Teaching Physics at University by using « Open source » and « home-made » modelling tools

Hervé Bulou, IPCMS, Strasbourg

herve.bulou@ipcms.unistra.fr

Université de Strasbourg Excellence University

IPCMS

cnrs

« Simulation and Modelling of Complex Materials and Phenomena » Team

F. Maingot de la Grassière

Permanent Members

- O. Bengone
- H. Bulou
- C. Goyhenex
- F. Maingot de la Grassière

« Simulation and Modelling of Complex Materials and Phenomena» - SMCMP Team

Materials

- Biomineralization materials
- Implants
- Supported molecules & molecular mats
- High Entropy Alloys

Phenomena

- Nucleation & Growth in liquid phase
- Material Aging
- Transport phenomena in nanodevices
- Gas Phase Catalysis

Methods

- Molecular Dynamics (classical & *ab initio*)
- Monte Carlo (kinetic & Metropolis)
- Totale energy caclulation (Density Functional Theory, Force Fields)
- Linear response theory

« Simulation and Modelling of Complex Materials and Phenomena» - SMCMP Team

Methods

Theoretical framework

Molecular Dynamics
(classical & *ab initio*)

- Classical physics
- *Ab initio*
  Car-Parrinello, Born-Oppenheimer

Monte Carlo
(Kinetic & Metropolis)

- Classical physical

Totale energy caclulation
(Density Functional Theory, Force Fields)

- Classical physical
- *Ab initio*
  Density Functional Theory (DFT)

Linear Response Theory

- *Ab Initio*
  (Time-Dependent DFT)

