.4: Free packet dispersion (2 different momenta)

is clearly visible in Fig. 7.4. How that dispersion relates to the expected analytical value is represented in Fig. 7.5. Also represented in the same figure is the average value for the position of the packet, which follows a path consistent with Ehrenfest's theorem. The expressions for both expection values are:

$$< x >= x_0 + \frac{p_0}{m}t$$

and

$$\sigma(t) = \sqrt{\sigma_0^2 + \frac{\hbar^2 t^2}{4m^2\sigma_0^2}}$$

Looking carefully at the graphs of Fig. 7.5 we note that as the wavepacket approaches the boundary our values deviate further from the theoretical values, as we start to observe some numerical reflection from the walls. This is particularly visible in the right panel of Fig. 7.4.

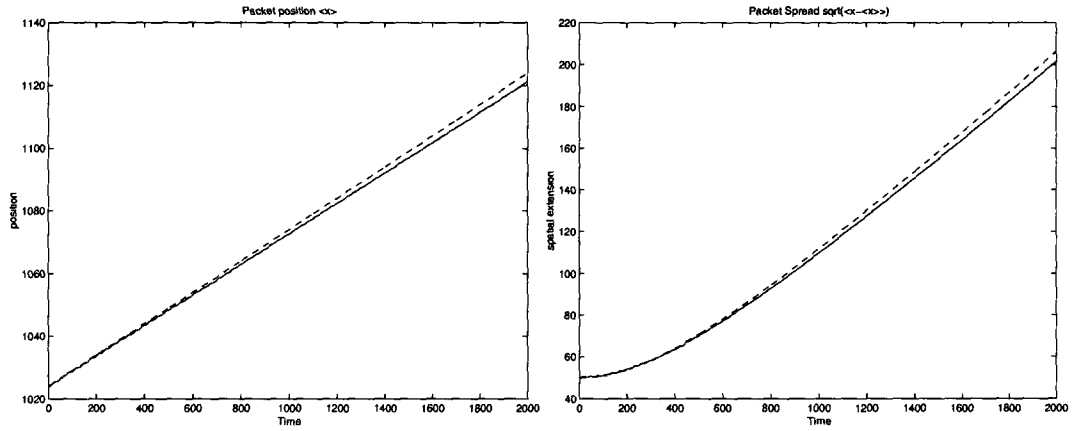


Figure 7.5: Mean position (left) and standard deviation (right) of the wavepacket vs time

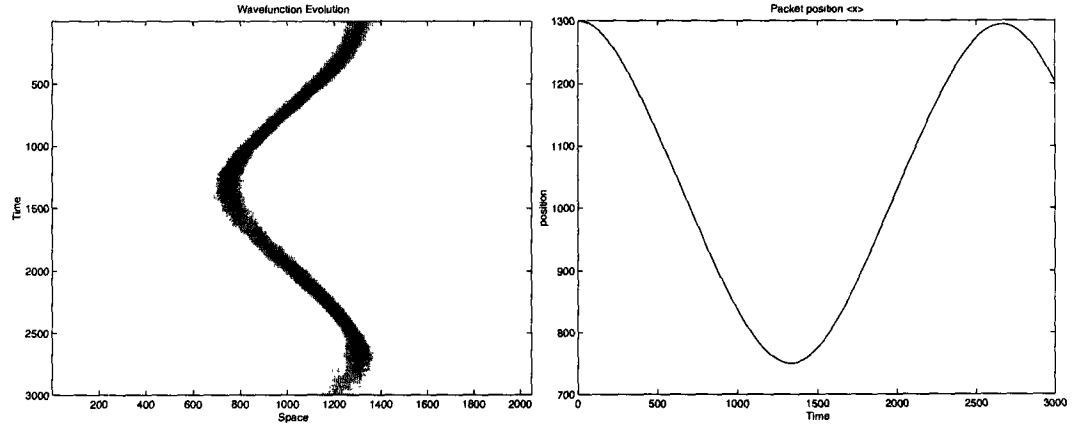


Figure 7.6: Wavepacket in an quadratic potential (left) and its average position (right)

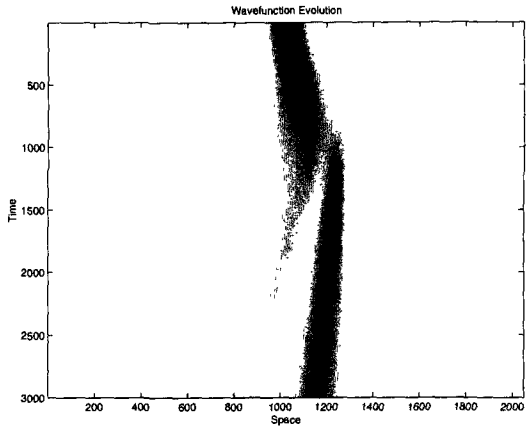


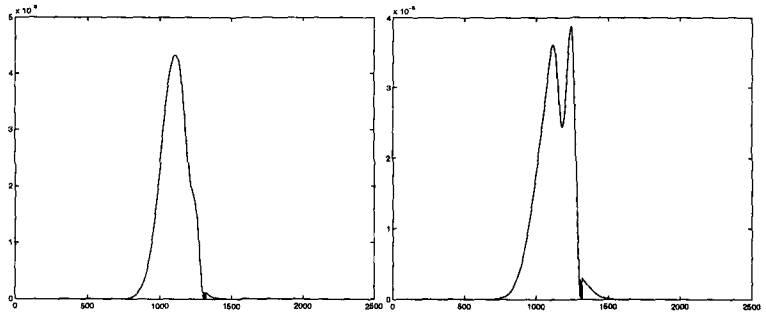
Figure 7.7: Wavepacket hitting a barrier

Harmonic Oscillator

In the next case we begin with a Gaussian packet with zero initial momentum placed in an harmonic oscillator potential. In Fig. 7.6 we see when plotting the mean position of the packet that there is some numerical dissipation present as the position is slightly smaller after one period.

Barrier Penetration

Finally in the last case, instead of having zero potential everywhere, we place a 20 unit long barrier at position 1300. In Fig. 7.8 intense interference can be seen at the edge of the "wall" and a weak evanescent wave continues along the interrupted path of the Gaussian packet, across the barrier to the left side of Fig. 7.7. Here we don't face the same problems with nodes or interference as in the Bohmian methods we considered before. The reason for this is that we are dealing more with a wave based method closer in its internals to our reference Moyer method of Sec. 3.3.



(a) t=800 (b) t=1100

Figure 7.8: Interference and exponential decay inside the potential barrier

7.6 Critique of the Majorama LBE Method

As we have seen the Majorama Lattice Boltzmann method seems to work, however as we just saw it really should belong to the class of wavefunction based methods such as the split operator method we used previously in 3.3. We still have hard domain walls and a fixed space-time grid, with all the associated limitations. What this method does do however, is give us a glimpse of a connection between cellular automata and quantum simulation, and in doing so it gives us a hint that an equivalent Bohmian based cellular automaton may also be devisable.

7.7 A Look Back at the Bohmian Picture

Looking back at the Bohmian equations of motion, with $Q = -\frac{\hbar^2}{2m}\frac{\nabla\sqrt{\rho}}{\sqrt{\rho}}$ and where $\frac{D}{Dt}$ represents the material derivative:

$$\begin{cases} \frac{\partial v}{\partial t} + \nabla(\rho v) = 0 \Leftrightarrow \frac{Dp}{Dt} + \rho\nabla v = 0 \\ \frac{\partial v}{\partial t} + (v\nabla)v = -\frac{1}{m}\nabla(V+Q) = \frac{Dv}{Dt} \end{cases} \tag{7.5}$$

And then looking at the equation for a classical fluid (Euler equation):

$$\frac{\partial v_i}{\partial t} + (v\nabla)v_i = -\frac{1}{m}\partial V_i - \frac{1}{m\rho}\partial_j \delta_{ij} P \tag{7.6}$$

We see that the first equation 7.5 expresses the conservation of probability, and the second maps term by term onto the classical fluid equation 7.6 except for the pressure term that must map to Q for the analogy to hold. In fact ([15]), if we substitute the pressure tensor P_{ij} by $\sigma_{ij} = -\frac{\hbar^2}{4m}\rho\partial_{ij}\log\rho$ plus any tensor whose divergence is zero, we get two formally equivalent equations.

So a Bohmian system is, at least formally, equivalent to an inviscid (i.e. time reversible and non-dissipative) irrotational and compressible fluid. When trying to extrapolate this fact to the lattice Boltzmann framework we will have two macroscopic real fields to work with, v the velocity and ρ the probability density, an analog of the fluid density.

7.7.1 Lattice Boltzmann Revisited

We reintroduce the Boltzmann formalism in a lattice space with N discretized velocity vectors $\vec{v_i}$. In the i direction, the Boltzmann equation in the BGK([58]) approximation reads:

$$f_i(\vec{r} + \tau\vec{v_i}, t + \tau) - f_i(\vec{r}, t) = \Omega_i(f) = \frac{1}{\xi}\left(f_i^0(\vec{r}, t) - f_i(\vec{r}, t)\right)$$

At each lattice site, these N discrete directions combine in the following way to reconstitute the real macroscopic fields:

$$\begin{cases} \rho = \sum_{i=1}^{N} m_i f_i \\ \rho\vec{v} = \sum_{i=1}^{N} m_i f_i \vec{v_i} \\ \text{higher order terms} \end{cases} \tag{7.7}$$

where the lattice specific weights m_i are determined in order to preserve isotropy. For instance, in the D2Q9 lattice, a two dimensional lattice with 9 velocity vectors, the diagonal terms will have to be weighted differently since they have different lengths.

