2 Dimensional Ising Model of Ferromagnetic-Paramagnetic Phase Transition.

Jack Mayo

(Dated: 27 March 2015)

By modelling a material as a lattice of Ising particles of sizes **8x8**, **16x16** and **32x32** the Phase transition between **Ferromagnetic** and **Paramagnetic** states was observed as a function of the system Temperature. The system configuration was simulated using the **Metropolis Algorithm** and the Curie Temperature for each lattice size wa found to be 8x8 $(2.34 \pm 0.02)K$, 16x16 $(2.32 \pm 0.02)K$, 32x32 $(2.26 \pm 0.02)K$ with only the 32x32 lattice achieving the accuracy of the researched value of 2.27K [1](L. Onsager).

I. INTRODUCTION

The interaction between orbital spin and nuclear spin causes a magnetic moment from a single atom on the micro scale, on the macro scale the magnetisation of a material is dependent on the alignment of all of those individual spin directions. In a ferromagnetic material the particle spins have a net alignment causing an over all permanent magnetic field. In a paramagnetic material the spins are randomly oriented leading to no net magnetic field. However in the presence of an external magnetic field spin aligned domains form within the material causing a non-permanent net magnetisation.

Below a critical temperature, known as the Curie temperature, if a particle flips the spin to mis-align with its neighbouring particles it will not cause enough of an energy perturbation to cause neighbouring particles however as the temperature increases the perturbed particles gain more of an energy perturbation causing neighbouring particles to flip reducing the overall magnetisation. As the temperature approaches the Curie temperature the change in magnetisation increases and the phase transition between ferromagnetic and paramagnetic states occurs.

In order to model this transition a computer simulation of Ising particles can be created with an increasing temperature measuring the magnetisation of a system and calculating the Specific Heat in order to experimentally determine the Curie temperature.

II. THE ISING MODEL

A two dimensional lattice of particles can be simulated in which each vertex of the lattice is an Ising particle with either positive or negative spin. The lattice also contains an un-physical property in which the edges are connected in both the *'x'* and *'y'* dimensions so every vertex has 4 nearest neighbours even the ones at the edge of the lattice. Initially the system is entirely made up of positive spins aligned for a maximum magnetisation. A single particle is chosen at random and flipped the local energy is then measured to see whether it will change the spin of its neighbours or whether its neighbours will cause the spin to revert to its original alignment.

The Ising simulation was created using C++ with a custom class for the Ising particles and data collection. Fig 1. shows an asci representation of the system at 1.5K there are few misaligned spins representing a ferromagnetic state. Fig 2. shows the same system at 5K past the Curie temperature with clear spin aligned domains and low net magnetisation representing a paramagnetic state.

FIG. 1. 32x32 Lattice of Ising particles at T=1.5K spin direction represented by + and - characters.

```
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + - + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + - + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + - + +
+ - + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + - + + + + + + +
+ + + + + + + + + - + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + - + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + - + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ - + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ - + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
```

III. MONTE CARLO METHOD

A. The Metropolis Algorithm

The system is initialised in the spin aligned state and a particle is chosen at random, the energy change was then calculated for the Temperature *'T'* using (1).

$$E = -J \sum_{\langle i,j \rangle} S_i S_j \tag{1}$$

Where *'S'* is the spin direction at a location and *'J'* is the interaction energy between a + and - spin. A

