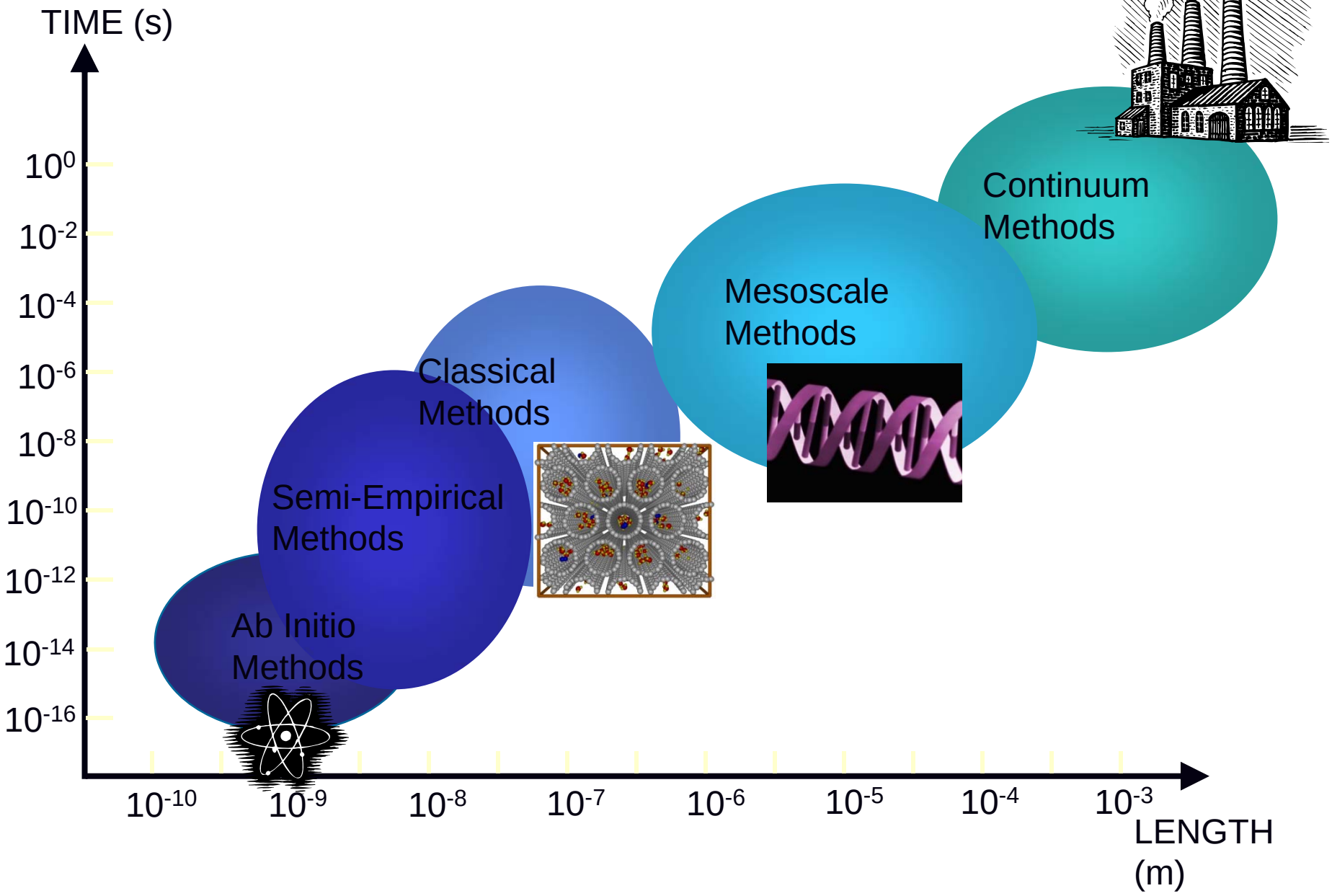
计算物理
- 分子动力学模拟
COMPUTATIONAL PHYSICS
- Molecular Dynamics Simulation

南京大学物理学院

SIMULATION SIZES and METHODS



Ab Initio and DFT Calculations (Quantum Mechanics)

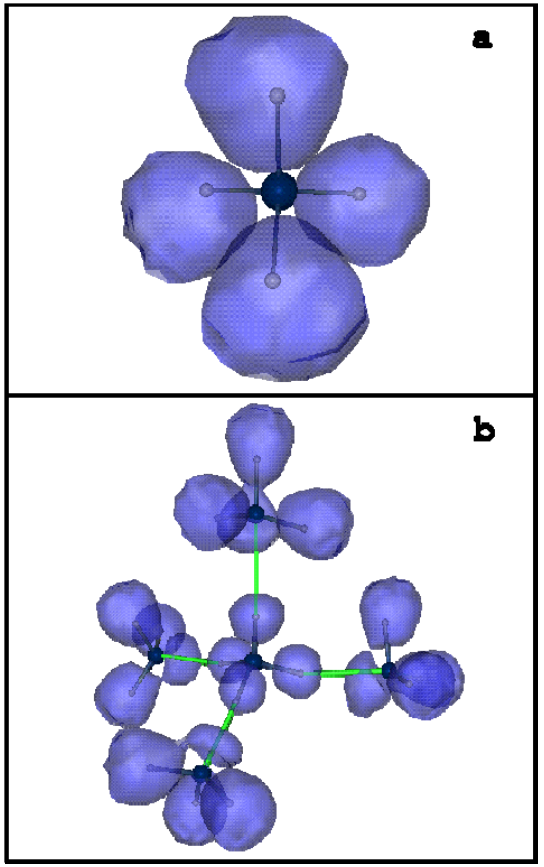
计算与电子自由度有关的原子性质

优点：

- 精确的计算基态能量
- 共价键，电荷转移

缺点：

- 计算量大，只能用于小体系，几百个原子
- 只能计算短时间尺度过程 <100 ps.
- 求解方程须作近似



Electron localization function for (a) an isolated ammonium ion and (b) an ammonium ion with its first solvation shell, from *ab initio* molecular dynamics. From Y. Liu, M.E. Tuckerman, *J. Phys. Chem. B* 105, 6598 (2001)

Semi-empirical Methods

包含部分经验参数，能处理电子自由度

优点：

共价键，电荷转移.

较大体系， $O(10^3)$ atoms.

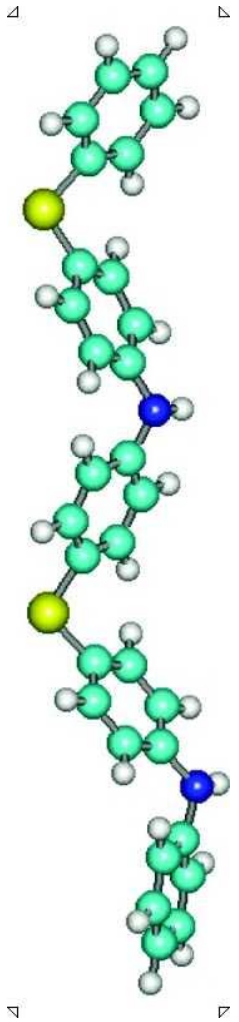
较长时间尺度过程， $O(10)$ ns.

