華中科技大學

本科生毕业设计[论文]

折叠石墨烯层间力对热导率的影响的研究

院　　系　能源与动力工程

专业班级　热动 1110 班

姓　　名　宋琪琛

学　　号　U201111701

指导教师　杨诺

2015 年 6 月 15 日

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密口，在　　年解密后适用本授权书

2、不保密口　。

（请在以上相应方框内打“√”）

作者签名：　　年　月　日

导师签名：　　年　月　日

摘　　要

近年来，科学界和工业界对于具有优异电学、热学性质的石墨烯十分关注。折叠石墨烯作为一种石墨烯的特殊存在形式，具有不同于完美石墨烯的性质。然而目前对于折叠石墨烯的研究仅集中在小尺寸的折叠石墨烯纳米带上，对于更接近实际尺寸的折叠石墨烯则无人研究。由于石墨烯本身具有热导率随尺寸变化的规律，大尺寸的折叠石墨烯的具体性质可能与折叠石墨烯纳米带的性质不同，需要单独研究。

本文通过非平衡分子动力学对无穷大的折叠石墨烯体系进行了研究。得到了折叠石墨烯热导率的尺寸依赖关系，通过外推法得到尺寸无穷大的折叠石墨烯热导率，以表征实际尺寸的折叠石墨烯的热输运能力。得到了热导率对折叠数目依赖关系。同时对于具有不同的层间距的折叠石墨烯的热导率进行了计算，通过统计原子间作用力所传递的热流解释了热导率随着层间距减小而增大的变化规律，并通过声子功率谱密度分析了折叠处对热阻的贡献和层间作用的影响。

关键词：折叠石墨烯；非平衡分子动力学；尺寸效应；范德瓦尔斯力；功率谱密度

Abstract

Graphene has attracted the interests of researchers both in academia and industry due to its extraordinary electrical and thermal transport properties. Folded graphene, a derivative of graphene, has different properties from pristine graphene arise from its novel structure. However, most of the related studies are focused on folded graphene nanoribbons of small sizes, while folded graphene of realistic sizes is investigated by no one. Given the fact that there exists a strong size effect on thermal conductivity of graphene, the large-area folded graphene may be a different case and thus should be investigated.

Here, we use non-equilibrium molecular dynamics to study thermal transport in large-area folded graphene. The dependence of folded graphene's thermal conductivity on length and number of folds is obtained and thermal conductivity of infinitely large folded graphene is attained by extrapolation to characterize the thermal properties of folded graphene of realistic sizes. Cases of various inter-plane distances are studied and the impact of van del Waals interaction on thermal conductivity is investigated by directly recording the heat flux by long-range atomic interactions. The contribution to thermal resistance from the curve part in the structure with varying inter-plane distances is studied by calculating the vibration power spectrum.

Key Words： folded graphene; non-equilibrium molecular dynamics; size effect; van del Waals interaction; vibration power spectrum

目　　录

1 绪论

1.1 研究背景、目的和意义

煤、石油等化石能源的日渐枯竭迫使人们寻求有效可行的替代方案。以中国为例，煤炭占一次能源消耗的七成。热电厂在发电过程中不仅消耗了大量煤资源，同时产生了大量的废热。内燃机燃烧过程中有 60-70%能源以废热的形式被浪费掉[1]。中国机动车数量超过 2.5 亿，这些机动车向环境排放了大量的废热。化工厂中进行的化学过程也伴随着大量废热产生。为了利用这些废热，提高总的能源利用效率，基于热电温差发电的余热回收利用吸引了研究人员的目光。

热电发电技术可以将热能直接转化为电能，并且没有任何污染。当前热电发电技术的主要研究方向是找到热电转换效率较高的材料[2]。衡量热电材料的一个重要指标是材料的 *ZT* 因子，通常 *ZT* 因子越大，热电转换效率越高。它与材料的塞贝克系数 *S*、热导率 κ、电导率 σ 有关，可以表示为 $ZT = S^2\sigma T/\kappa$。由公式可以看出，通过调控材料的输运性质提高 *ZT* 因子可以提高材料的热电转换效率。

石墨烯是具有 sp^2 杂化电子轨道的单层碳原子结构，具有绝佳的电学性质。石墨烯中电子在狄拉克附近的群速度十分大，约 8.5×10^5 m/s, 使得石墨烯具有卓越的电子输运性质, 实验[3]证明了石墨烯具有较高的电导率，约 5.3×10^{-4} ohm^{-1}。通过改变石墨烯的结构带来的量子约束以在石墨烯能带中产生带隙，为将石墨烯应用于下一代微电子和热电装置带来可能[4]。石墨烯中的声子平均自由程达到微米级别，使得石墨烯具有优异的热输运性质。通过拉曼光谱实验测量的热导率约 2000−5000 W mK^{-1}[5], 通过电学测量得到石墨烯热导率约为 600W mK^{-1}(300K)[3]。不同研究小组得到的实验值差异较大。理论计算得到的石墨烯热导率比天然物质中热导率(约 2000W mK^{-1})最高的金刚石还要高[6]。对于石墨烯材料的塞贝克系数，实验得到数值的很小，在 300K 时约 8.2 μV/K [7]。

为了提高石墨烯热电材料的 *ZT* 因子，设法提高塞贝克系数是一种方法，而调制石墨烯的热导率也是一种方法（本课题专注于这种方法）。过去几年，人们在显微镜下发现了折叠石墨烯的存在[8]；随后，研究人员找到了控制形成这种折叠结构的方法[9]（如图 1-1）。研究表明，在石墨烯结构中创造折叠是一种有效降低石

墨烯热导率的方法[10, 11]。

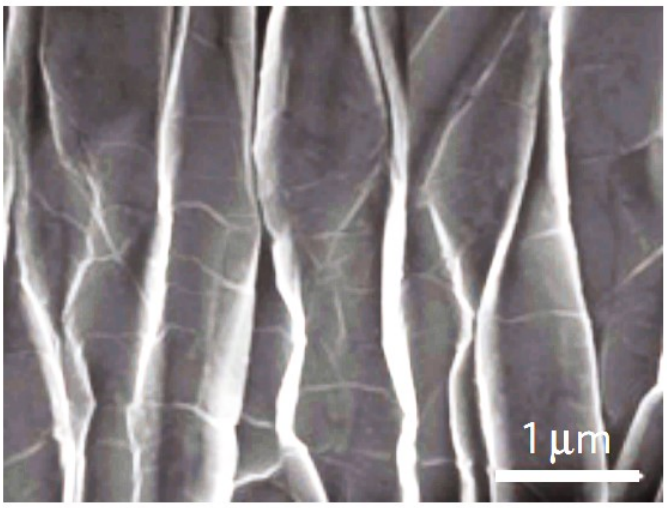

图 1-1　利用基底释放应力制备的折叠石墨烯的扫描电子显微镜（SEM）图像[9]。

1.2　国内外研究概况

对于石墨烯热导率的研究方法主要分为实验和模拟两种方法。多层折叠的石墨烯在结构上与多层石墨烯有相似之处。美国研究人员 Jang 等人[12]通过拉曼测量的方法得到了随着多层石墨烯层数减小而增大的热导率，产生这种现象的原因是层数变小导致了减小了的低能量声子层间耦合和逐渐变弱的层间散射（主要是声子 Umklapp 散射，指的是三声子散射中，两个声子碰撞产生第三个声子越过第一布里渊区的一种散射）。

目前对折叠石墨烯的研究主要以模拟为主。常见的模拟方法是解声子玻尔兹曼方程、分子动力学等。折叠石墨烯中的弯折部分与碳纳米管的结构有相似性。Lindsay 等人[13]发现碳纳米管的热输运性质与其曲率相关，并且得到了热导率随着曲率变小先减小后增大并最终趋于常数的规律。他们对此的解释是：曲率较大时，随着曲率减小，半径增大，弹道输运（ballistic transport）的特性越来越不明显，热导率降低；曲率较小时，由于弯曲打破了石墨烯原本有的反射对称性（reflection symmetry）从而使得选择准则（selection rule）更加复杂，曲率越小，声子散射越弱，热导率越高，最终趋于石墨烯的热导率。但是求解声子玻尔兹曼方程的局限性在于只能处理较小的体系；玻尔兹曼方程默认声子输运是满足粒子

特性的，在声子平均自由程与系统特征尺寸相近时，则必须考虑声子的波动性。与多层石墨烯、碳纳米管的类比给我们带来了启示：石墨烯结构变异和层间作用对声子散射的贡献。

国内研究人员 Yi[14]等人利用分子动力学（Molecular Dynamics, MD）分析了具有折叠结构的石墨烯展开变为平面（planar）石墨烯的临界温度，但是没有研究其热输运性质。Xie 等人[15]通过格林函数（Green's function）的方法得到了折叠石墨烯的电输运性质。Yang 等人[10]通过非平衡分子动力学（non-equilibrium molecular dynamics, NEMD）计算了折叠石墨烯纳米带的热导率，并找到了热导率对层间距离以及折叠数目的依赖关系，即层间距离越小，折叠数目越多，热导率越低；通过非平衡格林函数（non-equilibrium Green's function, NEGF）分析了声子输运情况，并用相图分析了散射增强的原因，即折叠石墨烯的色散关系说明其具有更大的 Umklapp 散射的概率；然而模拟的体系比较小且结果没有具体呈现出层间力的效应。Zhang 等人[11]通过与同等表面积的碳纳米管作对比，采用 NEMD 研究了折叠石墨烯纳米带的热导率，辅以振动本征矢分析（vibrational eigen-mode analysis），探讨了折叠石墨烯边缘对热导率的影响，得到具有不同边界的石墨烯纳米带，热导率相差较大；然而他们的计算并没有考虑多层折叠的情况。

总的来说，采用实验方法对折叠石墨烯热导率的探索很少，而模拟的工作相对丰富。上述采用了分子动力学的研究的一大优势是，能更加直截了当地了解具有结构差异的折叠石墨烯的热输运性质与结构的关系。但几乎所有的模拟都在研究较小的模拟单元（纳米带），对于表征实际尺寸的折叠石墨烯的热导率则无能为力。本课题将通过分子动力学处理相对较大的体系，并采用合理的推论，得到更接近真实尺寸的折叠石墨烯热输运的性质。同时将对层间作用力对于声子输运和热导率的影响作用进行详细探讨，探索通过折叠降低石墨烯热导率的背后机制。

1.3 分子动力学简介

经典分子动力学（classical molecular dynamics simulations）是模拟原子间相互作用的时间演化过程的一种计算方法，它主要遵守经典力学中的牛顿第二定律，

$$\mathbf{F}_i = m_i \mathbf{a}_i \qquad (1\text{-}1)$$

其中，i 是原子编号，m_i 是第 i 个原子的质量，$\mathbf{a}_i$ 是加速度，$\mathbf{a}_i = d^2\,\mathbf{r}_i/dt^2$ ，$\mathbf{F}_i$ 是原子所受作用力。给出原子的初始位置和速度，通过势函数可以求得每个原

子所受的力；根据原子受力情况计算加速度就可以计算下一步原子的位置和速度。EMD[16]依据处于平衡态时能量在系统中的涨落，记录系统的自相关函数，采用 Green-Kubo 方法来计算热导率。NEMD 则通过在系统中创造非稳态获得系统的物理性质，求热导率的通常做法是通过创造温度梯度和热流。

本次模拟计算采用了 NEMD，主要原因是 NEMD 模拟的是非稳态的输运性质，且折叠石墨烯平面内输运能力最强，即便产生了折叠，热流依然主要沿着层内输运，这个时候利用傅利叶定律定义的计算得到的热导率更能表征这种平面内的输运能力。

1.3.1 模拟过程

（1）设置原子初始位置和速度

一般原子的初始位置设置为平衡位置，初始速度按照某种方式赋值。常用的一种方式是按照某一温度 T_{init} 下产生遵循高斯分布的随机数，则初始速度是的体系满足初始温度 T_{init}。一般 T_{init}=2T_0，T_0 是处于平衡态时模拟所处的温度。这样做是因为初始状态势能为零，动能等于内能，随着模拟的进行，势能和动能趋向于相等且各自等于内能一半；取两倍关系符合能量守恒，使得模拟更快达到稳态。

（2）模拟原子的运动

MD 的核心部分是势函数，它的选取决定了计算的准确性。力可以通过势函数相对于原子位移的梯度得到，

$$\mathbf{F}_i = -\nabla_{\mathbf{r}_i} V\left(\mathbf{r}_1,...,\mathbf{r}_N\right) \tag{1-2}$$

V 是势函数，可能是采用经验势函数，也可能利用密度泛函理论（DFT）计算修正势函数。通过牛顿第二定律得到每个原子的加速度 $\mathbf{a}_i$。利用 Velocity-Verlet 算法[17]可以得到每一步的原子的速度和位置。具体的计算过程如下，

$$\mathbf{r}\left(t+\Delta t\right) = \mathbf{r}\left(t\right) + \mathbf{v}\left(t\right)\Delta t + \left(1/2\right)\mathbf{a}\left(t\right)\Delta t^2 \tag{1-3}$$

$$\mathbf{v}\left(t+\Delta t/2\right) = \mathbf{v}\left(t\right) + \left(1/2\right)\mathbf{a}\left(t\right)\Delta t \tag{1-4}$$

$$\mathbf{a}\left(t+\Delta t/2\right) = -\frac{1}{m}\nabla V\left(\mathbf{r}\left(t+\Delta t\right)\right) \tag{1-5}$$

$$\mathbf{v}\left(t+\Delta t\right) = \mathbf{v}\left(t+\Delta t/2\right) + \left(1/2\right)\mathbf{a}\left(t+\Delta t\right)\Delta t \tag{1-6}$$

1.3.2 物理量计算

MD 是统计力学方法的一个分支，如果用 Γ 表示研究体系内各个原子的坐标和动量，用 $H(\Gamma)$表示体系的哈密顿量（Hamiltonian），则一个微正则系综满足 $\delta(H(\Gamma)-E)$，δ 是狄拉克(Dirac)函数。该关系保证系统处于能量为 E 的状态。对于正则系综，温度 T 为常数，分布满足玻尔兹曼(Boltzmann)函数的形式 exp(－$H(\Gamma)/k_{\mathrm{B}}T$)。统计物理指出，物理量可以由系综平均值得到，

$$\langle A\rangle=\frac{1}{N_T}\sum_{t=1}^{N_T}A(t) \tag{1-7}$$

其中，$A(t)=f\left(\mathbf{r}_1(t),...,\mathbf{r}_{\mathbf{N}}(t),\mathbf{v}_1(t),...,\mathbf{v}_{\mathbf{N}}(t)\right)$为系统某一时刻一系列动量和位置。 分子动力学满足杜龙—佩蒂特极限（Dulon-Petit limit, $C_p=3N_Ak_B$），温度可表示为，$\langle T_\alpha\rangle=\frac{1}{N_TN}\sum_{t=1}^{N_T}\sum_{i=1}^{N}\frac{m_iv_{i,\alpha}^2(t)}{k_B}$，$\alpha$ 为坐标轴方向。值得注意的是石墨烯的德拜温度很高（平面内模式为 2300，Z 模式 1287 K）[18]，远高于室温，因此室温下的 MD 模拟是不能正确表征系统的热容的。但是最后计算得到的热导率与实验值相差不大，所以这种 MD 模拟又是可取的[19, 20]。

1.3.3 控制模拟进行的条件

（1）边界条件

受制于计算资源所限，MD 所能研究的体系是有限的，通常在纳米尺度。有限的系统必然存在边界，且边界的影响常常是显著的。对于固定边界（fixed boundary）处于固定边界的原子动能为零，在整个模拟过程中，边界原子的位置固定不变。对于自由边界（free boundary），处在边界的原子没有被人为施加约束，与边界内部原子没有本质区别，但是通常在边界的原子的邻居数量少于内部原子。对于周期性边界（periodic boundary），处在边界的原子被人为施加了与其他原子的作用。

如图 1-2，对于具有周期性边界条件的模型，一个处在边界的原子 a 不仅与内部原子有作用，同时与跨过整个研究体系的另一个边界的原子 b 具有相互作用。相当于一个边界原子与一个“虚拟”的原子 b’具有相互作用，这个“虚拟”的原子 b’的速度、加速度与跨过整个体系的另一个原子 b 完全相同，但是位置坐标相差一个固定值 L（模拟系统的某一维度的长度）。这样就避免了边界原子邻居数目少于内部原子的问题。

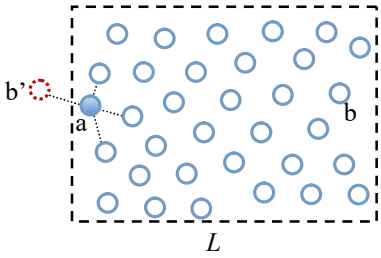

图 1-2　周期性边界示意图，虚线表示具有作用力的原子对

（2）热库

NEMD 需要在系统中产生温度梯度，热库可以保持这种温度差。热库体现为动力学方程之外的力学作用项。一种常用的一种热库叫做 Nosé-hoover 热库[21, 22]。如图 3，以一维原子链为例，处在高温和低温 Nosé-hoover 热库中的原子[23]分别满足，

$$\dot{\mathbf{p}}_{i,\mathrm{H/C}} = \mathbf{F}_{i,\mathrm{H/C}} - \zeta_{\mathrm{H/C}}\mathbf{p}_{i,\mathrm{H/C}} \tag{1-8}$$

式中，下标 H 和 C 分别代表高温和低温热库中的原子，$\mathbf{p}_{i,\mathrm{H/C}}$ 表示原子的动量，$\mathbf{F}_{i,\mathrm{H/C}}$ 表示由势函数计算得到的力，参数 $\zeta_{i,H/C}$ 满足，

$$\dot{\zeta}_{i,\mathrm{H/C}} = \frac{1}{\theta_{\mathrm{H/C}}}\left(\frac{\left|\mathbf{p}_{i,\mathrm{H/C}}\right|^2}{m_{i,\mathrm{H/C}}k_B T_{\mathrm{H/C}}} - 1\right) \tag{1-9}$$

式中，$\theta_{H/C}$ 是原子与热库的耦合强度，$m_{i,H/C}$ 是原子的质量，$T_{i,H/C}$。通过公式，$J_{\mathrm{H/C}} = \sum_i \boldsymbol{f}_{i,\mathrm{H/C}} \cdot \mathbf{v}_i = \sum_i -\zeta_{\mathrm{H/C}}\mathbf{p}_{i,\mathrm{H/C}} \cdot \mathbf{v}_i$ 可以记录热库的能量损失/能量获得作为系统热流，其中 $\boldsymbol{f}_{i,H/C}$ 是热库对应的附加力项。

