Applications to Statistical Mechanics

Alessandro Torcini

LPTM - Université de Cergy-Pontoise

Statistical mechanics

The Monte Carlo sampling methods find one very important application in Statistical mechanics.

Statistical mechanics allows to bridge the scales from microscopic dynamics to macroscopic observables

1. Microscopic dynamics tell us how the single atoms behaves due to the forces : Newton's Law or Hamilton's equations
2. Macroscopic observables are Energy, Pressure, Specific heat, . . . that are measured in experiments
3. Physical laws usually concern macrosocpic observables for systems made of many, many particles : Ohm's law, Diffusion Law, Gravity Law, . . .
4. How can I measure macroscopic observables starting from the dynamics of $\simeq 10^{23}$ atoms per cm^3 (Avogadro number $N_A = 6.022140 \times 10^{23}$) ?

Probabilistic concepts, we do not rely anymore on Newton's Law that is at the basis of the atoms' movement, but we employ probability distribution functions (PDF) describing the state of an entire system

The PDF depends on the Total Energy of the system

Phase Space

We have a system made of N particles, each particle i is characterized by a coordinate x_i (these can be a positions in space or angles) and by a conjugate momentum $p_i = mv_i$ (to simplify position and velocity v_i).
The state of the system is characterized by a point $\Gamma = (X, P)$ in the Phase space, where

$$X = (x_1, x_2, x_3 \ldots, x_N) \quad \text{and} \quad P = (p_1, p_2, p_3, \ldots, p_N)$$

1. the Phase Space is a $2N$-dimensional space.
2. A point Γ represents the Microscopic Configuration of the system
3. A system is described by an Hamiltonian $\mathcal{H}(x_1, x_2, \ldots, p_1, p_2, \ldots)$ which represents the energy of the system composed of kinetic $\mathcal{K}$ and potential $\mathcal{V}$ energies
4. Simple Example : For a classical system composed of N particles of mass m and interacting via two body interactions $v(x_i, x_j)$:

$$\mathcal{H} = \mathcal{K} + \mathcal{V} = \sum_{i=1}^{N} \frac{p_i^2}{2m} + \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} v(x_i, x_j) = \sum_{i=1}^{N} \frac{mv_i^2}{2} + \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} v(x_i, x_j)$$

Microscopic Approach

The dynamics of the particles x_i, p_i is determined by the Hamilton's equations

$$\frac{dx_i}{dt} = \frac{\partial \mathcal{H}}{\partial p_i} \tag{1}$$

$$\frac{dp_i}{dt} = -\frac{\partial \mathcal{H}}{\partial x_i} , \qquad i = 1, \ldots, N \tag{2}$$

if one knows the initial values at time $t = 0$ of all the positions and momenta, he can determine the time evolution of $x_i(t), p_i(t)$ from the solution (numerical integration) of the Hamilton's equation.

This is the approach of the so-called Molecular Dynamics to find the macroscopic properties of gas, liquids and solids.

One can arrive to simulate maybe $10^6 - 10^7$ atoms nowdays, but in a mol of the system there are already 10^{23} atoms, we are far from a detailed description of a real system.

For transport properties of gas, liquids sometimes $N = 500$ particles are sufficient to give an good estimate of the macroscopic properties, but this will not work for phase transitions, where there are long-range correlatons

Statistical Approach

The Hamiltonian can be used also to determine the PDF of observing at thermal equilibrium at a temperature T a certain microscopic state $\Gamma = (X, P)$
Boltzmann and Gibbs have shown that the PDF has the following expression

$$\boxed{p(\Gamma_l) = \frac{e^{-\beta\mathcal{H}}}{\mathcal{Z}} = \frac{e^{-\beta\mathcal{E}_l}}{\mathcal{Z}} \qquad \beta = \frac{1}{K_b T}}$$

with normalizaton factor given by

$$\mathcal{Z} = \int dx_1 \dots dx_n dp_1 \dots dp_n e^{-\beta\mathcal{H}(x_1,\dots,x_n,p_1,\dots,p_N)} = \sum_l e^{-\beta\mathcal{E}_l}$$

where $\mathcal{Z}$ is the Partition Function, a fundamental quantity in statistical mechanics.
$K_B = 1.38064852 \times 10^{-23} J K^{-1}$ is the Boltzmann Constant

Canonical Ensemble (N,V,T)

Free Energy, Internal Energy, Entropy

All these quantities can be derived from the Partition Function $\mathcal{Z}$

The Free Energy is given by

$$\boxed{F(N,V,T) = -K_B T \log \mathcal{Z} \qquad \mathcal{Z} = \mathrm{e}^{-\beta F}}$$

The Internal Energy is given by

$$\boxed{\mathcal{U}(N,V,T) = -\frac{\partial \log \mathcal{Z}}{\partial \beta} = \sum_l \mathcal{E}_l p(\Gamma_l)}$$

or equivalently

$$\boxed{\mathcal{U}(N,V,T) = -T^2 \left(\frac{\partial F/T}{\partial T}\right)_{V,N}}$$

Canonical Ensemble (N,V,T)

Free Energy, Internal Energy, Entropy

The Entropy is given by

$$\boxed{S(N,V,T) = -K_B \sum_l p(\Gamma_l) \log p(\Gamma_l) = -\frac{\partial F}{\partial T} = -K_B \beta^2 \frac{\partial \log \mathcal{Z}/\beta}{\partial \beta}}$$

