UNIVERSITY OF ILLINOIS

May 5 19 87

THIS IS TO CERTIFY THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Julio James Arrocio

ENTITLED Path Integral Monte Carlo Simulations of

Many Fermion Systems

IS APPROVED BY ME AS FULFILLING THIS PART OF THE REQUIREMENTS FOR THE

DEGREE OF Bachelor of Science In Chemistry

Instructor in Charge

APPROVED:

HEAD OF DEPARTMENT OF Chemistry

O-1364

Path Integral Monte Carlo Simulation

of

Many Fermion Systems

By

Julio James Arrecis

Thesis

for the

Degree of Bachelor of Science

in

Chemistry

College of Liberal Arts and Sciences

University of Illinois

Urbana, Illinois

1987

TABLE OF CONTENTS

INTRODUCTION

In recent years, Path Integral formulations of Statistical Mechanics have become an increasingly powerful tool for dealing with both chemical and physical phemonenema. As a discretized polymer chain, the imaginary-time Path Integral is a popular tool that can be used to describe both single particles and many body phenonema and also can be readily adapted for Monte Carlo computer simulations. However, a big problem arises when one begins to include Fermi particle interactions for systems as small as two particles.[1] Clearly, if one is to use such powerful techniques for dealing with complex many body systems, then one must be able to effectively include these Fermion interactions into an algorithm. Unfortunately, most attempts to deal with Fermi exclusions have been unsucessful.

Many existing algorithms are able to evaluate the imaginary-time Schrodinger kernel for distinguishable particles. In general, the density matrix can be written as an imaginary-time Path Integral between the two states, x_1 and x_2[2]

$$\rho(x_1,x_2)=\int dx_1 \int dx_2 \ldots \int dx_n \exp(-\beta H)=\int \mathcal{D}x \exp(-\beta \mathcal{H})$$

and the trace of the density matrix yields the partition function. For distinguishable particles, the resulting

partition function is just the sum of postitve definite terms:

$$Z_{initial} = \sum_{i=0}^{n} \exp(-\beta E_i) = e^{-\beta E_0} + e^{-\beta E_1} + e^{-\beta E_2} + \ldots$$

where the energies are ordered, $E_0 < E_1 < E_2 < \ldots$ and each E_i is the eigenvalue of the instanteous Hamiltonian of the system. However, now if we incorporate the permutation operator (P) into our kernel, we see that the partition function for an exchanged particle is no longer the sum of positive terms:

$$Z_{exchange} = e^{-\beta E_0} - e^{-\beta E_1} + e^{-\beta E_2} - \ldots\ldots$$

where the minus signs are the results of the antisymmetric wavefunctions. The problem now arises when we try to write the total partition function as a sum of the initial and exchanged parts:

$$Z_{total} = Z_{initial} - Z_{exchanged}$$

Unfortunately, the ground state seems to drop out and for systems at low temperatures and this is unacceptable.

In this paper, we propose a new way to look at the Fermion problem through imaginary-time Path Integrals. Although much of this work parallels that of V.Elser[3], we wish to use a much simpler model that will illustrate our

ideas a bit more clearly. We model one-dimensional Fermions in an infinite well as a discritized Path Integral that has only Fermi interaction potential. By using the Pauli exclusion principal, we find the nodes of the total wavefunction and use these nodes to define a new subdomain of phase space. In general, this technique removes the Fermi potential from the Path Integral (partition function) and places it as a hard wall boundry condition. Then by applying Path Integral Monte Carlo techniques to the new subdomain of phase space and show that the simulated equilibrium polymer motions in this subdomain give the expected ground state values for the Fermion system. We can compare some of our computer results toanalytic calculations for one and two particle systems and find that there is fair agreement.

METHODS

For our simple system, we consider the free particle Hamiltonian of the form:

$$\mathcal{H}=1/2*(m_1v_1^2+m_2v_2^2+m_3v_3^2+\ldots+m_nv_n^2)$$

where v_i is the imaginary-time velocity. For two particle this is just:

$$\mathcal{H}=1/2*m_1v_1^2+1/2*m_2v_2^2$$

Effectively, there is potential term which is due to the

Fermi interactions, however we choose to make this into our boundry conditions, so it disappears from this expression. Since we have eliminated the cross terms in the Hamiltonian, we can write the imaginary-time Path Integral as a product of two single particle paths:

$$\int \mathcal{D}x_1 \int \mathcal{D}x_2 \exp(-\beta\mathcal{H}_1)\exp(-\beta\mathcal{H}_2) = \int \mathcal{D}x_1 \exp(-\beta\mathcal{H}) \int \mathcal{D}x_2 \exp(-\beta\mathcal{H}) \quad (1)$$

where $\beta=1/kt$ and the Hamiltonian is, in imaginary-time, discretized as:

$$\beta\mathcal{H}_j = m_j * 1/(2*\Delta\beta)*(x_{i+1}-x_i)^2,$$

p= the number of beads in the polymer chain and $\Delta\beta=\beta/p$. By making small movements about an equilibrium conformatiom, we claim that this polymer chain will duplicate the system of interest. Notice that the right hand side of equation (1) is a polymer of dimension two and for every new particle we add, its dimension increases by a particle dimensionality, D. In this form, the Path Integral duplicates the motions of the entire system as one jiggling polymer in phase space. Now, we wish to incorporate the nodes of the wavefunction into our phase space.

