Computational Physics - Final Projects

Fredrik Gustafsson

March 2019

1 Introduction

When trying to find solutions to physics problems one sometimes end up with equations which are difficult to solve by hand, if even possible. Another problem might be that it is simply not feassible to do the calculations by hand. Such problms might arrise when studying chaotic systems or when examinig quantum mechanical problems and models such as the ising model.

2 Theory

2.1 Scattering by a central potential

When a particle with initial kinetic energy E and impact parameter b approaches a potential it will undergo a scattering event. In the scattering event the particle will pass by the force center and be deflected. The result from this is that the particle will emerge with the same energy but at an angle Θ compared to the incident direction. Using the fact that the angular momentum is conserved on can start to derive an equation for the scattering angle using polar coordinates. Conservation of angular momentum implies

$$L = mvb = mr^2 \frac{d\theta}{dt}. \tag{1}$$

If we then use conservation of energy we get

$$\frac{1}{2}m\left(\frac{dr}{dt}\right)^2 + \frac{L}{2mr^2} + V = E. \tag{2}$$

Using r instead of t as the independent variable in 1 yields

$$\frac{d\theta}{dr} = \frac{d\theta}{dt}\left(\frac{dr}{dr}\right)^{-1} = \frac{bv}{r^2}\left(\frac{dr}{dt}\right)^{-1} \tag{3}$$

and solving 2 for $\frac{dr}{dt}$

$$\frac{dr}{dt} = \pm\frac{bdr}{r^2}\left(1 - \frac{b^2}{r^2} - \frac{V}{E}\right)^{-1/2}. \tag{4}$$

Using $\theta = \pi$ when r= ∞ on the incoming trajectory branch the scattering angle can be calculated analytically as

$$\Theta = \pi - 2\int_{r_{min}}^{\infty} \frac{b}{r^2}\left(1 - \frac{b^2}{r^2} - \frac{V}{E}\right)^{-1/2} \tag{5}$$

Where r_{min} is the distance of closest approach. Using a final transformation by assuming that there is a value r_{max} for which we can neglect V yields a final approximation which can be solved numerically

$$\Theta = 2b\left[\int_b^{r_{max}} \frac{dr}{r^2}\left(1 - \frac{b^2}{r^2}\right)^{-1/2} - \int_{r_{min}}^{r_{max}} \frac{dr}{r^2}\left(1 - \frac{b^2}{r^2} - \frac{V}{E}\right)^{-1/2}\right] \tag{6}$$

2.2 Ising model

The ising model represents a set spin degrees of freedom interacting with each other and a magnetic field. This might represent atoms in a solid. In a two dimensional case each spin will be located on a N by N square lattice and can be labeled as $S_{i,j}$ or s_α where α is a generic site label. The Hamiltonian of the system can be written as

$$H = -J\sum_{\alpha\beta} S_\alpha S_\beta - B\sum_\alpha S_\alpha \tag{7}$$

where J is the magnetic interaction between neighboring spins and B the magnitude of the magnetic field. Since we want to model a continous system on a smaller finite grid lattice periodic boundary conditions are used.

IN this particular case what is interesting is the thermodynamic properties and how they change with temperature. The magnetization can be calculated as

$$M = \sum_S w(S) \left(\sum_\alpha S_\alpha \right) \tag{8}$$

the susceptibility

$$x = \frac{\partial M}{\partial B} = \sum_S w(S) \left(\sum_\alpha S_\alpha \right)^2 - M^2. \tag{9}$$

To calculate the specific heat one first needs to calculate the energy as

$$E = \sum_S w(S) H(S) \tag{10}$$

and finally the specific heat can be calculated as

$$C_B = \sum_S w(S) H^2(S) - E^2. \tag{11}$$

In equation 8-11 above w(S) is calculated as

$$w(S) = \frac{e^{-H(S)}}{Z} \tag{12}$$

where Z is

$$Z = \sum_S e^{-H(S)} \tag{13}$$

3 Implementation/solution

3.1 Scattering

At first the analytic solution was calculated when E< $U_0 = V$. Starting frome equ.5 however since r_{max}<r V = 0 which give us

$$\Theta = \pi - 2 \int_{r_{max}}^{\infty} \frac{b dr}{r^2} \left(1 - \frac{b^2}{r^2} \right)^{-1/2}. \tag{14}$$

Starting by substituting $u = \frac{b}{r} \rightarrow dr = -\frac{r^2}{b} du$ which gives us

$$- \int_{\frac{b}{r_m ax}}^{0} \frac{1}{\sqrt{1-u^2}} = -sin^{-1}(0) + sin^{-1}(\frac{b}{r_{max}}) = sin^{-1}(\frac{b}{r_{max}}) \tag{15}$$

which gives us

$$\Theta = \pi - 2sin^{-1} \left(\frac{b}{r_{max}} \right). \tag{16}$$

The second case is when $U_0 = V < E$ in which case we want to solve the integral

$$\int_{r_{min}}^{r_{max}} \frac{dr}{r^2 \sqrt{1 - \frac{b^2}{r^2} - \frac{V}{E}}}. \tag{17}$$

Starting by substituting $u = \frac{b}{r}$ and $v^2 = 1 - \frac{V}{E}$. Solving this and substituting back yields the solution

$$sin^{-1}\left(\frac{b}{r_{max}\sqrt{i - \frac{V}{E}}}\right) \tag{18}$$

To solve the problem numerically boole integration was used and the program was executed with different values on b, from 0 to r_{max}.

