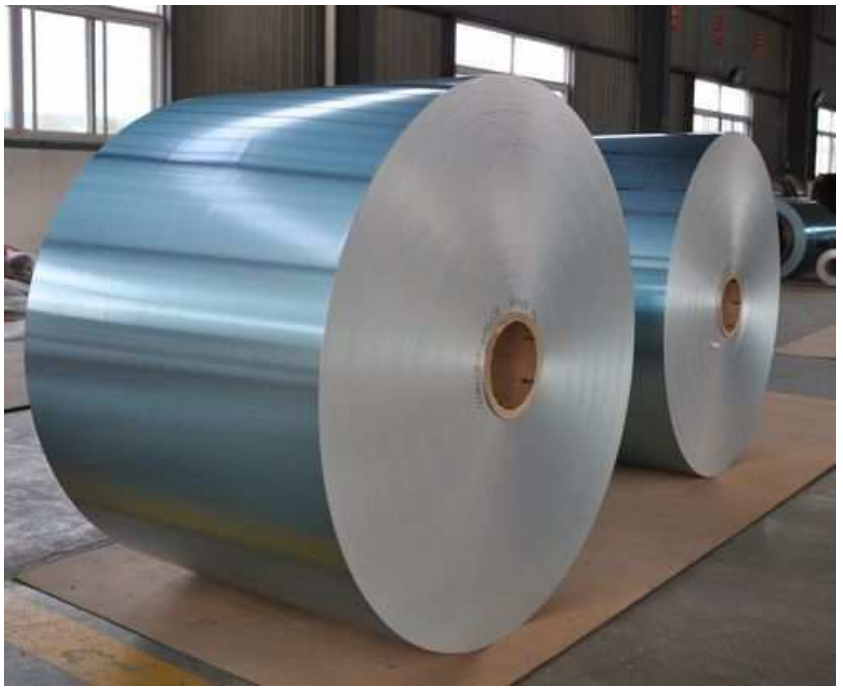
Applications to Statistical Mechanics II

Alessandro Torcini

LPTM - Université de Cergy-Pontoise

Binary Alloy

Usually a material like Aluminium is never used in a pure form to create material, usually Aluminium is fused with other materials (like Zinc) to create an Alloy with better resistance properties.



Zn-Al alloy Phase Diagram

The alloys have usual very complicated phase diagrams

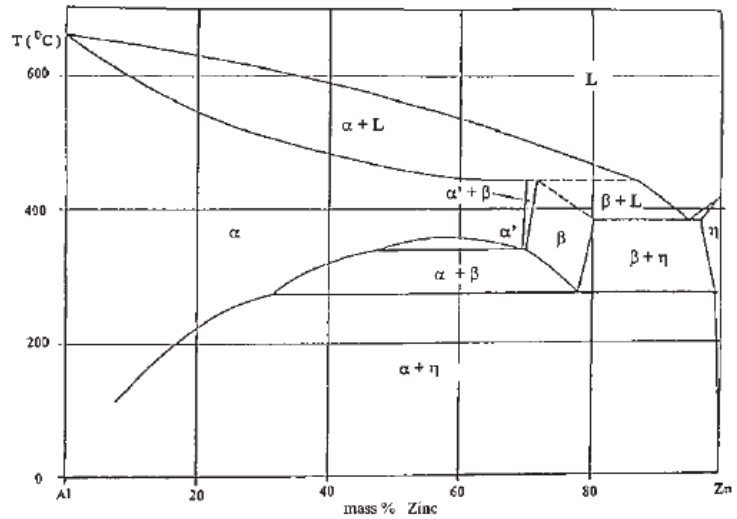
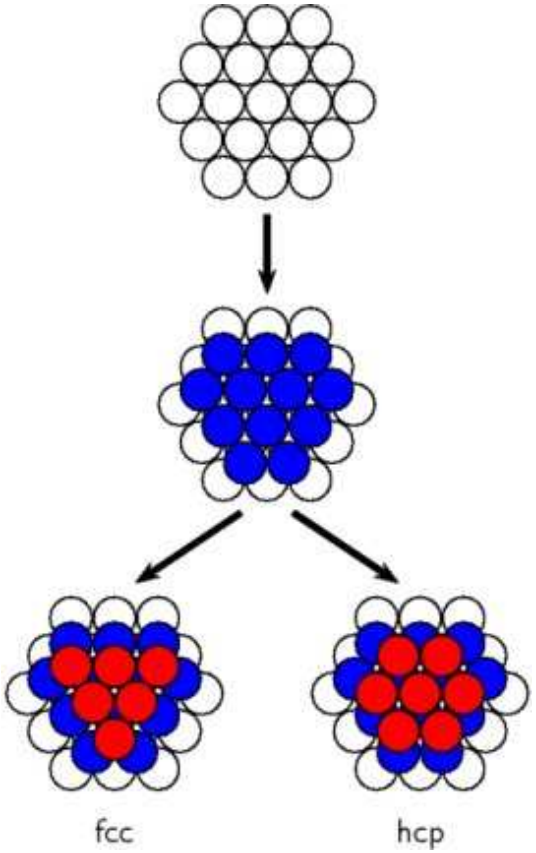


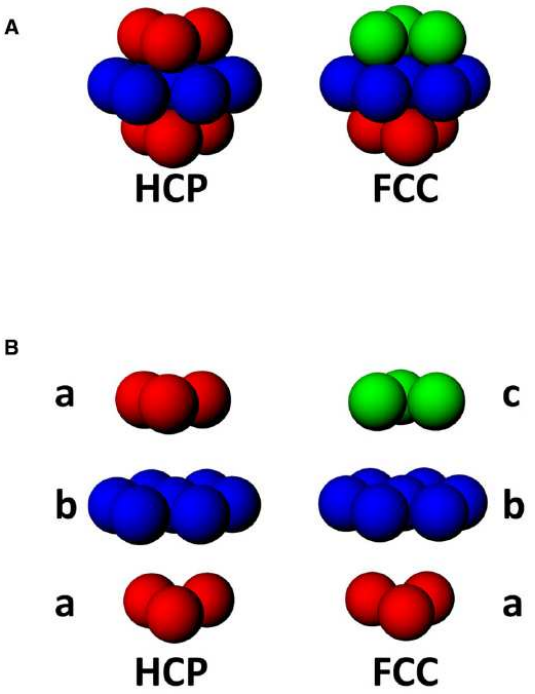
Fig. 1 Phase diagram of binary Zn-Al alloy.

1. L = liquid
2. α - a solid phase (FCC) rich of Al
3. β - a solid phase (FCC) rich of Zn
4. η - a solid phase with different symmetry (HCP) rich of Zn

The low temperature hexagonal close-packed (HCP) phase
The high temperature face-centered cubic (FCC) phase