To determine the nodes for phase space, we call upon an elementary concept from quantum mechanics, the Pauli exclusion principle. We write the total ground state

wavefunction for this two particle system as a Slater determinate:

$$\Psi = \det \begin{bmatrix} \psi_1(1)\psi_1(2) \\ \psi_2(1)\psi_2(2) \end{bmatrix}$$

which, by using the normalized ground state wavefunctions for a free particle, is given explicitly by:

$$\Psi_{ground} = (2/\sqrt{6}*L)*[\sin(\pi x_1/L)\sin(2\pi x_2/L) - \sin(2\pi x_1/L)\sin(\pi x_2/L)].$$

The nodes for this system are found when $x_1 = x_2$. Geometrically, this means that phase space has a 45° nodal line in it, and this line behaves exactly like an impenetrable barrier, separating the subdomains from each other.

For this system, the nodes are trivial to find In general, however,this can be a difficult task. One can try to search for exchange nodes by inspection but a more clever way is write the time derivative of the probability density as a surface integral:

$$\int_V \nabla\cdot [\Psi^*\nabla\Psi - (\nabla\Psi)^*\Psi]\, d\tau = \int_S [\Psi^*(\frac{\partial}{\partial r}\Psi) - (\frac{\partial}{\partial r}\Psi)^*\Psi]\, dS$$

In this form, the gradient of the wavefunction determines the slope of the probability density as a function of the

surface S. As the surface S extends out over the regions to be considered, the value of the integrand will diminish as S approachs a node. Thus here is a way to find the nodes of such a system if the total wavefunction is known. However, knowing the total wavefunction of a system is a stiff requirement and generally, this is not known. Recently, other researchers have suggested that if a trial wavefunction that is close to the system's is known, that one could find the real nodes of the system by applying the above technique and relaxing the nodes of the trial system[4].

Regardless of the technique, once the nodes are found, one can proceed to set the subdomain of phase space and sample the new region with Path Integral methods. Specifically, a chain of beads is laid down at random with the constraints that any two beads cannot be in the same place at the same time and that they cannot step outside the physical dimensions of the box. Now we apply the Metropolis Monte Carlo sampling techinque[5], calculating the energy of the i^{th} particle as:

$$E_i = m/2\Delta\beta \left((x_{i+1} - x_i)^2 + (x_i - x_{i-1})^2 \right).$$

The i^{th} particle is now moved a random amount and the new energy is calculated. If the new energy is less then the old energy, the move is accepted. If not, the logarithm of a

random number is compared to the energy difference, $-(E_{new}-E_{old})$ via the Metropolis Algorithm. As a result, the higher level configuations (paths) are accepted with a Gaussian probability distribution. Whatever the outcome of the move, an average is taken over the new configuation, even if it has remained the same, to find the relevant statistical quatities, namely the energy and distributional information. The energy of a Path Integral is given by the derivative, with respect to beta, of the partition function:

$$\langle E\rangle=\langle -(\frac{\partial}{\partial\beta}\ln\,[\rho(x,x)])\rangle=\langle\, 1/2\Delta\beta - N/(2\beta*\Delta\beta)*\Sigma\,(x_{i+1}-x_i)^2\,\rangle.$$

The first term comes from a normalization constant that is hidden in the Path Integral formalism (i.e. in the $\mathcal{D}x$ terms)[6].

SIMULATION RESULTS

A series of computer simulations were run on both one and two particle systems. The first simulation was of a single particle system and was done to check to see if the algorithm was working. Clearly the values for both the probability density and the average energy should not depend on the fact that the particle is a Fermion. Moreover, the analytic expressions for both the density and the average energy are known, so the simulation results must agree with these quantities.

Figure 1 shows the computer simulation of a single

particle using the Path Integral Monte Carlo method as outlined above. The analytic result expected is found from the probability distribution:

$$\mathcal{P}(x) = \Psi^{*}\Psi = 2/L * \sin^{2}(\pi x/L)$$

When compared with the simulation results, we get excellent agreement (See Fig.1). The average energy of the system can be calculated easily for the expectation value of the Hamiltonian:

$$\langle E \rangle = \int \Psi^{*} \mathcal{H} \Psi \, d\tau = \langle \Psi | \mathcal{H} | \Psi \rangle = \frac{n^{2}\pi^{2}}{2*L^{2}} \quad \text{(energy in Hartees)}$$

For a system largely in the ground state,the value of <E> should correspond to that of n=1. With the numerical value for a box of length = 15.0 Bohr radii, the energy should be 0.0219 hatrees. The simulation results for the same system give <E>=0.0189 hatrees which is in fair agreement with the theoretical result.