Now f_0 can be quite complicated, but we are going to consider a Taylor expansion in the macroscopic fields v and ρ, thus establishing a connection between the macroscopic equilibrium and the collisional microphysics.

$$f_i^0 = a\rho + b\rho\frac{\vec{v_i}.\vec{u}}{v^2} + c\rho v_{i\alpha}v_{i\beta}u_\alpha u_\beta + ...$$

Here a, b and c are coefficients to be determined; ρ and v are the macroscopic fields, and u_i are the lattice vectors.

We can now insert this Taylor expanded form of the Boltzmann equilibrium function in the equations of eq. 7.7. The first equation implies that in a "collision" or "step" $\sum m_i f_i^0 = \rho$, which will give a lattice-dependent value for a (in the D2Q9 case $a = \frac{1}{9}$). Similarly the remaining coefficients of f_i^0 can be determined ([52]), given enough conservation relations as those of eq. 7.7.

7.7.2 Lattice Boltzmann Method and Bohmian Mechanics

In general, the Lattice Boltzmann method works for problems that can be set up in the form equations of continuity which are the result of one or more conservation laws([53]). For instance for some hypothetical macroscopic fields Y and Z, the conservation relations for the zeroth and first moments,

$$\begin{cases} \sum_\alpha f_\alpha^0 X = Y \\ \sum_\alpha f_\alpha^0 \hat{e_\alpha} X = Z \end{cases} \tag{7.8}$$

together with the space-time evolution given by the Boltzmann equation $f(r+\Delta t\hat{e_\alpha}, t+\Delta t) = \frac{1}{\tau} f_i^0 + (1-\frac{1}{\tau}) f_i^0$, will generate the macroscopic differential relation (of continuity):

$$\frac{\partial (Y)}{\partial t} + \nabla(Z) = 0 + \mathcal{O}(\epsilon) \tag{7.9}$$

So for instance if we want $\frac{\partial \rho}{\partial t} - \nabla(\rho u) = 0$ to apply, we can identify $Y = \rho$ and $Z = \rho u$, which are generated if we set $X = 1$, giving:

$$\begin{cases} \sum_\alpha f_\alpha^0 = \rho \\ \sum_\alpha f_\alpha^0 \hat{e_\alpha} = \rho u \end{cases}$$

these may now used to determine coefficients in our particular form for the function f_α^0, as shown in Sec. 7.7.1.

Bohmian Case

We need to work with the Bohmian equations of motion, in an Eulerian frame since the LBM lattice is fixed in space:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \rho u = 0 \\ \frac{\partial u}{\partial t} + u \nabla u = -\frac{1}{M} \nabla (V + Q) \\ Q = -\frac{\hbar^2}{2m} \frac{\nabla \sqrt{\rho}}{\sqrt{\rho}} \end{cases} \tag{7.10}$$

We already saw in the previous section how we can generate the first of these equations, the probability conservation relation by setting $X = 1$, $Y = \rho$ and $Z = \rho u$.

Now we wish to generate something of the form, with $Q = -\frac{\hbar^2}{2m}\frac{\nabla\sqrt{\rho}}{\sqrt{\rho}}$:

$$\frac{\partial u}{\partial t} + u\nabla u = -\frac{1}{M}\nabla(V+Q)$$

We certainly cannot use $\sum_\alpha f_\alpha^0 = u$ for this one since we already set it to ρ.

We can however set $X = \hat{e_\alpha}$ in eq. 7.8, generating:

$$\begin{cases} \sum_\alpha f_\alpha^0 \hat{e_\alpha} = \rho u \\ \sum_\alpha f_\alpha^0 \hat{e}_{\alpha i}\hat{e}_{\alpha j} = Z \end{cases} \tag{7.11}$$

We now see that $Y = \rho u$ this time and Z should be chosen so that the final equation coincides with the Bohmian momentum equation . This is a little bit tricky as we shall see. Looking at the equations 7.11 we have set $Y = \rho u$, so we want a term in $\frac{\partial \rho u}{\partial t} = u\frac{\partial \rho}{\partial t} + \rho\frac{\partial u}{\partial t}$. We can use the equation of continuity here, and obtain $\rho\frac{\partial u}{\partial t} = \frac{\partial \rho u}{\partial t} + u\nabla\rho u = \frac{\partial \rho u}{\partial t} + \nabla u\rho u - \rho u\nabla u$

Substituting back in eqs. 7.10 we get:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla\rho u = 0 \\ \frac{\partial \rho u}{\partial t} + \nabla u\rho u - \rho u\nabla u + \rho u\nabla u = -\frac{\rho}{M}\nabla(V+Q) \Leftrightarrow \frac{\partial \rho u}{\partial t} + \nabla u\rho u = -\frac{\rho}{M}\nabla(V+Q) \\ Q = -\frac{\hbar^2}{2m}\frac{\nabla\sqrt{\rho}}{\sqrt{\rho}} \end{cases}$$

which is equivalent to an Euler Fluid with a pressure term $-\frac{\rho}{M}\nabla(V+Q)$ (or an external force term).

We can further pull the potential term inside the differential operator, thus expressing all the Bohmian equations in terms of continuity relations

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \rho u = 0 \\ \frac{\partial \rho u}{\partial t} + \nabla \left(u \rho u + \frac{\rho}{M}(V+Q) \right) = 0 \\ Q = -\frac{\hbar^2}{2m} \frac{\nabla \sqrt{\rho}}{\sqrt{\rho}} \end{cases}$$

Thus our Z should be equal to $u\rho u + \frac{\rho}{M}(V+Q)$, giving the complete set of conservation relations for our model:

$$\begin{cases} \sum_\alpha f_\alpha^0 = \rho \\ \sum_\alpha f_\alpha^0 \hat{e_\alpha} = \rho u \\ \sum_\alpha f_\alpha^0 e_{\alpha i} \hat{e}_{\alpha j} = u \rho u + \frac{\rho}{M}(V+Q) \end{cases} \tag{7.12}$$

Now we define

$$f_\alpha^0 = a\rho + b\rho \vec{v}.e_\alpha + c(\vec{v}.e_\alpha)^2 + d(\vec{v}^2.e_\alpha^2) + e(V,Q) \tag{7.13}$$

which, together with the 2DQ9 lattice properties:

$$\begin{cases} \sum_\alpha e_\alpha = 0 \\ \sum_\alpha e_{\alpha i} e_{\alpha j} = 6e^2 \delta_{ij} \\ \sum_\alpha e_{\alpha i} e_{\alpha j} e_{\alpha k} = 0 \\ \sum_\alpha e_{\alpha i} e_{\alpha j} e_{\alpha k} e_{\alpha l} = 4e^4(\delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) - 6e^4 \delta_{ijkl} \end{cases}$$

will set some constraints on the coefficients in eq. 7.13 via the same procedure of Sec. 7.7.1. The resultant behaviour of this lattice Boltzmann system should be a close analog of the Bohmian equations:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \rho u = 0 + \mathcal{O}(\epsilon^2) \\ \frac{\partial \rho u}{\partial t} + \nabla \left(u \rho u + \frac{\rho}{M}(V + Q) \right) = 0 + \epsilon R_2 + \mathcal{O}(\epsilon^2) \\ Q = -\frac{\hbar^2}{2m} \frac{\nabla \sqrt{\rho}}{\sqrt{\rho}} \end{cases}$$

with an extra term ([59]) $R2$, a viscous term that can be tuned via τ, since $R_2 = (\tau - \frac{1}{2})(\frac{\partial^2 \pi_{ij}^{(0)}}{\partial t_0 \partial x_j} + \frac{\partial^2 P_{ijk}^{(0)}}{\partial t_0 \partial x_k})$ with $\pi_{ij}^0 = \sum_\alpha f_\alpha^0 e_{\alpha i} e_{\alpha j}$and $P_{ijk}^0 = \sum_\alpha f_\alpha^0 e_{\alpha i} e_{\alpha j} e_{\alpha k}$.

We pause here because we realize that by setting $Z = u\rho u + \frac{\rho}{M}(V + Q)$ in eqs. 7.12, we are going to end up with Q intact in the Boltzmann distribution function f. That is to say that, unlike in the standard lattice Boltzmann method, where we have constant coefficients in f, here at each different lattice point, we are going to have to compute $\nabla^2 \rho$ and $\nabla \rho$, which is an enormous extra overhead for the method.

7.8 Comments

First it should be noted that the classic Lattice Boltzmann method already has a significant overhead on its own (since it deals with N velocity vectors at each site), this hindrance is usually compensated by the ability to tune the microphysics to deal with specialized problems, complex boundaries, phase mixtures, etc.

Secondly, our Bohmian lattice Boltzmann method is set in an Euler frame (see eq. 7.10), so it does not benefit from the speed increase yielded by sparse grids of the Lagrangian implementations of previous chapters.

Finally, to my knowledge the only other way to include the quantum term in the Lattice Boltzmann theory is in the form of an external applied force. This means that the quantum force term still has to be calculated at each lattice point and does not "emerge" as some sort of collective behaviour (as say viscosity does in classic LBM). Given the non-local form of Q, this is a very expensive computational operation to

perform at each point.

Having concluded that the lattice Boltzmann method is probably not the optimal way to implement the Bohmian equations, we should take care not to discount cellular automata methods in general as a means of a quantum simulation. We are aware of the amazing things that different cellular automata can simulate ([43]) and other CA methods certainly will warrant further investigations.