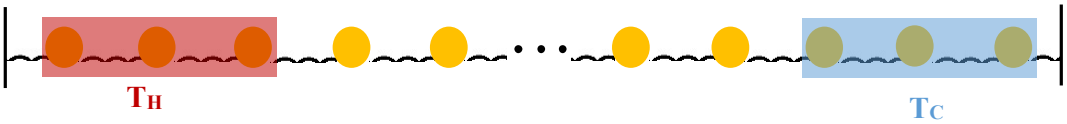

图 1-3　一维原子链中的相互作用示意图，左侧和右侧分别被施加了热源和冷源

1.3.4 从模拟获取声子的行为

（1）声子态密度谱和功率谱

MD 模拟的输运过程，实际模拟了声子输运的过程。声子态密度（density of states, DOS）描述了每种能量可能占有的态的数目。某个能量的 DOS 越大，该能量下可能占据的态越多。DOS 为零则表示这个能量下不存在任何声子态。一段连续的 DOS 为零的能量范围表示了存在一个声子带隙（bandgap）。声子态密度谱能够展现系统中原子振动满足的规律，帮助了解系统的输运性质。

通过计算原子速度的自相关函数 γ (t)可以得到 DOS，

$$\gamma(t)=\left\langle\sum \mathbf{v}_i(t)\cdot\mathbf{v}_i(0)\right\rangle\Big/\left\langle\sum \mathbf{v}_i^2(0)\right\rangle \tag{1-10}$$

对速度自相关函数进行傅利叶变换即可得到归一化的频谱。

Wiener-Khintchine 理论[24]证明，对于广义平稳（Wide-sense stationary, WSS）过程，其自相关函数的傅利叶变换是功率谱密度（Power Spectrum）。功率谱密度由速度的傅利叶变换得到，

$$P(\nu)=\frac{m}{\nu_c}\left|\int_{-\infty}^{+\infty} v(t)e^{-2\pi i\nu t}dt\right|^2 \tag{1-11}$$

其中，ν_c 是系统的最高频率。而功率谱密度也可以表示为，

$$P(\nu)=DOS(\nu)\cdot h\nu\cdot f(E) \tag{1-12}$$

其中，f(E)是分布函数。声子满足 Bose-Einstein 分布，但是经典 MD 模拟，基于高温假设，因此有，

$$f(E)=\frac{k_BT}{h\nu} \tag{1-13}$$

（2）声子平均自由程

不论是电子还是声子，其输运能力由载流子的散射决定，没有任何的散射的输运称为弹道输运，这种情况下电导率/热导率的大小不会被约束。声子在输运过程中会发生散射，正是这种散射产生了热阻。通过气体动理论（kinetic theory of gases）可以得到声子输运对应的热导率为 $k=\frac{1}{3}nc_vu\lambda$ ，其中 n 为单位体积内的载声子浓度，c_v 是比热，u 是声子群速度，λ 是声子平均自由程，$\lambda=u\tau$ ，τ 是连续两次碰撞发生的平均时间间隔，被称为弛豫时间（relaxation time）[25]，因此可以将平均自由程理解为声子在相邻两次碰撞间输运的平均距离。

2　折叠石墨烯模拟细节

2.1　模拟对象结构及边界条件

如图 2-1，折叠石墨烯的结构是通过在一个等长度的石墨烯上加上若干个弯折得到的。y 方向为宽度方向，z 方向为垂直于平面层的方向。折叠石墨烯的长度被定义为，$L = (n+1)\,L_{plane} + n\,L_{curve}$，n 是折叠的数目，$L_{plane}$ 和 L_{curve} 分别是平面部分和弯折部分的长度。图 2-2 (a)是一个总长为 *L*（共 300 层原子，*L*=36.7 nm）的折叠石墨烯的结构示意图。所有的弯曲部分都是由 5 层原子组成的，因此 $L_{curve} \equiv (5+1) \times \sqrt{3}/2 \times a$，其中 a 为石墨烯的晶格常数，a=1.418Å。在未进行模拟时，弯曲部分的原子在侧视图中处在一个同心圆上，自身构成一个圆内接正十二边形的一半。初始时，相邻的平面之间的距离为 0.474 nm。

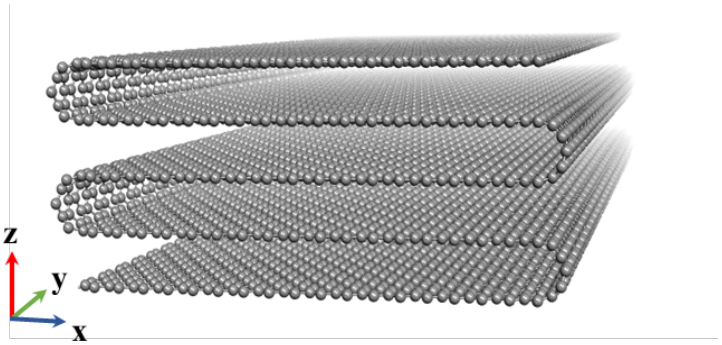

图 2-1 折叠石墨烯的三维视图

折叠石墨烯上下各有一层基底，以保证石墨烯在 z 方向没有漂移。同时第一层原子和最后一层原子在 x，y 方向上被设置了固定边界条件，以防止石墨烯结构出现较大的变形和在 x，y 方向上漂移。最靠近第一层的五层原子（第 2 到第 6 层原子）被施加高温热浴，温度维持在 $T_h = T_0\,(1+\Delta)$。而最靠近第 N 层原子的五层原子（第 N－5 到第 N－1 层原子）被施加高温热浴，温度维持在 $T_h = T_0\,(1-\Delta)$。本次模拟 T_0=300K，Δ=0.1。值的注意的是，在这种设置下，沿着 *L* 的方向温度梯度

约为 10^8 K/m，这显然是无法在现实中实现的，但是在这种非真实的模拟设置下得到的结果与实验值依然是吻合的[19, 20]，因此依然得到了采用。

如图 2-2(b)，为了获得无限大的折叠石墨烯的性质，宽度方向上（y 方向）使用了周期性边界。结构的宽度为 21.27 nm（10 个 C 原子）。之所以选取这个宽度，是因为任何大于该宽度的结构的热导率与该宽度下的结构热导率几乎没有区别，即热导率在宽度方向上在 21.27 nm 时已经收敛。因此，从图 2-1 中也可以看出，在宽度方向石墨烯是无限延伸的。

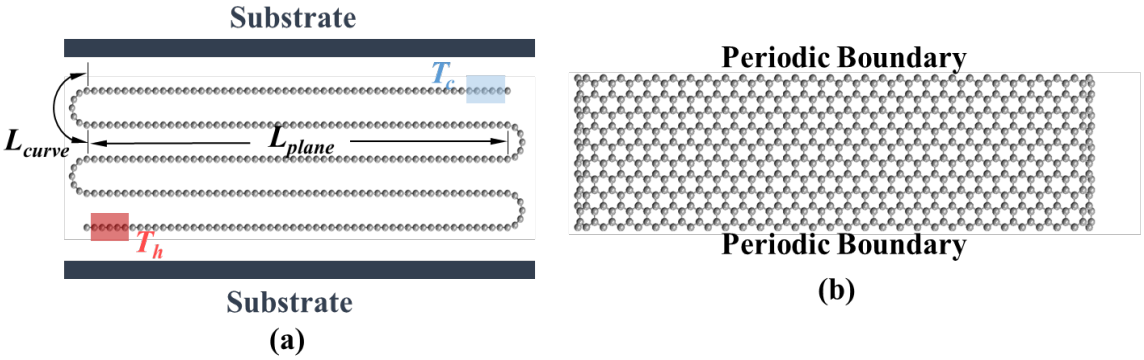

图 2-2　模拟体系示意图 (a) 弛豫前的折叠石墨烯的结构，L_{plane}, L_{curve} 分别为 6.75 nm 和 0.737 nm，总长度 L 为 36.7 nm，折叠数目为 4 折。靠近两端分别被施加了热源（红色）和冷源（蓝色），上侧和下侧分别有一层基底模拟基底的作用。最两端原子在 x, y 方向被固定。(b) 宽度为 2.13 nm，宽度方向上使用了周期性边界。

2.2　模拟的参数设置

2.2.1　势函数的形式

MD 模拟中，势函数的选择直接决定了模拟结果的精度。本次模拟使用了 Morse[26, 27]势函数来描述最近邻碳原子之间的成键作用，是一种二体力作用势；使用了简谐余弦势函数（harmonic cosine energy potential）[28]来描述中心原子 i 及两个最近邻原子 j，k 形成一个夹角 θ_{ijk}，这个夹角在平衡位置处等于 $\theta_C = 120°$，是一种三体作用势；采用了 Lennard-Jones 12-6 势函数来描述碳原子间长程作用，例如处于不同平面层原子间的范德瓦尔斯（van der Waals）相互作用[14]。综合上

述三种势函数，最终采用的势函数形式如下，

$$U\left(r_{ij},\theta_{ijk}\right)=K_{Cr}\left(e^{-\gamma\left(r_{ij}-r_C\right)}-1\right)^2+\frac{1}{2}K_{C\theta}\left(\cos\theta_{ijk}-\cos\theta_C\right)^2$$

$$+4\varepsilon_{CC}\left[\left(\frac{\sigma_{CC}}{r_{ij}}\right)^{12}-\left(\frac{\sigma_{CC}}{r_{ij}}\right)^{6}\right] \qquad (2\text{-}1)$$

右手边，第一项表示只与最近邻的成键两原子间距离相关的 Morse 势函数，第二项表示只与原子 i，j，k 夹角的余弦相关的简谐余弦势函数，第三项表示只与具有范德瓦尔斯力的原子间距离相关的 Lennard-Jones 势函数。力常数 K_{Cr} = 478.9 kJ mol^{-1}， $K_{C\theta}$ = 562.2 kJ mol^{-1}， ε_{CC} = 0.2880 kJ mol^{-1}；平衡位置几何参数 r_c = 1.418Å，θ_C = 120°；范德瓦尔斯力的参数 σ_{CC} = 0.3407 nm；其他参数 γ =2.187 Å^{-1}。

2.2.2 模拟过程的设置

所有进行的模拟都采用了 Verlet 算法。时间步长被取为 0.5 fs，模拟的步数为 6×10^6 步，总的模拟时间为 3 ns。在最初的 3×10^5 步，上下两个基底的 Z 方向位置被不断向着压缩内部的折叠石墨烯的方向运动。对于演化过程的每一步，压缩的距离约为 10^{-6} nm，相对于原子的位移（约 10^{-4} nm）较小，避免了大的形变甚至结构中的键断裂。经过这样的压缩过程，折叠石墨烯平面层间的距离从最初的 0.474 nm 变为 0.35 nm。此后，基底的位置不再变化，折叠石墨烯平面层间的距离基本维持在 0.35 nm。为了探索不同的层间力的带来的不同的输运性质，进行了层间距离最终被压缩到 0.30 nm 的一组模拟。

采用了 12 组不同的初始条件，初始的速度按照麦克斯韦分布分布进行初始化，原子初始位置都在平衡位置。使用 MPI 程序接口在 12 个 CPU 上同时进行计算，最后对 12 个核心所计算出的结果进行平均。

2.3 热导率的计算

NEMD 中常采用傅利叶定律计算热导率，其形式如下，

$$\kappa=-\frac{J}{A\nabla T} \qquad (2\text{-}2)$$

其中，J 为系统中的热流，单位为 W；∇T 是温度梯度，单位为 K/m；A 为横截面积，单位为 m^2。在这里 A 为宽度与厚度的乘积，一般石墨烯的厚度被定义为 0.335 nm（与石墨的层间距一致），因此 A=0.713nm^2。

2.3.1 温度梯度的计算

处在热浴中的原子的温度是恒定的，由于高温和低温热浴施加在系统的两端，因此系统内部将形成一个温度梯度。但是由于采用的 Nosé-hoover 热库是一种人为的作用，在热源附近经常会形成较大的温度下降（附录图 -1）。为了减小这种影响，选择第六层原子所处的温度为 T_h，第 N-5 层所处的温度为 T_c。

$$\nabla T=\frac{T_h-T_c}{L}=\frac{T_{6\text{th}}-T_{(\text{N-5})\text{th}}}{L} \tag{2-3}$$

其中，每层原子的温度由 1.3.2 中的公式得到，即第 i 层的原子温度为，

$$T_{i\text{th}}=\frac{1}{3}\cdot\frac{1}{10N_T}\sum_{t=1}^{N_T}\sum_{j=1}^{10}\frac{m_{10(i-1)+j}v_j^2(t)}{k_B} \tag{2-4}$$

式中的 1/3 表示三个维度的温度的平均。

2.3.2 热流的计算

NEMD 常常采用通过热源记录热流的方法，热源单位时间所做的功就是系统中的热流。1.3.3 中描述了通过热源的记录热流的方法。在热源处记录的单位时间的能量失去为 ε_1，冷源处记录的单位时间的能量获得为 ε_2，则热流 $J=(\varepsilon_1+\varepsilon_2)/2$。

注意到，层间的范德瓦尔斯作用可以直接参与热输运，把热量从热源送到冷源。根据公式，可以通过记录范德瓦尔斯力与速度的乘积原子间通过范德瓦尔斯力传递的热流，具体形式如下[29-31]，

$$Q_{i\to j}=\frac{1}{2}\left\langle \mathbf{F}_{ji}\cdot\mathbf{v}_j-\mathbf{F}_{ij}\cdot\mathbf{v}_i\right\rangle \tag{2-5}$$

其中，$Q_{i\to j}$ 表示稳态时从 i 原子传递到 j 原子的能流，$\mathbf{F}_{ji}$ 是原子 j 受到的原子 i 施加的的作用力（$\mathbf{F}_{ji}=-\mathbf{F}_{ij}$），$\mathbf{v}_i$，$\mathbf{v}_j$ 分别是第 i 个原子和第 j 个原子的速度。i 原子和 j 原子分别处在所研究的热流通过的横截面两侧。如果单独统计范德瓦尔斯力所传递的热流，则只需要令 $\mathbf{F}_{ji}$ 为 i 原子和 j 原子间的范德瓦尔斯力，则系统中由范德瓦尔斯力传递的能量流为 $\sum_{i=1}^{a}\sum_{j=1}^{b}Q_{i\to j}$，a，b 是横截面两侧的原子数量。

为了研究层间作用对于热输运的影响，可以直接记录由范德瓦尔斯力所传递的热流。如图 2-3 所示，箭头表示的是通过层间范德瓦尔斯力直接传递的热流，热流从截面一侧传递到另一侧。尽管研究的折叠石墨烯存在总长变化或者折叠变化的情况，但是研究体系具有很强的相似性，因此统一将假想的热流截面选取在第二层平面和第三层平面之间。

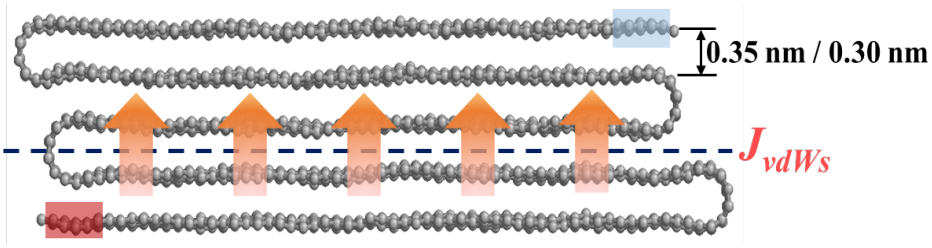

图 2-3　通过式 2-5 记录通过某假想截面的热流的示意图。虚线表示热流通过的截面，处在自下向上第二个平面和第三个平面部分之间。箭头表示通过层间范德瓦尔斯力直接传递的热流 J_{vdWs}。对于不同层间距、不同尺寸或折叠数目的情况，虚线位置都取在自下向上第二、三平面部分之间。

3　模拟结果及讨论

3.1　折叠石墨烯中的温度梯度

图 3-1 是在如上述设置的模拟条件下最终形成的温度梯度，横坐标是折叠石墨烯的长度。两侧施加了热浴的部分，温度梯度保持为零。内部原子沿着长度方向形成了温度梯度，依据傅利叶定律，将在长度方向上产生热流。具有弯折的折叠石墨烯中的温度没有呈现像二维结构的完美石墨烯呈现出线性的分布，而是呈现出阶梯状的分布。在弯曲部分（8 nm，15 nm，22 nm，29 nm 各自附近），温度梯度的值更大。在水平部分，温度随着长度的增大变化很小，接近水平，表示平面

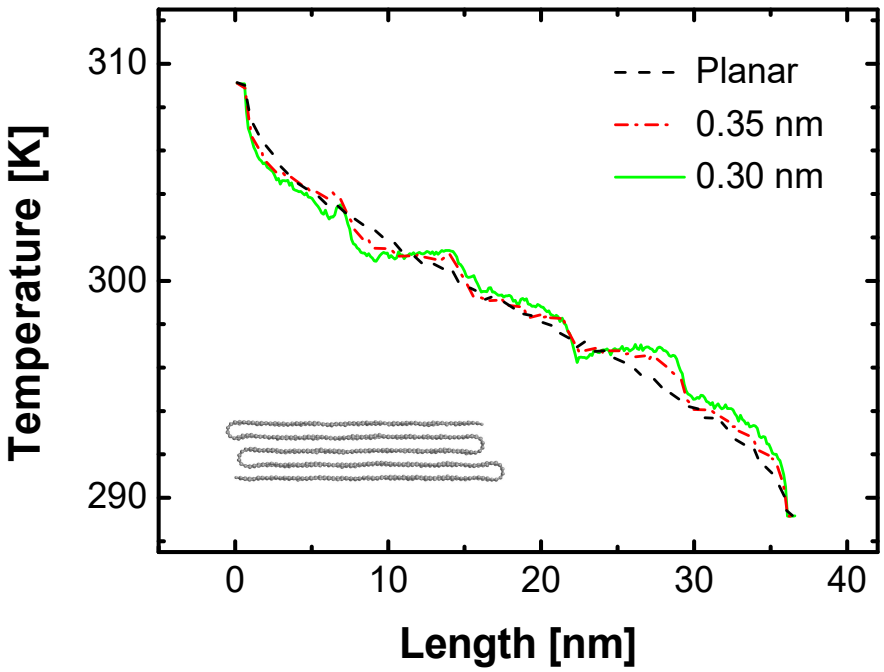

图 3-1　折叠石墨烯中沿着长度 L 的方向的温度梯度，其长度 L 为 36.7 nm，折叠数目为 4。点划线表示层间距为 0.35 nm 的折叠石墨烯，实线表示层间距为 0.30 nm 的折叠石墨烯，虚线表示相同总长的完美石墨烯。小图是弛豫后的结构，弯曲部分位于 8 nm，15 nm，22 nm，29 nm 各自附近。