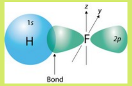
Classical physics

Force Fields

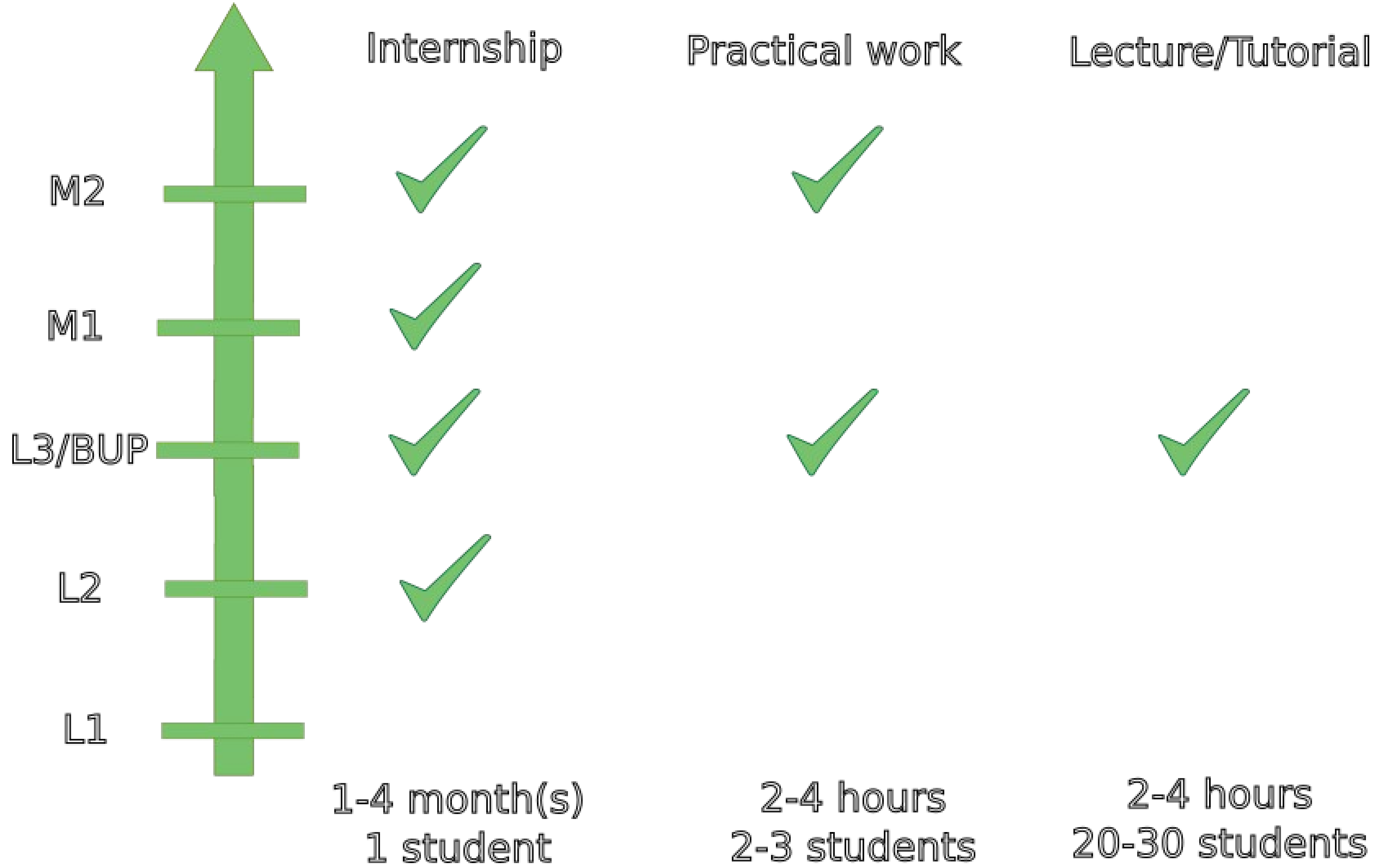
- « Home-made » codes
- LAMMPS

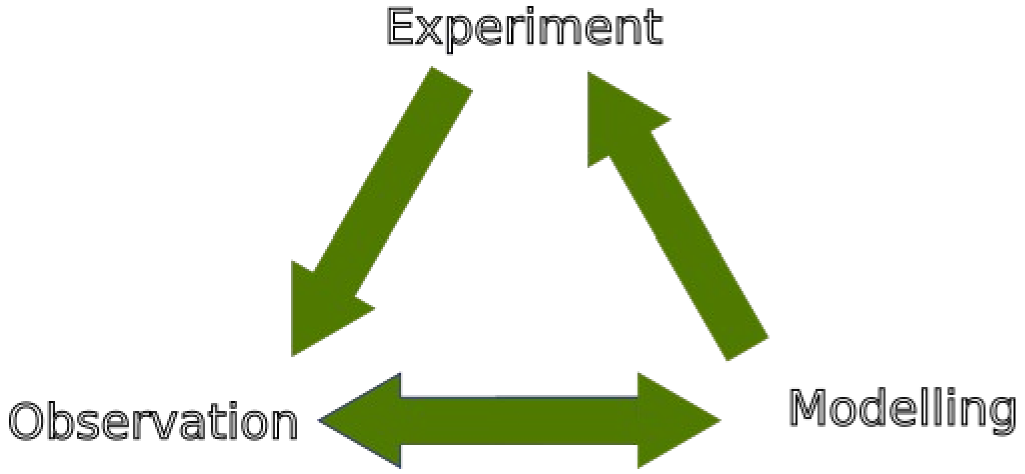
Quantum mechanics

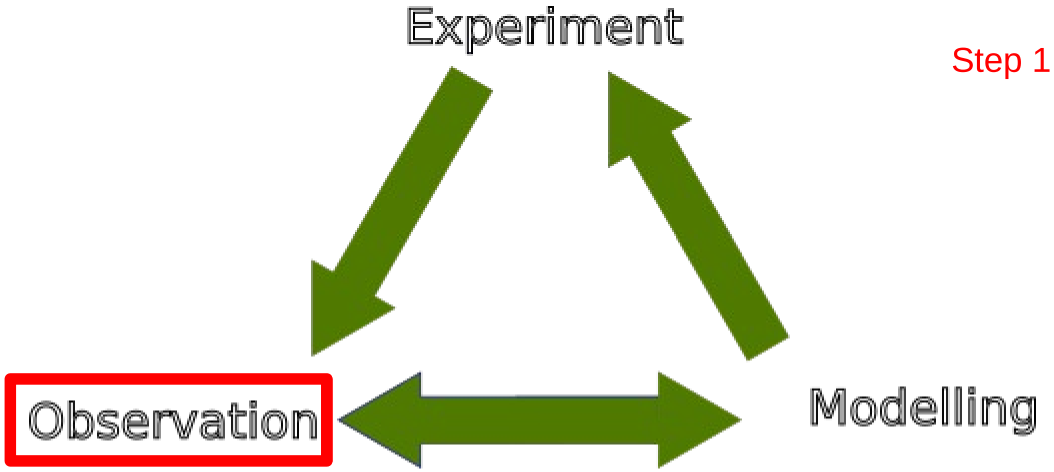
Electromagnetic interaction



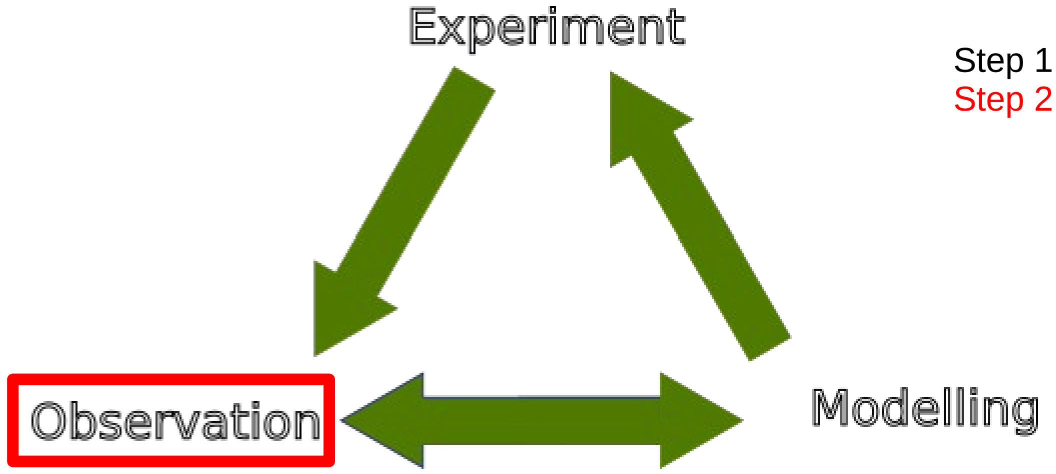
- « Home-made » Codes
- CP2K, Octopus, Quantum Espresso, SALMON


Internship
Practical work
Lecture/Tutorial
M2
M1
L3/BUP
L2
L1
1-4 month(s)
1 student
2-4 hours
2-3 students
2-4 hours
20-30 students