FCC versus HCP configurations





Zn-Al alloy Phase Diagram

The configuration depends on how the melt is cooled

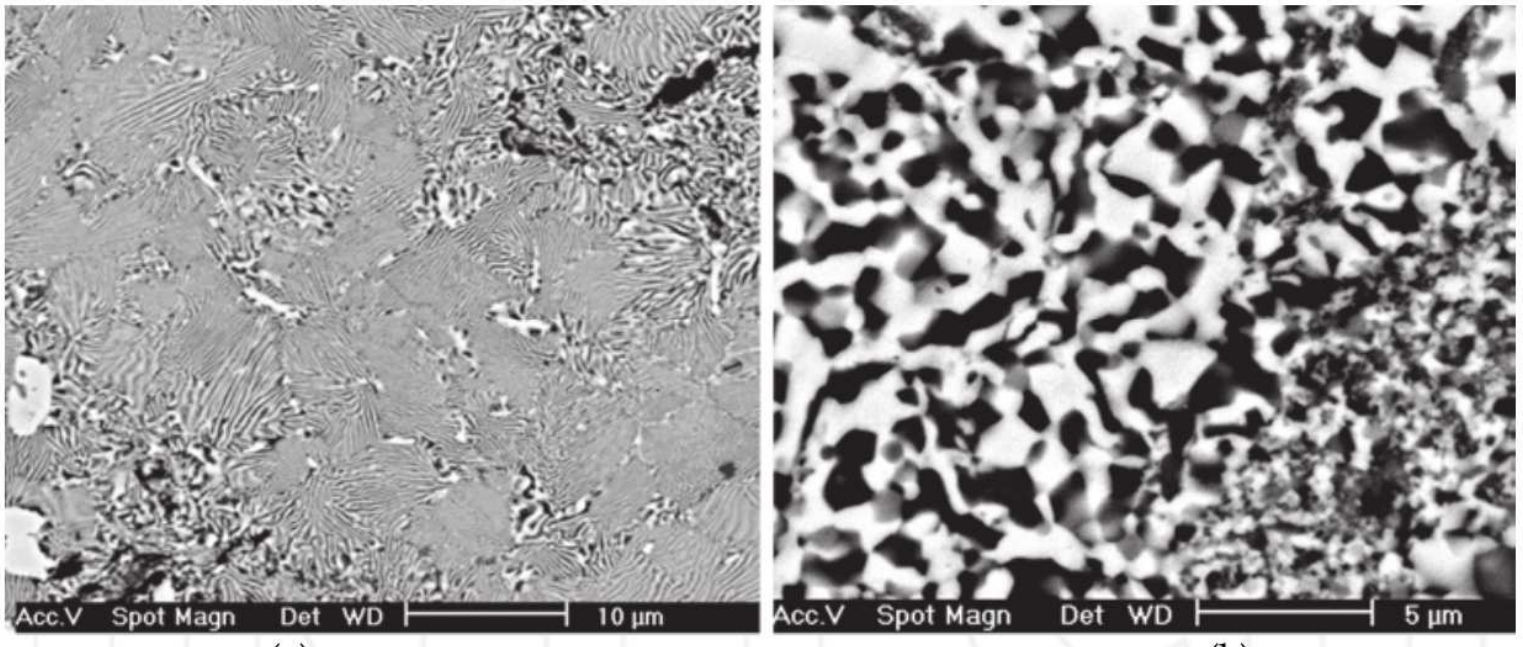


Fig. 2. Microstructures in Zn-Al-Cu. (a) Slowly cooled alloy. (b) Quenched alloy

1. slowly cooled
2. or quenched - fast cooled

A simple model for an alloy

In an alloy made of atoms A and B

1. We can use the Ising Model in 2d

$$\mathcal{H} = -J \sum_{i,j}^{*} s_i s_j \qquad where \qquad s_i = 1(A) \qquad s_i = -1(B)$$

   where the sum is restricted $*$ to nearest neighbours spins

2. In an alloy the number of atoms of one species or of the other cannot change, $N_A = const$ and $N_B = const$
3. We cannot flip the spins, this is not allowed because he would change N_A and N_B
4. Therefore the allowed moves are those where two neighbours spins exchange of place

The Kawasaki model for binary alloy separation

The moves in this case correspond to flipping two nearest neighbours spins at the same time and to estimate the energy variation $\Delta\mathcal{E}$ associated to this move and to employ the Metropolis algorithm to decide if the move is accettable or not.
The Kawasaki model for a binary alloy is

$$\mathcal{H} = -J \sum_{i,j}^{*} s_i s_j \qquad where \qquad s_i = 1(A) \qquad s_i = -1(B)$$

The Metropolis Algorithm

1. Choose randomly the spin s_n with coordinates $i \in [0, L-1]$ and $j \in [0, L-1]$
2. Choose randomly one of the four neighbours spin s_m
   (a) If the two spins are parallel restart
   (b) if they are anti-parallel estimate the energy variation $\Delta\mathcal{E}$ due to the two spin flipping $s_n = -s_n$ and $s_m = -s_m$
3. Continue with the Metropolis algorithm

Energy variation $\Delta\mathcal{E}$

The energy variation $\Delta\mathcal{E}$ for the contemporary flip of two adjacent spins s_n, s_m is given by

if the spins s_n, s_m do not interact

1. Energy variation due to the flip of spin s_n is

$$\Delta\mathcal{E}_n = 2Js_n \sum_{\mu\in D_n} s_\mu = 2Js_n S_D(n)$$

   where $S_D(n)$ is the sum of the spins that are neighbours of s_n

2. Energy variation due to the flip of spin s_m is

$$\Delta\mathcal{E}_m = 2Js_m \sum_{\mu\in D_m} s_\mu = 2Js_m S_D(m)$$

   where $S_D(m)$ is the sum of the spins that are neighbours of s_m

3. the total energy variation is

$$\Delta\mathcal{E} = \Delta\mathcal{E}_m + \Delta\mathcal{E}_n = 2Js_n[S_S(n) - S_D(m)]$$

since $s_m = -s_n$

Energy variation $\Delta\mathcal{E}$

But the spins s_n, s_m do interact, they are nearest neighbours

If we exchange two spins the interaction energy among them does not change, this contribution has been counted two times and it should be subtracted therefore

$$\Delta\mathcal{E} = 2Js_n[S_S(n) - S_D(m)] - 4Js_n s_m$$

But the two spins are anti-parallel therefore the energy variation is

$$\boxed{\Delta\mathcal{E} = 2Js_n[S_S(n) - S_D(m)] + 4J}$$

How to choose randomly the spin and its nearest neighbour to flip ?

```
i,j= np.random.randint(l), np.random.randint(l)
inn = np.random.random_integers(0,1)
iu = 2*np.random.random_integers(0,1)-1
if inn != 0:
     i1,j1=i+iu,j
else:
     i1,j1=i,j+iu
i1,j1 =i1 %l,j1 %l
if lattice[i, j]*lattice[i1, j1] <0 :
     move(i,j,i1,j1)
```

The Metropolis' Move

```
def deltaE(i,j,i1,j1):
    '''Energy difference for a spin exchange of 2 neighbours'''
    # periodic  boundary condtions
    SD = lattice[(i - 1) % l, j] + lattice[(i + 1) % l, j] + \
         lattice[i, (j - 1) % l] + lattice[i, (j + 1) % l]
    SD1 = lattice[(i1 - 1) % l, j1] + lattice[(i1 + 1) % l, j1] + \
         lattice[i1, (j1 - 1) % l] + lattice[i1, (j1 + 1) % l]
    var=2*J*lattice[i,j]*(SD-SD1)+4*J
    return var

def move(i,j,i1,j1): # Montecarlo Move
    dE = deltaE(i, j, i1, j1)
    if dE < 0:
        lattice[i, j] = -lattice[i, j]
        lattice[i1, j1] = -lattice[i1, j1]
        return
    if np.random.random() < np.exp(-dE*beta):
        lattice[i, j] = -lattice[i, j]
        lattice[i1, j1] = -lattice[i1, j1]
        return
    return
```

Phase Transition

The Kawasaki model is able to reproduce some known experimental fact for the phase transitions in binary alloys :

1. For $T > T_c$ the two metals are well mixed
2. For $T < T_c$ one can observe a phase separation with each component of the alloy occupying only a localized spatial area
3. The evolution of the phase separation strongly depends on the initial relative proportion of the two metals A and B

   (a) If this proportion in around 50%, one oberve the so-called spinodal decomposition or coarsening, where where filament-like structures connecting the atoms coarsen and grow in time

   (b) If one of the two metal is more abundant than the other, the metal that is in minority starts forming droplets which coalescence in time and finally only one large one will remain, plus several minor ones

How to plot more graphs

```
T=0.95
beta=1./T  # K_B =1
nit = 4 # number of iterations
print("temperature",T)
iumax=1
for t in range(0,nit):
    for iu in range(0,iumax):
        mc=0
        while mc < n*K: # K MC steps is n*K moves
            # the data are more independent
            i,j= np.random.randint(l), np.random.randint(l)
            inn = np.random.random_integers(0,1)
            iu = 2*np.random.random_integers(0,1)-1
            if inn != 0:
                i1,j1=i+iu,j
            else:
                i1,j1=i,j+iu
            i1,j1 =i1 %l,j1 %l
            if lattice[i, j]*lattice[i1, j1] <0 :
                move(i,j,i1,j1)
                mc+=1
```