Figure 3 shows the simulation results for a two particle system. Ploted here is a quantity known as the g(r). This quantity is a distribution function that measures the number of nearest neighbors at distance r away. For our system, g(r) measures the number of Fermion at a distance of r away. This function, g(r), is useful in the theory of liquids becuase it can related to many physical properties,

A numerical integration of this function was used and is also plotted in figure 3 as a solid line. Although there is quite a bit of flucuation in the simulation curves, we see that there seems to be fair agreement between the analytic and simulation results. In addition to the distributions, the average energy for the two particle system was collected. For a two Fermion system, the average energy is given by the expectation value of the Hamiltionian in the total wavefunction basis:

$$\langle E \rangle = \langle \Psi_{total} | \mathcal{H} | \Psi_{total} \rangle$$

Again, after much tedious integration, we find for a system that is largely in the ground state, that the average energy should converge to $\langle E \rangle = 32/(27*L) = 0.079$ Hartrees. The computer results give good agreement, with an average energies for several run listed in Table 1. Although much of the variance in these values can be attributed to the simulation technique itself, there still was some difficulty duplicating those numbers with any regularity. We feel that this is p obably a computer simulation problem and that it can be accounted for by considering the nature of our simulation algorithms.(See Figure 3 and Table 1)

CONCLUSIONS:

The algorithm outlined above seems to give good numerical results for the ground state energies and is

modeling the Fermion system quite well. Overall, the results are very encouraging and may provide a new technique for dealing with many Fermion statistics, however there are more numerical tests to be done. The present computer coding is not that reliable. One of the possible reasons for the this is that the Monte Carlo technique employed was of single particle moves, which is a bad statistical idea. Although the simplest in concept, single particle move introduces large fluctuations in statistical parameters because the polymer doesn't change much from move to move. In contrast, Path Integral programs today use the method of normal modes, which increases the amount of phase space sampled and also cuts down on fluctuations in statistics. Moreover, the single particle technique can give irreproducable results as we have cited however, what we did find was encouraging. In conclusion, I would to thank the entire Wolynes group for their help and special thanks to Raymond Cline, Atsuo Kuki and of course, Peter G. Wolynes.

REFERENCES

[1] R.P. Feynman and A.R. Hibbs, in *Quantum Mechanics and Path Integrals* (McGraw-Hill, New York, 1965), p.292.

[2] Ibid, p.276

[3] V. Elser, Phys. Rev. A 34, 3 (1986).

[4] D.M. Ceperley and B.J. Alder, Phys. Rev. Lett. 45, 566 (1980)

[5] N. Metropolis, A.W. Metropolis, M.N. Rosenbluth, A.H. Teller, and E. Teller, J. Chem. Phys. 21, 1087 (1953)

[6] R Hall, Path Integral Approaches To Quantum Statistical Mechanics, Ph.D thesis, Columbia University (1984)

FIGURE 1 single particle distribution (square=simulation)