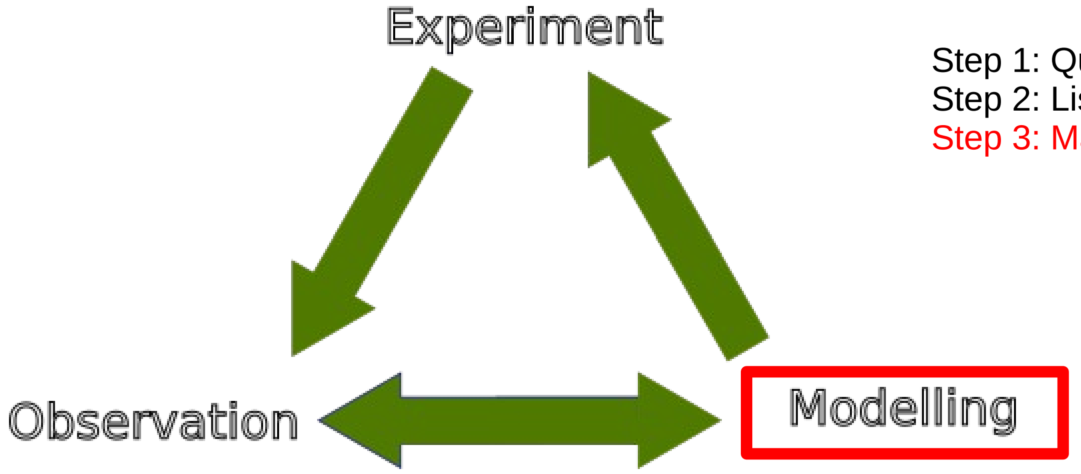

Experiment
Observation
Modelling



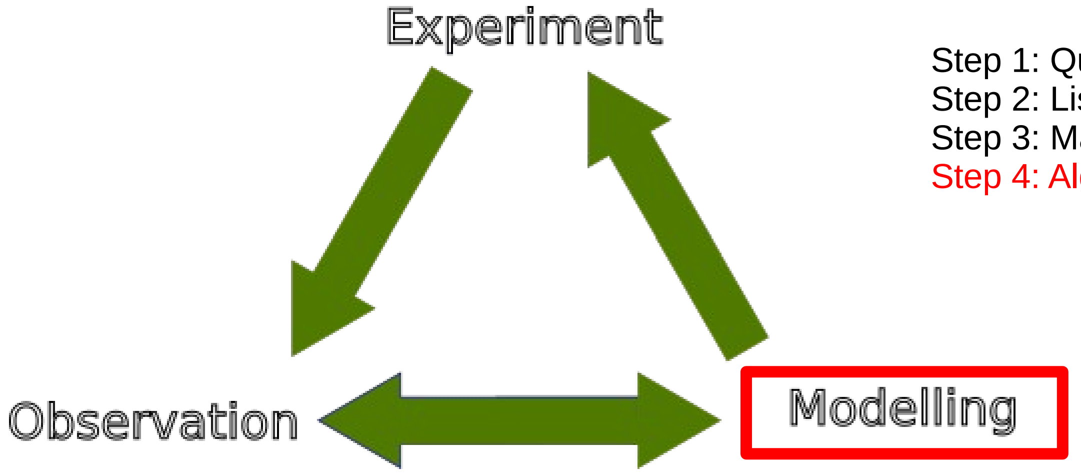
Step 1: Qualitative description of the observation



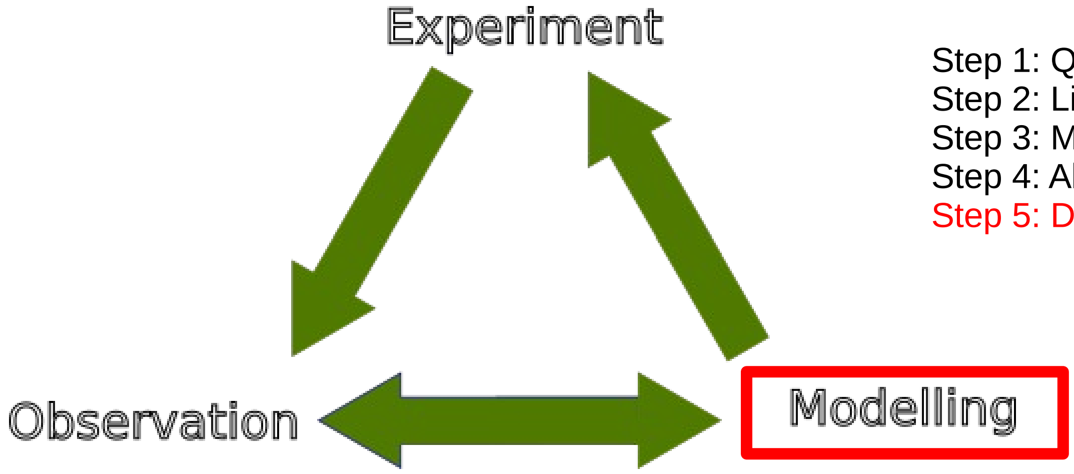
Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved



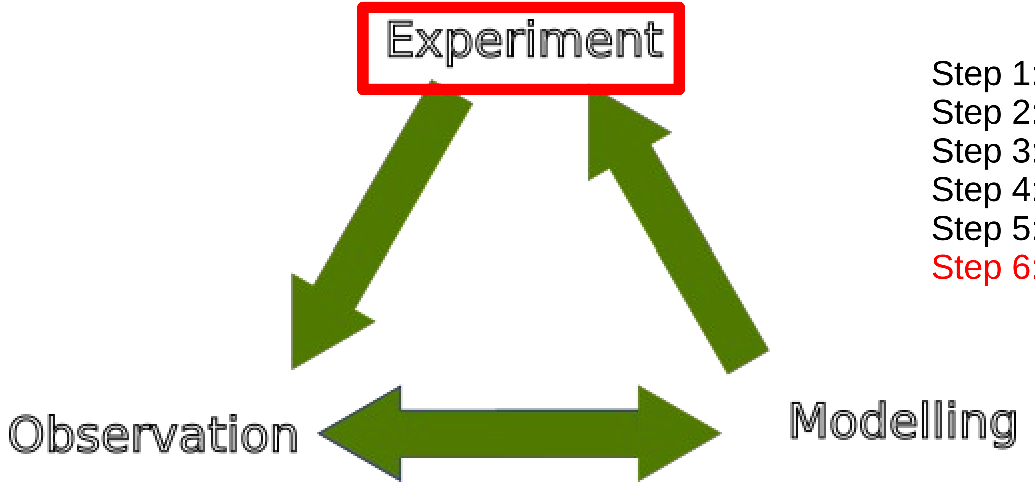
Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation



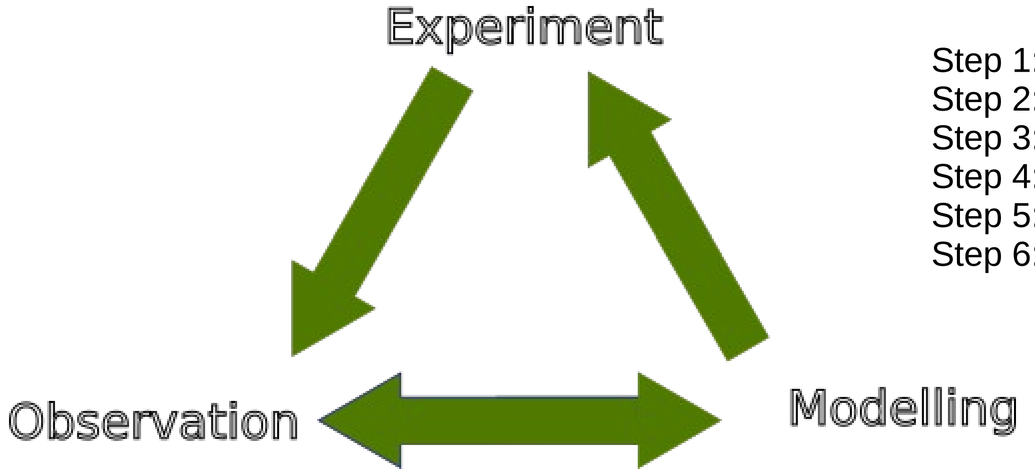
Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation
Step 4: Algorithmic formulation of observation



Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation
Step 4: Algorithmic formulation of observation
Step 5: Development of code to reproduce the observation



Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation
Step 4: Algorithmic formulation of observation
Step 5: Development of code to reproduce the observation
Step 6: Exploitation of the model to imagine new experiences



Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation
Step 4: Algorithmic formulation of observation
Step 5: Development of code to reproduce the observation
Step 6: Exploitation of the model to imagine new experiences

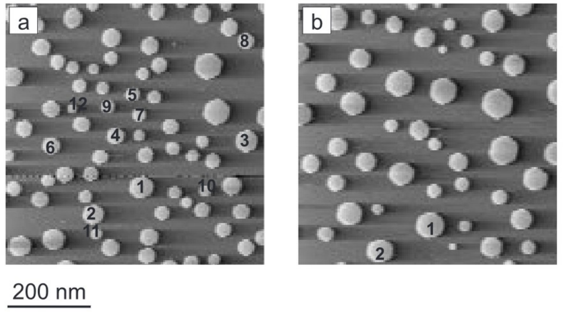
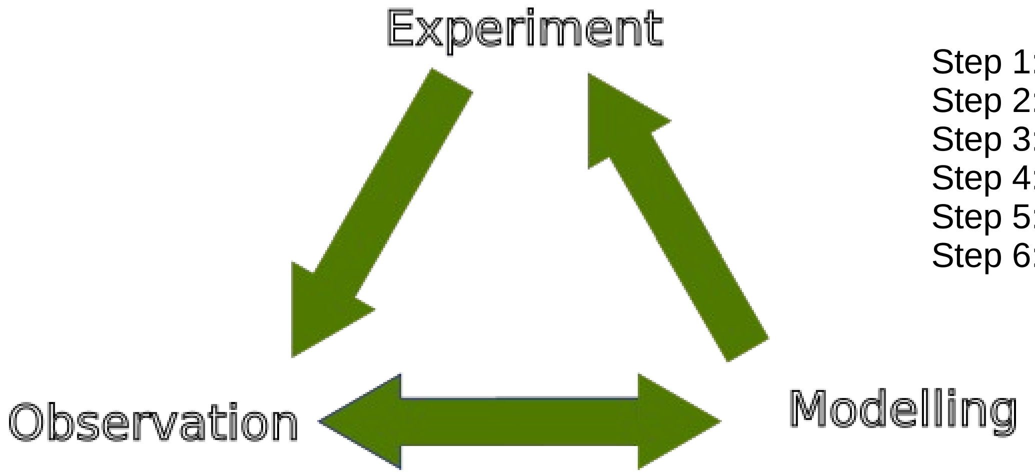


Fig. 1a,b. STM images taken from a movie sequence showing Ostwald ripening of 0.21 ML Ag on Ag(111) at room temperature. The left image (**a**) was recorded 38 minutes after the completion of deposition, the right image (**b**) shows the same sample another 6 h and 56 min later. The area of the islands marked by numbers on the first image is plotted in Fig. 2 as a function of time. For better orientation, the islands 1 and 2 are marked on the right image as well

Rosenfeld et al, Appl Phys A 69, 489 (1999).



Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation
Step 4: Algorithmic formulation of observation
Step 5: Development of code to reproduce the observation
Step 6: Exploitation of the model to imagine new experiences

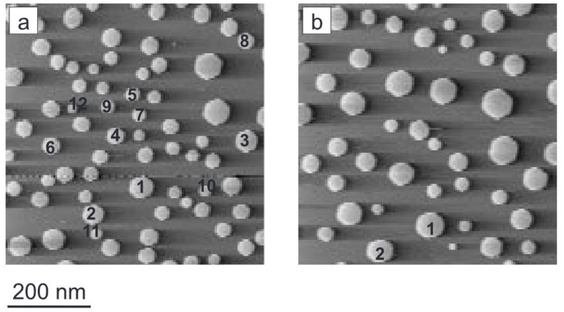
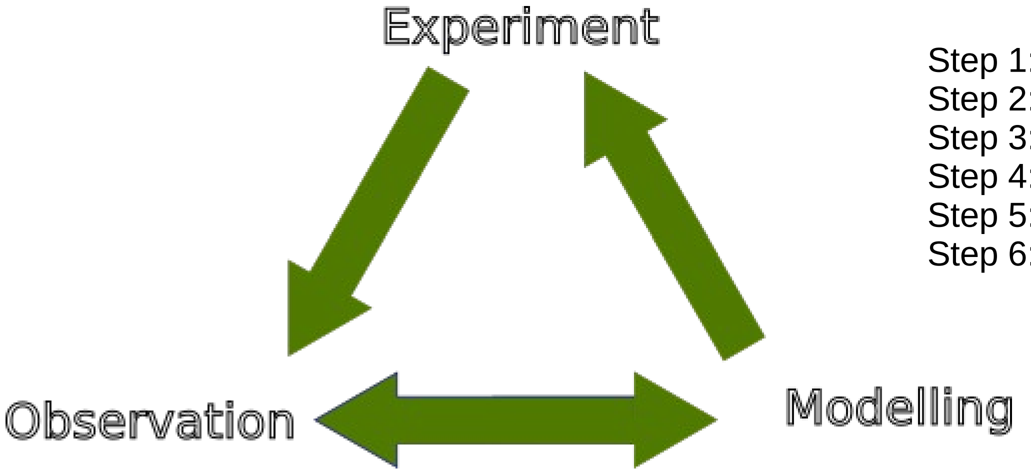


Fig. 1a,b. STM images taken from a movie sequence showing Ostwald ripening of 0.21 ML Ag on Ag(111) at room temperature. The left image (**a**) was recorded 38 minutes after the completion of deposition, the right image (**b**) shows the same sample another 6 h and 56 min later. The area of the islands marked by numbers on the first image is plotted in Fig. 2 as a function of time. For better orientation, the islands 1 and 2 are marked on the right image as well

Mathematics: Taylor development

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \frac{\mathrm{d}\mathbf{r}(t)}{\mathrm{d}t}\delta t + \frac{1}{2}\frac{\mathrm{d}^2\mathbf{r}(t)}{\mathrm{d}t^2}\delta t^2 + \mathcal{O}(\delta t^3)$$

Rosenfeld et al, Appl Phys A 69, 489 (1999).



Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation
Step 4: Algorithmic formulation of observation
Step 5: Development of code to reproduce the observation
Step 6: Exploitation of the model to imagine new experiences

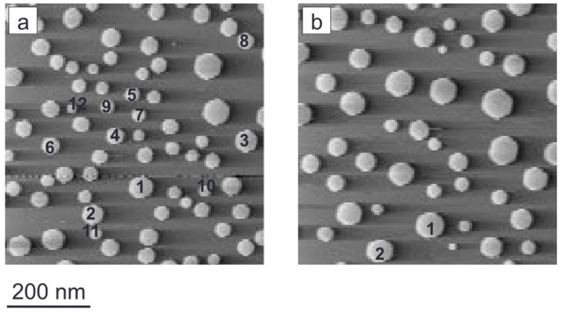


Fig. 1a,b. STM images taken from a movie sequence showing Ostwald ripening of 0.21 ML Ag on Ag(111) at room temperature. The left image (**a**) was recorded 38 minutes after the completion of deposition, the right image (**b**) shows the same sample another 6 h and 56 min later. The area of the islands marked by numbers on the first image is plotted in Fig. 2 as a function of time. For better orientation, the islands 1 and 2 are marked on the right image as well

Rosenfeld et al, Appl Phys A 69, 489 (1999).

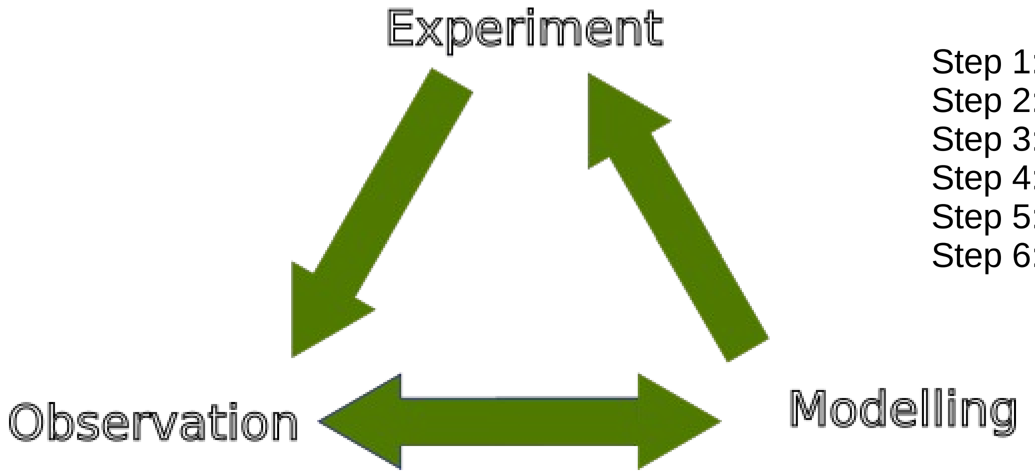
Mathematics: Taylor development

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \frac{\mathrm{d}\mathbf{r}(t)}{\mathrm{d}t}\delta t + \frac{1}{2}\frac{\mathrm{d}^2\mathbf{r}(t)}{\mathrm{d}t^2}\delta t^2 + \mathcal{O}(\delta t^3)$$

Physics: Newton equation

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \frac{\mathbf{p}(t)}{m}\delta t + \frac{1}{2}\frac{\mathbf{F}(t)}{m}\delta t^2 + \mathcal{O}(\delta t^3)$$

$$\mathbf{p}(t+\delta t) = \mathbf{p}(t) + \frac{1}{2m}\left(\mathbf{F}(t+\delta t) + \mathbf{F}(t)\right)\delta t + \mathcal{O}(\delta t^3)$$



Step 1: Qualitative description of the observation
Step 2: List of basic phenomena involved
Step 3: Mathematical formulation of the observation
Step 4: Algorithmic formulation of observation
Step 5: Development of code to reproduce the observation
Step 6: Exploitation of the model to imagine new experiences

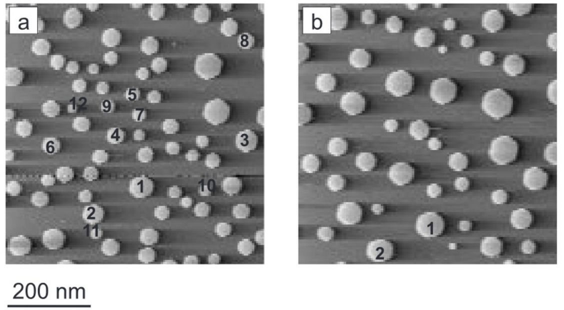


Fig. 1a,b. STM images taken from a movie sequence showing Ostwald ripening of 0.21 ML Ag on Ag(111) at room temperature. The left image (**a**) was recorded 38 minutes after the completion of deposition, the right image (**b**) shows the same sample another 6 h and 56 min later. The area of the islands marked by numbers on the first image is plotted in Fig. 2 as a function of time. For better orientation, the islands 1 and 2 are marked on the right image as well

Rosenfeld et al, Appl Phys A 69, 489 (1999).

Mathematics: Taylor development

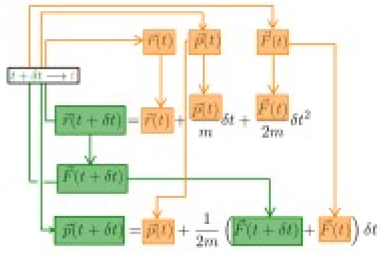
$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \frac{\mathrm{d}\mathbf{r}(t)}{\mathrm{d}t}\delta t + \frac{1}{2}\frac{\mathrm{d}^2\mathbf{r}(t)}{\mathrm{d}t^2}\delta t^2 + \mathcal{O}(\delta t^3)$$

Physics: Newton equation

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \frac{\mathbf{p}(t)}{m}\delta t + \frac{1}{2}\frac{\mathbf{F}(t)}{m}\delta t^2 + \mathcal{O}(\delta t^3)$$

$$\mathbf{p}(t+\delta t) = \mathbf{p}(t) + \frac{1}{2m}\left(\mathbf{F}(t+\delta t) + \mathbf{F}(t)\right)\delta t + \mathcal{O}(\delta t^3)$$

Computational science: Velocity-Verlet algorithm


$t+\delta t \longrightarrow t$
$\vec{r}(t)$
$\vec{p}(t)$
$\vec{F}(t)$
$\vec{r}(t+\delta t) = \vec{r}(t) + \frac{\vec{p}(t)}{m}\delta t + \frac{\vec{F}(t)}{2m}\delta t^2$
$\vec{F}(t+\delta t)$
$\vec{p}(t+\delta t) = \vec{p}(t) + \frac{1}{2m}\left(\vec{F}(t+\delta t) + \vec{F}(t)\right)\delta t$

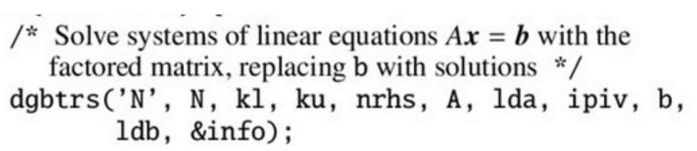
Why python?