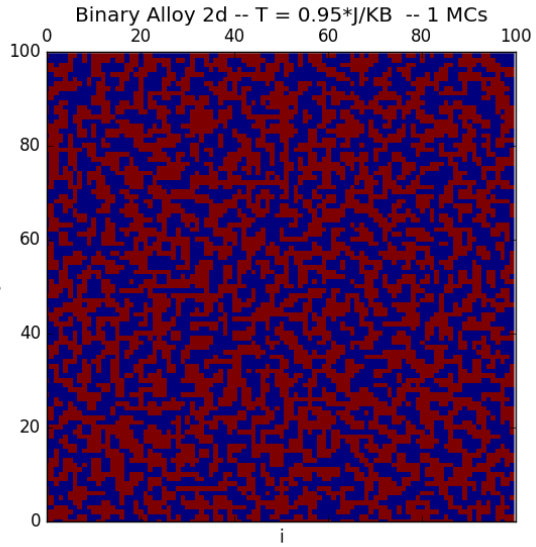
How to plot more graphs

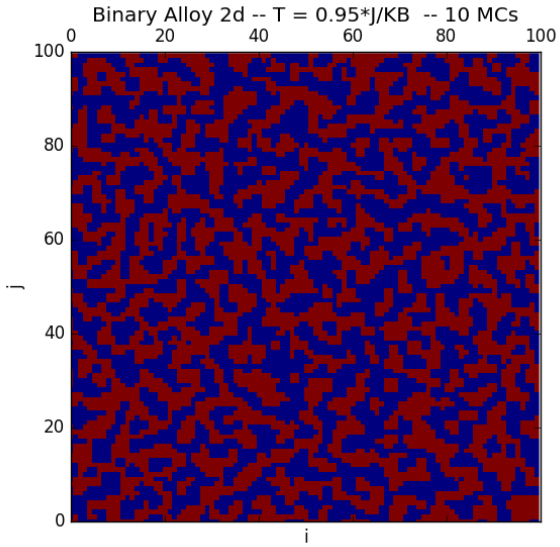
```
    plt.matshow(lattice)
    plt.xlabel("i")
    plt.ylabel("j")
    plt.title("Binary Alloy 2d -- T = 0.95*J/KB ")
    plt.xlim(0,l)
    plt.ylim(0,l)
    plt.legend()
    iumax=iumax*10

plt.show()
input()
```

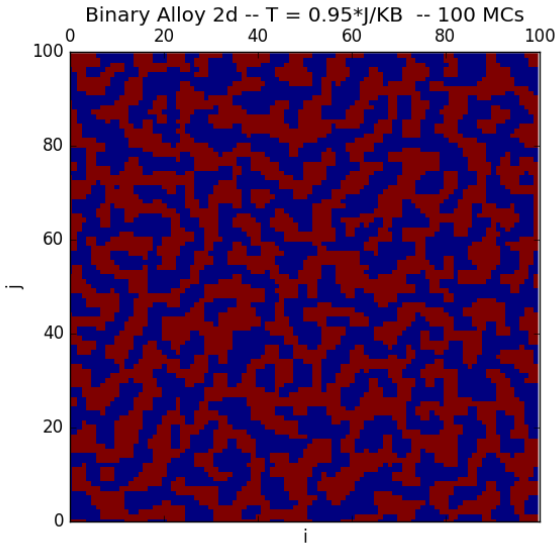
If I choose random initial conditions, almost 50% for A and B, what do I observe ?

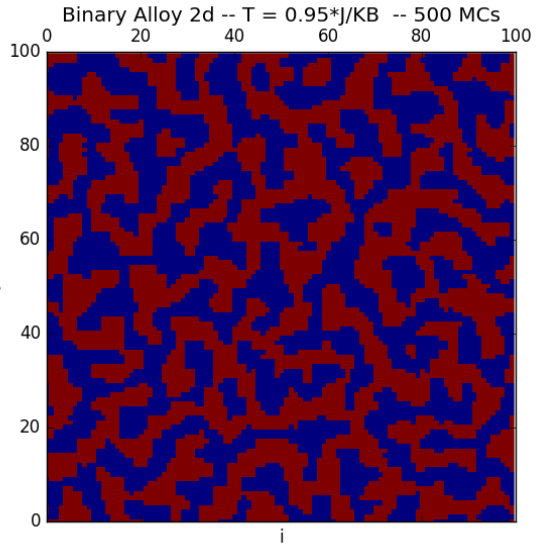
IDLE3

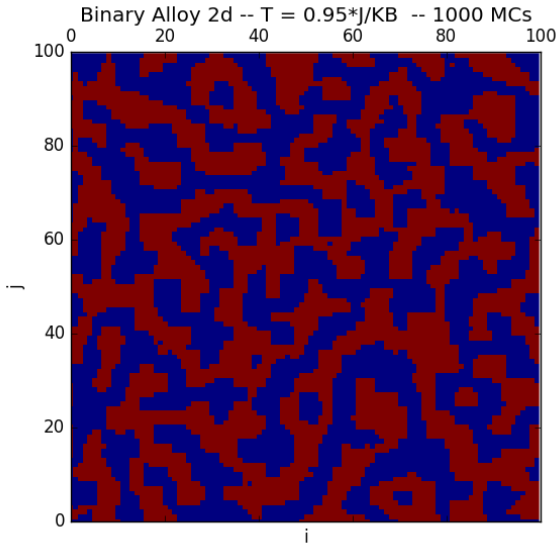
Coarsening - probability $= 50\%$

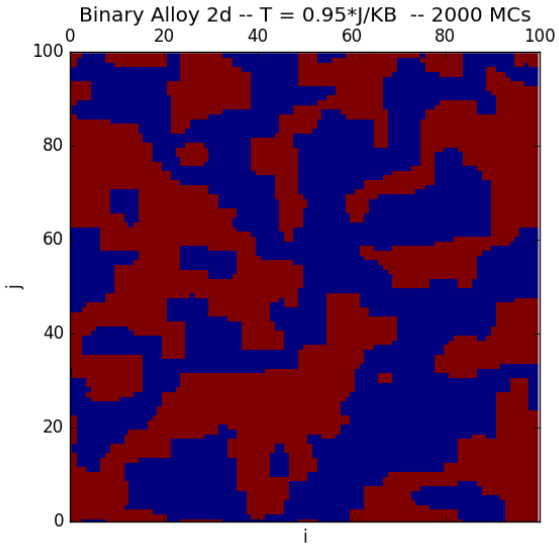












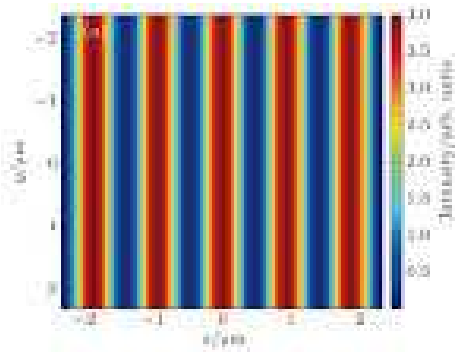
Coarsening - what do measure ?

