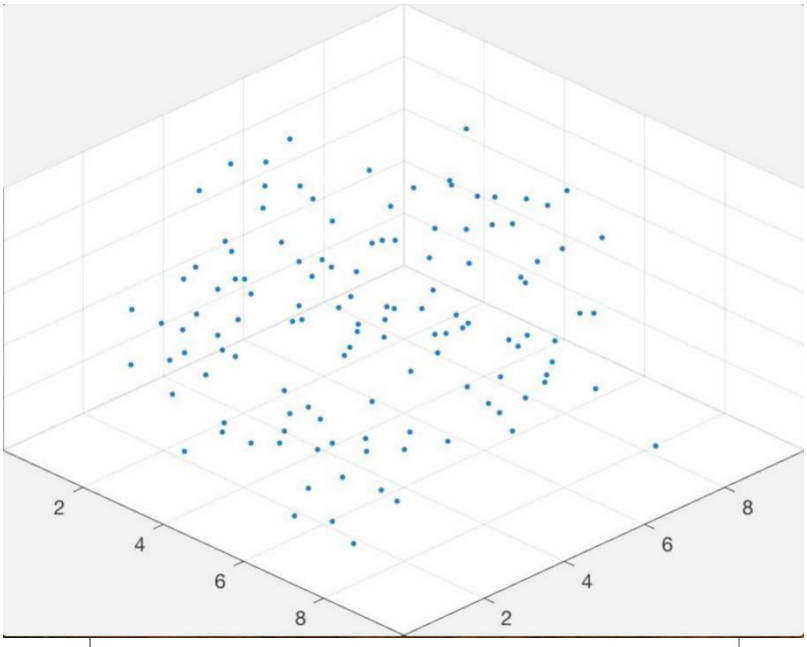
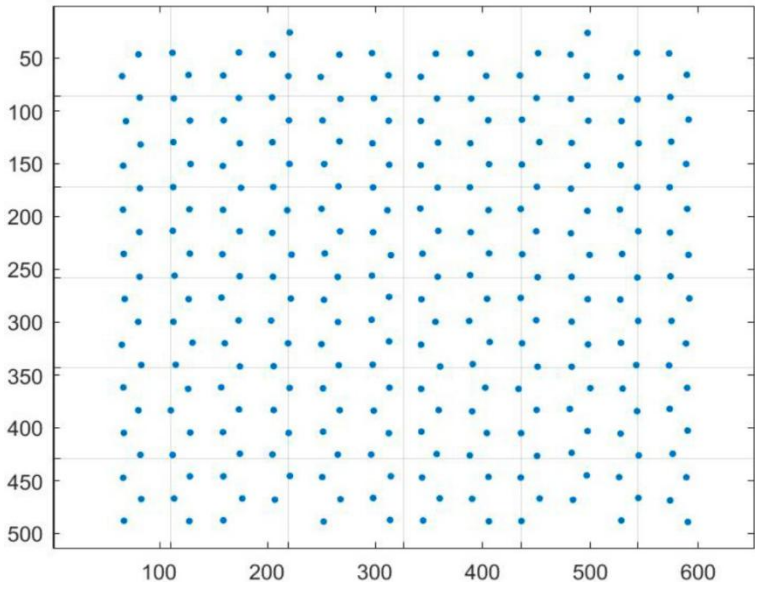
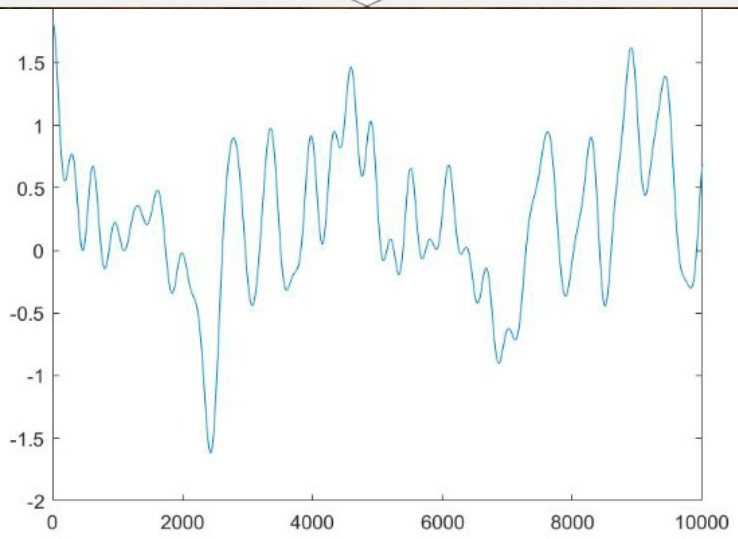
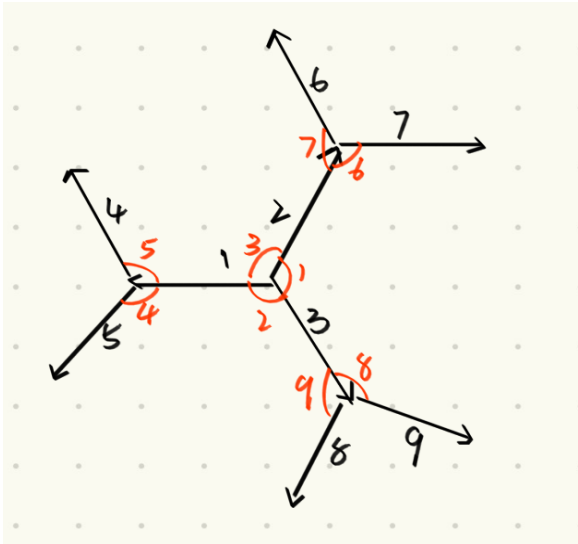