- easier to use than C++ or FORTRAN

numpy.linalg.eigh
numpy.linalg.solve(a, b)

/

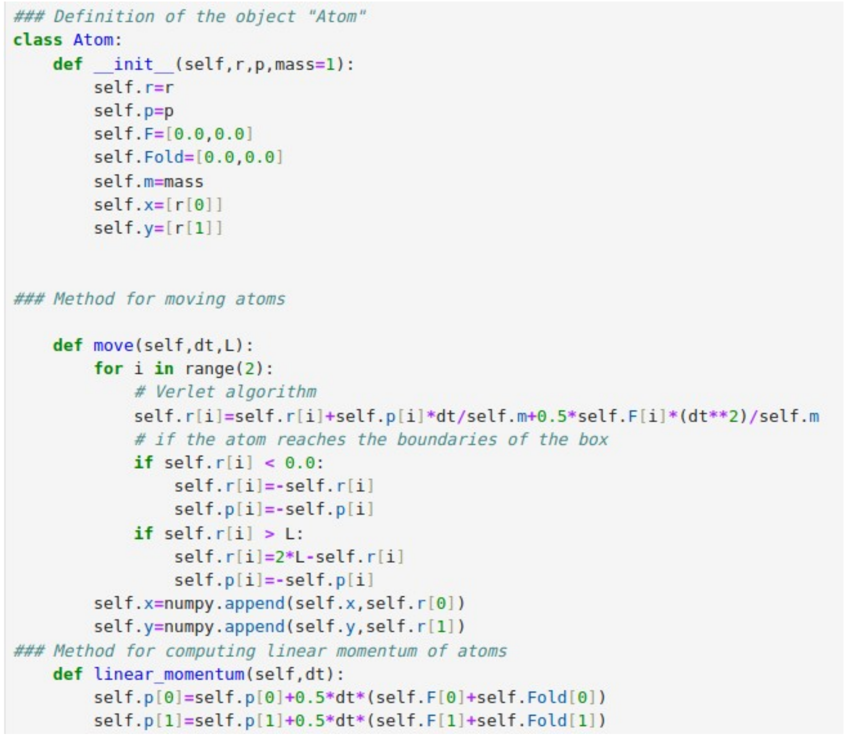
Lapack



```
/*  Solve systems of linear equations Ax = b with the
    factored matrix, replacing b with solutions  */
dgbtrs('N', N, kl, ku, nrhs, A, lda, ipiv, b,
        ldb, &info);
```

Why python?

- easier to use than C++ or FORTRAN
- allows an object-oriented approach to programming



```python
### Definition of the object "Atom"
class Atom:
    def __init__(self,r,p,mass=1):
        self.r=r
        self.p=p
        self.F=[0.0,0.0]
        self.Fold=[0.0,0.0]
        self.m=mass
        self.x=[r[0]]
        self.y=[r[1]]


### Method for moving atoms

    def move(self,dt,L):
        for i in range(2):
            # Verlet algorithm
            self.r[i]=self.r[i]+self.p[i]*dt/self.m+0.5*self.F[i]*(dt**2)/self.m
            # if the atom reaches the boundaries of the box
            if self.r[i] < 0.0:
                self.r[i]=-self.r[i]
                self.p[i]=-self.p[i]
            if self.r[i] > L:
                self.r[i]=2*L-self.r[i]
                self.p[i]=-self.p[i]
        self.x=numpy.append(self.x,self.r[0])
        self.y=numpy.append(self.y,self.r[1])
### Method for computing linear momentum of atoms
    def linear_momentum(self,dt):
        self.p[0]=self.p[0]+0.5*dt*(self.F[0]+self.Fold[0])
        self.p[1]=self.p[1]+0.5*dt*(self.F[1]+self.Fold[1])
```

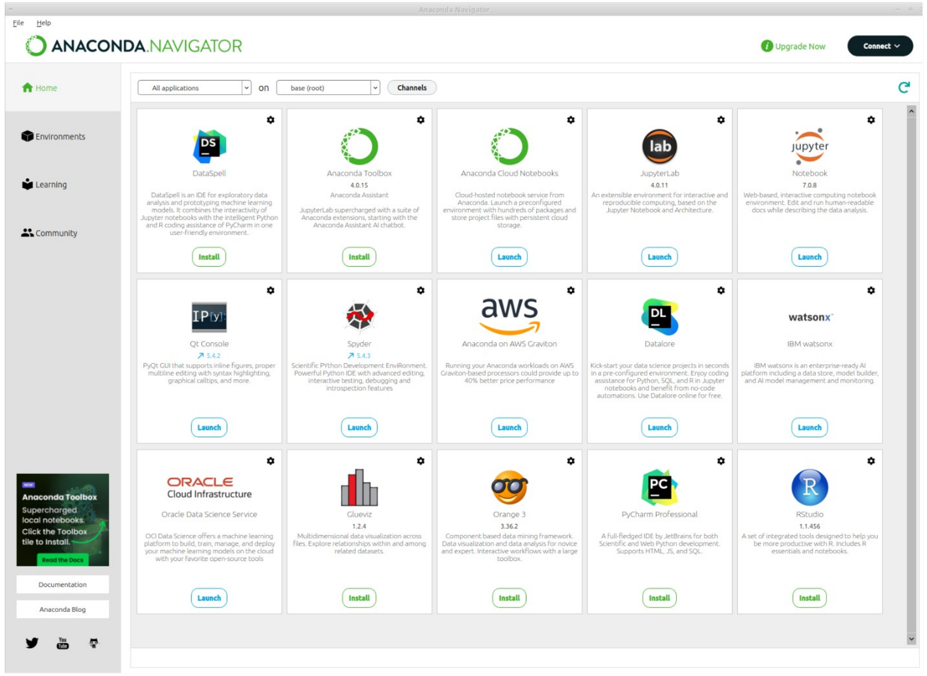
Why python?