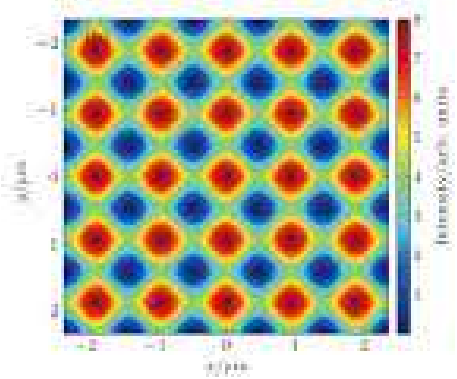
1. In this case it makes no sense to measure the magnetization because the number of atoms A and B is fixed, and also M
2. In 2 spatial dimensions, we can measure the number N_c of places where the spins change sign in the lattice, this is the number of links connecting spins of different sign.
3. N_c represents the Contact Area between the metal A and the metal B
4. The typical size of one phase L_c can be obtained as

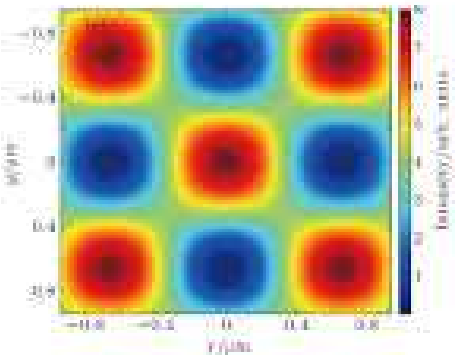
$$L_c \propto \frac{N}{(N_c/2)} = \frac{L^2}{(N_c/2)}$$

5. Suppose one has regular areas of side L_c of atoms A and B inside the total square of side L, how much is in this case N_c ?

Estimating N_c







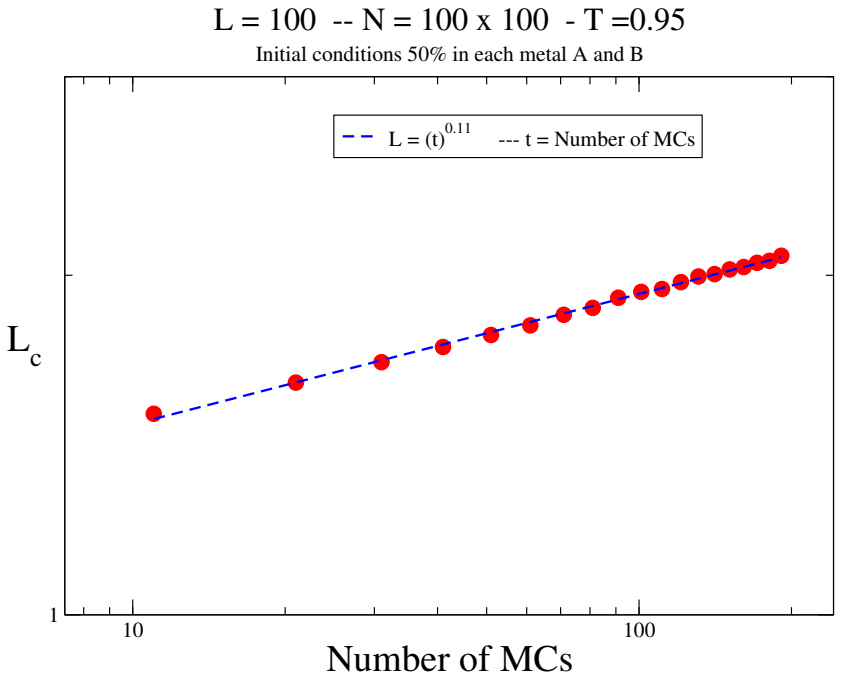
We should now find a way to count N_c let us make an example at the black-board

A volunteer ?

Counting the number of anti-parallel spins

```
def  nch(i,j):
    '''counts the number of anti-parallel spins among neighbours'''
    ss = lattice[(i - 1) % l, j] + lattice[(i + 1) % l, j] + \
         lattice[i, (j - 1) % l] + lattice[i, (j + 1) % l]
    ss=ss*lattice[i,j]
    if ss == 4:
        return 0
    if ss == 2:
        return 1
    if ss ==0 :
        return 2
    if ss == -2:
        return 3
    if ss == -4:
        return 4
    return
```

Scaling of the characteristic lenght



$$L_c = t^{\beta} \qquad \beta > 0 \qquad \lim_{t \to \infty} L_c \to \infty$$

The characteristic dimension of the domains diverge in the infinite time

Non equal initial probability

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

def init_lattice_p(l,p):
    '''Create a lxl lattice with random binomial spin configuration'''
    '''Atom A have a density p and atoms B have a density 1-p'''
    lattice = 2*np.random.binomial(1,p,size=(l,l))-1
    print(lattice)
    return lattice

def deltaE(i,j,i1,j1):
    '''Energy difference for a spin exchange of 2 neighbours'''
    # periodic  boundary condtions
    SD = lattice[(i - 1) % l, j] + lattice[(i + 1) % l, j] + \
         lattice[i, (j - 1) % l] + lattice[i, (j + 1) % l]
    SD1 = lattice[(i1 - 1) % l, j1] + lattice[(i1 + 1) % l, j1] + \
         lattice[i1, (j1 - 1) % l] + lattice[i1, (j1 + 1) % l]
    var=2*J*lattice[i,j]*(SD-SD1)+4*J
    return var
```

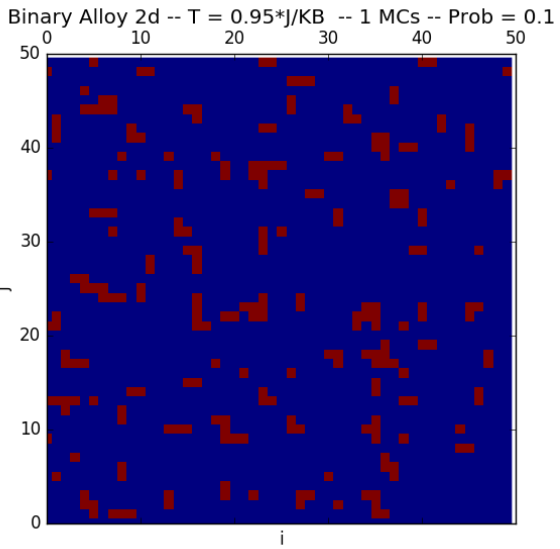
Non equal initial probability

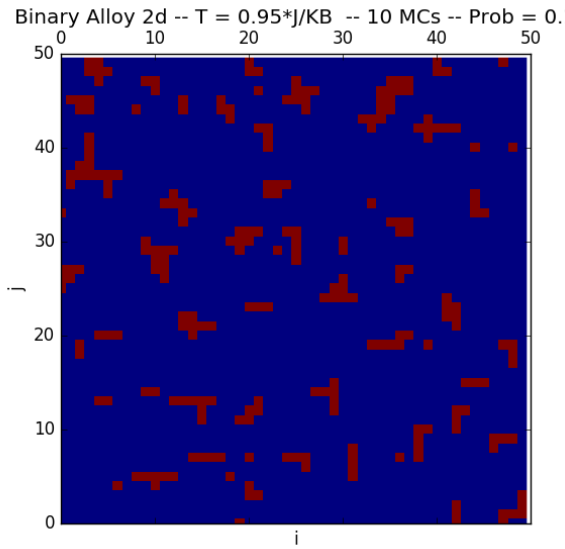
```
def move(i,j,i1,j1): # Montecarlo Move
    dE = deltaE(i, j, i1, j1)
    if dE < 0:
        lattice[i, j] = -lattice[i, j]
        lattice[i1, j1] = -lattice[i1, j1]
        return
    if np.random.random() < np.exp(-dE*beta):
        lattice[i, j] = -lattice[i, j]
        lattice[i1, j1] = -lattice[i1, j1]
        return
    return

