计算材料第一次作业

王宇祺 武逸飞 张抱朴 张锦程

目录：

计算材料第一次作业

目录：

1. 题目一：使用 C 语言对 md2 体系进行简单热力学分析
    1.1 概要
    1.2 计算实现
        1.2.1 量纲分析
        1.2.2 热力学计算
    1.3 模拟结果
        1.3.1 原子平均距离分析
        1.3.2 初始时刻体系能量变化分析
        1.3.3 平衡时刻体系能量分析
    1.4 md1与md2对比
2. 借助 Python 开源拓展库对 md2 体系进行可视化处理
    2.1 概要
    2.2 代码实现
        2.2.1 初始化 & 弛豫
        2.2.2 粒子加速度、平均势能和平均距离的计算
        2.2.3 粒子平均动能与平均能量的计算
        2.2.4 主程序结构
    2.3 结果
3. 题目二：计算Ar气凝结为固体时的单空位形成能和双空位结合能
    3.1 单空位形成能
    3.2 双空位结合能
4. 补充：使用 lammps 并行版本对 md2 进行重制
    4.1 概要
    4.2 in文件设置
        4.2.1 units & atom_style 设置
        4.2.2 boundary设置
        4.2.3 lattice & region 设置
        4.2.4pair_style & pair_coeff 设置
        4.2.5compute 设置
        4.2.6 thermo & thermo_style & dump 设置
        4.2.7 velocity & timestep & fix & run设置
    4.3 结果分析

1. 题目一：使用 C 语言对 md2 体系进行简单热力学分析

1.1 概要

首先我们将在原有的 C 语言代码基础上略加改进，对液态 Ar 系统进行粗略的分子动力学计算，模拟计算原子的平均势能、总能、平均距离、体系温度在原子间相互作用力下随时间的变化，并比较上述物理量随系统初始温度改变的变化情况。氩原子的势函数采用了经典的 Lennard-Jones 对势：

$$V(r) = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6})]$$

故粒子之间的作用力为：

$$F(r) = -\frac{\partial V}{\partial r} = \frac{24\epsilon}{\sigma}[2(\frac{\sigma}{r})^{13} - (\frac{\sigma}{r})^{7}]$$

其中 r 是两个氩原子核之间的距离，ϵ 是势能大小的量度，σ 是两个原子势能为 0 时原子核的距离。各参数的数值参考 A.Rahman 使用的数据，取 $\epsilon = 1.65 \times 10^{-21}\text{J}$，$\sigma = 3.4\ \text{Å}$，Ar 原子质量 $m = 6.69 \times 10^{-26}\text{kg}$。为降低计算难度，计算过程中取 ϵ, σ, m, t 为1的无量纲量。由时间的量纲 $t = \sqrt{\frac{kg \cdot m^2}{J}}$，知单位时间为 $2.17 \times 10^{-12}s$。最终的模拟结果根据量纲代入对单位对应的真实数据即可。

1.2 计算实现

1.2.1 量纲分析

我们对原始的md2.cpp程序做了两方面的修改，第一方面是在原子数密度rho和温度T做出的数值修改，根据原始程序的 $N = 1, rho = 1$ 计算液Ar的密度：

$$\rho = \frac{Nm}{(\sqrt[3]{\frac{N}{rho}}\sigma)^3} \approx 1.7 \times g \cdot cm^{-3}$$

此密度比 A.Rahman 使用的 $94.4K$ 下的 $1.374g \cdot cm^{-3}$ 和$130K$下的 $1.16g \cdot cm^{-3}$ 都要大，因此系统的模拟温度应当比 $94.4K$ 要低。

根据量纲分析法，带入模拟条件下的距离单位、时间单位、质量单位，可计算出 $T = 1$ 对应的实际温度应为：

$$T = \frac{kg \times (m \cdot s^{-1})^2}{J \cdot K^{-1}} = \frac{6.69 \times 10^{-26} kg \times (156.7m \cdot s^{-1})^2}{10^{-23} J \cdot K^{-1}} \approx 119K$$

为了与真实实验形成参照，本模拟选用了 Rahman 的实验条件，因此我们将更改 rho 和 T 的数值，使系统密度和温度符合 A.Rahman 采取的 $94.4K$ 下的$1.374g \cdot cm^{-3}$ 和 $130K$ 下的$1.16g \cdot cm^{-3}$

经计算，$rho = 0.807$，$T = 0.793$ 和 $rho = 0.682$，$T = 1.092$ 时分别对应于上述两种情况，两种情况下的立方体棱长均通过$\sqrt[3]{\frac{N}{rho}}$计算。

1.2.2 热力学计算

第二方面是添加了计算原子平均势能、总能和平均距离的功能。在计算平均势能时，为了与计算作用力时的思路保持一致，我们也采取了镜像法，即如果两个原子在某一方向上的距离大于立方体晶格的一半，则计算作用力时考虑另一个立方体系统中的镜像原子，使两原子在此方向上的距离小于立方体晶格的一半。具体代码如下：

