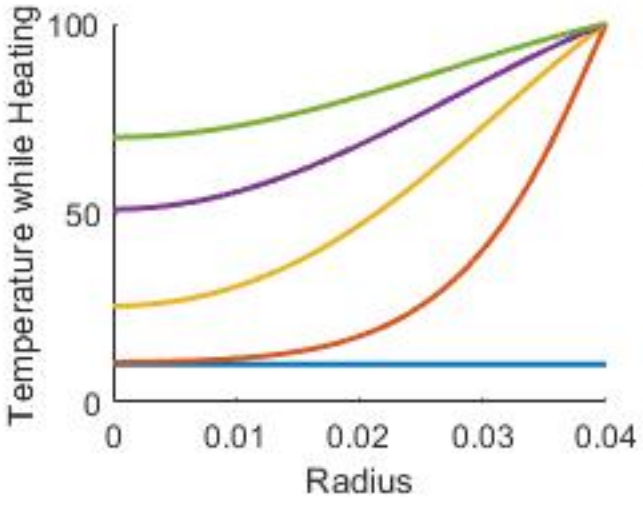
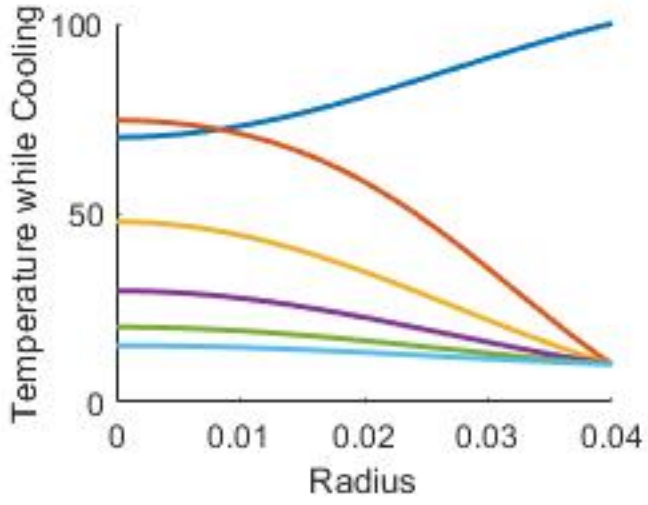
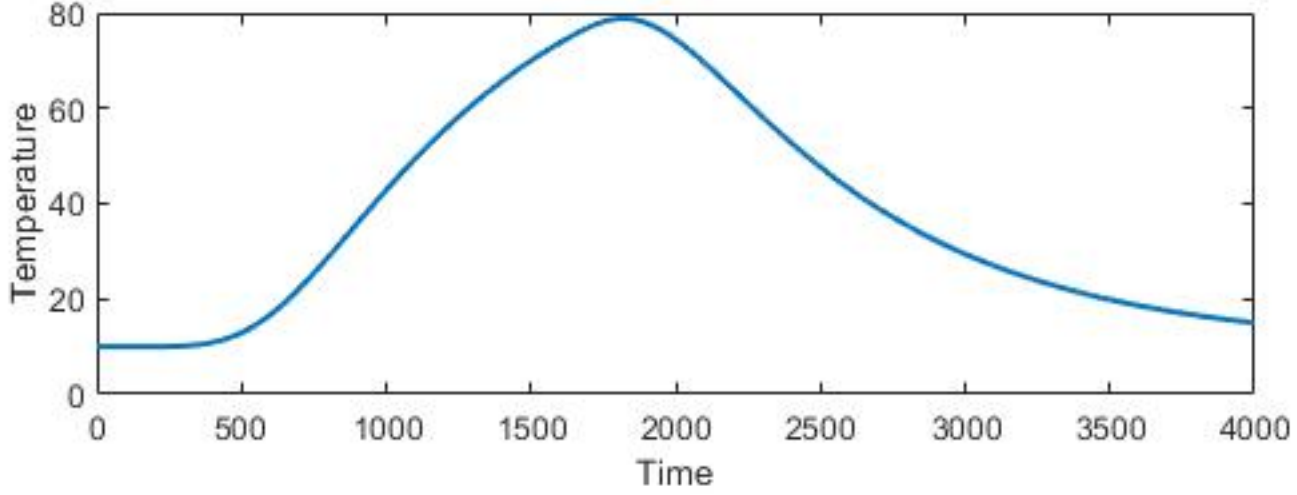
```
%*********************************************************************
% The following program calculates temperature distribution inside a
% a potato (initially room temperature)and dipped in water, also at room
% temperature. The water is slowly heated until boiling. Then the potato is
% removed from the boiling water and kept at room temperature.
%*********************************************************************

R=0.04;k=0.05;
K=0.8;c=3390;rho=1070;
a=K/(c*rho);
M=101;
h=R/(M-1);

%*************************************************************************
% R is the radius of the potato, K is its thermal conductivity, c its
% specific heat, and rho its mass density. M is the no. of discretization
% points along the radial line and N that of time. k=0.05 is each time
% step.
%*************************************************************************

alpha=a*k/(h^2);
u0=100;
u_enough=70;
x=0:h:R;
%u=zeros(M,N);
u_room=10;
u(1:M,1)=u_room;

%*********************************************************************
% Room temperature is 10 degrees celsius, u0=100 is the temperature of
% the boiling water.
%*********************************************************************
subplot(2,2,1)

n=1;
while u(1,n)<u_enough
    for m=M-1:-1:2
        u(m,n+1)=(1-2*alpha)*u(m,n)+alpha*((m-2)*u(m-1,n)+m*u(m+1,n))/(m-1);
    end

    u(1,n+1)=u(2,n+1);

    if u(M,n)< u0
       u(M,n+1)=u(M,n)+k/4;
    else
        u(M,n+1)=u(M,n);
    end
    n=n+1;

end
HeatingStopIteration=n;
```

```
N=n;

hold on

for i=1:floor(N/4):N
plot(x,u(:,i),'Linewidth',1.5)
end
xlabel("Radius");
ylabel("Temperature while Heating");
hold off
%*********************************************************************
% The if-else statement above raises the temperature of potato from 10
% to 100 till centre is at u_enough degrees. The surface temperature
% increases k/4 degrees in one time step of k seconds each. Thus, it checks
% if u(M,n) has reached u0.
%*********************************************************************

u0=u_room;
u_enough=u_room+5;
u(M,n)=u_room;
while u(1,n)>u_enough
    for m=M-1:-1:2
        u(m,n+1)=(1-2*alpha)*u(m,n)+alpha*((m-2)*u(m-1,n)+m*u(m+1,n))/(m-1);
    end

    u(1,n+1)=u(2,n+1);

    if u(M,n)> u0
       u(M,n+1)=u(M,n)-k/4;
        else u(M,n+1)=u(M,n);
    end
    n=n+1;

end

CoolingStopIteration=n;
subplot(2,2,2)
N=n;
hold on

for i=HeatingStopIteration-2:floor((N-HeatingStopIteration)/5):N
plot(x,u(:,i),'Linewidth',1.5)
end
xlabel("Radius");
ylabel("Temperature while Cooling");
hold off
%*********************************************************************
% The if-else statement above raises the temperature of potato from 100
% to 10 till centre is at u_room+5 degrees. The surface temperature
% decreases k/4 degrees in one time step of k seconds each. Thus, it checks
% if u(M,n) has reached u_room.
```

```
%**************************************************************************

Heating_Time=k*(HeatingStopIteration-1)
Cooling_Time=k*(CoolingStopIteration-HeatingStopIteration-1)

%**************************************************************************
% The above code tells time taken to heat and cool
%**************************************************************************

subplot(2,2,[3,4])
for i=1:n
    time(i)=i*k;
end
plot(time,u(1,:),'Linewidth',1.5);
xlabel("Time");
ylabel("Temperature");


%**************************************************************************
% The temperature as a function of time
%**************************************************************************
```