层内的温度变化很小，热导率相对于弯曲部分更大。随着层间距减小，层间作用的增强，这种阶梯状分布更加明显，平面部分的温度差异更小，甚至在局部出现相反的温度梯度。

平面部分中的这种近似水平分布的温度可能源自于折叠结构引入的层间作用。相邻两平面层内各自有下降的温度分布，各自有热流输运。然而层间的范德瓦尔斯力也可以传递热流，且由于相邻平面层的原子比处在同一平面层的原子的温度差异更大，某一层的高温部分和相邻一层的低温部分直接接触，将有一定比例的热流直接从层间传递。由于平面层间热流的存在，平面层内的温度梯度被削弱。当层间距减小，层间作用增加时，层间热流可能增大，使得平面层内的梯度进一步削弱。原先平面层的低温部分可能因为与相邻一层的高温部分接触，温度变得比同层内其他位置高，这解释了相反的温度梯度的出现。下文将利用温度梯度求得热导率，并分析层间作用对热导率的影响来证明上述假设。

3.2 无穷大体系的热导率

在 NEMD 模拟中，模拟得到的结果常常是具有很强的尺寸依赖性。对于尺度较小的系统，热导率通常与尺寸有关，并且热导率的数值被尺寸的大小所限制。这是因为声子的平均自由程收到了系统边界的约束。这种规律被称作卡斯米尔极限(Casimir limit) [32]。系统的实际平均自由程可以表示为[20]

$$\frac{1}{l_{eff}}=\frac{1}{l_{\infty}}+\frac{1}{l_{sys}} \tag{3-1}$$

如果认为声子群速度 v_c 与尺寸无关，则由 1.3.3 中式 $k=\frac{1}{3}nc_v v_g l$ 知，

$$\frac{1}{\kappa}=\frac{1}{\kappa_{\infty}}\left(1+\frac{l_{\infty}}{l_{sys}}\right) \tag{3-2}$$

这里，l_{∞}是无穷大的系统中的声子平均自由程，l_{sys} 是系统的尺寸, κ_{∞}是无穷大系统的热导率。由公式可以看出，$1/\kappa \sim 1/l_{sys}$。通过模拟具有不同 l_{sys} 的体系，得到 $1/\kappa$ 与 $1/l_{sys}$ 的关系，用直线拟合所得到的热导率数据，采用外推的方法的到 $1/l_{sys}$ 为零的情况，即得到了 $1/\kappa_{\infty}$。

对于一维纳米线，l_{sys} 通常是纳米线的直径[33]。对于二维的薄膜，l_{sys} 通常是薄膜的厚度的两倍[33]。对于三维系统，l_{sys} 通常是模拟体系的尺寸 L 的四分之一[20]。（这里 l_{sys} 没有呈现与维度相关的规律性是因为，讨论一维、二维系统时，使用了

面积与体积比表征散射的强弱；讨论三维系统时，则是通过模拟具有不同大小的周期性边界的三维体系得到的结论）。对于我们研究的折叠石墨烯，尽管是三维体系，但是不具有各项同性，不能够简单地直接认为 $l_{sys} = L/4$，因此为了得到无穷大的折叠的热导率，必须找到正确的 l_{sys} 的表达形式。

如图 3-2，热导率的倒数 $1/\kappa$ 与 L_{plane} 的倒数 $1/L_{plane}$ 成线性关系。对于某一个固定总长 L 的折叠石墨烯，当折叠数目 n 变大时，由 $L = (n+1) L_{plane} + n L_{curve}$ 知，L_{plane} 将变小，$1/L_{plane}$ 将变大。不同颜色代表不同总长度 L 的折叠石墨烯。同一总长度下，n=4，5，6 和 7，对应着同一种颜色的自左向右四个数据点。对于某一固定长度的折叠石墨烯，折叠数目的增加，L_{plane} 减小时，热导率减小，这与 Yang[10] 的结论一致。相应颜色的直线表示对数据点的线性拟合，这种线性关系表明对于固定总长 L 的折叠石墨烯，热导率 $\kappa \sim L_{plane}$。

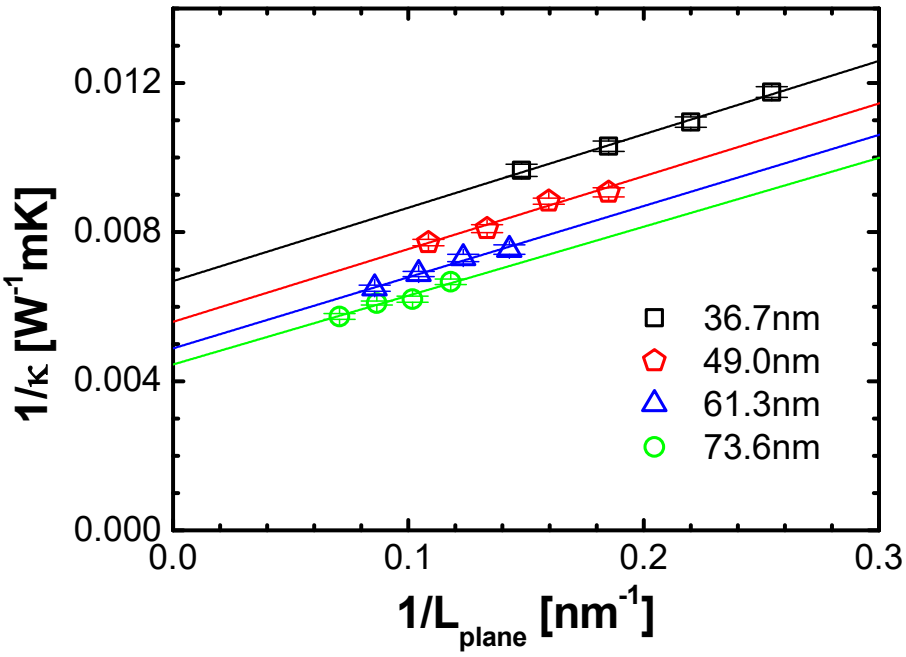

图 3-2　层间距为 0.35 nm 的折叠石墨烯热导率的倒数与平面部分尺寸 L_{plane} 的倒数的关系。不同颜色的点代表具有不同总长 L 的折叠石墨烯，L 的取值有四个，为 36.7 nm, 49.0 nm, 61.3 nm 和 73.6 nm。对于相同 L 取值的情况，注意到 L_{plane} 的数值的变化是因为折叠的数目发生了变化。

这种线性关系可能是源自于声子自由程被边界所限制。石墨烯的声子平均自由程约 10^2 nm[34]。所研究的体系 x 方向上最大的尺寸不超过 15 nm。如果认为此时系统的平均自由程被 x 方向的边界所约束，那么平面内的声子输运具有弹道输运的性质，热导率随着尺寸增大而线性增大[34, 35]。

注意到，对于一个总长 L 固定不变的石墨烯，L_{plane} 的取值是离散的，依赖于 L 和 n 的取值。上述线性关系外推至 L_{plane} 趋于无穷大的情况需要 L 趋于无穷大，而这是模拟所不能实现的。此外，从图中可以明显看出，对于具有相同的折叠数目 n 的折叠石墨烯，L 的取值不同时，热导率是不同的。因此，图 3-2 揭示了 L_{plane} 影响热导率 κ 的因素之一，但不是唯一因素，总长 L 也是影响因素之一。

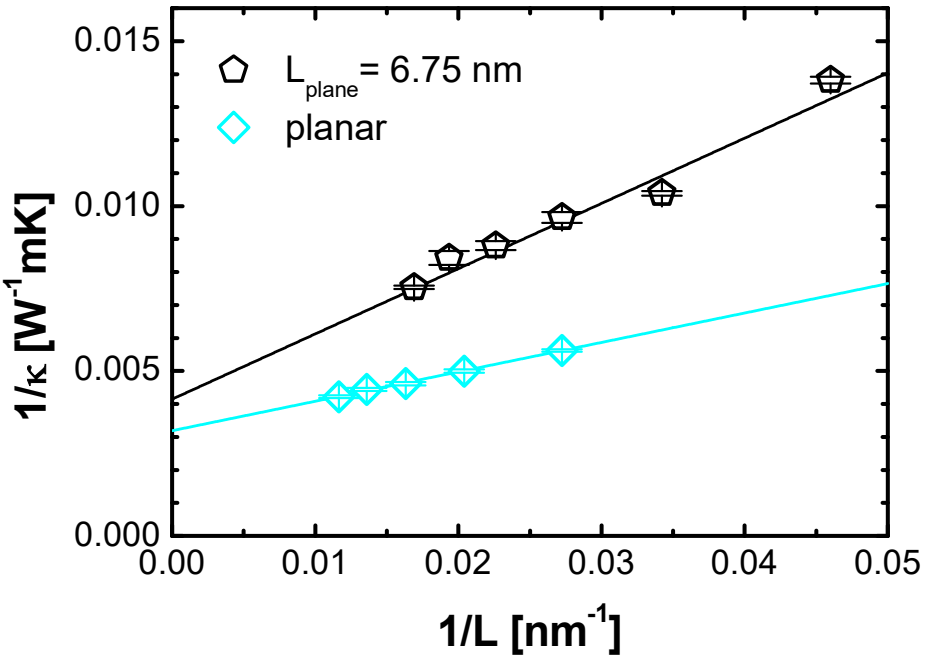

图 3-3　黑色空心五边形数据点表示具有相同平面长度 L_{plane}，不同折叠数目的折叠石墨烯热导率，从右向左 n 取值为 2 ~ 7，层间距为 0.35 nm。黑色线为数据点的线性拟合，表示了折叠石墨烯热导率的倒数与总长度 L 的倒数的关系。蓝绿色菱形空心点表示不同长度的完美石墨烯的热导率的倒数，蓝绿色线为数据点的线性拟合，表示二维结构的完美石墨烯热导率的倒数与长度的倒数关系。

由图 3-3 可以看到，折叠石墨烯的热导率的倒数与总长度 L 成正比。注意到在这里，L 的增大的同时，折叠的数目 n 也在增大。L 越大时，热导率越大。这与上文提到的 NEMD 模拟中常见的尺寸效应吻合，此时 l_{sys}=l。折叠石墨烯的热导率比对应长度的二维结构的完美石墨烯要低。当 L 趋于无穷大时，完美石墨烯热导率为 313 W/mK，这与实验得到的具有基底的石墨烯热导率基本一致[3]；折叠石墨烯热导率为 241 W/mK。平面长度为 6.75 nm 的无穷多折叠、无穷长的折叠石墨烯热导率比完美石墨烯降低了约 24 %。且当尺寸越小时，降低的越明显，从拟合后的直线来看，最多能降低 45 %（L = 21.7 nm 时）。

3.3 不同层间距离的折叠石墨烯的热导率

折叠石墨烯的热导率的大小对于层数的依赖关系如图 3-4 所示。图 3-4(a)是 0.35 nm 层间距时，折叠石墨烯的热导率与层数的关系。不同形状的数据点表示不同的总长 L 的折叠石墨烯的热导率。四条参考线自下而上依次是长度为 36.7 nm，49.0 nm，63.3 nm，76.6nm 的完美石墨烯的热导率。对于某一个长度的石墨烯，热导低于完美石墨烯的热导率，与上文结论一致，且折叠数目越多，热导率越低。右边是层间距为 0.30 nm 时，热导率的下降相对于 0.35 nm 的情况不明显。因此对于具有相同总长度的石墨烯和折叠石墨烯，$\kappa_{planar} > \kappa_{folded,\ 0.30nm} > \kappa_{folded,\ 0.35nm}$。这与[10]中热导率随着层间压缩增大而减小的结论是不同的。折叠石墨烯的尺寸比折叠石墨烯纳米带要大很多，出现这种差别可能是层间作用的差别：大面积的折叠石

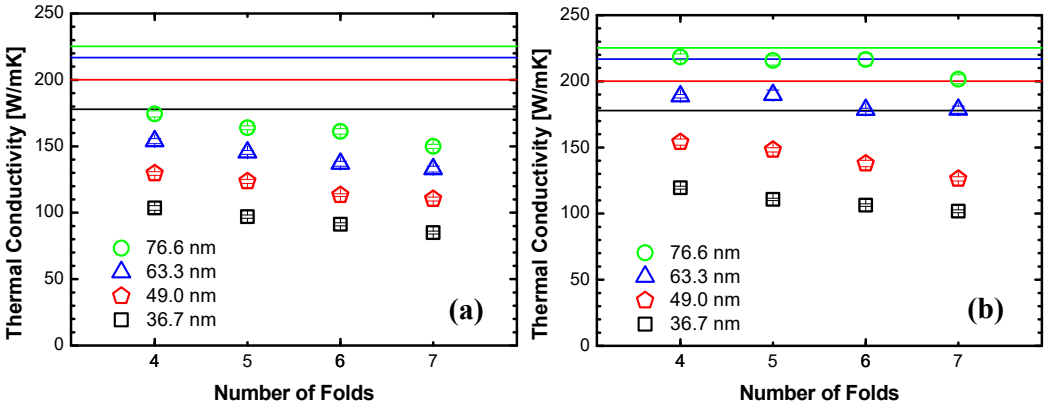

图 3-4　层间距离为 (a) 0.35 nm，(b) 0.30 nm 的折叠石墨烯的热导率与折叠的数目的关系，作为参考的水平线是相同总长 L 的平面结构的完美石墨烯的热导率。

墨烯中的层间范德瓦尔斯力作用虽然弱，但是由于面积较大，依然可以产生较大的层间热流，有利于热的输运。下面将具体对层间热流进行讨论，证明这种假设。

3.4 层间力对热导率的影响：从层间热流分析

折叠石墨烯热导率随着层间力增大而增大的结论与纳米带中的结论恰好相反，是因为层间作用对于热输运的贡献变得不可忽视。图 3-5 表示长度为 36.7 nm 的折叠石墨烯中，总热流与范德瓦尔斯力传递的热流随着折叠的数目的变化。可以看到，折叠数目越多，系统中的总热流越小，这与上一节中减小的热导率趋势保持一致。层间热流随着折叠数目的增多，呈现大致减小的趋势（层间距为 0.35 nm 的 n=5 石墨烯比 n=4 和 n=6 的情况都要小，这可能是由于记录层间作用时只选取了一个虚拟截面，导致记录的热流与真正的层间热流有微小差异）。当层间距减小时，层间热流明显增加，直接导致总的热流增大。且层间热流占总热流的比例也明显增加，表示增强了的层间作用使得层间输运对热导率的贡献增加。热流增大意味着有更多的热输运通道存在，即声子发生的散射更少。下面从功率谱密度来分析层间作用的影响。

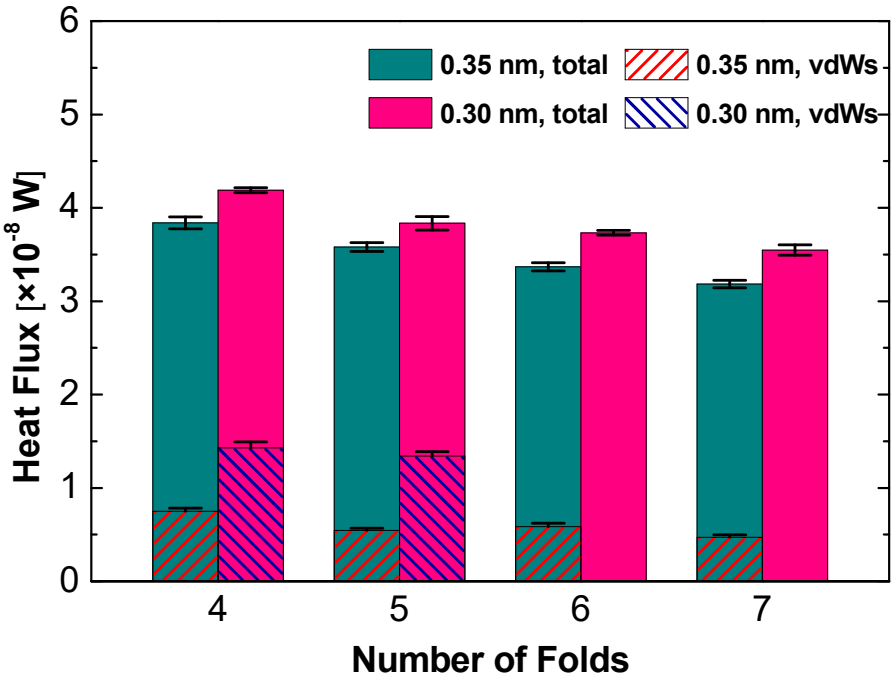

图 3-5　系统内总热流（total）和层间范德瓦尔斯力传递的热流（vdWs）随着层间距离的变化（0.35 nm，0.30 nm），以及随着折叠数目的变化

3.5 层间力对热导率的影响：从功率谱分析

声子的功率谱密度也叫振动功率谱，它表示单位频率的声子输运的功率。如果对于某个频率ν，声子的功率谱密度较大，那么该频率对应的声子占据的态的数目更多[36]，如果声子功率谱密度为零，则表示系统中不存在该频率的声子。图 3-6 是折叠石墨烯中不同位置的原子的功率谱密度，图 3-6 (a)、(b)分别是是压缩距离为 0.35 nm 和 0.3 nm 的折叠石墨烯平面部分和弯曲部分的功率谱密度。折叠石墨烯的总长为，36.7 nm，折叠数目为 4，平面部分为自下向上第二个平面层中的 60 个原子的数据求平均得到，弯曲部分由自下向上第二个弯折处的 60 个原子得到。

当层间距离为 0.35 nm 时，平面部分和弯曲部分的功率谱密度有明显的不同。弯曲部分的低频部分更加平缓，主要分布在 0～20 THz 的频率范围内，平面部分的低频部分更加陡峭，主要分布于 0～10 THz 的频率范围内，而 10～20 THz 对应的功率谱密度则极小，这个范围内弯曲部分的功率谱密度的局部最高值。声子从经过平面/弯曲交界处时，必须通过散射改变自身频率才能通过。由于声子满足 Bose-Einstein 分布，低频声子对于热输运的贡献更大，因此低频部分的差异对于热阻的增加有着主要贡献。高频部分，尽管平面部分和弯曲部分的功率谱形状上相近，但是尖峰所处的位置有所不同：弯曲部分的尖峰相对于平面部分朝着低频方向移动（红移），这也会导致部分声子发生散射，降低了热导率。因此，具有弯折结构的石墨烯与完美石墨烯相比，因为在弯折/平面交界处存在声子输运的失配，热导率更低，这也解释了图 3-1 中弯曲部分更陡峭的温度分布。