如化学反应过程

缺点：

精度下降

需要输入参数

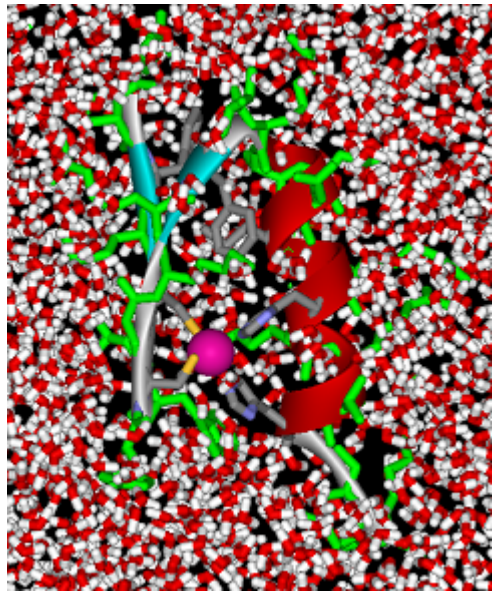


Structure of an oligomer of polyphenylene sulfide phenyleneamine obtained with the PM3 semiempirical method. From R. Giro, D.S. Galvão, *Int. J. Quant. Chem.* 95, 252 (2003)

Classical Molecular Simulations

使用经验力场，通过求解经典运动方程，

计算粒子运动规律以及体系的热力学性质。



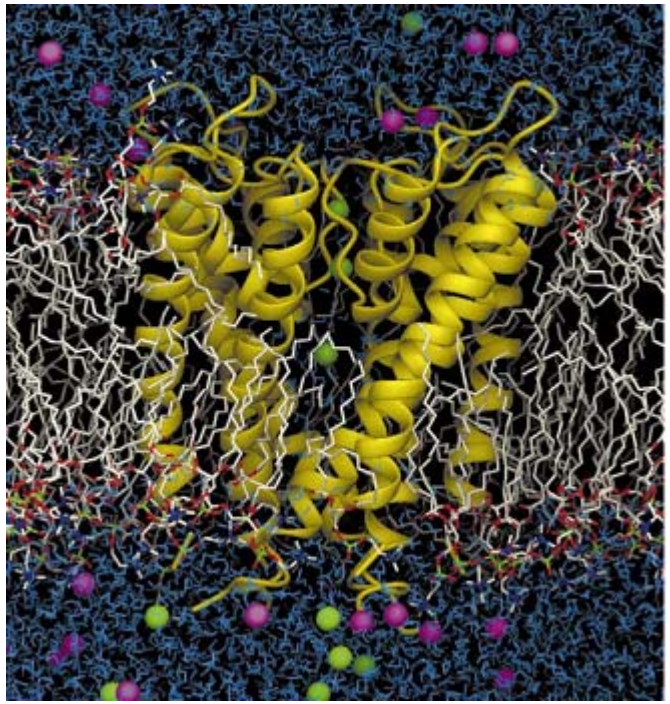
优点：

可以计算更大的复杂体系的结构、动力学与热力学性质, $10^4 – 10^6$ atoms.

可以计算更长时间尺度的动力学过程，μs～ms.

缺点：

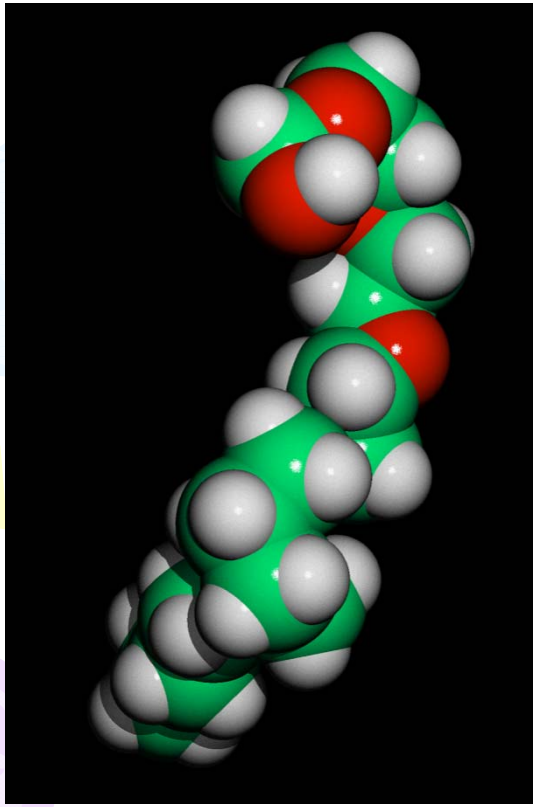
结果依赖与使用的力场参数



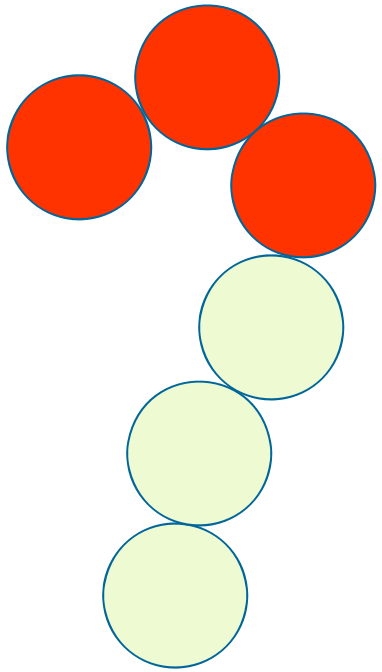
Properties: heat capacity, phase equilibrium, solvation, PVT behavior, diffusion coefficients, surface tension, solubility

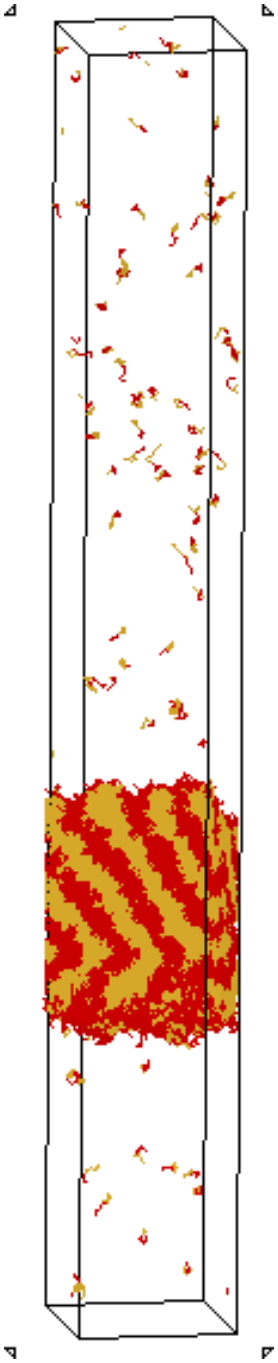
Mesoscale Methods

在全原子模型的基础上，对快自由度积分，
得到感兴趣的满自由度的运动规律。



Mapping





Phase equilibrium between a lamellar surfactant-rich phase and a continuous surfactant-poor phase in supercritical CO_2, from a lattice MC simulation. From N. Chennamsetty, K.E. Gubbins.

优点：

能研究更大的体系 O(10^{8-9}) atoms.
能研究更长时间尺度的动力学过程 up to O(1) s.

