Path Integral Ground State Quantum Monte Carlo Calculations for a Double Well Potential Surface

Britni Ratliff
Department of Chemistry, University of Tennessee
bratlif1@utk.edu

Introduction

Tunneling is a quantum phenomenon that occurs when a particle encounters a potential energy barrier. The incident particle can either tunnel through the barrier or be reflected from the barrier. The probability of transmission through the barrier by tunneling decreases exponentially as the mass and thickness of the barrier increases (Atkins, 2002). Tunneling is especially significant when analyzing multiple well potential energy surfaces. For example, in a double-well potential energy surface with a ground state energy level below the potential energy barrier (Fig.1), the molecule can convert between the two local minima by tunneling through the energy barrier. In this case, the particle never actually achieves enough energy to go over the barrier and instead goes through it to reach the other side (Fig. 2). The umbrella vibration of NH_3 is one example of a double-well potential energy surface in which the molecule tunnels between the two local energy minima (Fig. 3). Since tunneling impacts a molecule's rotational and vibrational constants, it also affects the resonance frequencies of the molecule.

High resolution telescopes are used to monitor the emission spectra of molecules in interstellar space. The observed resonance frequencies are identified by comparing them to a library of experimental and theoretical results. However, scientists observe many resonance frequencies that do not correspond to any known spectra. A significant portion of these frequencies are in the radio frequency region of the electromagnetic spectrum. Radio frequency emissions are most often a result of the small difference in energy between two energy levels split due to tunneling. If an efficient way were known to accurately calculate the vibrational wavefunction and hence the vibrational energy levels, theoretical predictions could help guide astronomers in identifying these unknown resonance frequencies. Scientists could predict a molecule that might exist in space and calculate the resonance frequencies that would be observed from this molecule. If the predicted frequency matched an observed frequency,

synthetic chemists could then work to create the molecule, measure its resonance frequencies, and experimentally confirm its existence in interstellar space.

The total number of vibrational degrees of freedom (DOF) of a molecule is given by:

$$\underbrace{3N}_{\text{Total DOF}} - \underbrace{3}_{\text{Translational DOF}} - \underbrace{\begin{cases} 2 \text{ linear} \\ 3 \text{ nonlinear} \end{cases}}_{\text{Rotational DOF}} = \text{Total \# Vibrational DOF}$$

where N is the number of atoms in the molecule. The wavefunction that describes the vibrational motion of the molecule will have the same number of dimensions as it has vibrational degrees of freedom. For a diatomic molecule, there is only one vibrational degree of freedom. There are several efficient methods for solving the one-dimensional vibrational Schrödinger equation, including the Numerov-Cooley method. However, when a molecule has more than two atoms, the vibrational Schrödinger equation becomes multi-dimensional. If it is assumed that there is no coupling between the degrees of freedom, the multi-dimensional Schrödinger equation can be broken down into a system of first order Schrödinger equations. In reality, all non-trivial multiple dimensional systems involve coupling between the degrees of freedom. The computational effort required to solve the multi-dimensional vibrational Schrödinger equation using standard methods increases exponentially with the number of vibrational degrees of freedom.

Quantum Monte Carlo allows systems involving a large number of vibrational dimensions to be solved with significantly less effort. Quantum Monte Carlo (QMC) is a computational technique that applies algorithms of classical mechanics to solve quantum mechanical problems that are not easily solved using other numerical methods. Since QMC is a relatively new method, experiments are still being conducted to determine which molecules, potential energy surfaces, and wavefunctions can be accurately solved using this technique. Monte Carlo calculations differ from other simulation methods by being stochastic, or using a random number generator. Most Monte Carlo calculations depend on random walks where the program makes a series of random moves, usually moving towards lower values of the energy but sometimes moving against the gradient. An algorithm is then used to accept or reject this new configuration. Depending on the type of QMC being used, a specific function is applied to the old and new configuration. The move is either accepted or rejected by comparing the results of this function before and after the move. With QMC, the computational effort increases

linearly with the number of vibrational degrees of freedom making QMC more efficient than standard approaches for solving the multi-dimensional Schrödinger equation.

However, QMC can only be used to find the lowest energy level of a system with the specified node symmetry pattern and is not very accurate for solving multiple node problems. This makes QMC most useful in zero- or one-node problems, which correspond to the ground and first excited states of a molecule. The Boltzmann distribution confirms that at low temperatures, these are the energy levels that are most heavily occupied. Fortunately, these are the two energy levels that tunneling impacts the most significantly.

Overview of Path Integral Ground State

The path integral ground state (PIGS) method (Sarsa, 2000) is a QMC computational technique used to calculate ground state quantities. This method differs from other QMC simulation methods by allowing ground state properties to be calculated without the errors that result from extrapolating the numerical data. PIGS calculations utilize both the time dependent Schrödinger Equation and the time independent Schrödinger Equation

$$\hat{H}\Psi(x,t) = i\hbar \frac{\partial \Psi(x,t)}{\partial t} = E\Psi(x,t)$$

where $\hat{H}$ is the Hamiltonian operator, $\Psi(x,t)$ is the time dependent wavefunction of the particle, t is time, $\hbar$ is Planck's constant divided by 2π, and E is the energy of the particle. This differential equation can be solved to give

$$\Psi(x,t) = \underbrace{\exp(-iEt/\hbar)}_{\text{Time-dependent phase factor}}\underbrace{\psi(x)}_{\text{Time-independent solution}}$$

which shows that the time dependent wavefunction is composed of a time-independent spatial part $\psi(x)$ which solves the time independent Schrödinger equation, so that $\hat{H}\psi(x) = E\psi(x)$, and a time-dependent phase factor, $\exp(-iEt/\hbar)$. Note that the time independent solution is simply the time dependent solution at time zero

$$\psi(x) = \Psi(x, \tau = 0).$$

In terms of an imaginary time variable $\tau = i\,t$, any time dependent function can be written as a linear combination of the system's solutions to the time dependent Schrödinger equation

$$P(x,\tau) = \sum_{n=1}^{\infty} c_n \Psi_n(x, \tau = 0) \exp(-E_n \tau / \hbar)$$

where $P(x,\tau)$ is the linear combination after the passage of τ units of imaginary time, n labels the energy levels and wavefunctions of the system, c_n is a constant term that expresses the contribution that wavefunction Ψ_n makes to the linear combination, and E_n is the energy of level n. As n gets large, E_n of the wavefunction also increases in magnitude which exponentially decreases its overall contribution to the linear combination. When the unit of imaginary time is longer, the contributions of the excited states to the wavefunction decay more quickly leaving only information about the particle's ground state wavefunction, $c_1\Psi_1(x,\tau)$. The next to last energy level to die out is the first excited state. The linear combination must be propagated far enough in imaginary time so that the exponential factor $\exp\left(-\frac{E_2\tau}{\hbar}\right)$ is much smaller than the factor $\exp\left(-\frac{E_1\tau}{\hbar}\right)$. In order to insure that only information about the ground state remains, we need

$$\exp\left(-\frac{(E_2 - E_1)\tau}{\hbar}\right) << 1. \qquad (1)$$

In PIGS, the particle, atom, or molecule of interest is represented by a chain of N beads. Each bead represents the same particle just at a different imaginary time. A propagator is used to advance each consecutive inner bead by an additional imaginary time $\Delta\tau$. The imaginary time propagator $\hat{P}_\tau$ is an operator

$$\hat{P}_\tau = \exp(-\hat{H}\tau / \hbar)$$

and when it acts on a function, it advances that function forward in imaginary time. If we propagate an arbitrary function forward for a large amount of imaginary time, the function that results will be proportional to Ψ_1, the ground state wavefunction of our Hamiltonian operator, $\hat{H}$. However, it is difficult to compute the propagator for a large interval of imaginary time because there is not an efficient and accurate way to estimate the propagator. When the change in imaginary time $\Delta\tau$ is small, the short time approximation to the propagator can be written as

$$\exp(-\hat{H}\Delta\tau/\hbar) = \exp(-(\hat{T}+V)\Delta\tau/\hbar) \approx \exp(-V\Delta\tau/2\hbar)\exp(-\hat{T}\Delta\tau/\hbar)\exp(-\hat{V}\Delta\tau/2\hbar) \quad (2)$$

This expression is approximate because the operators $\hat{T}$ and V do not commute. The error in this estimate increases with $\Delta\tau^2$. Therefore, the propagator is most accurate when it advances the function only a small amount of imaginary time. This approximation allows us to approximate a long-time propagator as the successive action of N short-time propagators (Fig. 4). In effect, this means increasing the length of the chain. This explains why PIGS simulations can provide information on the ground state wavefunction: interior beads in the PIGS simulation have been propagated to a larger overall value of imaginary time so that contamination from higher energy wavefunctions has been gradually eliminated.

The PIGS QMC simulation of a particle in a double-well potential begins by arbitrarily placing N particles along an x coordinate axis between -1 and 1 (Fig. 5). Then, each particle is moved in succession to a new position according to $x \rightarrow x+\Delta x$ where $\Delta x=\alpha\xi$, α is a preset maximum displacement, and ξ is a random number between -1 and 1. The probability of the chain having the new configuration is then calculated by the following equation

$$P = \Psi_{tr}(x_1)\Psi_{tr}(x_n)\exp\left(-\frac{\Delta\tau}{\hbar}\sum_{k=1}^{N-1}F(x_k, x_{k+1})\right) \tag{3}$$

$$F(x_k, x_{k+1}) = \frac{V(x_k)+V(x_{k+1})}{2} + \frac{m(x_{k+1}-x_k)^2}{2(\Delta\tau)^2} \tag{4}$$

where P is the probability, Ψ_{tr} is the approximate or trial wavefunction of the particle, $\Delta\tau$ is the imaginary time step between the beads, V is the potential energy of a particle, and m is the mass of the particle. The function $\exp\left(-\frac{\Delta\tau}{\hbar}F(x_k, x_{k+1})\right)$ represents the small time step propagator from x_k to x_{k+1}. As shown in equation 2, the imaginary-time step propagator can be approximated by the product of three exponentials involving $\hat{T}$ and V. The middle portion of the short-time approximate propagator is the free-particle propagator in imaginary time:

$$\exp\left(-\hat{T}\Delta\tau/\hbar\right).$$

The free-particle propagator is also mathematically represented as

$$\exp\left(-\frac{m(x-x_o)^2}{2\hbar\Delta\tau}\right)$$

which appears in the second summand of $F(x_k, x_{k+1})$ with $-\frac{\Delta\tau}{\hbar}$ factored out in front. The first and third exponential terms in the small time step approximation to the propagator are

$$\exp\left(-\frac{V\Delta\tau}{2\hbar}\right)$$

which appears in the first summand of $F(x_k, x_{k+1})$ with $-\frac{\Delta\tau}{\hbar}$ factored out in front. The sum in equation 3 applies the propagator to the trial wavefunctions $\Psi_{tr}(x_1)$ and $\Psi_{tr}(x_N)$ a total of N-1 times. Since the middle bead is (N-1)/2 time steps away from the end beads, applying the propagator (N-1)/2 times to $\Psi_{tr}(x_1)$ yields the distribution of the wavefunction for the middle bead. Similarly, applying the propagator the remaining (N-1)/2 times to $\Psi_{tr}(x_N)$ also yields the distribution of the wavefunction for the middle bead. Since the probability distribution of a wavefunction is given by $|\Psi(x)|^2$, the entire product gives the probability distribution of the middle bead which is known to most accurately represent the equilibrium distribution of the double well potential energy surface.