FIG. 2. 32x32 Lattice of Ising particles at T=5.0K spin direction represented by + and - characters.

```
+ - - - + + - - - + + - - + + + - + - + + + + - + + + + + + - +
+ + - + + + + - - - + - - - + + + + - + - + + - - - - + + + + +
+ + - - + - - - - - - - - - + + + - + - - - - - - - + - - - + +
+ + - - - - - + + + - + - - - + - - - - - - - - - - - - - - + -
- + + - - - - - + - - + + - - + - - - - - + + + - - + - - - - -
+ - - - - - - - - - - - + - - + + - + - - + + + + - - + - + - +
+ - - + - - + + - - - - - + - + + - + + + + + + + - - - - - - +
+ - - - - - + + - - - + + + - + - + + - - - - - - - - - - - + +
+ - - - - + + + + + - - - - - + + + - - - - - + - - - - - - + +
- - - - - + - + + - - - - - + + - + - + + - + + + + + - + - - -
- - - - - + + + + - + - - - - - + + + + + + - - + + + + + - + -
+ - - + + - + - + + + - + - - - - - - + + + - - + + + + - + + -
+ + - - - - - + + + + + + - - + - - - + + + - + - - + + - + + -
- - + + - + + + + + + + + - - - - - + + - + + - - - + + - + - -
- - - - + + + + - - - + + + + + + - + - - - - - + - - + + + - -
- + - + - + + + - - + + - + + + + - - + - + + - + - - - + + - +
- + - - - + + + - + + - - + - + + + + + + + - - + + + + + + - -
+ + + - - + + + + + - + + + - - + + + + - - - - + + - + + + + -
+ + - + + + + - - - + + + + + + + - - + - - - - - + + + + - + +
- - - - + + + + - + + - + - - + - - + - + + - + + + + + - - - -
- + + - + + + - - - + + + - + - - - + + + - - - + - + + + - - -
- + + + + - + + - - + + + - + - + + - - + + - - - + + + + - - +
- - + - - - - - + - - - - - + - + + + - + - - + - - - + - - + -
- + - + - - + + + - - - - - + - + + + + + + + + - - - - - - - -
- - - + + + + + - - - - - - + + + + + + - - - - - - + + - - - -
+ + + + + - + - - + - + + - - + + + + - + - - + + - + + - - - -
+ + + + - + + - - + + - + - - - - - - + - - + + + - + + - - + -
+ - - - - - - - - + - + + + - - + - + - - + + + - + + + - + - -
+ - - - - - - - + + + - + - - - + + + - + - - + + + - + - + - -
+ + + - + + + + - + - - + + - - - + - + - - - + + + + - - - - +
+ + + + + + - - - + - - - - + + - + + + + + + + + - - + + - - -
+ + + + + - - - + + + + - - - + - - + + + + - + + + + - + + - +
```

random number *'r'* between 0 and 1 is generated using the *'Rand()'* function so that if (2) is satisfied the spin is flipped.

$$r < e^{\frac{-\Delta E}{k_b T}} \tag{2}$$

Then the method is repeated for a number of Monte Carlo simulations 'N_{MCS}.'

B. Data Collection

In order for the simulation to reach a thermalised state, the first half of the Monte Carlo simulations are discarded and only one configuration every set of simulations is collected to avoid collecting multiples of the same states. The number of Monte Carlo Simulations of 500000 was chosen with Temperature increments of 0.02K between 0.50 and 6.00 Kelvin to limit the time for each lattice run to under 30 mins running on a single 4.2Ghz processor.

C. Magnetisation Calculation and Error

The magnetisation per lattice spin at each temperature was calculated using (3) where N_s is the number of spins in the lattice.

$$m = \frac{1}{N_s} |\sum_i S_i| \tag{3}$$

if 'm' is calculated for each configuration of the thermalised system the average value per configuration was calculated using (4) where N_c is the number of configurations for which the magnetisation was calculated.

$$\langle m \rangle = \frac{1}{N_c} \sum_{Configurations} m(c) \tag{4}$$

The error in the magnetisation cannot be calculated using standard error formulae and have to be calculated using bootstrapping this leads to the error in magnetisation (5)

$$\Delta m = \sqrt{\frac{\langle m^2 \rangle - \langle m \rangle^2}{N_c}} \tag{5}$$

D. Specific Heat Calculation and Error

The specific heat of the lattice is a function of the energy and temperature of the configuration as well as the spin (6)

$$C_v = \frac{\langle E^2 \rangle - \langle E \rangle^2}{N_s T^2} \tag{6}$$

This gives a specific heat value per spin of the lattice and when plotted against temperature the peak specific seat is at the Curie temperature. The error in specific heat also has to be calculated in the same method as magnetisation (7)

$$\Delta C_v = \sqrt{\frac{\langle E^2 \rangle - \langle E \rangle^2}{N_c}} \tag{7}$$

IV. RESULTS

A. Magnetisation

In Fig 3. the phase transition can be seen as the point with the highest '$\frac{dm}{dT}$' which is a clear point shows the trends between lattices sizes 8x8, 16x16 and 32x32 with a much sharper phase transition for larger lattices due to greater probability of random fluctuations due to the increased lattice vertexes.

B. Specific Heat

In Fig 4. as the lattice size increases the peak sharpness also increases. The Curie temperature is the temperature at which the heat capacity is at a maximum.