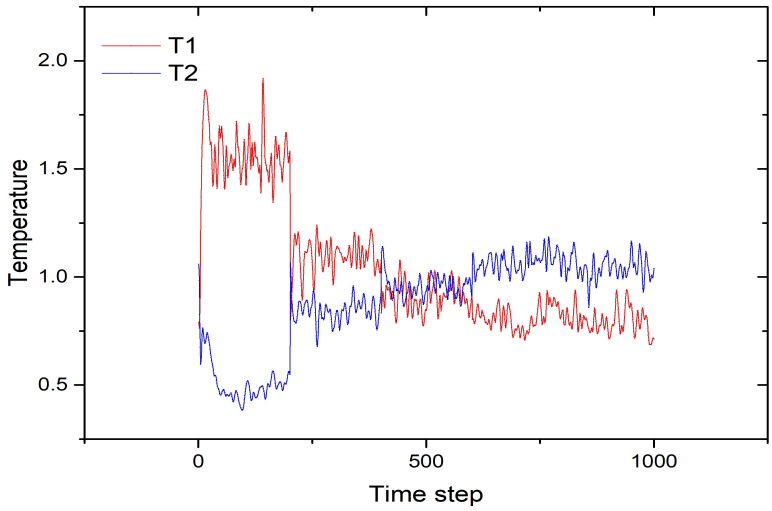
```
            if (abs(rij[k]) > 0.5 * L) {
                if (rij[k] > 0)
                    rij[k] -= L;
                else
                    rij[k] += L;}
            rSqd += rij[k] * rij[k];
```

原子的总能量等于势能与动能之和，平均总能通过平均势能与动能之和得到。平均动能由 $\frac{1}{2}\sum_{i=1}^{N}\sum_{\alpha=x,y,z}v_{\alpha}^{2}$ 计算,将平均动能与平均势能的数值相加即可得到平均总能。

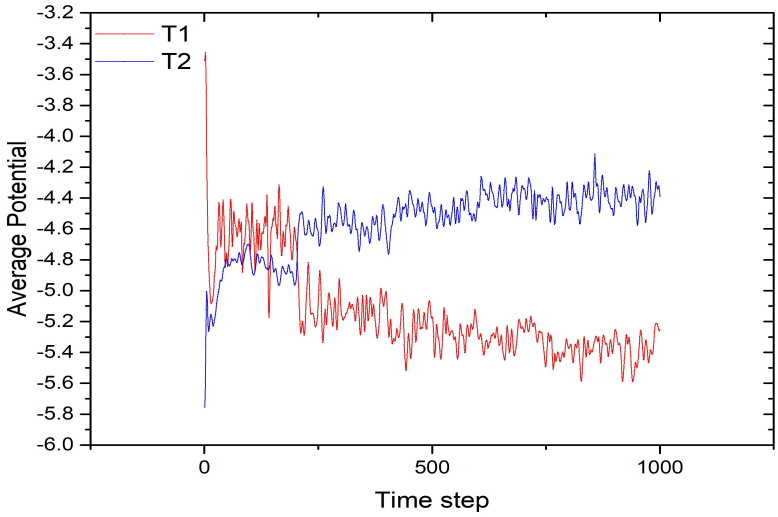
计算平均距离时，我们并没有采取镜像法。在计算作用力和平均势能时采取镜像法的原因是距离更近的原子对有更显著的相互作用，但计算距离时原子核之间的长距离和短距离是平权的，因此不能采用镜像法。

1.3 模拟结果

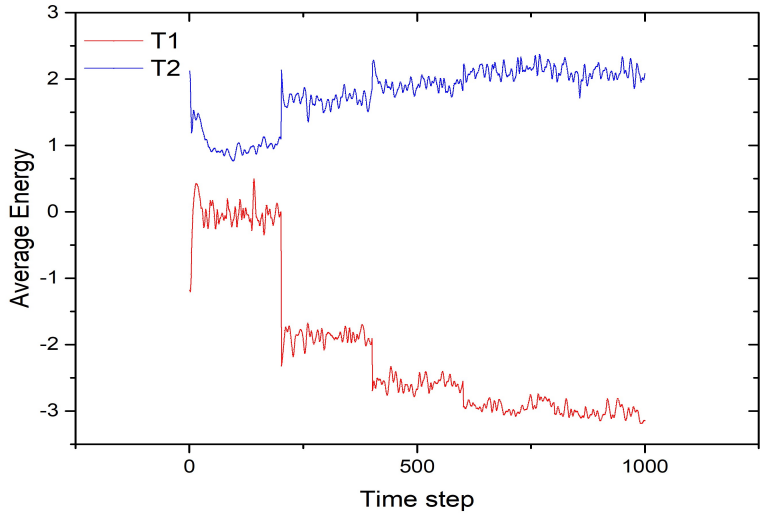
$T=0.793$，$rho=0.807$ 和 $T=1.092$，$rho=0.682$ 条件下温度、平均势能、总能、原子间平均距离随演化的改变趋势分别如图 1，2，3，4 所示，其中 T1 红线和 T2 蓝线分别代表 $T=0.793$ 和 $T=1.092$ 的模拟结果。