(b) 1Pa



From “Zero” to our MD

—Programming and application of a MD project

组员:刘诗韵 池志豪 唐泽宸
指导老师:吴健

摘要

分子动力学(MD)是计算物理常用的模拟方法。基于动力学方程、边界条件与势函数,分子动力学模拟离散时间下的分子运动,从而得到体系性质的有关结论。在本大作业的第一部分中，我们依据计算物理课程讲义，基于 MATLAB 程序建立了我们的分子动力学模拟程序。势函数方面，我们按照大作业要求内置了 Lennard-Jones 势能，并给出了一般性的二体势编程方法；算法方面，我们在程序中实现了 Verlet 算法、Predictor- Corrector4 算法以及 Predictor-Corrector6 算法。我们的模拟程序模拟 103 个原子体系时,效率约为一秒钟 10 个时间步。

作为应用的第一部分,我们主要探究了石墨烯体系的性质。我们在模拟中,生成了以石墨烯实验参数为基准的石墨烯模型；以两体相互作用为基准,给出了以平衡位置和势阱深度为参数的 LJ 势能模型；给出了不考虑和考虑次近邻相互作用的 LJ 势能，并进行比较；给出了在 LJ 势能模拟下，对于边界给定特定应力的动态模拟，进行了该模型杨氏模量和理论值的对比；针对石墨烯体系，结合实际计算中常用的 Rebo 势，对程序的稳定性进行改进；但结果没有成功，文中给出了具体的构架方法和失败的可能原因的分析。

总目录

Part I Our MD Program

目录

1 Outline

本程序旨在实现基本的分子动力学模拟功能。按照讲义“建立和运行分子动力学模拟”部分的流程，我们需要明确如下内容：

1、动力学方程与对应的有限差分算法

2、模拟采用的势能

3、边界条件

4、初始化与输出

其中，第一点将在本篇第二节中讨论，第二、三点将在第三节中讨论，第四点将在 第四节中讨论。最后，我们对程序效率进行测评，并给出程序主体中的变量表，方便后续二次开发。

2 Algorithm for MD Simulation

本部分中，我们实现了多种 MD 计算的算法，包括课上上所述的 Ver1et 方法与 Predictor-Corrector 方法。讲义上对 Predictor-Corrector 方法仅是略有涉及，特别是对于“最为常用”的 Predictor-Corrector 6 算法，并未给出具体实现形式。我们查阅了 Predictor-Corrector 方法主要提出者之一的 Gear 教授于 1971 年出版的专著 Numeri- ca1 Va1ue Prob1emt in Ordinary Diferentia1 Equation，进行了算法的实现。

2.1 Single-step Method: Verlet

Ver1et 算法具有非常简单的形式：

$$\mathrm{r}(\mathrm{t}+\delta\mathrm{t}) = 2r(t) - r(t-\delta t) + a(t)(\delta t)^2 \tag{1}$$

从编程的角度来说，实现 Ver1et 算法的主要注意点包括：

1、存储的物理量包括当前时刻位置、加速度与上一时刻位置

2、算法的每一次时间步更新仅需计算当前时刻的受力

因此，Ver1et 算法运行过程中，对 N 个原子的体系，共计需要存储 3 个 N*3 的数组，即当前时刻位置，上一时刻位置，当前时刻加速度。算法中每个时间步流程如下：

1、计算当前坐标下的加速度

2、根据当前坐标与上一时刻的坐标，更新下一时刻坐标

3、重新对“上一时刻”、“这一时刻”坐标进行赋值

其中主要的计算资源消耗在于计算当前坐标下的加速度。

2.2 Multi-step Methods: PC4 and PC6

ppt 上提到的 Predictor-Corrector 算法是一种求解微分系统的多步(multi-ttep)方法。Predictor-Corrector 算法由 Gear 教授于 1969 年提出，并在 1971 年出版的专著中进行了系统的总结。

Predictor-Corrector 算法的基本思想是：欲求解微分系统

$$\mathrm{y}^{(\mathrm{k})} = f\left(x, y, y^{(1)}, y^{(2)}, \ldots y^{(k-1)}\right) \tag{2}$$

已知 y_n-1(以及更之前的步骤)的信息时，引入预测子(Predictor)与修正子(Cor- rector)：

$$\mathrm{y}_{\mathrm{n}(0)} = B y_{n-1} \tag{3}$$

$$\mathrm{y}_{\mathrm{n},(\mathrm{m}+1)} = y_{n,(m)} + cG\left(y_{n,(m)}\right) \tag{4}$$

从来达到更高级别代数精度的目的。其中 y_n 为一个向量，包括 y 的函数、导数、高阶导数等信息。

二阶微分系统的 k 级 Predictor-Corrector 方法的正则形式是指：存储的

$$\mathbf{y_n} = \left(y, hy^{(1)}, \frac{h^2}{2}y^{(2)}, \ldots, \frac{h^{k-1}}{(k-1)!}y^{(k-1)}\right)^T \tag{5}$$