FIG. 3. Magnetisation in A/m plotted against Temperature in Kelvin

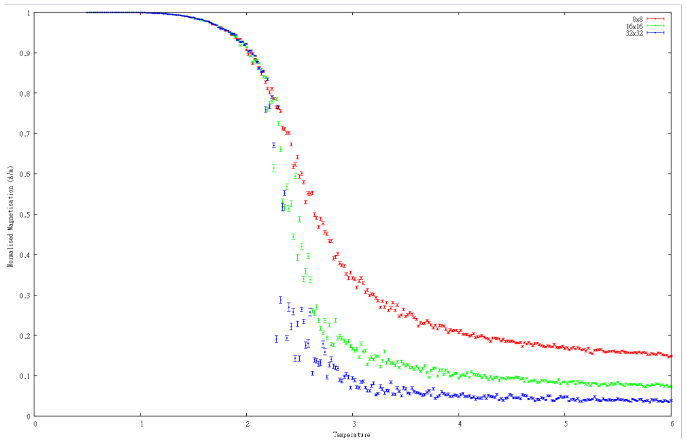
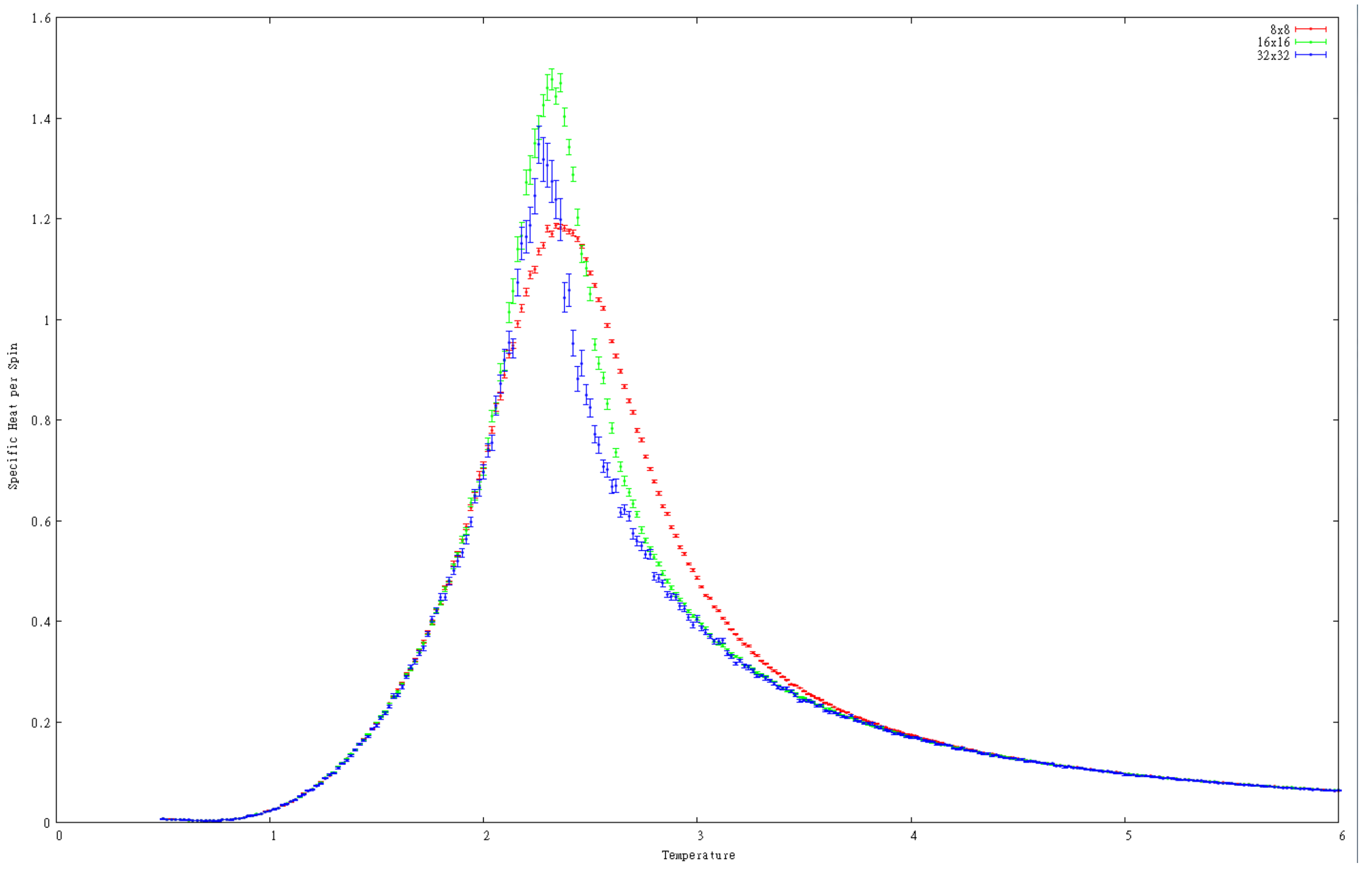