当层间距离为 0.30 nm 时，平面部分的功率谱密度曲线与弯曲部分的相比，尖峰所处位置一致，形状大致一致。注意到弯曲部分的功率谱尖峰高度低于平面部分，说明平面部分的声子色散更弱。在 7～10 THz 的频率范围内，平面部分的功率谱密度更小，这保证两条曲线的面积积分是一致的。层间距为 0.30 nm 的情况比起 0.35 nm 的情况，声子的失配情况更少，系统具有更少的散射，因此热导率稍高。但是，折叠的存在使得散射必然存在，因此热导率依然比完美热导率低。

层间距从 0.35 nm 减小至 0.30 nm，平面部分的变化主要出现在低频 0～10 THz 的两个尖峰被压缩变得瘦高，全峰半宽高因此增大，其他位置变化不大。这说明低频部分具有差别，平面部分的热导率也是不同的。弯曲部分的变化则较为

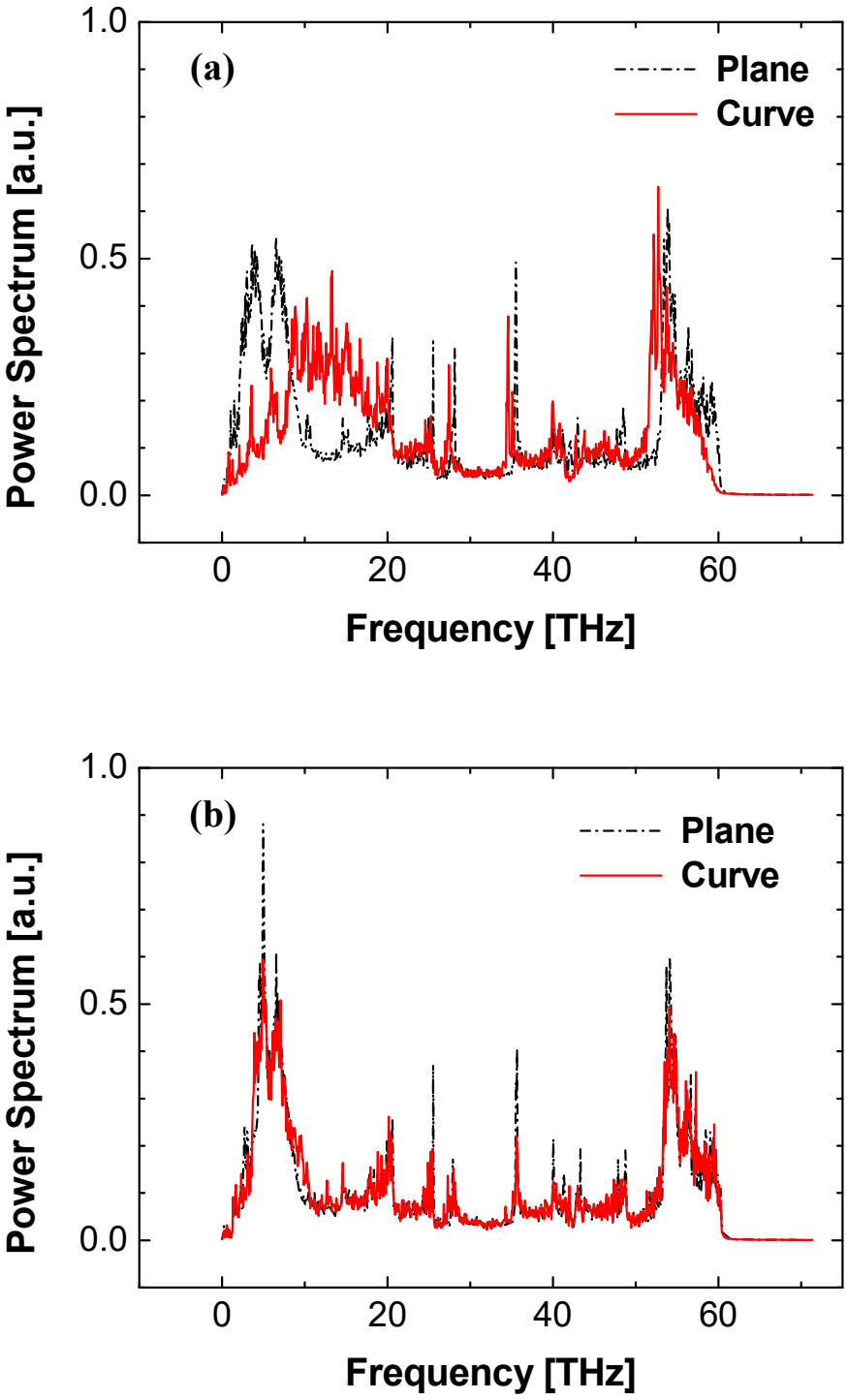

图 3-6　(a) 压缩层间距至 0.35 nm 的折叠石墨烯中，平面部分和弯曲部分的原子的功率谱密度。(b) 压缩层间距至 0.30 nm 的折叠石墨烯中，平面部分和弯曲部分的原子的功率谱密度。

明显。增强了的层间作用使得低频 0～20 THz 的尖峰被压缩至 0～10 THz 内，低频声子的分布更加集中在更低频的位置，且峰的形状也发生了明显改变，由许多分散的峰变为比较明显的、紧凑分布的两个尖峰。增强的层间作用则使高频部分则发生了蓝移，频率大于 34 THz 时对应的尖峰位置朝着高频移动。

综上，折叠的引入使得平面部分和折叠部分存在声子模式失配，声子散射增加而热导率降低，层间距从 0.35 nm 变为 0.30 nm，层间作用力增强，弯曲部分和平面部分声子失配减少，声子散射减少，折叠对热导率的降低幅度有所减小。因此相同的总长度下，$\kappa_{planar} > \kappa_{folded,\ 0.30nm} > \kappa_{folded,\ 0.35nm}$。这与纳米带中的热导率随层间距减小而增大的结论相反，是因为纳米带中层间作用可以忽略，热导率下降主要来自于折叠对层内输运的影响，折叠石墨烯中的层间作用的贡献更大，而且随着尺寸的增大，折叠对热导率的减小作用也有所削弱。

4 结论

4.1 总结与展望

利用 NEMD 模拟折叠石墨烯中的热输运，发现折叠石墨烯表现出了不同于完美石墨烯的性质。折叠的存在降低了热导率。这是因为弯曲部分和平面部分的声子行为不同，折叠/平面交界处将发生明显的散射。当折叠的数目越多时，一方面这种散射发生的更多，另一方面平面部分的长度变小，将限制具有长自由程声子的出现，这两方面都增加了系统的热阻，导致热导率下降。系统总长度越大，这种热导率的减小越来越不明显，这是因为体系变大后，允许的长自由程声子增多，折叠所影响到的声子的比例越来越小。

相邻平面层的通过范德瓦尔斯力的直接输运削弱了各自平面层内的温度梯度，使得平面层内出现接近水平的温度分布。具有不同层间距的折叠石墨烯的热导率不同，层间距越小，热导率越高。当层间距减小时，层间范德瓦尔斯作用增强，层间通过范德瓦尔斯力作用直接传递的热流越多，增强的层间作用提供了更多的热输运的通道。增加的层间力还使得弯曲部分与平面部分的声子行为差异更小，发生的散射减少。这些因素都导致了更高的热导率。

本文发现对于较大面积的折叠石墨烯，相对于完美石墨烯热导率下降并不多。要想用于热电应用，应该尽量使用小尺寸的折叠石墨烯，并创造尽可能多的折叠。层间距离应该避免较小，以减少热在层间直接输运。

4.2 可改进的方面

第一，本次模拟只计算了两组层间距不同的情况，要想得到更详细的依赖关系，应该计算其他层间距的情况。第二，模拟出现的效应被归结为长程声子受到影响，但是没有给出证明。依据文献[31]，可以通过从 MD 抽取出折叠石墨烯中声子自由程的信息，得到具有不同自由程的声子对热导率的贡献，证明这种解释。第三，只计算了一种平面部分长度的情况，其他平面部分长度的情况则不得而知。

致谢

首先感谢我的导师杨诺教授对我的毕业设计进行的指导。杨诺教授强调学术论文的严谨性，关注细节，同时教会我在学术演讲中讲述专业内容时要注重表达的简洁和清晰。杨诺教授也是我本科阶段二课的导师，他不仅在我的第一段研究经历中提供了无私的支持，保持了严苛的要求，更对我的重大人生选择上提供了中肯的建议。在此表示诚挚感谢！

感谢纳米传热实验室的所有成员，我们每周一次的例会、在实验室的热烈讨论都给予了我切实的提高和对于学术进一步的热爱。特别地，我要感谢钱鑫学长，他的探索精神启发了我走上学术之路，在我申请出国留学时帮助我开拓了视野。感谢金泽林硕士，他充满耐心地回答了我所有的疑问。感谢安盟博士，他与我的合作让我真切体会到团队协作的重要性。

感谢国家超级计算天津中心（National Supercomputer Center in Tianjin）和纳米传热实验室对本文使用的计算资源的支持。

感谢我的特优生导师向军教授，尽管我最终没有选择煤燃烧方面的研究，但他鼓励我找到了自己喜欢的研究方向。

感谢我的室友们，他们对我的作息习惯给予了充分的包容，对我生活给予了关怀。

最后感谢我的父母，他们严格的家教培养了我坚韧的性格，他们对我的包容，对我可能犯错时的提醒，是我成长过程中最宝贵的财富。

参考文献

[1] J. Jadhao and D. Thombare, *Review on exhaust gas heat recovery for IC engine*, International Journal of Engineering and Innovation Technology (IJEIT) Volume **2** (2013).

[2] D. G. Cahill, P. V. Braun, G. Chen, D. R. Clarke, S. Fan, et al., *Nanoscale thermal transport. II. 2003–2012*, Applied Physics Reviews **1** 011305 (2014).

[3] J. H. Seol, I. Jo, A. L. Moore, L. Lindsay, Z. H. Aitken, et al., *Two-dimensional phonon transport in supported graphene*, Science **328** 213 (2010).

[4] Z. Wang, R. Xie, C. Bui, D. Liu, X. Ni, B. Li, and J. Thong, *Thermal transport in suspended and supported few-layer graphene*, Nano Lett. **11** 113 (2011).

[5] S. Ghosh, I. Calizo, D. Teweldebrhan, E. P. Pokatilov, D. L. Nika, A. A. Balandin, W. Bao, F. Miao, and C. N. Lau, *Extremely high thermal conductivity of graphene: Prospects for thermal management applications in nanoelectronic circuits*, Appl. Phys. Lett. **92** 151911 (2008).

[6] L. E. F. Torres, S. Roche, and J.-C. Charlier, *Introduction to graphene-based nanomaterials: from electronic structure to quantum transport*, Cambridge University Press, 2014.

[7] X. Xu, Y. Wang, K. Zhang, X. Zhao, S. Bae, et al., *Phonon Transport in Suspended Single Layer Graphene*, arXiv:1012.2937 (2010).

[8] Z. Ni, Y. Wang, T. Yu, Y. You, and Z. Shen, *Reduction of Fermi velocity in folded graphene observed by resonance Raman spectroscopy*, Phys. Rev. B **77** (2008).

[9] J. Zang, S. Ryu, N. Pugno, Q. Wang, Q. Tu, M. J. Buehler, and X. Zhao, *Multifunctionality and control of the crumpling and unfolding of large-area graphene*, Nat. Mater. **12** 321 (2013).

[10] N. Yang, X. Ni, J.-W. Jiang, and B. Li, *How does folding modulate thermal conductivity of graphene?*, Appl. Phys. Lett. **100** 093107 (2012).

[11] H. Zhang, T. Zhou, G. Xie, J. Cao, and Z. Yang, *Thermal transport in folded zigzag and armchair graphene nanoribbons*, Appl. Phys. Lett. **104** 241908 (2014).

[12] W. Jang, Z. Chen, W. Bao, C. N. Lau, and C. Dames, *Thickness-dependent thermal conductivity of encased graphene and ultrathin graphite*, Nano Lett. **10** 3909 (2010).

[13] L. Lindsay, D. Broido, and N. Mingo, *Diameter dependence of carbon nanotube thermal conductivity and extension to the graphene limit*, Phys. Rev. B **82** 161402 (2010).

[14] L. J. Yi, Y. Y. Zhang, C. M. Wang, and T. C. Chang, *Temperature-induced unfolding of scrolled graphene and folded graphene*, J. Appl. Phys. **115** 204307 (2014).

[15] Y. Xie, Y. Chen, X. L. Wei, and J. Zhong, *Electron transport in folded graphene junctions*, Phys. Rev. B **86** 195426 (2012).

[16] R. Kubo, *The fluctuation-dissipation theorem*, Reports on Progress in Physics **29** 255 (1966).

[17] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, *A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters*, J. Chem. Phys. **76** 637 (1982).

[18] V. Tewary and B. Yang, *Singular behavior of the Debye-Waller factor of graphene*, Phys. Rev. B **79** 125416 (2009).

[19] Y. Wang, A. K. Vallabhaneni, B. Qiu, and X. Ruan, *Two-dimensional thermal transport in graphene: a review of numerical modeling studies*, Nanoscale and Microscale Thermophysical Engineering **18** 155 (2014).

[20] P. K. Schelling, S. R. Phillpot, and P. Keblinski, *Comparison of atomic-level simulation methods for computing thermal conductivity*, Phys. Rev. B **65** 144306 (2002).

[21] S. Nosé, *A unified formulation of the constant temperature molecular dynamics methods*, J. Chem. Phys. **81** 511 (1984).

[22] W. G. Hoover, *Canonical dynamics: Equilibrium phase-space distributions*, Phys. Rev. A **31** 1695 (1985).

[23] A. Dhar, *Heat transport in low-dimensional systems*, Adv. Phys. **57** 457 (2008).

[24] D. W. Ricker, *Echo signal processing*, Springer Science & Business Media, 2003.

[25] M. Kaviany, *Heat transfer physics*, Cambridge University Press, 2008.

[26] N. Yang, G. Zhang, and B. Li, *Carbon nanocone: A promising thermal rectifier*, Appl. Phys. Lett. **93** 243111 (2008).

[27] Y. Quo, N. Karasawa, and W. A. Goddard, *Prediction of fullerene packing in C60 and C70 crystals*, Nature **351** 464 (1991).

[28] R. E. Tuzun, D. W. Noid, B. G. Sumpter, and R. C. Merkle, *Dynamics of fluid flow inside carbon nanotubes*, Nanotechnology **7** 241 (1996).

[29] S. Lepri, *Thermal conduction in classical low-dimensional lattices*, Phys. Rep. **377** 1 (2003).

[30] O. Narayan and A. P. Young, *Continuum and lattice heat currents for oscillator chains*, Physical review. E, Statistical, nonlinear, and soft matter physics **80** 011107 (2009).

[31] K. Sääskilahti, J. Oksanen, S. Volz, and J. Tulkki, *Frequency-dependent phonon mean free path in carbon nanotubes from nonequilibrium molecular dynamics*, Phys. Rev. B **91** (2015).

[32] H. Casimir, *Note on the Conduction of Heat in Crystals*, Physica **5** 495 (1938).

[33] Y. F. Zhu, J. S. Lian, and Q. Jiang, *Re-examination of Casimir limit for phonon traveling in semiconductor nanostructures*, Appl. Phys. Lett. **92** 113101 (2008).

[34] E. Pop, V. Varshney, and A. K. Roy, *Thermal properties of graphene: Fundamentals and applications*, MRS Bulletin **37** 1273 (2012).

[35] Z. Rieder, J. Lebowitz, and E. Lieb, *Properties of a harmonic crystal in a stationary nonequilibrium state*, Journal of Mathematical Physics **8** 1073 (1967).

[36] N. Yang, T. Luo, K. Esfarjani, A. Henry, Z. Tian, J. Shiomi, Y. Chalopin, B. Li, and G. Chen, *Thermal Interface Conductance Between Aluminum and Silicon by Molecular Dynamics Simulations*, J. Comput. Theor. NanoS. **12** 168 (2015).