由于引入多步修正并不改善方法的阶数，因此仅采用一个修正子，即每步迭代包括：

$$\mathbf{y_{n,(0)}} = By_{n-1}G\left(y_{n,(1)}\right) = \frac{h^2}{2}f\left[y_{n,(0)}\right] - y_{n,(0),3}y_n = y_{n,(0)} + cG\left(y_{n,(1)}\right) \tag{6}$$

其中，c 为一个 k 维的向量。Gear 在其专著中给出了二阶正则形式的向量分量。当 k=4 时

$$\mathbf{c} = \left(\frac{1}{6}, \frac{5}{6}, 1, \frac{1}{3}\right)^T$$

当 k=6 时

$$\mathbf{c} = \left(\frac{3}{20}, \frac{251}{360}, 1, \frac{11}{18}, \frac{1}{6}, \frac{1}{60}\right)^T$$

B矩阵根据位置的Taylor展开即可获得。特别地，当 k=4时，有

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 3 & 6 & 10 \\ 0 & 0 & 0 & 1 & 4 & 10 \\ 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

从编程的角度来说，实现 Predictor-Corrector k 算法的主要注意点包括：

1、存储的物理量包括坐标的 0 至k-1 阶导数

2、算法的每一次时间步更新需计算两次受力

因此，对N个原子的体系，Predictor-Corrector k 算法共计需要存储k个 N*3 的数组。算法中每个时间步流程如下：

1、对下一时刻坐标进行预测

2、根据预测的坐标修正加速度

3、根据修正的加速度修正坐标各阶导数实际数值

4、更新相空间状态

3 Potentials and Boundary Conditions

3.1 Potential

按照大作业要求，我们采用 Lennard-Jones 势能作为默认势能，Lennard-Jones势能具有形式：

$$4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12}-\left(\frac{\sigma}{r0}\right)^{6}\right) \tag{7}$$

受力大小为：

$$24\epsilon\left(-\frac{2}{r}\left(\frac{\sigma}{r}\right)^{12}-\frac{1}{r}\left(\frac{\sigma}{r}\right)^{6}\right) \tag{8}$$

对于截断的处理策略为：倘若两个原子之间的间距小于截断半径才进行力的计算。

本程序没有内置的参数 ε，σ，但留下了代表“原子种类”的数据接口，可以对原子种类进行识别后从外部来源读取参数。“原子种类”同样可以用于读取原子的质量参数。

鉴于 Matlab 对于向量运算较快的特性，我们采用策略：首先计算并存储原子之间的位矢，而后计算力的大小。在Lennard-Jones势能提供作用力为有心力的前提下，仅需将力的大小投影到位矢上即可。其中力的大小具有正负：若为负则提供排斥力，反之 则提供吸引力。此外根据牛顿第三定律，作用力与反作用力大小相同、方向相反，我们基于此减小了一半的计算量。

由于我们的近邻搜索是一个两体过程，且存储了两体之间的位矢与距离，因此原则上各种形式的两体势均可以由本程序框架计算。对于三体或更多体势能，则需要修改程序骨架。

3.2 Boundary Conditions

本程序骨架部分实现了三个方向上的周期性边界条件，具体表现为在计算原子距离 是否小于截断半径、以及计算位矢时基于周期性边界条件进行了处理。当一个时间步后倘若原子坐标超出范围，则将被周期性边界条件限制回到同一晶胞内。三个方向上的周期长度均为可调参数。

基于周期边界条件也可以简单地实现开边界条件：只需要令开边界条件方向上的长度趋向于无穷大，这个方向便不体现出任何周期性。

对于其他边界条件，本程序主体中没有涉及，需要重新编程。

4 Outputs and Initialization

4.1 Initialization

初始化过程涉及赋予原子初始速度、坐标、原子种类、周期性边界的周期等等。由于初始化

种类非常多样，因此本程序主体中并未给出具体的初始化方法。作为替代，我们尽可能使得使用时只需要输入真实存在的物理量，其他需要存储的变量则由程序完成初始化，具体包括：

1、Verlet 算法中，根据初始位置与速度，为初始的”上一步位置“赋值

2、Predictor-Corrector 算法中，除非特别声明，否则初始的坐标的各个高阶导数均为0

4.2 Observables

由于需要的物理量种类十分多样，程序主体中并无默认输出的物理量。

对于非含时性质（如平均动能）等，可以在每一时间步原位计算后存储，这种输出 对存储空间并无太多影响。

对于含时性质，则需要存储不同时间步的物理量，此时需要分配额外的存储空间。

4.3 Visualization

我们利用 Matlab 自带的 getframe 函数，对于粒子的运动过程录制成了影片，以检验程序的合理性。附件为 125 个原子运动的视频。由于初始原子距离比平衡距离远，存储的势能随时间推移逐步转化为动能，最终一些粒子动能过大，模拟停止。