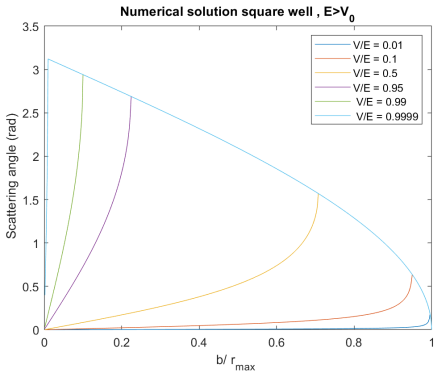
3.2 Ising model

The ising model was implement such as each position either having a 1 or -1 spin after which monte carlo simulation was performed first with metropolis and then with heat bath algorithm on the system. While sweeping the lattice a random position was chosen and the change in energy was calculated. In the metropolis case the spin changed sign if the change in energy was negative or if a random generated number was smaller than $e^{-dE/T}$. In the heat bath algorithm the spin changed to 1 if the random number was smaller than p_i or -1 else, analog to 1 with probability p_i and -1 with probability $1-p_1$ where $p_i = \frac{e^{dE/T}}{1+e^{dE/T}} = \frac{1}{1+e^{-dE/T}}$.In both cases meassurements of the energy and magnetization didn't start until $\frac{3}{4}$ of the time steps had been taken in order to have the system reach equilibrium. After which the thermodynamic properties was calculated. Finally J was changed from 1 to -1 for one run, B set to non zero for another run and finally a change to allow anti ferromagnetic interaction with the second closest neighbors and ferromagnetic with its closest.

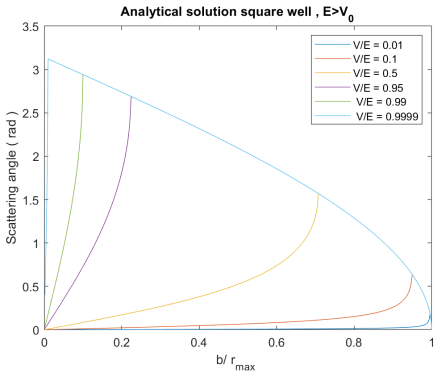
4 Result

4.1 Scattering

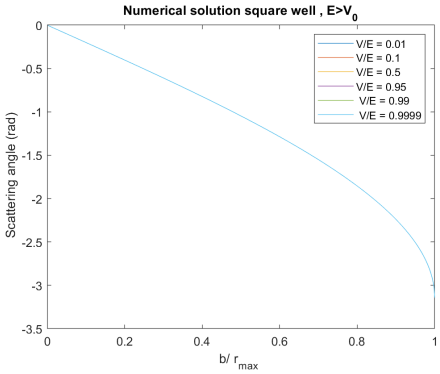
After implementing the numerical and analytical solution in matlab the program was ran twice, for $U_0 < E$ and $U_0 > E$. For the case $E > U_0$ the result was calculated with U_0 as 0.01E, 0.10E, 0.5E, 0.95E, 0.99E and 0.9999E. In the other cacse $E < U_0$ the cslculations where done with U_0 as 1.1E, 5E, 10E, 50E, 100E and 150E. In both cases the input parameter was varied with 500 points lineary spaced between 0 to 1. The resulting plots are shown below in figure 1.



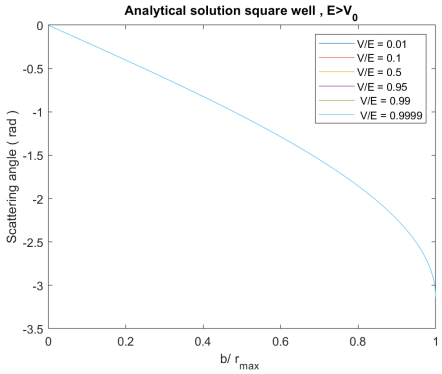
(a) Numeric solution for $E > U_0$.



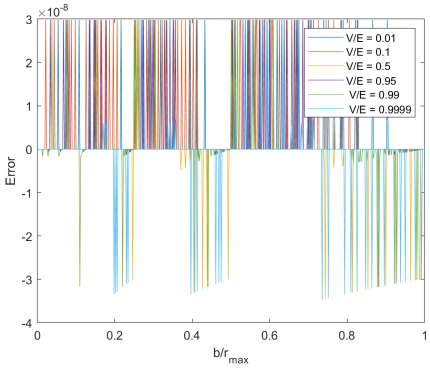
(b) Analytic solution for $E > U_0$.



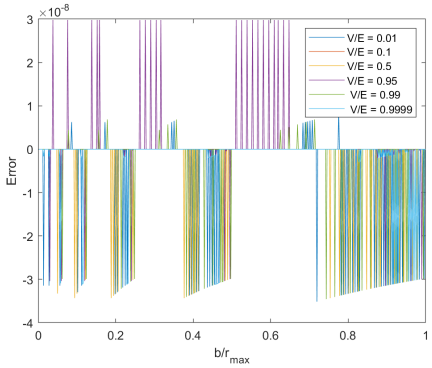
(c) Numeric solution for $E < U_0$.



(d) Analytic solution for $E > U_0$.



(e) Error when $E > U_0$.



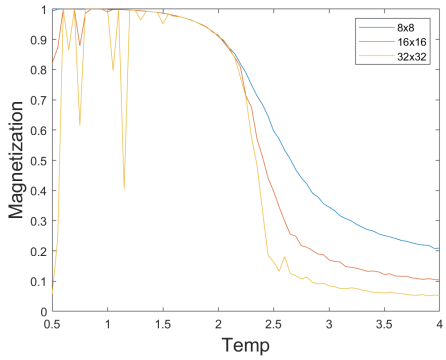
(f) Error when $E < U_0$.

Figure 1: Results for scattering simulation.

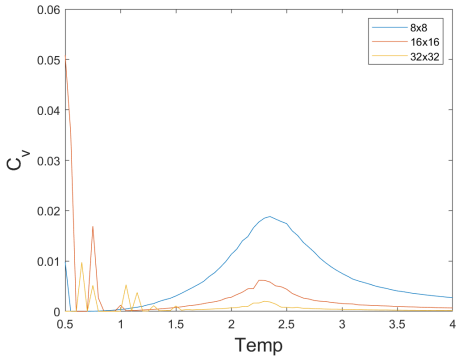
4.2 Ising model

When simulating the ising model $2^8 * N^2$, where N is the dimension of the model, steps were taken to equilibriate the model. After which further 2000000 steps were done in which the magnetization, and energy was calculated. From these measurements the heat capacity, 4th order cumulant and magnetic fluctuation could be calculated. This was repeated for the three different grid sizes 8, 16 and 32 and for temperatures between 1.5 and 4 with increments of 0.05, all temperatures are given in kelvin.