It is easy to verify that

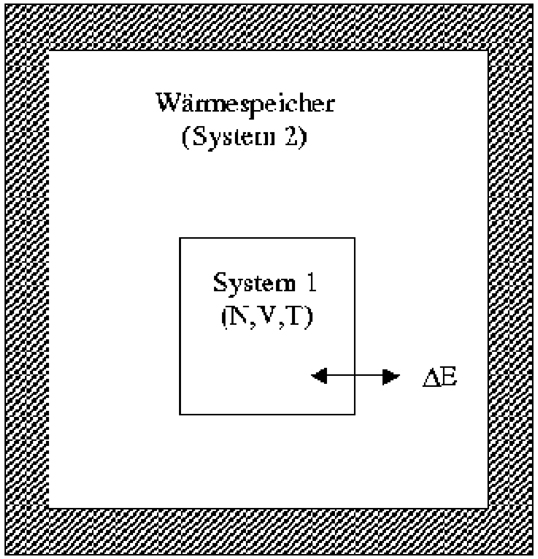
$$\boxed{F(T) = \mathcal{U}(T) - TS(T)}$$

since

$$\mathcal{U} - TS = -T^2 \left(\frac{\partial F/T}{\partial T} \right) + T \frac{\partial F}{\partial T} = T \left[\frac{\partial F}{\partial T} + TF \frac{1}{T^2} - \frac{T}{T} \frac{\partial F}{\partial T} \right] = F$$

Canonical Ensemble (N,V,T)

Usually real systems are described within the so called Canonical Ensemble



The system 1 contains N particles in a volume V at a temperature T, the temperature is maintaned constant thanks to energy exchange with a larger system containing it : the thermostat (or reservoir).
The canonical ensemble is therefore characterized by the macroscopic variables (N, V, T)

Each macroscopic state (N, V, T) can correspond to many different microscopic states/configurations $\Gamma_l = (X_l, P_l)$, each characterized by a different energy

$$\mathcal{E}_l = \mathcal{H}(X_l, P_l)$$

The energy of the state $\mathcal{E}_l$ is its Hamiltonian $\mathcal{H}(X_l, P_l)$

Canonical Ensemble (N,V,T)

The probability to observe a certain state/configuration fixed (N, V, T) is given by

$$\boxed{p(\Gamma_l) = \frac{\mathrm{e}^{-\beta \mathcal{E}_l}}{\mathcal{Z}} \qquad \mathcal{Z} = \sum_l \mathrm{e}^{-\beta \mathcal{E}_l}}$$

The average of a macroscopic observable A of the system is therefore given by

$$\boxed{\langle A(N,V,T) \rangle = \sum_l A(\Gamma_l) p(\Gamma_l)}$$

and its variance is

$$\boxed{\sigma_A^2 = \sum_l A^2(\Gamma_l) p(\Gamma_l) - \left(\sum_l A(\Gamma_l) p(\Gamma_l) \right)^2}$$

Canonical Ensemble (N,V,T)

One of the most important observable is the average internal energy of the system

$$\boxed{\mathcal{U}(N,V,T) = \sum_l \mathcal{E}_l p(\Gamma_l) = \frac{\sum_l \mathcal{E}_l \mathrm{e}^{-\beta \mathcal{E}_l}}{\mathcal{Z}} = \langle \mathcal{H} \rangle}$$

and the variance of the Hamiltonian, that is the specific heat at constant volume

$$\boxed{C_V = \left(\frac{\partial \mathcal{U}}{\partial T}\right)_{V,N} = \frac{K_B}{T^2}\left[\sum_l \mathcal{E}_l^2 p(\Gamma_l) - \left(\sum_l \mathcal{E}_l p(\Gamma_l)\right)^2\right] = \frac{K_B}{T^2}[\langle \mathcal{H}^2 \rangle - \langle \mathcal{H} \rangle^2]}$$

Demonstration

$$C_V = -\frac{\partial \beta}{\partial T}\frac{\sum_l \mathcal{E}_l^2 \mathrm{e}^{-\beta \mathcal{E}_l}}{\mathcal{Z}} - \frac{\sum_l \mathcal{E}_l \mathrm{e}^{-\beta \mathcal{E}_l}}{\mathcal{Z}^2}\frac{\partial \mathcal{Z}}{\partial T}$$

$$C_V = -\frac{\partial \beta}{\partial T}\frac{\sum_l \mathcal{E}_l^2 \mathrm{e}^{-\beta \mathcal{E}_l}}{\mathcal{Z}} + \frac{\sum_l \mathcal{E}_l \mathrm{e}^{-\beta \mathcal{E}_l}}{\mathcal{Z}^2}\left(\sum_l \mathcal{E}_l \mathrm{e}^{-\beta \mathcal{E}_l}\right)\frac{\partial \beta}{\partial T}$$

$$C_V = \frac{K_B}{T^2}\left[\frac{\sum_l \mathcal{E}_l^2 \mathrm{e}^{-\beta \mathcal{E}_l}}{\mathcal{Z}} - \frac{(\sum_l \mathcal{E}_l \mathrm{e}^{-\beta \mathcal{E}_l})^2}{\mathcal{Z}^2}\right] \quad \blacksquare$$

Metropolis' Algorithm

1. Therefore Statistical Mechanics of equilibrium seems reduced to the estimation of averages
2. The problem is now to generate configurations Γ_l distributed accordingly to the PDF $p(\Gamma_l)$
3. This is what we are going to do with Montecarlo Methods, and in particular with the Metropolis' algorithm
4. The Metropolis' algorithm accept a modification from a configuration Γ_i to Γ_j according to the transition PDF

$$T_\lambda(\Gamma_i \to \Gamma_j) = min\left[1, \frac{p(\Gamma_j)}{p(\Gamma_i)}\right] \qquad \text{where} \qquad p(\Gamma_i) = \frac{\mathrm{e}^{-\beta\mathcal{E}_i}}{\mathcal{Z}}$$