附录

S.1 热导率计算的处理细节

同等尺寸的完美石墨烯的热导率是计算折叠石墨烯热导率的重要参考。由于石墨烯的热导率计算具有明显的尺寸依赖性，因此必须对不同尺寸的石墨烯进行单独计算。图　给出了不同尺寸下的温度分布。可以看出，由于人为热浴的作用，处于边界的原子的温度分布存在明显的温度跳跃，导致了温度分布的非线性。为了减少这种作用的影响，计算温度梯度时将这部分原子去掉，利用剩余原子的温度数据进行拟合温度梯度。因为去掉的原子数目是不同的，故ΔT 的取值也是变化的。

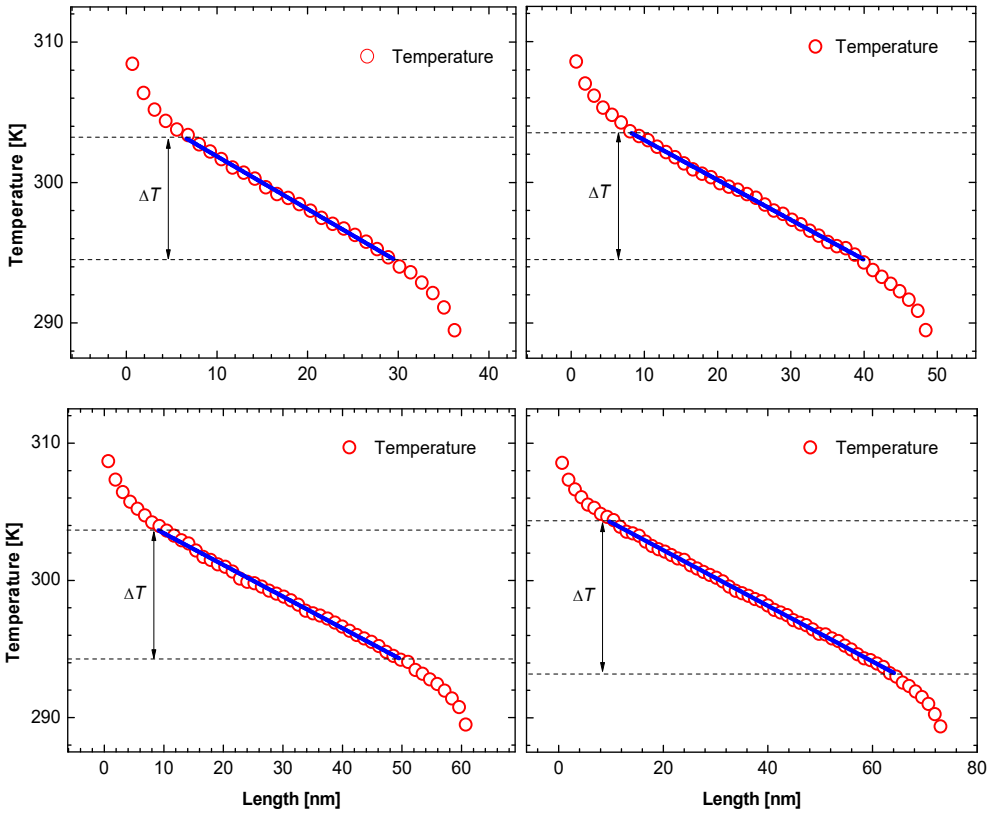

图 S-1　长度为 36.7nm，48.9nm，61.3nm 和 73.6nm 的沿着长度方向温度分布曲线及温度梯度的线性拟合。

S.2 声子态密度的计算

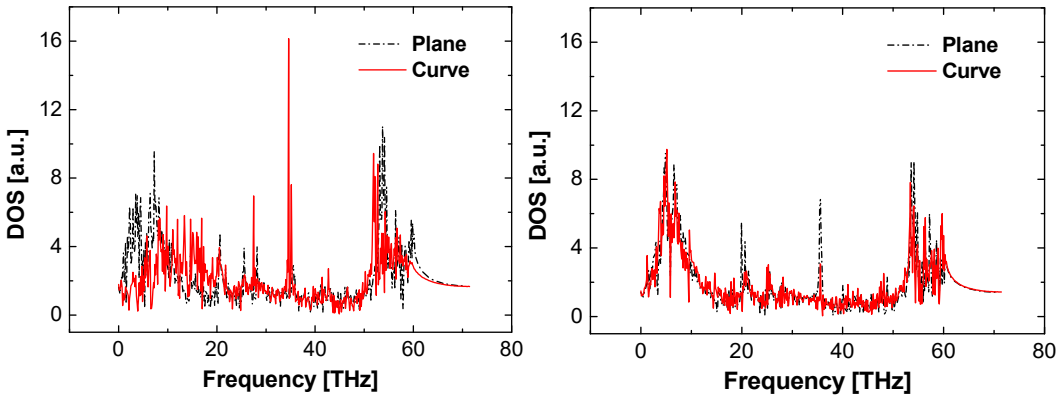

图 S-2　压缩层间距至 0.35 nm，0.30 nm 的折叠石墨烯中，平面部分和弯曲部分的原子的功率谱密度。

图 S-2 是平面部分和弯曲部分的声子态密度，所记录的原子位置与图 3-6 中功率谱统计的原子完全一致，结构参数也与图 3-6 所计算的结构弯曲一致。声子态密度通过对自相关函数的傅利叶变换得到。在高温近似下，声子态密度和功率谱密度只相差一个系数 k_BT。比较图 S-2 和图 3-6 也可以发现，声子态密度曲线和功率谱密度曲线形状大致一致。与功率谱密度分析得到的结论也是一致的。主要的区别是在层间距为 0.35 nm 的情况时，弯曲部分声子态密度在 34 THz 有明显的尖峰，且尖峰高度最高，而功率谱中的最高尖峰分布在其他位置，这可能是计算的时间不够长或者用于平均的数据少导致的误差。