global lattice,J,beta,l
J=1
l= 50   # lenght of the lattice
n= l * l  # number of sites
K=1      # parameter for the MC
# random initial conditions
lattice = init_lattice_p(l,0.1)
```

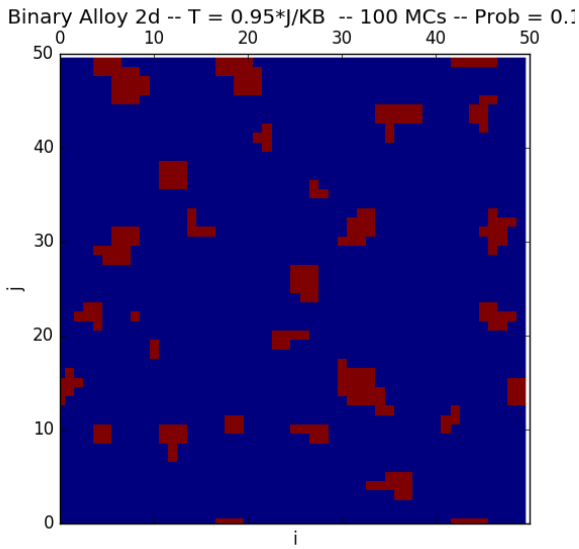
Non equal initial probability

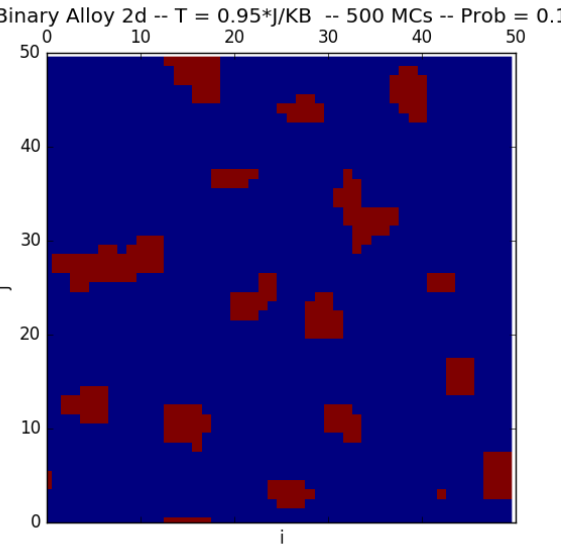
```
T=0.95
beta=1./T  # K_B =1
nit = 1000 # number of iterations
print("temperature",T)
iumax=1
for t in range(0,nit):
        print(t)
        mc=0
        while mc < n*K: # K MC steps is n*K moves
            # the data are more independent
            i,j= np.random.randint(l), np.random.randint(l)
            inn = np.random.random_integers(0,1)
            iu = 2*np.random.random_integers(0,1)-1
            if inn != 0:
                i1,j1=i+iu,j
            else:
                i1,j1=i,j+iu
            i1,j1 =i1 %l,j1 %l
            if lattice[i, j]*lattice[i1, j1] <0 :
                move(i,j,i1,j1)
```

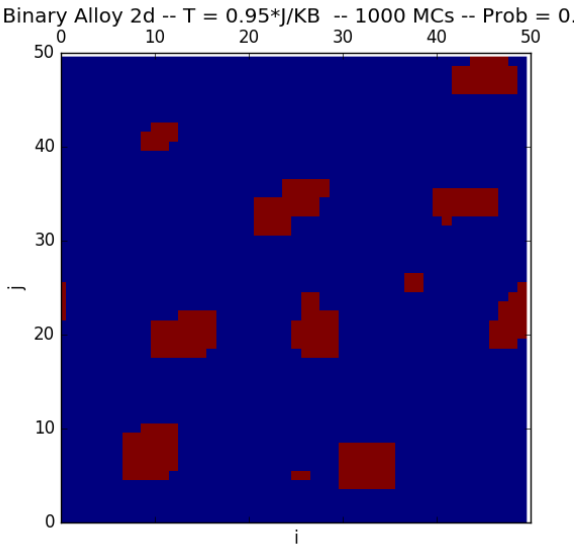
Droplet coalescence - probability = 10%

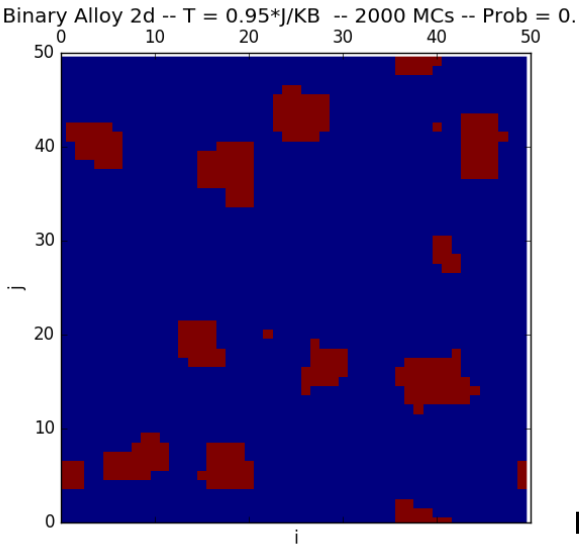




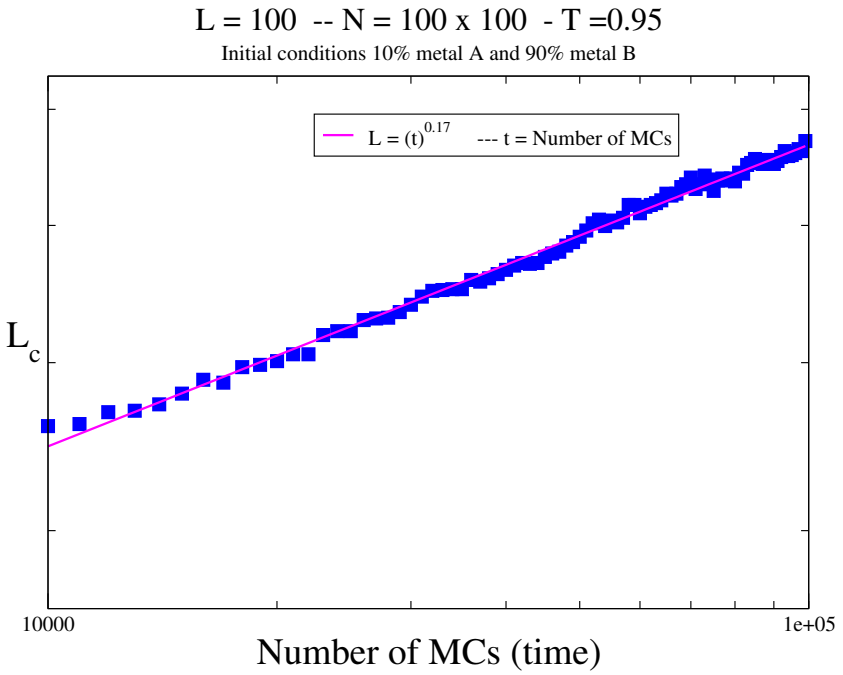








Scaling of the characteristic lenght



$$L_c = t^{\beta} \qquad \beta > 0 \qquad \lim_{t \to \infty} L_c \to \infty$$

The characteristic dimension of the droplet diverges in the infinite time (MC steps)

Fast quenching vs Slow cooling

1. The properties of the material depends in how fast its temperature is reduced from the liquid state down to the solid state
2. If the temperature is reduced rapidly we say that the system is quenched
3. If the temperature is decreased in small steps we say that the material is slowly cooled

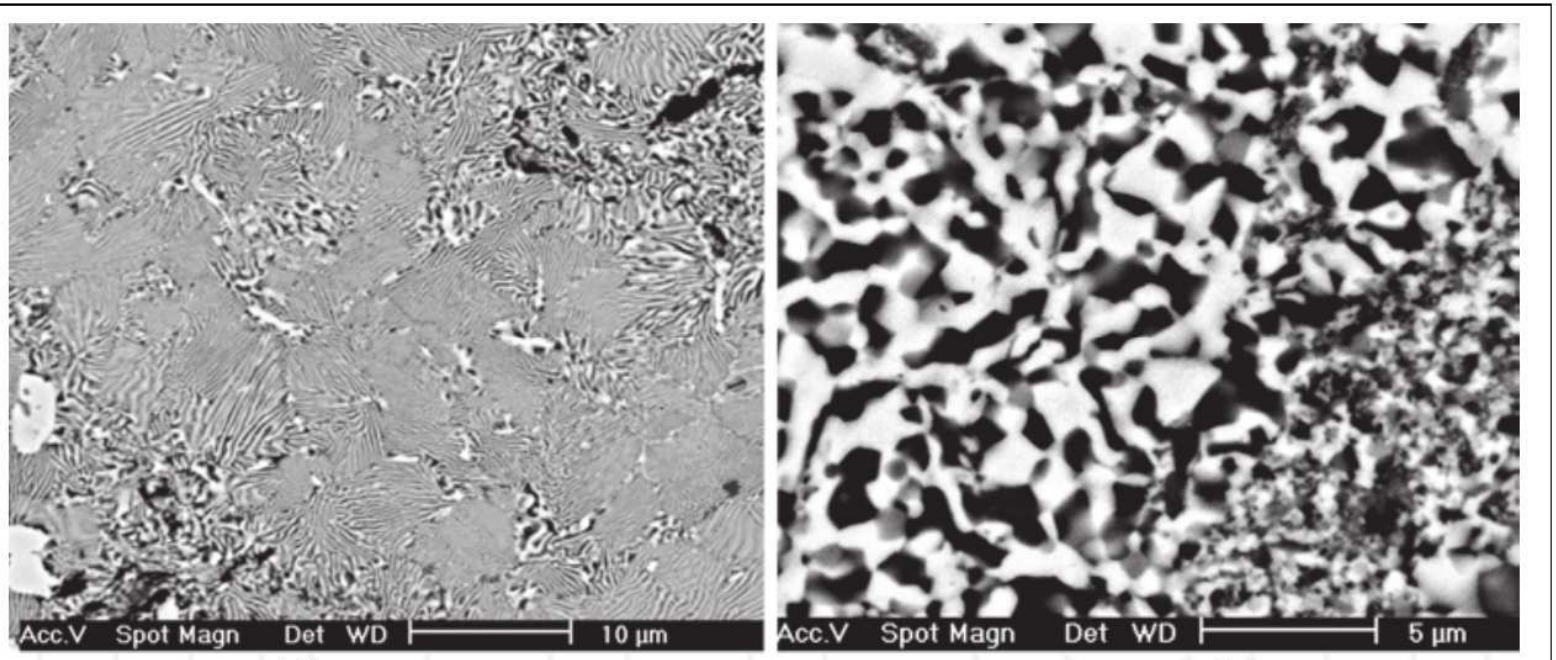


Fig. 2. Microstructures in Zn-Al-Cu. (a) Slowly cooled alloy. (b) Quenched alloy

Quenching - Opening of Multiple files