5. The ratio becomes

$$\frac{p(\Gamma_j)}{p(\Gamma_i)} = \frac{\frac{\mathrm{e}^{-\beta\mathcal{E}_j}}{\mathcal{Z}}}{\frac{\mathrm{e}^{-\beta\mathcal{E}_i}}{\mathcal{Z}}} = \mathrm{e}^{-\beta\Delta\mathcal{E}}$$

6. $\Delta\mathcal{E} = \mathcal{E}_j - \mathcal{E}_i$ is the energy variation associated to the proposed change of state Γ_i to Γ_j

Metropolis' Algorithm

The acceptance probablity for a modification of a microscopic configuration will be therefore

$$T_\lambda(\Gamma_i \rightarrow \Gamma_j) = min\left[1, \mathrm{e}^{-\beta\Delta\mathcal{E}}\right]$$

How do we proceed ?

1. We have a microscopic configuration of the particles of our system $\Gamma_i = (X_i, P_i)$ at time t of energy $\mathcal{E}_i$
2. We modify for example the position of one particle in the system at time $t+1$ and we get a new trial configuraton $\Gamma_j = (X_j, P_j)$ at time $t+1$ of energy $\mathcal{E}_j$
3. If $\mathcal{E}_j < \mathcal{E}_i$ the system energy is reduced and the new configuration accepted
4. If $\mathcal{E}_j > \mathcal{E}_i$ the new configuration Γ_j is accepted with a probability $\mathrm{e}^{-\beta\Delta\mathcal{E}}$

Equilibrium State

An equilibrium configuration corresponds to a minimum value of the Helmholtz's Free Energy $F = U - TS$ for a system that can just exchange heat with the reservoir (mechanically isolated)

Demonstration - Thermodynamics

1. For an isothermal transformation $T = constant$ (canonical ensemble) from state A to B from the second law of thermodynamics, one has :

$$\int_A^B \frac{\delta Q}{T} \leq S(B) - S(A) = \Delta S$$

2. Since $T = constant$ then

$$\frac{\Delta Q}{T} \leq \Delta S$$

   where ΔQ is the heat absorbed by the system during the transformation ;

3. By using the first law of thermodynamics $\Delta W = \Delta Q - \Delta U$ one gets

$$\Delta W \leq T\Delta S - \Delta U \rightarrow \Delta W \leq -\Delta F$$

   where ΔW is the work done by the system

Equilibrium State

1. Thus, the equilibrium of an isothermal system which does not perform work $\Delta W = 0$ (mechanically isolated) always looks for a minimum of Helmholtz Free Energy

$$\Delta F = F(B) - F(A) \leq 0 \rightarrow F(A) \geq F(B)$$

2. Therefore, irreversible processes happen spontaneously, until the minimum is reached

$$\Delta F = 0 \rightarrow F = F_{min}$$

3. The Free Energy takes in account the conflicting role of entropic effects, which tends to increase S, and energetics effects, that tend to reduce U
   (a) At $T = 0$ the configuration of equilibrium corresponds to a minimum of U
   (b) At $T \rightarrow \infty$ the equilibrium corresponds to a maximum of S

Physical Meaning of the Metropolis’ Algorithm

The equilibrium configurations minimize the Helmholtz’s free energy

$$F = \mathcal{U} - TS$$

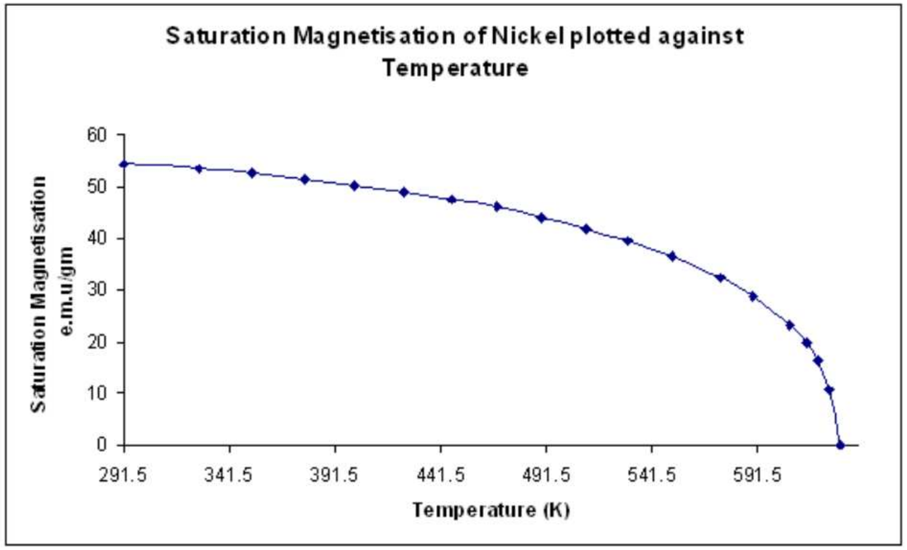
which is a balance between

1. the internal energy $\mathcal{U}$, which tends to a minimum
2. the entropy S which tends to a maximum