S.3 程序源代码

用来建立折叠石墨烯结构的 Fortran90 程序：

```
! Folded Grephene! 包含位置 和0速度，0加速度
module Parameters
!     Contains parameters of Folded Graphene
      !INTEGER, parameter               :: Model_No   = 100 ! 100=homogeneous;1=graded 1/N 2/N ... N/N ;  0=half big -half
small
      REAL, PARAMETER :: PI=3.14159265358979323846264338327950288419             ! 2= graded mass every layer

     INTEGER,PARAMETER          :: Case_No=12 !MPIf90 no of cores for MPI !20140428qichen
     CHARACTER*70              :: SampOut_base='READ_pva1_T300_LAY000_00F0_FG' !BPB11 pos vel acc !20140428qichen
      CHARACTER*80              :: SampOut(Case_No)     !20140428qichen
       CHARACTER*80                     :: SampOut_4  = 'READ_Neib_LAY000_00F0_FG.dat'
      CHARACTER*70              :: SampOut_2_base = 'READ_3D_LAY000_00F0_FG'   !20140428qichen
      character*80                 :: SampOut_2(Case_No)                          !20140428qichen
       character*80                     :: SampOut_3  = 'READ_2D_LAY000_00F0_FG.pdb'
       !character*80                    :: SampOut_1  = 'READ_tem_LAY000_00F0_FG.dat'
      CHARACTER*80                     :: SampOut_5  = 'READ_Npl_LAY000_00F0_FG.dat'
      CHARACTER*80                     :: SampOut_6  = 'READ_Mlist_LAY000_00F0_FG.dat'
      CHARACTER*80                     :: SampOut_7  = 'READ_BClist_LAY000_00F0_FG.dat' ! 边界粒子的编号
    CHARACTER*80              :: SampOut_8_base  = 'READ_Van_Neib_LAY000_00F0_FG' !范德华力 的 邻居表 包含邻居编号，平移距
离，邻居数目!20140608qichen
```

```
    CHARACTER*80              :: SampOut_8(Case_No)!20140608qichen
    CHARACTER*80              :: SampOut_9_base  = 'READ_Sub_Grp_pairs_LAY000_00F0_FG' !20150302qichen
    CHARACTER*80              :: SampOut_9(Case_No) !20150302qichen

   INTEGER  :: rdseed(90)=(/127,131,137,139,149,151,157,163,167, 179,181,191,193,197,199,211,223,227,229, &
                                    233,239,241,251,257,263,269,271,277,281,
283,293,307,311,313,317,331,337,347,349, &
                                    353,359,367,373,379,383,389,397,401,409,
419,421,431,433,439,443,449,457,461,463, &
                                    467,479,487,491,499,503,509,521,523,541,
547,557,563,569,571,577,587,593,599,601, &
                                    607,613,617,619,631,641,643,647,653,659,173 /) !MPIf90
hu20130606!MPIf90 !20140428qichen

      integer, parameter                 :: NLAY    =300 ! No of layers, length of tube= Nlay*a*3^0.5/2; 10lay=1.26nm
      integer, parameter                 :: Ntube1=5       ! armchair 10 10
      integer, parameter                 :: Ntube2=5

      integer                                  :: NPL     =Ntube1+Ntube2 ! only right for armchair
      integer                                  :: N =NLAY*(Ntube1+Ntube2) ! NLAY*NPL
    integer,parameter                ::n_folded=6 !弯曲的个数
    double precision                 ::r_van_1,r_van_2 !20141006qichen
    INTEGER, DIMENSION(:,:), ALLOCATABLE :: around_shift
      INTEGER, DIMENSION(:), ALLOCATABLE   :: around_flag
      integer, dimension(:), allocatable   :: Npl_list    ! No of partical per layer
      integer, dimension(:), allocatable   :: Npl_mov! No of partical moved per layer= whcih without 4 neighbor
      integer, dimension(:), allocatable   :: BC_list
      integer                                          :: N_BC
      integer, parameter                :: DIM  = 3       ! Dimension
      double precision, parameter :: a    = 1.418E-10       ! C-C disk 1.458 SWNT 1.418 morse配套应该是1.418
      double precision            :: offset_gauss !=1.458E-11   ! 位置偏移最大值 0.0*a
      double precision, parameter :: T    = 2.*300.d0 !100.0d0       ! temperature
      double precision, parameter :: kb   = 1.38E-23
      double precision, parameter :: mass = 0.012E-23/6.023
      double precision, parameter  :: mass_ratio = 1
      DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: mass_list ! graded nework
      double precision            :: Aera     =Ntube1*3.*a*1.4E-10 ! 1.4*2PiR, only right for armchair

      INTEGER                                          :: Naround=3
    integer, dimension(:), allocatable  ::Van_num     !20140504qichen
    integer, dimension(:,:), allocatable ::Van_around  !20140504qichen
    integer, dimension(:,:), allocatable ::Neib_sec_near  !20141222qichen
    integer,dimension(:,:), allocatable  ::Van_around_shift !20140504qichen
    integer, dimension(:),   allocatable ::Sub_grp_pairs_num !20150302qichen
    integer, dimension(:,:), allocatable ::Sub_grp_pairs  !20150302qichen
    DOUBLE PRECISION,DIMENSION(:,:),allocatable   :: pos_sub_down   !20150302qichen
    DOUBLE PRECISION,DIMENSION(:,:),allocatable   :: pos_sub_up   !20150302qichen
    INTEGER ,parameter                  :: Van_top=600    !20140520qichen 邻居上限
    integer,dimension(:),allocatable     :: Mark                      !20140821qichen
    DOUBLE PRECISION, DIMENSION(:,:),allocatable     :: slp2            !20140821qichen
    DOUBLE PRECISION, DIMENSION(:),allocatable       :: slp1,slp_x         !20140821qichen
    integer,dimension(:),allocatable               ::Plane_No              !用来实时存储各个平面和弯曲的编
号      !20140821qichen
    integer, dimension(:), allocatable    ::Plane_flag  !20140504qichen
    INTEGER, DIMENSION(:,:), ALLOCATABLE:: around
    double precision                 ::lay_dist !distance between layers
    double precision                 ::plane_len !the length of the plane part
    double precision                           :: z_top        !20140517qichen the topest atom's z coordinate
    double precision                           :: z_low         !20140517qichen the lowest atom's coordinate
    INTEGER                   :: N_plane_up =0,N_plane_down=0 !20150302qichen  这里 N_plane_up/down 指的是动态变化的最上/
下 平面部分，即受基底作用力的部分

      logical                              :: VelAcc= .TRUE.      ! if vel & acc write to file?
      double precision, dimension(:,:), allocatable  :: pos     ! positions
      double precision, dimension(:,:), allocatable  :: vel     ! positions
      double precision, dimension(:,:), allocatable  :: acc     ! positions

      double precision, dimension(:,:),allocatable   :: Pos2d_big  ! POS FOR 2D GRAPHITE
      double precision, dimension(:,:),allocatable   :: temp  ! TEMPERATURE
end module Parameters

module SWPotential
      DOUBLE PRECISION, PARAMETER :: SW_A      =5.3732037
      DOUBLE PRECISION, PARAMETER :: SW_B      =0.50824571
      DOUBLE PRECISION, PARAMETER :: SW_p      =4.d0
      DOUBLE PRECISION, PARAMETER :: SW_pp1    =5.d0 !SW_p+1
      DOUBLE PRECISION, PARAMETER :: SW_q      =0.d0
      DOUBLE PRECISION, PARAMETER :: SW_qp1    =1.d0 !SW_q+1
```

```
      DOUBLE PRECISION, PARAMETER :: SW_sa     =1.8943619d0
      DOUBLE PRECISION, PARAMETER :: SW_lamda  =18.707929
      DOUBLE PRECISION, PARAMETER :: SW_gama   =1.2d0
      DOUBLE PRECISION, PARAMETER :: SW_costh  =-1./2.d0
      DOUBLE PRECISION, PARAMETER :: SW_sigm   =1.26329438E-10 ! 1.418E-10/2^(1/6)
      DOUBLE PRECISION, PARAMETER :: SW_epsl   =4.96/(6.24150636309E18) ! J/pair = 4.96 eV/pair
      CONTAINS
      FUNCTION fun_f2(r0)
            DOUBLE PRECISION :: fun_f2,r,r0
            r=r0/SW_sigm   ! 此处如果用r=r/sw 将会在整个程序里都改变r 即rik
                  IF (r < SW_sa) THEN
                        fun_f2 = -1.d0*SW_A*(  SW_B*SW_p/(r**SW_pp1)  + &
                                    ( SW_B/(r**SW_p) - 1.d0 )/((r-SW_sa)*(r-SW_sa)) ) * dexp(1.d0/(r-SW_sa))  !SW_B SW_p
      SW_q
                  ELSE
                        fun_f2 = 0.d0
                  END IF
            RETURN
      END FUNCTION fun_f2

      FUNCTION fun_f3(v_rij0,s_rij0,v_rkj0,s_rkj0,costh)
      USE Parameters
            DOUBLE PRECISION :: s_rij0,s_rkj0,s_rij,s_rkj,costh
            DOUBLE PRECISION,DIMENSION(DIM) :: fun_f3,v_rij0,v_rkj0,v_rij,v_rkj
            v_rij=v_rij0/SW_sigm
            s_rij=s_rij0/SW_sigm
            v_rkj=v_rkj0/SW_sigm
            s_rkj=s_rkj0/SW_sigm
            IF (s_rij < SW_sa .and. s_rkj< SW_sa) THEN
                  fun_f3(:) =
SW_lamda*fun_H1(s_rij,s_rkj)*( SW_gama*fun_H2(costh)*v_rij/((s_rij-SW_sa)*(s_rij-SW_sa)*s_rij) + &
                        2.d0*(costh-SW_costh)*( v_rij*costh/(s_rij*s_rij) - v_rkj/(s_rij*s_rkj) )  )
            ELSE
                  fun_f3 = 0.d0
            END IF
            RETURN
      END FUNCTION fun_f3

      FUNCTION fun_H1(rij,rkj)
            DOUBLE PRECISION :: fun_H1,rij,rkj
            fun_H1 = dexp( SW_gama/(rij-SW_sa) + SW_gama/(rkj-SW_sa) )
            RETURN
      END FUNCTION fun_H1

      FUNCTION fun_H2(costheta)
            DOUBLE PRECISION ::fun_H2, costheta
            fun_H2 = (costheta-SW_costh)*(costheta-SW_costh)
            RETURN
      END FUNCTION fun_H2

      FUNCTION fun_V2(r0)
            DOUBLE PRECISION :: fun_V2,r,r0
            r=r0/SW_sigm
                  IF (r < SW_sa) THEN
                        fun_V2 = SW_epsl*SW_A*(SW_B/(r**SW_p) - 1.d0  )*dexp(1.d0/(r-SW_sa))
                  ELSE
                        fun_V2 = 0.d0
                  END IF
            RETURN
      END FUNCTION fun_V2

      FUNCTION fun_V3(v_rij0,s_rij0,v_rkj0,s_rkj0,costh)
            USE Parameters
            DOUBLE PRECISION :: fun_V3,s_rij,s_rkj,s_rij0,s_rkj0,costh
            DOUBLE PRECISION,DIMENSION(DIM) :: v_rij,v_rkj,v_rij0,v_rkj0
            v_rij=v_rij0/SW_sigm
            s_rij=s_rij0/SW_sigm
            v_rkj=v_rkj0/SW_sigm
            s_rkj=s_rkj0/SW_sigm
            IF (s_rij < SW_sa .and. s_rkj< SW_sa) THEN
                  fun_V3 = SW_epsl*SW_lamda*fun_H1(s_rij,s_rkj)*fun_H2(costh)
            ELSE
                  fun_V3 = 0.d0
            END IF
            RETURN
      END FUNCTION fun_V3
end module SWPotential
```

```
!###########################################################################################
!########     MAIN PROGRAME     START      ##############################################
!###########################################################################################


program Positions

!USE DFLIB
use Parameters
use SWPotential
implicit none

      integer                 :: i,j,k,k1,i_gauss,i_lay,i_ver,i_inl,i_1,i_0,i_nei,i_big,i_case  ! CNCone !20140428qichen
      double precision :: theta0,theta1,theta2     ! CNCone
      Logical                 :: flag1, flag2
      double precision :: r0,r1,r2       ! radius of tube
      double precision, dimension(:), allocatable     :: v0       ! 温度为T的平均速度
      double precision, dimension(12) :: gauss ! guass 分布的时候用到
      real :: ag,ag_1

    Call    Name_files  !BPB11  !BPB15f90 补全文件名! 20140428qichen
      CALL  CREATE_2D_GRAPHITE      ! Part 1 计算一个2D Graphite平面
      CALL  Get_Npl_LIST       ! Part 3 Get Npl_list
    !CALL   PLANE_TO_TUBE           ! Part   将2D平面坐标转换成TUBE 3D坐标
      CALL    PLANE_TO_CURVE      !part
    CALL    NEIGHBOR_LIST          ! Part 2 计算生成  Neighbor list
    CALL    Get_Van_around_list !part 生成Van der Waals force neighbor list
    CALL    Initiating_substrate !part 生成基底的位置 ! 20150302qichen
    CALL    Get_Sub_grp_pairs   ! part to generate the relationship of subtrate atom and sample atom !20150302qichen
      CALL  Get_Npl_Move_LIST ! Part 4 Get Npl_mov
      CALL  Get_BC_LIST             ! Part 5 Get BClist 边界粒子的编号
      !CALL GAUSS_OFFSET_POSITION
      CALL  compute_masslist  ! Part 6 MASS LIST
      ! Part 8 计算生成 !加速度
      allocate( temp(DIM+1,N) )
      !do   i_big=1,100
            !CALL EVOLVE_SAMPLE          ! Part 10 优化结构
            !CALL compute_temperature     ! Part 11         20140428qichen
            !write(*,'(1x,"### AVE Tmp ###",E16.8 ,"### Max Tmp ###",E16.8)') SUM(temp(DIM+1,:))/N,
MAXVAL(temp(DIM+1,:))  !20140428qichen
           !if(  MAXVAL(temp(DIM+1,:)) < 1.d0) then
           do i_case=1,Case_No       !BPB15f90   !20140428qichen
              vel(:,:)=0.d0 !20140428qichen
              CALL     RANDOM_VELOSITIES(i_case)     ! Part 7 计算生成 和温度  !20140428qichen
              temp(:,:)=0.d0 !Nuo20130614
              CALL     compute_temperature     ! Part 11         20140428qichen
                  write(*,'(1x,"### AVE Tmp ###",E16.8 ,"### Max Tmp ###",E16.8)') SUM(temp(DIM+1,:))/N,
MAXVAL(temp(DIM+1,:))
             !  CALL    Initial_momentum(1,N)   ! BPB8_F90 动量归零         !20140428qichen
                   ! CALL   compute_temperature     ! Part 11
                    CALL    WRITE_OUT(i_case)            ! Part 9 ! Write out Data     !20140428qichen
              print '(a,I10)', '****** Success Case ', i_case    !20140428qichen
           enddo    !BPB15f90                                      !20140428qichen
           go to 911
            !endif
      !enddo


911 deallocate( around)
    deallocate( around_shift)
      deallocate( around_flag)
      deallocate( mass_list )
      !deallocate( Npl   )
      deallocate( pos )
      deallocate( vel )
      deallocate( acc )
      deallocate( Pos2D_big )
      deallocate(temp)
      deallocate( Npl_list )
      deallocate( Npl_mov )
      deallocate( BC_list )
    deallocate( Van_num )  !20140504qichen
    deallocate( Van_around )  !20140504qichen
    deallocate( Van_around_shift)  !20140504qichen
    deallocate( Sub_grp_pairs) !20150302qichen
    deallocate( Sub_grp_pairs_num) !20150302qichen
    deallocate( pos_sub_down)  !20150302qichen
    deallocate( pos_sub_up)    !20150302qichen
```

```
    deallocate( Neib_sec_near ) !20141222qichen
    deallocate( slp2 )            !20140821qichen
    deallocate( slp1 )                  !20140821qichen
    deallocate( slp_x )                 !20140821qichen
    deallocate(Plane_No)                    !20140821qichen
STOP
end program Positions
!#########################################################################################
!########     MAIN PROGRAME   END         ##########################################
!#########################################################################################
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!Name_files!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE Name_files !BPB11 !BPB15f90  !20140428qichen start
      use Parameters    !20140428qichen
      implicit none
      INTEGER:: i,j,i1,i2,icase
      CHARACTER*1 :: tmpcha1,tmpcha2

do icase=1,Case_No

      !ASII of '0' =48, '9'=57
      IF(icase>99) stop 'CHA'

      i1=icase/10
      i2=mod(icase,10)
      tmpcha1=CHAR(i1+48)
      tmpcha2=CHAR(i2+48)
      if(LEN_TRIM(SampOut_2_base)+6 > LEN(SampOut_2(icase)) ) stop 'Len'   !20140428qichen
      SampOut_2(icase)=Sampout_2_base(1:LEN_TRIM(SampOut_2_base))//'_'//tmpcha1//tmpcha2//'.pdb' !BPB11    !20140428qic
hen
      if(LEN_TRIM(SampOut_base)+6 > LEN(SampOut(icase)) ) stop 'Len'
      SampOut(icase)=SampOut_base(1:LEN_TRIM(SampOut_base))//'_'//tmpcha1//tmpcha2//'.dat' !BPB11  !20140608qichen start
    if(LEN_TRIM(SampOut_8_base)+6 > LEN(SampOut_8(icase)) ) stop 'Len'
      SampOut_8(icase)=SampOut_8_base(1:LEN_TRIM(SampOut_8_base))//'_'//tmpcha1//tmpcha2//'.dat' !BPB11 !20140608qichen
end
    if(LEN_TRIM(SampOut_9_base)+6 > LEN(SampOut_9(icase)) ) stop 'Len'
      SampOut_9(icase)=SampOut_9_base(1:LEN_TRIM(SampOut_9_base))//'_'//tmpcha1//tmpcha2//'.dat'
enddo
      RETURN
    END SUBROUTINE Name_files !BPB11            !20140428qichen end
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!Name_files!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 1 计算一个2D Graphite平面
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE CREATE_2D_GRAPHITE ! Part 1 计算一个2D Graphite平面
      !USE DFLIB
      use Parameters
      implicit none

      integer i,j,k
      double precision :: alfa     !  angle alfa of a in circle whose perimeter is (3a*Npl/2)
      double precision :: r       ! radius of tube
      double precision :: dist     ! distence between layer
      double precision, dimension(:), allocatable :: alfa1     ! POSTITIONS of odd layer
      double precision, dimension(:), allocatable :: alfa2     ! POSTITIONS of even layer
      !integer                :: k1,i_lay,i_ver,i_inl,i_1,i_0,i_nei  ! CNCone
      !double precision :: theta0,theta1,theta2     ! CNCone
      !NPL=2*NAM
      !N = NLAY*NPL
      dist=a*3.0**(0.5)/2.0

      !allocate( pos(DIM,N) )
      allocate( Pos2d_big(DIM,N) )  ! polar coordinates on 2D surface;
                                                ! Pos2d_big(N_big,1) instead of x, Pos2d_big(N_big,2) instead of y
      allocate( alfa1(Npl) )       ! 奇数层Y
      allocate( alfa2(Npl) )             ! 偶数层Y

      ! get the POSTITIONS of particle in 1st layer circle
      alfa1(1)=a/2.0
      alfa1(2)=3.0*a/2.0
      do i=3,Npl,2
            alfa1(i)=alfa1(i-1)+2.0*a
            alfa1(i+1)=alfa1(i)+a
      enddo
      ! get the POSTITIONS of particle in 2nd layer circle
```

```
      alfa2(1)=0.0
      alfa2(2)=2*a
      do i=3,Npl,2
            alfa2(i)=alfa2(i-1)+a
            alfa2(i+1)=alfa2(i)+2.0*a
      enddo
      ! get the POSTITIONS of particle in ALL layer circle
      do i=1,Npl
            Pos2d_big(1,i)=0.0       ! Get the coordinates of 1st layer
            Pos2d_big(2,i)=alfa1(i)

            Pos2d_big(3,i)=0.0
            Pos2d_big(1,(i+Npl))=0.0  ! Get the coordinates of 2nd layer
            Pos2d_big(2,(i+Npl))=alfa2(i)
            Pos2d_big(3,(i+Npl))=dist
      enddo
      ! get coordinates from 3rd layer to last layer
      do i=2,(Nlay-1),2 ! Layer's No. - 1
     if( i==(Nlay-1))  then
       do j=1,Npl
                  Pos2d_big(1,i*Npl+j)=Pos2d_big(1,j)          ! odd layers
                  Pos2d_big(2,i*Npl+j)=Pos2d_big(2,j)
                  Pos2d_big(3,i*Npl+j)=i*dist
                  !Pos2d_big(1,(i+1)*Npl+j)=Pos2d_big(1,j+Npl)      ! even layers
                  !Pos2d_big(2,(i+1)*Npl+j)=Pos2d_big(2,j+Npl)
                  !Pos2d_big(3,(i+1)*Npl+j)=(i+1)*dist
       enddo
      else
       do j=1,Npl
                  Pos2d_big(1,i*Npl+j)=Pos2d_big(1,j)          ! odd layers
                  Pos2d_big(2,i*Npl+j)=Pos2d_big(2,j)
                  Pos2d_big(3,i*Npl+j)=i*dist
                  Pos2d_big(1,(i+1)*Npl+j)=Pos2d_big(1,j+Npl)       ! even layers
                  Pos2d_big(2,(i+1)*Npl+j)=Pos2d_big(2,j+Npl)
                  Pos2d_big(3,(i+1)*Npl+j)=(i+1)*dist
       enddo
      endif
      enddo
      RETURN
END SUBROUTINE CREATE_2D_GRAPHITE ! Part 1 计算一个比较大的2D Graphite平面

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part   将2D平面坐标转换成台式3D坐标
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  PLANE_TO_TUBE          ! Part 3 将2D平面坐标转换成TUBE 3D坐标
      !USE DFLIB
      use Parameters
      implicit none

      integer                 :: i
      double precision :: r0       ! radius of tube
      double precision :: theta0    ! 位置对应的角度

      allocate( pos(DIM,N) )
      allocate( vel(DIM,N) )
      allocate( acc(DIM,N) )
      pos=0.0d0
      vel=0.0d0
      acc=0.0d0
      ! Pos2d_big(1,:)(2,:)变化  (3,:)不变
      r0 = NPL*a*(3.D0)/(4.d0*PI)  ! 切面圆的直径 3^(1/2)*dc_c*(M^2+MN+N^2)^(1/2) /2PI
      do i=1,N
            theta0 = Pos2d_big(2,i)/r0   ! 弧长Pos2d_big(2,i)求角度
            pos(3,i)=DCOS(theta0)*r0                !20140425 qichen 3 & 1 swap
            pos(1,i)=DSIN(theta0)*r0                  !20140425 qichen 3 & 1 swap
      enddo
      pos(2,:)=Pos2d_big(2,:)                         !20140425 qichen 3 & 1 swap
      RETURN
    END SUBROUTINE      PLANE_TO_TUBE          ! Part 3 将2D平面坐标转换成台式3D坐标
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!Part 得到每一层的原子数!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  Get_Npl_LIST                ! Part  计算生成  Npl_list
      !USE DFLIB
      use Parameters
      implicit none
      integer                 :: i,j
