PHYS580 Lab11 Report

Yicheng Feng
PUID: 0030193826

November 3, 2019

Workflow: I use the Linux system, and code C++ in terminals. For visualization, I use the C++ package – ROOT (made by CERN) to make plots. At the end, the reports are written in LaTeX.

The codes for this lab are written as the following files:

- `ising_model_2d.h` and `ising_model_2d.cxx` for the class `IsingModel2D` to simulation the 2D Ising model and calculate the magnetization and energy per spin.
- `lab11.cxx` for the main function to make plots.

To each problem of this lab report, I will attach the relevant parts of the code. If you want to check the validation of my code, you need to download the whole code from the link `https://github.com/YichengFeng/phys580/tree/master/lab11`.

(1) Use the starter programs `ising.m` and `calculate_spin.m` (or your own equivalent routines) to simulate the Ising ferromagnet. First, try a small square grid (e.g., 10×10 or 15×15) to see how these programs work. Then, turn graphics off* for speed and do production runs on a bigger lattice, say, 50×50 (or even larger), at temperatures T sampled in regular increments both above and below $T_C \approx 2.27$ (measured in units of J/k_B), for a few different values of the magnetic field H (measured in units of J/μ). Make sure to calculate for $H = 0$, as well as positive and negative H. For initial conditions, you can try, e.g., having all spins up. Observe and note how the magnetization per spin (m) and energy per spin (E) converge towards their equilibrium values, as well as the qualitative features of the equilibrium configurations that may be reached. For example, check how the rate of convergence depends on the temperature and/or the field.

* Use the lattice plots of red (up) and blue (down) sites displayed by the codes to get an idea of how the spin updates proceed, but later do turn the graphics off for faster calculations. A few hundred MCS per spin should suffice to see the trends in most (but not necessarily all) cases. Here, trends refers to the following: duration of the initial transient, time needed for an accurate average in equilibrium, and the magnitude of fluctuations.

Physics explanation:

A 50×50 grid is simulated for the 2D Ising model. Various temperatures and magnetic fields are tried. To save space, I only plot some combinations of them: $T = 2.00, 3.00$ and $H = -0.20, 0.00, 0.20$. The initial state is all spin up.

The duration of the initial transient will be roughly estimated by eyes. The stable states will be measured after $t = 50$ when the initial transients are obviously finished. The stable states are fitted by a horizontal line. For the temperature close to T_C, the stable states should be measured for a longer time, because the flipping might happen. Since the temperatures here are quite far from T_C, the stable states are only measured during $50 < T \leq 100$.

Figure 1–6 show the simulation results with various combinations of T and H. Left pad is the spin map when stable. Middle pad is the time dependence of the magnetization per spin (m). Right pad is the time dependence of the energy per spin (E). The numerical results are packaged into the following table.

T (J/k_B)	H (J/μ)	m	$\sigma[m]$	E	$\sigma[E]$	initial transient
2.00	0.00	0.9175 ± 0.00188	0.0133	-1.754 ± 0.00457	0.0323	10
	0.20	0.9473 ± 0.00140	0.0099	-2.015 ± 0.00416	0.0294	3
	-0.20	-0.9489 ± 0.00111	0.0078	-2.019 ± 0.00330	0.0233	32
3.00	0.00	-0.0205 ± 0.01000	0.0071	-0.832 ± 0.00580	0.0410	30
	0.20	0.5475 ± 0.00456	0.0032	-1.171 ± 0.00580	0.0410	18
	-0.20	-0.5466 ± 0.00467	0.0033	-1.159 ± 0.00739	0.0523	12

When the input magnetic field goes from positive to negative, the final magnetization per spin also goes from positive to negative, so the final state is farther away from the initial state. Therefore, it tends to take longer in the initial transient ($T = 2$).