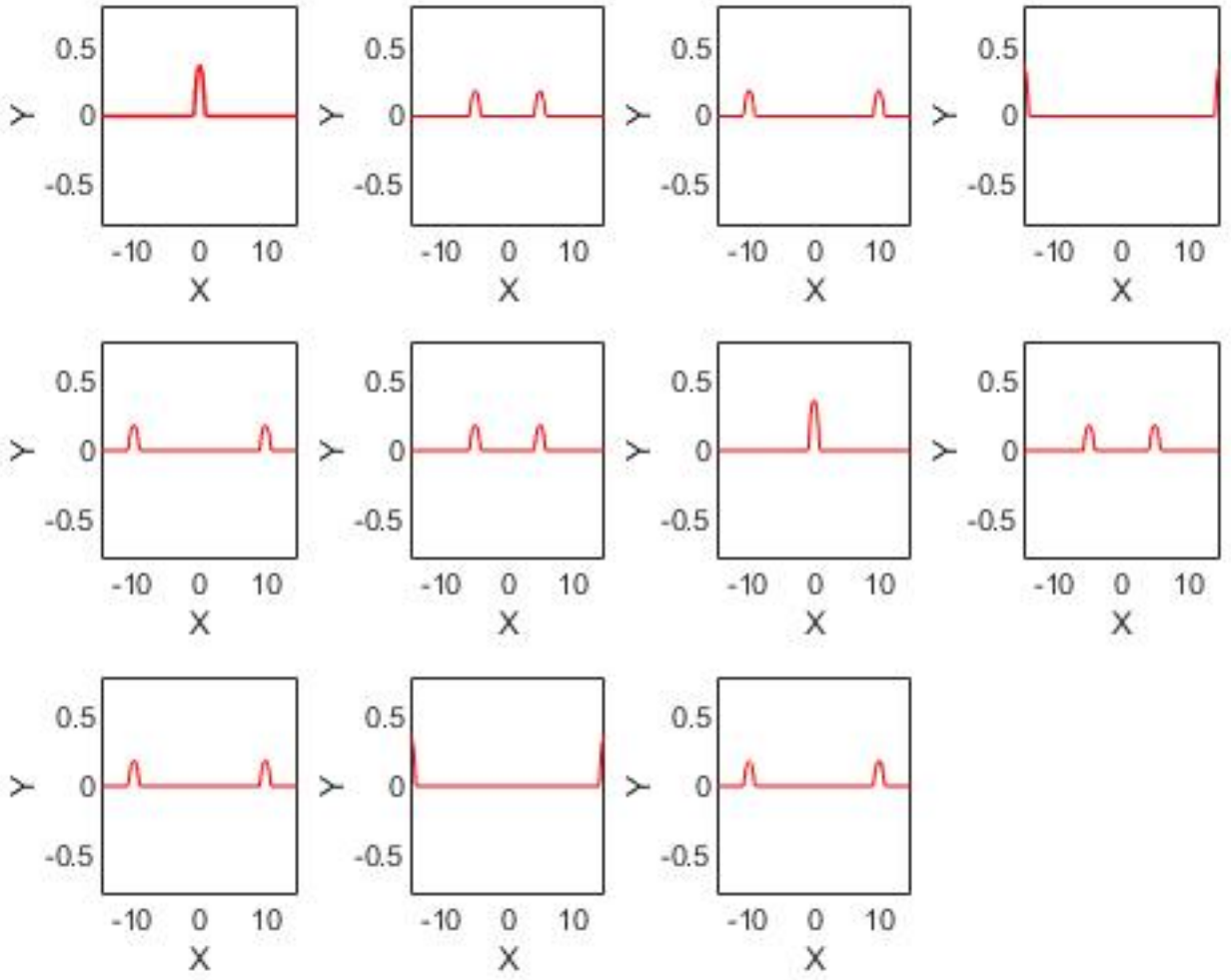

Temperature while Heating
100
50
0
0
0.01
0.02
0.03
0.04
Radius


Temperature while Cooling
100
50
0
0
0.01
0.02
0.03
0.04
Radius


Temperature
80
60
40
20
0
0
500
1000
1500
2000
2500
3000
3500
4000
Time

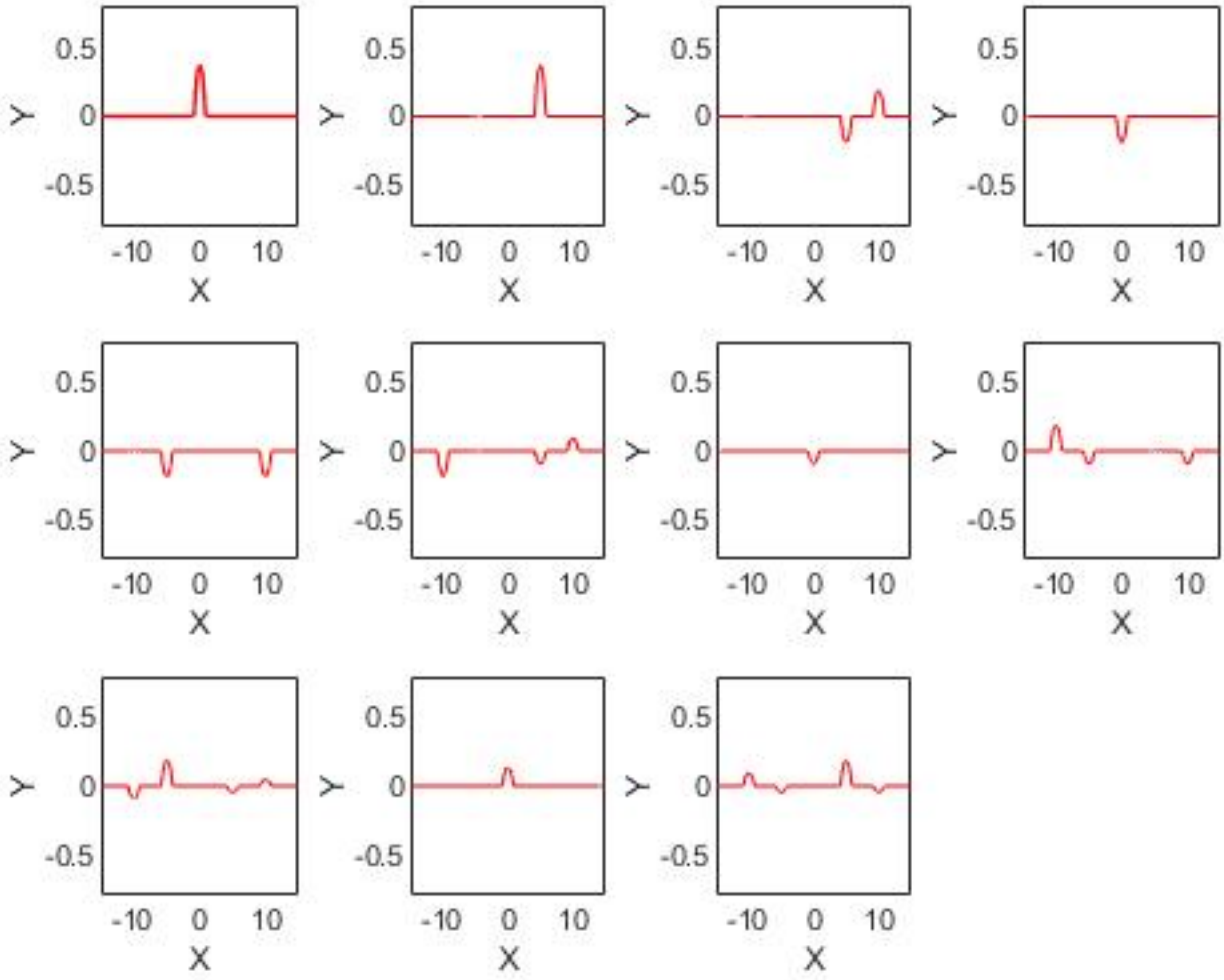
```
%************************************************************************
%In this program, both ends of the string are free to move vertically.
%************************************************************************

c=1;
L=30;h=0.01;
M=L/h+1;
x=linspace(-L/2,L/2,M);
k=0.01;
a=c*k/h;

%************************************************************************
% All the string parameters as well as step size for x (h) and that for t
% (k) are set above. T is the tension and mu is the mass per unit length.
%************************************************************************

u(1,:)=0;u(M,:)=0;
p=1+(L/2-1)/h;q=1+(L/2+1)/h;
u(p,1)=0;u(q,1)=0;
u(p,2)=0;u(q,2)=0;

%************************************************************************
% The initial shape of the string, a small pulse at the centre, are fixed