The trial wavefunction used for propagation is a double Gaussian distribution

$$\Psi_{tr}(x) = e^{-a(x+b)^2} + e^{-a(x-b)^2} \qquad (4)$$

with constant a and peaks centered at $\pm b$. The values of a and b are strategically selected based on the mass of the particle. The basic potential energy function $V(x)$ used in the calculations is

$$V(x) = x^4 - cx^2 .$$

The coefficient c of the x^2 term can be varied to change the energy barrier through which the particle must tunnel.

Once a bead has been moved, the Metropolis Algorithm for Monte Carlo (Metropolis, 1953) is used to accept or reject this new configuration based on the probability weight function described above. If the probability of the new chain of beads is greater than the old probability, the simulation unconditionally accepts the move. However, if the new probability is less than the old probability, the algorithm will conditionally accept or reject the move. The ratio of the new probability to the old probability is calculated and compared to a random number generated between zero and one. If the ratio is greater than the random number, then the new configuration is accepted. If not, the simulation rejects the move. By allowing the simulation to accept a

higher energy configuration, the algorithm is given the opportunity to climb out of local minima so that the global minimum can be located. This process repeats for a preset number of iterations.

According to Equation 3, the acceptance of a new move is dependent on the position of all the other particles. Since the interior beads in the PIGS simulation are at larger values of the imaginary time, the equilibrium probability density is most accurately represented by following the position of the middle particle. The x-axis is divided into 200 bins of width ω=0.025. After each iteration, the position of the middle bead is recorded, filed into the appropriate bin, and used to compile a histogram representing the probability density of the middle bead. In order to optimize this simulation, factors such as the number of beads, number of iterations, starting position of the beads, imaginary time step, mass of the particle, maximum displacement, and the trial wavefunction are varied.

The Model System

The constant c for the double-well potential energy surface was chosen to be $c = 0.5$ so that the tunneling splitting is moderate and the ground state is below the barrier (Fig. 1). (All units are reported in atomic units hereafter.) The double well potential energy surface is a one-dimensional system for which the vibrational wavefunction can be accurately solved using the Numerov-Cooley method. The Numerov-Cooley method is known to accurately calculate the wavefunction, including tunneling effects, for a simple double well potential energy surface. The Numerov-Cooley results are used as a reference to determine if the PIGS QMC simulation program is producing accurate results. Once the PIGS QMC simulation program accurately models tunneling, it can be used for systems involving multiple atoms moving in multiple dimensions.

Accuracy of the Simulations

To help determine which parameters would generate the most accurate results using the PIGS simulation, a series of calculations were run. Since accuracy of the PIGS simulation is primarily controlled by the length of the chain N, slice of imaginary time $\Delta\tau$, and the total amount of imaginary time to the middle bead, other parameters were held constant to optimize these variables. Each bead was assigned a mass of 60 and allowed a maximum displacement of

0.50. The trial wavefunction with a=-5.5 and b=0.375 was used for both the initial and the end bead. The end beads were kept stationary at $x = \pm 1$ throughout the simulation. This forced the simulation to more accurately represent tunneling since it guaranteed that beads were distributed between -1 and 1. If the end beads had been allowed to move, the entire chain could have moved into one of the local minima thereby limiting the accuracy of the simulation. The total number of iterations varied with the number of beads, with the number of iterations per bead held constant at 10 million iterations per bead. For example, a chain of 41 beads has 39 possible moving beads which gives a total of 390 million iterations. Similarly a chain with 121 beads was run for 1,190 million iterations.

To verify that the probability distribution of the middle bead of the PIGS simulation was the most accurate representation of the equilibrium distribution of the wavefunction, every tenth bed was monitored. The probability distribution of each bead was compared to the $|\Psi(x)|^2$ calculated from the Numerov-Cooley method. For a chain with N=121 beads, Figures 7 and 8 show that as the bead increases in imaginary time from the end of the chain, the probability distributions become a more accurate representation of the equilibrium probability distribution calculated from the Numerov-Cooley method. Figure 8 focuses on the middle bead, bead 61, and the beads in the immediate vicinity. Bead 41 is included in both Figure 7 and Figure 8 to serve as a reference point. Bead 61, the middle bead, appears to be in good agreement with the Numerov-Cooley results. Theoretically, any bead other than the middle bead should have a partner on the other half of the chain that has been propagated an equivalent amount of imaginary time and has a similar equilibrium distribution that is reflected through the x=0 symmetry point. In this case, bead 51 and bead 71 were each propagated fifty times, ten steps fewer than the middle bead. It is important to note the symmetry in their probability distributions. Any slight deviation in their equilibrium distribution is most likely the result of statistical error. This variation would be eliminated if the simulation were able to perform an infinite number of iterations.

To quantitatively judge the accuracy of the distribution, the error between the distribution and that obtained from the Numerov-Cooley method was calculated. The error was calculated by summing the square of the differences between the height of each bin in the PIGS simulation and the height of the same bin in the Numerov-Cooley data. If the PIGS simulation exactly duplicated the Numerov-Cooley data, then this error would be zero. Figures 9 and 10 display the

error in the probability distribution as a function of bead number. Figure 10 focuses in on the center beads to show that bead 61 is indeed the most accurate representation of the probability of the equilibrium distribution.

The accuracy of the PIGS simulation is primarily controlled by N and $\Delta\tau$. The middle bead needs to be separated from the ends of the chain by a large amount of imaginary time which is given by $\frac{(N-1)\Delta\tau}{2}$ so that the middle bead will be distributed according to $|\Psi(x)|^2$. However, the error in the approximation to the propagator increases with $\Delta\tau^2$ so we need $\Delta\tau$ to be small to ensure each link in the chain is a good estimate of the propagator. As expected, chains involving more beads create more accurate results because the total imaginary time is increasing. As the time slice is increased, the error initially decreases since the total amount of imaginary time to the middle bead is increasing (Fig. 11). This accurately reflects the claim that as the imaginary time step increases, the contribution from the excited states decays leaving only information about the ground state wavefunction. When the time step is held constant and the length of the chain of beads is increased from N=41 to N=121, the simulations become more accurate (Fig. 13). However, when the slice of imaginary time becomes greater than 1.0, the short-time approximation to the propagator begins to become increasingly inaccurate (Fig. 12). As the imaginary time step increases from $\Delta\tau$=1.0 to 5.0, the results diverge from the Numerov-Cooley results (Fig. 14).

In order to determine if the first excited state had adequately decayed, the quantity in equation 1 was plotted against the error of the middle bead for several values of imaginary time steps (Fig. 15 and Fig. 16). The Numerov-Cooley method provided the energies for E_1 and E_2 which were -0.00691806 and 0.0460846, respectively. As expected, the smaller this quantity, the smaller the error of the equilibrium distribution of the middle bead.

Discussion

This research has focused on maximizing the accuracy of the PIGS QMC simulation. We have found that for the double-well model system considered here each imaginary time step should be less than or equal to 1.0 in order to obtain accurate results (Fig. 12 and Fig. 14). This limit on the imaginary time step is due to the limitations of the short-time approximation to the propagator. The total amount of imaginary time needed for all of the excited states to decay

leaving only information about the ground state is obtained from $\exp\left(-\frac{(E_2 - E_1)\tau}{\hbar}\right)$. This quantity must be less than or equal to 0.2 for the error in the middle bead distribution to be below 10^{-4} (Fig. 16). Experimental data from interstellar space directly measures the energy gap between the ground and first excited state of the molecule and therefore equation 1 can be solved for the total amount of imaginary time τ needed. These two things, the size of the time step combined with the total amount of imaginary time needed, tell us the minimum number of beads needed for an accurate PIGS QMC simulation.

Future research could focus on maximizing both the efficiency and accuracy of the simulation. The total computational time for a simulation is proportional to the number of beads N and number of iterations L. It is desirable to minimize the amount of time needed to get accurate results. In future research the lower bound on L would be found. It is possible that there might be a relationship between the bead displacement Δx and L. It seems that using a small Δx would require more iterations in order to obtain an accurate equilibrium distribution while a large Δx might cause the simulation to overshoot the fine details in the probability distribution in the tunneling region. Further calculations are needed to optimize these parameters.

