Investigation into the Properties and Application of the Diffusion Monte Carlo Method

by

Ogheneovie Orieka

Submitted to the

Department of Mechanical Engineering

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2014

© 2014 Massachusetts Institute of Technology. All rights reserved.

Signature redacted

Signature of Author: ________________________________

Department of Mechanical Engineering

June 6, 2013

Signature redacted

Certified by: ________________________________

Nicolas Hadjis Constantinou

Professor of Mechanical Engineering

Thesis Supervisor

Signature redacted

Accepted by: ________________________________

Anette Hosoi

Professor of Mechanical Engineering

Undergraduate Officer
Investigation into the Properties and Application of the Diffusion Monte Carlo Method

by

Ogheneovie Orieka

Submitted to the Department of Mechanical Engineering on May 27, 2014 in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Mechanical Engineering

ABSTRACT

This paper shall be a discussion of the properties of the Diffusion Monte Carlo (DMC) method and its applications. The discussion shall cover the basic theory behind the algorithm and the class of problems it is designed to solve: quantum systems. The algorithm will prove to be adept in producing the ground state energy and ground state wave function for systems governed by the Schrodinger equation. In this paper, familiar problems like the ground state energy of the harmonic oscillator and the Morse oscillator shall be solved using the DMC method and compared to their well-known analytic results. This will support the argument that the DMC may be used to solve problems with which the ground state energy and ground state wave function are unknown. The program outlining its use in solving these problems may be found in the accompanying appendix.

Thesis Supervisor: Nicolas Hadjiconstantinou
Title: Professor of Mechanical Engineering
Table of Contents

Abstract 3
Table of Contents 4
List of Figures 5
1. Introduction 6
2. Schrödinger Equation 6
3. Monte Carlo Method 10
4. Diffusion Monte Carlo 10
   4.1 Derivation 10
   4.2 Applications 15
   4.3 Potential Errors 15
5. Results 18
6. Conclusion 20
7. Appendices 21
   Appendix A: Input 21
   Appendix B: InitializeReplicas 22
   Appendix C: Walk 22
   Appendix D: Branch 22
   Appendix E: Count 23
   Appendix F: Output 23
8. Bibliography 24
## List of Figures

<table>
<thead>
<tr>
<th>Figure 1:</th>
<th>Ground State Energy</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2:</td>
<td>Ground State Wavefunction</td>
<td>19</td>
</tr>
</tbody>
</table>
Introduction

Monte Carlo methods have proven to be useful in a wide variety of situations. One of these is determining the properties of quantum systems. These algorithms are known as Quantum Monte Carlo methods (QMC). Four notable types of QMCs are the Variational Monte Carlo method (VMC), Green’s Function Monte Carlo method (GFMC), Path Integral Monte Carlo method (PIMC), and the Diffusion Monte Carlo method (DMC). The methods all have their advantages and disadvantages over each other. For instance, the DMC is computationally more complex than the VMC but it can work with larger time steps than the GFMC. The DMC method works by simulating the random motion of particles and whether the particles multiply or disappear. It can derive the ground state energy and ground state wave function of the quantum system in question. In this paper, an overview of the DMC method will be given, starting with the Schrodinger equation, then Monte Carlo methods, and then the DMC method. Then the algorithm will be described in more detail and the solution for the harmonic oscillator shown.

Schrodinger Equation

One of the most important equations in physics, the Schrodinger equation gives a way to describe the quantum attributes of a system. It is the equivalent for quantum mechanics of Newton’s Second Law for classical mechanics. From it, the wave function of a system can be determined, which gives the probability of finding certain particles at a position at a time. A linear partial differential equation, its general statement is:

\[ H\psi = E\psi \]
where $H$ is the Hamiltonian operator, $E$ represents the energy of the state, and $\Psi$ represents a wave function. Here, $\Psi$ is considered an eigenfunction much like in linear algebra and $E$ is its eigenvalue. Knowing this, the above equation can be described as: “when the Hamiltonian operator $H$ acts on the wave function $\Psi$, the result is the wave function multiplied by its energy state $E$. $\Psi$ may now be known as a stationary state”.

