***Form 1.cs***

﻿using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace IsingMonteCarlo

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Form2 frm2 = new Form2();

MonteCarlo mc = new MonteCarlo(this,frm2);

int n\_sweeps = 100;//at every temperature value, spin system will be swept

//n\_sweep times

double M=0, Mave=0,Tc=0;//M is magnetization during one sweep

//Mave is average magnetization of all sweeps

bool check;

frm2.Show();

for(double T=0.1;T<10;T=T+0.2)//Temperature loop for MFT as well as Monte Carlo

{

Mave = 0;

for (int sweep = 0; sweep < n\_sweeps;sweep++)//sweep loop

{

M = 0;//at the start of each sweep, M is taken zero

for (int i = 1; i < (mc.n\_spins - 1); i++)//both for loops represent

//single sweep

{

for (int j = 1; j < (mc.n\_spins - 1); j++)

{

mc.E1 = mc.EnergyCal(i, j);//it computes internal energy

//of spin at ith row and jth column, before its flipping

//now assumingly flip the spin

mc.spinsystem[i, j] = mc.spinsystem[i, j] \* -1;//flip the spin

mc.E2 = mc.EnergyCal(i, j);//it computes internal energy

//of spin at ith row and jth column, after its flipping

check = mc.Flip(T);//check wheter spin flipping

//is accepted or not

if (check == true)

{

//accept the flip

if (mc.spinsystem[i, j] == -1)//if spin-down

mc.gg.FillEllipse(mc.bred, 300 + j \* 10, 200 + i \* 10, 8, 8);

else//if spin-up

mc.gg.FillEllipse(mc.bblue, 300 + j \* 10, 200 + i \* 10, 8, 8);

}

else//check==false means flipping is rejected

{

mc.spinsystem[i, j] = mc.spinsystem[i, j] \* -1;

}

M = M + mc.spinsystem[i, j];//

}//internal for loop

}//external for loop

Mave = Mave + M / (mc.n\_spins \* mc.n\_spins - 4 \* mc.n\_spins + 4);

}//end of sweeps loop

Mave = Mave / n\_sweeps;//average magnetization at single value

//of temperature

//\*\*\*\*\*\*\*\*\*\*MFT Computations for the average magnetization\*\*\*\*\*\*

for (double s = 0; s < 1.5; s = s + 0.01)

{

if(Math.Abs(s-Math.Tanh(mc.J\*mc.Z\*s/(mc.KB\*T)))<0.001)

mc.gg1.FillEllipse(mc.bred, 150 + (float)T \* 100,

400 - (float)s \* 300, 8, 8);

}

mc.gg1.FillEllipse(mc.bblue, 150 + (float)T \* 100,

400 - (float)Mave \* 300, 8, 8);

if (Math.Abs(Mave) < 0.01) { Tc = T;}

//graph of average magnetization versus T

}//Temperature Loop

MessageBox.Show("Tc=" + Tc.ToString());

}

private void button2\_Click(object sender, EventArgs e)

{//First order phase transition

Form2 frm2 = new Form2();

MonteCarlo mc = new MonteCarlo(this, frm2);

int n\_sweeps = 100;//at every temperature value, spin system will be swept

//n\_sweep times

double M = 0, Mave = 0, T = 0.2;//M is magnetization during one sweep

//Mave is average magnetization of all sweeps

bool check;

mc.gg.DrawLine(mc.pblue, 150, 300, 950, 300);

mc.gg.DrawLine(mc.pblue, 550, 700, 550, 30);

mc.gg.DrawString("M", mc.f, mc.bblue, 500, 100);

mc.gg.DrawString("H", mc.f, mc.bblue, 800, 320);

//frm2.Show();

for (double H = -5; H < 5; H = H + 0.2)//H loop

{

Mave = 0;

for (int sweep = 0; sweep < n\_sweeps; sweep++)//sweep loop

{

M = 0;//at the start of each sweep, M is taken zero

for (int i = 1; i < (mc.n\_spins - 1); i++)//both for loops represent

//single sweep

{

for (int j = 1; j < (mc.n\_spins - 1); j++)

{

mc.H = H;

mc.E1 = mc.EnergyCal(i, j);//it computes internal energy

//of spin at ith row and jth column, before its flipping

//now assumingly flip the spin

mc.spinsystem[i, j] = mc.spinsystem[i, j] \* -1;//flip the spin

mc.E2 = mc.EnergyCal(i, j);//it computes internal energy

//of spin at ith row and jth column, after its flipping

check = mc.Flip(T);//check wheter spin flipping

//is accepted or not

if (check == true)

{

//accept the flip

if (mc.spinsystem[i, j] == -1)//if spin-down

mc.gg.FillEllipse(mc.bred, 300 + j \* 10, 200 + i \* 10, 8, 8);

else//if spin-up

mc.gg.FillEllipse(mc.bblue, 300 + j \* 10, 200 + i \* 10, 8, 8);