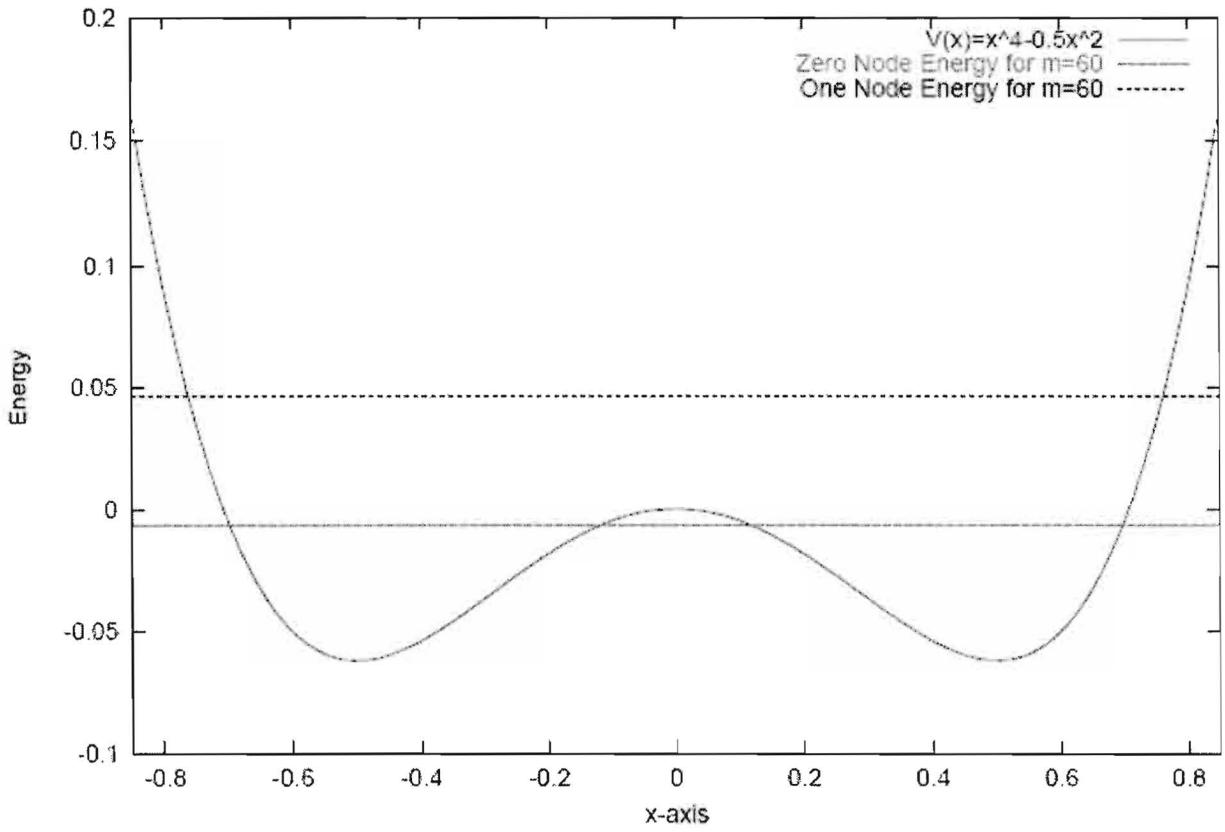


Figure 1

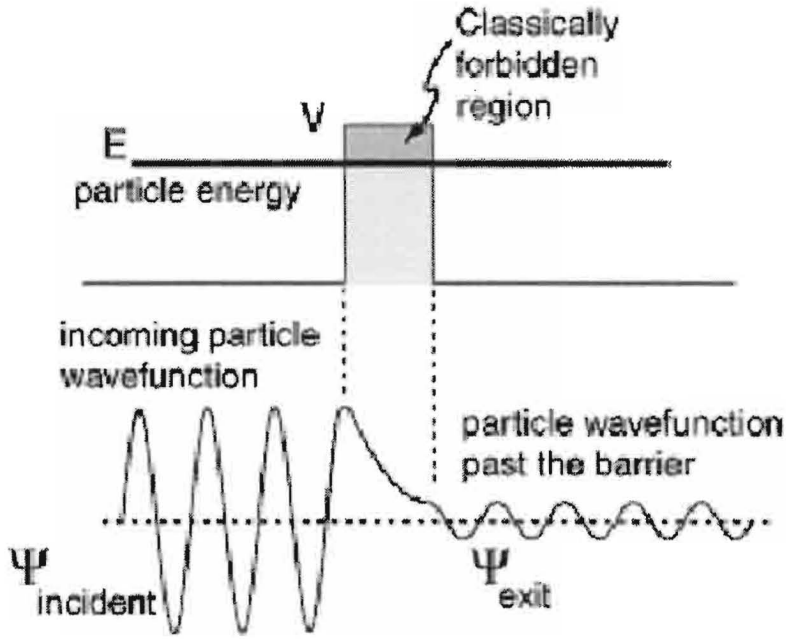


Figure 2 (Nave, 2005)

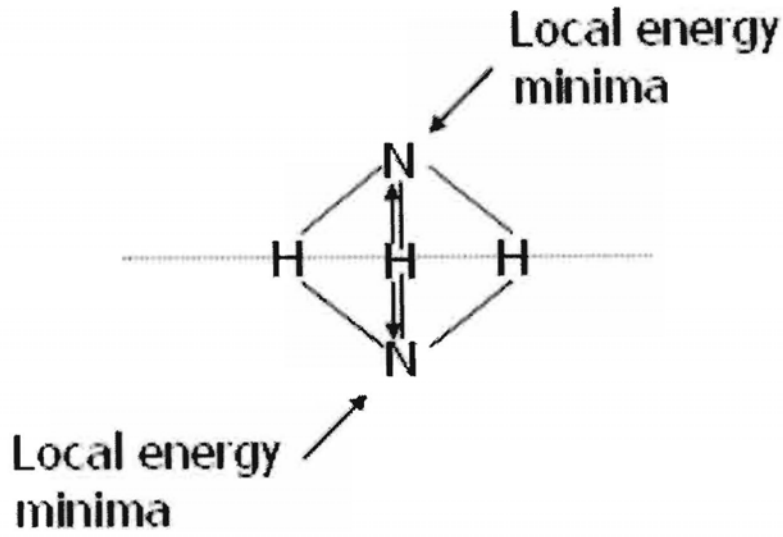


Figure 3

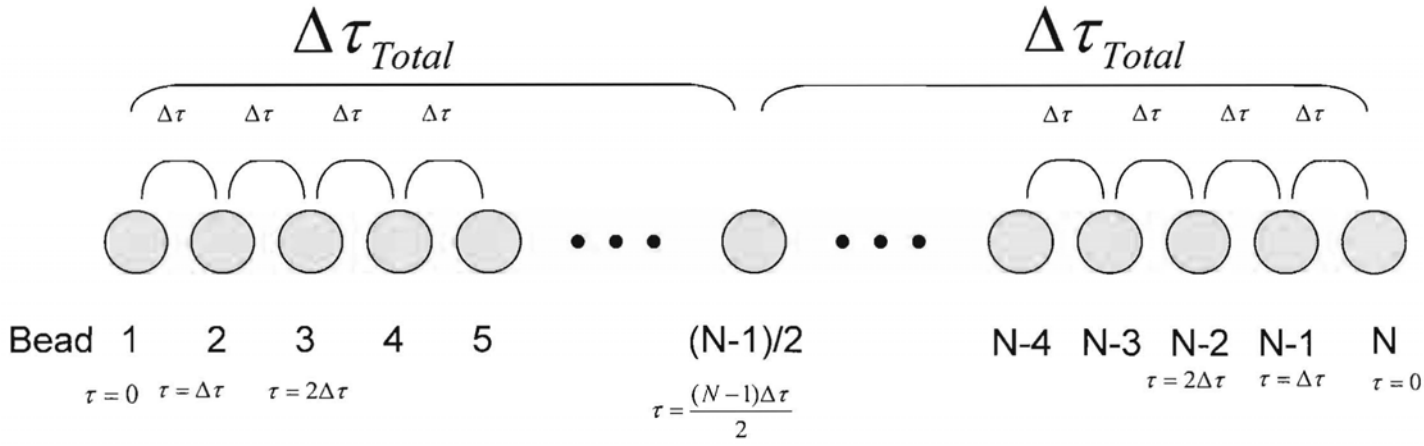
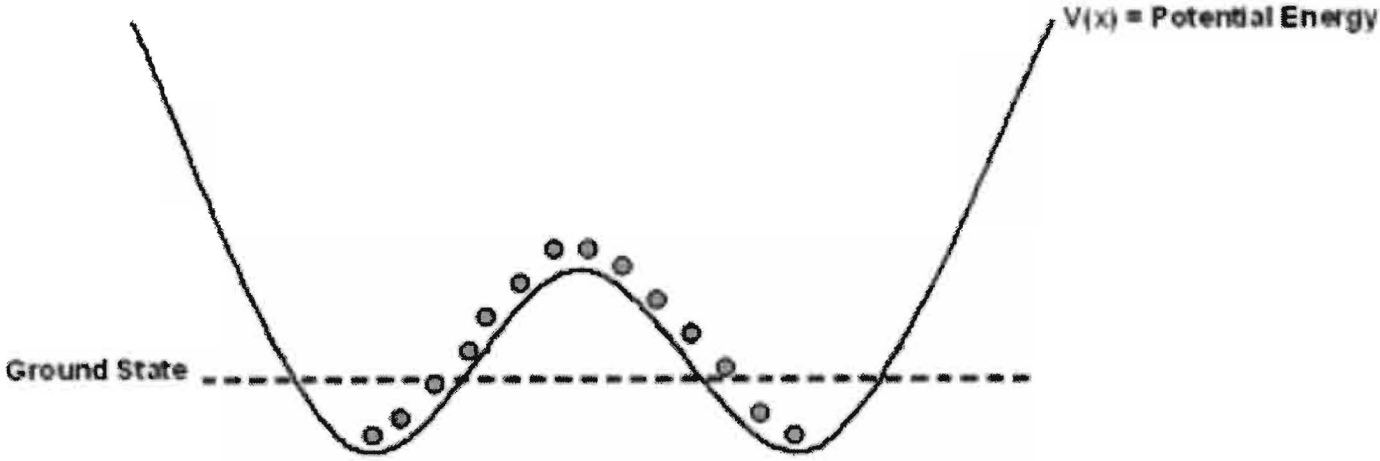