In the Metropolis’ Algorithm the balance is achieved by

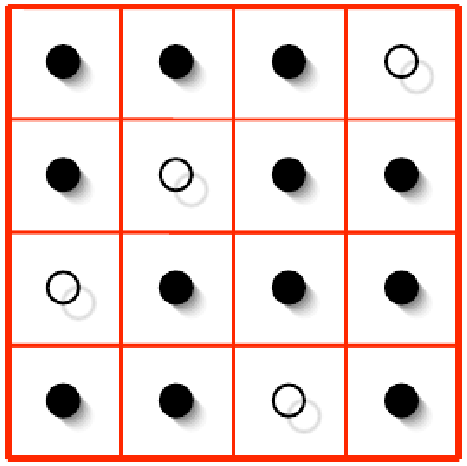
1. accepting all proposal $\Gamma_i \to \Gamma_j$ for which energy decreases ($\Delta\mathcal{E} \leq 0$)
2. accepting those proposal for which the energy increases $\Delta\mathcal{E} > 0$ with a probability $\mathrm{e}^{-\frac{\Delta\mathcal{E}}{K_B T}}$ that depends on the temperature T
   - (a) For $T \to 0$ the propability to accept proposal that increase the energy goes to zero
   - (b) For $T \to \infty$ such probability becomes one, all the proposal are accepted, if they increase or decrease energy, since energy becomes irrelevant

The paramagnet to ferromagnet transition


Saturation Magnetisation of Nickel plotted against Temperature
Saturation Magnetisation e.m.u/gm
0
10
20
30
40
50
60
291.5
341.5
391.5
441.5
491.5
541.5
591.5
Temperature (K)

1. The microscopic magnetic moments s_i associated to each atom interact among them in a magnetic material (e.g Nickel) .
2. If two microscopic magnetic moments are parallel in adjacent (nearby) atoms the energy is lower than if they are anti-parallel
3. The thermal fluctuations prevent the alignement
4. Therefore, for sufficiently low temperature energy decrease favours the alignement of all the atomic moments giving rise to a macroscopic magnetization,
5. Below a temperature T_c the systems is a ferromagnet, with a finite magnetization, above it is a paramagnet with a zero magnetization : at $T = T_c$ we have a phase transition

The Ising Model



Ising model : the simplest model for a magnetic material

1. A two dimensional regular lattice with N sites
2. In each site (i,j) a variable $s_k = \pm 1$ (The spin)
3. A configuration is $\Gamma = (s_1, s_2, s_3, \ldots, s_N)$
4. 2^N configurations
5. The magnetization is given by $M = \sum_k s_k$

The Hamiltonian of an Ising model (or the energy of the configuration Γ) is given by

$$\mathcal{H}(\Gamma) = -J \sum_{k,h}^{*} s_k s_h - H \sum_k s_k$$

where J is the interaction term and H is an external applied magnetic field

$*$ The sum $\sum_{k,h}$ is restricted to the nearest neighbours

The Ising Model

$$\mathcal{H}(\Gamma) = -J \sum_{k,h}^{*} s_k s_h - H \sum_k s_k$$

1. The interaction energy among 2 spins is given by

$$-J s_k s_h = \begin{cases} -J & \text{if} \quad s_k = s_h \qquad \text{(parallel)} \\ J & \text{if} \quad s_k = -s_h \qquad \text{(antiparallel)} \end{cases}$$

2. At $T = 0$ K the system has no thermal fluctuations, all the spins will be parallel, since the minimal energy is achieved for parallel spins.
3. The minimal energy state is called ground state and it can have $s_i = +1 \;\; \forall i$ or equivalently $s_i = -1 \;\; \forall i$
4. A system is ferromagnetic if the average magnetizaton $M = \sum_k s_k \neq 0$
5. At $T = 0$ K the magnetization $M = \pm N$ – At $T \to \infty$ the spins are pointing 50% up and 50% down therefore $M = 0$ (paramagnetic phase)
6. At some critical temperature $T_c > 0$ the system passes from ferromagnetic to paramagnetic, we should find such temperature

The Ising Model

$$\mathcal{H}(\Gamma) = -J \sum_{k,h}^{*} s_k s_h - H \sum_k s_k$$

1. The presence even of a very small field H breaks the symmetry

   (a) If $H > 0$ the ground state will have $M = +N$ $(s_k = +1\ \forall k)$

   (b) If $H < 0$ the ground state will have $M = -N$ $(s_k = -1\ \forall k)$

2. Lars Onsager in 1944 has solved analytically the Ising Model in 2d with zero field $H = 0$ (Nobel Prize for Chemistry in 1968)

$$M/N = \begin{cases} 0 & \text{if} \quad T > T_c \\ (1 - [\sinh(2J/K_B T)]^{-4})^{1/8} & \text{if} \quad T \geq T_c \end{cases}$$

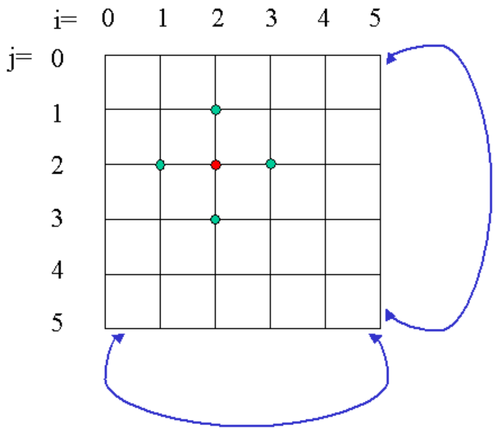
$$T_c = \frac{J}{K_B} \frac{2}{\log 1 + \sqrt{2}} \simeq 2.2691853 \frac{J}{K_B}$$

We will use the Monte Carlo method to solve the very difficult problem

The Metropolis' Algorithm

Ising Model with $H = 0$

$$\mathcal{E}(\Gamma) = -J \sum_{k,h}^{*} s_k s_h \qquad \Gamma = (s_1, s_2, \dots, s_N) \qquad s_k = \pm 1$$