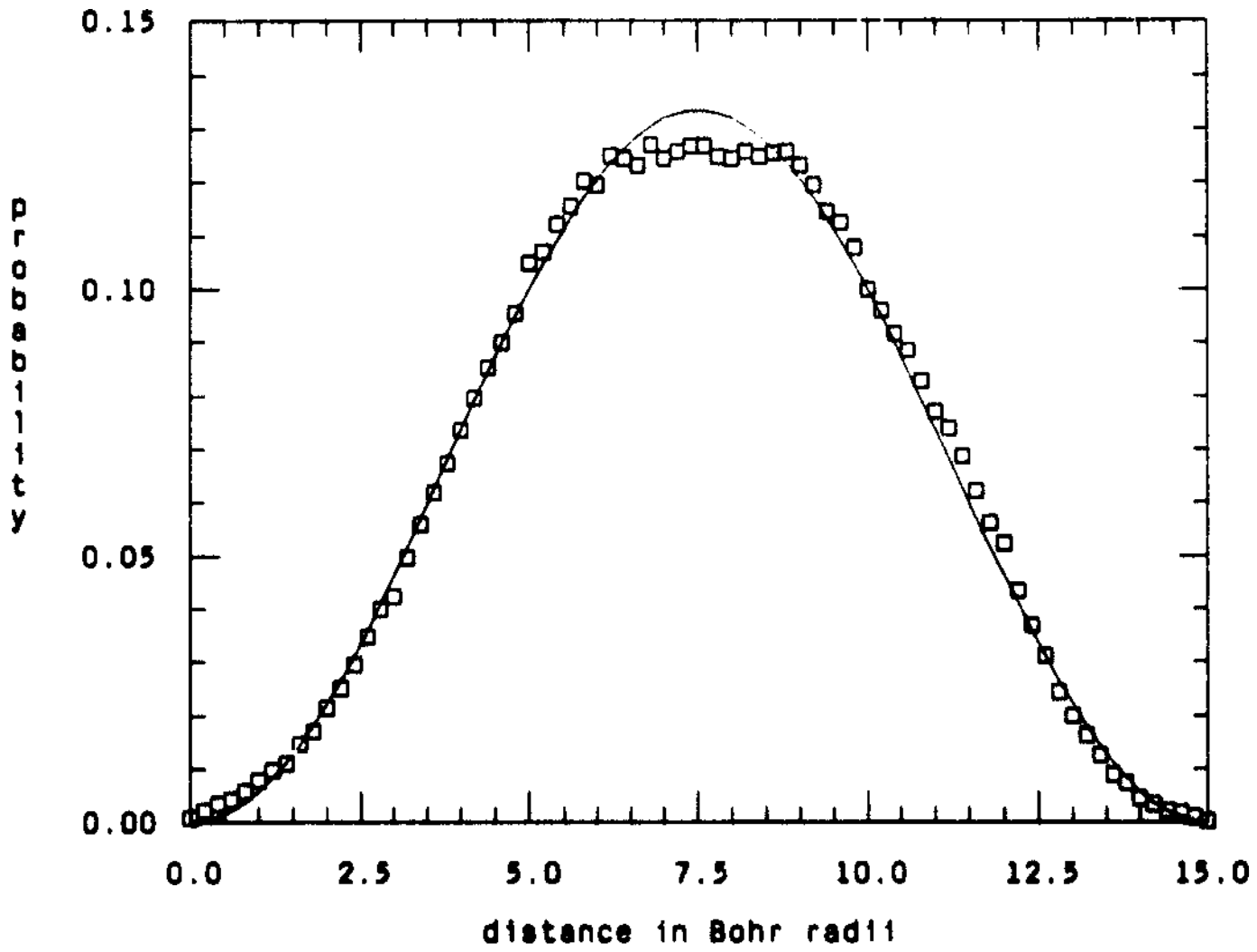
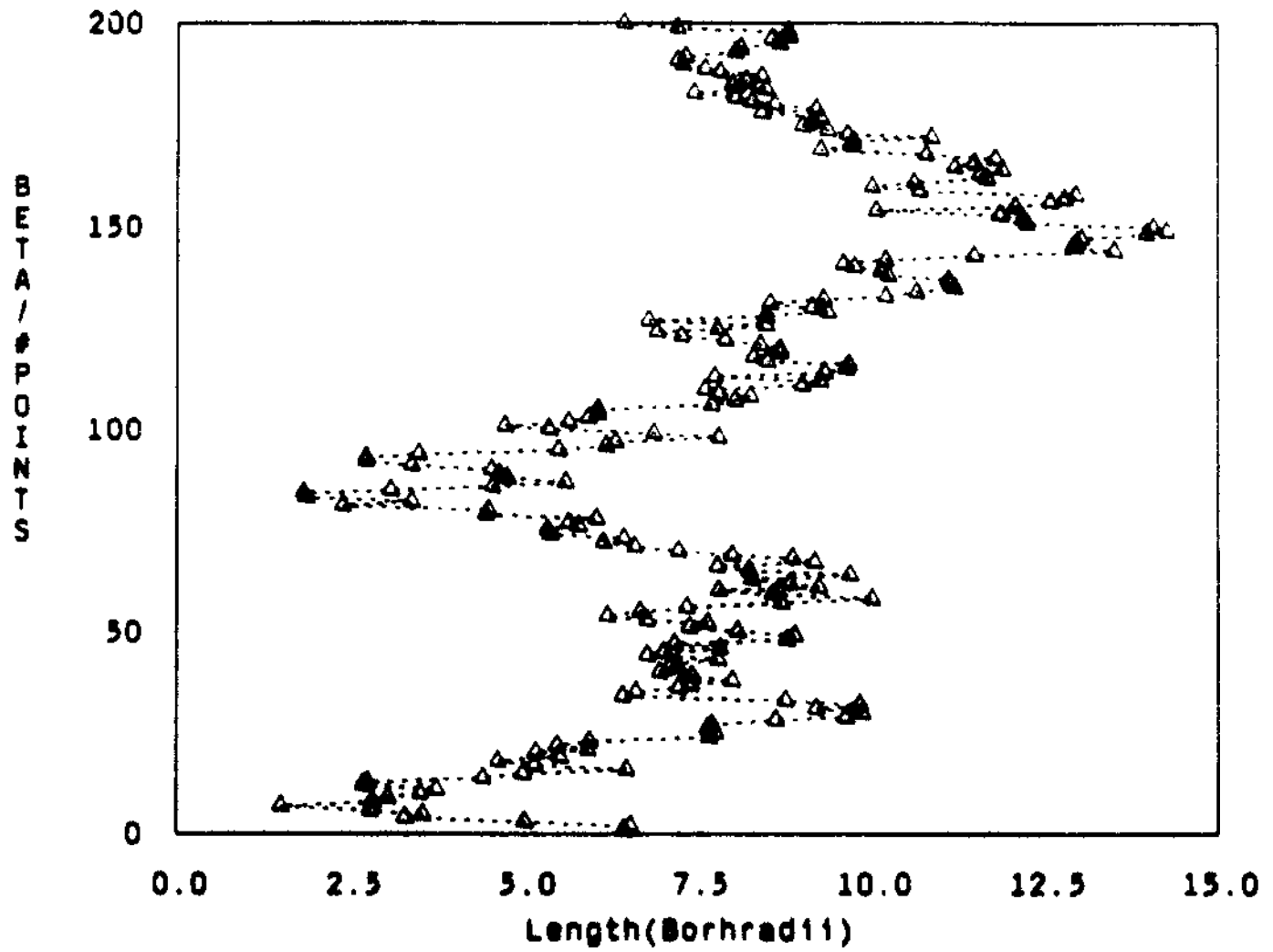