      double precision :: Lw,Hg
```

```
      allocate(Npl_list(Nlay))
    Npl_list(:)=0.d0
    Lw=-0.1*a-0.5*sqrt(3.d0)*a
    Hg=0.1*a-0.5*sqrt(3.d0)*a
      searchlay: do i=1,Nlay
       Lw=Lw+0.5*sqrt(3.d0)*a
       Hg=Hg+0.5*sqrt(3.d0)*a
       searchall:do j=1,N
          if( (Pos2d_big(3,j)>Lw) .and. (Pos2d_big(3,j)<Hg)) Npl_list(i)=Npl_list(i)+1
       enddo searchall
        enddo searchlay
END SUBROUTINE    Get_Npl_LIST                ! Part 计算生成  Npl_list
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!Part 将平面弯折!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  PLANE_TO_CURVE
    !use DFLIB
    use Parameters
    implicit none

    integer                       :: i,j,k
    integer,parameter              :: nR=6     !内接多边形的边的条数(每边长度按sqrt(3.d0)*a计算)，推荐nR为奇数的两倍，如6,
10,14,
    double precision               :: R       ! radius of CURVE
    double precision               :: theta1      ! 对应的角度
    double precision               :: dist
    INTEGER,dimension(:),allocatable :: icenter !记录弯曲中心的lay's no.
    integer                       :: bot                        !没有参与弯曲的、依然在水平面内的最后一个粒子行编号
    integer                       :: top                   !参与弯曲的、曲面内最后一个粒子行编号，行指的是i
     double precision :: R_tp1,R_tp2 ,dd , Coscos!20140903 qichen

 allocate(pos(DIM,N))
 allocate(vel(DIM,N))
 allocate(acc(DIM,N))
 allocate(icenter(20))      !    至多有20个褶
 allocate(plane_flag(N)) !20140504qichen !20140517qichen
   pos=0.0d0
   vel=0.0d0
   acc=0.0d0
   icenter=0  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!i有奇数偶数之分
   plane_flag=0 !20140504qichen  !20140517qichen

  dist=0.5*a*3.0**(0.5)
  R=  (3.0**(0.5))*a /(4.d0 *DSIN(PI/dble(2*nR)))  !20140903qichen !修正以后，侧视图是正多边形
  lay_dist=2.d0*R           !20140429qichen
  k=1                                                  !底层以及1st fold
  icenter(k)=(Nlay-1-n_folded*nR)/(n_folded+1)+nR/2+1 !Npl/(n_folded+1)-2  20140328                                !找到
中心点的层数,按No.1lay去选择
  bot=icenter(k)-nR/2                              !bot=(icenter(1)-nR/4)*Nlay
  top=icenter(k)+nR/2 !top=(icenter(1)+nR/4+1)*Nlay
  plane_len=Pos2d_big(3,Npl*bot)-Pos2d_big(3,1)           !20140429qichen 2D中3对应3D中1


      pos(2,:)=Pos2d_big(2,:) !20140903qichen

      do i=0,bot-1
          do j=1,Npl
            pos(3,i*Npl+j)=Pos2d_big(1,i*Npl+j)   !20140425 qichen 3 & 1 swap
            pos(1,i*Npl+j)=Pos2d_big(3,i*Npl+j)          !20140425 qichen 3 & 1 swap
            plane_flag(i*Npl+j)=1   !20140517qichen
         enddo
      enddo                                               !draw the atom which still remains in the plane(i=odd)

      do i=bot,top-1
          do j=1,Npl
             theta1= (Pos2d_big(3,i*Npl+j)-Pos2d_big(3,bot*Npl))*PI/(dble(nR)*dist)     ! 弧长Pos2d_big(2,i)求角度
             pos(3,i*Npl+j)= R-DCOS(theta1)*R+pos(3,bot*Npl)   !20140425 qichen 3 & 1 swap
             pos(1,i*Npl+j)=DSIN(theta1)*R+pos(1,bot*Npl)      !20140425 qichen 3 & 1 swap
             if (i==top-1)then   !20140923qichen
                 plane_flag(i*Npl+j)=2   !20140517qichen !20140923qichen
             else
                 plane_flag(i*Npl+j)=0  !20140923qichen
             endif
          enddo
      enddo                                               !draw the atom of the curve
```

```
        !20140903qichen
          R_tp1= sqrt(dot_product(pos(:,(i-2)*Npl)-pos(:,(i-1)*Npl),pos(:,(i-2)*Npl)-pos(:,(i-1)*Npl)))
          write(*,'(G20.10)') R_tp1 * 1E10
          R_tp2= sqrt(dot_product(pos(:,(i-2)*Npl)-pos(:,(i-3)*Npl),pos(:,(i-2)*Npl)-pos(:,(i-3)*Npl)))
             write(*,'(G20.10)') R_tp2 * 1E10
             dd=sqrt(dot_product(pos(:,(i-1)*Npl)-pos(:,(i-3)*Npl),pos(:,(i-1)*Npl)-pos(:,(i-3)*Npl)))
             write(*,'(G20.10)') dd * 1E10
              Coscos     = (R_tp1*R_tp1+ R_tp2*R_tp2 - dd*dd) / (2.0d0*R_tp1*R_tp2)
              write(*,'(G20.10)') 180* ACOS(Coscos) /PI
        !20140903qichen
        do i=top,Nlay-1
            do j=1,Npl
               pos(1,i*Npl+j)=pos(1,top*Npl)-dble(i+1-top)*dist     !20140425 qichen 3 & 1 swap
               pos(3,i*Npl+j)=pos(3,top*Npl)                 !20140425 qichen 3 & 1 swap
               plane_flag(i*Npl+j)=2   !20140517qichen
           enddo
        enddo                                              !draw the atom which form the upper plane


   do k=2,n_folded
      if(mod(k,2)==0)then
       icenter(k)=(Nlay-1-n_folded*nR)/(n_folded+1)+nR+icenter(k-1)        !Npl/(n_folded+1)+icenter(k-1)
20140328                                  !找到中心点的层数,按No.1lay去选择
       bot=icenter(k)-nR/2                                          !bot=(icenter(1)-nR/4)*Nlay
       top=icenter(k)+nR/2                                         !top=(icenter(1)+nR/4+1)*Nlay

         do i=bot,top-1
             do j=1,Npl
              theta1= (Pos2d_big(3,i*Npl+j)-Pos2d_big(3,bot*Npl)) *PI/(dble(nR)*dist)      ! 弧长Pos2d_big(2,i)求角度
              pos(3,i*Npl+j)= R-DCOS(theta1)*R+pos(3,bot*Npl)           !20140425 qichen 3 & 1 swap
              pos(1,i*Npl+j)=pos(1,bot*Npl)-DSIN(theta1)*R
              if (i==top-1)then   !20140923qichen
                  plane_flag(i*Npl+j)=k+1   !20140517qichen !20140923qichen
              else
                  plane_flag(i*Npl+j)=0  !20140923qichen
              endif
            enddo
         enddo                                                    !draw the atom of the curve
         do i=top,Nlay-1
             do j=1,Npl
              pos(1,i*Npl+j)=pos(1,top*Npl)+dble(i+1-top)*dist   !20140425 qichen 3 & 1 swap
              pos(3,i*Npl+j)=pos(3,top*Npl)       !20140425 qichen 3 & 1 swap
              plane_flag(i*Npl+j)=k+1!20140517qichen
           enddo
         enddo

      else

icenter(k)=(Nlay-1-n_folded*nR)/(n_folded+1)+nR+icenter(k-1)  !Npl/(n_folded+1)+icenter(k-1)
       !找到中心点的层数,按No.1lay去选择
       bot=icenter(k)-nR/2                                          !bot=(icenter(1)-nR/4)*Nlay
       top=icenter(k)+nR/2                                          !top=(icenter(1)+nR/4+1)*Nlay


         do i=bot,top-1
             do j=1,Npl
              theta1= (Pos2d_big(3,i*Npl+j)-Pos2d_big(3,bot*Npl))*PI/(dble(nR)*dist)   ! 弧长Pos2d_big(2,i)求角度
              pos(3,i*Npl+j)= R-DCOS(theta1)*R+pos(3,bot*Npl) !20140425 qichen 3 & 1 swap
              pos(1,i*Npl+j)=pos(1,bot*Npl)+DSIN(theta1)*R   !20140425 qichen 3 & 1 swap
              if (i==top-1)then   !20140923qichen
                  plane_flag(i*Npl+j)=k+1  !20140517qichen !20140923qichen
              else
                  plane_flag(i*Npl+j)=0  !20140923qichen
              endif
           enddo
         enddo                                                   !draw the atom of the curve
         do i=top,Nlay-1
             do j=1,Npl
              pos(1,i*Npl+j)=pos(1,top*Npl)-dble(i+1-top)*dist      !20140425 qichen 3 & 1 swap
              pos(3,i*Npl+j)=pos(3,top*Npl)                  !20140425 qichen 3 & 1 swap
              plane_flag(i*Npl+j)=k+1!20140517qichen
             enddo                     !20140425 qichen 3 & 1 swap
         enddo                    !20140425 qichen 3 & 1 swap
       endif                  !20140425 qichen 3 & 1 swap
    enddo                   !20140425 qichen 3 & 1 swap
```

```
RETURN
END SUBROUTINE    PLANE_TO_CURVE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 2 计算生成  Neighbor list
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  NEIGHBOR_LIST          ! Part 4 计算生成  Neighbor list
!USE DFLIB
use Parameters
implicit none
integer                 :: i_0,i_1,KL,KH,i,i_nei,i_x,i_y
double precision :: r0,r1,r2
double precision, dimension(DIM) :: pos_tmp
allocate(around_shift(Naround*DIM,N))
allocate(around_flag(N))
allocate(around(Naround,N))

around_shift=0
around_flag=0
around=0

r1=a*0.8d0
r2=a*1.2d0

cyc41:do i_1 = 1, N
   KL=1 !KL = MAX(1,i_1-Nlay-1 )    ! 搜索下限!20140524qichen
   KH= N!KH = MIN(N,i_1+Nlay+1 )    ! 上限 !20140524qichen
  i=i_1
  i_nei=0
    cyc43a:do i_0 =KL,KH
                    r0 = sqrt( dot_product(pos(:,i)-pos(:,i_0),pos(:,i)-pos(:,i_0)) )
                      if(  (r0 > r1) .and.  (r0 < r2) )then
                           i_nei=i_nei+1
                     if(i_nei > Naround)then
                                 write(*,'(1x,"### i_neighbor > Naround ###",I10)') i
                               stop
                             endif
                            around(i_nei ,i) = i_0
                endif
    enddo cyc43a
!平移距离是3*a*Nlay/4
      if(i_nei < Naround)then
       around_flag(i_1)=1
          !  if(i_1 <= NPL)then        !20140429qichen mistake start
                   !     KL=N-NPL+1
                   !     KH=N
                   !  elseif(i_1> N-NPL)then
                   !         KL=1
                   !         KH=NPL
             !     else
             !      continue
            !endif                    !20140429qichen mistake end
          KL=1 !KL = MAX(1,i_1-Nlay-1 )    ! 搜索下限  !20140429qichen !20140524qichen
          KH=N !KH = MIN(N,i_1+Nlay+1 )    ! 上限      !20140429qichen !20140524qichen
         cyc43b:do i_0= KL,KH
                do i_x=-1,1,2
                    if(mod(Npl,2)==0)then
                        pos_tmp(2)=pos(2,i_1)+dble(3*i_x*Npl/2)*a
                    else
                        pos_tmp(2)=pos(2,i_1)+dble(   i_x  *(      ( 3*(Npl-1) / 2+ 3  )     )     )*a
                    endif
                    pos_tmp(3)=pos(3,i_1)
                    pos_tmp(1)=pos(1,i_1)
                    r0 = sqrt( dot_product(pos_tmp(:)-pos(:,i_0), pos_tmp(:)-pos(:,i_0)   ) )
                    if(  (r0 > r1) .and.  (r0 < r2) )then
                                    i_nei=i_nei+1
                                    if(i_nei > Naround)then
                                          write(*,'(1x,"### i_neighbor > Naround ###",I10)') i_1
                                          stop
                                    endif
                                    around(i_nei ,i_1) = i_0
                                  if(mod(Npl,2)==0)then
                            around_shift((i_nei-1)*DIM+2,i_1)=-3*i_x*Npl/2 !20140425 a mistake
                        else
                            around_shift((i_nei-1)*DIM+2,i_1)=-1* i_x  *      ( 3*(Npl-1) / 2+ 3  )    !20140425 a mistake
                        endif ! i_1 和i_0 平移符号相反
```

```
                endif
              enddo
             enddo cyc43b
           endif


       if(i_nei < Naround )then
                write(*,'(1x,"### i_neighbor < Naround ## I =   ###",I10)') i_1
                !stop
           endif
enddo cyc41
    z_low=pos(3,1)   !20140517qichen
    z_top=pos(3,N)    !20140517qichen


  !  do i=1,Npl
  !     around_flag(i)=0               !20140425 qichen
  !  enddo
   ! do i=N-Npl+1,N
  !     around_flag(i)=0               !20140425 qichen
  !  enddo
  !cyc51:do i=1,N !20140504qichen start !搜索在平面内的原子
   !      if ((pos(1,i)>plane_len) .or. (pos(1,i)<0))then
  !          continue
  !       else
   !         plane_flag(i)=1
  !       endif
   !     end do cyc51    !20140504qichen end
    RETURN
    END SUBROUTINE      NEIGHBOR_LIST          ! Part 4 计算生成  Neighbor list
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part  计算生成  Npl_mov 每一层拥有4个neighour 的粒子个数
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

SUBROUTINE  Get_Npl_Move_LIST      ! Part  计算生成  Npl_mov
      !USE DFLIB
      use Parameters
      implicit none
      integer                 :: i,j,k,cont
      double precision :: r
      allocate(Npl_mov(Nlay))
      do i=1,Nlay
            cont=0
            k=sum( Npl_list(1:i) ) - Npl_list(i)
            do j=1,Npl_list(i)
                  if( around(Naround,k+j) /= 0)cont=cont+1
            enddo
            Npl_mov(i)=cont
      enddo
      RETURN
END SUBROUTINE    Get_Npl_Move_LIST      ! Part 4 计算生成  Neighbor list
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part  计算生成  BC_list 没有4个neighour 的粒子
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  Get_BC_LIST            ! Part  计算生成 BC_LIST
      !USE DFLIB
      use Parameters
      implicit none
      integer                 :: i,j,k,cont
      double precision :: r
      allocate(BC_LIST(N))
      cont=0
      do i=1,N
            if( around(Naround,i) == 0)then
                  cont=cont+1
                  BC_list(cont)=i
            endif
      enddo
      N_BC= cont
      RETURN
    END SUBROUTINE      Get_BC_LIST            ! Part 4 计算生成  Neighbor list


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 计算 Van De Waal force's list !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE Get_Van_around_list
    !USE DFLIB
```

```
    use Parameters
    implicit none
    integer           ::i=0,j=0,k=0,i_nei=0,i_nei_sec=0,mk=0,l1=0,l2=0,l3=0 !20140821qichen !20141222qichen
    double precision           :: N_temp  !20141019qichen
    double precision    ::r0 !,r1,r2 ! 20141006qichen
    double precision,dimension(DIM) ::pos_tmp
    allocate(Van_num(N))
    allocate(Van_around(Van_top,N))        !sample: (5,7,8,9 ...)  5 , 7,8,9 ...instead of the number of the atom's neighbor.
    allocate(Van_around_shift(Van_top,N)) !sample: (0,0,1,0,5,5.....) 15 stands for 15 neighbors 0,0,1 instead of the
coordinate of the first neighbor 0,5,5 the second,....
    allocate(Neib_sec_near(9,N))  !to record each atom's second nearest neib no. 20141222qichen
    allocate(slp2(Nlay-2,2))           !20140821qichen
    allocate(slp1(Nlay-1))                 !20140821qichen
    allocate(slp_x(Nlay-1))                !20140821qichen
    allocate(Mark( 2*n_folded))                !20140821qichen
    allocate(Plane_No(N))                  !20140821qichen
    Van_num(:)=0
    Van_around(:,:)=0
    Van_around_shift(:,:)=0
    Neib_sec_near=0
    pos_tmp(:)=0.d0
    slp_x=0.d0   !20140821qichen
    slp1=0.d0    !20140821qichen
    slp2=0.d0    !20140821qichen
    Mark= 0     !20141226qichen
    mk=1        !20140821qichen
    Plane_No=0  !20140821qichen
    r_van_1=3.0E-10    !关于r_van_1的探讨：次近邻r=a* 3^0.5= 2.449 故r_van_1>2.449 !20141006qichen
    r_van_2=8.875E-10  !20140520qichen    为了搜到更多邻居 !20141006qichen

    !qichen20140821 start
    cycv21: do l1=1,Nlay-1
       slp_x(l1) = (sum( pos(1,l1*Npl+1:(l1+1)*Npl))- sum( pos(1,(l1-1)*Npl+1:l1*Npl)) ) /dble( Npl )
       slp1(l1)= ( sum( pos(3,l1*Npl+1:(l1+1)*Npl))- sum( pos(3,(l1-1)*Npl+1:l1*Npl)) ) / (slp_x(l1)*dble(Npl))! 0.5 * a
* 3^0.5 = 1.228E-10 slp1 表示空间梯度

    enddo cycv21
    cycv22: do l2=1,Nlay-2
                slp2(l2,1)=  slp1(l2+1)-slp1(l2)
                IF ( (slp2(l2,1)<0.1d0) .and. (slp2(l2,1)> -0.1d0) ) then
                     slp2(l2,2)=1.d0 !  这里的0.3是用来鉴别空间梯度变化幅度的一个值，小于0.3代表是平面，大于0.3代表是弯曲部分
                else
                    slp2(l2,2)= 0.d0
                endif
    enddo cycv22
    cycv3: do l3=4,Nlay-5
              if
(((slp2(l3,2)+slp2(l3+1,2)+slp2(l3+2,2)+slp2(l3+3,2))==0).and.((slp2(l3-1,2)*slp2(l3-2,2)*slp2(l3-3,2)) >0 ))then
                Mark(mk)=l3
                mk=mk+1
              elseif
(((slp2(l3-3,2)+slp2(l3-2,2)+slp2(l3-1,2)+slp2(l3,2))==0).and.((slp2(l3+1,2)*slp2(l3+2,2)*slp2(l3+3,2)) >0 )) then
                Mark(mk)=l3+2
                mk=mk+1
              endif
    enddo cycv3
    cycv41: do l1=1,(Mark(1)+1)*Npl !20140903qihcen
              Plane_No(l1)=1
    enddo cycv41
    cycv42: do l1=1,n_folded -1
                do l2=Mark(2*l1-1)+1,Mark(2*l1)-3
                    !Van_num( l2*Npl+1: (l2+1)*Npl  )=0
                    Plane_No(l2*Npl+1: (l2+1)*Npl)=0
                enddo
                do l3=Mark(2*l1)-2,Mark(2*l1+1)
                    Plane_No( l3*Npl+1:(l3+1)*Npl)=l1+1
                enddo
    enddo cycv42
    cycv43: do l2= Npl*(Mark(2*n_folded-1)+1)+1,Npl*(Mark(2*n_folded)-2)
              !Van_num( l2*Npl+1: (l2+1)*Npl  )=0
             Plane_No(l2)=0
    enddo cycv43
    cycv44: do l1=Npl*( Mark(2*n_folded) -2)+1 ,N
              Plane_No(l1)= n_folded + 1
    enddo cycv44
    if ( sum( Plane_No(:)-Plane_flag(:))/=0 ) print*,' WARNING: Please check the code again for distinguish the curve and
the plane '    !20140428qichen
    !qichen20140821 end
```

```
    cycv1:do i=1,N
            !if(Plane_No(i)==0)goto 131 !20141226qichen
            i_nei=0
            cycv10: do  j=1,N  !20140830qichen
                        !if(j==i)goto 138
                        !if((Plane_No(j)==0).or.(abs(Plane_No(i)-Plane_No(j))/=1))goto 138 !20141226qichen
                            cycv101:do k=-1,1
                                    pos_tmp(2)=pos(2,i)+dble(3*k*Npl/2)*a  !20140830qichen
                                    pos_tmp(1)=pos(1,i)
                                    pos_tmp(3)=pos(3,i)
                                    r0 = sqrt( dot_product(pos_tmp(:)-pos(:,j),pos_tmp(:)-pos(:,j)) )
                                if((r0<R_van_2).and.(r0>R_van_1))
then !( abs(j-i)/Npl>10).and.() !20141006qichenthen   !20140830qichen .and.(abs(j/Npl-i/Npl)>3)
                                    i_nei=i_nei+1
                                    Van_around(i_nei,i)=j
                                    Van_around_shift(i_nei,i)= -1*3*k*Npl/2 !20140425 a mistake  !20141222qichen
                                    goto 138
                                endif
                            enddo cycv101
                    138 continue
             enddo cycv10      !20140830qichen
            Van_num(i)=i_nei      !20140830qichen
           ! 131 continue !20141226qichen
    enddo cycv1
    !20150302qichen start to get the amount of atoms on the top plane and the lowest plane
    N_plane_down = 0 !20150120qichen
    N_plane_up   = 0 !20150120qichen
    do i=1,N                                    !20140517qichen
      if(Plane_No(i)==1)then
          N_plane_down=N_plane_down+1   !20140517qichen
      endif
      if(Plane_No(i)==n_folded+1)then
          N_plane_up=N_plane_up+1   !20140517qichen
      endif
    enddo  !20140517qichen
      N_plane_up=N_plane_up+10*Npl     !20150302qichen
      N_plane_down=N_plane_down+10*Npl !20150302qichen
    !20150302qichen start

    RETURN
 END SUBROUTINE Get_Van_around_list

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 5 计算生成   高斯位置偏移 常数 offset_gauss 便宜最大值（暂时不用）
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  GAUSS_OFFSET_POSITION        ! Part 5 计算生成   高斯位置偏移 常数 offset_gauss 便宜最大值

!USE DFLIB
use Parameters
implicit none

integer                 :: i,j,i_gauss
double precision, dimension(12) :: gauss ! guass 分布的时候用到
real :: ag,ag_1


offset_gauss=0.*a
      do i=1+Npl,N-Npl  ! CNCone
            do j=1,3
                  do i_gauss=1,12
                    CALL RANDOM(ag)
                    gauss(i_gauss)=ag
                  enddo
                  pos(j,i)=pos(j,i)+(sum(gauss)/6-1)*offset_gauss
            enddo
      enddo
RETURN
END SUBROUTINE    GAUSS_OFFSET_POSITION        ! Part 5 计算生成   高斯位置偏移 常数 offset_gauss 便宜最大值
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!! Part 6 MASS LIST !!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE compute_masslist  ! graded nework ! Part 6 MASS LIST
use Parameters
INTEGER :: i,j,k

allocate( mass_list(N) )
mass_list=1.d0
```

```
RETURN
END SUBROUTINE compute_masslist   ! graded nework ! Part 6 MASS LIST


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 7 计算生成
! 产生随即速度（高斯分布）
! angle1: 产生随即角度（0，180）之间存入angle1(N)
!          要符合线形密度分布 正比于sin(angle1):实现方法 当（0，1）随机数 ag_1
!          属于[0，sin(angle1)] 则选择这个angle1;如果大于sin(angle1)则丢弃
! angle2: 产生随即角度（0，360）之间 存入angle2(N)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  RANDOM_VELOSITIES(i_case)          ! Part 7 计算生成   !20140428qichen
!USE DFLIB