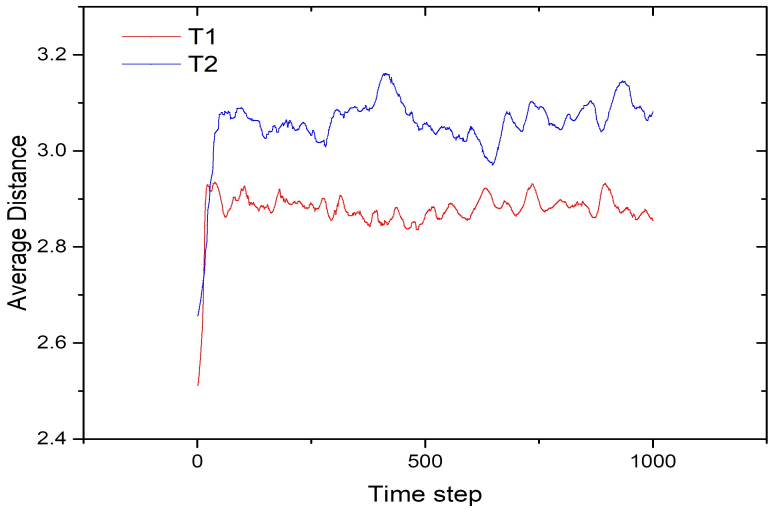
图一：T1 和 T2 温度随演化的改变趋势



图二：T1 和 T2 平均势能随演化的改变趋势



图三：T1 和 T2 能量随演化的改变趋势



图四：T1 和 T2 平均距离随演化的改变趋势

1.3.1 原子平均距离分析

根据立方体棱长的初始化条件，于是原子在不同温度下的最近邻距离 $x_{0.793}$和 $x_{1.092}$ 分别为：

$$x_{0.793}=\frac{\sqrt[3]{\frac{N}{rho}}}{3\sqrt{2}}\approx 1.013;\quad x_{1.092}=\frac{\sqrt[3]{\frac{N}{rho}}}{3\sqrt{2}}\approx 1.071$$

由此我们不难理解图四中 T1,T2 粒子平均距离的变化趋势，对于$\sigma=1$ 的 L-J 势，每两个粒子间的平衡距离为 $\sqrt[6]{2}\approx 1.122$ ，大于 $x_{0.793}$，$x_{1.092}$ 的值，粒子受到斥力，因而倾向于远离对方，平均距离增大。

1.3.2 初始时刻体系能量变化分析

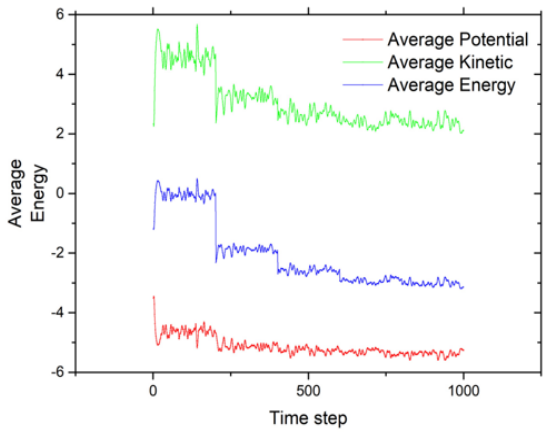
观察图1中 $T=0.793$ 和 $T=1.092$ 下的温度变化曲线，发现在 $T=0.793$ 时温度在模拟初期会有明显的上升而T=1.092时却出现明显的下降。因为系统温度是由原子平均动能计算得到，有如下表达式：

$$T=\frac{\sum_{i=1}^{N}m_iv_i^2}{3(N-1)}=\frac{\sum_{i=1}^{N}v_i^2}{3(N-1)}$$

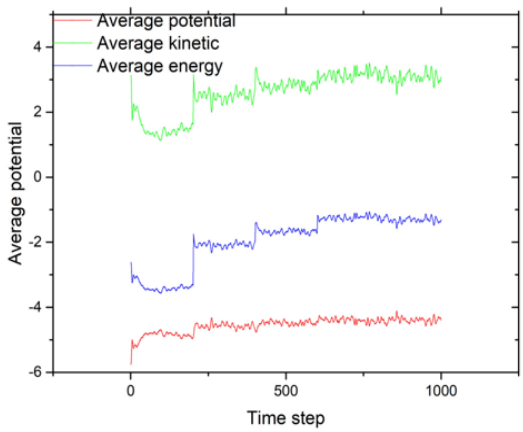
动能可在原子间相互作用力下改变，此时势能和动能发生相互转化，这可以部分解释动能在两种温度下变化趋势不同的原因：比较图 3 中两个温度下的势能曲线，在 $T=0.793$ 下势能迅速降低而在 $T=1.092$ 下势能迅速升高。可以看出在 $T=0.793$ 时的初始时刻，势能转化为动能；而在 $T=1.092$ 时的初始时刻，动能转化为势能。