% above.
%************************************************************************

for n=p+1:q-1
u(n,1)=exp(-1/(1-x(n)^2));
u(n,2)=u(n,1);
end

%************************************************************************
% Initial velocity profile of the string is set above.
%************************************************************************

subplot(3,4,1)
plot(x,u(:,1),'r','Linewidth',1)
xlabel("X");
ylabel("Y");
axis([-L/2 L/2 -0.8 0.8])

for i=1:10
T(i)=5*i;
N(i)=T(i)/k+1;
u(1,:)=0;u(M,1)=0;u(M,2)=0;
for n=2:N(i)
    u(M,n+1)=a^2*(2*u(M-1,n)-u(M,n-1))+2*(1-a^2)*u(M,n);
    u(1,n+1)=a^2*(2*u(2,n)-u(1,n-1))+2*(1-a^2)*u(1,n);
    for m=2:M-1
        u(m,n+1)=a^2*(u(m+1,n)+u(m-1,n))+2*(1-a^2)*u(m,n)-u(m,n-1);
    end
```

```
52 end
53 subplot(3,4,i+1)
54 plot(x,u(:,N(i)),'r')
55 axis([-L/2 L/2 -0.8 0.8])
56 xlabel("X");
57 ylabel("Y");
58 end
59 %*********************************************************************
60 % The equations are solved up to different time instants and the pulse
61 % position and shape plotted.
62 %*********************************************************************
```