- easier to use than C++ or FORTRAN
- allows an object-oriented approach to programming
- organization into modules

```
import numpy
from matplotlib import pyplot
```

Why python?

- easier to use than C++ or FORTRAN
- allows an object-oriented approach to programming
- organization into modules
- a wide range of IDE (Integrated Development Environment) (spyder, jupyter, etc.)



Anaconda Toolbox
Supercharged local notebooks. Click the Toolbox tile to Install.
Read the Docs

DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

Anaconda Toolbox
4.0.15
Anaconda Assistant
JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.

Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

JupyterLab
4.0.11
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
7.0.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console
↗ 5.4.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
↗ 5.4.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Anaconda on AWS Graviton
Running your Anaconda workloads on AWS Graviton-based processors could provide up to 40% better price performance

Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

IBM watsonx
IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.36.2
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Importing libraries

- to perform mathematical and scientific calculations

[1]:
```
import numpy
```

- To draw curves

[2]:
```
from matplotlib import pyplot
```

- to generate random numbers needed to initialize the velocities of the atoms

[3]:
```
import random
```

Interatomic interaction

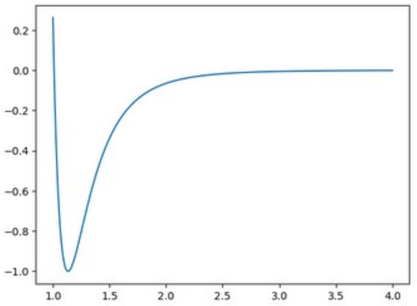
Lennard-Jones

$$V_{LJ}(r_{ij}) = 4\epsilon\left(\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right)$$

Potential energy of the crystal:

$$E_{\text{pot}} = \sum_{i=1}^{N-1}\sum_{j>i}^{N} V_{LJ}(r_{ij})$$

[4]:
```
eps=-1000.0
sig=0.25

sig=1.01
eps=1.0
def V(r,eps=eps,sig=sig):
    return 4*eps*((sig/r)**12-(sig/r)**6)
def gradV(r,rx,eps=eps,sig=sig):
    return (-24*eps*rx*(2*(sig/r)**12-(sig/r)**6)/r**2)
r=numpy.linspace(1.0,4.0,1000)
fig = pyplot.figure()
ax = fig.add_subplot()
pyplot.plot(r,V(r))
#ax.set_xlim(0.0,4.0)
#ax.set_ylim(-0.00001,0.00005)
#pyplot.legend()
pyplot.show()
```



Definition of objects

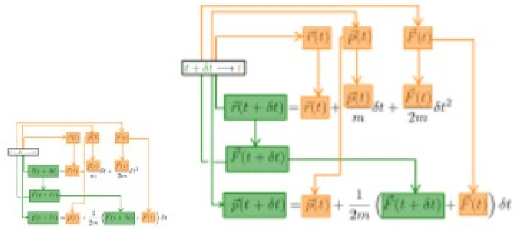
"Atom" object

The move() method moves an atom using a Velocity-Verlet algorithm.

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \frac{\mathbf{p}(t)}{m}\delta t + \frac{1}{2}\frac{\mathbf{F}(t)}{m}\delta t^2 + \mathcal{O}(\delta t^3)$$

$$\mathbf{p}(t+\delta t) = \mathbf{p}(t) + \frac{1}{2m}(\mathbf{F}(t+\delta t) + \mathbf{F}(t))\delta t + \mathcal{O}(\delta t^3)$$

Velocity-Verlet algorithm


$\vec{r}(t)$
$\vec{p}(t)$
$\vec{F}(t)$
$t+\delta t \longrightarrow t$
$\vec{r}(t+\delta t) = \vec{r}(t) + \frac{\vec{p}(t)}{m}\delta t + \frac{\vec{F}(t)}{2m}\delta t^2$
$\vec{F}(t+\delta t)$
$\vec{p}(t+\delta t) = \vec{p}(t) + \frac{1}{2m}\left(\vec{F}(t+\delta t) + \vec{F}(t)\right)\delta t$

[6]:
```
### Definition of the object "Atom"
class Atom:
    def __init__(self,r,p,mass=1):
        self.r=r
        self.p=p
        self.F=[0.0,0.0]
        self.Fold=[0.0,0.0]
        self.m=mass
        self.x=[r[0]]
        self.y=[r[1]]


### Method for moving atoms

    def move(self,dt,L):
        for i in range(2):
            # Verlet algorithm
            self.r[i]=self.r[i]+self.p[i]*dt/self.m+0.5*self.F[i]*(dt**2)/self.m
            # if the atom reaches the boundaries of the box
            if self.r[i] < 0.0:
                self.r[i]=-self.r[i]
                self.p[i]=-self.p[i]
            if self.r[i] > L:
                self.r[i]=2*L-self.r[i]
                self.p[i]=-self.p[i]
        self.x=numpy.append(self.x,self.r[0])
        self.y=numpy.append(self.y,self.r[1])
### Method for computing linear momentum of atoms
    def linear_momentum(self,dt):
        self.p[0]=self.p[0]+0.5*dt*(self.F[0]+self.Fold[0])
        self.p[1]=self.p[1]+0.5*dt*(self.F[1]+self.Fold[1])
```

"Crystal" object

Definition of the object "Crystal"