1.3.3 平衡时刻体系能量分析

同时我们注意到，在这么一个体系中，总的能量并不守恒，在每一步之间都存在着稍许的波动，而每隔一定步数，又会产生一个大的跳跃，经过分析，我们认为，前者可能是因为存在着不同模拟单元之间能量的交换 (边界上的能流密度不为0)，而后者则是因为在模拟过程中，我们选择了引入名为 rescaleVelocities 的速度重置函数，粒子的速度大小被该函数改变，动能突变，进而系统的能量也发生突变。



T=0.793 时能量随时间的改变



T=1.092 时能量随时间的改变

图五：T1 和 T2 平均距离随演化的改变趋势

系统最终的总能量和初始的总能量不尽相同，然而由于本实验模拟的是等温系统，动能的稳定值和初始值相同，那么发生变化的只能是势能项，初步猜测是因为平衡时刻体系结构和初始结构能量不同所导致的。图五显示，T=0.793 时系统最终结构较初始的 fcc 稳定， T=1.092 时系统最终结构较初始的 fcc 更不稳定。

1.4 md1与md2对比

在阅读 lec1 并梳理 md1 代码的设计思路时，我们分析后认为参数 $v_{Max}=0.1, L=10$ 不甚合理。根据 $L=10$ 的参数设定，64 个Ar 原子位于 $L=10$ 的立方体中时，系统的密度 $\rho \approxeq 1.7 kg \cdot m^{-3}$，更接近于气态。取室温条件作为参照，此时$\rho \approxeq 1.7837 kg \cdot m^{-3}$，可以估计此时系统的温度为 300 K。沿用原子速度在 $\pm v_{max}$ 均匀分布的假设，并根据能量均分定理应有：

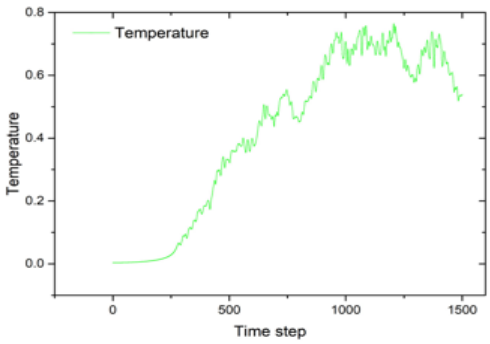
$$\frac{m}{2} N \int_{-v_{max}}^{v_{max}} \frac{v^2}{2v_{max}} dx = \frac{3}{2}(N-1)k_B T$$

$$v_{max} \approx 10^3 m \cdot s^{-1}$$

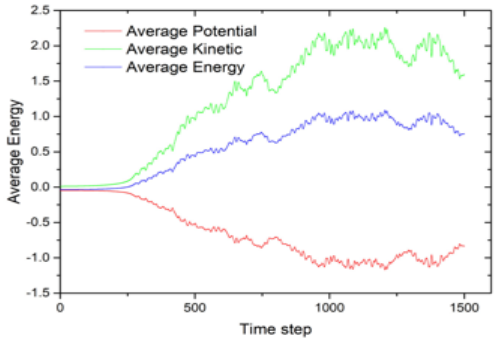
对于 $v_{Max}=0.1$ 而言，代入此时的长度和时间单位后，对应的速度约为 $15.67 m \cdot s^{-1}$，可见此参数的量级设置并不合理。

反过来，根据 $v_{Max}=0.1$，可计算得系统初始温度为 $10^{-3} K$，此条件下系统为固态，故而此时系统的密度出现了数量级偏差。再加上 lec1 中所指出的 md1 程序中原子数不守恒、速度分布不精确、初始位置设定有偏差等问题，可见 md1 参数设置和模型建立过于简单。

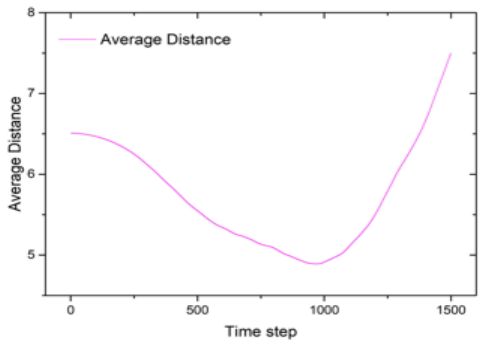
据此，我们认为很难在保持原本代码的框架下做出精确的模拟，如果设置更精确的初始条件，则与 md2 的工作有所重复。除了添加计算平均势能、平均动能、平均能量、系统温度和平均距离的代码，我们没有对md1做更多的改动。模拟计算的结果如下：



md1 系统温度随时间的改变



md1 体系能量随时间的改变



md1 平均距离随时间的改变

图六：N=64 时 md1 的结果

由图，我们发现系统始末的温度差量级为 $10^2 K$，即使在 1500 步后温度曲线仍然没有出现平缓的趋势；1500 步模拟后原子的平均动能、平均势能和平均能量仍然没有达到稳定，而 md2 中 T=1.092 和 T=0.793 条件下能量基本在模拟 600 步后达到稳定；原子之间平均距离在 1000 步后出现明显的上升，说明原子在运动过程中发生了逸散，这也证明了 md2 代码中引入周期性边界条件和镜像法则的合理性。综上，md1无法得到较为准确和合理的模拟结果。

