PHYS5630 Computational Physics I (Fall 2020)

Ising and Metropolis

Today we will explore topics in the Ising model and Metropolis method

To begin copy the ising starter code from GitHub to begin. Add your name to the Readme.md file.

The following programs employ the Metropolis method to simulate thermal systems. See class notes for some background text (Ising2 notes).

Input to the programs are defined as follows:

- nsweeps = total number of spin configurations (trials) tested
  We average over these trials to calculate an observable, eg the net magnetization of a system
- h = value of external magnet field in units $H/(k_bT)$
- $\beta = 1/(k_bT)$

Examples (1) and (2) calculate thermal averages for simple systems.
Examples (3) and (4) explore the thermalization of larger 1 and 2d systems using simulated annealing (Metropilis algorithm).
Examples (5) and (6) apply simulated annealing to measure more properties of thermalized systems.

1) onespin.cpp

Examine update function in the program. This is the core implementation of the Metropolis method.

In this example of a single magnetic dipole in an external field, you will control the value of the external field and the number of spin configurations to test. (Temperature effects are implicitly incuded in the value of h=H/kT). In the context of these examples, a sweep is a trial for flipping a spin. The flip is either accepted or reject following the Metropolis method.

$$P(state) \propto exp^{[-h\sigma]}$$

Examine the program before you begin to understand how the Metropolis method is applied.

1. Run several times with a small number of sweeps and a moderate external field: nsweeps = 100, h=0.4. Notice the large stat. fluctuations in the results.
2. Switch to 100000 sweeps
   a. Run with h=0 (why is this exactly right?)
   b. Run with h=4

c. Run with h=-4

How does the simulation compare to the expected result in the limit of many sweeps?

Tired of typing in these parameters over and over again? Try this:

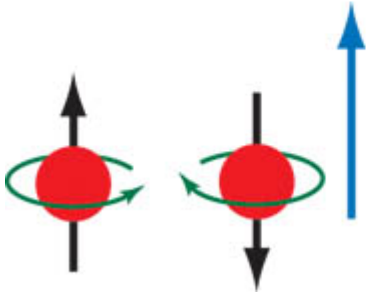
```
echo "100000 4" | ./onespin
```

This trick will be even more convenient to use below.

2) twospin.cpp

In this example two dipoles are included, so the interaction of neighbors must be considered to determine the energy of the system.

$P(state) \propto exp^{[-\sigma_1\sigma_2\beta-(\sigma_1+\sigma_2)h]}$.



The “environment” for σ_1 is given by: $\beta\sigma_2 + h$, this quantity is multiplied by -1 when σ_1 is flipped, so we consider exp^[env_new-env_old] when determining whether to accept a new spin configuration. (similar comment for σ_2). As usual β is defined to be 1/kT.

1. Run with nsweeps = 100000, h=0, $\beta = 4$ (n.b. larger $\beta =$ lower temperature).
    - What do you expect for $< \sigma >=< \sigma_1 + \sigma_2 > /2$
    - What about $< \sigma_1\sigma_2 >$
2. Repeat for $\beta = 0.1$ (a larger temperature wrt the first run)
3. h=4, $\beta = 0.1$

Make sure you understand the methods and results of this simulation before moving on.

3) ising1d.cpp

This program performs a basic 1d Ising simulation. Stretch your terminal so it is at least 30 lines long. Results of *nsweep* trials for a chain of length *N* are printed in ascii. The analytic result for average magnetization in the long time limit is also printed.

Notes on the code:

- end points are handled by adding 0 spin entries at beginning/end of array (these are not updated)
- the sweep function has the full environment variable $\beta\sigma_{i-1} + \beta\sigma_{i+1} + h$
- Notice “ntherm” sweeps are done before starting to “thermalize” the system. Purpose is to get system into a typical configuration for the given beta, h, before simulating sweeps for any calculations

Run with the following options (Use N=#of spins=70, nsweep~30 for all trials below. You can run more sweeps to examine trends over longer time scales, but one the system is thermalized, results should be qualitatively the same for any time period. Using a longer chain will give results closer to the ideal large L limit.):

- h=0, $\beta = 1.5$
  What trends do you see in the domain walls?
  Run a couple of times to see the variations.
- What do you expect for smaller beta (= higher Temperature)?
- h=0, $\beta = 0.5$
- h=0, $\beta = 0.1$
- h=1, $\beta = 0.5$
- h=0.2, $\beta = 0.5$
- h=-0.2, $\beta = 0.5$

You can turn off the visualization by editing the code and recompiling:

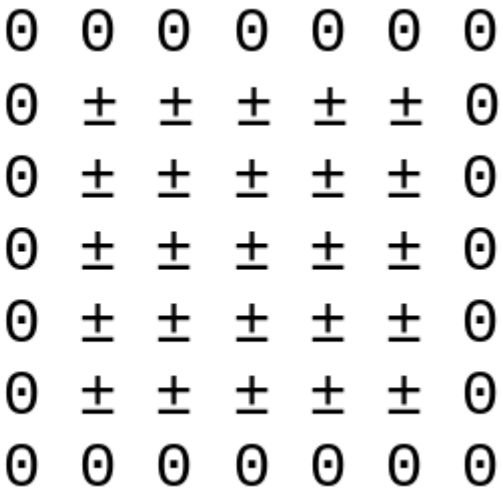
```
const int VisualDisplay=1;  // 1 or 0 to turn on/off display of chains
```

You should be able to observe that increasing the length of the chain and (especially) the number of sweeps gets the simulation to agree better with the theoretical results.

4) ising2d.cpp

2D version of above. See class notes for background information.

This one prints out in 64-lines, so stretch out your terminal. You can also reduce the font size in your terminal to display the output if necessary.



```
0 0 0 0 0 0 0
0 ± ± ± ± ± 0
0 ± ± ± ± ± 0
0 ± ± ± ± ± 0
0 ± ± ± ± ± 0
0 ± ± ± ± ± 0
0 0 0 0 0 0 0
```

Program notes:

- uses “hot start”, ntherm=0. So you can see the thermalization in the display
- Magnetization is the volume average $\bar{\sigma}$ for the current state of the system, not averaged over monte carlo time
- The Display Lattice function prints the lattice, then sleeps for a few seconds between plots so you can see the various system states
- Lattice is hardwired to 64×64 sites

Run program with (#configs at least ~10):

- h=0.2, $\beta = 1$
- h=0.2, $\beta = 0.1$
- h=0, $\beta = 0.1$
- h=0.5, $\beta = 1$
- Also: try running with larger beta after thermalizing for some time (sent ntherm>0 in the code), eg (ntherm=(200, 500, 1000), 500 configurations, h=0, beta=(1,2,4), etc)

The following two examples are illustrations of the Metropolis method to measure the properties of a system as it cools under approximate thermal equilibrium

.

5) ising2d_vs_T.cpp

Computes $< \sigma >$ as a function of T. This is a basic implementation of a simulated annealing process. Writes results to output file and a “typical” configuration is printed to the screen for each T.

Program notes:

- ntherm=1000, “initializes” system for each temperature
- review main() for details

To do:

- Run with parameters: nsweeps=1000, h=0, T_max = 5, #T's =50. Here nsweeps is the number of randomized changes to the lattice per temperature step.
- plot result with gnuplot to see phase transition

```
gnuplot> set xlabel "T"
gnuplot> set ylabel "magnetization"
gnuplot> plot "ising2d_vs_T.dat" w linespoints
```

For 2D (not 3D) it's possible to get an exact theoretical result:

$$T_c = \frac{2}{ln(1+\sqrt{2})} \simeq 2.26919$$

Inexact results for the phase transition (non-zero about T_c) are due to (eg.):

- finite volume
- step sizes in T
- finite simulation time (statistical errors)

Run again with h=0.01(-0.01) and plot the result. Now the phase transition is rounded off, why?

6) ising1d_vs_T.cpp

Run with params: N(number of spins)=100, nsweep=1000, h=0, T_max=5, #T's=50

Plot the result “ising1d_vs_T.dat” and you'll find that $< \sigma > \neq 0$ for small $T \neq 0$, even though the 1D system should not magnetize at finite temperature.
Q: Why?
A: The simulation gets stuck, if it is run very, very long (many sweeps per temperature), it will work eventually. eg,

the following kind of changes will occur:

```
 +++++++ -> +++----
```

Other Monte Carlo algorithms can be used to avoid this problem.

Explore your results with different values of N and nsweep.

Q: How to tell if a phase transition is real?

A: Measure

$$< \bar{\sigma}^2 > \equiv < (\frac{1}{N}\sum_i \sigma_i)^2 >$$

i.e. MC time average of the volume average. As $N \to \infty$, answer will converge.

Explore with other parameters, eg N=1000, T_max=2, #T's=100, etc.

Exercise

The ising2d_v2_T example above implements simulated annealing on a 2D grid and writes out the average magnetization (M) versus temperature of the system.

Following Assessment “15.4.1–15.4.2 Equilibration, Thermodynamic Properties” in Landau, modify this example to also calculate the specific heat (C) and mean energy (E) of the system as a function of temperature.

Plot the three quantities (E,M,C) versus T and upload your result as **ising.pdf** .
Make sure to label your plots clearly.

ex/ising.txt · Last modified: 2020/11/02 03:18 by hirosky