When the temperature is high ($T > T_C$), the stable state will go close to $m = 0$, which is the exact the case with $T = 3$, $H = 0$, so the initial state would have no contribution to the final state. When the temperature is low ($T < T_C$), the stable state will be close to $m = \pm 1$ depending on H and the initial state.

Plots:

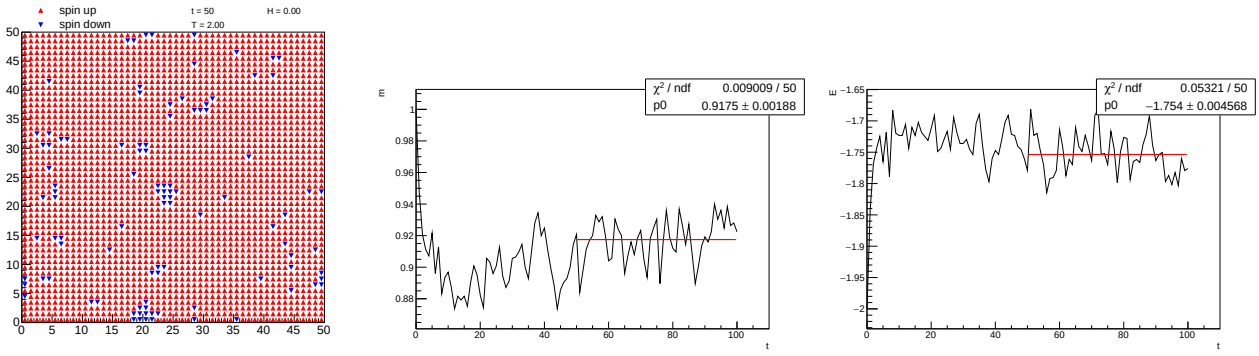

Figure 1: The plots above show the simulation result of the 2D Ising model with $T = 2.00$ and $H = 0.00$. Left pad is the spin map when stable. Middle pad is the time dependence of the magnetization per spin (m). Right pad is the time dependence of the energy per spin (E).

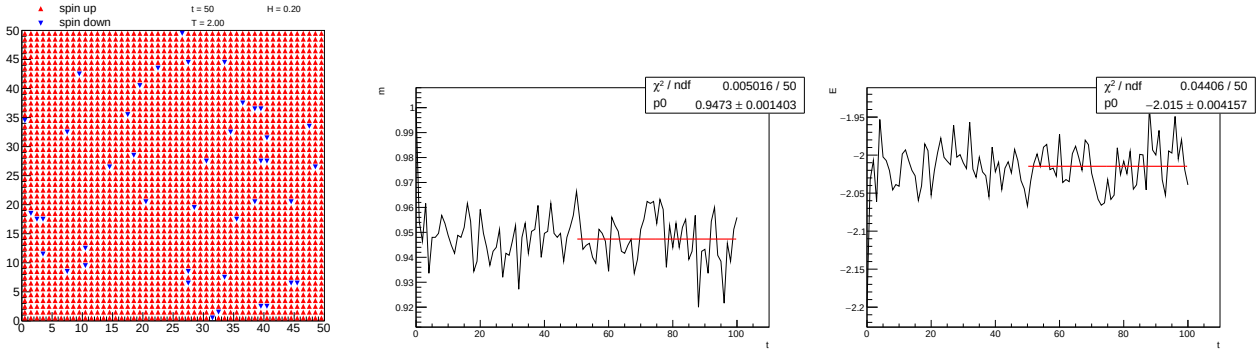

Figure 2: The plots above show the simulation result of the 2D Ising model with $T = 2.00$ and $H = 0.20$. Left pad is the spin map when stable. Middle pad is the time dependence of the magnetization per spin (m). Right pad is the time dependence of the energy per spin (E).