2. 借助 Python 开源拓展库对 md2 体系进行可视化处理

2.1 概要

Python 是一种开源的解释型脚本语言。众多开源的科学计算软件包都提供了 Python 的调用接口，例如著名的计算机视觉库 OpenCV、三维可视化库 VTK，python 社区也拥有一些强大的开源计算库如 numpy 、 matplotlib 等。本次的可视化引用了下面的拓展库：

```
import random
import numpy as np
import math
import matplotlib.pyplot as plt
import cv2
```

2.2 代码实现

2.2.1 初始化 & 弛豫

首先引入粒子坐标和速度初始化的函数，其中 initPositions() 和 initVelocities(T) 的功能实现和 lec1 中代码相似，此处省略。此后的 main 函数中，我们遍历一个不同 T 的列表，并把粒子速度置为该温度下的高斯分布，以便在同一个程序中实现不同温度下的 md2 体系模拟。代码如下：

```
def initialize(T):
    r = initPositions()
    v = initVelocities(T)
    return r,v
...
# main 函数中
for T in [0.005,0.01,0.05,0.1,0.5,1,2,4]:
    step = 1
    r,v = initialize(T)
```

接着引入名叫 rescaleVelocities 的函数，同 lec1 中 C 语言的实现方式。此函数可以克服初始条件中的一些随机因素，每隔 500 模拟步数，便将粒子速度分布重置为 T 时的高斯分布，这样有利于系统弛豫时间的缩短：

```
def rescaleVelocities(T):  # adjust the instanteous temperature to T
...
# main 函数中
if step % 200 == 0:
    rescaleVelocities(T)
```

2.2.2 粒子加速度、平均势能和平均距离的计算

首先，在程序的函数申明部分，我们引入一些量的默认值和某些能够储存程序产生的计算结果的数据结构(如列表和 numpy数组)：

```
name = 'md2_0504'; steps = 2000; dt = 0.01  # 模拟名称，总模拟步数，模拟时间间隔
N = 1000; rho = 1  # 粒子数目和粒子数密度
r = np.zeros([N,3]); v = np.zeros([N,3]); a = np.zeros([N,3])  # 位置，速度，加速度(三维)
P_sums = []; K_sums = []; E_sums = []; d_avers = []  # 总势能，总动能，总能量，平均距离
Ps = np.zeros([N]); Ks = np.zeros([N])  # 单粒子势能，单粒子动能
```

其次定义加速度计算函数，此处同 lec1 中 C 语言的实现方式：

```
def computeAccelerations():
…
```

然后，为了节省算力，我们将单粒子势能、体系总势能和粒子平均距离的计算过程放在同一个函数中：

```
def instantaneousPD():
    global r,Ps
    d_aver = 0.0; d_sum = 0.0; count = 0; p_sum = 0
    for i in range(N):  # loop over all distinct pairs i,j
        for j in range(N):
... # 此处进行 i、j原子之间相对距离的计算，并用 lec1 中的镜像法对原子坐标进行变换
            rSqd += rij[k] * rij[k]
            if rSqd != 0:
                Ps[i] += 4*(math.pow(rSqd,-6) - math.pow(rSqd,-3)) # Add j's effect on i
            d_sum += pow(rSqd,0.5)
    d_aver = d_sum/count
    for i in range(N):
        p_sum += Ps[i]
    return p_sum/2/N, d_aver  # Divided by 2N here since each particle is counted twice in our calculation
```

其中 p_sum(某一模拟步时粒子的总势能) 和 d_aver (某一模拟步时粒子的平均距离) 以参数传递的形式传递给 main 函数，而单粒子势能则通过我们一开始创建的全局变量 Ps 传递(一个 python 列表)

2.2.3 粒子平均动能与平均能量的计算

与 lec1 温度计算代码相似，我们首先计算出粒子速率的平方，然后代入$E_k = \frac{1}{2}mv^2$ 中计算。

```
def instantaneousKinetic():
    global v,Ks
    k_sum = 0
    for i in range(N):
        for k in range(3):
            Ks[i] += 0.5*v[i][k]*v[i][k]
            k_sum += Ks[i]
    return k_sum/N
```

