**电 子 科 技 大 学**

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

**学士学位论文**

**BACHELOR THESIS**



论文题目

专 业

学 号

作者姓名

指导教师

摘 要

自从2004年石墨烯被发现以来，探寻其他的新型二维晶体材料一直是二维材料研究领域的前沿。但是人们发现二维材料的过程很大程度上具有偶然性和盲目性，人们目前还没有比较系统的发现新的可行二维材料的有效方法。而本篇论文的目的就是提出一种新的\*\*二维材料的自动生成方法\*\*，为人们寻找新的二维材料提供一种试探性的思路。我们的思路是在给定三维材料的基础上，沿着某一个晶面进行解理，最后通过适当的调整，来计算最有可能存在的结构。我们使用此方法计算了MoS2, C等经典的二维材料结构，经过验证结果的准确性，说明我们的方法是可行的。实现结果见后面章节。

关键词：`半导体` `二维材料` `软件` `自动化`

第一章 著 论

1.1 研究工作的背景及意义

新的材料可以改变世界，这也是我们谈论铜器时代和铁器时代的原因。具体来说，不锈钢和硅片使现代世界成为可能。但是现在，一种新类型的只包含一层原子层的材料渐渐的展现出强大的发展潜力。这种我们一般称之为的二维材料，在过去的一些年中迅速发展，同时，更多理论上可能存在的二维材料也慢慢被人们发现。

以新的二维半导体材料及器件为代表的后摩尔时代的典型材料和新器件结构，能够解决后摩尔时代超大规模集成电路发展中的核心问题，实现器件尺寸缩小，获得诸如速度高、芯片小、成本低、截止频率高等先进集成电路，尤其是5nm技术节点之后的集成电路，实现信息的更快加速、高效地传输处理。

而在这个过程中，发现更多的可行二维材料无疑会给人们对于对于新材料的发现提供巨大的帮助。

对于已知的材料，人们可以拓展其生成二维材料，而对于未知的材料，我们可以通过自动生成方法进行一定程度的分析，进一步掌握未知材料的更多参数，所以我认为我的毕设题目是很有意义，并且具有一定的实际价值的。

1.2 二维材料自动生成方法国内外进展及现状

自 2004 年石墨烯被发现以来，探寻其他新型二维晶体材料一直是二维材料研究领域的前沿。正如石墨烯一样，大尺寸高质量的其他二维晶体不仅对于探索二维极限下新的物理现象和性能非常重要，而且在电子、光电子等领域具有诸多新奇的应用。

近年来，除石墨烯外，二维六方氮化硼、过渡族金属硫化物、氧化物、黑磷等二维材料也被制备出来，极大地拓展了二维材料的性能和应用。过渡族金属碳化物是一类庞大的材料家族，它结合了陶瓷和金属的特性，一方面具有很高的强度和硬度，以及高熔点、高温下优异的稳定性和抗腐蚀性，良好的抗热震性和低的化学反应活性；另一方面，它们具有优异的催化活性，在诸多化学反应中可与常用的贵金属催化剂相媲美。此外，很多过渡族金属碳化物，如 Mo2C、W2C、WC、TaC 及 NbC 等，都具有超导特性。因此，过渡族金属碳化物在电子、催化、储能、极端条件下使用的工具等领域有着广泛的应用。

1.3 本论文主要贡献及创新

该方法使用比较先进的软件集成技术，将传统的材料与自动化测试技术相结合，充分运用计算机的强大计算能力来发现新的材料。同时简化传统的计算流程，使人力从大量繁复的计算中解脱出来，将更多的精力放在整体的设计，架构的改进上，进而提高生产力，增加科研工作者的劳动效率。

同时该软件使用FORTRAN、PYTHON等脚本语言编写，可以同时在各个平台，如Windows、Linux，甚至编译成Android手机上运行的文件，极大的方便了科研工作者的科研工作，充分利用身边的计算资源，为科研工作者的科研工作铺平了道路。