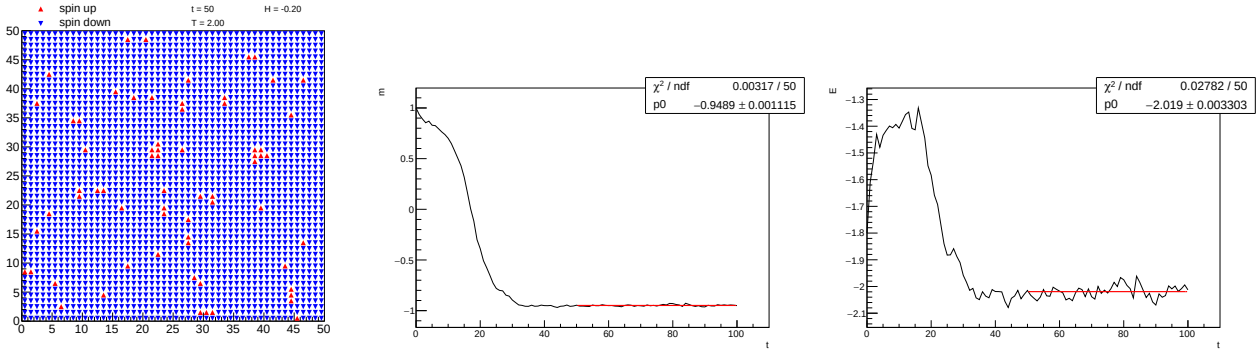

Figure 3: The plots above show the simulation result of the 2D Ising model with $T = 2.00$ and $H = -0.20$. Left pad is the spin map when stable. Middle pad is the time dependence of the magnetization per spin (m). Right pad is the time dependence of the energy per spin (E).

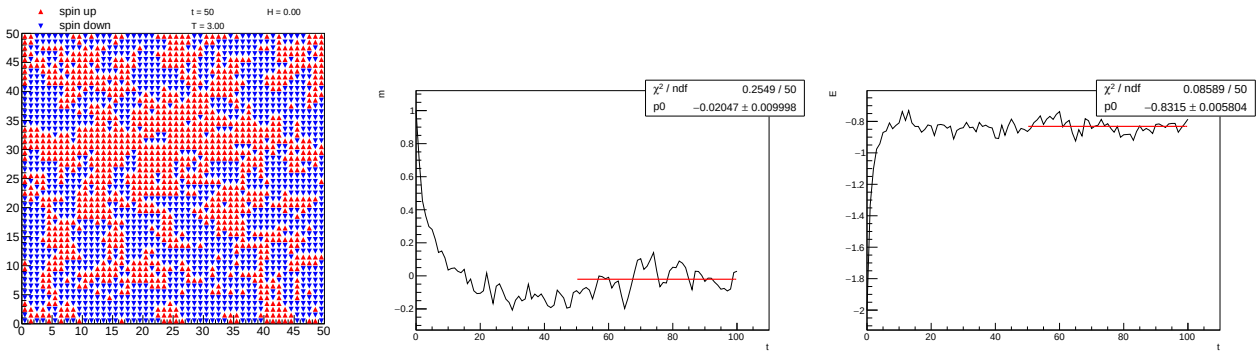

Figure 4: The plots above show the simulation result of the 2D Ising model with $T = 3.00$ and $H = 0.00$. Left pad is the spin map when stable. Middle pad is the time dependence of the magnetization per spin (m). Right pad is the time dependence of the energy per spin (E).