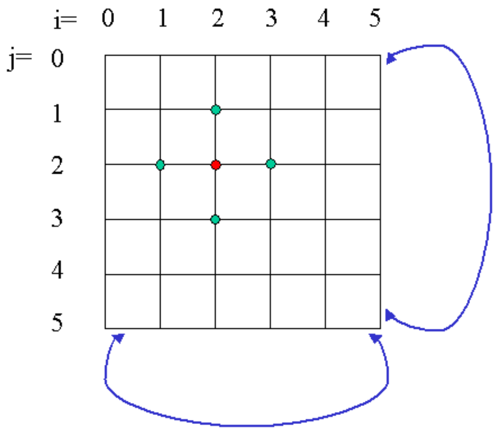
Montecarlo Strategy

1. We select randomly a site (i, j) in the lattice with spin s_k
2. We flip its spin $s_k \rightarrow -s_k$ and we get a new trial configuration $\Gamma' = (s_1, s_2, \dots, -s_k, \dots, s_N)$
3. We estimate the new energy $\mathcal{E}(\Gamma')$ and the energy variation $\Delta\mathcal{E} = \mathcal{E}(\Gamma') - \mathcal{E}(\Gamma)$
4. If $\Delta\mathcal{E} < 0$ the new configuration Γ' is accepted
5. If $\Delta\mathcal{E} > 0$, we extract a random number $0 < r < 1$
   (a) if $r < \mathrm{e}^{-\frac{\Delta\mathcal{E}}{K_B T}}$ the configuration is accepted
   (b) otherwise, the spin s_k is not flipped

The Metropolis' Algorithm

Ising Model with $H = 0$

$$\mathcal{E}(\Gamma) = -J \sum_{k,h}^{*} s_k s_h \qquad \Gamma = (s_1, s_2, \ldots, s_N) \qquad s_k = \pm 1$$



Montecarlo Strategy II

We should now estimate the energy variaton $\Delta\mathcal{E}$ due to one spin flip $s_k \to -s_k$

1. $N - 1$ spins remain unchanged
2. the energy variation is due only to the terms in the sum in which s_k appears
3. In two dimension each spin has 4 neighbours D, therefore 4 terms change in the sum

$$\Delta\mathcal{E} = 2Js_k \sum_{\mu \in D} s_\mu = 2J \begin{cases} -4 \\ -2 \\ 0 \\ 2 \\ 4 \end{cases}$$

Random Generators

In order to perform MonteCarlo simulations we need good random generators, to generate uniform numbers between $[0, 1)$ we will use

`np.random.random()`

```
>>> np.random.random()
0.47108547995356098
>>> type(np.random.random())
<type 'float'>
>>> np.random.random((5,))
array([ 0.30220482,  0.86820401,  0.1654503 ,  0.11659149,  0.54323428])
```

Three-by-two array of random numbers from [-5, 0) :

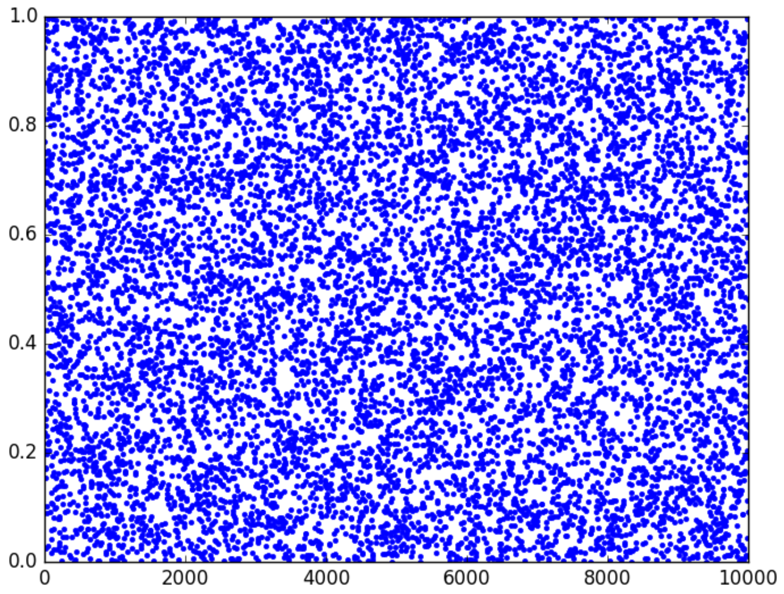
```
>>> 5 * np.random.random((3, 2)) - 5
array([[-3.99149989, -0.52338984],
       [-2.99091858, -0.79479508],
       [-1.23204345, -1.75224494]])
```

Is it a good random generator ? Yes

Random Generators

```
import numpy as np
import matplotlib.pyplot as plt

plt.plot(range(10000), np.random.random(10000), '.')
plt.savefig('numpy-random.png')
plt.show()
```

It gives uniformly distributed random numbers



Random Generators

Initial configuration of the Spins

As a first step we should inizialize configuration of the N spins on the lattice of $N = L \times L$ sites with random values $s_i = +1$ or $s_i = -1$

We will use the integer random generator

`(2*np.random.random_integers(1,size=(L,L))-1)`

```
import matplotlib.pyplot as plt
import numpy as np

def init_lattice(l):
    # Create a nxn lattice with random spin configuration
    lattice = (2*np.random.random_integers(0,1,size=(l,l))-1)
    return lattice

l=100   # lenght of the lattice
n= l * l  # number of sites
lattice = init_lattice(l)

plt.matshow(lattice)
plt.show()
```