FIG. 4. Specific Heat per Spin plotted against Temperature in Kelvin



C. Curie Temperature

From the specific heat peaks the values obtained for the Curie temperature for each lattice are as follws; 8x8 $(2.34 \pm 0.02)K$, 16x16 $(2.32 \pm 0.02)K$, 32x32 $(2.26 \pm 0.02)K$ the error being the magnitude of the temperature step used in the simulation. The 32x32 compares well to the researched value of 2.27K [1](L. Onsager) for a two dimensional array whilst the 8x8 and 16x16 models don't quite meet the accuracy of the larger lattice.

V. CONCLUSION

Whilst the 32x32 lattice is within accuracy of the researched value a more accurate and precise value could have been obtained using an again larger lattice with even more Monte Carlo simulations a source of error that hasn't been explored is the randomness of the 'rand()' function when used in the simulation. As 'rand()' has a range of [0,32767] reducing it to the between 0 and 1 , and 0 and the size of the lattice is clearly divisible by the output range that combined with truncation produces an uneven distribution leading to some numbers being preferential and a non uniform number distribution that will only be exacerbated by larger Monte Carlo repetitions. Unfortunately the simulation does not lend itself to multi-threading increasing the potential time taken to run simulations making multi-core processors less efficient. Extrapolating into three dimensions would give a more physically applicable value for the curie temperature but again is not feasible with the available computational power.

VI. REFERENCES

[1] L. Onsager, Phys. Rev. 65, 117 (1944). http://dx.doi.org/10.1103/PhysRev.65.117

VII. APPENDIX

1. Compiler details

MinGw gcc using the Eclipse Luna IDE and in-built CDT Toolchain. All code is hosted on DCVS https://github.com/jackomayo/isingsimulation.git

2. Main

This specific run is for a very long running simulation purely to see the difference caused by smaller T increment and more MC simulations the data taken in the report was from variables specified in the report.

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#include <fstream>
#include <iomanip>
#include "Isingsystem.h"
#include "Isingparticle.h"
#include "Statistics.h"
#define ISINGSIZE 64
using namespace std;

int main (void)

{

cout<<"Are we cooking?\n.........................\n\n"<<endl;

setvbuf(stdout, NULL, _IONBF, 0);
setvbuf(stderr, NULL, _IONBF, 0);


 ofstream myfile;
 myfile.open ("64x64simulationresultsmega.txt");


 //energy variables and perturbation condition

double E_i,E_f,deltaE,r;
float T;//for some reason wont work as a double

IsingSystem lattice;

 int i,j;
 int const Nmcs=100000, rmax=100;

 //variable init
 double mag_av, Merr,Cerr,Cv;

 cout<<"Cooking data:"<<endl;

for(T=0.5;T<6;T+=0.02){

//debug print to make sure all works
//cout<<"t= "<<T<< endl;


  Statistics specheat;
  Statistics magnet;
  Statistics specheaterror;

   for(j=0;j<rmax;j++){

//declare energy and megnetisation set for individial configurations



     for(i=0;i<Nmcs;i++){
       //particle choose
       lattice.choose();


       E_i=lattice.localEnergy();

       //change spin
       lattice.perturb();

       //sets energy
       E_f=lattice.localEnergy();

       deltaE=E_f-E_i;

       //random number for pet cond.
       r=((double) rand()/RAND_MAX);

       //Perturbation condition
       if (r>exp(-1*(deltaE/T))) lattice.perturb();


       //point conditions to prevent multiple configurations and for thermalisation to occur

       if(i>Nmcs/2 && i%(ISINGSIZE*ISINGSIZE)==0){



           magnet.add(lattice.magnetisation());

           specheat.add(lattice.totalEnergy());


           }



     }




      Cv =(1/((T*T)*(ISINGSIZE*ISINGSIZE))) *( (specheat.getSqAverage() - specheat.getAverage()*specheat.getAverage()));


       specheaterror.add(Cv);



   }




     //Magnetisation
     mag_av=magnet.getAverage();
     //specific heat error
     Cerr=(1/sqrt(specheaterror.getNumber()))*sqrt(specheaterror.getSqAverage() - pow(specheaterror.getAverage(),2));