The above equation is a formulation describing stationary states. For a particle moving in an electric field, it may be formulated as:

$$E\Psi = \left(\frac{-\hbar^2}{2m} \nabla^2 + V\right)\Psi$$

where the $V$ term stands for potential energy and the other stands for kinetic energy. They both make up the Hamiltonian in that equation.

The DMC method focuses not on the time independent Schrödinger equation like the above but on time dependent formulations:

$$H\Psi = \frac{i\hbar}{\hbar} \frac{\partial}{\partial t} \Psi$$

The equation for a particle moving in an electric field is:

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

The above equation shall be the main equation of interest for the DMC method. As mentioned above, we are interested in the motion of particles which justifies its use over the stationary formulation. Since the equation is of such great import, let us now attempt to justify using it.
Suppose we have a wavefunction for a free particle, and we would like to create a wave equation that would explain its motions – and that would work in other cases. If we assume that the free particle has no potential energy, then we can describe its wavefunction as

\[ \Psi(x, t) = e^{i(kx - \omega t)} \]

and a kinetic energy

\[ E = \hbar \omega = \frac{\hbar^2 k^2}{2m} \]

Noting the following two equations:

\[
\frac{\partial}{\partial t} \left( e^{i(kx - \omega t)} \right) = -i\omega e^{i(kx - \omega t)}
\]

\[
\frac{\partial^2}{\partial x^2} \left( e^{i(kx - \omega t)} \right) = -k^2 e^{i(kx - \omega t)}
\]

we may now derive that

\[ -i\hbar \frac{\partial}{\partial t} \left( e^{i(kx - \omega t)} \right) = \frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \left( e^{i(kx - \omega t)} \right) \]

which implies that

\[ -i\hbar \frac{\partial}{\partial t} \Psi = \frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \Psi \]

To generalize this to problems where the potential energy is not equal to 0, we may add a potential energy term to the equation:

\[ -i\hbar \frac{\partial}{\partial t} \Psi = \frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \Psi + \nabla \Psi \]
The above equation may of course be generalized to three dimensions, giving us the time
dependent formulation of the Schrödinger equation.

As an aside, the time independent formulation of the Schrödinger equation may be derived from
the time dependent formulation. Taking a separation of variables

$$\psi(x, t) = \phi(x) \chi(t)$$

through substitution into the time dependent formulation, one will finally get

$$E\phi(x) = \left(-\frac{\hbar^2}{2m} \nabla^2 + V \right)\phi(x)$$

which is now time independent.

The interesting thing to note about these equations is that they can apply to things and
phenomena not necessarily considered as part of quantum mechanics. The Hamiltonian can be
split into kinetic and potential parts. What specific equation goes into \( V \) depends on what subject
is presently being studied or analyzed. For instance, the harmonic oscillator is described by:

$$V = \frac{1}{2} m \omega^2 x^2$$

and a Morse oscillator is described by:

$$V = D(e^{-2ax} - 2e^{-ax})$$

Monte Carlo Method

Let us now discuss the Monte Carlo method. It is a method of analysis that relies on generating
random variates to solve problems dependent on what the resulting probability distribution
means. For instance, the classic problem of using darts to determine the area of an irregular shape is certainly an example of the method. If a known area encloses the irregular shape, and a set of darts are thrown randomly into the area, then it becomes possible to estimate the area of the shape. To generalize, one may, using a random sample, declare that parts of data that meet a requirement are a 1 and failure to meet the requirement is a 0 and determine the percentage of successes. Although the distribution will be random at first, the proportions will quickly converge to the correct probabilities. This is applicable to a whole host of problems — from probability studies to problems in the field of engineering like reliability testing or simulating how users may use a product.