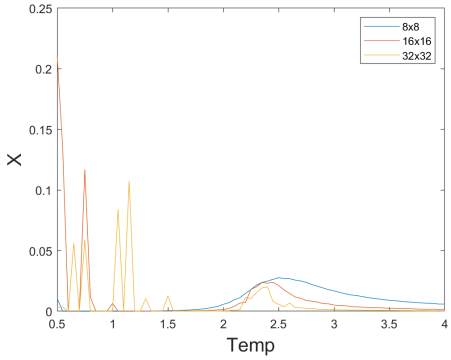
Figure 2 shows the result when doing the simulation from temperatures between 0 and 4. In this case the coupling was set to 1 and a 0 external magnetic field.



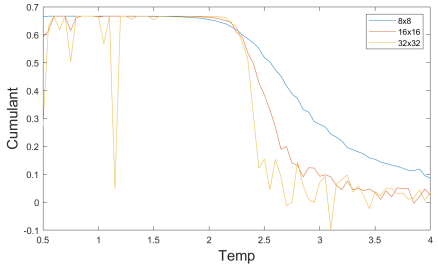
(a) Absolute value of magnetization for temperatures between 0 and 4.



(b) Heat capacity for temperatures between 0 and 4.



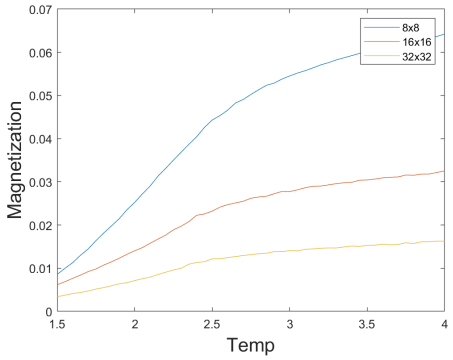
(c) Magnetic fluctuation for temperatures between 0 and 4



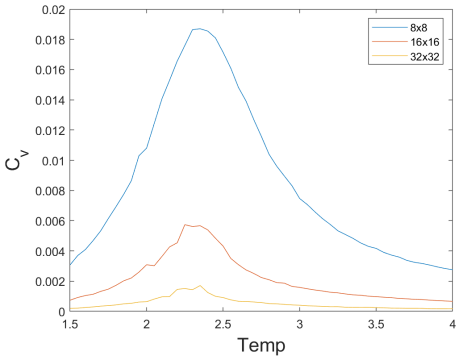
(d) Cumulant for temperatures between 0 and 4.

Figure 2: Ising simulation with metropolis algorithm J = 1, B = 0.

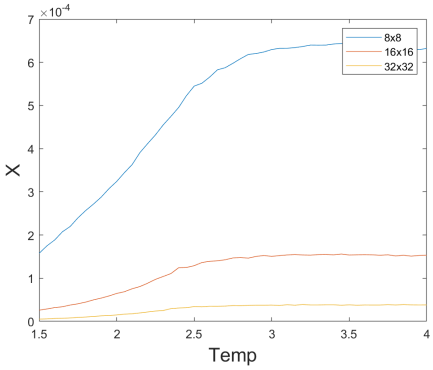
After investigating the base case, J=1 and B=0, the case in which J was negative, the coupling being anti-ferromagnetic, was tested. The results of which is shown below in figure 3.



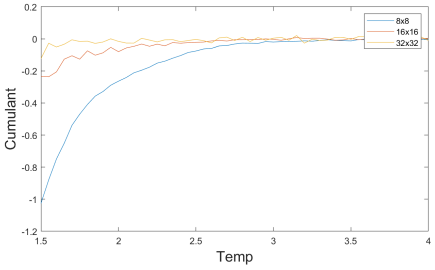
(a) Absolute value of magnetization for temperatures between 1.5 and 4.



(b) Heat capacity for temperatures between 1.5 and 4.



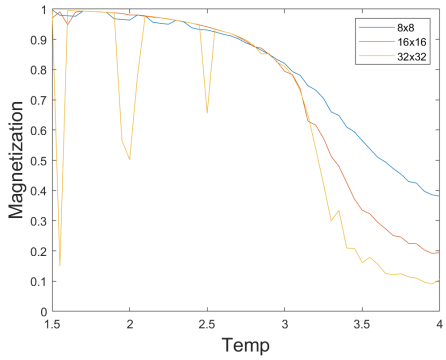
(c) Magnetic fluctuation for temperatures between 1.5 and 4



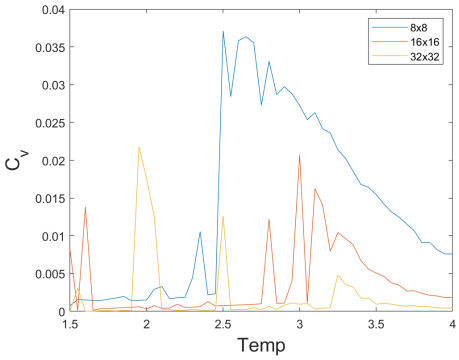
(d) Cumulant for temperatures between 1.5 and 4.

Figure 3: Ising simulation with metropolis algorithm J = -1, B = 0.

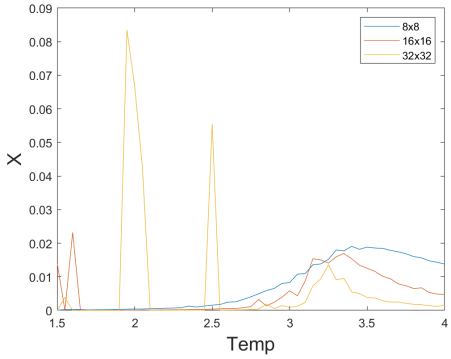
Next the case in which there was an external magnetic field with magnitude 1 or -1 was tested the results of which is presented in figure 4 and figure 5.