}

else//check==false means flipping is rejected

{

mc.spinsystem[i, j] = mc.spinsystem[i, j] \* -1;

}

M = M + mc.spinsystem[i, j];//

}//internal for loop

}//external for loop

Mave = Mave + M / (mc.n\_spins \* mc.n\_spins - 4 \* mc.n\_spins + 4);

}//end of sweeps loop

Mave = Mave / n\_sweeps;//average magnetization at single value

//of temperature

mc.gg.FillEllipse(mc.bblue, 550 + (float)H \* 50,

400 - (float)Mave \* 300, 8, 8);

//graph of average magnetization versus H

}//H Loop

//Backward Direction

for (double H = 5; H >=-5; H = H - 0.2)//H loop

{

Mave = 0;

for (int sweep = 0; sweep < n\_sweeps; sweep++)//sweep loop

{

M = 0;//at the start of each sweep, M is taken zero

for (int i = 1; i < (mc.n\_spins - 1); i++)//both for loops represent

//single sweep

{

for (int j = 1; j < (mc.n\_spins - 1); j++)

{

mc.H = H;

mc.E1 = mc.EnergyCal(i, j);//it computes internal energy

//of spin at ith row and jth column, before its flipping

//now assumingly flip the spin

mc.spinsystem[i, j] = mc.spinsystem[i, j] \* -1;//flip the spin

mc.E2 = mc.EnergyCal(i, j);//it computes internal energy

//of spin at ith row and jth column, after its flipping

check = mc.Flip(T);//check wheter spin flipping

//is accepted or not

if (check == true)

{

//accept the flip

if (mc.spinsystem[i, j] == -1)//if spin-down

mc.gg.FillEllipse(mc.bred, 300 + j \* 10, 200 + i \* 10, 8, 8);

else//if spin-up

mc.gg.FillEllipse(mc.bblue, 300 + j \* 10, 200 + i \* 10, 8, 8);

}

else//check==false means flipping is rejected

{

mc.spinsystem[i, j] = mc.spinsystem[i, j] \* -1;

}

M = M + mc.spinsystem[i, j];//

}//internal for loop

}//external for loop

Mave = Mave + M / (mc.n\_spins \* mc.n\_spins - 4 \* mc.n\_spins + 4);

}//end of sweeps loop

Mave = Mave / n\_sweeps;//average magnetization at single value

//of temperature

mc.gg.DrawEllipse(mc.pred, 550 + (float)H \* 50,

400 - (float)Mave \* 300, 12, 12);

//graph of average magnetization versus H

}//H Loop

}//event handler

}//end of class

}//end of namespace

***Form2.cs***

﻿using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace IsingMonteCarlo

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

}

}

﻿ ***MonteCarlo.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

namespace IsingMonteCarlo

{

class MonteCarlo

{

//data variables

public Graphics gg,gg1;

public SolidBrush bred, bblue;

public Pen pblue,pred;

public Font f;

//take two dimensional array to represent two dimensional spin system

public int [ , ] spinsystem;

public int n\_spins=21;

bool test;

Random obj;

public double E1, E2, E\_Flip, P\_Flip, r, J, KB,Z,mu,H;

public MonteCarlo(Form1 frm1,Form2 frm2)

{

gg = frm1.CreateGraphics();

gg1 = frm2.CreateGraphics();

bred = new SolidBrush(Color.Red);

bblue = new SolidBrush(Color.Blue);

pblue = new Pen(Color.Blue, 5);

pred = new Pen(Color.Red, 5);

f = new Font("Arial", 16);

spinsystem = new int[n\_spins, n\_spins];

obj = new Random();

test = false;

E1 = 0; E2 = 0; E\_Flip = 0; P\_Flip = 0; r = 0; J = 1; KB = 1;

mu = 1; Z = 4; H = 0;

//show the spin system

for (int i = 0; i < n\_spins; i++)

{

for (int j = 0; j < n\_spins; j++)

{

r = obj.NextDouble();

if (r <= 0.5)

{

spinsystem[i, j] = 1;//spin up

gg.FillEllipse(bblue, 300 + j \* 10, 200 + i \* 10, 8, 8);

}

else

{

spinsystem[i, j] = +1;//spin up

gg.FillEllipse(bblue, 300 + j \* 10, 200 + i \* 10, 8, 8);

}

}

}//spin system shown

}//end of constructor

//other functions

public double EnergyCal(int r, int c)

{

return (- (int)J \* spinsystem[r, c] \* (spinsystem[r - 1, c]

+ spinsystem[r + 1, c] +

spinsystem[r, c - 1] + spinsystem[r, c + 1]+mu\*H));

}

public bool Flip(double T)

{

E\_Flip = E2 - E1;

if (E\_Flip <= 0)

test=true;//spin will flip

else//E\_flip>0

{

P\_Flip = Math.Exp(-E\_Flip / (KB \* T));//probability of flipping

r=obj.NextDouble();//random probability

if (P\_Flip >= r)

test = true;

else

test = false;

}

return test;

}

}

}