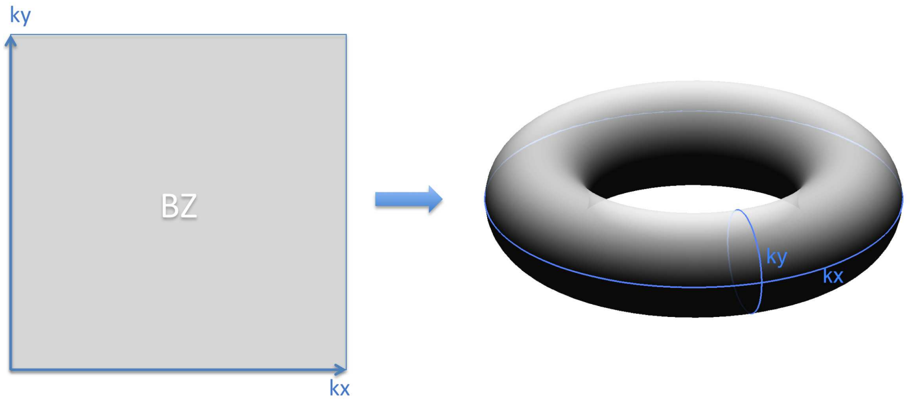
Periodic Boundary Conditions (PBCs)

The indeces of the spins on the $2d$ lattice are

$$i = 0, \dots, L-1 \qquad j = 0 \dots, L-1$$

Which are the neighbours of the spin (i, j) ?

1. 4 neighbours : $(i, j-1)$ $(i, j+1)$ $(i-1, j)$ and $(i+1, j)$
2. If $i = L-1$ One neighbour is out of the lattice (L, j)
3. Periodic Boundary Conditions $L \to 0$ the neighbour is $(0, j)$
4. The two dimensional lattice becomes a Torus



Periodic Boundary Conditions (PBCs)

How to implement PBCs in Python ?

We use the function Modulus % already introduced in the first part of the course

We want to estimate the sum S_D of the four spin neighbours of the spin s_k with random position (i, j), which enters in the estimaton of the nergy variation

$$\Delta\mathcal{E} = 2Js_k \sum_{\mu \in D} s_\mu = 2Js_k S_D$$

```
i = np.random.randint(l)
j = np.random.randint(l)

# Periodic Boundary Conditions
SD = lattice[(i - 1) % l, j] + lattice[(i + 1) % l, j] + \
     lattice[i, (j - 1) % l] + lattice[i, (j + 1) % l]
```

The spin s_k corresponds to $lattice[i, j]$

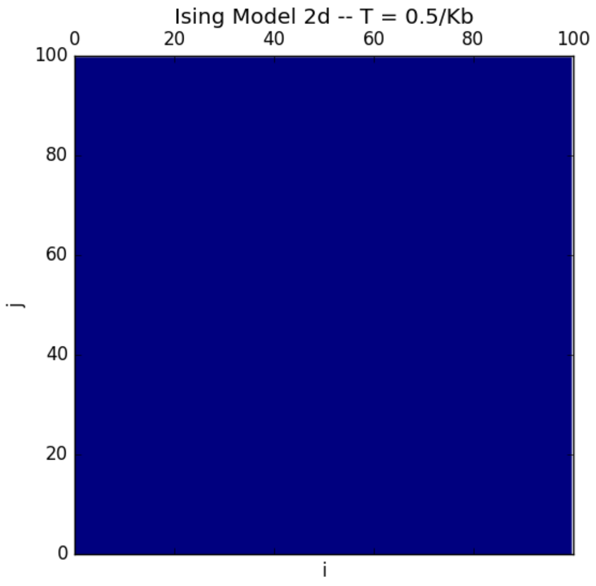
The Montecarlo Algorithm for the Ising 2d

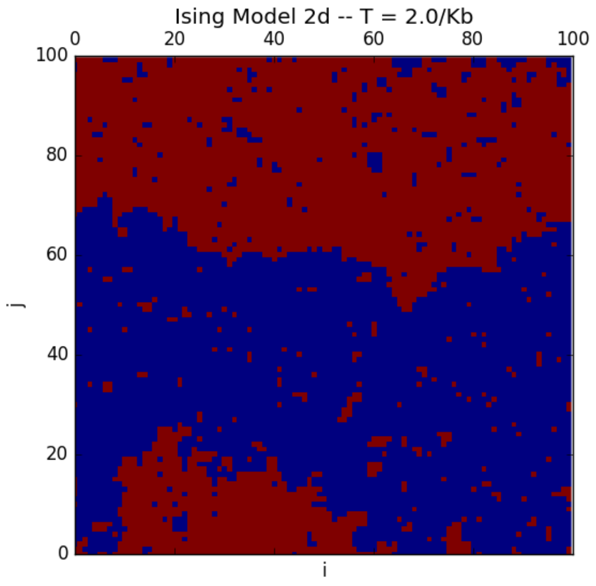
```
import matplotlib.pyplot as plt
import numpy as np
def init_lattice(l): # Create a nxn lattice with random spin configuration
    lattice = (2*np.random.random_integers(0,1,size=(l,l))-1)
    return lattice

def deltaE(i,j): #Energy difference for a spin flip  - PBCs
    SD = lattice[(i - 1) % l, j] + lattice[(i + 1) % l, j] + \
         lattice[i, (j - 1) % l] + lattice[i, (j + 1) % l]
    return 2*J*lattice[i,j]*SD

def move(): # a MC move
    i,j = np.random.randint(l), np.random.randint(l)
    dE = deltaE(i, j)
    if dE < 0:
        lattice[i, j] = -lattice[i, j]
        return
    if np.random.random() < np.exp(-dE*beta):
        lattice[i, j] = -lattice[i, j]
        return
    return
```