0.5
0
-0.5
Y
-10
0
10
X

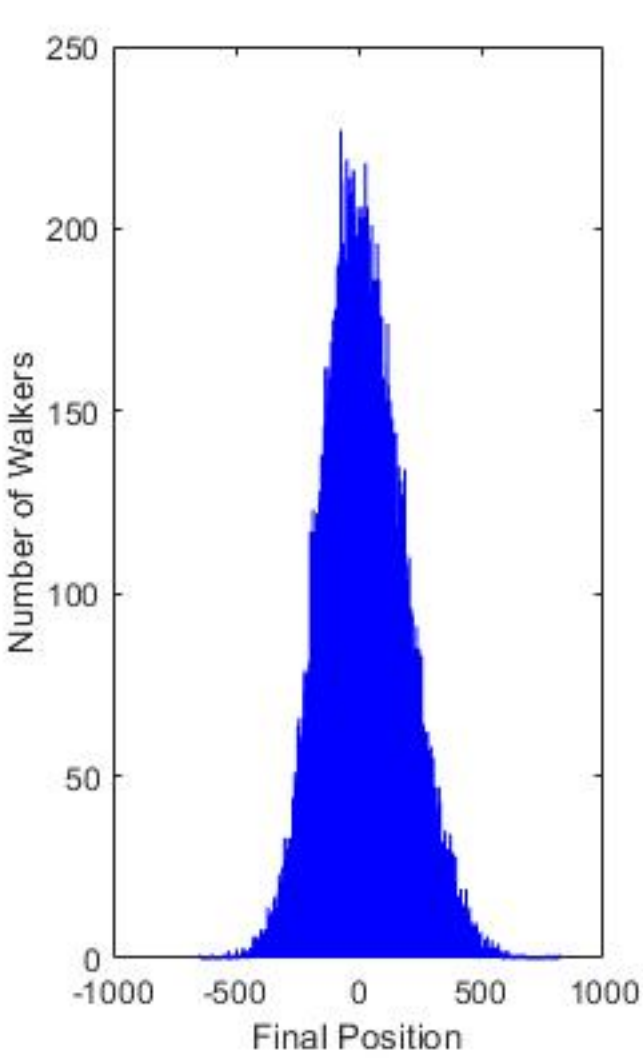
```
%************************************************************************
% This program is about wave motion in a string with both ends fixed with a
% bead a position 3L/4.
%************************************************************************

T=0.1;mu=0.1;
c=sqrt(T/mu);
L=30;h=0.01;
bead_pos=L/4;
M=L/h+1;
x=linspace(-L/2,L/2,M);
bead_pos_M=0;
for i=1:M
    if x(i)<bead_pos
        bead_n=x(i+2);
        bead_p=x(i);
        bead_pos_M=i+1;
    end
end
k=0.01;
a=c*k/h;
%************************************************************************
% All the string parameters as well as step size for x (h) and that for t
% (k) are set above. T is the tension and mu is the mass per unit length.
%************************************************************************

u(1,:)=0;u(M,:)=0;
p=1+(L/2-1)/h;q=1+(L/2+1)/h;
u(p,1)=0;u(q,1)=0;
u(p,2)=0;u(q,2)=0;

%************************************************************************
% The initial shape of the string, a small pulse at the centre, are fixed
% above.
%************************************************************************

for n=p+1:q-1
u(n,1)=exp(-1/(1-x(n)^2));
u(n,2)=u(n,1)*(1+2*k*x(n)/((1-x(n)^2)^2));
end

%************************************************************************
% Initial velocity profile of the string is set above.
%************************************************************************

subplot(3,4,1)
plot(x,u(:,1),'r','Linewidth',1)
xlabel("X");
ylabel("Y");
axis([-L/2 L/2 -0.8 0.8])

```

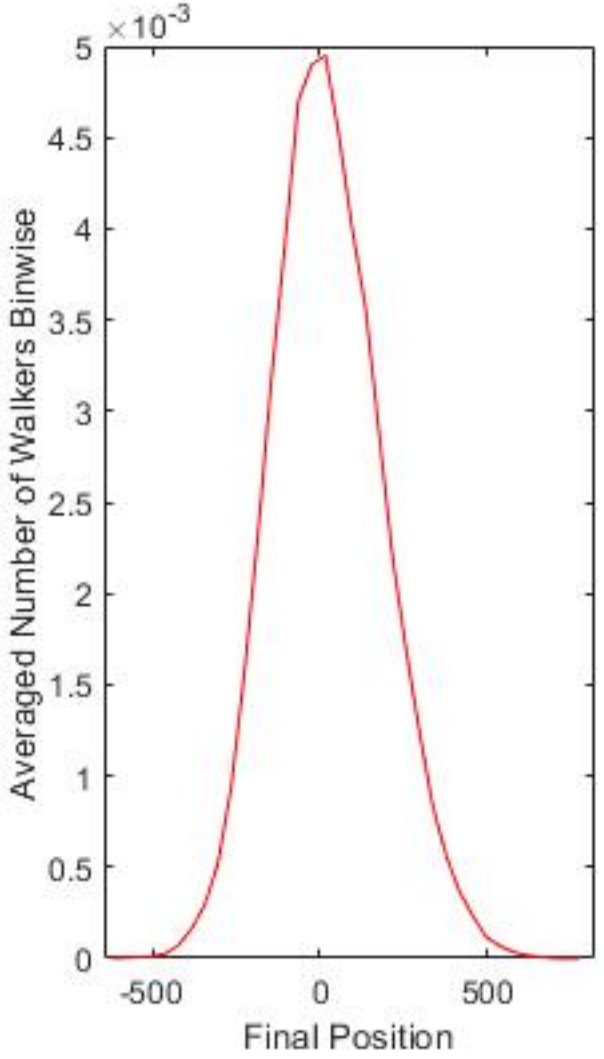
```
52 for i=1:10
53 T(i)=2.5*i*2;
54 N(i)=T(i)/k+1;
55 u(1,:)=0;u(M,:)=0;
56 for n=2:N(i)
57     for m=2:M-1
58         if m==bead_pos_M
59             u(m,n+1)=(u(m-1,n)+u(m+1,n))/2;
60         else
61         u(m,n+1)=a^2*(u(m+1,n)+u(m-1,n))+2*(1-a^2)*u(m,n)-u(m,n-1);
62         end
63     end
64 end
65 subplot(3,4,i+1)
66 plot(x,u(:,N(i)),'r')
67 xlabel("X");
68 ylabel("Y");
69 axis([-L/2 L/2 -0.8 0.8])
70 end
71
72 %**************************************************************************
73 % The equations are solved up to different time instants and the pulse
74 % position and shape plotted.
75 %**************************************************************************
```