```
global lattice,J,beta,l
J,l=1, 100
n= l * l  # number of sites
K=1       # parameter for the MC
file=['quenchT4.png','quenchT1.png','quenchT0.25.png','quenchT0.0625.png']
# random initial conditions 50%
lattice = init_lattice(l)
# warming up phase at high temperature
T=4.00
beta=1./T  # K_B =1
for iu in range(0,100):
        print(iu)
        mc=0
        while mc < n*K: # K MC steps is n*K moves
            # the data are more independent
            i,j= np.random.randint(l), np.random.randint(l)
```

Quenching - Opening of Multiple files

```
            inn = np.random.random_integers(0,1)
            iu = 2*np.random.random_integers(0,1)-1
            if inn != 0:
                i1,j1=i+iu,j
            else:
                i1,j1=i,j+iu
            i1,j1 =i1 %l,j1 %l
            if lattice[i, j]*lattice[i1, j1] <0 :
                move(i,j,i1,j1)
                mc+=1
plt.matshow(lattice)
plt.xlabel("i")
plt.ylabel("j")
plt.title("Binary Alloy 2d quenched  T =4 --> T=0.0625 3 steps")
plt.xlim(0,l)
plt.ylim(0,l)
plt.savefig(file[0])
```

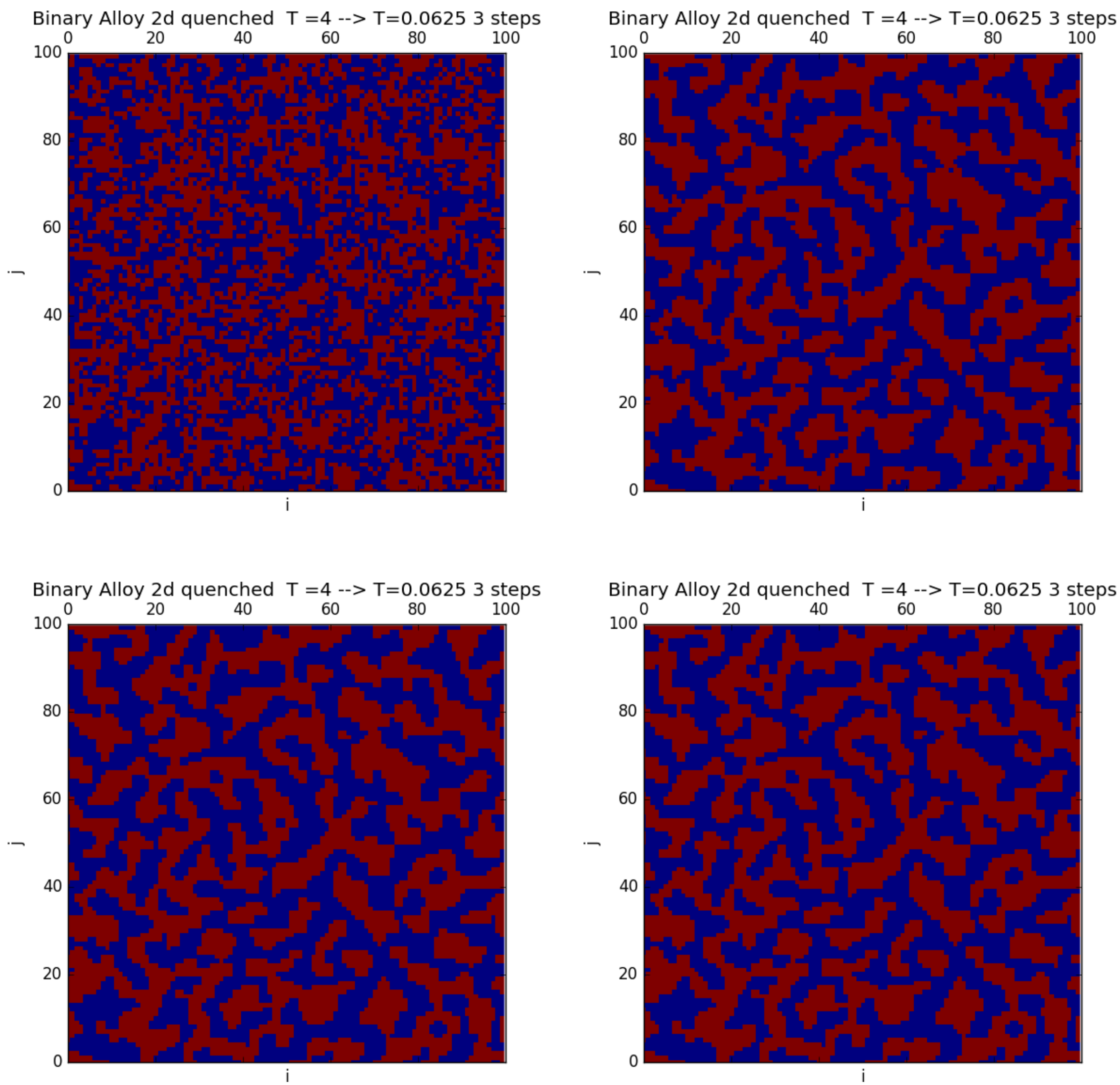
Quenching - Opening of Multiple files