缺点：

只能给出定量的结果，可信度较差
由于做了较大的近似，限制了所适用的物理问题范围

Continuum Methods

认为体系实连续的，系统性质用场来描述；通过数值求解唯像方程来描述系统的性质。

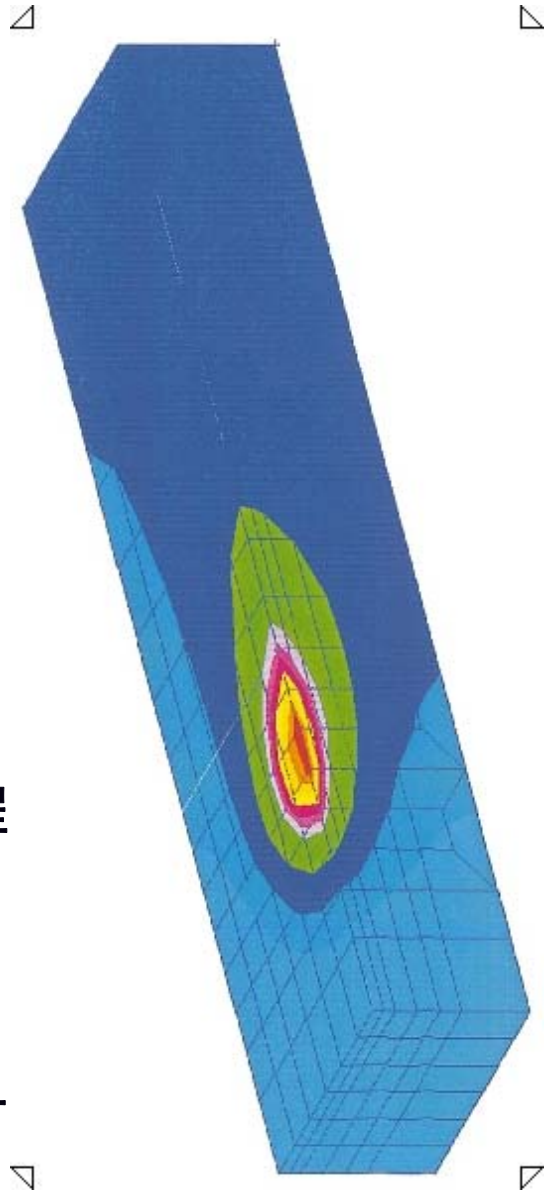
优点：

原则上可以研究任意大小任意长时间尺度的动力学过程

缺点：

需要输入基本的参数，如粘滞系数、扩散系数、状态方

不能解释涉及分子与电子自由度的现象

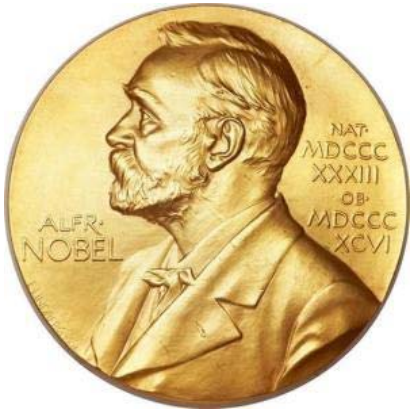


Temperature profile on a laser-heated surface obtained with the finite-element method. From S.M. Rajadhyaksha, P. Michaleris, *Int. J. Numer. Meth. Eng.* 47, 1807 (2000)

分子动力学模拟
Molecular Dynamics Simulation

1. 分子动力学是在原子、分子水平上求解多体问题的重要的计算机模拟方法。
2. 通过求解所有粒子的运动方程，分子动力学方法可以用于模拟与粒子运动路径相关的基本过程。
3. 一般而言，粒子的运动行为为经典的Newton运动方程所描述。
4. 和量子力学相结合的模拟 → QM/MM simulations

2013诺贝尔奖化学奖




马丁-卡普拉斯

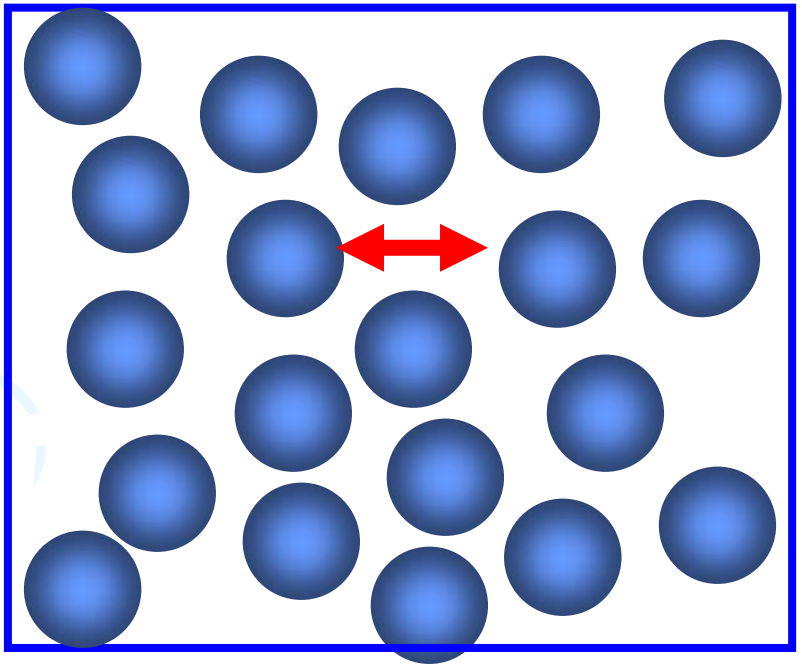

迈克尔-莱维特


亚利耶-瓦谢尔

Martin Karplus，Michael Levitt, and Arieh Warshel
奖励他们在“发展复杂化学体系多尺度模型”方面所做的贡献。

基本思路

根据粒子的运动规律(如牛顿方程）对系统中的每个粒子的运动方程进行数值求解，得到各个时刻每个粒子的坐标和动量(即相空间中的运动轨迹)，再利用统计计算方法得到体系的静态和动态特征。



MD的中心问题

Equation of motion:

$$F = ma$$

$$F = m\frac{d^2}{dt^2}x$$

$$\Rightarrow \quad x(t), v(t) \Rightarrow\Rightarrow A(t)$$

基本步骤

1.初始化

2.求力

3.积分运动方程

4.计算物理量

```
void mdrun()
{
    init()                          ！ 初始化
    t=0
    while (t < tmax)   {        ！ MD循环
       force(f)                     ！ 计算力
       integrate(f,x)               ！ 积分运动方程
       t=t+delt_t
       sample_calc_A()          ！ 抽样平均
    }
}
```

初始化

任务：给定体系各个粒子的初始坐标和动量

初始位置往往根据已知条件直接给出

初始速度需要根据模拟温度抽样给出

1. 按Maxwell分布
2. 速度重标度

麦克斯韦速度分布

$$f(\vec{v})d\vec{v} = (\frac{m}{2\pi k_B T})^{3/2} e^{-\frac{\beta m}{2}(v_x^2+v_y^2+v_z^2)} dv_x dv_y dv_z$$

服从参数为 μ 和 σ 的正态分布 $N(\mu, \sigma^2)$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \quad -\infty < x < \infty$$

于是每个方向速度分量满足方差为 $\sigma^2 = k_B T / m$ 的正态分布。

具体步骤：

1）生成方差为1的标准正态分布。

2）所得随机数乘以方差的平方根，即乘以 $\sigma = \sqrt{k_B T / m}$

于是：$\overline{v^2} = 3\overline{v_x^2} = 3\sigma^2 = 3k_B T / m$ **(对于二维，3→2)**

Python program

```
NP = 100  # number of particles
T0 = 100.0 # temperature
vx = np.zeros(NP)
vy = np.zeros(NP)
def init():
    global vx,vy

    vx = np.random.randn(NP) * np.sqrt(T0)
    vy = np.random.randn(NP) * np.sqrt(T0)
    vz = …

    return
```

$$\sigma = \sqrt{k_B T / m} = \sqrt{T}$$

(when $kB = 1,\ T = 1$)

速度重标度方法

$$\overline{E_k} = \frac{3}{2}k_B T = \frac{1}{N}\frac{1}{2}\sum_{i=1}^{N} m_i v_i^2$$

v_i可以随机赋值

$$\Rightarrow \quad T = \sum_i m_i v_i^2 \,/\, 3Nk_B$$

(对于二维，3→2)