FIGURE 2: The Path Integral behaves like a polymer chain as shown in the simulations on the following page. The dotted lines are the imaginary spring bonds that connect the beads together. Plotted along the y axis is the imaginary time parameter, $\Delta\beta = \beta/p$. The value of for the system will determine the local energy of an individual bead. As the number of bead grows, the value of $1/\Delta\beta$ grows larger, and consequently it becomes harded to move the beads around. Here shown is two single particle chains at different times during the simulation, but both near equilibrium. As time goes on, the polymer eventually samples all of phase space. To get the equilibrium properties of the system, we must start and end the chains at the same place in the box however, that position does not have to be unique.


200
150
100
50
0
BETA/#POINTS
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Length(Borhradii)

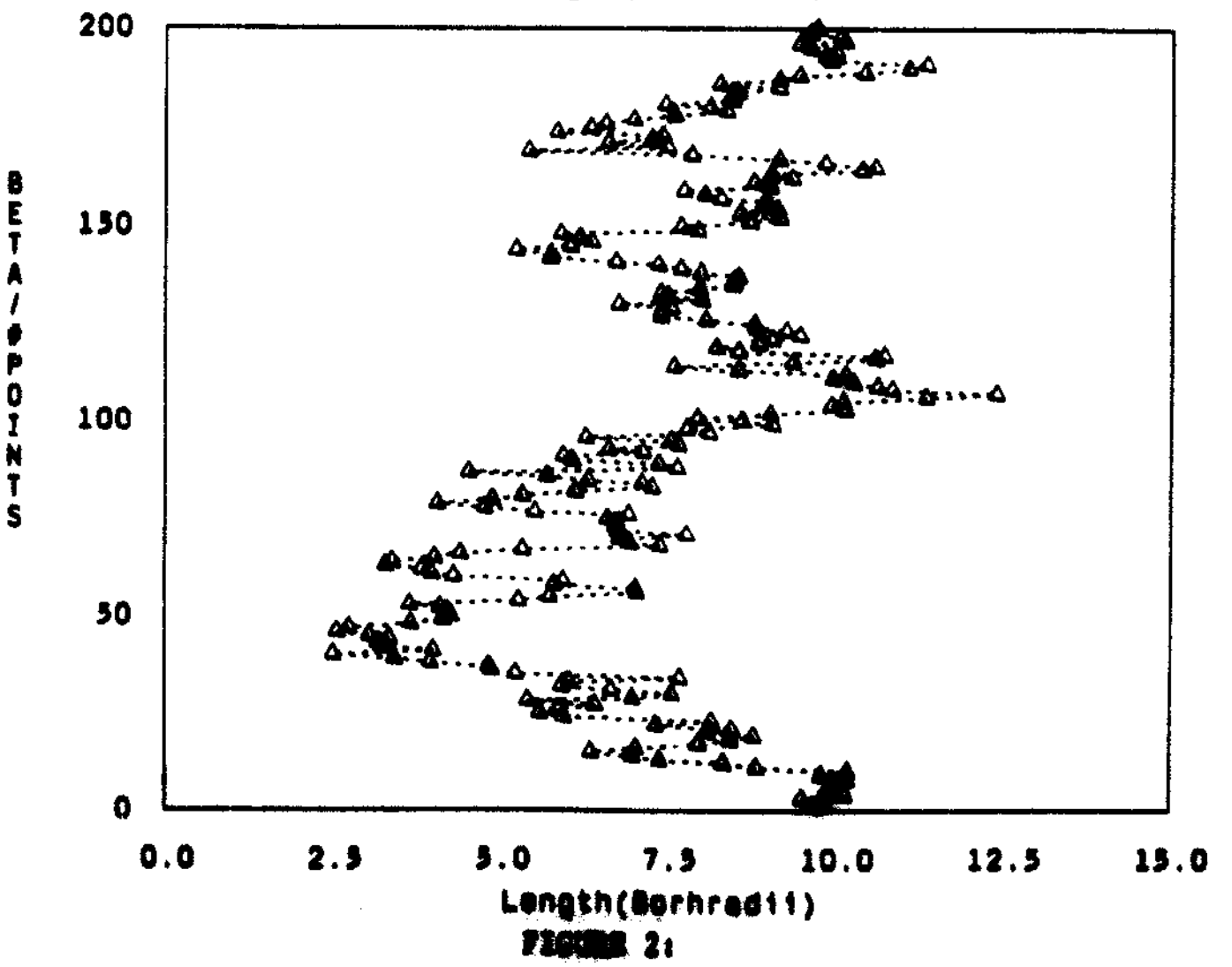
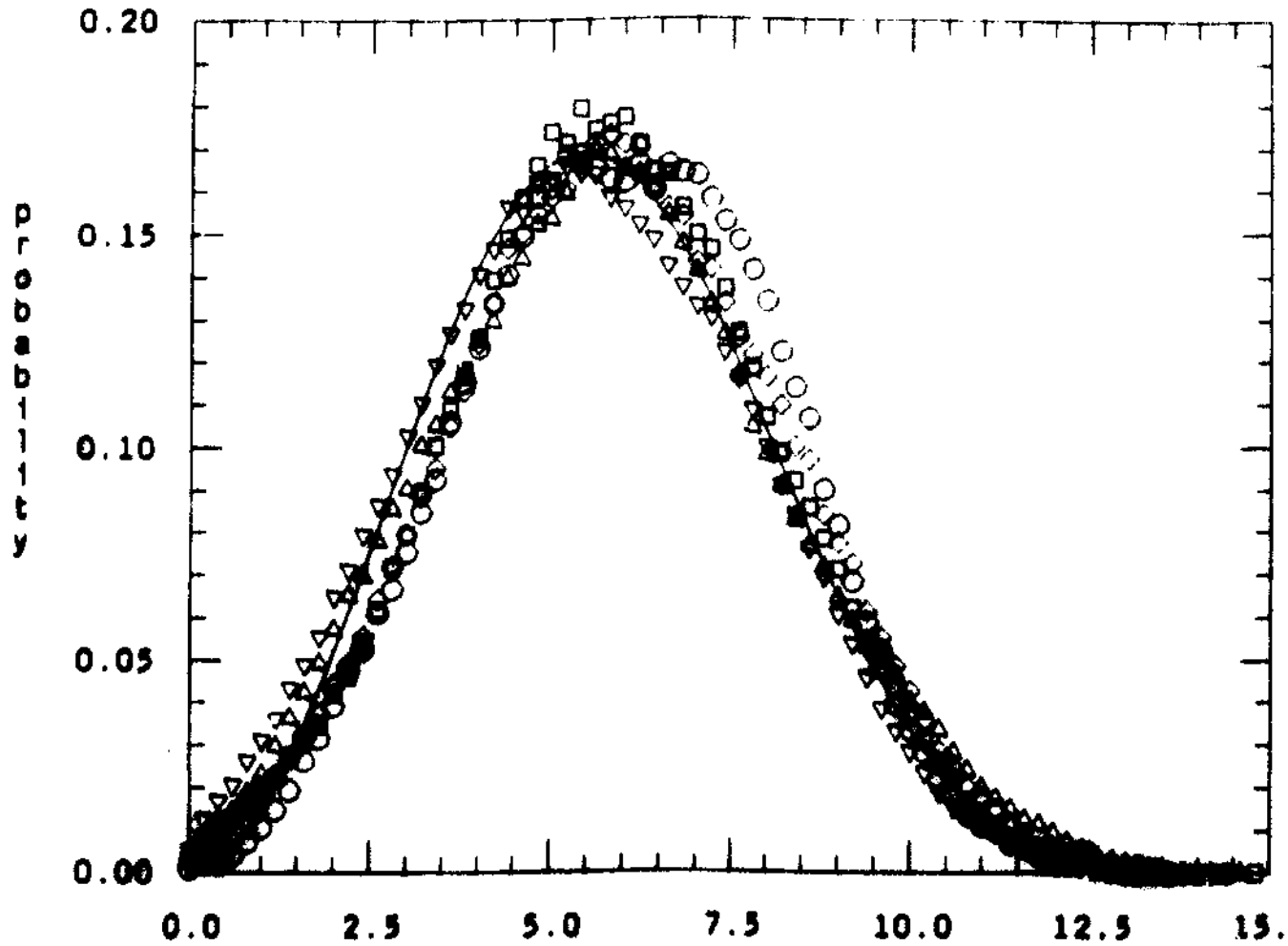

200
150
100
50
0
BETA/#POINTS
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Length(Borhradii)

FIGURE 2:

FIGURE 3 g(r) for two Fermions



SIMULATION CONSTANTS:

temperture= 2000K ; beta= 157.873 [(Hatrees)E-1] ; No. of beads= 200

step size= 1.0 [Bohr radii]

FIGURE 3: Shown here is the simulation results for several two particle g(r) distributions. Although there is quite a bit of flucuation, the overall structure is in good agreement with the analytic result given by a solid line.