Monte Carlo methods are different from most other numerical computational methods in that the user need not even write down the ordinary or partial differential equation used to govern the system. However, the user must know some probability density function that governs the system. From the method, the user may sample randomly and then produce the mean and standard deviation of the results, which if done properly will lead to a meaningful answer.

The components of a Monte Carlo algorithm, broadly speaking, are as follows:

1. Probability density functions to describe the system
2. A random number generator to produce the random numbers
3. Sampling rules for sampling from the probability density functions
4. Scoring to keep track of the outcome
5. Error estimation to give a more complete picture of the outcome obtained
6. Variance reduction techniques for better results
7. Parallelization and vectorization techniques for faster results
Monte Carlo methods have a few advantages. As discussed earlier, to implement the method, the user does not need to know exactly what the domain of the solutions are. The user need not know its shape, for instance. All the user needs to have is the ability to check if a possible solution is in the correct domain or not to use the Monte Carlo method. The Monte Carlo method also does not require smoothness for convergence. Nor does the rate of convergence slow down as the method is generalized to higher dimensions. The method also produces an estimated answer very quickly, which can be a boon for projects with relatively high error margins.

**Diffusion Monte Carlo**

**Derivation**

To begin with, let us look again at the time dependent formulation of the Schrödinger equation:

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

An astute reader may note the fact that the above equation looks just like the classic diffusion equation

\[
\frac{\partial u}{\partial t} = \nabla^2 u + Fu
\]

It was in observing this form that the idea of simulating the problem through random walks came about – a similar process to diffusion, and the name Diffusion Monte Carlo is born.

At this point, we must now note that the general solution to the one-dimensional form of the time independent Schrödinger equation
\[
-i\hbar \frac{\partial}{\partial t} \psi = \frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \psi + V\psi
\]

is of the form

\[
\psi(x, t) = \sum_{n=-\infty}^{\infty} c_n \phi_n e^{-\frac{i E_n}{\hbar} t}
\]

We can now modify the time dependent formulation of the Schrödinger equation by a shift of the energy scale

\[
-i\hbar \frac{\partial}{\partial t} \psi = \frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \psi + (V - E_R)\psi
\]

where \(E_R\) is the reference energy of the system. This leads to the solution becoming

\[
\psi(x, t) = \sum_{n=-\infty}^{\infty} c_n \phi_n e^{-\frac{i (E_n - E_R) t}{\hbar}}
\]

Next, we can do a Wick rotation, which is simply transforming the equations from real time to imaginary time. This is done by introducing \(\tau = it\), leading to

\[
\hbar \frac{\partial}{\partial \tau} \psi = \frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \psi + (V - E_R)\psi
\]

and the solution

\[
\psi(x, t) = \sum_{n=-\infty}^{\infty} c_n \phi_n e^{-\frac{E_n - E_R}{\hbar} \tau}
\]

Knowing that \(E_0\) is the lowest ground state energy, we can note that having \(E_R > E_0\) means that the wavefunction solution grows infinitely fast, while \(E_R < E_0\) means that the wavefunction
would go to 0. Only when $E_R = E_0$ will there be convergence. This feature is used by the DMC to produce the lowest ground state energy, since it'll be the only one it can converge on.

In its present form, the equations are not very useful. The Wick rotated equation may be integrated using path integral formalism. This comes from the Feynman path integral solution of the time-independent Schrodinger equation. Through this, the wave function can be turned into a multidimensional integral suitable for solution by the Monte Carlo method. This paper will not delve into the specifics, but it will note a final result:

$$
\Psi(x, t) = \lim_{N \to \infty} \int_{-\infty}^{\infty} \left( \prod_{j=0}^{N-1} dx_j \right) \prod_{n=1}^{N} W(x_n) \times P(x_n, x_{n-1}) \Psi(x_0, 0)
$$