Velocity-Scaling: $\lambda = \left(T_0 / T\right)^{1/2}$

$$T_0 = \lambda^2 T$$

初始速度： $v_0 = \lambda \cdot v$

```
# NP is the number of particles
# T0 is the target temperature
…
vx = np.random.random(NP) - 0.5
vy = np.random.random(NP) - 0.5
…
v2 = 0.0
for i in range(NP):
    v2 += vx[i]*vx[i]+vy[i]*vy[i]
T = 0.5* v2 / NP
lamda = np.sqrt(T0/T )
vx *= lamda
vy *= lamda
```

$$T = \sum_i m_i v_i^2 / 3Nk_B$$

$$\lambda = \left(T_0 / T\right)^{1/2}$$

$$v_0 = \lambda \cdot v$$

积分运动方程

已知初始速度，坐标和力，通过数值求解运动方程来求各时刻每个粒子的坐标和速度。

$$\vec{F}_i = -\nabla_i U = m_i \frac{d^2}{dt^2}\vec{r}_i$$

- **Verlet**
- **Velocity Verlet**
- **Leapfrog**
- **…**

Verlet rule

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$$

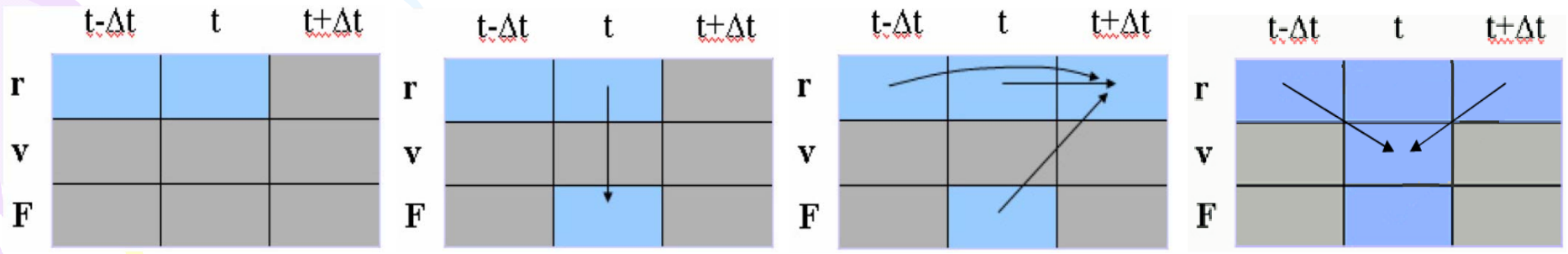
$$r(t-\Delta t) = r(t) - v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$$

add them together

$$\Rightarrow \quad r(t+\Delta t) = 2r(t) - r(t-\Delta t) + \Delta t^2 a(t), \qquad (a = F/m)$$

$$v(t) = [r(t+\Delta t) - r(t-\Delta t)]/2\Delta t$$

- 已知t-△t和t时刻的位置
- 由t时刻位置，可得t时刻的受力
- 由t时刻的受力及t-△t和t时刻的位置，可得t+△t时刻的位置
- 由t+△t及t-△t时刻的位置，可得t时刻的速度



- **Advantages:**
  - **1.** Integration does not require the velocities, only position information is taken into account.
  - **2.** Only a single force evaluation per integration cycle. (Force evaluation is the most computationally expensive part in the simulation).
  - **3.** This formulation, which is based on forward and backward expansions, is naturally reversible in time (a property of the equation of motion).

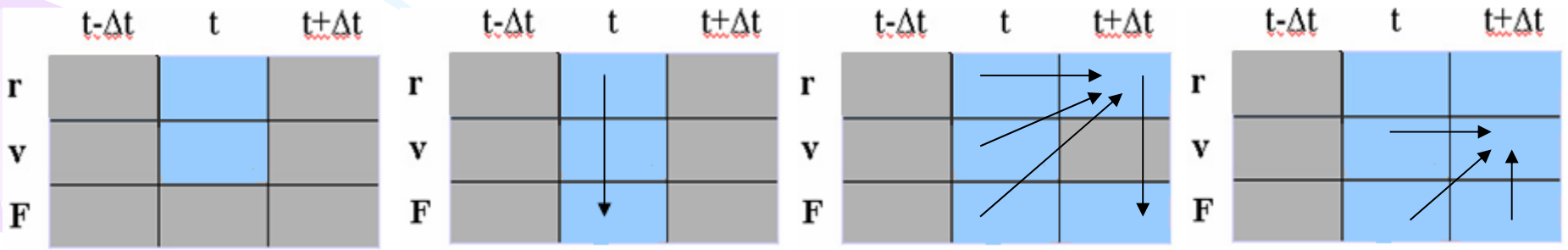
- **Disadvantages:**
  - The velocities, which are required for energy evaluation are calculated in an approximate manner only through the equation: **vn** = (**rn+1 - rn-1**)/*2 dt* . (large errors)
  - Need to know **rn+1** to calculate **vn! Bad, bad, bad,….**

Velocity Verlet rule

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}\Delta t^2 a(t)$$

$$v(t+\Delta t) = v(t) + \frac{1}{2}\Delta t[a(t) + a(t+\Delta t)]$$ 改进的欧拉公式

- 已知t时刻位置、速度
- 由t时刻的位置，可得t时候受力
- 由t时刻的受力和位置、速度，可得t+△t的位置和受力
- 由t+△t的受力和t时刻的速度和受力，可得t+△t时刻的速度

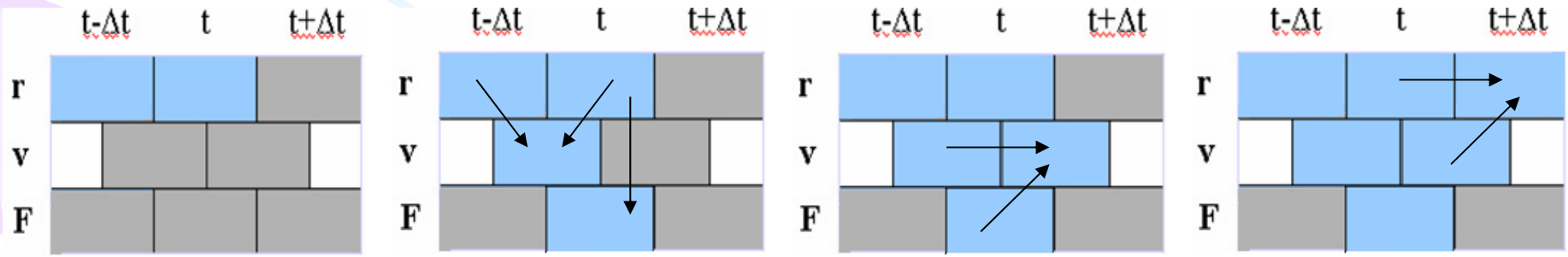


Leap frog rule

$$v_{n-1/2} = v(t - \frac{\Delta t}{2}) = \frac{r(t) - r(t-\Delta t)}{\Delta t} = \frac{r_n - r_{n-1}}{\Delta t}$$

$$v_{n+1} = v_{n-1/2} + a_n \Delta t \qquad r_{n+1} = r_n + v_{n+1/2} \Delta t$$

- 已知t-△t和t时刻的位置
- 由t-△t和t时刻的位置，可得t-△t/2时刻的速度和t时刻的受力
- 由t时刻的受力和t- △t/2时刻的速度，可得t+△t/2时刻的速度
- 由t+△t/2时刻的速度和t时刻的位置，可得t+△t时刻的位置



- **Advantages:**
  - **1.** Improved evaluation of velocities.
  - **2.** Direct evaluation of velocities gives a useful handle for controlling the temperature in the simulation.