实际计算中除非对运动过程有图像化需要，否则建议关闭可视化模块，以免影响效率。

5 Efficiency

对于二体势，原则上计算原子受力的总时间关于原子数 N 为 O(N^2)，但由于截断半径的存在，实际只有 O(N) 个力需要进行计算。计算的其他主要步骤耗时亦为 O(N)。

采用Verlet算法并关闭图形化输出时，对100个原子的体系，每秒大约可以进行200个时间步的模拟；对 1000个原子的体系，每秒大约可以进行10个时间步的模拟。我们的模拟程序效率相比一些文献提到的模拟程序效率尚有差距，无法进行特别大规模体系的模拟。

Predictor-Corrector算法无论阶数，效率均约为Verlet算法的一半，这是因为一个时间步骤中，Predictor-Corrector算法需要计算2次受力情况。

6 变量表

变量名	含义	变量名	含义
absforce	原子间作用力绝对值， 正为吸引，负为排斥	mass	原子质量
accel	原子加速度 （Ver1et 算法中间变量）	mov	视觉化输出的视频
ALGOType	算法种类，1为Ver1et， 2为 PC4，3为 PC6	N	原子数目
coord	原子位置 （Ver1et 算法变量）	parttype	原子种类
coordnext	原子下一时刻位置 （Ver1et算法中间变量）	phasespace	原子相空间坐标 （PC4、PC6变量）
dist	原子间距	POTType	势能类型
distvec	原子间位矢	rcut	截断半径
epsi1on	Lennard-Jones 势能参数	sigma	Lennard-Jonet势参数
force	受力向量	tmax	最大时间步数
1ength	各个方向上PBC的周期长度	timettep	时间步长

Part Ⅱ Graphene

目录

1 LJ Potential

众所周知，LJ 势能可以简写成如下形式：

$$V(r,A,B)=\frac{A}{r^{12}}-\frac{b}{r^{6}} \tag{1}$$

那么，在只考虑两体相互作用时，我们可以很容易给出 LJ 势能势阱深度和平衡位置关于参数AB的关系：

$$r0=\left(\frac{2A}{B}\right)^{\frac{1}{6}} \tag{2}$$

$$V0=-\frac{B^2}{4A} \tag{3}$$

考虑石墨烯结构的次近邻相互作用后，我们推导如下：

$$V'=V(r,A,B)+2V(\sqrt{3}r,A,B) \tag{4}$$

由此我们得到关系如下：

$$r0=\left(\frac{1462A}{783B}\right)^{\frac{1}{6}} \tag{5}$$

$$V0=-\frac{841B^2}{2924A} \tag{6}$$

2 Experiment Contents

2.1 Generate Graphene

对于该实验，我们需要生成平面结构的石墨烯，为方便起见，我生成了单层的石墨烯薄膜，具体代码如下

Listing 1: 生成代码展示

```
imperf = 0.01;
for i=1:N
    if rem(i,2)==0
        coord(2*i-1,1)=imperf*rand() + 0.426/2*rem(floor((i-1)),12)
            -0.071;
        coord(2*i-1,2)=imperf*rand() + 0.142*sqrt(3)*(rem(floor((i-1)
            /12),12));
        coord(2*i-1,3)=imperf*rand();
        coord(2*i,1)=imperf*rand() + 0.426/2*rem(floor((i-1)),12)
            +0.071;
        coord(2*i,2)=imperf*rand() + 0.142*sqrt(3)*(rem(floor((i-1)
            /12),12));
        coord(2*i,3)=imperf*rand();
    else
        coord(2*i-1,1)=imperf*rand() + 0.426/2*(rem(floor((i-1)),12))
            -0.071;
        coord(2*i-1,2)=imperf*rand() + 0.142*sqrt(3)*(rem(floor((i-1)
            /12),12)+0.5);
        coord(2*i-1,3)=imperf*rand();
        coord(2*i,1)=imperf*rand() + 0.426/2*(rem(floor((i-1)),12))
            +0.071;
        coord(2*i,2)=imperf*rand() + 0.142*sqrt(3)*(rem(floor((i-1)
            /12),12)+0.5);
        coord(2*i,3)=imperf*rand();
    end
end
```

赋予石墨烯六角晶格结构，单位是nm；我将2*N改为了原胞数量，因为石墨烯一个原胞内部有两个原子；微扰使用简单的 rand() 来给出，包括平面内和z方向

生成效果如下

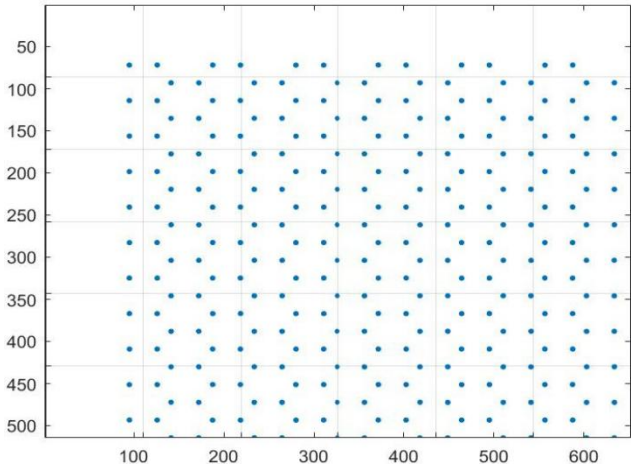


图 一：Graphene Generation

为了更具有泛用性，我们给出可调的行数/列数/层数，代码如下：

Listing 2: 生成参数

```
rows = 11;
colums = 12;
layers = 1;
N = rows*colums*layers ;
```