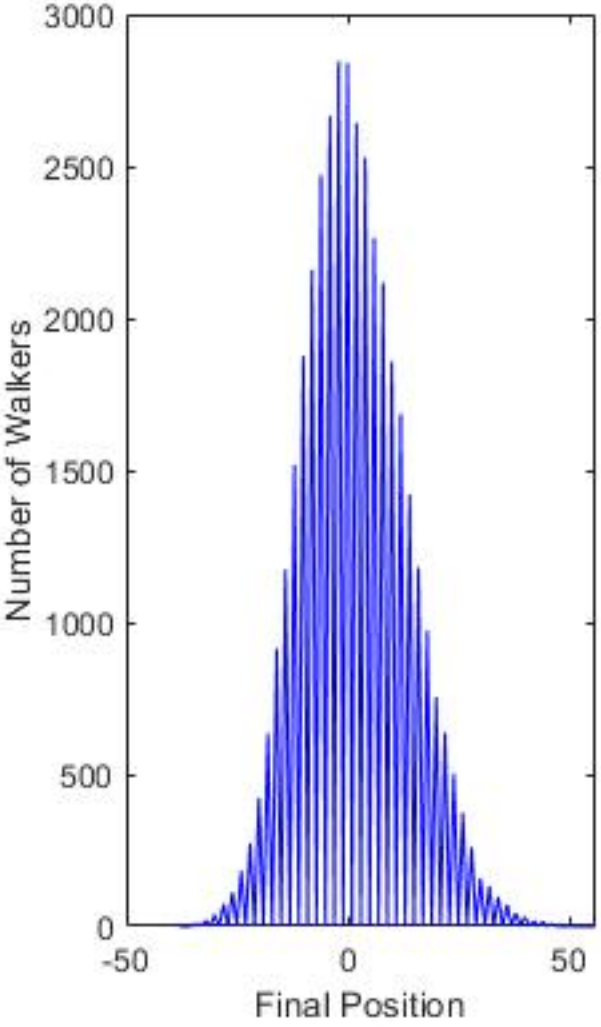

0.5
0
-0.5
Y
-10
0
10
X

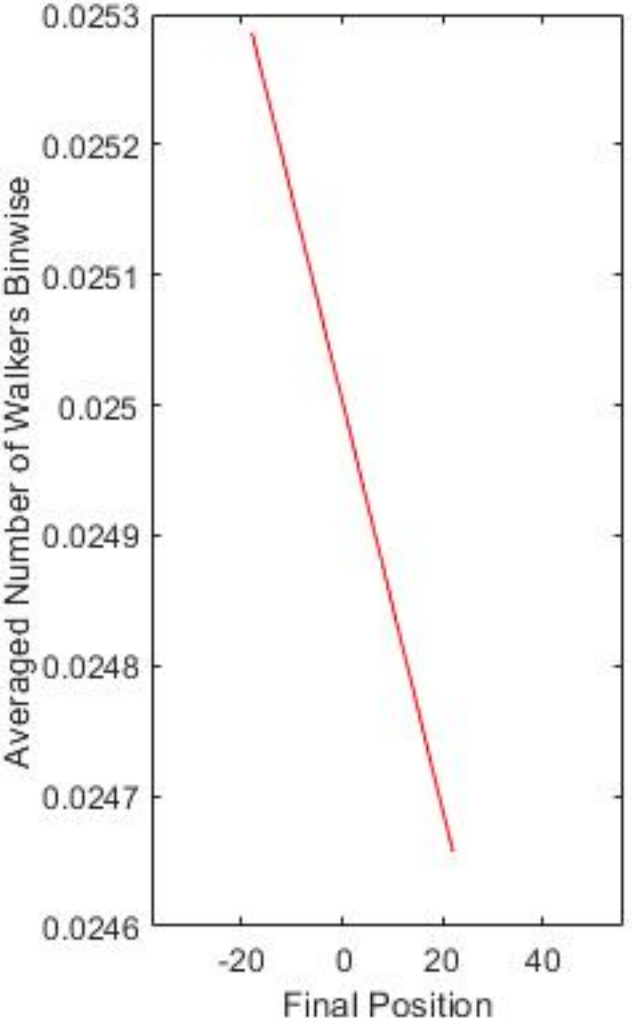
```
%**********************************************************************
% This programs calculates the probability distribution for the final
% displacement of a random walker in the large ensemble limit, which, thus,
% reduces to a normal distribution. The ensemble size is M (number of
% random walkers and the number of steps is N.
%**********************************************************************

N=20000;M=40000; %Change N to 100 to see 2nd result. Smaller N, higher
                 %probability to rach home
x=zeros(1,N);
%p=0.5;q=1-p;   % p and q are probabilities for step to the right and left
               % respectively.

for m= 1:M
    s=0;
for n=1:N
    if s>=0
        p=0.5/(1-0.5*s/N);
    else
        p=0.5;
    end
        q=1-p;
    r=rand;
    if r>=q
        s=s+1;
    else
        s=s-1;
    end
end
    x(m)=s;
end


x1=min(x);x2=max(x);
range=x1:1:x2;
count=zeros(1,x2-x1+1);

for m=1:M
    count(x(m)-x1+1)=count(x(m)-x1+1)+1;
end
subplot(1,2,1)
plot(range,count,'b')
xlabel("Final Position");
ylabel("Number of Walkers");
Reached_home=0;
for i=1:length(range)
    if range(i)==N/2
        Reached_home=count(i)
    end
end
Prob=Reached_home/M
```

```
52
53 Q=40;      % Q is the number of steps over which count is averaged out
54            % to smoothen the PDF.
55
56 count=count(1:1:x2-x1+1);
57
58 j=fix((x2-x1)/Q);
59
60 count1=zeros(1,j);
61
62 for i=1:j
63     for k=1:Q
64     count1(i)=count1(i)+ count(Q*(i-1)+k);
65     end
66     count1(i)=count1(i)/(M*Q/2);
67 end
68
69 y=x1+Q/2:Q:x2-Q/2;
70
71 hold on
72 subplot(1,2,2);
73 plot(y,count1,'r')
74 xlim([x1 x2])
75 xlabel("Final Position");
76 ylabel("Averaged Number of Walkers Binwise");
```