- **Disadvantages:**
  - **1.** The velocities at time *t* are still approximate.
  - **2.** Computationally a little more expensive than Verlet.
  - 3. We cannot calculate potential energy and kinetic energy simultaneously.

Leap frog algorithm

while(istep<nstep):

force() # → fx,fy

vx += dt*fx

vy += dt*fy

$$v_{n+1} = v_{n-1/2} + a_n \Delta t$$

rx += vx*dt

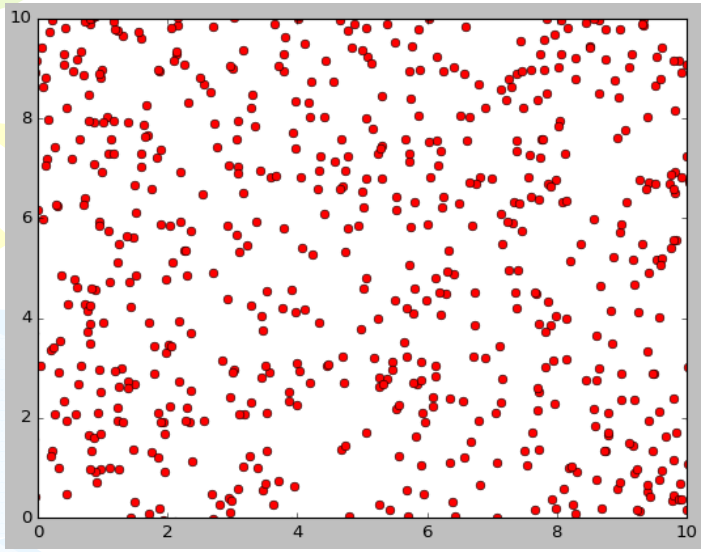
ry += vy*dt

$$r_{n+1} = r_n + v_{n+1/2} \Delta t$$

…

能量和力的计算

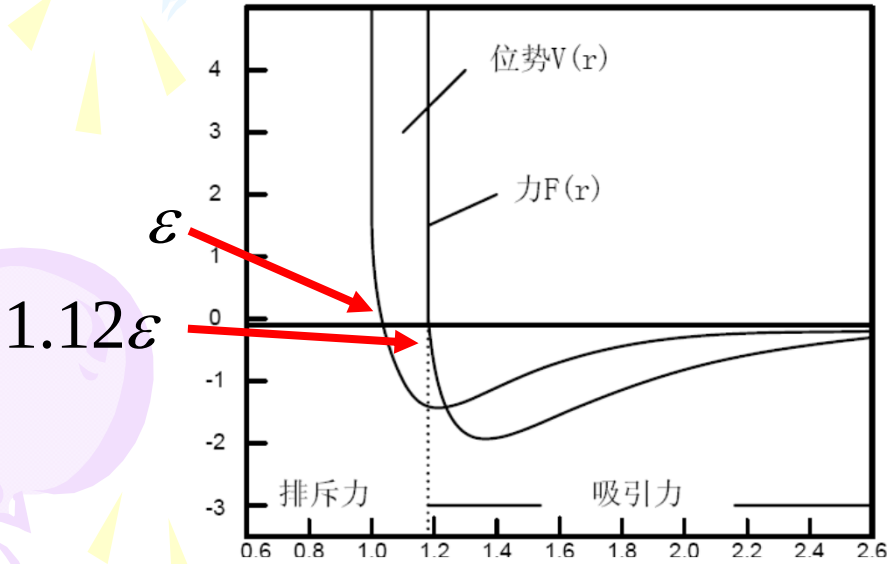
以范德华气体系统为例



N V T 系综

相互作用势：

$$U(r)=U_0\left[\left(\frac{\varepsilon}{r}\right)^{12}-\left(\frac{\varepsilon}{r}\right)^{6}\right]$$



$$\frac{\partial}{\partial r}U(r)=0 \quad \Rightarrow$$

$$r_0=2^{1/6}\varepsilon\approx 1.12\varepsilon$$

为粒子对平衡位置

For i in range(NP-1):

For j in range(i+1,NP):

$$U(r)=U_0\left[\left(\frac{\varepsilon}{r}\right)^{12}-\left(\frac{\varepsilon}{r}\right)^{6}\right]$$

$$U_{ij}=U_0\left[\left(\frac{\varepsilon}{r_{ij}}\right)^{12}-\left(\frac{\varepsilon}{r_{ij}}\right)^{6}\right]$$

$$f^{x}_{i\leftarrow j}=-\frac{\partial}{\partial x_i}U_0\left[\left(\frac{\varepsilon}{r_{ij}}\right)^{12}-\left(\frac{\varepsilon}{r_{ij}}\right)^{6}\right]$$

$$f^{x}_{i\leftarrow j}=U_0\left[12\left(\frac{\varepsilon}{r_{ij}}\right)^{12}-6\left(\frac{\varepsilon}{r_{ij}}\right)^{6}\right]\frac{x_i-x_j}{r_{ij}^2}$$

$$f^{x}_{j\leftarrow i}=-f^{x}_{i\leftarrow j}$$

$$f^{y}_{i\leftarrow j}=U_0\left[12\left(\frac{\varepsilon}{r_{ij}}\right)^{12}-6\left(\frac{\varepsilon}{r_{ij}}\right)^{6}\right]\frac{y_i-y_j}{r_{ij}^2}$$

$$f^{y}_{j\leftarrow i}=-f^{y}_{i\leftarrow j}$$

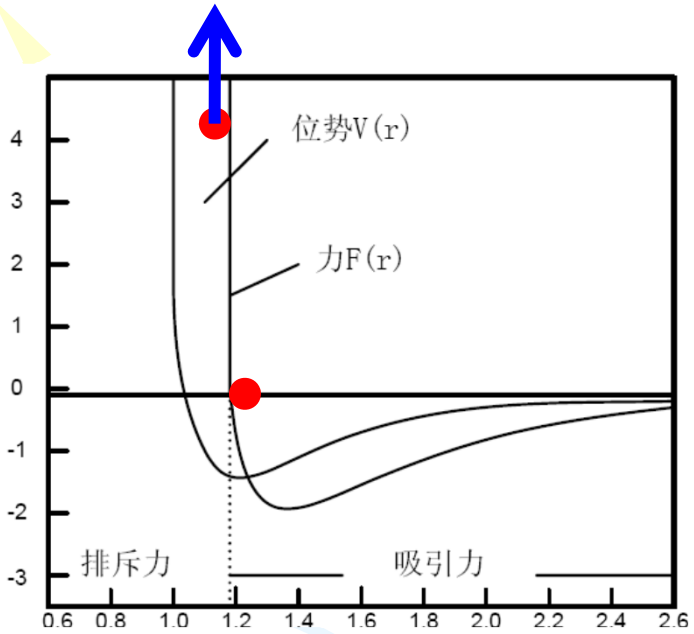
暂时考虑2D情况

```
def force():
    global fx,fy
    fx = np.zeros(NP)
    fy = np.zeros(NP)
    for i in range(NP-1):
        for j in range(i+1,NP):
            dx = rx[i]-rx[j]
            dy = ry[i]-ry[j]
            dr2 = dx*dx+dy*dy
            if( dr2>r2cut ):  continue
            dr6 = dr2*dr2*dr2
            dr12 = dr6*dr6
            fc = (12.0*epsilon12/dr12 - 6.0*epsilon6/dr6)/dr2
            fx1 = fc*dx
            fy1 = fc*dy
            fx[i] += fx1
            fy[i] += fy1
            fx[j] += -fx1
            fy[j] += -fy1
```

$$f_{i\leftarrow j}^{x} = U_0\left[12\left(\frac{\varepsilon}{r_{ij}}\right)^{12} - 6\left(\frac{\varepsilon}{r_{ij}}\right)^{6}\right]\frac{x_i - x_j}{r_{ij}^2}$$