Figure 4



$$V(x)=x^4 - 0.5\,x^2$$

Figure 5

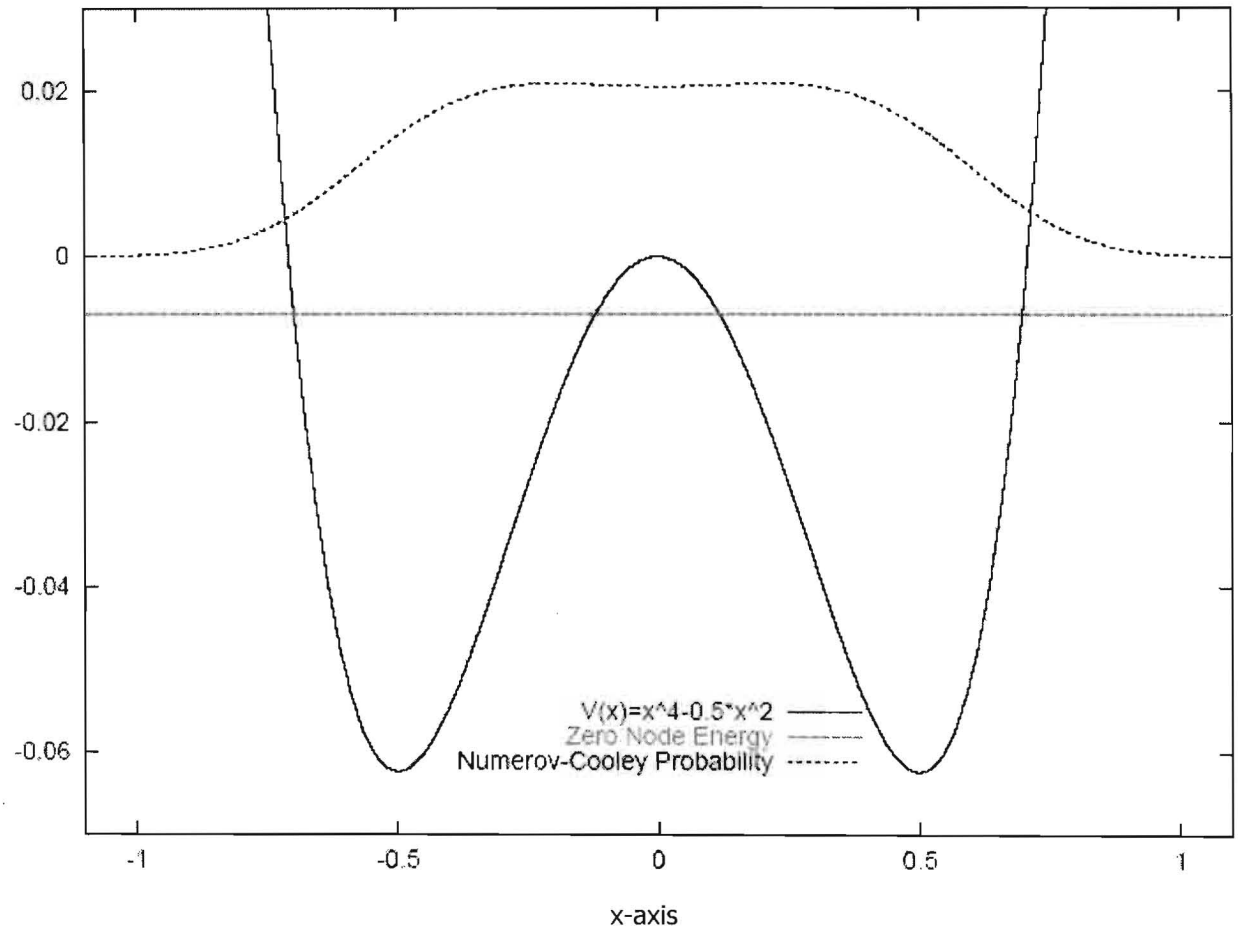


Figure 6

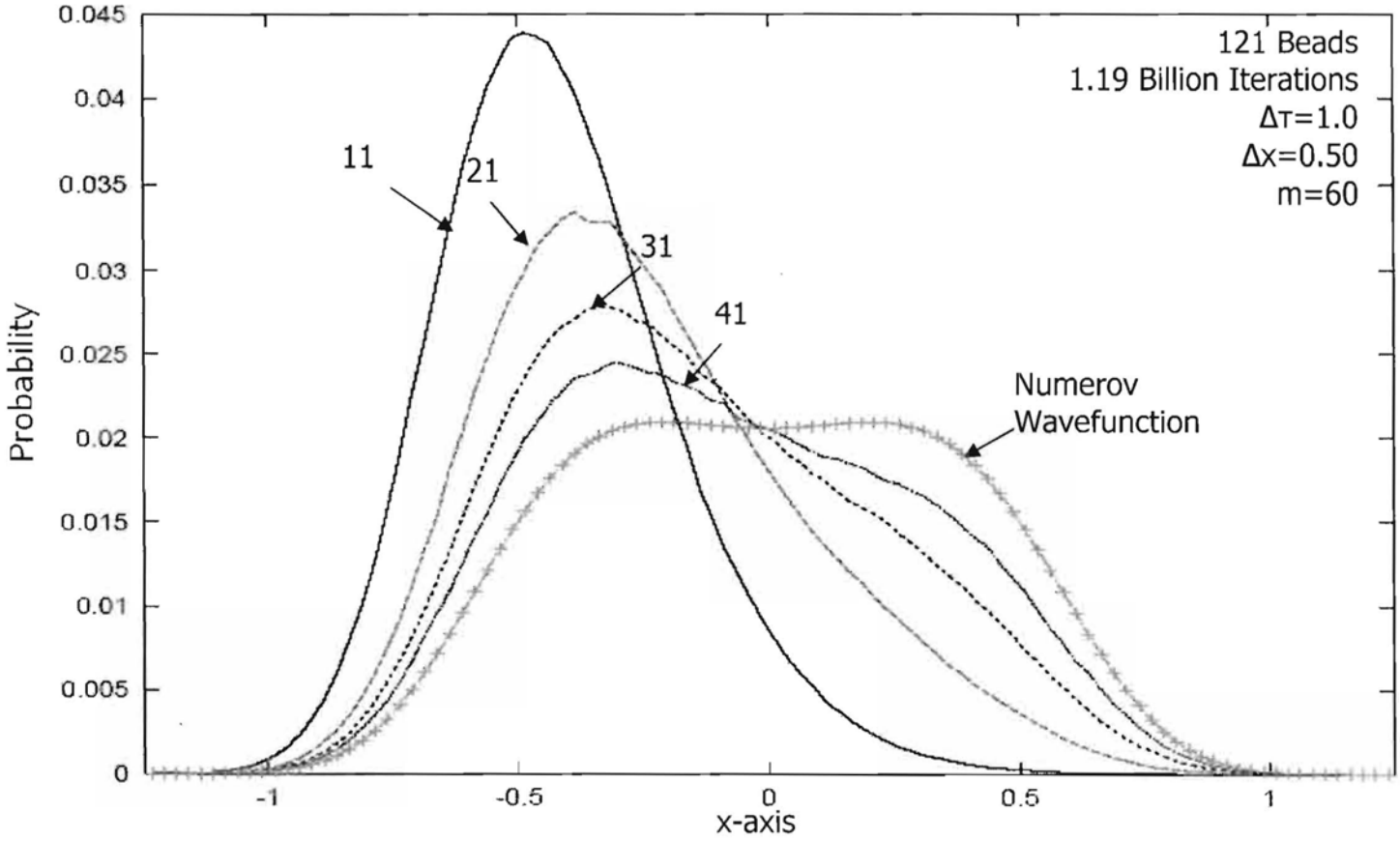

121 Beads
1.19 Billion Iterations
Δτ=1.0
Δx=0.50
m=60
11
21
31
41
Numerov
Wavefunction
Probability
x-axis
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
-1
-0.5
0
0.5
1

Figure 7

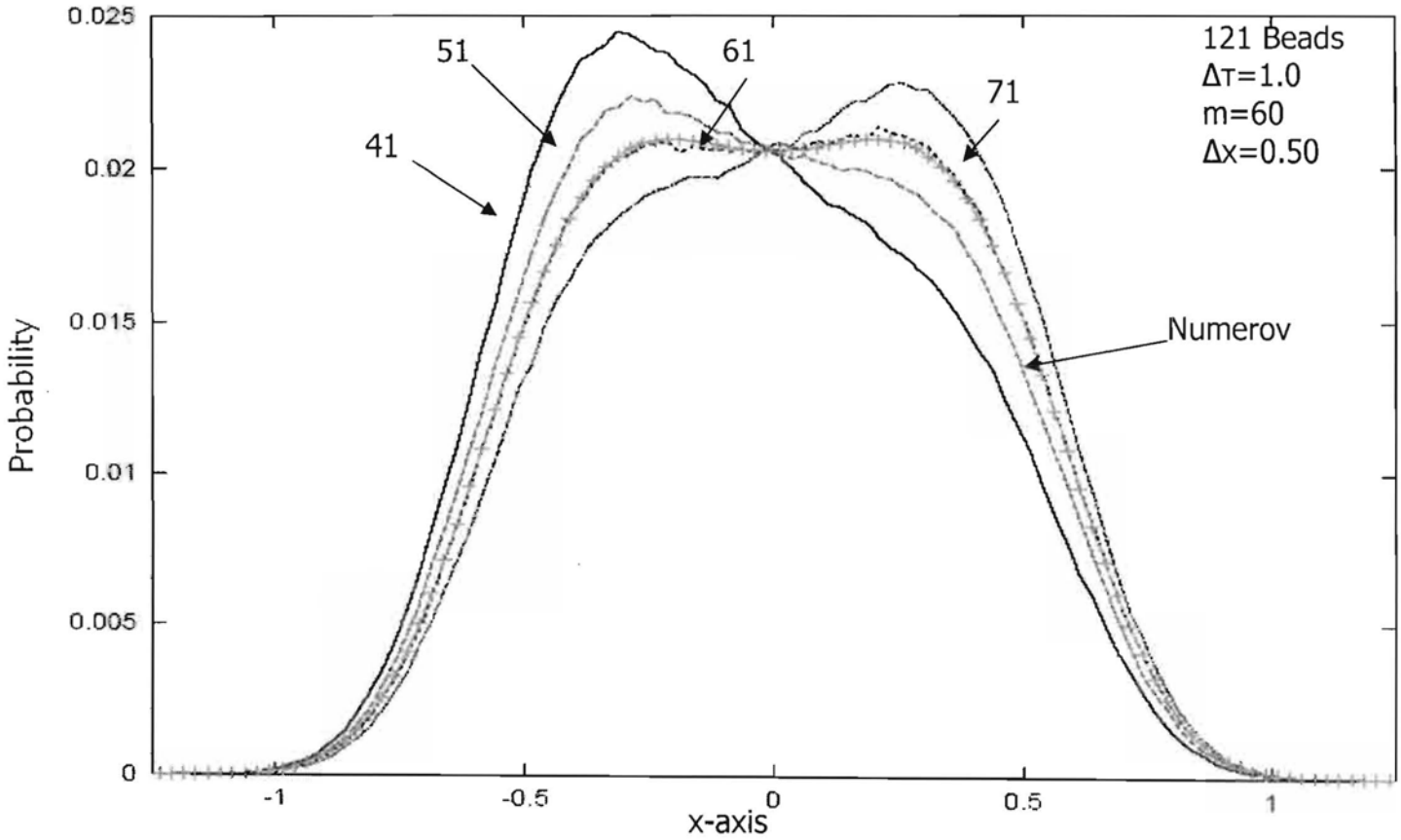

0.025
0.02
0.015
0.01
0.005
0
Probability
-1
-0.5
0
0.5
1
x-axis
51
61
71
41
121 Beads
Δτ=1.0
m=60
Δx=0.50
Numerov