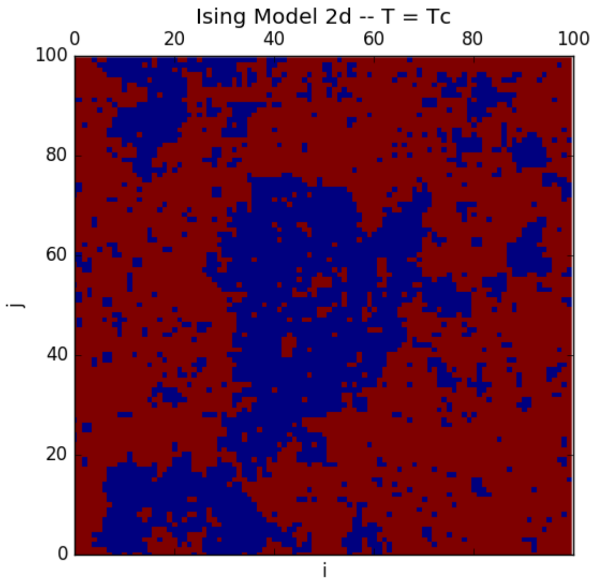
The Montecarlo Algorithm for the Ising 2d

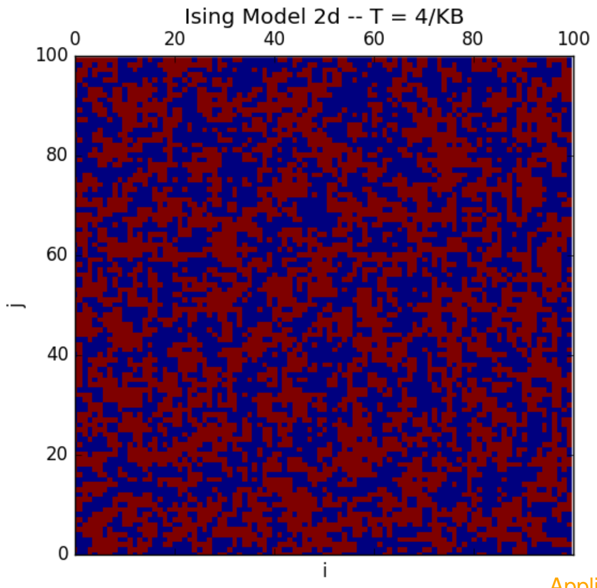
```
global lattice,J,beta,l
J, T, K =1, 4, 3 # coupling, temperature and parameter for the MC
l=100   # lenght of the lattice
n= l * l  # number of sites
beta=1/T
lattice = init_lattice(l)  # random initial conditions

for t in range(0,1000): # thermalization phase
    for mc in range(0,n): # 1 MC steps is n moves
        move()

for t in range(0,100):# generate a configuration
    for mc in range(0,n*K): # K MC steps is n*K moves
        # the data are more independent
        move()

plt.matshow(lattice)  # plot the configuration
plt.xlim(0,l)
plt.ylim(0,l)
plt.savefig('Ising2d.T4_N100.png')
plt.show()
```

The Configurations for the Ising 2d









Magnetization

The average magnetization per particle is defined as

$$m(T) = \frac{M(T)}{N} = \langle \frac{1}{N} \sum_k s_k \rangle$$

The quantity m is obtained by averaging $\frac{1}{N} \sum_k s_k$ over many different configurations during the dynamics of the Ising model

In proximity of the critical temperature $T_c \simeq 2.2691853 J/K_B$ the magnetization behaves as

$$\boxed{m(T) \propto \left(\frac{T_c - T}{T_c} \right)^{1/8} \qquad T \leq T_c}$$

obviously above the transition

$$m(T) = 0 \qquad \text{for} \qquad T > T_c$$

Magnetization

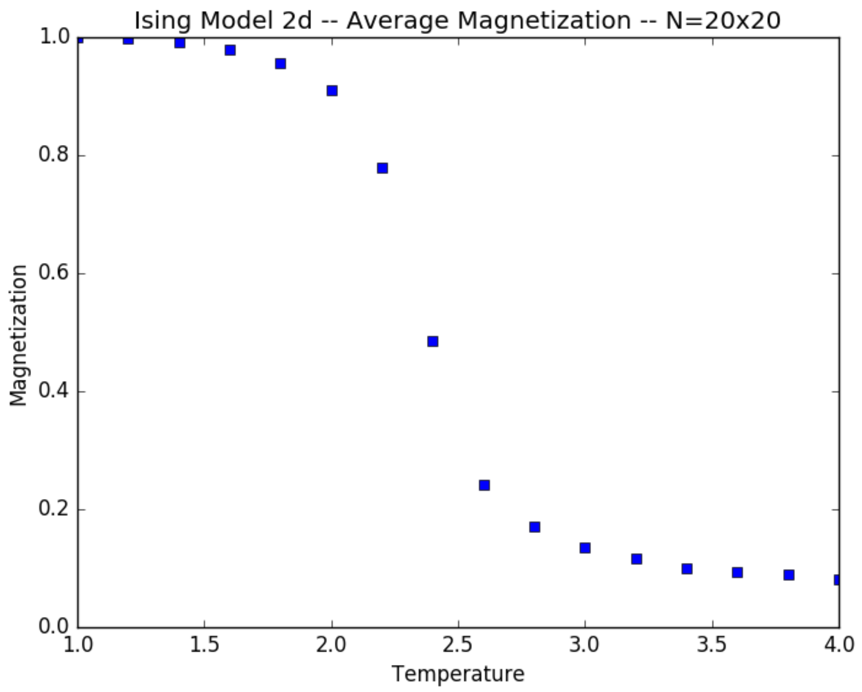
```
def magnetization(l):
    # estimate the instantaneous magnetization
    mm=0.
    for i in range (0,l):
        for j in range(0,l):
            mm=mm+lattice[i,j]
    mm=mm/(l*l)
    return mm

global lattice,J,beta,l
J=1
l=20   # lenght of the lattice
n= l * l  # number of sites
K=1      # parameter for the MC
beta=1./3.
lattice = init_lattice(l)  # random initial conditions
for t in range(0,1000):  # thermalization phase
   for mc in range(0,n): # 1 MC steps is n moves
       move()
print('thermalization finished')
```