where $P$ is a probability density of the random variable $x_n$ with mean $x_{n-1}$ and variance $\sigma = \sqrt{\hbar \Delta \tau / m}$. $W$ is a weight function

$$
W = \exp\left(- \frac{(V(x_n) - E_R)\Delta \tau}{\hbar} \right)
$$

The above integral can most ideally be solved with the Monte Carlo method. It may help to reinterpret the integrand as follows:

$$
W(x_N)P(x_N, x_{N-1}) \ldots W(x_1)P(x_1, x_0) \Psi(x_0, 0)
$$

$\Psi(x_0, 0)$ here is the initial wavefunction of the problem, which signifies the initial distribution of the particles to be part of the DMC method. The successive positions of the particles will be determined by

$$
x_n = x_{n-1} + \sigma \rho_n
$$

where $\rho_n$ is a normal distribution random variable.
The process can now begin a birth-death process, in which every particle is replaced with \( m \) number of particles, governed by

\[
m_n = \min(\lfloor W(x_n) + u \rfloor, 3)
\]

If \( m = 0 \), the particle is destroyed, if \( m = 1 \), the particle survives, if \( m = 2 \) then a new particle is born starting at position \( x_n \), and if \( m = 3 \), two new particles are born starting at position \( x_n \). A maximum of 2 new particles is put in place to prevent instabilities. The combination of the diffusion and the birth-death process will ensure that the distribution of the particles will match that of the ground state wavefunction.

One of the key elements in judging whether a particle lives or dies is the reference energy \( E_R \). The number of particles should change to ensure that the reference energy matches the ground state energy after enough iterations. To ensure this, first the average potential energy of the particles \( \langle V \rangle \) must be determined

\[
\langle V \rangle_i = \frac{1}{N_i} \sum_{j=1}^{N_i} V(x_i^{(j)})
\]

where \( N_i \) is the total number of particles at iteration \( i \).

With \( \langle V \rangle \), \( E_R \) is now

\[
E_R = \langle V \rangle + \frac{\hbar}{\Delta \tau} \left( 1 - \frac{N}{N_0} \right)
\]

where \( \Delta \tau \) is the timestep used. When \( E_R \) is stable, it will be equal to the ground state energy.

Now both the ground state energy and wavefunction can be found.
Applications

The DMC can be used in a large variety of applications. This paper shall demonstrate its use with a harmonic oscillator, but it has been used to investigate the following:

- Three dimensional electric gas
- Structure of nuclei
- Pairing in ultra-cold atomic gases
- Band structures of oscillators
- Defects in semiconductors
- Equations of states of solids

Potential Errors

No numerical algorithm is perfect, and the DMC has its share of flaws that could introduce errors into the analysis. These include

1. Statistical errors
2. Time step bias due to short time steps which may affect results
3. Population control bias due to the fluctuating number of particles.

Algorithm

One of the key things to do before starting the actual algorithm is to non-dimensionalize the equations. To do this, each physical quantity shall be represented as itself and its unit (e.g. $t \rightarrow tT$) and using $L, T,$ and $E$,

$$\frac{\partial}{\partial t} \Psi = \frac{\hbar T}{2mL^2} \frac{\partial^2}{\partial x^2} \Psi - \frac{TE}{\hbar} (V - E_R) \Psi$$
If $L, T,$ and $E$ are chosen such that

$$\frac{hT}{2mL^2} = \frac{1}{2}$$

$$\frac{TE}{\hbar} = 1$$

then the Schrödinger equation becomes

$$\frac{\partial}{\partial \tau} \Psi = \frac{1}{2} \frac{\partial^2}{\partial x^2} \Psi - (V - E_R) \Psi$$

This will give us new equations like:

$$W = \exp(-(V - E_R))$$

$$x_1 = x_0 + \sqrt{\Delta \tau} \cdot \rho$$

The actual algorithm can be split into a few steps:

1. Input: This is where important parameters like the initial and final number of replicas $N_0$ and $N_{max}$ (formerly known as particles), the number of time steps to run the simulation $\tau_0$, the duration of the timesteps $\Delta \tau$, limits of the coordinates for the spatial sampling of the replicas $x_{min}$ and $x_{max}$, and the number of special boxes that the replicas will be sorted in $n_b$. Here we used $N_0 = 500$, $N_{max} = 2000$, $\tau_0 = 1000$, $\Delta \tau = .1$, $x_{min} = -20$, $x_{max} = 20$, and $n_b = 200$.