Chapter 8

Conclusion

> "I think I can safely say that nobody understands quantum mechanics"
>
> Richard Feynman [55]

On a conceptual level it is somewhat comforting that Bohmian mechanics provides us with an ontological continuity between the worlds of the microscopic and macroscopic, quantum and classical, as we make use of the same concepts of trajectories, velocities, *etc.* which are familiar in the macroscopic world. This is what drew us to explore some aspects of the numerical side of Bohmian mechanics. To this effect we implemented fixed grid methods in 1D and 2D from which Bohmian trajectories were calculated. After a partial survey of direct numerical Bohmian methods, we considered different interpolation methods as ways of improving the stability of the calculations. In particular, we concluded that the smoothed splines method was more robust when simulating a model system for interference. Finally we investigated a possible link between Lattice Boltzmann Methods of the CA world and our Bohmian equations of motion.

We have concluded that as a numerical tool, direct Bohmian methods have some

stability shortcomings in specific situations (node problem, interference), and some performance drawbacks (Lattice Boltzmann). One has to realize though, we can't get something for nothing and this comes as a trade off from the extreme sparseness of the grids it uses which makes this method a viable alternative in situations where the dimensionality of the problem makes traditional grid methods with high resolution impracticable. Furthermore we saw problems that afflict Bohmian methods can be minimized by judicious use of point resampling, appropriate interpolation techniques and dynamical time integration scales.

Despite these remedies, one should not indiscriminately use Bohmian methods to solve every numerical problem, just as it would be unwise to use fixed grid methods to approach high dimensional problems, or situations where specific boundary conditions are required. (An exception is for 1D where TBCs are practical for grid based methods).

These two aspects, sparse grids and ability to impose arbitrary boundary conditions are then the major strengths of the Bohmian methods. These, incidentally are the shortcomings of fixed grid methods, just as the ability to deal well with nodes and interference is a source of trouble in the Bohmian methods, in a complementary fashion.

We should take away the idea that Bohmian methods and fixed grid methods complement each other nicely, and that, armed with both views we are much better equipped to face such numerical problems. This duality between these disparate approaches to the same numerical problems would almost be disconcerting were we not talking about quantum mechanics, the mother of innumerable dualities and paradoxes....

Bibliography

[1] Daniel Styer et al. "Nine Formulations of Quantum Mechanics", Am. J. Phys. **70** (2002) 288-97.

[2] L. de Broglie, "Recherches sur la theorie des quanta" Doctoral thesis, Université de Paris (1925)

[3] L. de Broglie, Ann. de Phys., Paris **3** (1925) 22.

[4] E. Madelung, "Quantentheorie in hydrodynamischer Form", Zeit. Phys. **40** (1926) 322-6.

[5] Bacciagaluppi, Valentini: "Quantum Theory at the Crossroads: Reconsidering the 1927 Solvay Conference", quant-ph/0609184, (2006)

[6] David Bohm, "A suggested interpretation of the quantum theory in terms of "hidden" variables", Phys. Rev. **85** (1952) I:166-79; II: 180-93

[7] D. Dürr, S. Goldstein and N. Zanghi, "David Joseph Bohm: 1917-1992", Foundations of Physics Letters, **6** (1993) 551-4; also P. Holland and J-P Vigier, Foundations of Physics **23** (1993) 5-6.

[8] R. E. Wyatt, "Quantum Dynamics with Trajectories", Springer Science, (2006)

[9] Hua Wu and D.W.L. Sprung, Phys. Lett. A **183** (1993) 413-7.

[10] C.L. Lopreore and R.E. Wyatt, "Quantum Wave packet dynamics with trajectories" , Phys. Rev. Lett. **82** (1999) 5190-4.

[11] F. Sales Mayor, A. Askar, and H.A Rabitz, "Quantum fluid dynamics in the Lagrangian representation and applications to photodissociation problems", J. Chem. Phys. **111** (1999) 2423-35

[12] A. Matzkin, " Are Bohmian trajectories real? On the dynamical mismatch between de Broglie-Bohm and classical dynamics in semiclassical systems", quant-ph/0609172 , (2006)

[13] W. Pauli, "Remarques sur le probleme des parametres cachés dans la mecanique quantique et sur la theorie de l'onde pilote" Andre George Ed. , "Louis de Broglie Physicien et Penseur", Paris: Albin Michel

[14] W. Myrvold, "On some early objections to Bohm's Theory", International Studies in the Philosophy of Science, **17** (2003) 7-24, see http://publish.uwo.ca/~wmyrvold/pub.htm

[15] P. R. Holland, "The Quantum Theory of Motion", Cambridge University Press, (1993)

[16] H. Goldstein, "Classical Mechanics", Addison-Wesley (1950) -
Note that subsequent editions omit the discussion on the relationship between classical Hamilton-Jacobi theory, Geometrical Optics and Wave-Mechanics

[17] A. Einstein, B. Podolsky, N. Rosen, "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? ", Phys. Rev. **47** (1935) 777 - 780.

[18] Y. Nogami, F. Toyama and W. van Dijk, Phys. Lett. A **270** (2000) 279-287.

[19] C. Phillipidis, C. Dewdney and B.J. Hiley, Nuovo Cimento **52 B** (1979) 15-28. "Quantum interference and the quantum potential"

[20] C. Moyer, "Numerov extension of transparent boundary conditions for the Schrödinger equation in one dimension.", Am. J. Phys **72** (2004) 351-360.

[21] W. van Dijk, F. M. Toyama, "Accurate numerical solutions of the time-dependent Schrödinger equation" , physics/0701150v1 or Phys. Rev. E **75** (2007) 036707

[22] S. Garashuk, V. Rassolov, "Semiclassical Dynamics based on quantum trajectories, Chem. Phys. Lett. **364** (2002) 562-7.

[23] http://physics.nist.gov/cuu/Constants/

[24] E.J. Heller , J. Chem. Phys. **62** (1975) 1544-55.

[25] M. Ehrhardt, "Finite Difference Schemes on unbounded Domains" , World Scientific, 2005, pp. 343-384

[26] A. Arnold, M. Ehrhardt, I. Sofronov,"Discrete transparent boundary conditions for the Schrödinger equation: Fast calculation, approximation and stability", Comm. Math. Sci. , (2003)

[27] Zisowsky A, Arnold A, Ehrhardt M and T. Koprucki, "Discrete transparent boundary conditions", Zeithschrift für angewandte Mathematik und Mechanik **58** (2005) 793-805.

[28] Gray S.K., Noid D.W. and Sumpter B.G., J. Chem. Phys. **101** (1994) 4062-72.

[29] J. Candy W. Rozmus, J. Comput. Phys. **92** (1991) 230-56.

[30] H. Yoshida, Celest. Mech. Dynam. Astron. **56** (1993) 27

[31] W. Press, S. Teukosky, W. Vetterling, B. Flannery , "Numerical Recipes: The Art of Scientific Computing", Cambridge University Press, (2002)

[32] C. Trahan, K. Hughes and R. Wyatt, "A new method for wave packet propagation: Derivative propagation along quantum trajectories", J. Chem. Phys. **118** (2003) 9911

[33] D. Babyuk and R. Wyatt, "Coping with the node problem in the hydrodynamic representation of quantum mechanics: The covering function method", J. Chem. Phys. **121** (2004) 9230

[34] B. Poirier, "Reconciling semiclassical and Bohmian mechanics: I stationary states", J. Chem. Phys. **121** (2004) 4501-4515

[35] C. Trahan and B. Poirier, "Reconciling semiclassical and Bohmian mechanics: II Scattering states for discontinuous potentials", J. Chem. Phys. **124** (2006)

[36] C. Trahan and B. Poirier, "Reconciling semiclassical and Bohmian mechanics: III Scattering states for continuous potentials", J. Chem. Phys. **124** (2006)

[37] J.G. Muga, J.P. Palao, B. Navarro, I.L. Egusquiza, "Complex absorbing Potentials", Phys. Rep. **395** (2004) 357-426

[38] C. de Boor, "A Practical Guide to Splines", Springer-Verlag, (1978)

[39] S. Wolfram, "Theory and Applications of Cellular Automata", World Scientific (1986)

[40] S. Wolfram, "A New Kind of Science", Wolfram publishing (2002)

[41] S. Wolfram, Cellular automaton fluids 1: Basic theory. Journal of Statistical Physics, **45** (1986) 471-529.

[42] K. Zuse, "Rechnender Raum", Friedrich Vieweg & Sohn, Braunschweig, (1969)

[43] A. Adamatzky et al, "Collision-Based Computing", Springer, (2002) , 491-

[44] J Maddox and E. Bittner, J. Chem. Phys. **115** (2001), 6309-16

[45] B. Kendrick, J. Chem Phys. **119** (2003) 5805-17

[46] T. Usuki, M. Saito, M. Takatsu, R.A. Kiehl and N. Yokoyama, Phys. Rev. B **52** (1995) 8244.

[47] R. Akis and D.K. Ferry and J.P. Bird, "Numerical simulation of quantum dots", Super. and Microstruc., **20** (1996) 323-9.

[48] L. Shifren, R. Akis, D.K. Ferry, "Correspondence between quantum and classical motion" Phys. Lett. A **274** (2000) 75-83.

[49] Hua Wu and D.W.L. Sprung, Phys. Lett. A **196** (1994) 229-236.