TABLE 1 : The average energies of a two Fermion system

0.787	Hartrees
0.789	Hartrees
0.875	Hartrees
0.845	Hartrees
0.818	Hartrees
0.831	Hartrees
0.711	Hartrees

```
c          ***************** path.for ************************************
c          This program calculates several paths using Feynmans
c          path integrals and collects information on the distances
c          of each point in last 10 paths. Also, two path are written.
c          ---------------------------------------------------------------
          REAL XY(1000,0:5),DIS(0:1000,2),TDEN(0:1000,2)
c          ---------------------------------------------------------------
c         Inialize the array xy with its correct boundries
c          ---------------------------------------------------------------
           open(unit=15,file='start.dat',status='old')
           read(15,*)iv
           read(15,*)xmax
           read(15,*)num
           read(15,*)box
           read(15,*)beta
           read(15,*)ibead
           delbeta=beta/(ibead-1)
           write(6,*)'delbeta=',delbeta
           read(15,*)keq
           read(15,*)kstat
           close(unit=15)
           write(6,*)'Data input complete'
           dxmax=xmax/num*1.0
           do 100 n=1,num+1
            do 110 j=1,ibead
             if(n.eq.num+1)then
               xy(j,n)=xmax
             else
               rx=ran(iv)
               xy(j,n)=(n-1+rx)*dxmax
               if(j.eq.ibead)then
                    xy(j,n)=xy(1,n)
               end if
             end if
110          continue
100         continue
            write(6,*)'Initialization is done'
c           ---------------------------------------------------------------
c          Calculating kinetic energy input path.
c           ---------------------------------------------------------------
            do 1000 k=1,keq
            do 1020 n=1,num
             do 1030 i=1,ibead-1
              if(i.eq.1)then
              xt1=(xy(i+1,n)-xy(i,n))**2+(xy(i,n)-xy(ibead-1,n))**2
              xt1=(.5/delbeta)*xt1
              else
              xt1=(xy(i+1,n)-xy(i,n))**2+(xy(i,n)-xy(i-1,n))**2
              xt1=(.5/delbeta)*xt1
              end if
c           ---------------------------------------------------------------
c            MOVE BY MONTE CARLO: Choose a move on one particle;
c            denote this by rmove.
c           ---------------------------------------------------------------
              dr=(ran(iv)-.5)*box
              rmove=xy(i,n)+dr
c           ---------------------------------------------------------------
c             CHECKING FOR BOUNDRY AND EXCHANGE CONDITIONS
c           ---------------------------------------------------------------
              if(rmove.le.xy(i,n-1).or.rmove.ge.xy(i,n+1))then
```

```
            rmove=xy(i,n)
           end if
c          ------------------------------------------------
c            CALCULATE NEW ENERGY AND TEST BY MONTE CARLO
c          ------------------------------------------------
           if(i.eq.1)then
           xt2=(xy(i+1,n)-rmove)**2+(rmove-xy(ibead-1,n))**2
           xt2=xt2*(.5/delbeta)
           else
           xt2=(xy(i+1,n)-rmove)**2+(rmove-xy(i-1,n))**2
           xt2=(.5/delbeta)*xt2
           end if
c          ------------------------------------------------
c          TEST THE CONFIGUATION AND ACCEPT BY MONTE CARLO.
c          ------------------------------------------------
            dglop=xt2-xt1
          IF(dglop.LE.0.0)THEN
           xy(i,n)=rmove
           if(i.eq.1)then
            xy(ibead,n)=rmove
           end if
          ELSE
           sz=-log(RAN(IV))
           IF(sz.gt.dglop)THEN
                  xy(i,n)=rmove
                  if(i.eq.1)then
                   xy(ibead,n)=rmove
                  end if
            end if
          end if
1030      CONTINUE
1020      CONTINUE
1000      CONTINUE
          vrite(6,*)'Equilibriations are done'
c         ------------------------------------------------------
c          Equilibriations are done; Start collecting statistics.
c         ------------------------------------------------------
          open(unit=21,file='enep.dat',status='new')
          energy=0.0
          tot=0.0
          do 2000 k=1,kstat
          do 2020 n=1,num
           do 2030 i=1,ibead-1
            if(i.eq.1)then
            xt1=(xy(i+1,n)-xy(i,n))**2+(xy(i,n)-xy(ibead-1,n))**2
            xt1=(.5/delbeta)*xt1
            else
            xt1=(xy(i+1,n)-xy(i,n))**2+(xy(i,n)-xy(i-1,n))**2
            xt1=(.5/delbeta)*xt1
            end if
c           ------------------------------------------------
c           MOVE BY MONTE CARLO: Choose a move on one particle;
c           denote this by rmove.
c           ------------------------------------------------
            dr=(ran(iv)-.5)*box
            rmove=xy(i,n)+dr
c           ------------------------------------------------
c            CHECKING FOR BOUNDRY AND EXCHANGE CONDITIONS
c           ------------------------------------------------
            if(rmove.le.xy(i,n-1).or.rmove.ge.xy(i,n+1))then