2. Initialize replicas: Creating the matrix $psips$ that will house the replicas and their statuses. $psips$ will have rows $i$ and columns $j$. The rows will store the status of each replica. It will be 1 if the replica is alive, and 0 if it is dead. The columns will have one reserved for the statuses and the rest devoted to storing the coordinates of the replica.
number of columns depends on the number of dimensions the coordinates need, but it will be number of dimensions plus one.

3. **Walk**: The displacement of the replicas are changed through the stochastic process. The present position of each replica will get \( \sqrt{\Delta t} \cdot \rho \) added, which will move them in different directions. As mentioned before, \( \rho \) is a normally distributed random variable with mean 0 and standard deviation 1.

4. **Branch**: The decisions on whether certain replicas live or die happen here. Once \( m \) has been calculated, the decision will then take place. As mentioned before, \( m = 0 \) means death for the replica, \( m = 1 \) will ensure it is left alone, \( m = 2 \) ensures one replication and \( m = 3 \) ensures two.

5. **Count**: After the reference energy has converged with the ground state energy, the number of replicas in each spatial box may be cumulatively tallied and normalized to construct the ground state wave function. Cumulative addition should provide a clearer result that taking a single snapshot in time.

6. **Test**: Walk and Branch will be used in the cumulative tally, and Test controls them.

7. **Output**: Shows the results of the program. In particular, the ground state energy, the standard deviation of that energy, the time evolution of the energy and the spatial distribution of the replicas.

**Results**

The DMC method was implemented for the one-dimensional harmonic oscillator. The potential energy once again is,

\[
V = \frac{1}{2} m \omega^2 x^2
\]
Once non-dimensionalized, the ground state energy $E_0 = .5$ and the ground state wave function $\phi_0(x) = \pi^{-1/4} \exp(-\frac{x^2}{2})$.

Our numerical result for the ground state energy is shown below:

![DMC Simulation of the Reference Energy of a Harmonic Oscillator](image)

*Figure 1: Ground State Energy*

Here, the simulation converges fairly quickly and confirms the theoretical answer of .5.
The result of the ground state wave function is shown below:

![Graph showing DMC Simulation of the Ground State Wavefunction of a Harmonic Oscillator](image)

*Figure 2: Ground State Wavefunction*

The wavefunction simulated here does not match expected results. This could mean a problem in the Walk routine or Count routine, or perhaps the Branch routine. The figure has the right shape for a function in the class of \( \exp(-\frac{x^2}{2}) \), but it is not the correct ground state wavefunction.

The specific code written in Matlab and used for the simulation is attached as an appendix to this paper. The execution differs at points in attempts of simplification, but it covers all grounds. Future work would be to optimize the code and eliminate some loops as the program is slow, or perhaps write it in a faster language like C++.
Conclusion

The DMC method is a powerful method for solving an interesting class of problems. It has its weaknesses, but the strengths of the algorithm have made it very popular. While somewhat difficult to execute properly, it can provide very accurate answers if implemented correctly.
Appendices

Appendix A: Input

% Setting constant values
NO = 500;
NMax = 2000;
t0 = 1000;
deltaT = .1;
xMin = -20;
xMax = 20;
nb = 200;
hbar = 1.054571726e-47;

E_R = -.5;

E = []; E(1) = E_R; % Contains all reference energies
Emean = []; Emean(1) = E_R; % Contains averaged reference energies

psips = InitializeReplicas(NO,NMax);