Handling Exceptions

Very very large force



- Very very … large force and thus velocities
- It may happen when delta-t is not small enough
- or simply due to bad luck
- It will break the whole system

Handling Exceptions

Velocity Verlet rule

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}\Delta t^2 a(t)$$

$$v(t+\Delta t) = v(t) + \frac{1}{2}\Delta t[a(t) + a(t+\Delta t)]$$

```
force()
…
if (force>fcutoff):
        force = fcutoff
```

Handling Exceptions

Leap frog rule

$$v_{n+1} = v_{n-1/2} + a_n \Delta t$$

$$r_{n+1} = r_n + v_{n+1/2} \Delta t$$

Very important!

Otherwise, MD will not work, particularly when delta-t is relatively large.

```
def mdrun():
    global rx,ry,vx,vy,fx,fy,T0,dt,time,istep

    force()

    vx += dt*fx
    vy += dt*fy
```

$$v_{n+1} = v_{n-1/2} + a_n \Delta t$$

```
    for i in range(NP):
        if( abs( vx[i] ) > vcut ):
            vx[i] = abs(vx[i])/vx[i] * vcut
        if( abs( vy[i] ) > vcut ):
            vy[i] = abs(vy[i])/vy[i] * vcut

    rx += vx*dt
    ry += vy*dt
```

$$r_{n+1} = r_n + v_{n+1/2} \Delta t$$

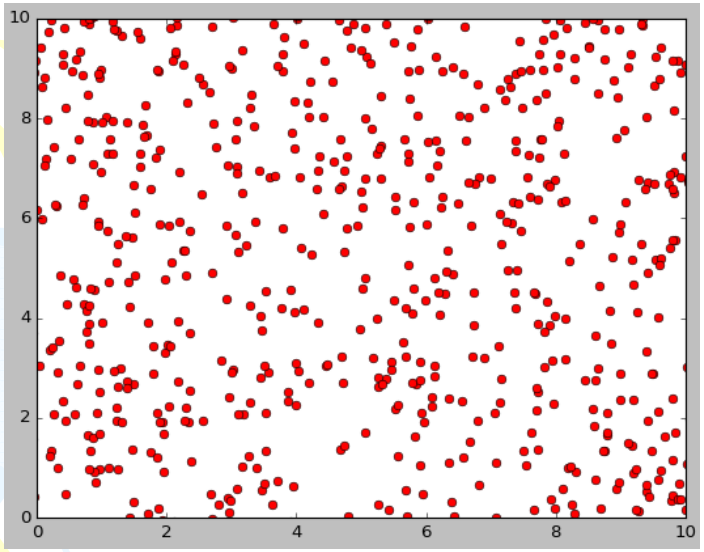
短程力的加速算法

1) 设定一个距离cutoff，超过的不计算。
2) 对于每一粒子*i*，维护一个近邻列表。

Neighbor list:

- 使用一个较大的cutoff，如cutoff+skin
- 每隔10-20步更新一次列表
- 记录每个原子的位移，当超过skin时更新list
- 进一步：grid searching

边 界 条 件

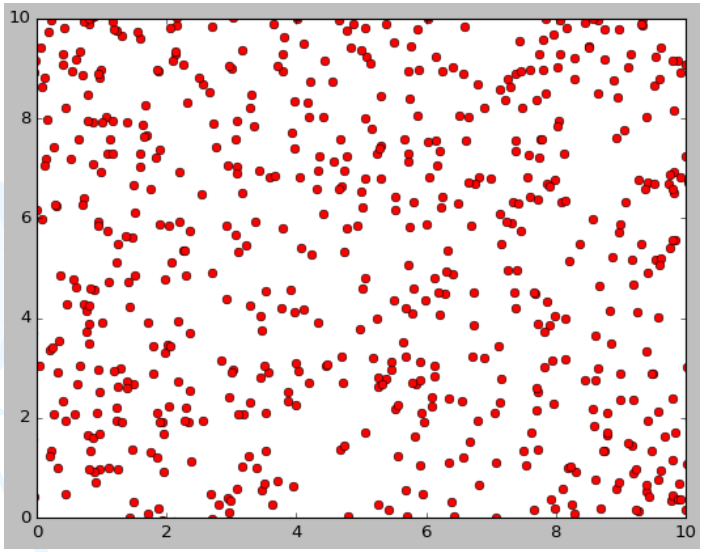


非周期：

- 软墙
- 硬墙

```
for i in range(NP):
    if( rx[i]<0 or rx[i]>L ):
        vx[i] = -vx[i]
    if( ry[i]<0 or ry[i]>L ):
        vy[i] = -vy[i]
```

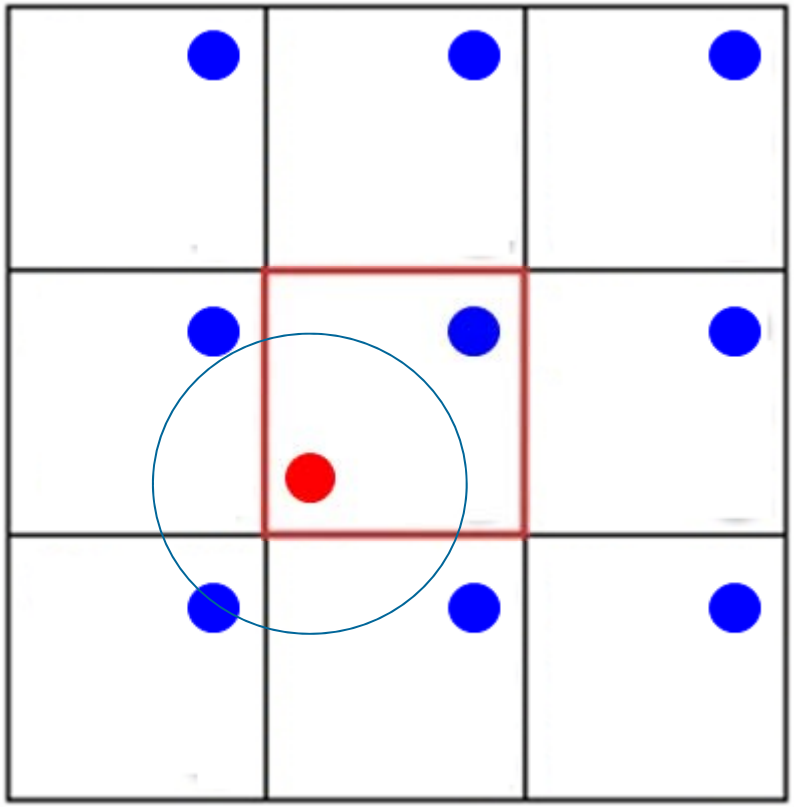
周期边界条件（PBC）
Periodic boundary condition



对x方向：

```
if( x[i]<0 ):
    x[i] += L
elif(x[i]>L):
    x[i] -= L
```

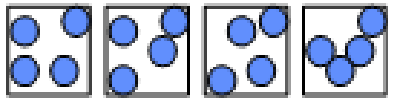
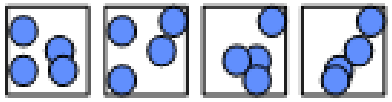
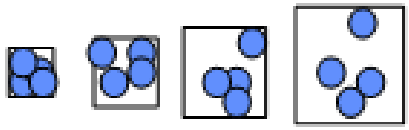
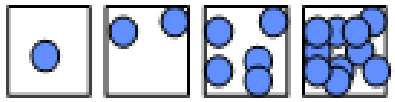
周期边界条件（PBC）