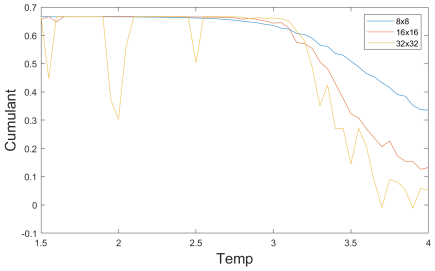
(a) Absolute value of magnetization for temperatures between 1.5 and 4.



(b) Heat capacity for temperatures between 1.5 and 4.

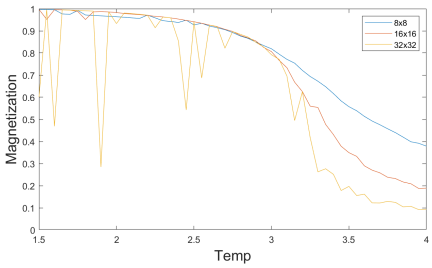


(c) Magnetic fluctuation for temperatures between 1.5 and 4

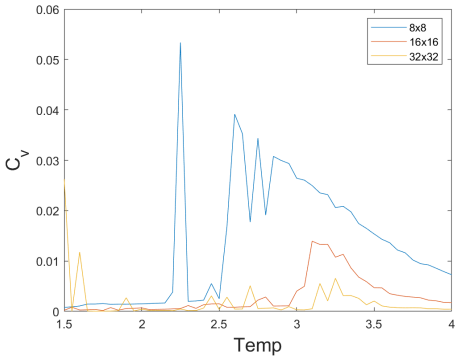


(d) Cumulant for temperatures between 1.5 and 4.

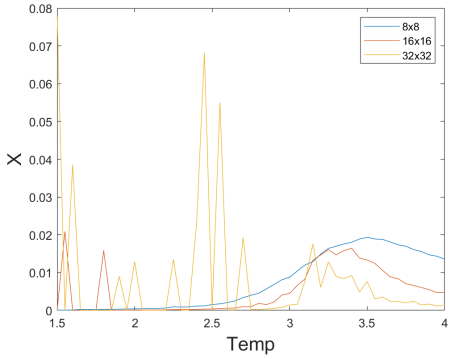
Figure 4: Ising simulation with metropolis algorithm J = 1, B = 1.



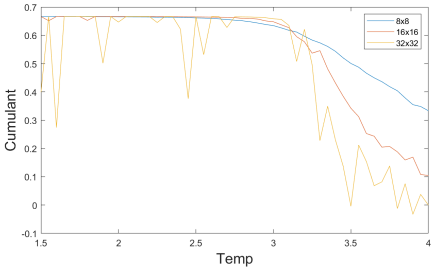
(a) Absolute value of magnetization for temperatures between 1.5 and 4.



(b) Heat capacity for temperatures between 1.5 and 4.



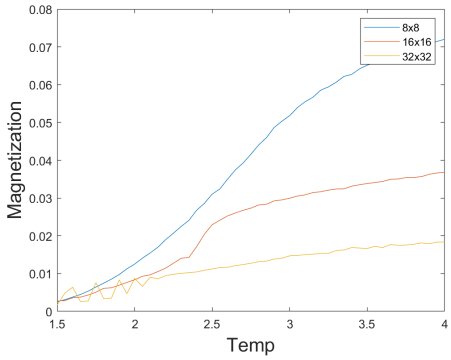
(c) Magnetic fluctuation for temperatures between 1.5 and 4



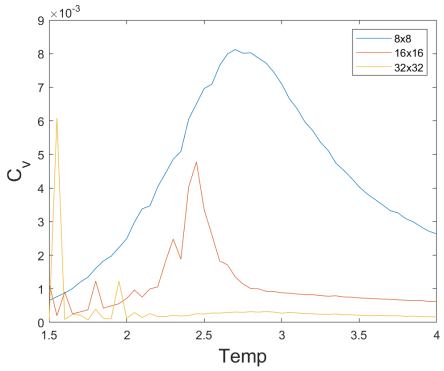
(d) Cumulant for temperatures between 1.5 and 4.

Figure 5: Ising simulation with metropolis algorithm J = 1, B = -1.

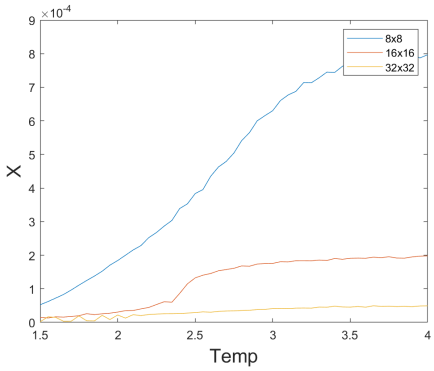
Finally the last thing investigated with the metropolis algorithm was when the spins were interacting with their closest neighbors ferromagnetically and anti-ferromagnetically with their second closest. During this the coupling factor was 1 and the external magneticfield was set to 0. The results are shown below in figure 6



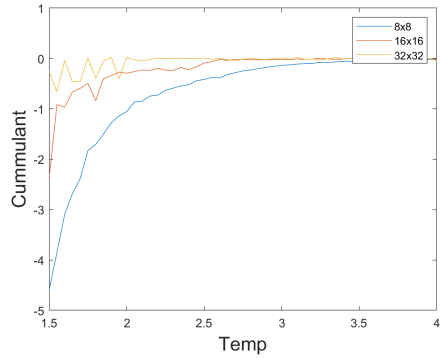
(a) Absolute value of magnetization for temperatures between 1.5 and 4.



(b) Heat capacity for temperatures between 1.5 and 4.



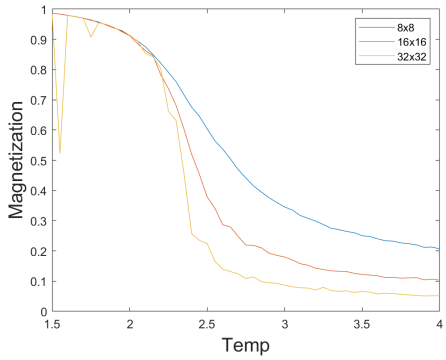
(c) Magnetic fluctuation for temperatures between 1.5 and 4



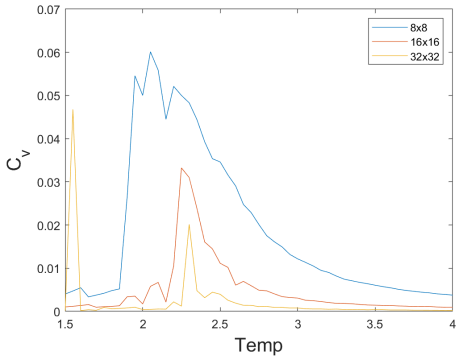
(d) Cumulant for temperatures between 1.5 and 4.