[7]:
```
class Crystal:
    def __init__(self):
        self.atoms=numpy.array([])
        self.Epot=0.0
        self.Ek=0.0

# Method adding atoms into crystal

    def add_atom(self,r=numpy.array([0.0,0.0]),p=numpy.array([0.0,0.0]),mass=1.0):
        self.atoms=numpy.append(self.atoms,Atom(r,p,mass=mass))
    def rij(self,i,j):
        rx=self.atoms[j].r[0]-self.atoms[i].r[0]
        ry=self.atoms[j].r[1]-self.atoms[i].r[1]
        rij=numpy.sqrt(rx*rx+ry*ry)
        return rx,ry,rij
    def force(self):
        n_atoms=len(self.atoms)
        self.Epot=0.0
        self.Ek=0.0
        for atm in self.atoms:
            atm.Fold[0]=atm.F[0]
            atm.Fold[1]=atm.F[1]
            atm.F[0]=0.0
            atm.F[1]=0.0
        for i in range(n_atoms-1):
            for j in range(i+1,n_atoms):
                rx,ry,rij=self.rij(i,j)
                self.atoms[i].F[0]=self.atoms[i].F[0]+gradV(rij,rx)
                self.atoms[i].F[1]=self.atoms[i].F[1]+gradV(rij,ry)
                self.atoms[j].F[0]=self.atoms[j].F[0]-gradV(rij,rx)
                self.atoms[j].F[1]=self.atoms[j].F[1]-gradV(rij,ry)
                self.Epot=self.Epot+V(rij)
```

Buiding of a 2D "Crystal" of 5 atoms

[8]:
```
L=10.0                      # Size of the box in meter
m=60.0                      # mass of the atoms in kg
velocity=1.0                # km/h                        # vitesse de 5 km/h <=> 1.4 m/s <=> p=83 kg.m/s
pini=m*(velocity*1e3/3600)  # linear momentum
```

Positions of the atoms

[9]:
```
positions=numpy.array([[L/2,L/2],[L/4,L/4],[3*L/4,L/4],[L/4,3*L/4],[3*L/4,3*L/4]])
```

Building the initial crystal "Ag"

[10]:
```
Ag=Crystal()
```

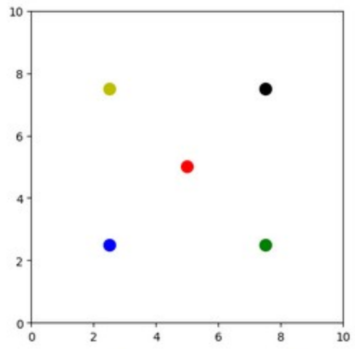
[11]:
```
for r in positions:
    px=pini*(random.random()-.5)
    py=pini*(random.random()-.5)
    Ag.add_atom(r=r,p=[px,py],mass=m)
```

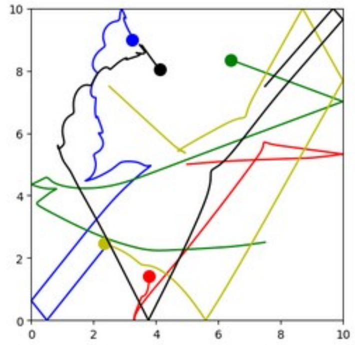
Computing initial forces

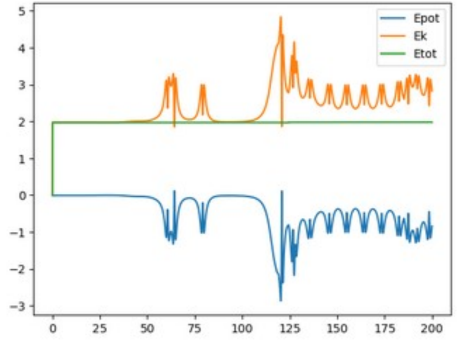
[12]:
```
Ag.force()

dt=.025
Epot=numpy.array([Ag.Epot])
Ek=numpy.array([Ag.Ek])
Etot=numpy.array([Ag.Epot+Ag.Ek])
temps=numpy.array([0.0])
```

Simulation of the movement of the atoms

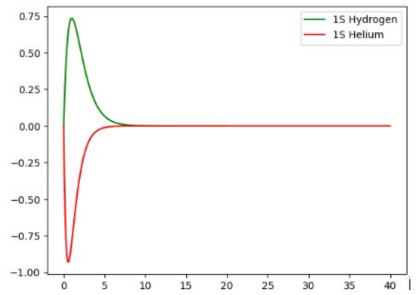
[13]:
```
number_total_of_steps=8000
freq=number_total_of_steps-1
for nstep in range(number_total_of_steps):
    for atm in Ag.atoms:
        atm.move(dt,L)
    Ag.force()

    for atm in Ag.atoms:
        atm.linear_momentum(dt)
        Ag.Ek=Ag.Ek+0.5*(atm.p[0]**2+atm.p[1]**2)/atm.m
    Epot=numpy.append(Epot,Ag.Epot)
    Ek=numpy.append(Ek,Ag.Ek)
    Etot=numpy.append(Etot,Ag.Epot+Ag.Ek)
    temps=numpy.append(temps,(nstep+1)*dt)
    if nstep%freq == 0:
        display(Ag.atoms)

#display(Cu.atoms)
fig = pyplot.figure()
ax = fig.add_subplot()
pyplot.plot(temps,Epot,label="Epot")
pyplot.plot(temps,Ek,label="Ek")
pyplot.plot(temps,Etot,label="Etot")
pyplot.legend()
pyplot.show()
```





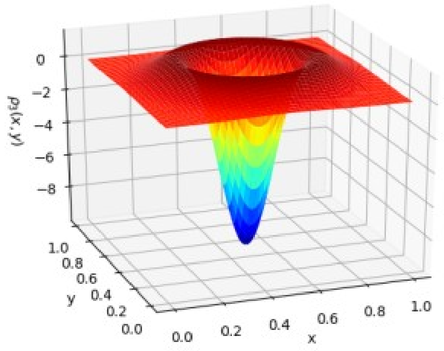


- Solving the one dimensional Schrödinger equation of the hydrogen and helium atom using the finite elements method

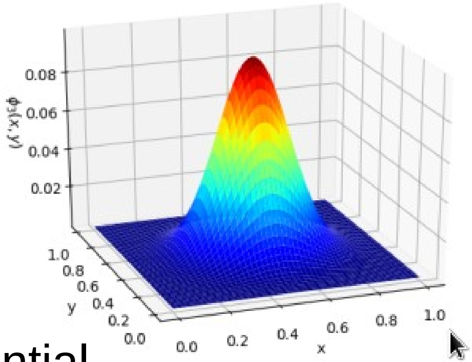


- Modelling of metallic nanoalloys at the atomic scale : chemical order/disorder transition
- Solving 2D Poisson equation by using Finite Difference Method

$$\nabla^2 \phi(\boldsymbol{r}) = -\frac{\rho(\boldsymbol{r})}{\varepsilon_0}.$$



Source



Potential