250
200
150
100
50
0
Number of Walkers
-1000
-500
0
500
1000
Final Position

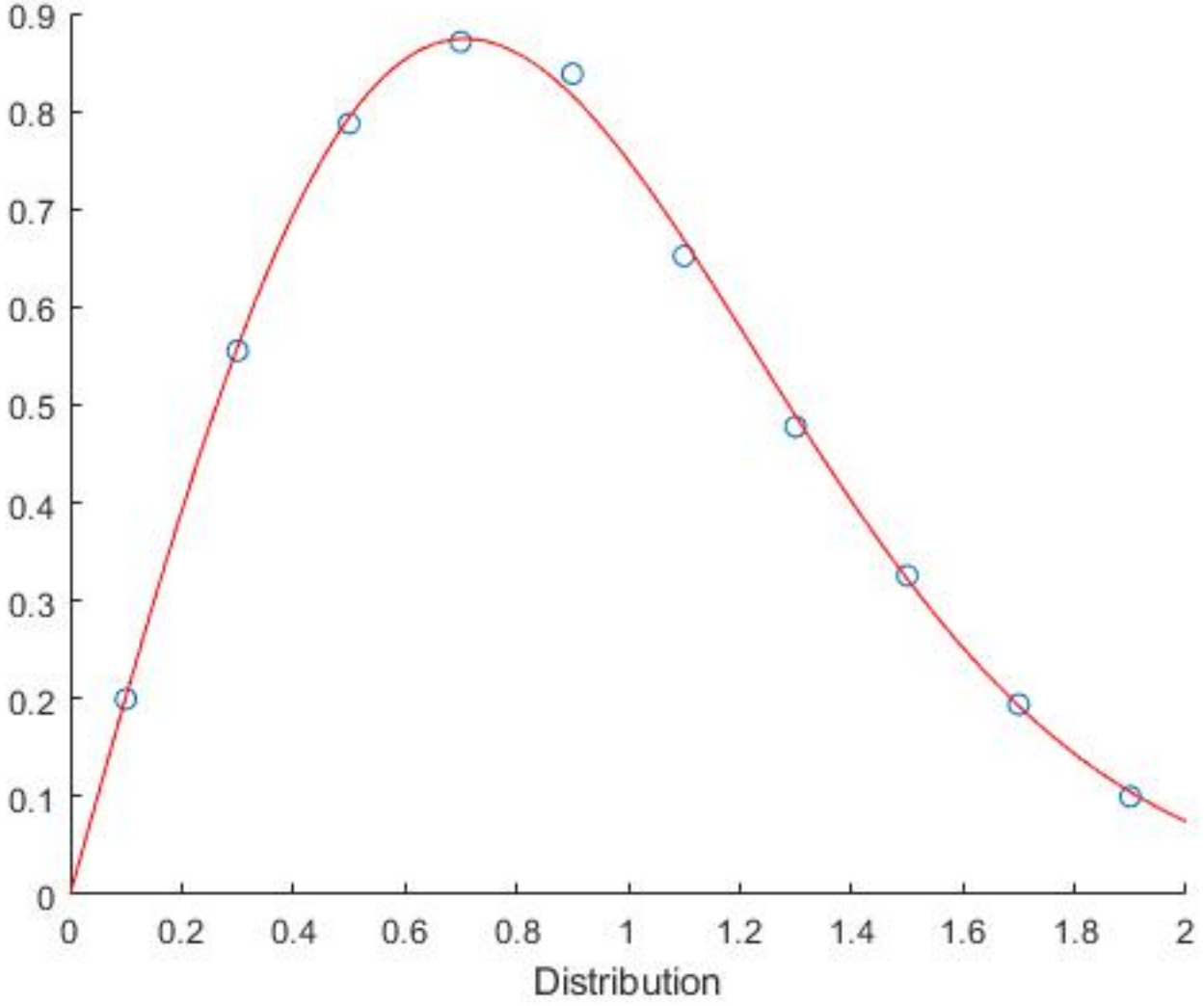

×10^-3
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Averaged Number of Walkers Binwise
-500
0
500
Final Position


