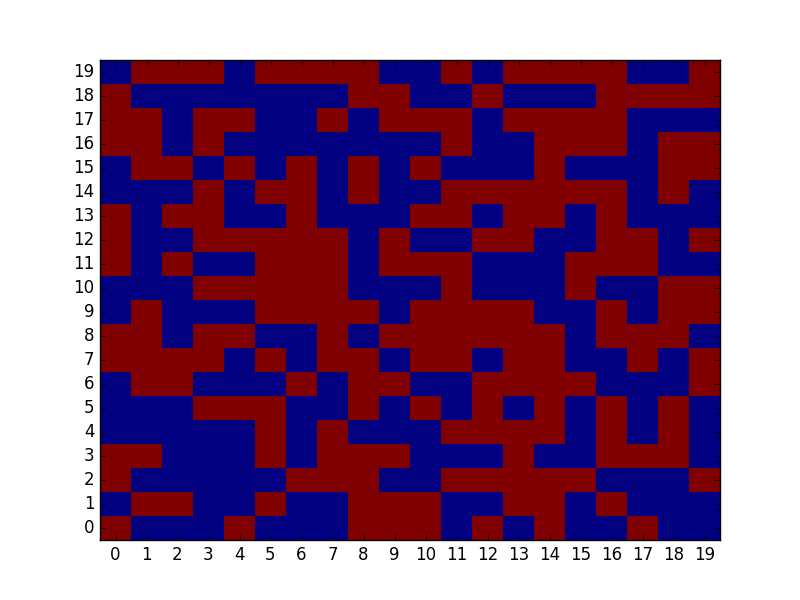
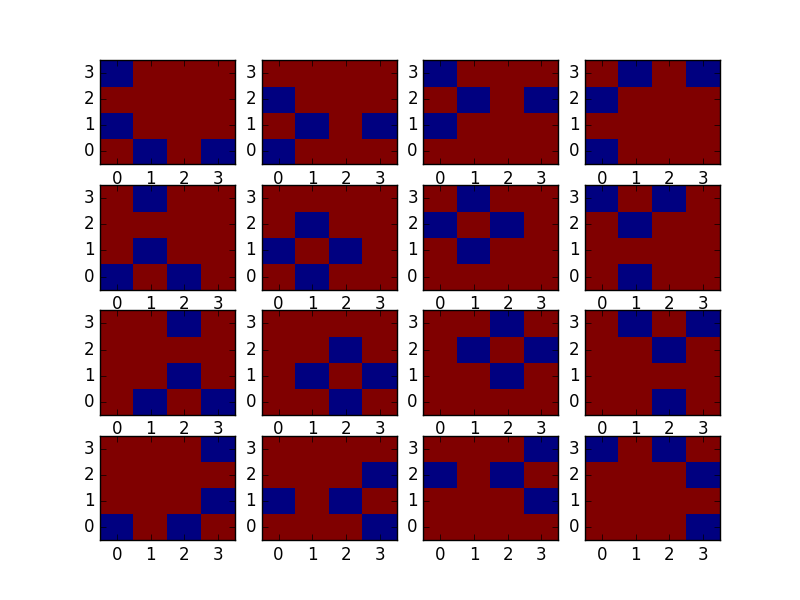
Atomic scale simulation HW6 Jinsheng Wang (NetID: jwang278)

1: system setup

initialize a data structure to hold the lattice site of 20 spins per side with ransom +1 or -1 given as below:



check for correct neighbors with 4 spins per side and all initialized with +1 at the beginning. The figure is shown below:



the above two figures shows that the system setup is correct.

**Conventional Metropolis**

1: T(x→x') = ?

T(x→x') = 1

2: T(x'→x) = ?

T(x'→x) = 1

3. Write down the exp(-δV) where δV = V\_new - V\_old.

exp(-δV) = exp(-beta\* compute\_spin\_energy(i)), where i is the random spin chosen at the beginning. compute\_spin\_energy() will compute the energy change if flip the spin at position i.

4. Finally write down A(x→x')

A(x→x') = min(1, exp(-δV)) = min(1, exp(-beta\* compute\_spin\_energy(i)))

Before flipping spin i is x state, after flipping I is x' state.

The Metropolis loop is in code, uploaded on compass.

The M2 results are shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Beta\*j | mean | stddev | Autocorr. time | Std error |
| 0.3 | 0.01797001 | 0.02182988 | 5.44469764 | 0.00169698 |
| 0.44069 | 0.52868067 | 0.05120047 | 63.45553730 | 0.01140974 |
| 0.8 | 0.99128721 | 0.00991965 | 2.26709166 | 0.00049759 |

# ****Heat Bath Algorithm****

1: T(x→x') = ?

T(x→x') = 1

2: T(x'→x) = ?

T(x'→x) = 1

3. Write down the exp(-δV) where δV = V\_new - V\_old.

exp(-δV) = exp(-beta\* compute\_spin\_energy(i)), where i is the random spin chosen at the beginning. compute\_spin\_energy() will compute the energy change if flip the spin at position i.

4. Finally write down A(x→x')

A(x→x') = min(1, 1/(1+exp(beta\*(E(x')-E(x)))) = min(1, 1/(1+exp(beta\* δV)))

Before flipping spin i is x state, after flipping I is x' state.

The Heat Bath loop is in code, uploaded on compass.

The M2 results are shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Beta\*j | mean | stddev | Autocorr. time | Std error |
| 0.3 | 0.01537209 | 0.02204718 | 12.42646949 | 0.00258919 |
| 0.44069 | 0.55478307 | 0.17856909 | 84.42900875 | 0.05466253 |
| 0.8 | 0.99230241 | 0.00959339 | 3.26104750 | 0.00057715 |

# ****Cluster moves****

1: T(x→x') = ?

T(x→x') = 1-exp(-2\*beta\*J\*s^2) (for x and x' are the same state)

2: T(x'→x) = ?

T(x'→x) = 1-exp(-2\*beta\*J\*s^2) (for x and x' are the same state)

3. Write down the exp(-δV) where δV = V\_new - V\_old.

exp(-δV) = exp(-beta\* compute\_spin\_energy(cluster)), where cluster is all spin chosen at the beginning. compute\_spin\_energy(i) will compute the energy change if flip the spin at position i.

4. Finally write down A(x→x')

A(x→x') = 1

Before flipping spin i is x state, after flipping I is x' state.

The Cluster loop is in code, uploaded on compass.

The M2 results are shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Beta\*j | mean | stddev | Autocorr. time | Std error |
| 0.3 | 0.01522350 | 0.02152795 | 0.99271830 | 0.00071458 |
| 0.44069 | 0.07961972 | 0.07631726 | 0.91929867 | 0.00290607 |
| 0.8 | ~0.99 | ? | ? | ? |

NOTE: The beta\*j =0.8 condition is just too slow, unable to finish 1000 sweeps.

# ****Dénouement****

Compare your results from the three algorithms. Do they all give the same answer? (If not, shouldn't you worry?) Which algorithm has the least autocorrelation time for 1000 sweeps?

Answer:

Their anwers are not exactly the same, but they are pretty close to each other. There is no need to worry since this is MC algorithms, errors are expected.

Cluster move has the lowest auto correlation time since this is not a local move, neighbours has less effect on the center of certain spins. The other two methods are just local, thus have larger correlation with each other.