Figure 8

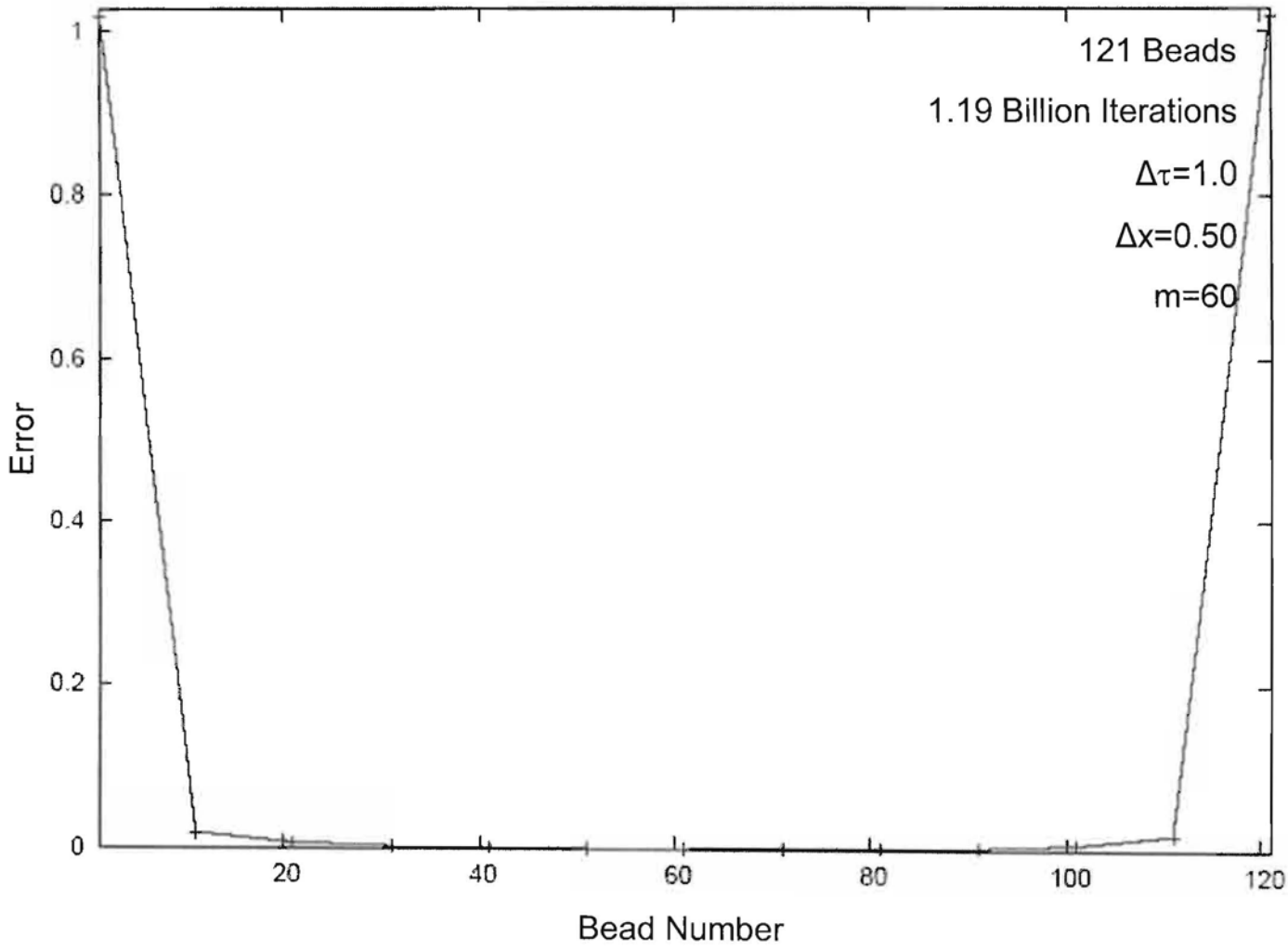

121 Beads
1.19 Billion Iterations
Δτ=1.0
Δx=0.50
m=60
Error
Bead Number
0
0.2
0.4
0.6
0.8
1
20
40
60
80
100
120

Figure 9

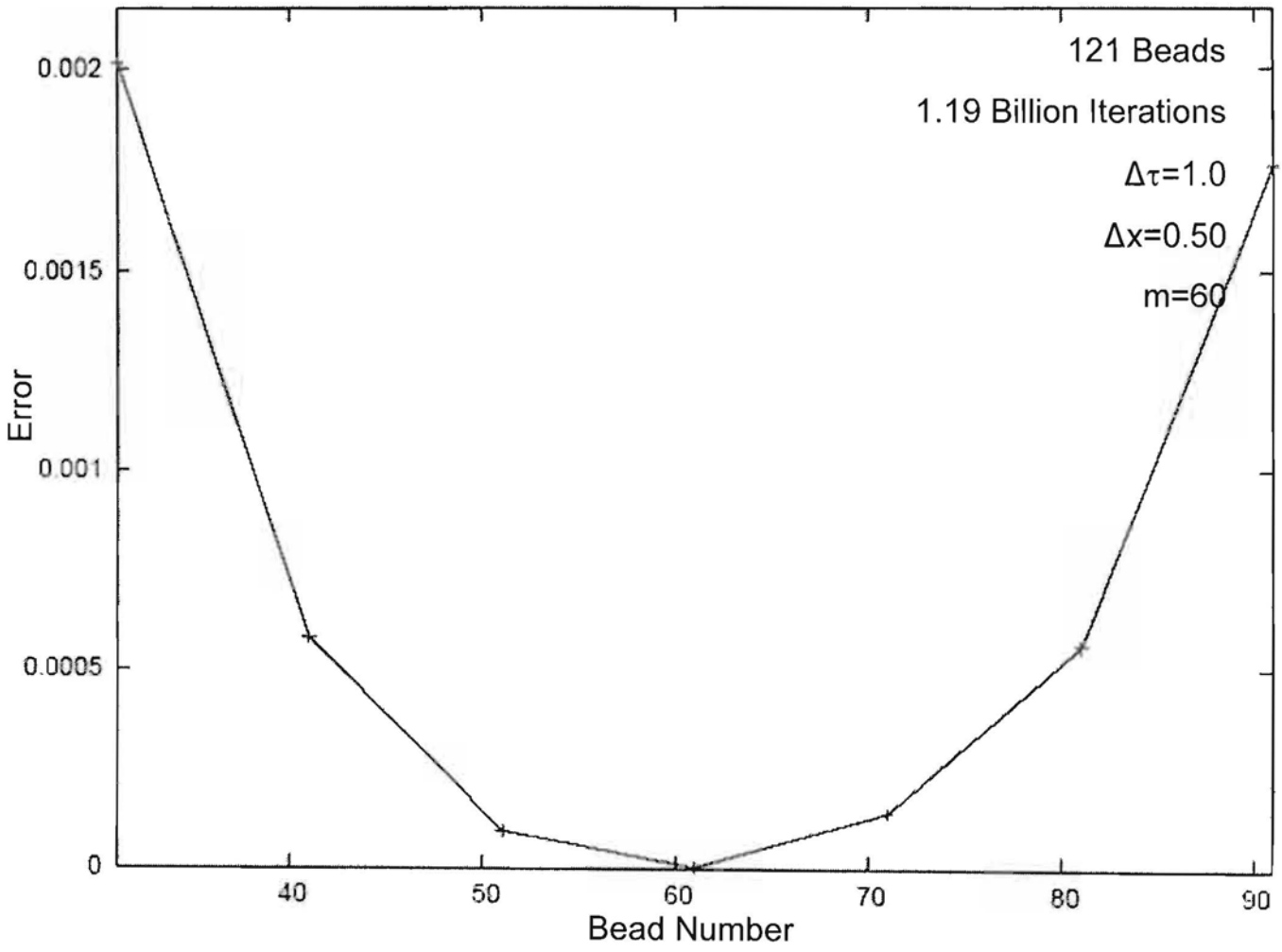

121 Beads
1.19 Billion Iterations
Δτ=1.0
Δx=0.50
m=60
Error
0.002
0.0015
0.001
0.0005
0
40
50
60
70
80
90
Bead Number

Figure 10

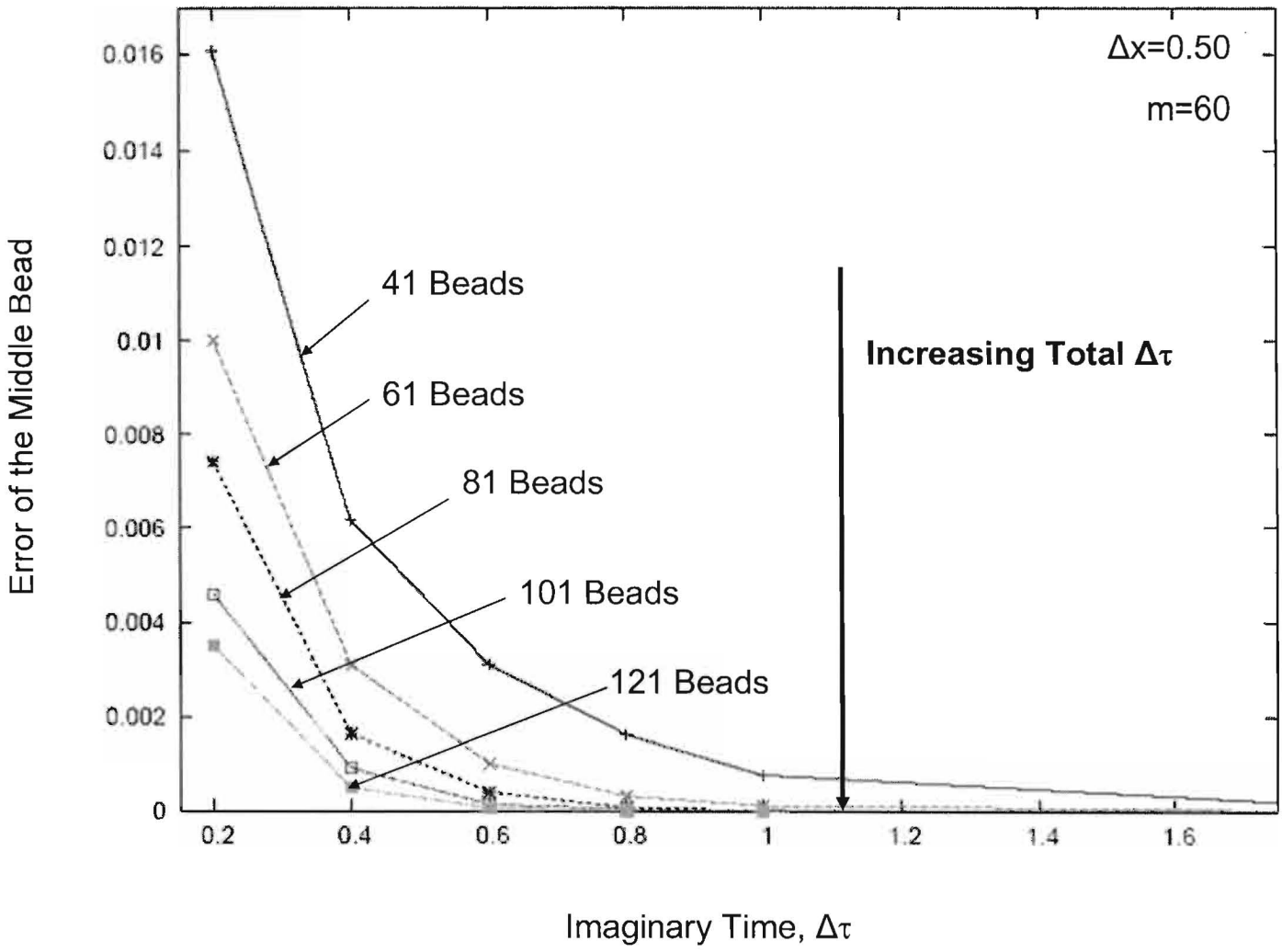

Δx=0.50
m=60
41 Beads
61 Beads
81 Beads
101 Beads
121 Beads
Increasing Total Δτ
Error of the Middle Bead
Imaginary Time, Δτ
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