3000
2500
2000
1500
1000
500
0
Number of Walkers
-50
0
50
Final Position

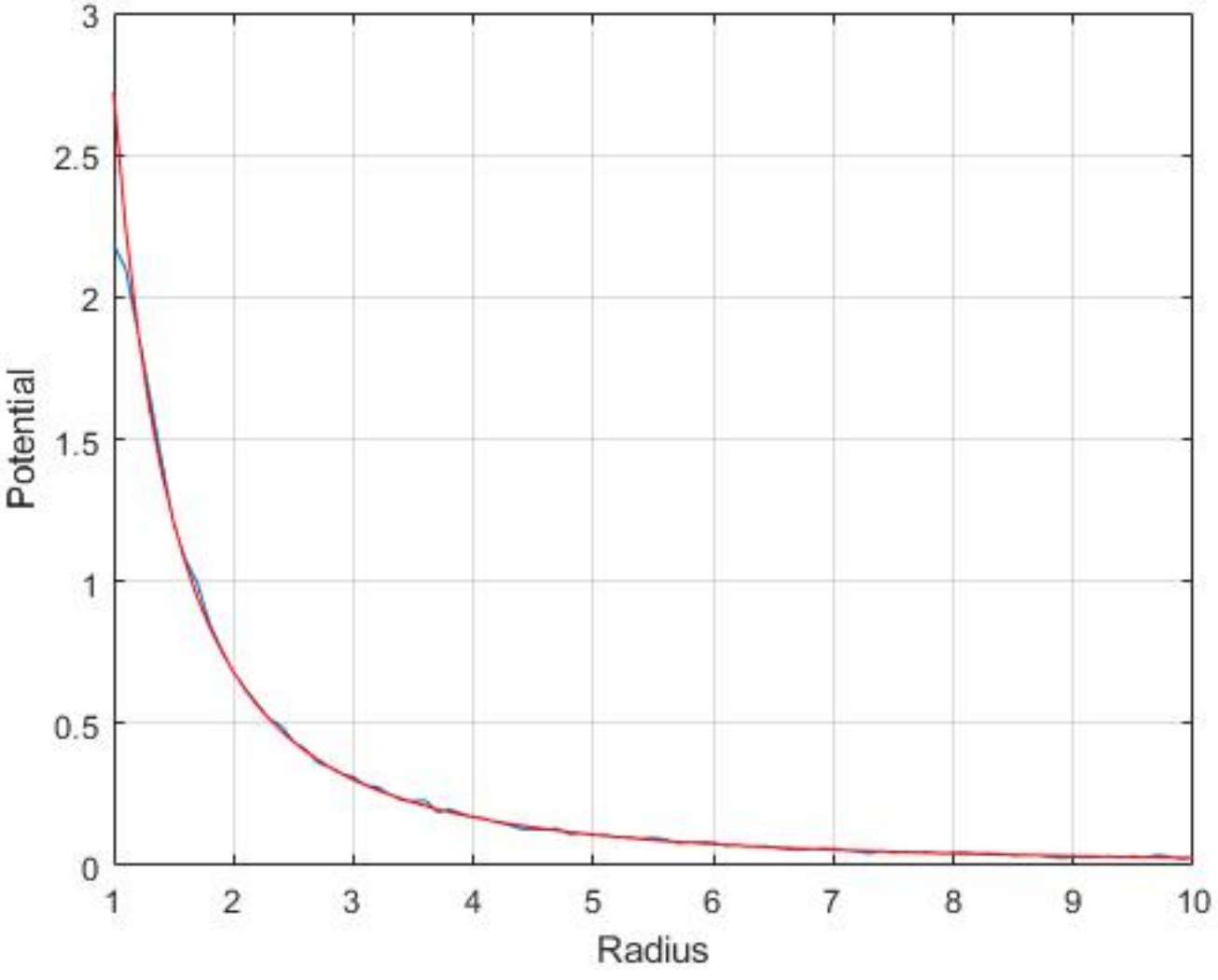

0.0253
0.0252
0.0251
0.025
0.0249
0.0248
0.0247
0.0246
Averaged Number of Walkers Binwise
-20
0
20
40
Final Position

```
%************************************************************************
% This program generates the normal distribution from a unform distribution
% and compares it with the actual normal distribution.
%************************************************************************

%sigma=1;mu=0;
Fun=@(x)x.*exp(-(x.^2))/(0.5 - 1/(2*exp(4)));
a=0;b=2;
h=0.0005;N=(b-a)/h+1;
x=a:h:b;

y(1)=Fun(x(1));

for n=2:N
    y(n)= y(n-1)+ h*Fun(x(n-1));
end

%************************************************************************
% The for loop above generates the integrated values of the normal
% distribution.
%************************************************************************

M=50000;

for k=1:M
z=rand;
for m=1:N-1
    if y(m)<=z & y(m+1)>=z
        s(k)=x(m);
        break
    end
end
end

%************************************************************************
% The above nested for loops map a random number y from a uniform
% distribution to the random number s statisfying normal distribution.
%************************************************************************

binsize=400*h;
K=(b-a)/binsize;
bin=zeros(1,K);
w=a+binsize/2:binsize:b-binsize/2;

for k=1:M
    p=fix((s(k)-a)/binsize)+1;
    bin(p)=bin(p)+1;
end

% The above loop averages the distribution over a bin of size 'binsize'.

```

```
52
53 PDF = bin/(M*binsize);
54
55
56 hold on
57 plot(w,PDF,'o')
58 plot(x,Fun(x),'r')
59 xlabel("Distribution");
60 hold off
61
```