Figure 6: Ising simulation with metropolis algorithm spins interacting ferromagnetically with closest neighbor and anti-ferromagnetically with second closest neighbor, J = 1, B = 0.

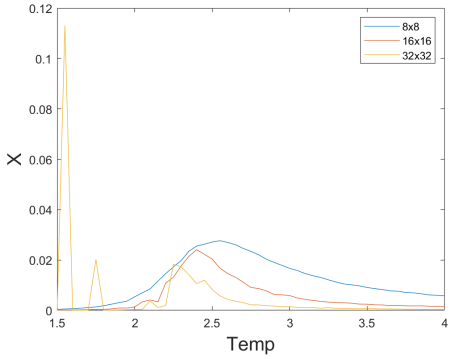
The last thing investigated with the ising model was when the algorithm was changed from metropolis to heat bath. During this test J was set to 1 and B to 0. Figure 7 below show the results from this experiment.



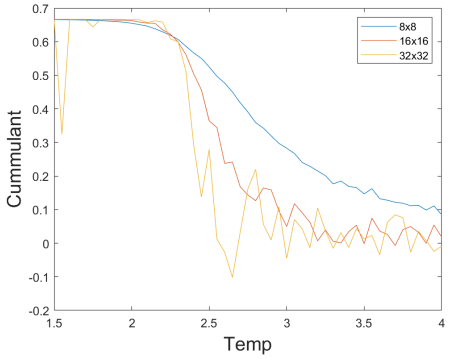
(a) Absolute value of magnetization for temperatures between 1.5 and 4.



(b) Heat capacity for temperatures between 1.5 and 4.



(c) Magnetic fluctuation for temperatures between 1.5 and 4



(d) Cumulant for temperatures between 1.5 and 4.

Figure 7: Ising simulation with Heat bath algorithm, J = 1, B = 0.

5 Discussion

5.1 Scattering

As we can see from figure 1 the numeric results agrees with the analytic in both cases. This is supported both by comparing the graphs and also from the error having a maximum value of $3 * 10^{-8}$. Further we can see that when $E < U_0$ the results are the same no mater the value of U_0/E. One can also see that the scattering angle decreases from 0 down to -3 with increasing value of the impact parameter. What is more interesting is the behaviour when $E > U_0$. For lower values of V/E the scattering angle remains 0 as the impact parameter increases for a longer time. The quotient also appears to affect how quick the scattering angle increases and the maximum value of it. As the quotient approaches 1 the scattering angle becomes around 3.1 for the first non zero value of the impact parameter. Further one can see that no matter the quotient the decrease of the scattering angle from its maximum value follows the same curve.

5.2 Ising model

From figure 2d we can see that the cumulant for the three different lattice sizes doesn't quite cross each others at one point instead the two crossings are at around 2.23 and 2.265. The critical temperature for a square lattice would be $\frac{1}{ln(1+\sqrt{(2)})} = 2.269$ this is fairly close to what is found in the experiment where

the difference could be due to the resolution used for the temperature. Further we can see from figure 2a that after around 2.25 the value of the magnetization decreases rapidly towards 0. This shows that on average the magnetization of the material is approaching 0 as the temperature is increasing. Figure 2b-2c both shows maximas around the critical temperature point, disregarding the early fluctuating measurements.

Comparing figure2 and 3 one can see that the change to antiferromagnetic coupling results in a magnetization starting from zero and slowly increasing, still lower than for J=1. The fluctuations were also lower, on a scale of 10^{-4}. The cumulant for the two larger lattice sizes stayed around 0 while the cumuland for the smalest started at -1 and increased towards 0. However the heat capacity does not appear to be affected by teh change in the coupling still reaching a maximum at around 2.26.

When applying an external magnetic field, positive or negative, as shown in figure 4 and figure **??** respectively. We can notice both that the results are not as smooth as before, however that could be due to the stochastic properties of the model, and more importantly that the critical temperature has been increased from around 2.25-2.3 to around 3-3.3.

When changing the interacting from just the closest to the closest and second closest neighbors the results becomes close to what we got from changing J from 1 to -1. The main difference is that now we also notice a difference in the heat capacity. The largest lattice size does not appear to have a maximum and the maximum for the two smaller sizes does not agree with each others being centered around different points. This could be due to $\sum_{\alpha\beta} S_\alpha S_\beta$ now having a larger variety of values from 8 to -8 instead of 4 to -4.

Finally when changing the algorithm from metropolis to heat bath one got smother results as show in figure 7. When using the heat bath algorithm the crossing of the different cumulants crosses each others at around 2.26 as expected. The main difference between the results from the heat bath and metropolis algorithm seems to be the heat capacity and thereby the energy. Comparing figure 7b and figure 2b one can see that the heat capacity from the heat bath algorithm reaches a larger value but with sharper peaks.

6 Conclusion

From the tests we can see that for the scattering angle if $E > U_0$ the quotient decides the maximum scattering angle. The scattering angle will decrease from its maximum as the impact parameter increases. In the case that $E < U_0$ the quotient does not appear to affect the scattering angle instead it will decrease from 0 towards -3 with increasing impact parameter.