如何计算距离？

- 选择9个镜像中最近的一个
- 对于3维情况，27个镜像

平衡态分子动力学模拟

Name	Probability distribution	Schematic
Microcanonical (EVN)	$\pi_i = \frac{1}{\Omega}$	
Canonical (TVN)	$\pi(E_i) = \frac{1}{Q} e^{-\beta E_i}$	
Isothermal-isobaric (TPN)	$\pi(E_i, V_i) = \frac{1}{\Delta} e^{-\beta(E_i + PV_i)}$	
Grand-canonical (TVμ)	$\pi(E_i, N_i) = \frac{1}{\Xi} e^{-\beta(E_i + \mu N_i)}$	

等温过程模拟

- Velocity-scaling
- Berendsen thermostat
- Andersen thermostat
- Brownian dynamics
- Nose-Hoover's method

Velocity-scaling

粒子的动能：

$$E_k = \frac{1}{2}\sum_i m_i v_i^{\ 2} \longrightarrow T = \frac{\sum_i m_i v_i^2}{3Nk_B}$$

$$E_k = \frac{3}{2} N K_B T$$

Velocity-Scaling:

$$T_0 = \lambda^2 T$$

$$v_{new} = \lambda \cdot v$$

Berendsen thermostat (weak coupling)

$$\frac{dT(t)}{dt} = \frac{1}{\tau}(T_0 - T) \rightarrow \Delta T = \frac{\Delta t}{\tau}(T_0 - T)$$

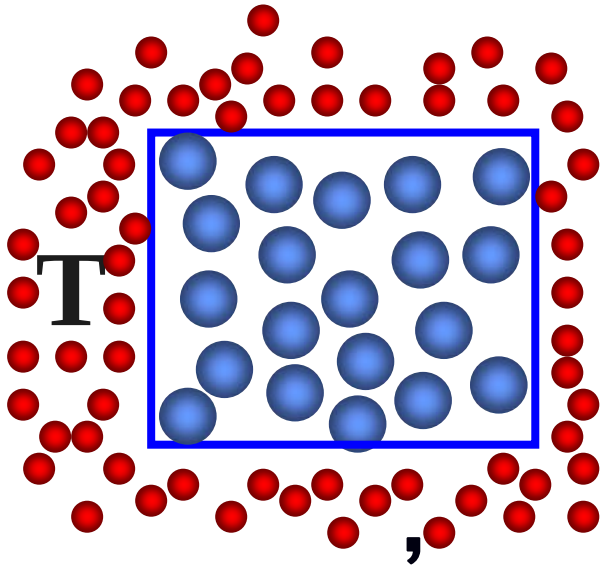
$$\lambda^2 = \frac{T + \Delta T}{T} = 1 + \frac{\Delta T}{T} = 1 + \frac{\Delta t}{\tau}\left(\frac{T_0}{T} - 1\right)$$

$$\rightarrow v(t) = \lambda v(t)$$

- **when τ=Δt, back to velocity scaling**
- **Problem? Non-physical, hot-spot**

Berendsen et al. JCP, 81:3684(1984)

Andersen热浴



步骤:

- 粒子和虚粒子相互碰撞
- 对于每一个粒子，如果满足

$$rand < v \bullet dt$$

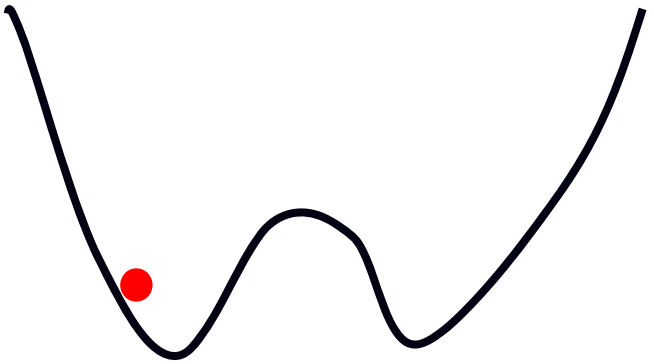
其中rand为[0,1)之间随机数，*v*为碰撞频率(需人为指定)，*dt* 为模拟时间步长，则：

- 重置此粒子速度。新的速度由Maxwell分布抽样获得，Maxwell分布的对应温度为目标温度。

Andersen, JCP,72,2384(1980)

Brownian dynamics

$$m\frac{d^2x}{dt^2} = F = F(x) - \gamma\frac{dx}{dt} + F_R$$



高斯白噪声：

$$< F_R(\mathrm{t}) >= 0; \quad < F_R(\mathrm{t}) \cdot F_R(\mathrm{t}') >= 2\,\mathrm{k_B T}\,\gamma\delta(\mathrm{t}-\mathrm{t}')$$

$$F_n = \frac{1}{\gamma}F(\mathrm{x}_n) - \gamma v_n + \sqrt{\frac{2\gamma T}{\Delta t}} \cdot \Gamma_{normal}$$

→ **标准正态分布随机数**

Nose-Hoover’s extended system method

See the Wikipedia

等压过程 (Volume-scaling)

$$\frac{dP(t)}{dt}=\frac{1}{\tau_P}(P_{bath}-P(t)) \qquad \Delta P=\frac{\Delta t}{\tau_P}(P_{bath}-P(t))$$

等温压缩系数 $\kappa=-\frac{1}{V}\frac{\Delta V}{\Delta P}$ $\Delta V=-\kappa V\Delta P$

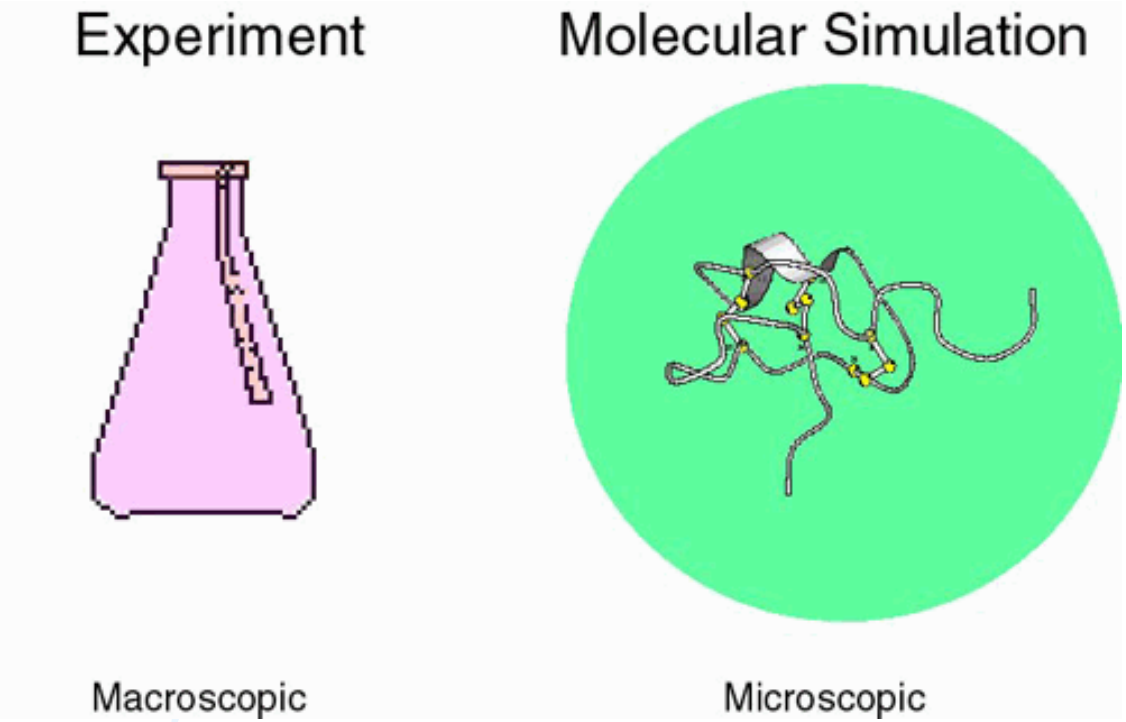
$$\Delta V=-\kappa V\Delta P=-\kappa V\frac{\Delta t}{\tau_P}(P_{bath}-P(t))$$