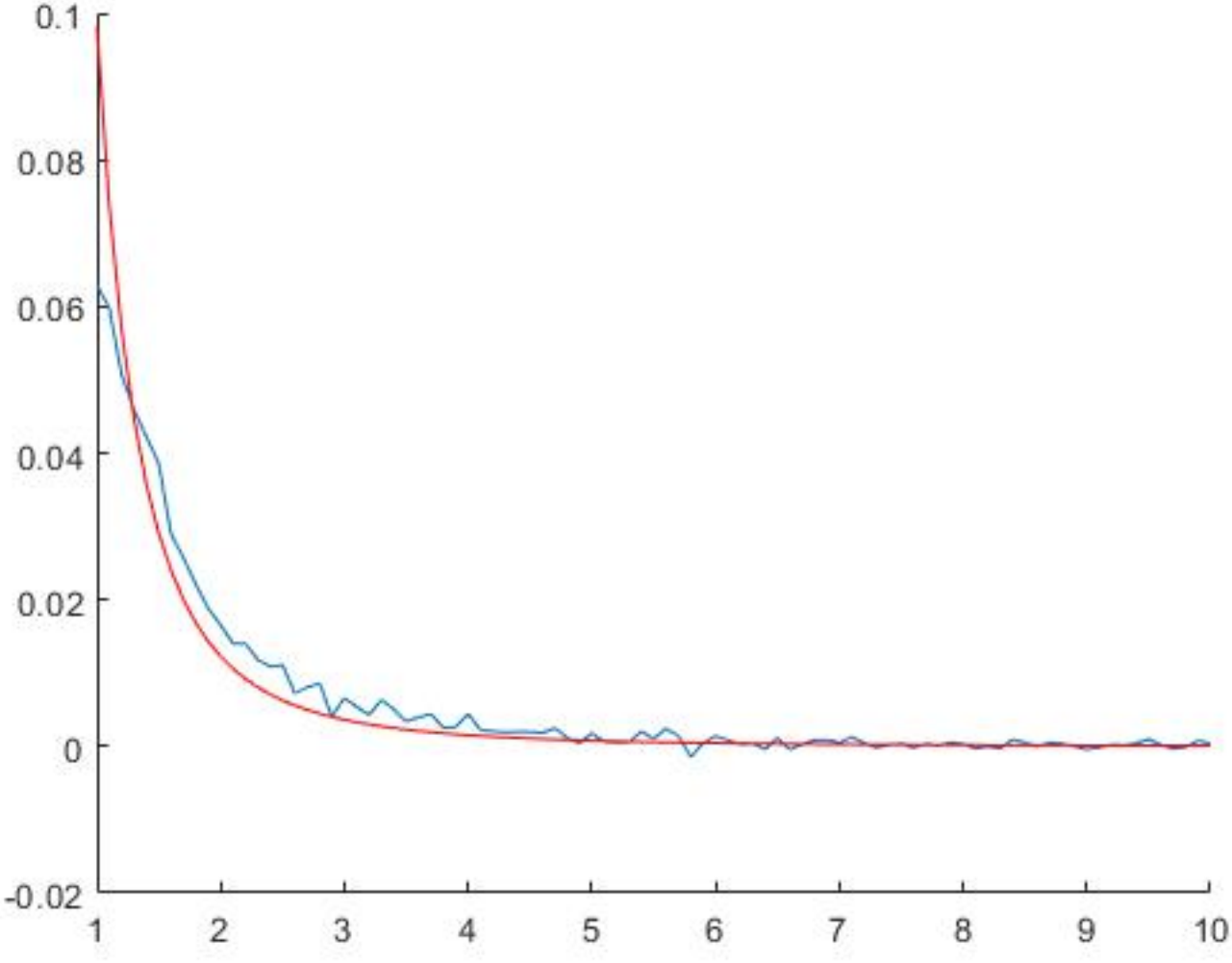

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Distribution

```
%************************************************************************
% This program computes the electric potential of a uniformly charged
% ellipsoid of radius R as a function of r, in a fixed direction (theta,phi)
% and compares it with a dipole with same moment.
%************************************************************************
theta=pi/6;phi=pi/4;
h=0.1;
r1=1;r2=10;
M=(r2-r1)/h+1;
r=r1:h:r2;

V=zeros(1,M);

for m=1:M

X=r(m)*sin(theta)*cos(phi);Y=r(m)*sin(theta)*sin(phi);Z=r(m)*cos(theta);
f=@(x,y,z)1./sqrt((x-X).^2+(y-Y).^2+(z-Z).^2);
R=1.0;

N=100000;
k=0;

for n=1:N
    x=-1+2*rand;
    y=-1+2*rand;
    z=sqrt(2)*(-1+2*rand);
    s=sqrt(x^2+y^2+z^2/2);
    if s<=1
        if z > 0
            V(m) = V(m) + f(x,y,z);
        else
            V(m) = V(m) - f(x,y,z);
        end
        k=k+1;
    end
end

V(m) = 4*sqrt(2)*pi*R^3*V(m)/(3*k);
end
Vdipole = pi*R^4*cos(theta)./(r.^2);

plot(r,V)
hold on
grid on
plot(r,Vdipole,'r')
xlabel("Radius");
ylabel("Potential");
hold off
```


3
2.5
2
1.5
1
0.5
0
Potential
1
2
3
4
5
6
7
8
9
10
Radius

```
 1
 2 %************************************************************************
 3 % This program computes the electric potential of a charged disc, one half
 4 % of which carries a positive uniform charge density and the other a
 5 % negative density. Comparison is made with the dipole contribution as the
 6 % monopole moment is zero.
 7 %************************************************************************
 8
 9 theta=pi/6;phi=pi/2;
10 h=0.1;
11 r1=1;r2=10;
12 M=(r2-r1)/h+1;
13 r=r1:h:r2;
14
15 V=zeros(1,M);
16
17 for m=1:M
18
19 X=r(m)*sin(theta)*cos(phi);Y=r(m)*sin(theta)*sin(phi);Z=r(m)*cos(theta);
20 f=@(x,y)1./sqrt((x-X).^2+(y-Y).^2+Z.^2);
21 R=1.0;
22
23 N=1000000;
24 k=0;
25
26 for n=1:N
27     x=-1+2*rand;
28     y=-1+2*rand;
29     s=sqrt(x^2+y^2);
30     if s<=1
31     if y > 0
32         if x>0
33             V(m) = V(m) + f(x,y);
34         else
35             V(m) = V(m) - f(x,y);
36         end
37     else
38         if x < 0
39             V(m) = V(m) + f(x,y);
40         else
41             V(m) = V(m) - f(x,y);
42         end
43     end
44     k=k+1;
45     end
46 end
47
48 V(m) = pi*R^2*V(m)/k;
49 end
50 %Vquad = pi*R^4*(3*(sin(theta)-1)^2)*sin(phi)./(8.*r.^3);%Change this
51 %Vquad=0.755897./r.^3;
```

```
52 Vquad=(pi*R^4/8)*(1- 2*sin(theta)^2*sin(2*phi))*sin(2*phi)/2./r.^3;
53 hold on
54 plot(r,V)
55 plot(r,Vquad,'r')
56 xlabel("Radius");
57 ylabel("Potential");
58
```


0.1
0.08
0.06
0.04
0.02
0
-0.02
1
2
3
4
5
6
7
8
9
10