[50] S. Succi ,"The Lattice Boltzmann Equation for Fluid Dynamics and Beyond" , Oxford University Press (2001)

[51] S. Succi, "Lattice Boltzmann Equation: Failure or Success?", Physica A **240** (1997) 221

[52] J. G. Zhou, "Lattice Boltzmann Methods for Shallow Water Flows", Springer (2004)

[53] D. Wolf-Gladrow, "Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction", Springer (2000)

[54] G. McNamara and G. Zanetti, Phys. Rev. Lett. **56** (1988), 1065-70

[55] R.P. Feynman "The Character of a Physical Law" , Cambridge Mass. (1965)

[56] N. Gershenfeld, " The Nature of Mathematical Modeling", Cambridge University Press, (1999)

[57] A. Donoso, C.C. Martens," Quantum tunneling using entangled classical trajectories", Phys. Rev. Lett. **87** (2001) 223202

[58] P. Bhatnagar, E. Gross and M. Krook, "A model for collison processes in gases", Phys. Rev. **94** (1954) 511-525

[59] Y. Guangwu, C. Yaosong and H. Shouxin, "Simple lattice Boltzmann model for simulating flows with shock wave ", Phys. Rev. E **59** , 454 - 459 (1999)

Appendix A - Transparent Boundary Conditions

This is a summary of the algorithm we used, based on Moyer's([20]) implementation of transparent boundary conditions (TBC).

It is a fixed grid method whose core is the Cayley approximation to the system propagator:

$$\Psi(x,t+\Delta) \approx \frac{1-iH\Delta/2}{1+iH\Delta/2}\Psi(x,t)$$

This is rewritten with $y(x,t) = \Psi(x,t+\Delta)+\Psi(x,t)$, a function of two consecutive time steps as:

$$y'' - (V - i\frac{2}{\Delta})y = i\frac{4}{\Delta}$$

or $y'' = gy + f$ with $g = V - 2\frac{i}{\Delta}$ and $f = i\frac{4}{\Delta}$, a form suitable to be discretized on a uniform grid with spacing h. The Numerov scheme is used resulting in:

$$w_{j+1} + w_{j-1} = (2 + h^2\frac{g_j}{d_j})w_j + h^2\frac{f_j}{d_j} \tag{8.1}$$

where $d = 1 - h^2\frac{g}{12}$ and $w = dy - h^2\frac{f}{12}$. We see that Eq. 8.1 is a recursion relation

that depends on three consecutive points $j-1$, j, $j+1$. It can be put in terms of two recursion relations that involve two new variables e and q defined by $w_{j+1} = e_j w_j + q_j$:

$$\begin{cases} e_j + \frac{1}{e_{j-1}} = 2 + h^2 \frac{g_j}{d_j} \\ q_j = \frac{q_{j-1}}{e_{j-1}} + h^2 \frac{f_j}{d_j} \end{cases}$$

To enforce the TBCs we define at the left boundary ($j = 0$), $w_{j+1} = \alpha w_0 + \beta = e_0 w_0 + q_0$, with α and β determined according to

$$\begin{cases} e_0 = \alpha = a_0 \pm \sqrt{a_0^2 - 1} \\ q_0 = d_0^*(a_0^* - e_0)\Psi_0 + d_0(a_0 - \alpha_0) \sum_{k=1}^{n} l_{n-k+1} \Psi_0^k \end{cases}$$

subject to $|e_0| > 1$, with $a_0 = 1 + h^2 g_0 / 2 d_0$ and $d_0 = 1 - h^2 g_0 / 12$ (the letter superscripts denote time steps and subscripts the position). The last term, a convolution, implies that a history of Ψ must be kept at the boundary, constituting an increasing performance penalty as the number of time steps increases. The convolved value l_n has the form

$$l_n = \frac{exp(-in\phi)}{2n-1} \left(P_n(\mu) - P_{n-2}(\mu) \right)$$

where P_n is the Legendre polynomial of order n, and

$$\begin{cases} \lambda = 2\frac{h^2}{\Delta} \\ c = 1 - i\frac{\lambda}{6d} \\ a = 1 + h^2 \frac{g}{2d} \\ \phi = arg\left(\frac{a^2-1}{c}\right) \\ \mu = \frac{1-|a|^2}{|1-a^2|} \end{cases}$$

The same procedure applied to the other border on the right yields,

$$w_j^n = \frac{q_{j-1} + \beta_j^n e_{j-1}}{1 - \alpha_j e_{j-1}}$$

which, using the definition of w, gives Ψ advanced in time:

$$\Psi_j^{n+1} = -\Psi_j^n + i\frac{h^2}{3\Delta}\frac{\Psi_j^n}{d_j} + \frac{w_j^n}{d_j}$$

Appendix B - MATLAB Code Listings

Split Operator code / Fixed Grid Code

```
%% SYSTEM  TIME AND SPACE
SIZE=512              %cells
XL=-40;XR=40;    %nanometers
TIMESTEPS=2000
eV=1.6022*1e-19;
hbar=1.0546*1e-34;
c_light=2.9979*1e8;
Melectron=0.51099*1e6;
Meffective=0.067; % GaAs
MASS=Melectron*Meffective; %eV
MASSE=MASS;
MASS_KG=MASS*eV/c_light^2;
m=MASS*eV/c_light^2;
MASS=1/2;
Lenght_scale=1;
Time_scale=1e3*hbar/(2*m);
Energy_scale=eV*2*m*1e-18/hbar^2;
Velocity_scale=1e-3*2*m/hbar;
M=m;
fsec=Time_scale;
%Check
cprime=c_light/1e6*Velocity_scale;
DELTA=.1*fsec;          %timestep in femtoseconds
J=SIZE;
H=(XR-XL)/(SIZE-1);     %grid size
```

```
X=XL:H:XR;
temp=X*0;
%% SYSTEM MASS AND POTENTIAL
Vj=temp; % Potential in eV
%Vj=Vj+1e3*exp(-1/2/.0125^2*(X-.75).^2);
%X=X*Xmscale;
Vj=Vj+(X>-10)&(X<-5)+(X>5)&(X<10);
Vj=Vj*.25;
Vj=Vj*Energy_scale;
%Initial wavefunction
Psi0=temp;
wave_sigma=sqrt(10);
X0=-25;
K=0.31;
Psi0=exp(-1/2/wave_sigma^2*(X-X0).^2).*exp(i*K*X);
%Normalize
%Psi0=Psi0*1/sqrt(normaliz(abs(Psi0.*Psi0),X));
% SYSTEM DEFINITION ENDS
PSI=zeros(TIMESTEPS,SIZE);
PSI(1,:)=Psi0;
ALPHAj=temp;
%Fixed
Gj=Vj-2*i/DELTA;
Gj=Gj*(2*MASS);
Dj=1-H^2/12*Gj;
Aj=1+H^2*Gj./(2*Dj);
j=J;
niuJ=(1-abs(Aj(j)^2))/abs(1-Aj(j)^2);
j=1;
niu1=(1-abs(Aj(j)^2))/abs(1-Aj(j)^2);
% If V1=VJ then niu1=niuJ symmetrical
% Calculate all LEGENDRE coeficients (#timesteps) ahead of time index
% 1—>n=0
%REPEAT 1 For X=1
j=1;
lambda=2*H^2/DELTA; %Fixed
c=1-i*lambda/(6*Dj(j)); %Fixed
phi=angle((Aj(j).^2-1)./c); %Fixed
niu=(1-abs(Aj(j)^2))/abs(1-Aj(j)^2);
% PLEASE NOTE PN(n) is actually Legrendre polynomial of order n-1
% So  use to refer to PN(m) m=n+1
PN=temp;LN=temp;
PN(1)=1;PN(2)=niu; % These are PN0 and PN1
for m=2:(TIMESTEPS-1)
```

```
%generate Legendre polynomials;
n=m-1;
PN(m+1)=2*niu*PN(m)-PN(m-1)-(niu*PN(m)-PN(m-1))/(n+1);
end;
LN(1)=niu*exp(-i*phi); % LN(0) would be -1; LNn is LN(n)
for n=2:(TIMESTEPS-1)
m=n+1;
LN(n)=exp(-i*n*phi)/(2*n-1)*(PN(m)-PN(m-2));
%LN(n)=exp(-i*n*phi)*(niu*PN(m-1)-PN(m-2))*(2*n+1)/(2*n-1)/(n+1);
end;
LN1=LN;
%REPEAT 2 for X=J
j=J;
lambda=2*H^2/DELTA; %Fixed
c=1-i*lambda/(6*Dj(j)); %Fixed
phi=angle((Aj(j).^2-1)./c); %Fixed
niu=(1-abs(Aj(j)^2))/abs(1-Aj(j)^2);
% PLEASE NOTE PN(n) is actually Lagrage polynomial of order n-1
% So  use to refer to PN(m) m=n+1
PN=temp;LN=temp;
PN(1)=1;PN(2)=niu; % These are PN0 and PN1
for m=2:(TIMESTEPS-1)
%generate Legendre polynomials;
n=m-1;
PN(m+1)=2*niu*PN(m)-PN(m-1)-(niu*PN(m)-PN(m-1))/(n+1);
end;
LN(1)=niu*exp(-i*phi); % LN(0) would be -1; LNn is LN(n)
for n=2:(TIMESTEPS-1)
m=n+1;
LN(n)=exp(-i*n*phi)/(2*n-1)*(PN(m)-PN(m-2));
%LN(n)=exp(-i*n*phi)*(niu*PN(m-1)-PN(m-2))*(2*n+1)/(2*n-1)/(n+1);
end;
LNJ=LN;
% ##################################
% ##################################
%                 LOOP
% ##################################
% ##################################
for N_INDEX=1:(TIMESTEPS-1)
N_INDEX
% ##################################
%STEP 1
% ##################################
Fj=4*i*PSI(N_INDEX,:)/DELTA;
```

```
Fj=Fj*2*MASS;
% ########################################
%STEP 2
% ########################################
%From the left
% Calc Ej  Fixed
t1=Aj(1)+sqrt(Aj(1)^2-1);
t2=Aj(1)-sqrt(Aj(1)^2-1);
if(abs(t1)>1) ALPHAj(1)=t1; else ALPHAj(1)=t2; end;
Ej(1)=ALPHAj(1);
%Recurrence
for j=2:SIZE
      Ej(j)=2*Aj(j)-1/Ej(j-1);
end
% Calc Qj Variable
j=1;
%convolution
SUM=0;
for k=1:N_INDEX
n=N_INDEX-k+1;
SUM=SUM+PSI(k,j)*LN1(n);
end;
j=1;
Qj(j)=conj(Dj(j))*(conj(Aj(j))-ALPHAj(j))*PSI(N_INDEX,j)+Dj(j)*(Aj(j)-
ALPHAj(j))*SUM;
%Recurrence
for j=2:SIZE
      Qj(j)=Qj(j-1)/Ej(j-1)+H^2*Fj(j)/Dj(j);
end
% ########################################
%STEP 3
% ########################################
%From the right
t1=Aj(J)+sqrt(Aj(J)^2-1);
t2=Aj(J)-sqrt(Aj(J)^2-1);
if(abs(t1)>1) ALPHAj(J)=t1; else ALPHAj(J)=t2; end;
j=J;
%convolution
SUM=0;
for k=1:N_INDEX
n=N_INDEX-k+1;
SUM=SUM+PSI(k,j)*LNJ(n);
end;
j=J;
```

```
BETAj(j)=conj(Dj(j))*(conj(Aj(j))-ALPHAj(j))*PSI(N_INDEX,j)+Dj(j)*(Aj(j)-
ALPHAj(j))*SUM;
Wj(J)=(Qj(J-1)+BETAj(J)*Ej(J-1))/(1-ALPHAj(J)*Ej(J-1));
%Recurrence for Wj
for j=(J):-1:2
      Wj(j-1)=(Wj(j)-Qj(j-1))/Ej(j-1);
end
% ##############################
%STEP 4
% ##############################
% Calculate Psi from Wj
PSI(N_INDEX+1,:)=PSI(N_INDEX,:).*(i*H^2./(3*DELTA*Dj)-1)+Wj./Dj;
end;
figure
image(abs(PSI)*100);
ylabel('Timestep (.1 fs)');xlabel('X From -40 to 40 (nm)');
```