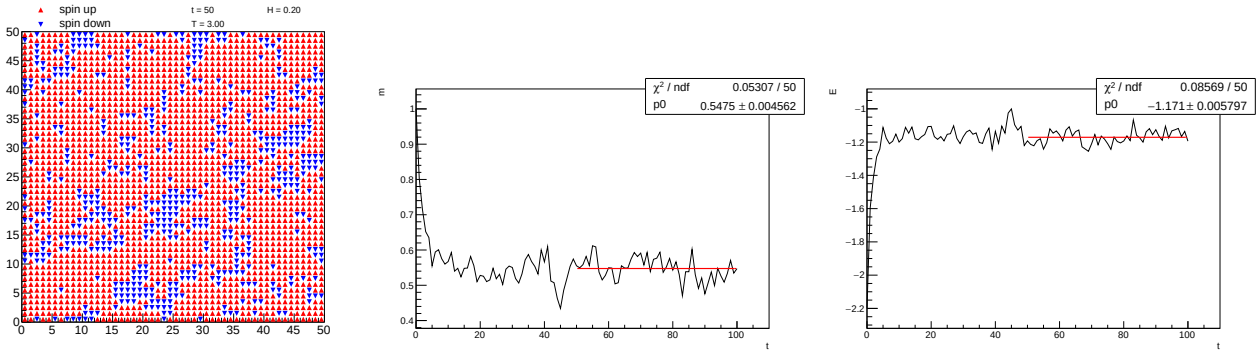

Figure 5: The plots above show the simulation result of the 2D Ising model with $T = 3.00$ and $H = 0.20$. Left pad is the spin map when stable. Middle pad is the time dependence of the magnetization per spin (m). Right pad is the time dependence of the energy per spin (E).

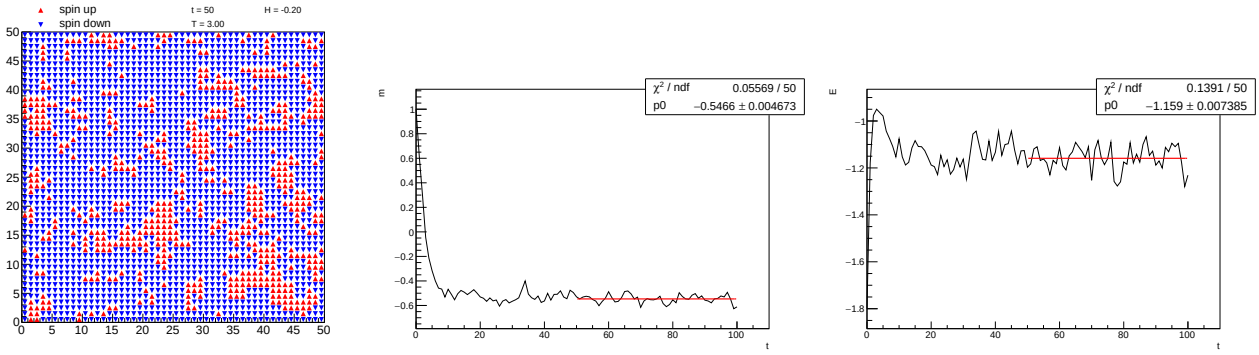

Figure 6: The plots above show the simulation result of the 2D Ising model with $T = 3.00$ and $H = -0.20$. Left pad is the spin map when stable. Middle pad is the time dependence of the magnetization per spin (m). Right pad is the time dependence of the energy per spin (E).

Relevant code:

For the initialization of the simulation

```
//------------------------------------------------------------------------//

IsingModel2D::IsingModel2D(int N, double T, double H, int MCS, int c) {

        _N = N;
        _T = T;
        _H = H;
        _MCS = MCS;
        _c = c; // (1: all up; -1: all down; 2: random)

        _now = 0;

        _spin.clear();
        for(int i=0; i<_N; i++) {
                vector<int> tmp;
                for(int j=0; j<_N; j++) {
                        int one_spin;
                        if(_c == 1) {
                                one_spin = 1;
                        } else if(_c == -1) {
                                one_spin = -1;
                        } else {
                                one_spin = 1.0*rand()/RAND_MAX<0.5?-1:1;
                        }
                        tmp.push_back(one_spin);
                }
                _spin.push_back(tmp);
        }

        _m = 0;
        _E = 0;
        for(int i=0; i<_N; i++) {
                int il = i-1<  0? i-1+_N:i-1;
                int ir = i+1>=_N? i+1-_N:i+1;
                for(int j=0; j<_N; j++) {
                        int jl = j-1<  0? j-1+_N:j-1;
                        int jr = j+1>=_N? j+1-_N:j+1;

                        _m += _spin[i][j];
                        double sum = _spin[il][j] + _spin[ir][j] + _spin[i][jl] + _spin[i][jr];
                        _E -= _spin[i][j]*(_H + 0.5*sum);
                }
        }
}

//------------------------------------------------------------------------//
```