For the ising model the coupling factor J appears to affect the magnetic properties of the model, magnetization, 4th order cumulant and fluctuation while not affecting the heat capacity. With the addition of an external magnetic field the critical temperature appears to be increased from 2.26 to 3-3.3. The heat bath algorithm appears to work as well as the metropolis algorithm. Since the crossing of the cumulants shows the phase transition temperature it could be estimated to be around 2.265-2.3 while it according to theory should be around 2.269. This then shows that the model gives a fairly good estimate of the phae transition temperature. Finally one can observe the finite size effects examples of which is how the magnetization of the smaller lattice is higher than for the larger ones and even they do not reach zero. This is due to us trying to model a infinite system with a finite number of points leading to it being in a semi ordered state and never truly unordered.

7 Appendix

7.1 Scattering

7.1.1 Project1.m

```
%Project 1
%Numerical and analytical solution

close all;
clear all;

%% Define varialbes and parameters
N = 500 ; %could take less integration points, but 500 gives a really nice resolution
rmax = 1 ;
b = linspace( 0 , rmax , N);
E = 1 ; %set as constant E = 1
theta = zeros(length(b),1);
analyticsol = zeros(length(b),1);

%take one of the V vectors below, to observe when V<0 |E|>|V|, V<0
%|V|>|E|, V>0 E > V, V>0 E < V

%V = -[ 0.01 0.10 0.5 0.95 0.99 0.9999 ]; % V<0 |E|>|V|
%V = -[1.1 5 10 50 100]; % V<0 |V|>|E|
V = [ 0.01 0.10 0.5 0.95 0.99 0.9999 ]; % V>0 E > V
%V = [1.1 5 10 50 100 150]; % V>0 E < V

%% Numerical solution

for i=1:length(V)

for j =1:length(b)

rmin = b(j) /sqrt(1 - (V( i )/E)) ; %rmin, comes from analytical solution

integral1 = @( x ) 1/((x^2 + b(j))*(sqrt(x^2 + 2*b(j)))) ;
integral2 = @( x ) 1/((x^2 + b(j))*(sqrt(x^2 + 2*b(j)))) ;

rtop1 = sqrt(rmax-b(j));
rlow1 = 0;
rtop2 = sqrt((sqrt(1-(V(i)/E))*rmax) - b(j));
rlow2 = sqrt((sqrt(1-(V(i)/E))*rmin) - b(j));

integral1 = boole( integral1 , rlow1 , rtop1 , N) ;
integral2 = boole( integral2 ,rlow2, rtop2 , N) ;

theta(j) = 4*b(j)*real( integral1 - integral2 ) ;

end
plotTheta(i,:) = theta;

figure(1);
plot(b/rmax , theta );
hold on ;

end
title (' Numerical solution square well , E>V_0 ' ) ;
xlabel ( 'b/ r_{max} ' ) ;
ylabel(' Scattering angle (rad)' );
```

```
legend ( 'V/E = 0.01 ' , 'V/E = 0.1 ' , 'V/E = 0.5 ' , 'V/E = 0.95 ',' V/E = 0.99 ' , ' ...
    V/E = 0.9999 ') ;

%% Analytical solution

for i=1:length(V)

for j =1:length(b)

rmin = b(j) / sqrt (1 - (V(i)/E));

figure(2);
x1 = b(j) / rmax ;
x2 = b(j) / rmin ;
analyticsol(j) = - 2*real( ( ( asin( x1 )- asin( 1 ) ) - asin(x1/ sqrt(1 - (V(i)/E)) ) + ...
    asin( x2/ sqrt(1 - (V(i)/E)) ) ) ) ;
end
plotAnalytic(i,:) = analyticsol;
plot (b/rmax , analyticsol ) ;
hold on
end
title (' Analytical solution square well , E>V_0 ' ) ;
ylabel( ' Scattering angle ( rad ) ' ) ;
xlabel ( 'b/ r_{max} ' ) ;
legend ( 'V/E = 0.01 ' , 'V/E = 0.1 ' , 'V/E = 0.5 ' , 'V/E = 0.95 ',' V/E = 0.99 ' , ' ...
    V/E = 0.9999 ') ;

figure(3)
plot(b/rmax, plotAnalytic-plotTheta)

ylabel('Error')
xlabel('b/r_{max}')
legend ( 'V/E = 0.01 ' , 'V/E = 0.1 ' , 'V/E = 0.5 ' , 'V/E = 0.95 ',' V/E = 0.99 ' , ' ...
    V/E = 0.9999 ') ;
```

7.2 Ising

7.2.1 mainv2.m

```
clear all;
N=[8 16 32];    %Lattice size
L=4000000;   %Number of simulations/temperature after equilibrium
J=1;         %Coupling
B=1;         %External magnetic field
temp = [1.5:0.05:4];    %Temperatures


%allocating required memory
grideqm = cell(1,length(temp)); %Store equilibriate states
plotM = zeros(length(N),length(temp)); %Matrix to plot M
plotX = zeros(length(N),length(temp)); %Matrix to plot X
plotCv = zeros(length(N),length(temp)); %Matrix to plot Cc
plotCummulant = zeros(length(N),length(temp)); %Matrix to plot cummulant


for a = 1:length(N) %Do for each lattice size
    for i=1:length(temp) %Equilibriate system for each temperature
    [grid] = equMetropolis2nd(N(a), temp(i), J, B);

```

```
%store equilibrated arrangements with
    grideqm{i}= grid;
end
disp('equilibration finished!');

%preallocating for speed
Mp=zeros(1,length(temp));
x=zeros(1,length(temp));
C=zeros(1,length(temp));
Ep=zeros(1,length(temp));


%calculating average properties of the model for each temperature step

for j = 1:length(temp)
    [Ms,Es,xs,Cs,Cums] = propCalcMetro2nd(N(a),temp(j),J,L,grideqm{j},B);
    disp(j);
    Mp(1,j)=Ms;
    Ep(1,j)=Es;
    x(1,j) = xs;
    C(1,j) = Cs;
    Cum(1,j) = Cums;
end
plotM(a,:) = Mp;
plotX(a,:) = x;
plotCv(a,:) = C;
plotCummulant(a,:) = Cum;
end

figure()
plot(temp, plotM)
legend('8x8','16x16', '32x32')
xlabel('Temp','FontSize', 16)
ylabel('Magnetization', 'FontSize', 16)

figure()
plot(temp, plotX)
legend('8x8','16x16', '32x32')
xlabel('Temp','FontSize', 16)
ylabel('X', 'FontSize', 16)

figure()
plot(temp, plotCv)
legend('8x8','16x16', '32x32')
xlabel('Temp','FontSize', 16)
ylabel('C_v', 'FontSize', 16)

figure
plot(temp, plotCummulant)
legend('8x8','16x16', '32x32')
xlabel('Temp','FontSize', 16)
ylabel('Cummulant', 'FontSize', 16)
```