```

```
            rmove=xy(i,n)
           end if
c          ------------------------------------------------
c            CALCULATE NEW ENERGY AND TEST BY MONTE CARLO
c          ------------------------------------------------
           if(i.eq.1)then
           xt2=(xy(i+1,n)-rmove)**2+(rmove-xy(ibead-1,n))**2
           xt2=xt2*(.5/delbeta)
           else
           xt2=(xy(i+1,n)-rmove)**2+(rmove-xy(i-1,n))**2
           xt2=(.5/delbeta)*xt2
           end if
c          ------------------------------------------------
c          TEST THE CONFIGUATION AND ACCEPT BY MONTE CARLO.
c          ------------------------------------------------
            dglop=xt2-xt1
          IF(dglop.LE.0.0)THEN
            xy(i,n)=rmove
            if(i.eq.1)then
             xy(ibead,n)=rmove
            end if
          ELSE
            zz=-log(RAN(IW))
            IF(zz.gt.dglop)THEN
                   xy(i,n)=rmove
                   if(i.eq.1)then
                     xy(ibead,n)=rmove
                   end if
             end if
             reject=reject+1.0
           end if
c          ------------------------------------------------
c          COLLECTING STATISTICAL DATA
c          ------------------------------------------------
           ktest=(k/100)*100
           t=k*1.0
           sum=0.0
           do 182 m=1,num
           do 181 j=1,ibead-1
            sum=(xy(j+1,m)-xy(j,m))**2+sum
181        continue
182        continue
           sum=0.5*num/delbeta-sum*0.5/(delbeta*beta)
           energy=sum+energy
           senergy=(sum)**2+senergy
c          ------------------------------------------------
c          collecting distribution information
c          ------------------------------------------------
           tot=tot+1.0
2030       continue
2020       continue
           if(k.eq.ktest)then
             vrite(6,*)k
             ene=energy/tot
             avar=sqrt((senergy/tot-(ene)**2)/tot)
             vrite(21,*)k,ene,avar
           end if
           call dt (xy,dis,ibead,num,xmax,k,kstat)
c          call tdensity (xy,tden,ibead,num,xmax,k,kstat)
2000       continue
```

```
c         ------------------------------------------------
c         writing the new accepted path into a data file
c         ------------------------------------------------
          vrite(6,*)'I am writing a path'
          ave=energy/tot
         OPEN(UNIT=19,FILE='path100.DAT',STATUS='NEW')
          do 320 n=1,num
         do 310 i=1,ibead
            vrite(19,*)xy(i,n)
310       continue
320       continue
          ratio=reject/tot
          vrite(6,*)'rejection ratio for 100 was ',ratio
          close(UNIT=19)
c         ------------------------------------------------
c         G(R) LOOP
c         ------------------------------------------------
          open(unit=16,file='disp.dat',status='new')
          dnorm=(ibead-1)*kstat*0.2
          limit=nint(xmax/0.2)
          do 500 i=0,limit
           if(dis(i,2).ne.0.0)then
           dis(i,2)=dis(i,2)/dnorm
           vrite(16,*)dis(i,1),dis(i,2)
           end if
500       continue
c         ------------------------------------------------
c          LOOP FOR THE DENSITY OF BOTH PARTICLES
c         ------------------------------------------------
c         limit=nint(xmax/0.2)
c         do 503 i=0,limit
c          if(tden(i,2).ne.0.0)then
c          tden(i,2)=tden(i,2)/dnorm
c          vrite(76,*)tden(i,1),tden(i,2)
c          end if
c503      continue
c         ------------------------------------------------
C          VRITE DATA
c         ------------------------------------------------
          do 340 n=1,num
             vrite(6,15)n,xy(1,n)
15           format(5x,'electron',i3,1x,'init. pos. vas',e12.6)
340       continue
          do 350 n=1,num
             vrite(6,16)n,xy(ibead,n)
16           format(5x,'electron',i3,1x,'final pos. vas',e12.6)
350       continue
          vrite(6,13)xmax,delbeta,box,beta,ibead,keq,kstat,ave
13        format(5x,'box length in Bohr radii=',e12.6,/,5x
     %   ,'del beta=',e12.6,/,5x
     %   ,'the largest movement possible ',e12.6,/,5x
     %   ,'beta vas=',e12.6,/,5x
     %   ,'the number of beads',i8,/,5x
     %   ,'equilibration passes=',i8,/,5x
     %   ,'statistical passes =',i8,/,5x
     %   ,'average energy=',e12.6)
          stop
          end
```

```
      SUBROUTINE DT(XY,DIS,IBEAD,NUM,XMAX,K,KSTAT)
c     ******** THIS IS MY DISTRIBUTION FUNCTION **********
      real dis(0:1000,2),xy(1000,0:5)
      do 250 n=1,num-1
      do 251 i=1,ibead-1
         r=xy(i,n+1)-xy(i,n)
         l=nint(r/0.2)
         dis(l,2)=dis(l,2)+1.0
         dis(l,1)=l*0.2
251   continue
250   continue
      return
      end



      SUBROUTINE TDENSITY(XY,TDEN,IBEAD,NUM,XMAX,K,KSTAT)
c     ******** THIS IS MY Density FUNCTION **********
      real tden(0:1000,2),xy(1000,0:5)
      do 250 n=1,num
      do 251 i=1,ibead-1
         r=xy(i,n)
         l=nint(r/0.2)
         tden(l,2)=tden(l,2)+1.0
         tden(l,1)=l*0.2
251   continue
250   continue
      return
      end
```