For the Metropolis Monte Carlo simulation

```
//------------------------------------------------------------------------//

void IsingModel2D::cal_once() {

        for(int i=0; i<_N; i++) {
                int il = i-1<  0? i-1+_N:i-1;
                int ir = i+1>=_N? i+1-_N:i+1;
                for(int j=0; j<_N; j++) {
                        int jl = j-1<  0? j-1+_N:j-1;
```

```
                        int jr = j+1>=_N? j+1-_N:j+1;

                        double sum = _spin[il][j] + _spin[ir][j] + _spin[i][jl] + _spin[i][jr];
                        double DeltaE = 2*_spin[i][j]*(sum + _H);
                        if(DeltaE<0 || exp(-DeltaE/_T)>1.0*rand()/RAND_MAX) {
                                _spin[i][j] = -_spin[i][j];
                                _m += 2*_spin[i][j];
                                _E += DeltaE;
                        }
                }
        }
}

//---------------------------------------------------------------------------//

void IsingModel2D::cal_until(int t) {

        if(!check()) return;

        for(int i=_now; i<t; i++) {
                cal_once();
        }

        _now = t;
}

//---------------------------------------------------------------------------//
```

(2) Now modify your program such that it reads an additional transient duration parameter and then calculates the means of the magnetization and energy together with the standard errors of the means over a given number of MCS per spin after the transient. Plot the magnetization and energy (with errors) as a function of temperature over a range of T around T_C, with magnetic field set to a very small positive value $H = 0+$ (this should help tame fluctuations). Set the duration of the transient and the number of MCS per spin over which to take the mean/fluctuation based on your observations from (1). Note any similarities and differences between m vs T for the Ising magnet and $P(p)$ vs p in percolation.

Physics explanation:

A 50×50 grid is simulated for the 2D Ising model. The initial state has all spin up. Here, I do the temperature scan from 1.5 to 3.5 with step $\Delta T = 0.05$. The input magnetic field is set to $H = 0.01$ to help tame fluctuations.

As we observabled in (1), the initial transient duration is usually smaller than $\Delta t = 50$. To remove the effect of it, the mean value and fluctuation are calculated from $1000 < t < 3000$, which is way outside the possible initial transient. The way to get the mean value and fluctuation of the stable states is as indicated in (1): fitting the stable range with a horizontal line.

Because the fitting range has $n = 2000$ data points, the fitting error ϵ would be very small and it would be hard to see the error bar in the T–m (Fig. 7) and T–E (Fig. 8) plots. We use the error and number of data points to get the standard deviation: $\sigma = \epsilon\sqrt{n}$.

The magnetization per spin m depending on T is shown in Fig. 7. The left pad shows the mean values with error bar; the right pad shows the standard deviation. The left pad is actually very similar to the $P(p)$–p plot: nonzero magnetization per spin m (percolation probability P) in the low temperature T (occupation probability p) decreases to zero at the critical temperature T_C (critical occupation probability p_C); then it keeps zero. The right pad is similar to $S(p)$-p plot: the fluctuation of m (susceptibility S) increase from 0 at low temperature T (occupation probability p) reaches maximum (theoretical infinity) at the critical temperature T_C (critical occupation probability p_C); it then goes down at high temperature T (occupation probability p).

The energy per spin E depending on T is shown in Fig. 8. The left pad shows the mean values with error bar; the right pad shows the standard deviation. E increases with temperature T, and it increases most quickly at the critical temperature T. The fluctuation of E increases in low T and decreases in high T range, and it reaches maximum at the critical temperature T_C.