在主程序中，我们每个模拟步骤计算一次粒子动能和势能，进而计算总能，最后将它们附加到相应的 list 的末尾：

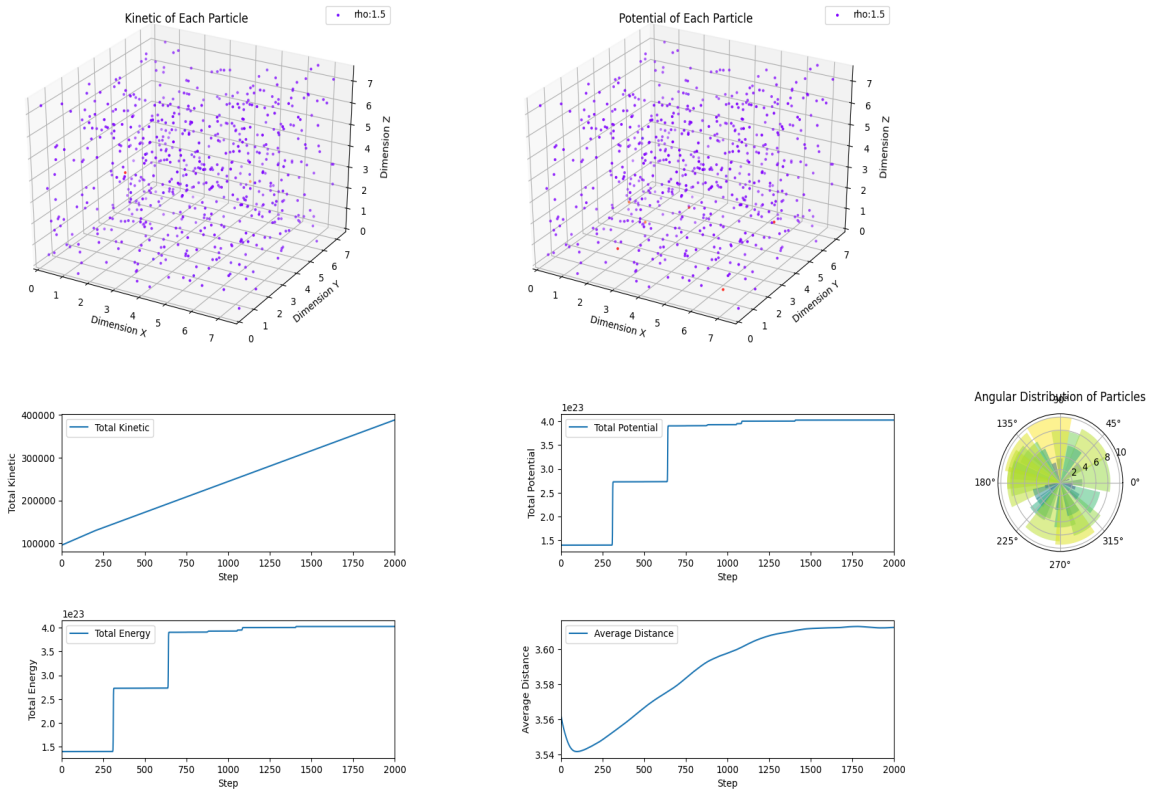
```
K_sum = instantaneousKinetic(); K_sums.append(K_sum)
P_sum, d_aver = instantaneousPD(); P_sums.append(P_sum)
E_sum = K_sum + P_sum; E_sums.append(E_sum)
```

2.2.4 主程序结构

```
for T in [0.005,0.01,0.05,0.1,0.5,1,2,4]: # 遍历各种温度
    ...# 初始化
    while step<=steps: # 在规定的步数内运行
        ...# 各种计算
        velocityVerlet 算法求解运动方程(dt) # Verlet 算法求解运动方程
        if step % 2 == 0:
            p.plot(r,v,N,rho,Ks,...) # 调用本目录下的画图 python 文件
        if step % 200 == 0:
            rescaleVelocities(T) # 重置速度
        step += 1
```

2.3 结果

使用 matplotlib 库，可以作出粒子热力学数据随时间变化的折线图、粒子在空间中分布的三维散点图，效果如图所示，图中散点的不同颜色表示了该粒子相应能量的高低，粒子能量由紫色向黄色再向红色依次增大，饼状图则形象地表示了在 x-y 平面上往不同方向运动的粒子数目。



图六：N=1000,rgo=1,T=0.05 时的样例

随后使用 opencv 库，便可以将不同模拟步数时的这些图像串联成视频，更加清楚地观察到体系性质随时间的变化情况。不同温度时的可视化结果如下：

T0.005； T0.01； T0.05； T0.1； T0.5； T1； T2；

3. 题目二：计算Ar气凝结为固体时的单空位形成能和双空位结合能

3.1 单空位形成能

对于氩，可由 LAMMPS 建立 30×30×30 的 BCC 晶格体系，具体程序见 in.E1.Ar ，运行结果见 E1-log.lammps 文件，用 OVITO 可视化处理后得到模拟过程的 MP4 视频与 图片。

公式计算空位形成能：

$$E_v = E_{defect} - (N - \frac{1}{N})E_{prefect}$$

其中 N 为总粒子数；

$E_{prefect}$ 为优化后完整晶胞的总能量；

E_{defect} 为优化后含有单空位晶胞的总能量。

则：

$$BCC结构的Ar中单空位形成能为 -4.26eV$$

3.2 双空位结合能

对于氩，LAMMPS 建立 30×30×30 的 BCC 晶格体系，具体程序见 in.E2.Ar，运行结果见 E2-log.lammps 文件，用OVITO可视化处理后的MP4视频与图片。

公式计算双空位形成能：

$$E_v = E_{defect2} - (N - \frac{2}{N})E_{prefect}$$

其中N为总粒子数；

$E_{prefect}$ 为优化后完整晶胞的总能量；

$E_{defect2}$ 为优化后含有双空位晶胞的总能量。

最后可以得到 BCC 结构的 Ar 中双空位形成能为 -2.93eV.