7.2.2 equMetropolis.m

```

function [grid] = equMetropolis(N, T, J, B)

grid = sign(0.5-rand(N,N)); %Initialize grid random 1 or -1
```

```
numIters = 2^9*N*N;       %Number of steps for equilibration

for iter = 1 : numIters
    % Pick a random spin
    x = randi(N,1);
    y = randi(N,1);

    % Find its nearest neighbors with periodic boundary
    if y~=1; left=grid(x,y-1);else left=grid(x,N);end
    if y~=N; right=grid(x,y+1);else right=grid(x,1);end
    if x~=1; up=grid(x-1,y);else up=grid(N,y);end
    if x~=N; down=grid(x+1,y);else down=grid(1,y);end

    neighbors = up+down+left+right;

    % Calculate energy change if this spin is flipped
    dE = 2 *(J*grid(x, y)*sum(neighbors)-(grid.*B));

    % Spin flip condition
    if dE <= 0
        grid(x, y) = - grid(x, y);
    else
        prob = exp(-dE / T); %save calculations to here due to comp. expensive.
        if rand(1) <= prob
        grid(x, y) = - grid(x, y);
        end
    end

    end
end
```

7.2.3 equMetropolis2nd.m

```

function [grid] = equMetropolis2nd(N, T, J, B)

grid = sign(0.5-rand(N,N)); %Initialize grid random 1 or -1
numIters = 2^9*N*N;       %Number of steps for equilibration

for iter = 1 : numIters
    % Pick a random spin
    x = randi(N,1);
    y = randi(N,1);

    % Find its nearest neighbors with periodic boundary
    if y == 1;
        left = grid(x,N)-grid(x,N-1);
    elseif y == 2;
            left = grid(x,1)-grid(x,N);
    else
        left = grid(x,y-1)-grid(x,y-2);
    end
    if y == N;
        right = grid(x,1)-grid(x,2);
    elseif y == N-1;
            right = grid(x,N)-grid(x,1);
    else
        right = grid(x,y+1)-grid(x,y+2);
    end
```

```
27       if x == 1;
28          up = grid(N,y)-grid(N-1,y);
29       elseif x == 2;
30               up = grid(1,y)-grid(N,y);
31       else
32          up = grid(x-1,y)-grid(x-2,y);
33       end
34       if x == N;
35          down = grid(1,y)-grid(2,y);
36       elseif x == N-1;
37               down = grid(N,y)-grid(1,y);
38       else
39          down = grid(x+1,y)-grid(x+2,y);
40       end
41
42       neighbors = up+down+left+right;
43
44       % Calculate energy change if this spin is flipped
45       dE = 2 *(J*grid(x, y)*sum(neighbors)-(grid.*B));
46
47       % Spin flip condition
48       if dE <= 0
49          grid(x, y) = - grid(x, y);
50       else
51          prob = exp(-dE / T); %save calculations to here due to comp. expensive.
52          if rand(1) <= prob
53          grid(x, y) = - grid(x, y);
54          end
55       end
56
57       end
58   end
```

7.2.4 equHeatBath.m

```
1
2   function [grid] = equHeatBath(N, T, J, B)
3
4   grid = sign(0.5-rand(N,N)); %Initialize grid random 1 or -1
5   numIters = 2^9*N*N;         %Number of steps for equilibration
6
7   for iter = 1 : numIters
8       % Pick a random spin
9       x = randi(N,1);
10      y = randi(N,1);
11
12      % Find its nearest neighbors with periodic boundary
13      if y~=1; left=grid(x,y-1);else left=grid(x,N);end
14      if y~=N; right=grid(x,y+1);else right=grid(x,1);end
15      if x~=1; up=grid(x-1,y);else up=grid(N,y);end
16      if x~=N; down=grid(x+1,y);else down=grid(1,y);end
17
18      neighbors = up+down+left+right;
19
20      % Calculate energy change if this spin is flipped
21      %dE = 2 *(J*grid(x, y)*sum(neighbors)-(grid.*B));
22
23      % Spin flip condition
24      p = 1/(1+exp(-2*J/T*neighbors));
```

```
25
26      % Spin flip condition
27     if rand() <p
28         grid(x,y) = 1;
29     else
30         grid(x,y) = -1;
31     end
32
33      end
34 end
```

7.2.5 propCalcMetro.m

```
1  function [Ms,Es,xs,Cs, Cum] = propCalcMetro(N,T,J,L,grid,B)
2  Mmean=zeros(1,L);
3  Emean=zeros(1,L);
4  for iter = 1 : L
5      % Pick a random spin
6
7  x = randi(N,1);
8  y = randi(N,1);
9
10     % Find its nearest neighbors
11     if y~=1; left=grid(x,y-1);else left=grid(x,N);end
12     if y~=N; right=grid(x,y+1);else right=grid(x,1);end
13     if x~=1; up=grid(x-1,y);else up=grid(N,y);end
14     if x~=N; down=grid(x+1,y);else down=grid(1,y);end
15
16     neighbors = up+down+left+right;
17
18     % Calculate energy change if this spin is flipped
19     dE = 2 * (J * grid(x, y) * neighbors+(grid.*B));
20
21     % Spin flip condition
22     if dE <= 0
23         grid(x, y) = - grid(x, y);
24     else
25         prob = exp(-dE / T);          % Boltzmann probability of flipping
26         if rand(1) <= prob
27         grid(x, y) = - grid(x, y);
28         end
29     end
30
31 %calculating magnetisation per spin
32 Mmean(1,iter) = abs(mean(grid(:))/numel(N));
33
34 %calculating energy per spin
35 sumOfNeighbors = ...
36       circshift(grid, [ 0  1]) ...
37     + circshift(grid, [ 0 -1]) ...
38     + circshift(grid, [ 1  0]) ...
39     + circshift(grid, [-1  0]);
40 Em = - J * grid .* sumOfNeighbors-B*grid;
41 E  = 0.5 * sum(Em(:));
42 Emean(1,iter) = E / numel(grid);
43
44
45
46 end
```

```
47
48 Ms=mean(Mmean);% Ms is a scalar with average of all the elements in array Mag_avg;
49 Es=mean(Emean); % Es is a scalar with average of all the elements in array Energy_avg;
50 xs=(mean(Mmean.^2)-mean(Mmean)^2)/T;
51 Cs=(mean(Emean.^2)-mean(Emean)^2)/T^2;
52 Cum = 1-((mean(Mmean.^4)/(3*mean(Mmean.^2)^2)));
53 end
```