Bohmian Code

Newtonian

```
%SIZE=512              %cells
XL=-40;XR=40;
nump=40;
porder=2;
maxgauss=1;
TOLERANCE=1e-5*nump;
SIZE=10000; %for psi,etc
TIMESTEPS=2000  %
DELTA=1;
DELTA=DELTA*fsec;              %timestep in femtoseconds
POTENTIAL='0*0.0061.*X.^4+50*cos(X)*0';
J=SIZE;
H=(XR-XL)/(SIZE-1);     %grid size
X=XL:H:XR;
temp=X*0;
%% SYSTEM MASS AND POTENTIAL
Vj=temp; % Potential in eV
%Vj=Vj+1e3*exp(-1/2/.0125^2*(X-.75).^2);
%X=X*Xmscale;
%Vj=Vj+(X>-10)&(X<-5)+(X>5)&(X<10);
```

```
%Vj=Vj*.25;
Vj=Vj+eval(POTENTIAL);
Vj=Vj*Energy_scale;
VV=Vj;
%Initial wavefunction
Psi0=temp;
wave_sigma=sqrt(10);
X0=20;
K=0.31;
%Psi0=exp(-1/2/wave_sigma^2*(X-X0).^2).*exp(i*K*X);
Psi0=exp(-1/2/wave_sigma^2*(X-X0).^2).*exp(i*K*X);
Psi1=exp(-1/2/wave_sigma^2*(X+X0).^2).*exp(i*K*X);
Psi0=Psi0+1/2*Psi1;
%Psi0=eval(PSI0);
%Normalize
%Psi0=Psi0*1/sqrt(normaliz(abs(Psi0.*Psi0),X));
% SYSTEM DEFINITION ENDS
PSI=zeros(TIMESTEPS,SIZE);
PSI(1,:)=Psi0;
rho=abs(Psi0).^2;
S=angle(Psi0);
%S=unwrap(S,[],2);
S=unwrap(S,[],1);
vel=diff(S);
%p0=log(abs(Psi0));
p0=abs(Psi0).^2;
cp0=cumtrapz(p0);
cp0=cp0/cp0(SIZE);
clear XX;
XX=ones(nump,TIMESTEPS)*NaN;
QQ=ones(nump,TIMESTEPS)*NaN;
PP=XX;
for i=1:nump
    i;
    %XX(i,1)=(X(siz)-X(1))/(nump+1)*i; % trial points uniformly distributed in spa
       % can also be density dependent on Psi0
     %  XX(i,1)=-35+(i-1)*30/nump;
          % tmpr=0;
      % while(1)
      % tmpi=ceil(rand(1)*siz);
       %tmpr=rand(1);
       %if(tmpr<p0(tmpi))XX(i,1)=X(tmpi);break;end;
       %end;
       prob1=1/(nump+1)*i;
```

```
        XX(i,1)=interpolate(X,cp0,prob1);
%        tmpi=getindex(cp0,1/(nump+1)*i);
%         XX(i,1)=X(tmpi); %dont use X
        % use this instead
        PD(i,1)=1/(nump+1)*i;
        end;
        QQ=XX;
        PP=0*XX;
        FF=XX*0;
        %QF=XX*0;
        %%%%%% LOOP
        dt=DELTA;
        VG=diff(VV);tmp=VG;VG=[tmp VG(SIZE-1)];
        go_back1step=0;
tstep=2;
global FF;
while(tstep<=TIMESTEPS)
   % this routine will try to get an analytical expression for rho;
       [coefs resi0 resi1 resi2 resi3]=analyse_rho21(QQ,PD,tstep-1,porder,maxgauss);
%        resi3=0*resi3;
%        resi2=0*resi2;
%        resi1=0*resi1;
       myresidues=[resi0;resi1;resi2;resi3];
         if(go_back1step)if(tstep>2)tstep=tstep-1;go_back1step=0;end;end;  % not run h
         tstep
         dt
    notvalid=1;
    error=0;

    while notvalid ,
      %notvalid
%      go_back1step
%      tstep
      if(go_back1step)if(tstep>2)tstep=tstep-1;go_back1step=0;end;end;
%        tstep
      %error
      %dt+9
      error=0;
      notvalid=0;
      for i=1:nump
    q0=QQ(i,tstep-1);
    p0=PP(i,tstep-1);
[qf, pf, ff ] = symplectic21(q0,p0,dt,X,QQ,PP,VG,i,nump,tstep,PD,coefs,...
porder,myresidues);
```

```
%[qf_2, pf_2, ff_2] = symplectic(q0,p0,2*dt,X,QQ,PP,VG,i,nump,tstep);
[qh1, ph1, fh1] = symplectic21(q0,p0,dt/2,X,QQ,PP,VG,i,nump,tstep,PD,...

coefs,porder,myresidues);

[qh2, ph2, fh2] = symplectic21(qh1,ph1,dt/2,X,QQ,PP,VG,i,nump,tstep,PD,...

coefs,porder,myresidues);

if(~isfinite(qh2)) fsdadsaerrrrrrr;end;
%    error=abs(ph2-pf); %momentum
error=abs(qh2-qf); %space
%error_lagre=abs(qf_2-qf); %space
%MATLAB BUG ;break; end;
if(error>TOLERANCE)
  not_valid=1;dt=dt/2;
  go_back1step=1;
end;
  % Check for trajectory crossing
  if((i>1)&(i<nump))
    if(qh2<QQ(i-1,tstep-1))|(qh2>QQ(i+1,tstep-1))
    %% bad HACK HACK
    qh2=1/2*(QQ(i-1,tstep-1)+QQ(i+1,tstep-1));
    ph2=1/2*(PP(i-1,tstep-1)+PP(i+1,tstep-1));
    %    cxzcxz
    if(tstep>3)tstep=tstep-3;end
    not_valid=1;dt=dt/2;
    go_back1step=1;
%    break cant use break because of a matlab bug
    %go back 1 step;
  end;end;
%if(error_large>TOLERANCE) not_valid=1;dt=dt*1.5;break; end;
    QQ(i,tstep)=qh2;
    FF(i,tstep)=fh2;
    PP(i,tstep)=ph2;
    TT(i,tstep)=dt;
    end;
    %Test for crossings.if crossed try again with timestep=1/2 current
    %Test for accuracy.try with 2x timestep..if accuracy below threshold timestep=

if(error<TOLERANCE)&(~go_back1step)not_valid=0;end;
  end;
  %we got valid
  %endkiio
  dt=dt*1.002;
tstep=tstep+1;
```

```
  end;
  subplot(3,1,1);
  pp=QQ+j*PP;
  %plot(pp(:,:)','x-')
  plot(QQ',PP');xlabel('space');ylabel('momentum');
  subplot(3,1,2);
%  plot(PP','x-')
  plot(QQ');xlabel('time');ylabel('space');
  subplot(3,1,3);
  plot(TT(1,:));xlabel('step');ylabel('time delta');
  z=cumsum(TT');
  figure
  plot(z,QQ');;ylabel('space');xlabel('linear time ');
grid
  dt*TIMESTEPS
(  sum(QQ(:,TIMESTEPS)~=sort(QQ(:,TIMESTEPS))))/nump
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function       [coefs resi0 resi1 resi2 resi3]=analyse_rho22(QQ,PD,time,porder,...

maxgauss,fourier_residue)

% this routine will try to get an analytical expression for rho;
% Get quadratic coeficients...
 zx=QQ(:,time);zy=PD;
 rho=gradient(zy,zx);
 CC=1/2*log(rho);
[siza sizb]=size(rho);
siz=max(siza,sizb);
%[maxi,mini,num_max,num_min]=findmaxmin(CC,siz,10,9);
maxi=siz;mini(1)=1;mini(2)=siz;num_max=1;num_min=2;
ngauss=num_min-1;
if(ngauss>maxgauss)ngauss=maxgauss;end;

  %%% ONLY WORKS FOR 1 GAUSSIAN NOW FIXME
  coef=zeros(ngauss,porder+1);
%generate coefs
for n=1:(num_min-1)
  istart=mini(n);
  iend=mini(n+1);
  %%BUG remove this hack
  istart=istart+1;
  iend=iend-1;