则：

$$Ar中双空位结合能 = 双空位形成能 - 单空位形成能 \times 2 = -2.93 + 4.23 \times 2 = 5.53eV$$

4. 补充：使用 lammps 并行版本对 md2 进行重制

4.1 概要

在本小组借助 python 对md2 中的 Ar 气体进行分子动力学的模拟的过程中，我们发现，随着粒子数 N 的增大，程序的运行效率明显下降，运行时间大大增强以至于难以完成计算，而且将 python 文件上传至学校的超算中心后，计算效率没有明显提升。

经过分析，我们认为，这是因为在构建程序的过程中采用了太多的循环嵌套，比方说粒子坐标的初始化，力和加速度的计算，势能的计算等，另一个原因是本次的 python 程序设计中，将计算模拟的过程和绘制数据图片的过程集成在了一起，每算几步便要调用绘图程序进行图片的绘制和储存，部分加重了处理器的负担。

根据以上分析，我们决定在更加复杂的情景时采用 lammps 方法对原有的 md2 程序进行重制。主要是因为 lammps 以C语言为基础，并且并行拓展性良好，效率高，同时它输出的数据为文本文档型，计算模拟和可视化过程分离，只要后期将产生的 md.xyz 文件导入 ovito 中，便能得到我们需要的模拟可视化视频、动能和势能曲线等结果。

4.2 in文件设置

in 文件是 lammps 的输入文件，借助 in 文件中的诸多命令，我们可以将 md2 中的初始模型、模拟环境和作用力场用 lammps 轻松重构出来，并定义原子/体系某些信息的计算，最终输出原子/体系的热力学信息。

4.2.1 units & atom_style 设置

units 命令可以用来定义整个模拟系统中的单位。本模拟出于避免数据处理时繁杂的单位换算的考虑，沿用了 md2 和 md1 中无量纲的设定，并将σ, ε 和玻尔兹曼常量等置为1。而对于中性稀有气体原子而言，我们可以采用 atomic类型的原子类型，代码如下：

```
units               lj
atom_style          atomic
```

4.2.2 boundary设置

boundary 命令是用来定义边界条件的，LAMMPS 提供了四种边界条件分别为：

- p 边界：周期性边界条件；
- f 边界：非周期性边界条件；
- s 边界：边界的距离会随着原子的运动而改变，以保证不丢失原子；
- m 边界：边界的距离会随着原子的运动而改变，但该方向长度有最小值；

在 md2 文件中，我们为了防止粒子的丢失而采用了周期性条件，采用镜像法的方法去计算每个粒子的势能和加速度，所以我们可以通过如下命令将 x，y，z 三个方向上的边界都设定为周期性的。

```
boundary            p p p
```

4.2.3 lattice & region 设置

在 md2 的模拟中，我们人为引入了粒子数密度 ρ 的概念，通过 ρ 和粒子总数 N 来计算得总模拟空间的大小L，再根据粒子数目N计算出晶格常数 a ，最后通过 3 个for循环遍历 x，y，z 坐标，根据算出的 a 来将原子一个一个地放置在空间中，以形成 fcc 的结构。

在 LAMMPS 中，粒子位置的初始化和模拟空间的创立是分开的。使用 lattice 命令可以定义晶格类型，晶格常数，以及晶向方向。

根据A.Rahman 使用的 $94.4K$ 下的 $1.374g \cdot cm^{-3}$ 和$130K$下的 $1.16g \cdot cm^{-3}$的数据（当然计算模拟时我们对真实液态的 Ar 做了一点假定，即它们在模拟的开始以 fcc 的形式存在），此时 Ar 原子 fcc晶格常数 $a_{0.793}$和 $a_{1.092}$ 分别为：

$$a_{0.793} = \sqrt[3]{\frac{4M}{N_A \rho}} \approx 5.7824\text{Å} = 1.701$$

$$a_{0.793} = \sqrt[3]{\frac{4M}{N_A \rho}} \approx 6.1181\text{Å} = 1.800$$

我们以这个结果作为我们模拟的初始条件。

```
variable a0 equal 1.701
variable x equal 4*${a0}
variable y equal 4*${a0}
variable z equal 4*${a0}
lattice             fcc 1.701
region              box block 0 x 0 y 0 z
create_box          1 box
create_atoms        1 box
```

上面的代码规定了晶格类型为 fcc ， 晶格常数为 1.701 ， 并创立一个 $4 \times 4 \times 4$ 的方形空间，刚好能容纳下 64 个 fcc 的晶格，也就是 256 个 Ar 原子。接着通过 create_box 命令告诉LAMMPS 名为 ‘box’ 的模型中总共有1类原子，最后 create_atoms ， 按照 lattice 命令的设定添加原子以填满盒子。

4.2.4pair_style & pair_coeff 设置

LAMMPS 模拟中，我们通过 pair_style & pair_coeff 告诉 LAMMPS 原子间相互作用势的类型，并给出势函数中的参数或者数值列表。

```
mass            * 1.0
pair_style      lj/cut 2.5
pair_coeff      * * 1 1
```