定义： $\lambda^3=\frac{V+\Delta V}{V}=1+\frac{\Delta V}{V}=1-\kappa\frac{\Delta t}{\tau_P}(P_{bath}-P)$

于是， $r_i^{new}=\lambda r_i$

其中：压强由Virial公式给出 $P=\frac{1}{V}\left(Nk_BT-\frac{1}{3}\sum_{i<j}r_{ij}f_{ij}\right)$

计算物理观测量


Experiment
Molecular Simulation
Macroscopic
Microscopic

遍历性假设

热力学 (系综)平均：某一时刻系综中所有系统的平均

动力学平均(时间)平均:某一系统在所有时间的平均

遍历性假设：

对无限长的轨迹，热力学平均＝动力学平均

或　　时间平均＝系综平均

Ensemble average:

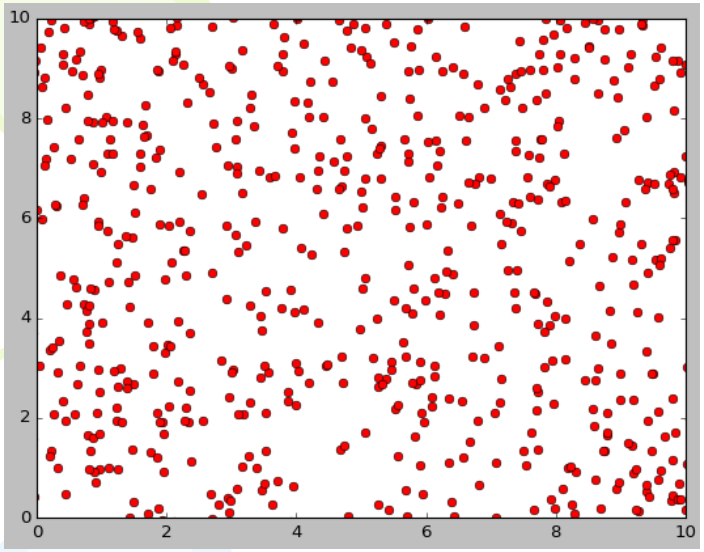
$$< A >_{ens} = \frac{\int A(p^N, r^N) \exp\left[-\beta H(p^N, r^N)\right] dp^N dr^N}{\int \exp\left[-\beta H(p^N, r^N)\right] dp^N dr^N}$$

Time average:

$$< A >_{time} = \lim_{M \to \infty} \frac{1}{M} \sum_{t=1}^{M} A(p^N, r^N)$$

假设二者相等，要求模拟时间够长，系统达到平衡。

Program time



N V T 系综

相互作用势：

$$U(r) = U_0\left[\left(\frac{\varepsilon}{r}\right)^{12} - \left(\frac{\varepsilon}{r}\right)^{6}\right]$$

```
NP = 100 # number of particles
L = 100.0 #length of the box
kB = 1.0
T0 = 100.0 # temperature
U0 = 1.0 # force coefficient
epsilon = 1.0 # equilibrium dist.

dt = 0.1 # time step
dt2 = dt*dt
time = 0.0
istep = 0

r2cut = (epsilon * 2)**2
vcut = L/50/dt
tcouple = dt # Berendsen thermostat
```

```
from matplotlib import pyplot as plt
from matplotlib import animation
rx = np.random.random(NP)*L
ry = np.random.random(NP)*L
#这里定义vx,vy,fx,fy等全局变量，均为长度为NP的数组
…
def animate(i):
    mdrun()  #update rx,ry
    line.set_data(rx, ry)
    if(i%10==0): output()
    return line

fig = plt.figure()
ax1 = fig.add_subplot(1,1,1,xlim=(0, int(L)), ylim=(0,int(L)))
line, = ax1.plot(rx, ry, 'ro')

mdinit()
anim1=animation.FuncAnimation(fig, animate, frames=1000,
          interval=1)
plt.show()
```

```
def mdrun():

  global rx,ry,vx,vy, vcut,fx,fy,T0,T,dt,time,istep

  force()  # updata fx,fy

  vx += dt*fx
  vy += dt*fy

  for i in range(NP):  # very important to control velocity
     if( abs( vx[i] ) > vcut ):
        vx[i] = abs(vx[i])/vx[i] * vcut
     if( abs( vy[i] ) > vcut ):
        vy[i] = abs(vy[i])/vy[i] * vcut

  rx += vx*dt
  ry += vy*dt
```

Leap frog

```
v2 = 0.0
for i in range(NP):
    v2 += vx[i]*vx[i]+vy[i]*vy[i]
T = 0.5* v2 / NP  #two-dimension
# Berendsen thermostat
lamda = np.sqrt( 1.0 + dt/tcouple*(T0/T-1.0) )
vx *= lamda
vy *= lamda

# Boundary control, hard wall
for i in range(NP):
    if( rx[i]<0 or rx[i]>L ):
        vx[i] = -vx[i]
    if( ry[i]<0 or ry[i]>L ):
        vy[i] = -vy[i]

istep += 1
time = time + dt
return
```

```
def force():
    global  rx,ry,vx,vy,fx,fy,T,r2cut,xi
    fx = np.zeros(NP)
    fy = np.zeros(NP)
    for i in range(NP-1):
        for j in range(i+1,NP):
            dx = rx[i]-rx[j]
            dy = ry[i]-ry[j]
            dr2 = dx*dx+dy*dy
            if( dr2>r2cut ): continue
            dr6 = dr2*dr2*dr2
            dr12 = dr6*dr6
            fc = (12.0*epsilon12/dr12 - 6.0*epsilon6/dr6)/dr2
            fx1 = fc*dx
            fy1 = fc*dy
            fx[i] += fx1
            fy[i] += fy1
            fx[j] += -fx1
            fy[j] += -fy1
    return
```

课后作业

完成对一箱vdw气体的分子动力学模拟，要求：

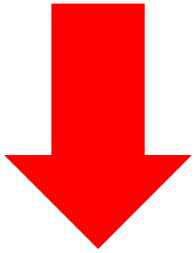
- 取NVT系综
- Leap frog rule (or velocity Verlet)
- Berendsen thermostat 或其它
- 边界条件：硬墙、软墙、或者PBC
- 作出系统构象随时间演化的动画
- 以及温度(或其它物理量，如动能、势能、压强)随时间的变化
- 亦可讨论不同参数下系统的行为

递交程序和相关PDF文件，python程序作为附件发送。

规范邮件标题：作业9-姓名-学号

下周上课前，发送至： njuphyhw@126.com

以上为一个 Toy 版
分子动力学模拟



E N D