Magnetization

```
T=0.
vt=[]
mt=[]
for t in range (1,21):
    T=T+0.2
    beta=1./T  # K_B =1

    vt.append(T)
    magnet=0.  # average magnetization
    nit = 10000  # number of iterations
    print("temperature",T)
    for t in range(0,nit):
        for mc in range(0,n*K): # K MC steps is n*K moves
            # the data are more independent
            move()
        magnet += abs(magnetization(l))
    magnet=magnet/nit
    mt.append(magnet)
```

Magnetization

The simulations are really long



How to write and read a file

In Python in order to open a file one should specify if one want to read, write etc

1. 'r' : use for reading
2. 'w' : use for writing
3. 'x' : use for creating and writing to a new file
4. 'a' : use for appending to a file
5. 'r+' : use for reading and writing to the same file

```
ff = open('name.dat','w') # open the file to write
ff = open('name.dat','r') # open the file to read
```

To write/read to/from a file

```
ff.write()
ff.read()
```

to close the file

```
ff.close()
```

How to write and read a file

The file "days.txt" contais the names of the days of the week

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

How can we read this file in Python ?

How can we write a file ?

How to write and read a file

```
path = 'days.txt'
days_file = open(path,'r')
days = days_file.read()



new_path = 'new_days.txt'
new_days = open(new_path,'w')

title = 'Days of the Week\n'
new_days.write(title)
print(title)

new_days.write(days)
print(days)

days_file.close()
new_days.close()
```

How to write and read a file

How can we write in a file numbers ordered in rows and separated by white spaces, like

$$\begin{array}{ccc} 31 & 22 & 18 \\ 13 & 14 & 15 \end{array}$$

A possibility is the following

```
ff = open('data.dat','w') # open the file to write
a=31
b=22
c=18
d=13
e=14
f=15
ff.write(str(a) + " " + str(b) + " "+ str(c) + "\n")
ff.write(str(d) + " " + str(e) + " "+ str(f) + "\n")
    # write in the file

ff.close()    # close the file
```

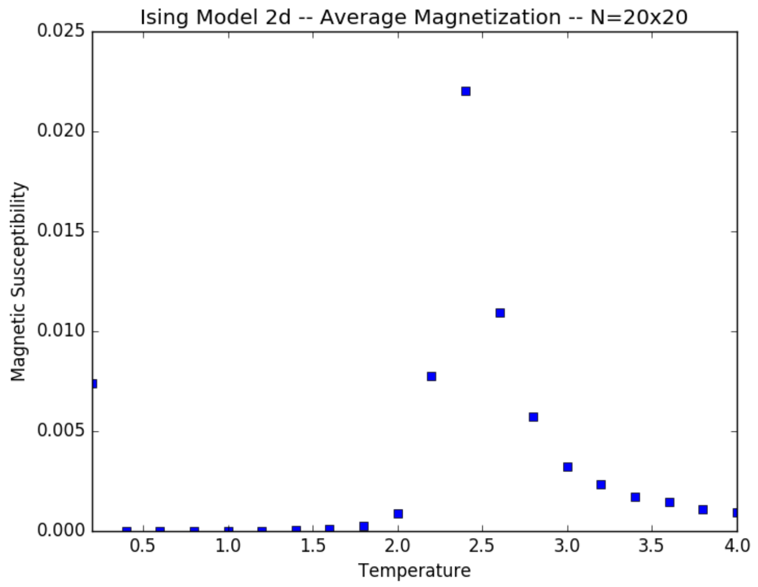
Magnetic Susceptibility

The variance of the magnetization $m(T)$ measures its fluctuations and it is called Magnetic Susceptibility

$$\boxed{\chi = \frac{1}{K_B T}\left[\langle m^2 \rangle - \langle m \rangle^2\right]}$$

The magnetic susceptibility diverges at the critical temperature T_c as

$$\boxed{\chi = \propto \frac{|T_c - T|}{T_c}^{7/4}}$$



Magnetic Susceptibility

```
ff = open('suscept.L20.dat','w') # open the file to write
T=0.
vt=[]
mt=[]
for t in range (1,21):
    T=T+0.2
    beta=1./T  # K_B =1

    vt.append(T)
    magnet=0.  # average magnetization
    magnet2 = 0.  # average square of the magnetization
    nit = 10000  # number of iterations
    print("temperature",T)
```

Magnetic Susceptibilty

```
    for t in range(0,nit):
        for mc in range(0,n*K): # K MC steps is n*K moves
            # the data are more independent
            move()
        magnet += abs(magnetization(l))
        magnet2 += magnetization(l)**2
    magnet=magnet/nit
    magnet2=magnet2/nit
    magnet2=(magnet2-magnet*magnet)/T
    mt.append(magnet2)
    ff.write(str(T) + " " + str(magnet2) + " "+ str(magnet) + "\n")
    # write in the file

ff.close()   # close the file
```