     //magnetisation error
         Merr=(1/sqrt(magnet.getNumber()))*sqrt(magnet.getSqAverage() - pow(magnet.getAverage(),2));



myfile<< setprecision(7) << T <<" "<< mag_av <<" "<< Merr << " "<< Cv <<" "<< Cerr << endl;




cout<< T <<" "<< mag_av <<" "<< Merr <<" "<< Cv <<" "<< Cerr << endl;


//printout asci representations below and above curie temp

if((T<=1.501 && T>=1.499) || (T<=5.001 && T>=4.999)){
        cout<<T<<endl;
        lattice.print();
        cout<<endl;
}


//reset array
  lattice.reset();

 }




myfile.close();

cout<<"cooked!"<<endl;

 return 0;
}
```

3. Isingsystem Object Implementation

```
/*
 * Isingsystem.cpp
 *
 *  Created on: Feb 24, 2015
 *      Author: jmayo
 */


#include<stdlib.h>
```

```
#include<stdio.h>
#include<iostream>
#include<cmath>
#include "Isingsystem.h"
#include "Isingparticle.h"
#define ISINGSIZE 64

// sets randomly selected particle to 0,0
IsingSystem::IsingSystem(){

//       initialisation necessary to work on windows (not on linux tho) shouldnt be necessary no idea why it is

        current_column=0;
        current_row=0;
        up=current_row+1;
        down=current_row-1;
        right=current_column+1;
        left=current_column-1;
 }
// selects a particle in the array
void IsingSystem::choose(){
  current_column=rand() % ISINGSIZE;
  current_row=rand() % ISINGSIZE;
 // cout<< "Co-ordinates are"<<current_column<< << current_row<< endl; //exercise 2 test debug

 }
//initial spin flip for peturbation
void IsingSystem::perturb(){
  particles[current_row][current_column].flipspin();
 }
// ASCI print of spin config used
void IsingSystem::print(){
  int i,j;
  for(i=0;i<ISINGSIZE;i++){
    for(j=0;j<ISINGSIZE;j++){
    if( particles[i][j].spinValue()>0 ){ printf("+ ");}

    else{ printf("- ");}
    }
  printf("\n");
  }
 printf("\n");
 }
//particle's neighbours
void IsingSystem::find_neighbours(){
  if (current_row==0) { up=ISINGSIZE-1;}
  else{ up=current_row-1;}

  if( current_row==ISINGSIZE-1) { down=0;}
  else{ down=current_row+1;}

  if(current_column==0) {left=ISINGSIZE-1 ;}
  else{ left=current_column-1;}

  if(current_column==ISINGSIZE-1){ right=0 ;}
  else{ right=current_column+1;}
 }
//all +ve
void IsingSystem::reset(){
  int i,j;
  for(i=0;i<ISINGSIZE;i++){
    for(j=0;j<ISINGSIZE;j++){
      if (particles[i][j].spinValue()<0) particles[i][j].flipspin();
    }
  }
 }

//Calculates energy of randomly selected particle
double IsingSystem::localEnergy(){
  double E=0;
  find_neighbours();
  if(particles[current_row][current_column].spinValue()==particles[up][current_column].spinValue())
  { E-- ;}
  else{ E++;}

  if(particles[current_row][current_column].spinValue()==particles[down][current_column].spinValue())
  { E--;}
  else{ E++;}

  if(particles[current_row][current_column].spinValue()==particles[current_row][left].spinValue())
  {E--;}
  else{ E++;}

  if(particles[current_row][current_column].spinValue()==particles[current_row][right].spinValue())
  { E--;}
  else{ E++;}
  return E;
 }
/*Sums and halves all the local energies of the array for total enegry of the system*/
double IsingSystem::totalEnergy(){
  double E_tot=0;
  int store_row,store_column;
  store_row=current_row;
  store_column=current_column;
  int i,j;
  for(i=0;i<ISINGSIZE;i++){
    for(j=0;j<ISINGSIZE;j++){
       current_row=i;
       current_column=j;
       E_tot+=localEnergy();
    }
  }
  current_row=store_row;

  current_column=store_column;
  return E_tot/2;
 }

// sums spin values for magnetisation
double IsingSystem::magnetisation(){
  double m=0;
  int i,j;
  for(i=0;i<ISINGSIZE;i++){
    for(j=0;j<ISINGSIZE;j++){
     m+=particles[i][j].spinValue();
    }
  }
 m*=(1/(pow(ISINGSIZE,2)));
  return fabs(m);
 }

IsingSystem::~IsingSystem() {
        // TODO Auto-generated destructor stub
}
```