在上面的代码中，我们用 mass 命令将所有类型的原子（只有一种）的质量设为 1 ， 通过 pair_style 命令我们将原子作用势设定为 Lennard-Jones 势，全局截断半径(距离大于则原子间作用力可忽略)为 2.5 ，通过 pair_coeff 命令我们将 ϵ, σ 分别设为 1 ， ‘* *’ 表示所有原子类型。

4.2.5compute 设置

lammps 中 compute 命令可在每一个时间步，对系统及系统中每个原子蕴含的物理量定义一个计算过程，并在产生输出文件时调用。

```
compute             1 all temp
compute             2 all pressure thermo_temp
compute             3 all pe/atom
compute             4 all ke/atom
compute             tpe all reduce sum c_3
compute             tke all reduce sum c_4
compute             5 all msd com yes average yes
compute             6 all pair/local dist
compute             7 all reduce ave c_6
```

在上面的代码中，我们定义了九种热力学量的计算，分别是温度，压力，每个粒子的动能和势能，总动能和总势能，

4.2.6 thermo & thermo_style & dump 设置

lammps 中 thermo 命令可用来设置在模拟中计算和打印热力学信息（比如温度、能量、压强）的频率；thermo_style 命令则可以输出所需要的热力学信息；最终输出的计算结果可通过dump命令写入文件。

```
thermo 10
thermo_style custom step temp press ke pe atoms etotal dt time c_7
dump 1 all custom 10 md2.xyz id type x y z vx vy vz c_3 c_4
```

上面的代码中，由 compute 定义的热力学信息每 10 步便输出一次；thermo_style 分别自定义输出了体系的运行的步数、温度、压强、动能、势能和总能量；dump 命令中all表示对于所有原子，500表示每运行500步输出一次；md2.xyz 表示输出文件的文件名，id 为原子的编号，x , y 和 z 为原子的坐标，vx ,vy 和 vz 为原子的速度，dt 为模拟步长，time 为距离初始时刻所经历的时间，c_3, c_4, c_7 分别为编号为3、4、7的计算中所计算的热力学量。

4.2.7 velocity & timestep & fix & run设置

velocity 命令可以赋予初始时刻原子的速度；timestep 命令设定了模拟步长；fix 命令可以设定模拟系综，如 nve , nvt , npt 等，还可以帮助固定原子和输出所需信息；整个程序的最后，我们可以通过 run 命令来设置模拟步数。

```
velocity all create 0.793 4928459 dist gaussian
timestep 0.001
fix 1 all nvt temp 0.793 0.793 $(100.0*dt)
run 10000
```

上面的代码中，我们赋予了所有原子在 0.793 温度单位时的一个高斯速度分布，其中 4928459 是生成速度随机数的种子；模拟步长是 0.001 个时间单位；fix 命令表示控制体系在温度为 0.793 温度单位的 nvt (正则系综) 环境中模拟，温度的弛豫时长大约为 100 个模拟步长。

由我们之前的分析，本次模拟中能量单位 $\epsilon = 1.65 \times 10^{-21}\mathrm{J}$，距离单位 $\sigma = 3.4\ \text{Å}$ ，此时温度单位 $T^* = \frac{Tk_B}{\varepsilon} = 119K$；最后用 run 命令将模拟执行 10000 步。

最后的 in 文件：

```
# Initialization
units               lj
atom_style          atomic
boundary            p p p

variable a0 equal 1.701
variable x equal 4*${a0}
variable y equal 4*${a0}
variable z equal 4*${a0}
# Reading(Creating) Model
lattice             fcc ${a0}
region              box block 0.0 ${x} 0.0 ${y} 0.0 ${z}
create_box          1 box
create_atoms        1 box

# Interacting
```

```
mass * 1.0
pair_style  lj/cut 2.5
pair_coeff  * * 1.0 1.0 2.5

# Define the outputs
compute             1 all temp
compute             2 all pressure thermo_temp
compute             3 all pe/atom
compute             4 all ke/atom
compute             tpe all reduce sum c_3
compute             tke all reduce sum c_4
compute             5 all msd com yes average yes
compute             6 all pair/local dist
compute             7 all reduce ave c_6

# Information
thermo 10
thermo_style custom step temp press ke pe atoms etotal dt time c_7
dump 1 all custom 10 md2_T0.793.xyz id type x y z vx vy vz c_3 c_4

# steer the simulation
velocity all create 0.793 4928459 dist gaussian
timestep 0.001
fix 1 all nvt temp 0.793 0.793 $(100.0*dt)
run 10000
```

4.3 结果分析

借助于 ovito 软件，我们得到了如下的结果：

1. $T = 0.793, rho = 0.807$： 模拟视频;
2. $T = 1.092, \ rho = 0.682$： 模拟视频;

从视频中可以清楚地看出，$T = 0.793$ 是的体系更加偏向于固体，原子都在固定的位置上振动，$T = 1.092$ 时的体系则更加具有液体的性质，原子运动剧烈变化着。这两种截然不同的运动方式可能暗含着这两个温度之间存在这某种意义的相变现象。