Figure 11

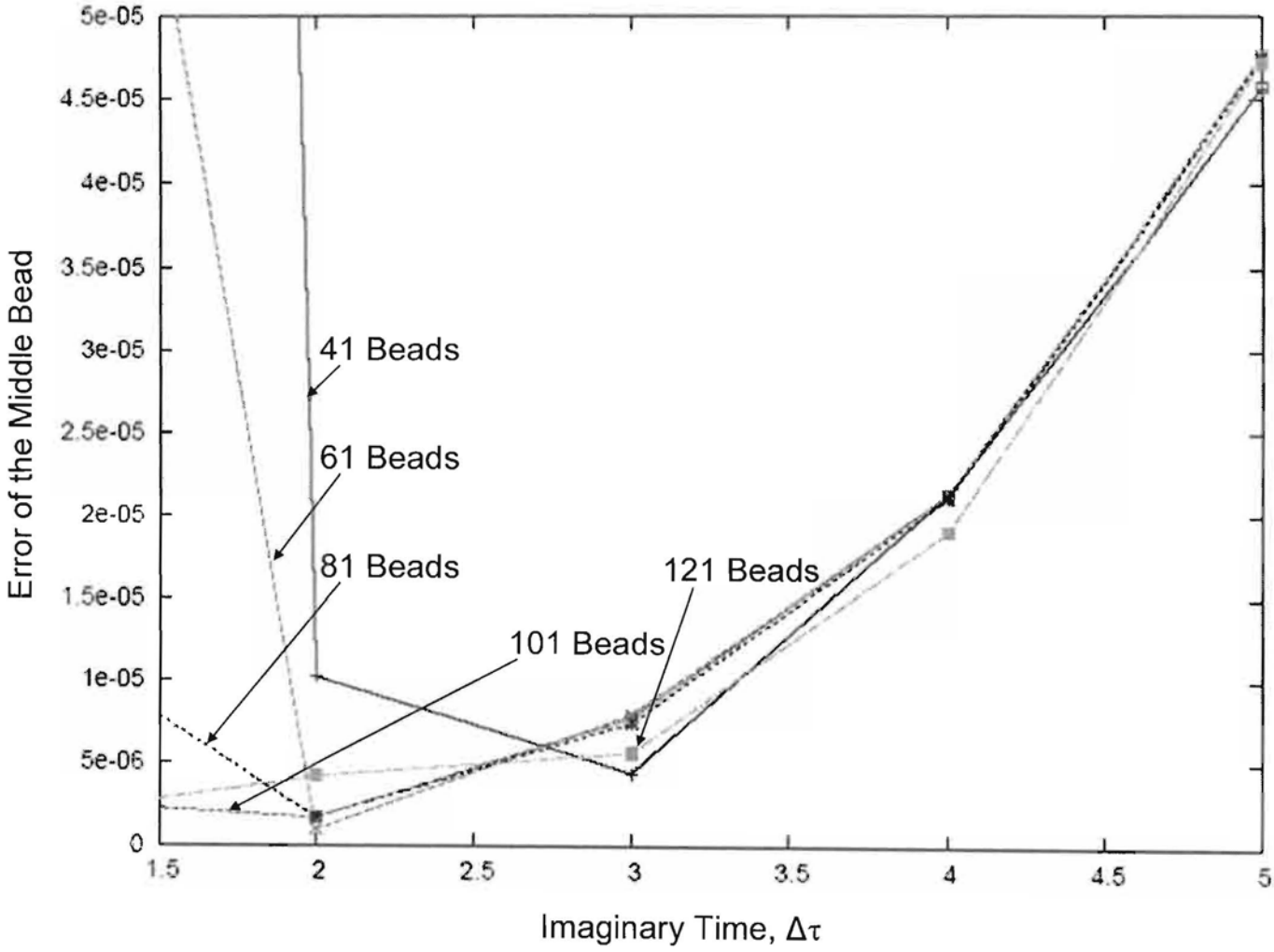

Error of the Middle Bead
Imaginary Time, Δτ
41 Beads
61 Beads
81 Beads
101 Beads
121 Beads
5e-05
4.5e-05
4e-05
3.5e-05
3e-05
2.5e-05
2e-05
1.5e-05
1e-05
5e-06
0
1.5
2
2.5
3
3.5
4
4.5
5

Figure 12

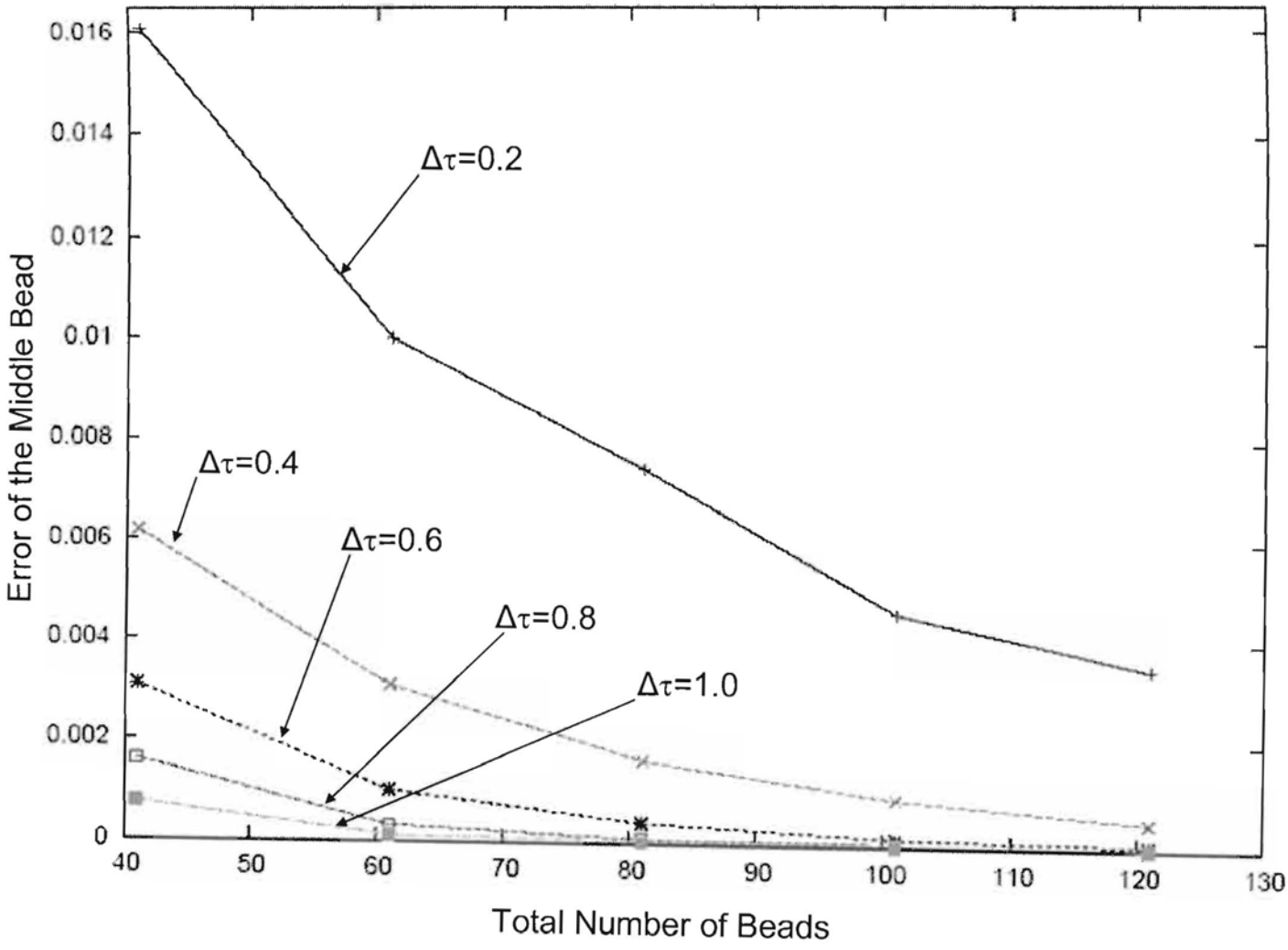

0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
Error of the Middle Bead
Δτ=0.2
Δτ=0.4
Δτ=0.6
Δτ=0.8
Δτ=1.0
40
50
60
70
80
90
100
110
120
130
Total Number of Beads