其中 rows/colums/layers 都是可调的参数。

2.2 Compare next-nearest and nearest

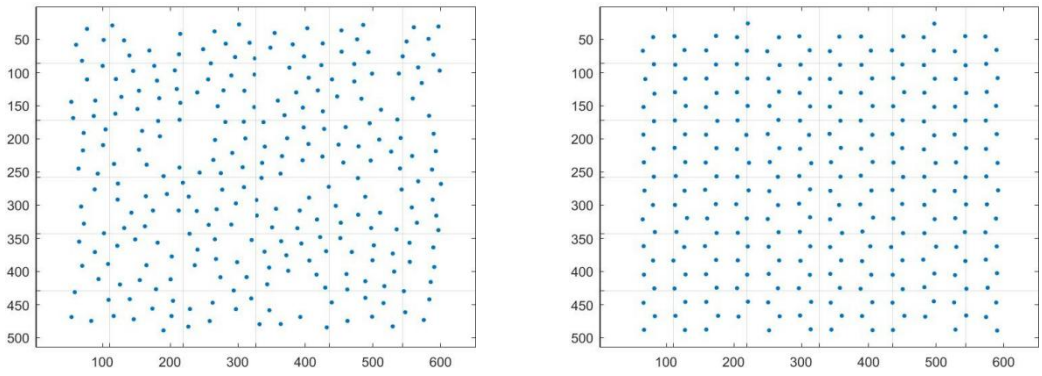
一个最简单的模型，就是我们选取合适的截断半径，让我们的势能几乎只作用于一 个最近邻的原子。

直觉上考虑，我们理所应当认为，次近邻的模型考虑得更周到，理应效果更好，但 是事实上我们得到了反直觉的结果

上图是给出0.01nm微扰后1000个时间步，0.0001s后的图像。

(参考 planenn001.gif 和 planen001.gif)

最开始，我还以为这是周期性边界条件的特性导致，为此制作了固定边界条件进行对比，结论同上，于是我们最终选择了最近邻模型进行模拟。



(a)Bad next-nearest (b) Good nearest

图二： Comparison

我们给出边界条件的方式如下：

Listing 3: 固定边界

```
            xminjud = ismember(i,xmin(1:rows));
            xmaxjud = ismember(i,xmax(1:rows));
            yminjud = ismember(i,ymin(1:colums));
            ymaxjud = ismember(i,ymax(1:colums));
            if xminjud||xmaxjud||yminjud||ymaxjud
                accel(i,:)=[0,0,0];
                coordprev(i,:)=coord(i,:);
            end
        end
    end
```

Listing 4: 周期边界

```
        for i = 1:2*N
            for j=1:3
                if coord(i,j)>(length(j))
                    coord(i,j)=coord(i,j)-length(j);
                else
                    if coord(i,j)<-(length(j)/2)
                        coord(i,j)=coord(i,j)+length(j);
                    end
                end
            end
        end
```

2.3 Push it

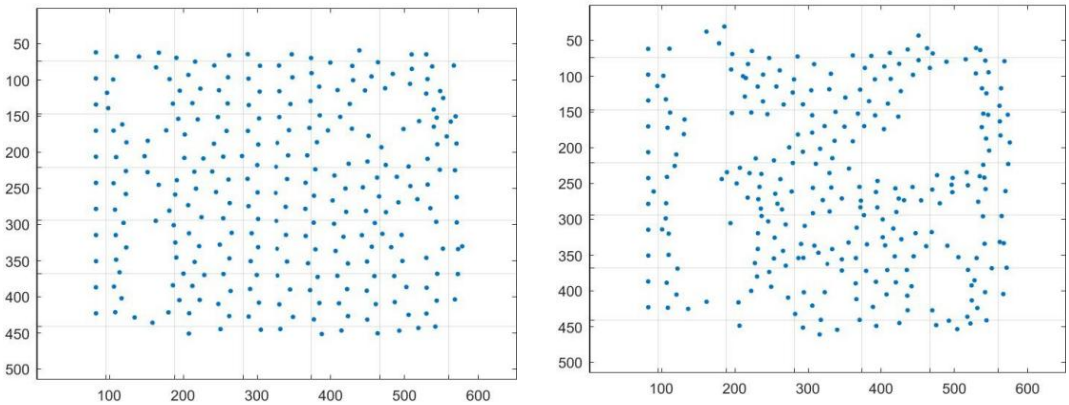
首先给出我给定边界拉力的方式：

Listing 5: 拉力边界

```
            xminjud = ismember(i,xmin(1:rows));
            xmaxjud = ismember(i,xmax(1:rows));
            yminjud = ismember(i,ymin(1:colums));
            ymaxjud = ismember(i,ymax(1:colums));
            if xminjud
                accel(i,:)=[0,0,0];
                coord(i,1) = -0.071;
                coordprev(i,:)=coord(i,:);
            elseif xmaxjud && ~(ymaxjud||yminjud)
                accel(i,:)= accel(i,:)+[1,0,0]*fx_per_atom*...
                    (1-(coord(i,1)-mean(coord(xmax(1:rows),1)))
                       /0.142)/mass(i);
            end
        end
    end
```