use Parameters
implicit none

integer                 :: i,j,k,k1,i_gauss,i_lay,i_case  ! CNCone !20140428qichen
double precision, dimension(:), allocatable     :: v0       ! 温度为T的平均速度
double precision, dimension(12) :: gauss ! guass 分布的时候用到
real(4) :: ag,ag_1
double precision, dimension(:), allocatable          :: angle1
double precision, dimension(:), allocatable          :: angle2

allocate( angle1(N))
allocate( angle2(N))
allocate( v0(N))

v0(:) = sqrt(DIM*T*kb/(mass*mass_list(:)) )            ! DIM

CALL SEED(rdseed(i_case) )    !MPIf90     ! 为随机数作准备 !20140428qichen
!CALL SEED(13846 )                                      ! 为随机数作准备  !20140428qichen
!CALL RANDOM_SEED
!do k=1,Nlay-2
!     do j=1,Npl
!           i= j + k*Npl
do i=1+Npl,N-Npl  ! CNCone
!
100          CALL RANDOM(ag)        !!!!! 实现angle1的非均匀分布，线形分布
             angle1(i)= ag*180.0D0
             CALL RANDOM(ag_1)
             if( ag_1 >= dsind(angle1(i)) ) goto 100

             CALL RANDOM(ag)
             angle2(i)=ag*360.0D0

            !下面几行产生高斯分布的随机数
            ! (sum(gauss)-6)/6+1=(sum(gauss)/6) 是遵循gauss分布的随机数 属于(0,2)平均值1
            ! ( ((sum(gauss)/6)-1) *2/100+1 )* 实现 1+/-2%
             do i_gauss=1,12
                    CALL RANDOM(ag)
                    gauss(i_gauss)=ag
             enddo
             vel(1,i)=( ((sum(gauss)/6)-1) *2/100+1 )*v0(i)*dsind(angle1(i))*dcosd(angle2(i))

             do i_gauss=1,12
              CALL RANDOM(ag)
              gauss(i_gauss)=ag
             enddo
             vel(2,i)=( ((sum(gauss)/6)-1) *2/100+1 )*v0(i)*dsind(angle1(i))*dsind(angle2(i))

             do i_gauss=1,12
              CALL RANDOM(ag)
              gauss(i_gauss)=ag
             enddo
             vel(3,i)=( ((sum(gauss)/6)-1) *2/100+1 )*v0(i)*dcosd(angle1(i))
end do
deallocate( v0)
deallocate( angle1)
deallocate( angle2)

do i=1,N
      if(around(Naround,i)==0 ) vel(:,i)=0.d0
enddo

RETURN
END SUBROUTINE    RANDOM_VELOSITIES       ! Part 7 计算生成
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
! Part ! make momentum of mass centre = 0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE Initial_momentum(nstart,nend)       ! BPB3_F90 ! BPB8_F90   20140428qichen start
      use Parameters  !20140428qichen
      implicit none
      INTEGER :: i,j,k,lis, i1,i2,i3,nstart,nend
      DOUBLE PRECISION :: moment_tmp(DIM)

      moment_tmp(:)=0.d0
      do i=1,size(moment_tmp)
            moment_tmp(i)=sum(vel(i,nstart:nend)*mass_list(nstart:nend) )
      enddo
      moment_tmp(:)=moment_tmp(:)/dble(nend-nstart+1)

      do i=1,size(moment_tmp)
            do j=nstart,nend
                  vel(i,j)=vel(i,j)-moment_tmp(i)/mass_list(j)
            enddo
      enddo

      do i=1,size(moment_tmp)
            moment_tmp(i)=sum(vel(i,nstart:nend)*mass_list(nstart:nend))
      enddo
RETURN
END SUBROUTINE Initial_momentum          ! BPB3_F90 20140428qichen end

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 9 ! Write out Data
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  WRITE_OUT(i_case)            ! Part 9 ! Write out Data !20140428qichen
use Parameters
implicit none
integer                  :: i,i_case          !20140428qichen
character*80     :: char_format  !20150302qichen
! Write out position velosity acceleration
open(unit=2,file=Sampout(i_case),status='replace', &        !20140428qichen
     action='write',err=700)
      write(2,'(A1,L2,I6)',err=900) '%',VelAcc,N     !  The '%' signals to ignore this line to the BallRoom program producing an image from the simulation
      do i=1,N
              write(2,'(1X,3E23.15)',err=900) pos(:,i)
      enddo
      do i=1,N
              write(2,'(1X,3E23.15)',err=900) vel(:,i)
      enddo
      do i=1,N
              write(2,'(1X,3E23.15)',err=900) acc(:,i)
      enddo
close(unit=2)


! Write out the Van der Waals force neighbor list   !20140504qichen start
open(unit=2,file=Sampout_8(i_case),status='replace', &
     action='write',err=700)
    write(2,'(1X,I16,2E23.15)',err=900) Van_top,r_van_1,r_van_2!20141006qichen 把搜索半径也输出 !20140520qichen 输出允许最大的van der waals 力的距 “邻居”数
    write(char_format,*)2*Van_top+2
    char_format=adjustl(char_format)
    char_format="(1x,"//trim(char_format)//"I8)"
    do i=1,N
        write(2,char_format,err=900) Plane_No(i),Van_num(i),Van_around(:,i),Van_around_shift(:,i)   !I8前面的数应该等于2+Van_top+DIM*Van_top !20140821qichen
    enddo
    do i=1,N
        write(2,'(1X,9I8)',err=900) Neib_sec_near(:,i)   !20141222qichen for searching second nearest neighbors
    enddo
close(unit=2)          !20140504qichen start


! Write out Substrate sample neighbor relationship !20150302qichen
open(unit=2,file=sampout_9(i_case),status='replace',&
action='write',err=910)
    write(char_format,*)2*Van_top+1
    char_format=adjustl(char_format)
    char_format="(1x,"//trim(char_format)//"I8)"
    do i=1,N_plane_up+N_plane_down
              write(2,char_format,err=900) Sub_grp_pairs_num(i),Sub_grp_pairs(:,i)  !20140425  !20140504qichen
      enddo
```

```
close(unit=2)

if(i_case==1)then !BPB12f  !20140428qichen

open(unit=2,file=Sampout_3,status='replace', &
     action='write',err=700)
      write(2,'("ATOM")',err=900)
      do i=1,N
             !write(2,200,err=900) i, Pos_big(:,i)*1.0E10,1.00, &
             write(2,200,err=900) i, Pos2D_big(:,i)*1.0E10,1.00, &
                      temp(DIM+1,i)     *300. /maxval(temp(DIM+1,:) )
      enddo
close(unit=2)


! Write out Neighbor list
open(unit=2,file=Sampout_4,status='replace', &
     action='write',err=700)
      write(2,'(1X,I16)',err=900)N
      write(2,'(1X,I16)',err=900)NPL
      write(2,'(1X,I16)',err=900)Nlay
    write(2,'(1X,E23.15)',err=900)lay_dist!20140429qichen
    write(2,'(1X,E23.15)',err=900)plane_len!20140429qichen
    write(2,'(1X,I16)',err=900)N_folded  !20140517qichen
    write(2,'(1X,E23.15)',err=900)z_low !20140517qichen
    write(2,'(1X,E23.15)',err=900)z_top !20140517qichen
!     write(2,'(1X,E23.15)',err=900)Aera
      do i=1,N
             write(2,'(1X,14I8)',err=900)
around(:,i),around_shift(:,i),plane_flag(i),around_flag(i)  !20140425  !20140504qichen
      enddo
close(unit=2)

! Write out 2 PDB files（图型文件raswin.exe） Data
open(unit=2,file=Sampout_2(i_case),status='replace', &
     action='write',err=700)
      write(2,'("ATOM")',err=900)
      do i=1,N
             write(2,200,err=900) i, pos(:,i)*1.0E10, 1.00, &
                      temp(DIM+1,i)     !*300. !/maxval(temp(DIM+1,:) )
      enddo
close(unit=2)

! Write out Npl
open(unit=2,file=Sampout_5,status='replace', &
    action='write',err=700)
      !write(2,'(1X,I16)',err=900)Model_No
      write(2,'(1X,I16)',err=900)N
      write(2,'(1X,I16)',err=900)Nlay
      !write(2,'(1X,E23.15)',err=900)Area
      do i=1,Nlay
             write(2,'(1X,I16)',err=900) Npl_list(i)
      enddo
      do i=1,Nlay
             write(2,'(1X,I16)',err=900) Npl_mov(i)
      enddo
close(unit=2)
! Write out BC_list
open(unit=2,file=Sampout_7,status='replace', &
    action='write',err=700)
      !write(2,'(1X,I16)',err=900)Model_No
      write(2,'(1X,I16)',err=900)N
      write(2,'(1X,I16)',err=900)N_BC
      do i=1,N_BC
             write(2,'(1X,I16)',err=900) BC_list(i)
      enddo
close(unit=2)

! Write out Mass list
open(unit=2,file=Sampout_6,status='replace', &
     action='write',err=700)
!     write(2,'(1X,I16)',err=900)Model_No
!     write(2,'(1X,I16)',err=900)Miss_angle
      write(2,'(1X,I16)',err=900)Nlay
      do i=1,N
             write(2,'(1X,F16.10)',err=900) mass_list(i)
      enddo
close(unit=2)
```

```
endif   !20140428qichen
RETURN

200 FORMAT ('ATOM  ',I5,1X,' C  ',' ','NTI',1X,' ','   1',' ',3X,3F8.3,2F6.0,'      ','     ',' ',' ')    !20140425
700 continue
    print*,'Write_Sample: WARNING: cannot open ',SampOut(i_case),' for write'    !20140428qichen
      close(unit=2)
stop
900 continue
    print*,'Write_Sample: WARNING: error when writing ',SampOut(i_case)  !20140428qichen
    close(unit=2)
stop
910 continue
    print*,'Write_Sample: WARNING: error when writing ',SampOut_9(i_case)  !20140428qichen
    close(unit=2)
stop
END SUBROUTINE    WRITE_OUT
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 10 ! 优化结构
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine Evolve_Sample     ! Part 10 ! 优化结构
      use Parameters
      !use Simulation_Control
      !use Statistics
      implicit none
      INTEGER :: step, step_big, i_t, i, j,d,I_k ,SUM_L ,SUM_L2, N_SIDE,i_fft ,l1,l2
      DOUBLE PRECISION :: ene_kin_aver,ene_pot_aver,ene_tot_aver,temperature, sum_a,&
                                  SUM_TL,SUM_T
      !DOUBLE PRECISION, DIMENSION(:,:),ALLOCATABLE:: zita_minus, zita_plus
      INTEGER ::             Nsteps= 2**2 !12
      DOUBLE PRECISION :: deltat= 4.E-16
      DOUBLE PRECISION :: rou  != 1.E14   !阻尼系数
      DOUBLE PRECISION, DIMENSION(DIM,N):: fri_for

      rou= 0.D0 !1.E13
      fri_for=0.d0
            time: do step=1,Nsteps
                  do i=1,DIM
                        fri_for(i,:)= -1.d0*vel(i,1:N)*rou
                  enddo
               pos(:,1:N ) = pos(:,1:N) + deltat*vel(:,1:N) &
                         + 0.5d0*(deltat*deltat)*( acc(:,1:N) + fri_for(:,1:N) )
               vel(:,1:N) = vel(:,1:N) + 0.5d0*deltat*( acc(:,1:N) + fri_for(:,1:N) )
               acc=0.d0

               call Compute_Forces
               vel(:,1:N) = vel(:,1:N) + 0.5d0*deltat*( acc(:,1:N) + fri_for(:,1:N) )
            enddo time
      RETURN
end subroutine Evolve_Sample
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine Compute_Forces
      use Parameters
      use SWPotential
      !use Statistics
      implicit none
      INTEGER :: i,j1,j01,j02
      DOUBLE PRECISION,DIMENSION(DIM)::    F2_all                ! store 2-doby-force
      DOUBLE PRECISION,DIMENSION(Naround):: For_sca        ! Force scalar    ! store 2-doby-force
      DOUBLE PRECISION,DIMENSION(Naround,DIM):: For_vec    ! Force vector    ! store 2-doby-force
      DOUBLE PRECISION,DIMENSION(DIM,Naround):: derta_i
      DOUBLE PRECISION,DIMENSION(DIM,3):: For_ang
      INTEGER,DIMENSION(Naround) ::  k_i
      DOUBLE PRECISION,DIMENSION(Naround) :: Rik          ! store 2-doby-force
      DOUBLE PRECISION,DIMENSION(DIM,Naround,Naround):: derta_kk
      DOUBLE PRECISION,DIMENSION(Naround,Naround) :: cosalfa,Rkk
      For_sca   = 0.d0        ! store 2-doby-force
      For_vec   = 0.d0        ! store 2-doby-force
      derta_kk  = 0.d0
      cosalfa       = 0.d0
      Rkk           = 0.d0
      !$omp  parallel do &
      !$omp  default(shared) &
      !$omp  private(i,j1,j01,j02) &
      !$omp  private(F2_all,Rkk,cosalfa) &
      !$omp  private(For_sca,For_vec,derta_i,For_ang,k_i,Rik,derta_kk)
      cyc2: do i = 1, N      ! cyc2 calculate particles except boundary layer particles
```

```
      if(around(Naround,i) /= 0)then ! fixBC
            ! 初始化变量
            cyc20: do  j1=1,Naround  ! store 2-doby-force
                  k_i(j1) =  around(j1,i)
                  if( k_i(j1)==0)then     ! EMPTY NEIGHBOR
                        derta_i(:,j1) = 1.d0 ! 设得很大 使得2体力截至函数=0
                        Rik(j1) = 1.d0
                  else
                        derta_i(:,j1) = pos(:,i)-pos(:,k_i(j1))
                        Rik(j1) = sqrt(dot_product(derta_i(:,j1) ,derta_i(:,j1) ))
                  endif
            enddo cyc20
            cyc21: do j01=1,Naround-1   ! create half triangle Matrix: derta_kk,Rkk,Cosalfa
                        do j02=j01+1,Naround
                              if(k_i(j01)==0 .or. k_i(j02)==0)then           ! boundary particle
                                    derta_kk(:,j01,j02)     =0.d0
                                    Rkk(j01,j02)            =0.d0
                                    Cosalfa(j01,j02)  =0.d0
                                    derta_kk(:,j02,j01)     =0.d0
                                    Rkk(j02,j01)            =0.d0
                                    Cosalfa(j02,j01)  =0.d0
                              else
                                    derta_kk(:,j01,j02)     =pos(:,k_i(j01))-pos(:,k_i(j02))
                                    Rkk(j01,j02)            =
sqrt(dot_product(derta_kk(:,j01,j02) ,derta_kk(:,j01,j02) ))
                                    Cosalfa(j01,j02) = (Rik(j01)*Rik(j01) + Rik(j02)*Rik(j02) - Rkk(j01,j02)*Rkk(j01,j02))
/ (2.0d0*Rik(j01)*Rik(j02))

                                    derta_kk(:,j02,j01) = derta_kk(:,j01,j02)
                                    Rkk(j02,j01)            = Rkk(j01,j02)
                                    Cosalfa(j02,j01)  = Cosalfa(j01,j02)
                              endif
                        enddo
            enddo cyc21

            !SW start
            ! 2-doby-force
            cyc22: do j1=1,Naround        ! store 2-doby-force
                  For_sca(j1)   = -1.d0*SW_epsl*fun_f2(Rik(j1))/SW_sigm
                  For_vec(j1,:) = For_sca(j1)*derta_i(:,j1) /Rik(j1)
            enddo cyc22                   ! store 2-doby-force
            F2_all = sum(For_vec,dim=1)         ! store 2-doby-force
            Acc(:,i) = Acc(:,i) + F2_all / (mass*mass_list(i))
            ! langevin
      !     flux1(i) = flux1(i) + 0.5d0*( dot_product(derta_i(:,1),fluxdir   )           *dot_product( vel(:,i),
      For_vec(1,:))  &
      !                                           + dot_product(derta_i(:,2),fluxdir )
      *dot_product( vel(:,i), For_vec(2,:) )  &
      !                                           !+ dot_product(derta_i(:,3),fluxdir)
      *dot_product( vel(:,i), For_vec(3,:) )  &
      !                                           + dot_product(derta_i(:,Naround),fluxdir)
*dot_product( vel(:,i),For_vec(Naround,:) ) )      ! store 2-doby-force


            ! 3-doby-force
            cyc23:  do j01=1,Naround-1
                  do j02=j01+1,Naround
                        !NotBC: if( (Rik(j01) < L_u) .and. (Rik(j02)<L_u ) )then  ! 边界粒子的空健 不被考虑
                        NotBC: if( k_i(j01)/=0 .and. k_i(j02)/=0  )then  ! 边界粒子的空健 不被考虑
                              !SW start
                              For_ang(:,1) =
-1.d0*SW_epsl*fun_f3(derta_i(:,j01),Rik(j01),derta_i(:,j02),Rik(j02),Cosalfa(j01,j02)) /SW_sigm
                              For_ang(:,2) =
-1.d0*SW_epsl*fun_f3(derta_i(:,j02),Rik(j02),derta_i(:,j01),Rik(j01),Cosalfa(j01,j02)) /SW_sigm
                              For_ang(:,3) = - For_ang(:,1)  - For_ang(:,2)
                              !SW end
                              Acc(:,k_i(j01)) = Acc(:,k_i(j01))+For_ang(:,1) / (mass*mass_list(k_i(j01)))
                              Acc(:,k_i(j02)) = Acc(:,k_i(j02))+For_ang(:,2) / (mass*mass_list(k_i(j02)))
                              Acc(:,i) = Acc(:,i)       +For_ang(:,3) / (mass*mass_list(i))
            ! langevin
            !                 flux2(k_i(j01))=flux2(k_i(j01)) -
dot_product(derta_i(:,j01),fluxdir)*dot_product(vel(:,k_i(j01)),For_ang(:,1))
            !                 flux2(k_i(j02))=flux2(k_i(j02)) -
dot_product(derta_i(:,j02),fluxdir)*dot_product(vel(:,k_i(j02)),For_ang(:,2))
                        endif NotBC
                  enddo
            enddo cyc23
      endif ! fixBC
```

```
      end do cyc2
      !$omp  end parallel do
      return
end subroutine Compute_Forces


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 11 温度纪录
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE  compute_temperature     ! Part 11
      !USE DFLIB
      use Parameters
      implicit none
      integer                  :: i,j,k,k1,i_gauss,i_lay  ! CNCone
      !! 动能 温度
      temp =0.d0
            j=DIM+1
         do i= 1, DIM
             temp(i,:) = temp(i,:)+ vel(i,:) * vel(i,:)
             temp(j,:) = temp(j,:)+ vel(i,:) * vel(i,:)
         end do
            temp = temp* mass/2.0d0
            do j=1,DIM
                  temp(j,:) = 2.0d0 * temp(j,:)*mass_list(:) /kb
            enddo
            temp(DIM+1,:) = 2.0d0 * temp(DIM+1,:)*mass_list(:) /(kb * DIM)
      RETURN
END SUBROUTINE    compute_temperature           ! Part 11
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE    Get_Sub_grp_pairs !Part 基底原子邻居列表 !20150302qichen
    !USE DFLIB
    use Parameters
    implicit none
    integer             ::i=0,j=0,k=0,i_nei=0
    double precision    ::r0
    double precision,dimension(DIM) ::pos_tmp
    allocate(Sub_grp_pairs(2*Van_top,N_plane_up+N_plane_down)) !20150302qichen
    allocate(Sub_grp_pairs_num(N_plane_up+N_plane_down)) !20150302qichen
    Sub_grp_pairs_num(:)=0
    Sub_grp_pairs(:,:)=0
    pos_tmp=0.d0
    do i=1,N_plane_down
        i_nei=0
        pos_tmp(1)=pos_sub_down(1,i)
        pos_tmp(3)=pos_sub_down(3,i)
        do j=1,N
            do k=-1,1
                pos_tmp(2)=pos_sub_down(2,i)+dble(3*k*Npl/2)*a  !20140830qichen
                r0 = sqrt( dot_product(pos_tmp(:)-pos(:,j),pos_tmp(:)-pos(:,j)) )
                if((r0 < r_van_2).and.(r0>r_van_1)) then
                    i_nei=i_nei+1
                    Sub_grp_pairs(i_nei,i)=j
                    Sub_grp_pairs(i_nei+Van_top,i)= -3*k*Npl/2
                endif
            enddo
        enddo
        Sub_grp_pairs_num(i)=i_nei
    enddo
    do i=N_plane_down+1,N_plane_down+N_plane_up
        i_nei=0
        pos_tmp(1)=pos_sub_up(1,i-N_plane_down)
        pos_tmp(3)=pos_sub_up(3,i-N_plane_down)
        do j=1,N
            do k=-1,1
                pos_tmp(2)=pos_sub_up(2,i-N_plane_down)+dble(3*k*Npl/2)*a  !20140830qichen
                r0 = sqrt( dot_product(pos_tmp(:)-pos(:,j),pos_tmp(:)-pos(:,j)) )
                if((r0 < r_van_2).and.(r0>r_van_1)) then
                    i_nei=i_nei+1
                    Sub_grp_pairs(i_nei,i)=j
                    Sub_grp_pairs(i_nei+Van_top,i)= -3*k*Npl/2
                endif
            enddo
        enddo
        Sub_grp_pairs_num(i)=i_nei
    enddo


END SUBROUTINE Get_Sub_grp_pairs !20150302qichen
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Part 生成基底原子坐标
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE Initiating_substrate !20150120qichen start
  !USE DFLIB
  use Parameters
  implicit none

  integer ::i=0,j=0,ntemp=0
  double precision,dimension(2,N_plane_up)::abtemp
  DOUBLE PRECISION, PARAMETER :: Sigma    =  3.3407d-10
  DOUBLE PRECISION, PARAMETER :: Z_balance = Sigma*(2.d0)**(1.d0/6.d0)     !20140517qichen!20140622qichen equilibrium
distant [m]

  allocate(pos_sub_down(DIM,2*N_plane_down))
  allocate(pos_sub_up(DIM,2*N_plane_up))

  pos_sub_down=0.d0
  pos_sub_up  =0.d0

  pos_sub_down(3,:) = z_low-Z_balance
  pos_sub_up(3,:)  = z_top+Z_balance

  do i=0,1
      do j=1,Npl_list(1)
          pos_sub_down(1:2,i*Npl_list(1)+j)=pos(1:2,i*Npl_list(1)+j)
          pos_sub_up  (1:2,i*Npl_list(1)+j)=pos(1:2,N-(i*Npl_list(1)+j)+1)
      enddo
  enddo
  ntemp=(size(pos_sub_down,dim=2))/Npl_list(1)-1
  do i=2,ntemp
      do j=1,Npl_list(1)
          pos_sub_down(2,i*Npl_list(1)+j)=pos_sub_down(2,(i-2)*Npl_list(1)+j)

pos_sub_down(1,i*Npl_list(1)+j)=pos_sub_down(1,(i-2)*Npl_list(1)+j)+2.0*(pos_sub_down(1,Npl_list(1)+1)-pos_sub_down(1,
1))
      enddo
  enddo
  ntemp=(size(pos_sub_up,dim=2))/Npl_list(1)-1
  do i=2,ntemp
      do j=1,Npl_list(1)
          pos_sub_up(2,i*Npl_list(1)+j)=pos_sub_up(2,(i-2)*Npl_list(1)+j)

pos_sub_up(1,i*Npl_list(1)+j)=pos_sub_up(1,(i-2)*Npl_list(1)+j)+2.0*(pos_sub_up(1,Npl_list(1)+1)-pos_sub_up(1,1))
      enddo
  enddo
  abtemp=pos_sub_up(1:2,1:N_plane_up)- pos(:,N:N-N_plane_up+1:-1)
  abtemp=pos_sub_down(1:2,1:N_plane_down)- pos(:,1:N_plane_down)

RETURN
909 continue
      print*,'## FATAL, POSSIBLE CAUSES:1.errors of N_plane_up/down readin 2. errors of pos readin '
      stop
      RETURN

END !20150120qichen end
```