Figure 13

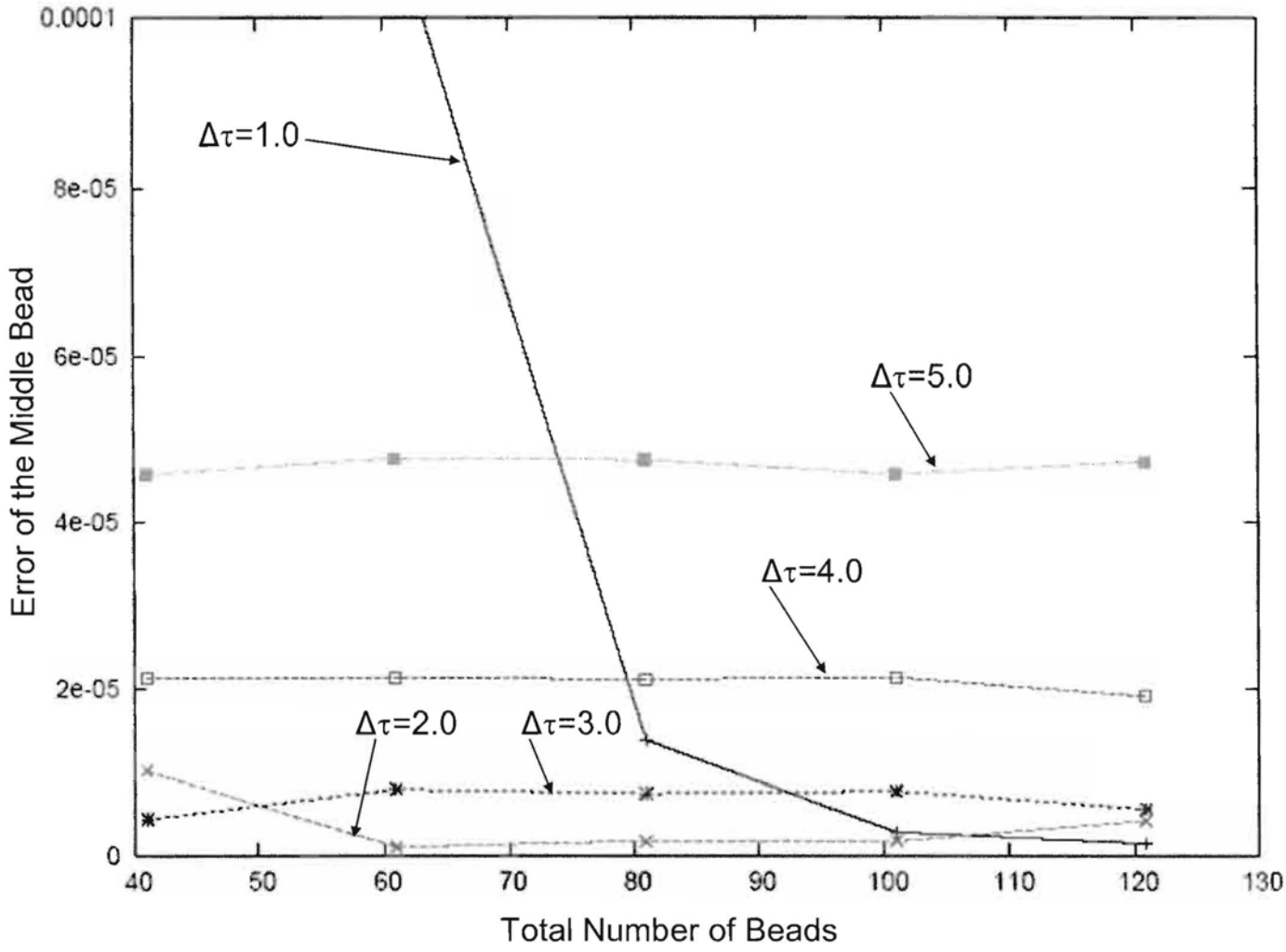

0.0001
8e-05
6e-05
4e-05
2e-05
0
Error of the Middle Bead
Δτ=1.0
Δτ=5.0
Δτ=4.0
Δτ=2.0
Δτ=3.0
40
50
60
70
80
90
100
110
120
130
Total Number of Beads

Figure 14

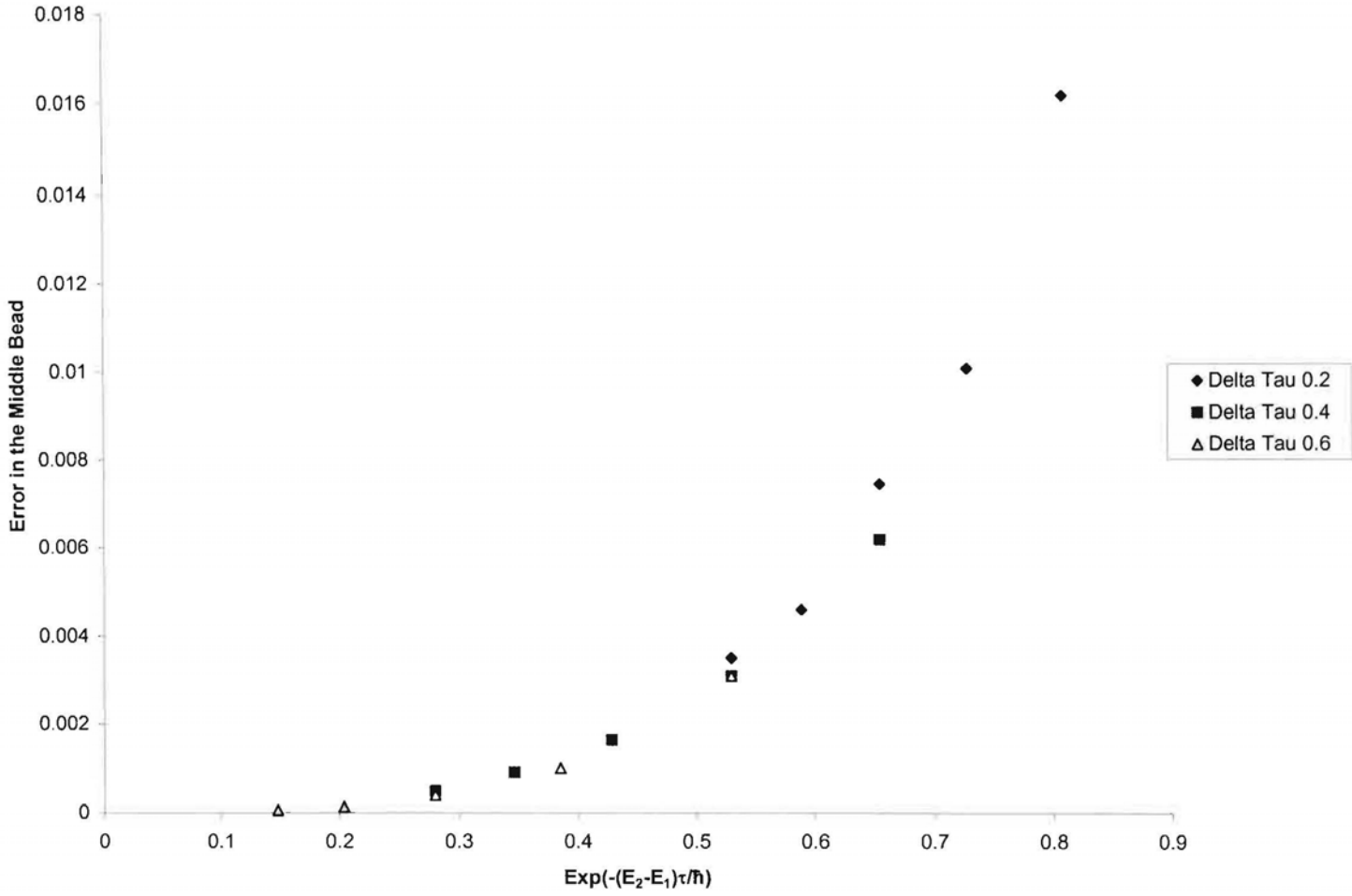

0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
Error in the Middle Bead
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Exp(-(E2-E1)τ/ħ)
Delta Tau 0.2
Delta Tau 0.4
Delta Tau 0.6

Figure 15

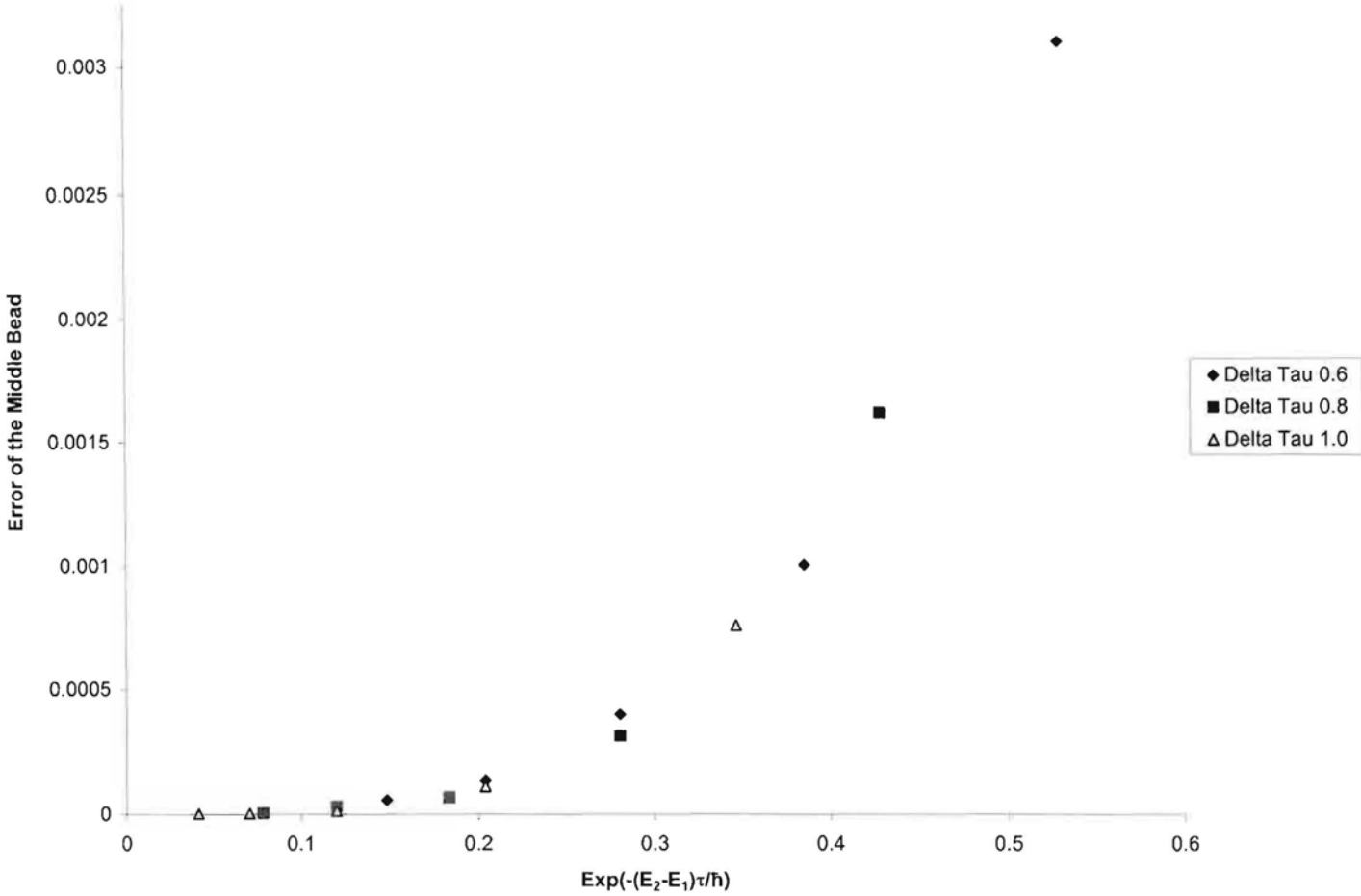

0.003
0.0025
0.002
0.0015
0.001
0.0005
0
Error of the Middle Bead
0
0.1
0.2
0.3
0.4
0.5
0.6
Exp(-(E2-E1)τ/ħ)
Delta Tau 0.6
Delta Tau 0.8
Delta Tau 1.0

Figure 16

Acknowledgements

I would like to thank Dr. Robert J. Hinde for his guidance and patience as a research mentor. He provided both insightful ideas and constructive reviews of this paper. Thanks to Patrick Moehlen for computer assistance and to Brent Magnusson for his support.