在生成稳定的图像之后，我测试了其平面力学特性，即固定一侧，另一侧给定固定的力，观察薄膜的行为。

我们发现，在拉力大约为lPa的时候，整体较为稳定，l0pa时，薄膜出现不稳定。



我们先观察不稳定的现象(参考xf_l0m8_l23gif):

(a)Broken1　　(b) Broken2

图 三: Comparison

多次实验后，我们看到，薄膜从撕扯端和固定端开始崩坏，这非常的符合直觉。

2.4 Stable Young's Modulus

在稳定的模型出现之后，我当然试图模拟其稳定的力学特性。

我在一侧给定拉力之后，观测薄膜末端位置的相对拉伸 $\Delta l/l$，得到结果如下：

(1)0.1Pa　　(b)1Pa

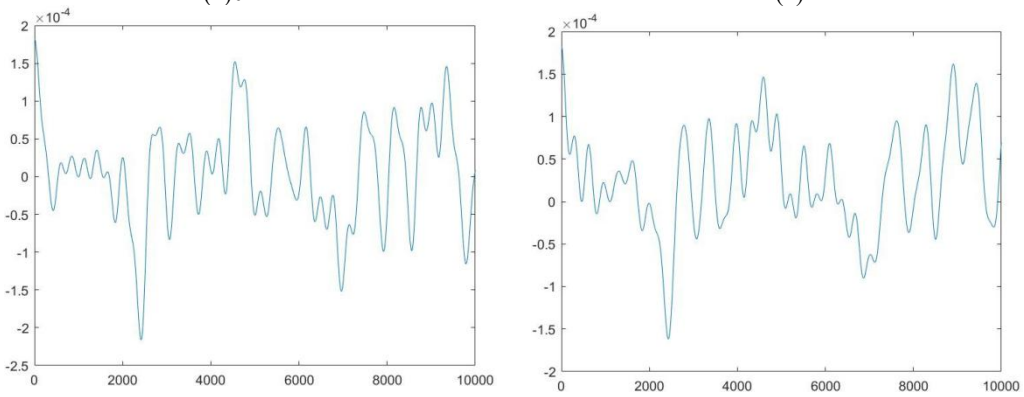


图 四: X stretch

我们发现，末端在外力的作用下会存在振动；对振动取平均，拉力确实带来了线性的拉伸。整体而言，我们发现在此 LJ模型下，取键能为 400kJ(通常认为的 C-C 键能) 的话，其杨氏模量约为 $10^5 - 10^6$Pa，这和实验测得的数据相去甚远，我们也认为，这正说明 LJ势能无法很好的模拟sp杂化的电子共价键。

Part III　Rebo on Graphene

目录

1　Rebo Potential

1.1 Concrete form of Rebo

Rebo势可以写作最近邻求和的形式：

$$\sum_j V^R(r_{ij}) - b_{ij}V^A(r_{ij}) \tag{1}$$

其中r是最近邻原子间距，VR、VA分别代表排斥势和吸引势；

$$\mathrm{b_{ij}} = \frac{1}{2}\left[b_{ij}^{\sigma-\pi} + b_{ji}^{\sigma-\pi}\right] + b_{\mathrm{ij}}^{\pi} \tag{2}$$

其中b^pi在平整的固体中取0，本次计算先只讨论石墨烯在二维平面内的弛豫；当计算表面起伏时，该项不能省略。具体各项形式为：

$$\mathrm{V^R}(r) = f^c(r)\left(1 + \frac{Q}{r}\right)Ae^{-\alpha r} \tag{3}$$

$$\mathrm{V^A}(r) = f^c(r)\sum_{n=1,2,3} B_n e\hat{}(-\beta_n r) \tag{4}$$

$$f_{ij}^c(r) = \begin{cases} 1 & r < D_{ij}^{\min} \\ [1+\cos((r - D_{ij}^{\min})/(D_{ij}^{\max} - D_{ij}^{\min}))]/2 & D_{ij}^{\min} < r < D_{ij}^{\max} \\ 0 & r > D_{ij}^{\max} \end{cases} \tag{5}$$

$$\mathrm{b_{ij}^{\sigma-\pi}} = \left[1 + \sum_{k\neq i,j} f_{ik}^c(r_{ik})G\left(\cos\left(\theta_{ijk}\right)\right)e^{\lambda_{ijk}} + P_{ij}(N_i^C, N_j^H)\right]^{-\frac{1}{2}} \tag{6}$$

其中各参数取值为：

$B_1 = 12\,388.791\,977\,98$ eV	$\beta_1 = 4.720\,452\,3127$ Å^{-1}	$Q = 0.313\,460\,296\,0833$ Å
$B_2 = 17.567\,406\,465\,09$ eV	$\beta_2 = 1.433\,213\,2499$ Å^{-1}	$A = 10\,953.544\,162\,170$ eV
$B_3 = 30.714\,932\,080\,65$ eV	$\beta_3 = 1.382\,691\,2506$ Å^{-1}	$\alpha = 4.746\,539\,060\,6595$ Å^{-1}
$D_{\min} = 1.7$	$D_{\max} = 2.0$	