  %%BUG remove this hack
```

```
tempy=CC(istart:iend);
tempx=QQ(istart:iend,time);
coef(n,:)=polyfit(tempx,tempy,porder);
  %SANITY CHECK
if(coef(n,1)>0)coef(n,:)=coef(n,:)*0;end;
residy=tempy-polyval(coef(n,:),tempx);
fresidy=dst(residy);
end;
specsize=max(max(size(residy)));
%res0=idst(residy); will be a check later
res0=zeros(1,specsize);
res1=zeros(1,specsize);
res2=zeros(1,specsize);
res3=zeros(1,specsize);
%should calculate this with idst...but like this it is a good check
N=specsize;
for n=1:N
  for k=1:N
    %% IF WE WANT TO LOWPASS THIS IS WHERE TO DO IT
    if(k>N/10)fresidy(k)=0;end; % 20%
    ss=sin(pi*k*n/(N+1));
    cc=cos(pi*k*n/(N+1));
  res0(n)=res0(n)+fresidy(k)*ss;
  res1(n)=res1(n)+fresidy(k)*cc*(pi*k/(N+1));
  res2(n)=res2(n)+fresidy(k)*(-ss)*(pi*k/(N+1))*(pi*k/(N+1));;
  res3(n)=res3(n)+fresidy(k)*(-cc)*(pi*k/(N+1))*(pi*k/(N+1))*(pi*k/(N+1));;
end;
end;
res0=res0*2/(N+1);
res1=res1*2/(N+1);
res2=res2*2/(N+1);
res3=res3*2/(N+1);
temperr=res0-residy';
err_in_idst=max(max(temperr));
%coef=polyfit(QQ(:,time),CC,porder);
%###############
%DEBUG GOODNESS OF FIT
figure(1);
hold off
subplot(2,1,1);
plot(QQ(:,time),CC,'x')
hold on
for n=1:ngauss
```

```
  istart=mini(n);
  iend=mini(n+1);
  %%BUG remove this hack
  istart=istart+1;
  iend=iend-1;
xstart=QQ(istart,time);
xend=QQ(iend,time);
vect=xstart:(xend-xstart)/200:xend;
drawnow
subplot(2,1,1);
plot(vect,polyval(coef(n,:),vect));
hold off
subplot(2,1,2);
plot(tempx,res0,'x-')
drawnow;
end;
hold off
%################
figure(2)
plot(QQ(:,1:time)');
coef
%z
coefs=coef;
resi0=res0;
resi1=res1;resi2=res2;resi3=res3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [q2, p2, f1 ] = symplectic21(q0,p0,dt,X,QQ,PP,VG,i,nump,tstep,PD,coefs,...

porder,myresidues)

%    q0=QQ(i,tstep-1);
%    p0=PP(i,tstep-1);
    %get force term from potential
%    curi=getindex(X,q0);
%if(curi>0)    f0=-VG(curi);end;
%f0=-2*q0;
%porder
  f0=extforces21(q0,p0,QQ,PP,i,nump,tstep,PD,coefs,porder,myresidues);
  f0=f0+0;
    p1=p0+dt/2*f0;
    q1=q0+dt/2*p1;
%get force term from potential
%curi=getindex(X,q1);
%if(curi>0)    f1=-VG(curi);end;
```

```
  %f1=-2*q1;
  f1=extforces21(q1,p1,QQ,PP,i,nump,tstep,PD,coefs,porder,myresidues);
  f1=f1+0; %other forces

    p2=p1+dt/2*f1;
    q2=q1+dt/2*p2;
 [q2, p2, f1];
 return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [retemp QF] =extforces21(q0,p0,QQ,PP,i,nump,tstep,PD,coefs,...

porder,myresidues)

  temp=0;
  resi0=myresidues(1,:);
  resi1=myresidues(2,:);
  resi2=myresidues(3,:);
  resi3=myresidues(4,:);
  %for n=1:nump
   % if n==i continue;end;
    %  d=((q0-QQ(n,tstep-1)));
     % u=sign(d);
      %temp=temp+100*1/d^2*(u);
      %end;

      %DEPRECATED ################################### DERIVATIVES

%      QVector=QQ(:,tstep-1);
%      list=get_nearest_neigh_wself(QVector,i,5);
%      PD_DIFFS= maple_deriv(QQ(list),PD(list)');
%      P0=PD_DIFFS(1);P1=PD_DIFFS(2); P2=PD_DIFFS(3);P3=PD_DIFFS(4);P4=PD_DIFFS(5)
      % This is the Q term already diff  to give force not potential
%      QF=-1/2*(P4*P1^2+P2^3-2*P2*P3*P1)/P1^3; %
%      temp=QF*0.001;
%DEPRECATED ###################################
% #### C=C0+C1*X +C2*X^2   +C3 X^3 ....
% CD1=C1+2*C2*X+3*C3*X^2+....
% CD2=2*C2+3*2*C3*X+....
     %coefs
 %    C0=coefs(3);       coef(49) (porder 48)
 %    C1=coefs(2);       coef(48)
 %    C2=coefs(1);       coef(47)
 %  CALC DERIVATIVES ANALYTICALLY
coef=0;
```

```
X=q0;
CD1=0;
 for n=1:(porder+1-1);
   coef=coefs(n);
   m=porder-n+1;
   CD1=CD1+coef*(m)*X^(m-1);
 end;
CD2=0;
 for n=1:(porder+1-2);
   coef=coefs(n);
   m=porder-n+1;
   CD2=CD2+coef*(m)*(m-1)*X^(m-2);
 end;
CD3=0;
if(porder>2)
 for n=1:(porder+1-3);
   coef=coefs(n);
   m=porder-n+1;
   CD3=CD3+coef*(m)*(m-1)*(m-2)*X^(m-3);
 end;
end;
     % %%%%%%%%% QF=-(2*cd1*cd2+cd3_
     QF=(2*CD1*CD2+CD3);
     %     temp=0.1*QF;
     % ##############################################
     [ngauss temp]=size(coefs);
     rho0=0;rho1=0;rho2=0;rho3=0;
% FOURIER RESIDUES
%APPROX TO LAST PARTICLE...NOT PRECISELY CORRECT
res0=0;
res1=0;
res2=0;
res3=0;
if(i>1)&&(i<nump-1)
res0=resi0(i);
res1=resi1(i);
res2=resi2(i);
res3=resi3(i);
end;
     % ONLY WORK WITH 1 GAUSSIAN
     %for n=1:ngauss
     n=1;
       %% RESOLVE ILL CONDITIONING
       a2=coefs(n,1);
```

```
        a1=coefs(n,2);
        a0=coefs(n,3);
%n
        expo=exp(a2*q0^2+a1*q0+a0); %% BUG

        rho0=rho0+expo*exp(res0);

        rho1=rho1+expo*(2*a2*q0+a1+res1);
        rho2=rho2+expo*(2*a2+res2+(2*a2*q0+a1+res1)^2);
        rho3=rho3+expo*(res3+3*(2*a2+res2)*(2*a2*q0+a1+res1)+(2*a2*q0+a1+res1)^3);
%     end;

%rho0
%rho0=rho0+expo*res0;
%rho1=rho1+expo*res1;
%rho2=rho2+expo*res2;
%rho3=rho2+expo*res3;
temp=1/2*(rho1^3-2*rho1*rho2*rho0+rho3*rho0^2)/rho0^3;
%checkc=(temp/2/alphan^2+miu)/q0;
% Harmonic osc
%get force term from potential
%curi=getindex(X,q1);
%f1=0;
%if(curi>0)    f1=-VG(curi);end;
  %f1=-2*q1;
  %     temp=0.1*QF;

  %% THE -0.001*p0 friction term will stabilize the density function
  %tempi=-.1*q0-0.005*p0;%-0.003*q0^3;;
  xx=q0-5;

%testing diff force terms

  %  tempi=+0.001*sech(xx)*tanh(xx)*1-0*q0-0.0000*q0^4;
%    tempi=-sech(xx)*tanh(xx)*1-0*q0-0.0000*q0^4;
    tempi=-sech(xx)*tanh(xx)*0-1*q0-0.0000*q0^4;
  retemp=temp+tempi;
  ;
```