References

Atkins, Peter and Paula, Julio de. Physical Chemistry. 7th Ed. New York: W.H. Freeman and Company, 2002.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. *Equation of State Calculations by Fast Computing Machines*. J. of Chem. Physics, vol. 21: 1087-1092, 1953.

Nave, R. *Tunneling, Barrier Penetration.* http://hyperphysics.phy-astr.gsu.edu/HBASE/quantum/barr.html. 20 Feb 2006.

Sarsa, A, K.E. Schmidt, and W.R. Magro. *A path integral ground state method.* J. of Chem. Physics, vol. 113: 1366-1371, 2000.

0.5potential.f

```
C     This file contains the equation for the double-well potential energy
C     surface with c=0.5.
      FUNCTION POTL (X)
      POTL = X**4-0.5*X**2
      RETURN
      END
```

60wave.f

```
C     This file gives the trial wavefunctions to be used for the first and
C     last bead of the chain.  This trial wavefunction was selected for a
bead
C     mass of 60.
      FUNCTION WAVE (X)
      WAVE = EXP(-5.5*(X+0.375)**2)+EXP(-5.5*(X-0.375)**2)
      RETURN
      END
```

ran2.f

```
C     This file generates a random number between 0 and 1.
      function ran2()
      parameter (im1=2147483563)
      parameter (im2=2147483399)
      parameter (am=1.0/2147483563.0)
      parameter (imm1=im1-1)
      parameter (ia1=40014)
      parameter (ia2=40692)
      parameter (iq1=53668)
      parameter (iq2=52774)
      parameter (ir1=12211)
      parameter (ir2=3791)
      parameter (ntab=32)
      parameter (ndiv=1+imm1/ntab)
      parameter (eps=1.0e-8)
      parameter (rnmx=1.0e0-eps)

      common /rancom/ iv(ntab), iy, idum2, irseed

      if (irseed.le.0) then

      irseed=max(-irseed, 1)
      idum2=irseed

      do 100 j=ntab+8, 1, -1
      k=irseed/iq1
      irseed=ia1*(irseed-k*iq1)-k*ir1
      if (irseed.lt.0) irseed=irseed+im1
      if (j.le.ntab) iv(j)=irseed
100   continue

      iy=iv(1)

      end if

      k=irseed/iq1
      irseed=ia1*(irseed-k*iq1)-k*ir1
```

```
      if (irseed.lt.0) irseed=irseed+im1

      k=idum2/iq2
      idum2=ia2*(idum2-k*iq2)-k*ir2
      if (idum2.lt.0) idum2=idum2+im2

      j=1+iy/ndiv
      iy=iv(j)-idum2
      iv(j)=irseed
      if (iy.lt.1) iy=iy+imm1

      ran2=amin1(am*float(iy), rnmx)
      return
      end
```

WEIGHT.f

```
C     This algorithm calculates the weight function that is used in PIGS QMC
C     calculations given by equations 2 and 3.
      SUBROUTINE WEIGHT (ARRAY, NBEAD, BMASS, TIME, W)
      DIMENSION ARRAY (NBEAD)
      TOTAL = 0
      DO 10 I=1,NBEAD-1
      VX = POTL (ARRAY(I))
      VX1 = POTL (ARRAY(I+1))
      F = (VX+VX1)/2+BMASS*(ARRAY(I+1)-ARRAY(I))**2/(2*TIME**2)
      TOTAL = TOTAL + F
10    CONTINUE
C      print *, 'total =', total
C      print *, 'time*total =', time*total
C      print *, 'exp(-time*total) =', exp(-time*total)
C      print *, 'alog(exp(-time*total)) =', alog(exp(-time*total))
      WX = WAVE (ARRAY(1))
      WX1 = WAVE (ARRAY (NBEAD))
      W = ALOG (WX) + alog(WX1) - (TIME*TOTAL)
C      print *, 'wx, wx1, w = ', wx, wx1, w
      RETURN
      END
```

Error.f

```
     Program Error
C This program takes a column of data from the bead Monte Carlo output and
the Numerov output and computes the error by summing the square of the
differences in each entry.
      DIMENSION xMCdata(200)
      DIMENSION yNum(200)
      DIMENSION Err(200)
      OPEN (1, FILE='../../../NumOutput/Num-0.5-60-0-middle-200-bins')
      TotErr=0
      DO 1, J=1,200
      Read (5, *) xMCdata(J)
      READ(1,*) t1, t2, yNum(J)
      Err(J)=abs(xMCdata(J)-yNum(J))
      TotErr=TotErr+Err(J)**2
1     Continue
      Print *, TotErr
      stop
```

```
      end
```

bead-60-2Stat.f

```
C     This file runs the QMC portion of the simulation.
      PROGRAM TEMPORARY
       DIMENSION ARRAY(1000)
       DIMENSION BIN(14,200)
C       CHARACTER * 40 FNAME, FNAME2, FNAME3, FNAME4
       COMMON /rancom/ iv(32), iy, idum2, irseed
       irseed = 0
       PRINT *, 'ENTER THE NUMBER OF REPETITIONS TO BE PERFORMED'
       READ *, IREP
       PRINT *, 'ENTER DELTA'
       READ *, DELTA
       PRINT *, 'ENTER MASS OF THE BEADS'
       READ *, BMASS
       PRINT *, 'ENTER THE DELTA T'
       READ *, TIME
       PRINT *, 'ENTER SNAPSHOT INTERVAL'
       READ *, ISNAP
       OPEN (1, FILE = 'STARTING')
       READ (1,*) NBEAD
C       WRITE (FNAME, 99) irep/1000000, delta, int(bmass), time, isnap
C99     format('51t-','r',i4.4,'.d',f4.2,'.m-',i2.2,'.t',f3.1,'.int',i3.3)
C       OPEN (7, FILE = fname)
C       WRITE (FNAME2, 98) irep/1000000, delta, int(bmass), time, isnap
C98     format ('1t-','r',i4.4,'.d',f4.2,'.m-',i2.2,'.t',f3.1,'.int',i3.3)
C       OPEN (8,FILE = fname2)
C       WRITE (FNAME3, 97) irep/1000000, delta, int(bmass), time, isnap
C97     format('25t-','r',i4.4,'.d',f4.2,'.m-',i2.2,'.t',f3.1,'.int',i3.3)
C       OPEN (9, FILE = fname3)
C       WRITE (FNAME4, 96) irep/1000000, delta, int(bmass), time, isnap
C96     format('75t-','r',i4.4,'.d',f4.2,'.m-',i2.2,'.t',f3.1,'.int',i3.3)
C       OPEN (10, FILE = fname4)

C      PRINT *, NBEAD

       DO 1, J=1, NBEAD
       READ (1,*) ARRAY(J)
1      CONTINUE
       CLOSE (1)

C       print *, nbead
       do 3, i=1,14
       DO 2, J=1,200
       BIN(i,J) = 0
2      CONTINUE
3      continue
C       print *, nbead

       CALL WEIGHT (ARRAY, NBEAD, BMASS, TIME, WOLD)

C       print *, nbead

       DO 10, I=1,IREP
```

```
C      print *, I, NBEAD

38     XX=RAN2()
       J=XX*(NBEAD-2) + 2
       if (J.eq.NBEAD) go to 38
c      J = RAN2()*(NBEAD-2) + 2
      DIST = (RAN2()*2.0*DELTA) - DELTA

C      print *, I, NBEAD

      ARRAY (J) = ARRAY(J) + DIST

C      print *, I, NBEAD

      CALL WEIGHT (ARRAY, NBEAD, BMASS, TIME, WNEW)

C      print *, I, NBEAD

      IF (WNEW.GT.WOLD.OR.WNEW.EQ.WOLD) THEN
        WOLD = WNEW
      ELSE
      R = exp(WNEW-WOLD)
        T = RAN2()
      IF (T.LT.R) THEN
        WOLD=WNEW
        ELSE
           ARRAY (J) = ARRAY(J) - DIST
        END IF
      END IF

C      print *, I, NBEAD

      IF (MOD (I, ISNAP).EQ.0) THEN
C     The following lines select a bead to monitor.  Then the number of
iteration and the bead position is written into a t-file.
C          T = ARRAY (((NBEAD-1)*3)/4.0)
C          X = ARRAY ((NBEAD+1)/2.0)
C          Y = ARRAY (1)
C          Z = ARRAY ((NBEAD-1)/4.0)
C       WRITE (7,*) I, X
C       WRITE (8,*) I, Y
C       WRITE (9,*) I, Z
C       WRITE (10,*) I, T

        DO 88, H=2,14 ! This is the start of my loop.
          N=(H-2)*10+1 ! This will give me 1,11,21,...,101.
          X=ARRAY(N) ! This will give me the position of bead N.
          J = 1 + (X+2.5)/0.025 ! This changes the position to a bin #.
          BIN(H,J) = BIN(H,J) + 1 ! This increases the element in (h,J) by 1
88      Continue
        end if
10      continue
        BINSUM=0
        DO 15, J=1,200
        BINSUM =BIN(2,J)+BINSUM ! Binsum should be the same for all columns

15     CONTINUE
```

```
       DO 77, I=2,14 ! One column at a time
       DO 25, J=1,200 ! Go down the row
          BIN(I,J)=BIN(I,J)/BINSUM ! and divide the element by binsum
25     Continue
77     Continue
        DO 20 J=1, 200 !go down the rows

        WRITE (6, 6000) J,(BIN(I, J), I=2, 14) !write out 12 columns for each
row

6000    FORMAT (I3, 1x, 13(F8.6, 1X))

20     CONTINUE

      stop
      end
```

Numerov.f

```
C     This is the program to compute the Numerov-Cooley probability
distribution.
      implicit real*8 (a-h, o-z)

      parameter (nmax=50000)

      character*40 fname

      common /mascom/ twomas

      dimension potl(nmax), r(nmax), psiout(nmax), prob(nmax)

c --- input potentials

      write (6, *) 'enter x^2 coefficient and mass:'

      read (5, *) c, zm

      twomas=2.0d0*zm

      xmin=-3.5
      xmax=3.5

      npts=7001

      vmin=xmin**4-c*xmin**2

      do i=1, npts

         x=xmin+(xmax-xmin)/(npts-1)*(i-1)

         r(i)=x
         potl(i)=x**4-c*x**2

         if (potl(i).lt.vmin) then
            rmin=x
            vmin=potl(i)
         end if
```

```
      end do

      write (6, 6000) vmin, abs(rmin)
6000  format ('minimum energy = ', f10.5, ' at x = +/- ', f10.5)

      v1=(rmin-0.01)**4-c*(rmin-0.01)**2
      v2=(rmin+0.01)**4-c*(rmin+0.01)**2

      force=(v2+v1-2.0*vmin)/(0.01**2)
      freq=sqrt(force/zm)

      eguess=0.5*freq+vmin

      write (6, 6010) eguess
```