From the simulation results, we can seen the critical temperature is about $T_C \approx 2.4$ in the 50×50 case, which is slightly different from the $T_C \approx 2.27$ in the 10×10 case.

Plots:

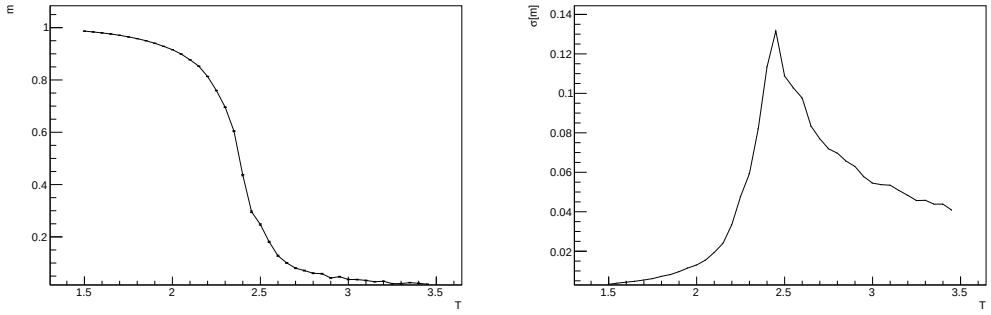

Figure 7: In the stable region, the mean value of m with error bar is shown in the left pad; the standard deviation is shown in the right pad.

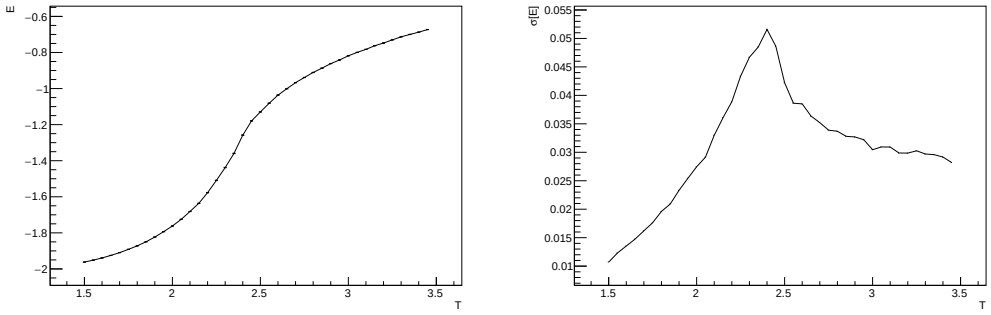

Figure 8: In the stable region, the mean value of E with error bar is shown in the left pad; the standard deviation is shown in the right pad.

Relevant code:

For the simulation code, it has already shown in (1).

```
//--------------------------------------------------------------------------//

void IsingModel2D::cal_once() {

        for(int i=0; i<_N; i++) {
                int il = i-1<  0? i-1+_N:i-1;
                int ir = i+1>=_N? i+1-_N:i+1;
                for(int j=0; j<_N; j++) {
                        int jl = j-1<  0? j-1+_N:j-1;
                        int jr = j+1>=_N? j+1-_N:j+1;

                        double sum = _spin[il][j] + _spin[ir][j] + _spin[i][jl] + _spin[i][jr];
                        double DeltaE = 2*_spin[i][j]*(sum + _H);
                        if(DeltaE<0 || exp(-DeltaE/_T)>1.0*rand()/RAND_MAX) {
                                _spin[i][j] = -_spin[i][j];
                                _m += 2*_spin[i][j];
                                _E += DeltaE;
                        }
                }
        }
}
```

```
22
23 //------------------------------------------------------------------------//
24
25 void IsingModel2D::cal_until(int t) {
26
27         if(!check()) return;
28
29         for(int i=_now; i<t; i++) {
30                 cal_once();
31         }
32
33         _now = t;
34 }
35
36 //------------------------------------------------------------------------//
```