θ (rad)	$G(\cos(\theta))$	$dG/d(\cos(\theta))$	$d^2G/d(\cos(\theta))^2$	$\gamma(\theta)$
0	8	—	—	1
$\pi/3$	2.001 4	—	—	0.416 335
$\pi/2$	0.375 45	—	—	0.271 856
0.6082π	0.097 33	0.400 00	1.980 00	—
$2\pi/3$	0.052 80	0.170 00	0.370 00	—
π	−0.001	0.104 00	0.000 00	—

图1 石墨烯中的Rebo参数取值[1]

篇幅限制，具体的计算表达式见代码。

1.2 Comparison with LJ potential

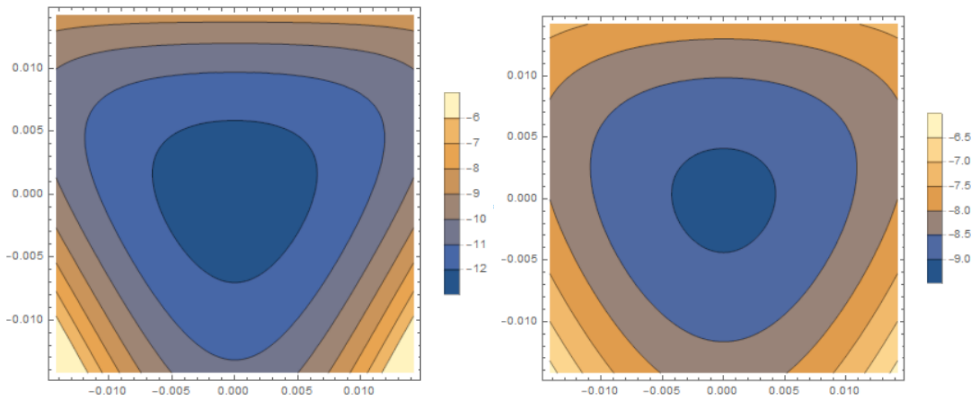


图2 LJ势和Rebo势{{x, -0.1A, 0.1A},{y, -0.1A, 0.1A}}等高线比较；单位eV

LJ势能参考池同学得到稳定收敛的参数组，即400kJ键能；Rebo势使用文献参数。容易看出，LJ势落差更大，键角与键角之间能量更低；Rebo较均匀，在原点的简谐振动模拟更好；如果原子初速度朝向键角之间，LJ模型更有可能逃逸。

2 Parameters specification

因熟悉的语言不同，且原本想进行应用方面的模拟，方便起见一开始选择python作为载体；为提高运行效率，尽量使用数组计算；参考hw11的周期边界条件取法，在所取周期周围添加一层原子模拟周期条件；按图示方式对原子编号，其中红色代表模拟周期条件的原子，蓝色代表所取周期内的原子，则计算受力时每个原子的近邻原子可以根据固定方式从编号中选取。且预计的应用模拟的状态原子在平衡位置附近，故不对最近邻原子列表进行更新。

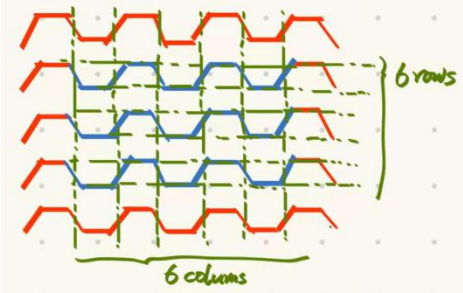


图3 原子编号方式(图示取了6行，6列原子)

同理，令红色原子加速度、速度、位移等为零则为取固定边界条件。为防止原子的一步位移与原子间距相比过小出现精度误差，代码中将原子位置分成两部分：偏离平衡位置的位移和平衡状态的位置，每次更新位移，计算力时再将两者合在一起。

加速的计算中，LJ势每个原子的加速度需要三组位移，完整的rebo势需要9组位移：计算加速度的函数corre_force(radiu1, radiu2, radiu3, radiu4, radiu5, radiu6, radiu7, radiu8, radiu9)(各自变量均为三维矢量)和对势进行求导的函数Dpotential(r1, r2, r3, r4, r5, cos2, cos3, cos4, cos5)的变量使用规范为：

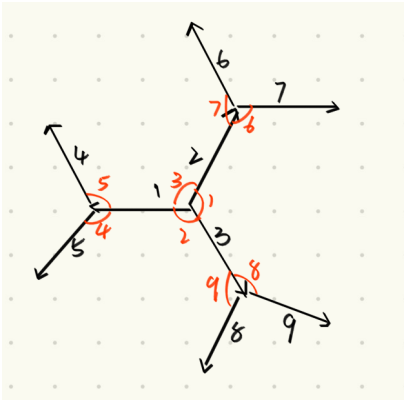


图4 加速度计算相关夹角、矢量标记规范

其中矢量4和5，6和7，8和9可以交换顺序；Dpotehtial中各矢量模顺序和夹角顺序对应。利用python数组计算较快的优点，对于同一列原子近邻原子取法相同，故可将radiu*扩展为(row/2, 3)维数组，一次计算一列的加速度；且由于加速度计算中许多矢量重复使用，这点在Rebo势的计算中更加明显，故直观上以两列原子作为计算加速度的单位，函数acc_two即为取两列原子相关原子距离计算加速度，函数accelerate即为将该过程重复遍布所取周期。