```
nit = 3 # number of iterations
iumax=100
for t in range(0,nit):
    T=T/4.   # rapid quenching
    beta=1./T  # K_B =1
    print("temperature",T)
    for iu in range(0,iumax):
        mc=0
        while mc < n*K: # K MC steps is n*K moves
            i,j= np.random.randint(l), np.random.randint(l)
            inn = np.random.random_integers(0,1)
            iu = 2*np.random.random_integers(0,1)-1
            if inn != 0:
                i1,j1=i+iu,j
            else:
                i1,j1=i,j+iu
            i1,j1 =i1 %l,j1 %l
            if lattice[i, j]*lattice[i1, j1] <0 :
                move(i,j,i1,j1)
                mc+=1
```

Quenching - Opening of Multiple files

```
    plt.matshow(lattice)
    plt.xlabel("i")
    plt.ylabel("j")
    plt.title("Binary Alloy 2d quenched  T =4 --> T=0.0625 3 steps")
    plt.xlim(0,l)
    plt.ylim(0,l)
    plt.savefig(file[t+1])

plt.show()
```

Quenching from $T = 4$ to $T = 0.0625$ in 3 steps



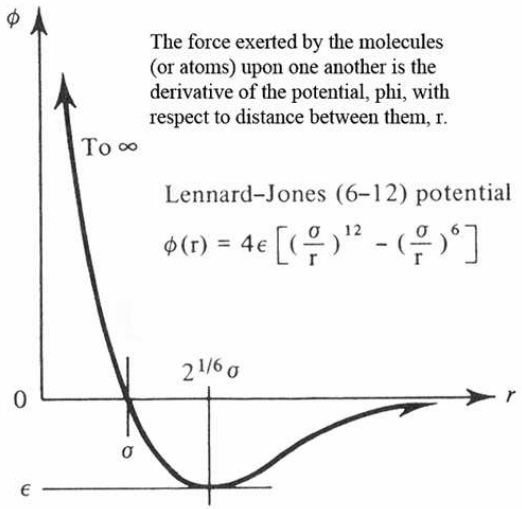
The Lennard-Jones Fluid

It is possible to simulate with a Montecarlo strategy also a continuous system like a fluid (gas or liquid). This is a classical system composed of N particles of mass m and interacting via a two body interactions $v(\vec{r}_i, \vec{r}_j)$:

$$\mathcal{H} = \mathcal{K} + \mathcal{V} = \sum_{i=1}^{N} \frac{p_i^2}{2m} + V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \sum_{i=1}^{N} \frac{m v_i^2}{2} + \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} v(\vec{r}_i, \vec{r}_j)$$

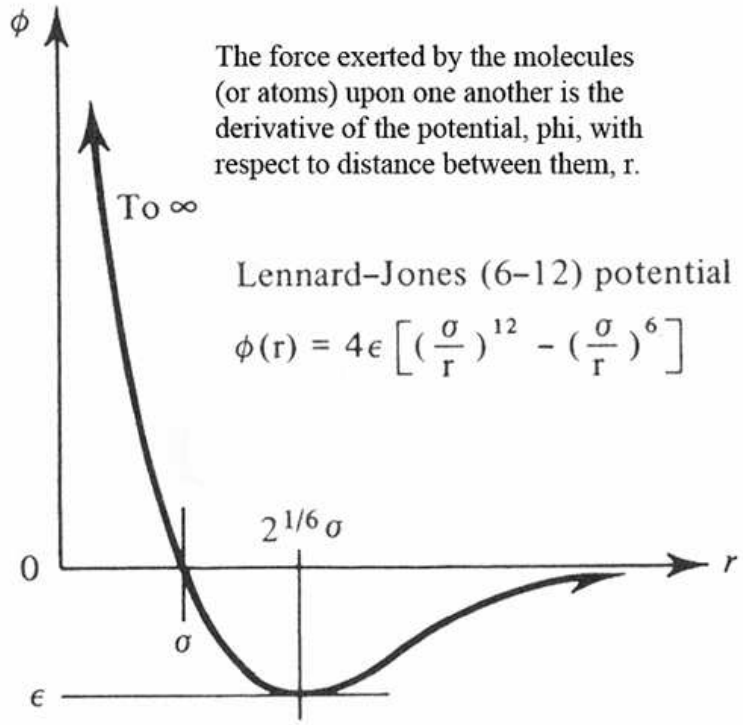
The most studied model is the Lennard-Jones fluid, which describes the dynamcs of Noble Gases (Ar, He, Ne, Kr, Xe, Rn), the corresponding two body potential can be written as :

$$v(\vec{r}_i, \vec{r}_j) = \Phi(r_{ij}) = 4\varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$



1. $r_{ij} = |\vec{r}_i - \vec{r}_j|$ is the distance among the two particles i and j
2. ε is the depth of the minimum of the potential
3. σ is the distance for which the potential vanishes

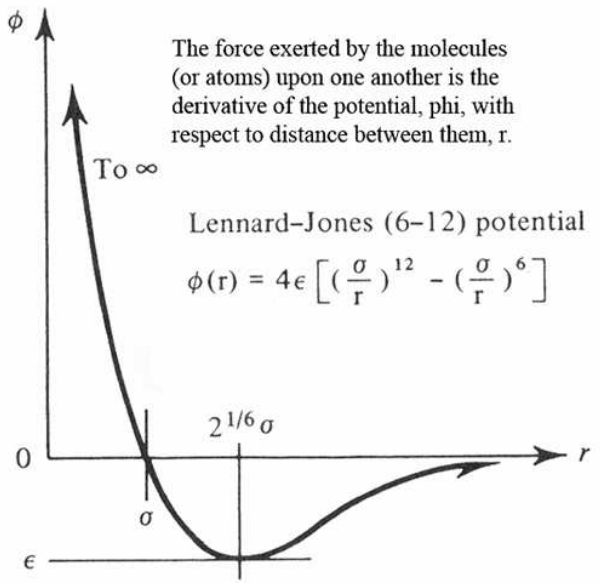
The Lennard-Jones Fluid



The force acting on the particles is given by $\vec{F} = -\nabla V(r) = -\frac{dV}{dr}\frac{\vec{r}}{r}$

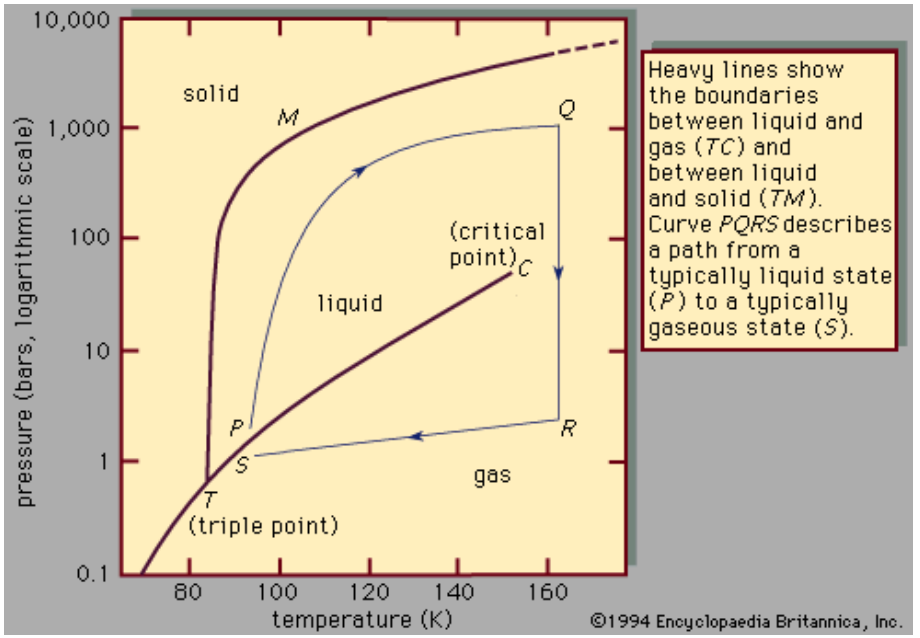
1. The force is repulsive ($F > 0$) for $r < r_{min}$
2. The force is attractive ($F < 0$) for $r > r_{min}$
3. The force is zero and the two particles are at equilibrium at a distance $r = r_{min} = 2^{1/6}\sigma$