QHJE

```
tic
POTENTIAL='0*t+1*x.^2'
```

```
ONEHALF=1/2*(1);
nump=50;porder=6; %linear
TOLERANCE=1e-5*nump;
TIMESTEPS=10000 %
dt=10^-2*1;
clear order points
global order points
order=2
points=order+1
%points=
% this last part is assuming X spacing of 1
clear XX;
XX=ones(nump,TIMESTEPS)*NaN;
QQ=ones(nump,TIMESTEPS)*NaN;
RH=XX;PP=XX;
VP=XX;

       QQ=XX;
       %PP=0*XX;
       FF=XX*0;
       %Q=XX*0;

DDSS=PP;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X=-10*1:20*1/(nump-1):10*1;
%  for i=1:max(size(X)) y(i)=QPotential(1,X(i));end; plot(X,y,'x-')
%
clear XX;
XX=ones(nump,TIMESTEPS)*NaN;
QQ=ones(nump,TIMESTEPS)*NaN;
RH=XX;PP=XX;
SS=XX;
VP=XX;
VP=VP*0;
for i=1:nump
        QQ(i,1)=X(i);
        RH(i,1)=log(rho(0,X(i)))/2;
        SS(i,1)=Sphase(0,X(i));
        PP(i,1)=vel(0,X(i));
        VP(i,1)=Potential(0,X(i));
end
for i=1:TIMESTEPS
VP(:,i)=VP(:,1); %time indep for now
end
```

```
%initial phase fix
temp=SS(:,1);
temp=unwrap(temp,2);
SS(:,1)=temp;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%STARTHERE
tstep=1;
ctime=-dt; %not zero
while(tstep<=TIMESTEPS),
dt
notvalid=1;go_back1step=0;
while notvalid,
if(go_back1step)
tstep=tstep-go_back1step;
tstep=max(tstep,1);
go_back1step=0;
end;
notvalid=0;
%{
%this is slow somehow
dt2=dt/2;
t=ctime;
for i=1:nump
x=QQ(i,tstep);
VP(i,tstep)=0;%eval(POTENTIAL); %time indep for now
end
plot(VP,'>-');
%%%%%%%%%% 2 half steps
%}
%faster here
x=QQ(:,tstep);
%VP(:,tstep)=10*10^-3*(x-2).^2;
VP(:,tstep)=0*1*10^-4*(x-2).^2.*(x-0).^2;
%% POTENTIAL HERE
%VP(:,tstep)=10^1*sech(4*(x-3));
VP(:,tstep)=0*8*10^-2*(x).^2;
qq00=QQ(:,tstep);
%{
if(tstep>1)
qp11=calcQP3(qq00,RH(:,tstep),RH(:,tstep-1),QP(:,tstep-1));
else
qp11=calcQP(qq00,RH(:,tstep));
end
[pp11 ddss11]=calcPPDDSS(qq00,SS(:,tstep)); %
```

```
qq12=QQ(:,tstep)+dt2*pp11; % only 1st order
%VP(:,tstep)=10*pp11.^2;%0*1*10^-4*(x-2).^2.*(x-0).^2;
%% POTENTIAL HERE
ss12=SS(:,tstep)+dt2*(ONEHALF*pp11.^2-(VP(:,tstep)+qp11));
[ptemp ddss12]=calcPPDDSS(qq12,ss12);
rh12=RH(:,tstep)+dt2*(-1/2*ddss12);
%rh12=RH(:,tstep)+dt2*(-1/2*ddss11);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(tstep>1)
qp12=calcQP3(qq12,rh12,RH(:,tstep),qp11);
else
qp12=calcQP(qq12,rh12);
end
[pp12 ddss12]=calcPPDDSS(qq12,ss12); %
qq22=qq12+dt2*pp12; % only 1st order
ss22=ss12+dt2*(ONEHALF*pp12.^2-(VP(:,tstep)+qp12));
[pp22 ddss22]=calcPPDDSS(qq22,ss22);
rh22=rh12+dt2*(-1/2*ddss12);
%rh22=rh12+dt2*(-1/2*ddss11);
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
if(tstep>1)
qp11=calcQP3(qq00,RH(:,tstep),RH(:,tstep-1),QP(:,tstep-1));
else
qp11=calcQP(qq00,RH(:,tstep),qq00,qq00);
end
%  qp11=threshold_smooth(qq00,qp11,1);
%qp11=fixborders2(qq00,qp11,3,0,1);
[pp11 ddss11]=calcPPDDSS(qq00,SS(:,tstep)); %% v=grad(S)
%pp11=fixborders2(qq00,pp11,3,0,1);
%ddss11=fixborders2(qq00,ddss11,3,0,1);
%  ddss11=threshold_smooth(qq00,ddss11,10);
%no need for this block use from prev one
qq1f=QQ(:,tstep)+dt*pp11; % only 1st order
ss1f=SS(:,tstep)-dt*(ONEHALF*pp11.^2+(VP(:,tstep)+qp11));
[ptemp ddss1f]=calcPPDDSS(qq1f,ss1f);
%    ddss1f=fixborders2(qq00,ddss1f,11,0,1);
%  ddss1f=threshold_smooth(qq1f,ddss1f,10);
rh1f=RH(:,tstep)+dt*(-1/2*ddss1f);
%rh1f=RH(:,tstep)+dt*(-1/2*ddss11);
%}
ITER_HIGHTHRESH=16;
ITER_LOWTHRESH=9;
```

```
THRESHOLD=10^-13;
X0=QQ(:,tstep);S0=SS(:,tstep);C0=RH(:,tstep);
X1=X0;S1=S0;C1=C0;
%  [DS0 DDS0]=calcPPDDSS(X1,S1);
%  [DC0 DDC0]=calcPPDDSS(X1,C1);
[DS0 DDS0 DC0 DDC0]=calcPPDDSS_alternate_zones(X1,S1,C1);
%  S0=S0+dt/2*(-(DS0.^2)+DDC0+DC0.^2);
%  DDC0=DDC0*0;DC0=DC0*0;
%  S1=S0+dt/2*(-(DS0.^2)+DDC0+DC0.^2); %jumpstart S1 for zones
%  figure(3);plot(S1);
notvalid=1;
while notvalid,
iter=0;DDS=X0;DDC=X0;err=100;
while abs(err)>THRESHOLD,
iter=iter+1;
LASTDDS=DDS;
%  [DS DDS]=calcPPDDSS_alternate(X1,S1);
%  [DC DDC]=calcPPDDSS_alternate(X1,C1);
[DS DDS DC DDC]=calcPPDDSS_alternate_zones_varpoly(X1,S1,C1);
thres=10^-10;
%  DDC=threshold_smooth2(X1,DDC,thres);
%  DDS=threshold_smooth2(X1,DDS,thres);
%  DC=threshold_smooth2(X1,DC,thres);
%  DS=threshold_smooth2(X1,DS,thres);
%  DDC=DDC*0;DC=DC*0;
%      DDS=fixborders2(X1,DDS,2,0,1);
%      DS=fixborders2(X1,DS,2,0,1);
%      DDC=fixborders2(X1,DDC,2,0,1);
%      DC=fixborders2(X1,DC,2,0,1);
err=max(max(LASTDDS-DDS));
X1=X0+dt*DS;
S1=S0+dt/2*(-(DS.^2)+DDC+DC.^2);
C1=C0-dt/2*DDS;
if(iter>50) errorinfiniteloop; end;
%  plot(X0,DDS);grid;drawnow
end
iter
if(iter>ITER_HIGHTHRESH)
        dt=dt/2; notvalid=1;
else
notvalid=0;
 end;
end;
%trapezoid semiimpl
```

```
X1=X0+dt/2*(DS+DS0);
S1=S0+dt/4*(-(DS.^2)-(DS0.^2)+DDC+DDC0+DC.^2+DC0.^2);
C1=C0-dt/4*(DDS+DDS0);
%  X1=X0+dt*DS;
%  S1=S0+dt/2*(-(DS.^2)+DDC+DC.^2);
%  C1=C0-dt/2*DDS;
%%%%%%%%%%%%%%%
%compare q1f to qq22
%maxerror=max(abs(qq22-qq1f));
%if(maxerror>TOLERANCE)
%  not_valid=1;dt=dt/2;
%  go_back1step=1;
%end
qq1f=X1;
if(min(diff(qq1f))<=0) %there was a crossing
toc
fefrefre;
end

end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% use the 2 half step results anyways
%  QP(:,tstep)=qp12;
%  PP(:,tstep)=pp12;
%  DDSS(:,tstep)=ddss12;
%  QQ(:,tstep+1)=qq22;
%  SS(:,tstep+1)=ss22;
%  DDSS(:,tstep+1)=ddss22;
%  RH(:,tstep+1)=rh22;
% Or force usage of single big step
  QP(:,tstep)=-1/2*(DDC+DC.^2);
  PP(:,tstep)=DS;
  DDSS(:,tstep)=DDS;
  QQ(:,tstep+1)=X1;
  SS(:,tstep+1)=S1;
  DDSS(:,tstep+1)=DDS;
  RH(:,tstep+1)=C1;
%pause
%order bug tstep+1
%plot(QQ(:,tstep),(RH(:,tstep)),'x-');drawnow;
%plot(QQ(:,tstep),exp(RH(:,tstep)),'x-');drawnow;
%plot(QQ(:,tstep),SS(:,tstep),'x-');drawnow;
%plot(QQ(:,tstep),QP(:,tstep),'x-');drawnow;
figure(1);
```

```
subplot(4,1,1);
plot(QQ(:,tstep),(RH(:,tstep)),'x-');
subplot(4,1,2);
plot(QQ(:,tstep),QP(:,tstep),'x-');
subplot(4,1,3);
plot(QQ(:,tstep),(SS(:,tstep)),'x-');
%p12
%plot(QQ(:,tstep),-1/2*2*ddss22,'x-');
[temp1 temp2]=calcPPDDSS(QQ(:,tstep),QP(:,tstep));
plot(QQ(:,tstep),DS,'x-')
subplot(4,1,4);
plot(QQ(:,tstep),DDS,'x-')
%plot(QQ(:,tstep),temp2,'x-')
%if(tstep>2)plot(QQ(:,tstep),(QP(:,tstep)-QP(:,tstep-1))./dt,'x-');end