% Birth and Death
for counter = 1:t0/deltaT
    psips = Walk(psips,deltaT);
    [E_R, psips,NFirst,NSecond] = Branch(psips,E_R,deltaT,NMax);
    E(counter+1) = E_R;
    Emean(counter+1) = mean(E);
    if NFirst == 0 || NSecond == 0 % Termination if all replicas are dead
        break
    end
end

% Ground State Wave function
counts = zeros(nb,1);
for counter = t0/deltaT
    psips = Walk(psips,deltaT);
    [E_R, psips,NFirst,NSecond] = Branch(psips,E_R,deltaT, NMax);
    [binranges, bincounts] = Count(psips,xMin,nb,xMax);
    counts = bincounts+counts;
end
normcounts = counts./norm(counts); % Normalization
plot(binranges,normcounts);

% Energy
[E0, Estd] = Output(E_R,Emean);
Appendix B: InitializeReplicas

function psips = InitializeReplicas(N0,NMax)
% Called by Input to make psips

psips = zeros(NMax,2);
psips(1:N0,1) = 1;

Appendix C: Walk

function psips = Walk(psips,deltaT)
% Process of distributing replicas

for counter = 1:length(psips)
    if psips(counter,1) == 1
        psips(counter,2) = psips(counter,2) + sqrt(deltaT)*randn; % random walk
    end
end

Appendix D: Branch

function [E_R, psips,NFirst,NSecond] = Branch(psips,E_R,deltaT, NMax)
% Begins the birth-death process and preliminary calculations of E0

hbar = 1.054571726e-47;
NFirst = sum(psips(:,1)); %Finding the initial number of replicas
lastmatrix = find(psips(:,1));
lastentry = lastmatrix(end); % Determining a stopping point
lastnumber = lastentry; % Need a stopping point that won't change
Vtotal = []; % Will hold potential energy to calculate average

%Birth-death Process
for counter = 1:lastnumber
    if psips(counter,1) == 1 % Check that it's alive
        V = .5*(psips(counter,2)^2); %non-dimensionalized harmonic oscillator
        Vtotal = [Vtotal V];
        W = exp(-(V - E_R));
        r = rand;
        m = min(fix(W + r),3);
        if m == 0
            psips(counter,1) = 0; % dead
        elseif m == 1
            Vtotal = [Vtotal V];
        elseif m == 2
            Vtotal = [Vtotal V V];
            for secondcounter = 1:length(psips)
            end
        end
    end
end
if pips(secondcounter,1) == 0
    pips(secondcounter,1) = 1;
    pips(secondcounter,2) = pips(counter,2);
    break
end
elseif m == 3
    Vtotal = [Vtotal V V V];
for thirdcounter = 1:2
    for secondcounter = 1:length(pips)
        if pips(secondcounter,1) == 0
            pips(secondcounter,1) = 1;
            pips(secondcounter,2) = pips(counter,2);
            break
        end
    end
end
end
if lastnumber > NMax %To prevent an infinite loop
    break
end
end
NSecond = sum(pips(:,1));

%Calculating E
if ~isempty(Vtotal)
    Vavg = sum(Vtotal)/NSecond;
    E_R = Vavg + hbar/deltaT*(1 - NSecond/NFirst); % Linearization
end

Appendix E: Count

function [binranges, bincounts] = Count(pips, xMin, nb, xMax)
% Counts for the ground state wave function
binranges = linspace(xMin, xMax, nb)'; % Setting the histogram bins
bincounts = histc(pips(:,2), binranges); % Counting the number of replicas in the right bins

Appendix F: Output

function [E0, Estd] = Output(E_R, Emean)
% Produces final output of E
E0 = E_R;
Estd = std(Emean); % standard deviation
figure(2); plot(Emean); % plot of energy

23
Bibliography


Nave, R. *Schrodinger Equation.* Retrieved from http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/schr.html