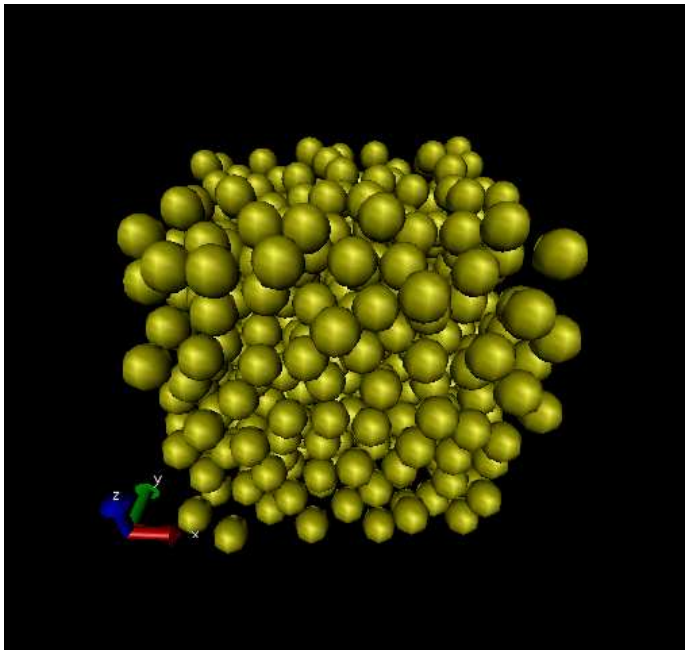
The Lennard-Jones Fluid



1. The short range repulsion term $1/r^{12}$ is due to the Pauli exclusion principle when the electrons of the two atoms come too near and tend to occupy the same energy levels ;
2. The Long range attractive term $1/r^{6}$ is due to teh van der Waals forces associated to fluctuating dipoles
3. The lowest energy configuration is the solid hexagonal close-packing (HCP) phase, that at higher teperature becomes a face-centered cubic (FCC) phase.

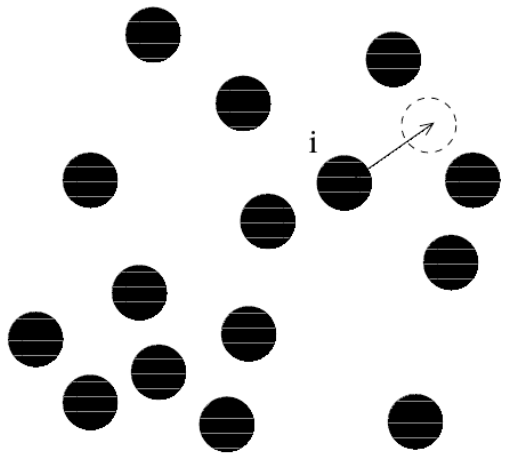
The Argon Phase Diagram





A LJ solid-fluid can be simulated in a NVT ensemble by emploing the Montecarlo Method

The Montecarlo Simulation



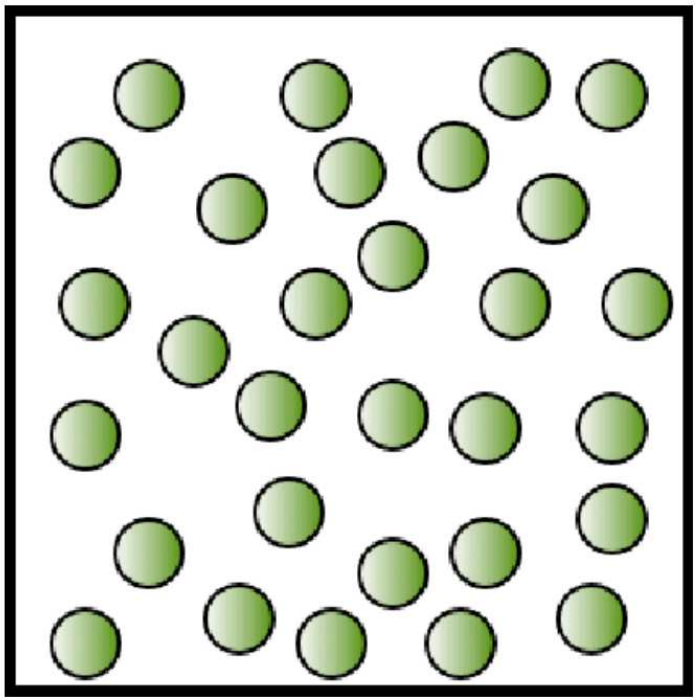
The Montecarlo Method

1. A LJ particle is randomly selected
2. The random particle i is randomly translated as follows

$$x_i \to x_i' = x_i + \delta\xi_1 \qquad y_i \to y_i' = y_i + \delta\xi_2 \qquad z_i \to z_i' = z_i + \delta\xi_3$$

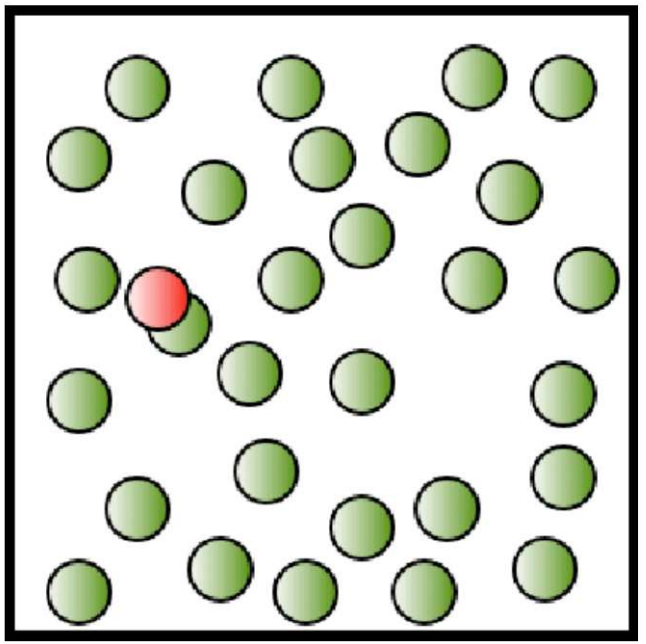
   where ξ_1, ξ_2, ξ_3 are uniform random numbers in $[-1, 1]$ and δ is the amplitude of the displacement
3. δ is chosen to be of the order of the average distance among particles

The Montecarlo Simulation



Initial configuration
Dense fluid

The Montecarlo Simulation



Move a molecule at random. High probability of overlap. *Move rejected*

The Montecarlo Simulation

The Metropolis Algorithm

1. The difference of energy among the two configurations to use in the Metropolis algorithm is now
   $\Delta E = \frac{1}{N}\sum_{i>j}[\Phi(r'_{ij}) - \Phi(r_{ij})] \qquad r'_{ij} = |\vec{r}'_i - \vec{r}_j|$
2. The sum is over all the couples of atoms, without repetitions, but it includes $N(N-1)/2$ terms $\leftarrow$ too much !
3. Therefore the potental is often truncated at a distance $r_c \simeq 2 - 3\sigma$

$$\Phi_{trunc}(r) = \begin{cases} \Phi(r) & \text{if } r \leq r_c; \\ 0 & \text{if } r > r_c \ . \end{cases}$$

4. Only the particles inside the radius r_c are considered in the energy estimation, this speeds up the algorithm (list of neighbours are employed)
5. Question : Why we do not consider the Kinetic Energy, but only the Potential one ?
6. Question : Which is a good initial condition ?

Molecular Liquids

For molecular systems, the elementary moves must change all the configurational degrees of freedom

1. rigid translation
2. rigid rotation
3. rotation about bonds
4. bond distortion