drawnow;
if pauses pause; end;
%plot(QQ(:,tstep),gradient(SS(:,tstep)),'x-');drawnow;
%pause(2)
deltat(tstep)=dt;
ctime=ctime+dt;
time(tstep)=ctime;
dt=dt;
ctime
tstep=tstep+1
if(iter<ITER_LOWTHRESH) dt=dt*1.01; end;
end
toc
%}%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [DS DDS DC DDC]=calcPPDDSS_alternate_zones(X,S,C)
% needs accurate initial S to work!
siz=max(size(X));
my_S=spapi(4,X,S);
sp1=fnder(my_S,1);t1=fnval(sp1,X);
sp2=fnder(my_S,2);t2=fnval(sp2,X);
sp3=fnder(my_S,3);t3=fnval(sp3,X); % may have to bump up order of splines
DS0=t1;  DDS0=t2;  DDDS0=t3;
my_C=spapi(4,X,C);
sp1=fnder(my_C,1);t1=fnval(sp1,X);
sp2=fnder(my_C,2);t2=fnval(sp2,X);
sp3=fnder(my_C,3);t3=fnval(sp3,X); % may have to bump up order

zones=zeros(1,siz);
zones=(DDC0>10^-8);
%enlarge by 1
```

```
enl=4;
zones=smooth(zones,enl*2+1);%zones=zones>0;
[numz breaks]=getnumtypezones(zones);
%  figure(2);plot(zones,'-x');drawnow;
starts=1;
for i=1:numz
% 0 for normal transition zones
% -1 for always splines
% 3 for always least sq
if(zones(starts)>0)
%splines or local
ends=breaks(i);
DC(starts:ends)=DC0(starts:ends);
DDC(starts:ends)=DDC0(starts:ends);
DS(starts:ends)=DS0(starts:ends);
DDS(starts:ends)=DDS0(starts:ends);
starts=ends+1;
else
ends=breaks(i);
%least sq or global
gx=X(starts:ends);gc=C(starts:ends);gs=S(starts:ends);
my_C=spap2(1,3,gx,gc);my_S=spap2(1,3,gx,gs);
sp1=fnder(my_C,1);t1=fnval(sp1,gx);
sp2=fnder(my_C,2);t2=fnval(sp2,gx);
DC(starts:ends)=t1;
DDC(starts:ends)=t2;
sp1=fnder(my_S,1);t1=fnval(sp1,gx);
sp2=fnder(my_S,2);t2=fnval(sp2,gx);
DS(starts:ends)=t1;
DDS(starts:ends)=t2;
starts=ends+1;
end;
end;
DC=DC';DDC=DDC';DDS=DDS';DS=DS';
  return;
%  sp=spapi(4,x,yerr);
%  ch = spapi(augknt(x,4,2), [x x], [yerr dy]);
%  ls=spap2(18,3,x,yerr); %1 quadratic pieces=x-k+1;
%  my_sp=spapi(4,X,S); % 3rd order spline (cubic)
my_S=spap2(1,3,X,S);
my_C=spap2(1,3,X,C);
%my_sp=csapi(X,S); %reg spline
%  my_sp=csaps(X,S); %smoothed
sp1=fnder(my_S,1);t1=fnval(sp1,X);
```

```
sp2=fnder(my_S,2);t2=fnval(sp2,X);
  DS=t1;
  DDS=t2;
sp1=fnder(my_C,1);t1=fnval(sp1,X);
sp2=fnder(my_C,2);t2=fnval(sp2,X);
  DC=t1;
  DDC=t2;
%  t1=fixborders(X,t1,2,2); %linear
%  t2=fixborders(X,t2,2,1);% quadratic
return
%  %
%  %
siz=max(size(X));
order=2;
points=order+1;
interpx=zeros(1,points);interpy=interpx;
  t1=interpx;t2=interpx;
for i=1:siz
originx=X(i);
indexes=get_n_neighbours(points,X,originx);
interpx=X(indexes);interpy=S(indexes);
sp=polyfit(interpx,interpy,order);
sp1=polyder(sp);sp2=polyder(sp1);
tt1=polyval(sp1,originx);tt2=polyval(sp2,originx);
%  t1=tt1;t2=tt2;
%    PP(i)=t1;
%    DDSS(i)=t2;
  t1(i)=tt1;
  t2(i)=tt2;
end
  t1=fixborders(X,t1,1,2)'; %
  t2=fixborders(X,t2,1,2)';
  PP=t1';
  DDSS=t2';
return;
 %my_sp_extended=my_sp;
%fnxtr(my_sp,2); %linear outside domain
  %CC(1)=fnval(my_sp_extended,zx(1));
 %CC(nump)=fnval(my_sp_extended,zx(nump));
%siz=max(size(S));
%pvalues=0.99; %optimal value
%weights=[1 1 ones(1,siz-4) 1 1];
%my_sp=csaps(X,S,pvalues,[],weights);          % smoothing aps
%my_sp=spapi(4,X,S); % 3rd order spline (cubic)
```

```
siz=max(size(S));
pvalues=0.99; %optimal value
weights=[1 1 1*ones(1,siz-4) 1 1];
%my_sp=csaps(X,S,pvalues,[],weights);
my_sp=csapi(X,S); % no blending
sp1=fnder(my_sp,1);
sp2=fnder(my_sp,2);
cub=csapi(X,S);
cub1=fnder(cub,1);
cub2=fnder(cub,2);
dasdsadsa
blend=[0 0 0 0 0 ones(1,siz-10) 0 0 0  0 0]';
%must be between 0-->cubic and 1--->smoothed
PP=fnval(sp1,X).*(blend)+fnval(cub1,X).*(1-blend);
DDSS=fnval(sp2,X).*(blend)+fnval(cub2,X).*(1-blend);
%must blend in csplines with smoothed ones at borders
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
siz=max(size(X));
interpx=zeros(1,points);interpy=interpx;
for i=1:siz
originx=X(i);
indexes=get_n_neighbours(points,X,originx);
interpx=X(indexes);interpy=S(indexes);
sp=polyfit(interpx,interpy,order);
sp1=polyder(sp);sp2=polyder(sp1);
t1=polyval(sp1,originx);t2=polyval(sp2,originx);
PP(i)=t1;
DDSS(i)=t2;
end
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Q=calcQP2_splines(X,RH,prevRH,prevQ)
delt=RH-prevRH;
%rescale
pvalues=0.99-(delt>0)*0.5;
pvalues=0.99;
%siz=max(size(X));
Q=0;

%my_sp=spap2(1,3,X,RH); % least sq quadratic
%my_sp=spapi(4,X,RH); % 3rd order spline (cubic) %continuous 2nd
%my_sp=csapi(X,RH); % cubic spline
%my_sp=spaps(X,RH,0);% smoothing aps
```

```
siz=max(size(RH));
%pvalues=0.99; %optimal value
pvalues=0.99; %optimal value
weights=[1 1 ones(1,siz-4) 1 1];
 my_sp=csaps(X,RH,pvalues',[],weights);         % smoothing aps
%my_sp=spap2(1,3,X,RH); %leastsq quadratic->correct behaviour for single gaussians
my_sp=csapi(X,RH); % cubic spline
%my_sp=spaps(X,RH,10^-3);
 my_sp_extended=my_sp;
 %fnxtr(my_sp,3); %quadratic outside domain

 %CC(1)=fnval(my_sp_extended,zx(1));
 %CC(nump)=fnval(my_sp_extended,zx(nump));
sp1=fnder(my_sp_extended,1);
sp2=fnder(my_sp_extended,2);
t1=fnval(sp1,X);t2=fnval(sp2,X);
Q=-1/2*(t2+t1.^2);
%Q=Q;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
siz=max(size(X));
Q=0;
interpx=zeros(1,points);interpy=interpx;
for i=1:siz
%order=6;
%if(i<floor(points/2))|(i>(siz-floor(points/2)))
%order=2;
%end;
originx=X(i);
originy=RH(i);
% get n closest points
indexes=get_n_neighbours(points,X,originx);
interpx=X(indexes);interpy=RH(indexes);
sp=polyfit(interpx,interpy,order);
sp1=polyder(sp);sp2=polyder(sp1);
t1=polyval(sp1,originx);t2=polyval(sp2,originx);
tempQ=-1/2*(t2+t1.^2);
Q(i)=tempQ;
Q=Q';
end
%}
function indexes=get_n_neighbours(n,X,originx)
% gets indexes to nearest n neighbours..n should be less than sizeX
[temps tempi]=sort(abs(X-originx));
```

```
result=tempi(1:n);
indexes=sort(result)';
%  return
siz=max(size(X));
me=tempi(1);
idx=1:n;
idx=me+idx-ceil(n/2);
adjl=min(idx(1)-1,0);
adjr=max(idx(n)-siz,0);
index=idx-adjl-adjr;
%siz=max(size(X));
%if(indexes(1)>1)&(indexes(n)<siz)
%indexes=(result(1):result(1)+n-1)-floor(n/2);
%end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```