7.2.6 propCalcMetro2nd.m

```
1  function [Ms,Es,xs,Cs, Cum] = propCalcMetro2nd(N,T,J,L,grid,B)
2  Mmean=zeros(1,L);
3  Emean=zeros(1,L);
4  for iter = 1 : L
5      % Pick a random spin
6
7  x = randi(N,1);
8  y = randi(N,1);
9
10     % Find its nearest neighbors with periodic boundary
11     if y == 1;
12         left = grid(x,N)-grid(x,N-1);
13     elseif y == 2;
14             left = grid(x,1)-grid(x,N);
15     else
16         left = grid(x,y-1)-grid(x,y-2);
17     end
18     if y == N;
19         right = grid(x,1)-grid(x,2);
20     elseif y == N-1;
21             right = grid(x,N)-grid(x,1);
22     else
23         right = grid(x,y+1)-grid(x,y+2);
24     end
25      if x == 1;
26         up = grid(N,y)-grid(N-1,y);
27     elseif x == 2;
28             up = grid(1,y)-grid(N,y);
29     else
30         up = grid(x-1,y)-grid(x-2,y);
31      end
32     if x == N;
33         down = grid(1,y)-grid(2,y);
34     elseif x == N-1;
35             down = grid(N,y)-grid(1,y);
36     else
37         down = grid(x+1,y)-grid(x+2,y);
38     end
39     neighbors = up+down+left+right;
40
41     % Calculate energy change if this spin is flipped
42     dE = 2 * (J * grid(x, y) * neighbors+(grid.*B));
43
44     % Spin flip condition
45     if dE <= 0
46         grid(x, y) = - grid(x, y);
47     else
48         prob = exp(-dE / T);          % Boltzmann probability of flipping
49         if rand(1) <= prob
```

```
50          grid(x, y) = - grid(x, y);
51          end
52      end
53
54  %calculating magnetisation per spin
55  Mmean(1,iter) = abs(mean(grid(:))/numel(N));
56
57  %calculating energy per spin
58  sumOfNeighbors = ...
59        circshift(grid, [ 0  1]) ...
60      + circshift(grid, [ 0 -1]) ...
61      + circshift(grid, [ 1  0]) ...
62      + circshift(grid, [-1  0]) ...
63      - circshift(grid, [0  2]) ...
64      - circshift(grid, [0  -2]) ...
65      - circshift(grid, [2  0]) ...
66      - circshift(grid, [-2  0]) ;
67  Em = - J * grid .* sumOfNeighbors-B*grid;
68  E  = 0.5 * sum(Em(:));
69  Emean(1,iter) = E / numel(grid);
70
71
72
73  end
74
75  Ms=mean(Mmean);% Ms is a scalar with average of all the elements in array Mag_avg;
76  Es=mean(Emean); % Es is a scalar with average of all the elements in array Energy_avg;
77  xs=(mean(Mmean.^2)-mean(Mmean)^2)/T;
78  Cs=(mean(Emean.^2)-mean(Emean)^2)/T^2;
79  Cum = 1-((mean(Mmean.^4)/(3*mean(Mmean.^2)^2)));
80  end
```

7.2.7 propCalcHeatBath.m

```
1   function [Ms,Es,xs,Cs, Cum] = propCalcHeatBath(N,T,J,L,grid,B)
2   Mmean=zeros(1,L);
3   Emean=zeros(1,L);
4   for iter = 1 : L
5       % Pick a random spin
6
7   x = randi(N,1);
8   y = randi(N,1);
9
10      % Find its nearest neighbors
11      if y~=1; left=grid(x,y-1);else left=grid(x,N);end
12      if y~=N; right=grid(x,y+1);else right=grid(x,1);end
13      if x~=1; up=grid(x-1,y);else up=grid(N,y);end
14      if x~=N; down=grid(x+1,y);else down=grid(1,y);end
15
16      neighbors = up+down+left+right;
17
18      % Calculate energy change if this spin is flipped
19      %dE = 2 * (J * grid(x, y) * neighbors+(grid.*B));
20      p = 1/(1+exp(-2*J/T*neighbors));
21
22      % Spin flip condition
23     if rand() <p
24         grid(x,y) = 1;
25     else
```

```
        grid(x,y) = -1;
    end

%calculating magnetisation per spin
Mmean(1,iter) = abs(mean(grid(:))/numel(N));

%calculating energy per spin
sumOfNeighbors = ...
      circshift(grid, [ 0  1]) ...
    + circshift(grid, [ 0 -1]) ...
    + circshift(grid, [ 1  0]) ...
    + circshift(grid, [-1  0]);
Em = - J * grid .* sumOfNeighbors-B*grid;
E  = 0.5 * sum(Em(:));
Emean(1,iter) = E / numel(grid);



end

Ms=mean(Mmean);% Ms is a scalar with average of all the elements in array Mag_avg;
Es=mean(Emean); % Es is a scalar with average of all the elements in array Energy_avg;
xs=(mean(Mmean.^2)-mean(Mmean)^2)/T;
Cs=(mean(Emean.^2)-mean(Emean)^2)/T^2;
Cum = 1-((mean(Mmean.^4)/(3*mean(Mmean.^2)^2)));
end
```