代码中coord单位nm，加速度使用dV/dr的方式计算，但带入的距离为nm，故加速度单位nm/ns^2；verlet算法更新为：

$$coord = 2*coord - coordprev + accel*timestep**2 \quad (7)$$

$$veloc = (coord - coordprev)/(2*timestep) \quad (8)$$

故veloc单位为m/s，timestep的单位为ns；代码中的timestep取值10^(-8)，故对应实际值0.01fs；池同学的稳定参数取值10^(-7),对应实际值0.1fs，从图像中约对应原子运动频率；但使用势函数在平衡位置附近的二阶导，可得碳原子振动频率约为10fs；模拟时间步小于理论值且相差较大，这也是需要进一步研究的问题。

3 Results and analysis of failure

使用LJ势模拟，原始位置取平衡位置，初始速度使用高斯分布；简单起见将原子运动限制在二维平面中。使用动能与温度的对应关系：

$$\frac{1}{2}mv^2 = \frac{3}{2}kT \tag{9}$$

$$imperf = \sqrt{\frac{kT}{m}} \tag{10}$$

故100K约合263；模拟结果是很快混乱：

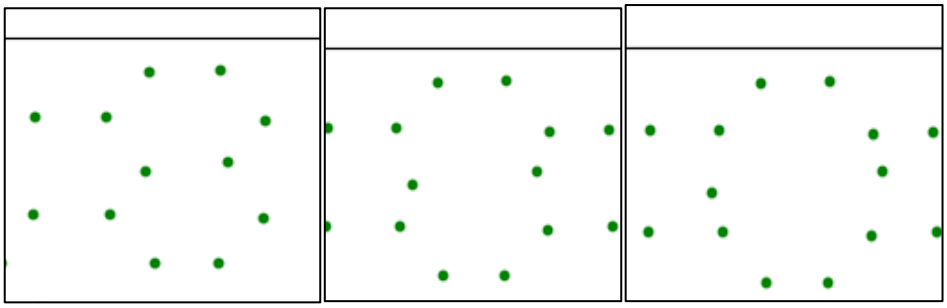


图5 LJ势模拟连续时间步

能发现两个原子脱离各自的平衡位置相背运动，最后超过截断距离，彻底脱离，进而引起系统混乱；基于以上原因，尝试用Rebo势；但Rebo势的实验结果类似，在可模拟的时间步内原子先为近似的匀速直线运动，脱离截断距离后系统混乱；为了降低模拟难度，将初始位移和速度均设为零，但结果仍发散；截取混乱前一刻的原子分布：

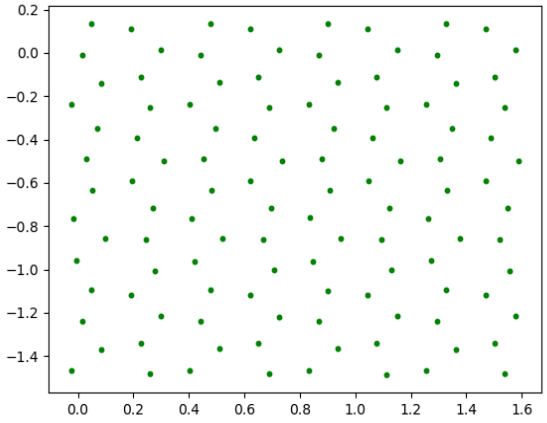


图6 Rebo势混乱前的原子分布

数值计算中由于浮点数的精度问题，一些数学运算不能取到精确的结果；当初始取平衡位置后，输出第一次计算得到的编号1(e.g.数组第0位)的原子加速度结果：

[1.77621841e-05 -2.52723694e-05 0.00000000e+00] (11)

该误差来自于周围三个最近邻原子的矢量和约为e-16，而原子偏离平衡位置后的“pot”(具体参见代码)项可达e+9,故；但该偏离不能当作微扰，因为微扰扰动时恢复力指向(微扰下新的)平衡位置，但该由于计算误差造成的平衡位置随原子位移变化，故原子不断向“平衡位置”移动，但平衡一直在改变，最终引起系统混乱；输出的第一步加速度中不为零的x方向居多，这与图6大部分原子的偏移趋势相符；该偏移与构造平衡位置时使用的np.sqrt(3)有关，而python中np.cos(np.pi/3)*2-1也正好约为e-16；python默认精度为17位，matlab默认50位，可能这造成了模拟上的差异。

周期边界和固定边界的差异影响较小。

曾考虑过减小势场来“模糊”初始加速度，但这只是相当于取更多时间步。受时间和精力限制，该模拟问题的结论止步于python计算精度的限制。

参考文献

[1] BRENNER.DW,SHENDEROVA.OA,HARRISON.JA,et al. A second-generation reactive empirical bond order(rebo) potential energy expression for hydrocarbons [J]. Journal of Physics: Condensed Matter, 2002, 14(4) : 783.

分工

第一部分：唐泽宸（1 至 6 页）

第二部分：池志豪（7 至 12 页）

第三部分：刘诗韵（13 至 17 页）

感谢吴老师、李助教本学期的授课！