1.4 本论文结构安排

文章结构的安排如下：

第一章：介绍了关于二维材料的相关背景，以及人们现在关于二维材料的研究现状。同时大体的关于本题目进行了解释。

第二章：首先概述了关于二维材料相关的基本概念与理论，接着提出了使用统计学的相关算法进行材料分析，并接着给出了具体的方法。

第三章：介绍了实验所需的环境，以及需要的环境的搭建。随后介绍了环境的配置，同时安装我们需要的一些软件，最后下载我们的源码进行相关实验。

第四章：本章节具体阐述了核心的实验过程。首先阐述怎样从三维晶体中提取信息，以及信息的筛选过程，最后进行相关信息的搜索与计算，得出最后结果。

第五章：对全文做出总结，并对进一步的工作做出展望。

第六章：致谢，感谢各方面的大力帮助

第二章 理论基础

2.1 晶体理论基础

考虑正格矢(a1, a2, a3)以及其对应的倒格矢(b1, b2, b3)，由晶体学知识可以知道他们的关系：

$$b\_i = 2 \pi \frac{a\_j x a\_k}{a\_i \* (a\_j x a\_k)}$$

倒格矢与正空间的晶面一一对应，而gm = m1b1 + m2b2 + m3b3 所确定的晶面间距为：

$$hehe$$

2.2 统计学习方法

2.2.1 解决问题

由于晶体的周期性质，我们寻找到的晶面也必然呈现周期的结构，如下图。而我们往往希望获取其中的一层原子来进行能量计算，因为我们这里创造性的运用统计学中的聚类方法，来从大量的原子层中分理处单个层的原子。

2.2.2 K-Means

K-Means聚类算法是最为经典的，同时也是使用最为广泛的一种基于划分的聚类算法，它属于基于距离的聚类算法。所谓的基于距离的聚类算法指采用距离作为相似性度量的评价指标，也就是说，当两个对象离的近，二者之间的距离比较小，那么它们之间的相似性就比较大。这类算法通常是由距离比较相近的对象组成簇，把得到紧凑而且独立的簇作为最终目标，因此，将这类算法称为基于距离的聚类算法。

K-Means聚类算法就是其中比较经典的一种算法。K-Means聚类是数据挖掘的重要分支，同时也是实际应用中最常用的聚类算法之一。本章重点是对K-Means聚类算法中的初始中心随机选取进行了分析与研究，给出了K-Means算法的思想和原理，优缺点的介绍以及现有的关于初始聚类中心选取的改进措施。

1967年，J.B,MacQueen提出的K-Means算法是目前为止在工业和科学应用中一种极有影响的聚类技术。K-Means聚类算法是一种常用的基于划分的聚类分析方法，该聚类算法的最终目标就是根据输入参数k(这里的k表示需要将数据对象聚成几簇)，然后把数据对象分成k个簇。该算法的基本思想：首先指定需要划分的簇的个数k值;然后随机地选择k个初始数据对象点作为初始的聚类中心;第三，计算其余的各个数据对象到这k个初始聚类中心的距离(这里一般采用距离作为相似性度量)，把数据对象划归到距离它最近的那个中心所处在的簇类中;最后，调整新类并且重新计算出新类的中心，如果两次计算出来的聚类中心未曾发生任何的变化，那么就可以说明数据对象的调整己经结束，也就是说聚类采用的准则函数是收敛的，表示算法结束(这里采用的是误差平方和的准则函数)。

K-Means聚类算法属于一种动态聚类算法，也称作逐步聚类法，该算法的一个比较显著的特点就是迭代过程，每次都要考察对每个样本数据的分类正确与否，如果不正确，就要进行调整。当调整完全部的数据对象之后，再来修改中心，最后进入下一次迭代的过程中。若在一个迭代中，所有的数据对象都己经被正确的分类，那么就不会有调整，聚类中心也不会改变，聚类准则函数也表明已经收敛，那么该算法就成功结束。 传统的K-Means算法的基本工作过程:首先随机选择k个数据作为初始中心，计算各个数据到所选出来的各个中心的距离，将数据对象指派到最近的簇中;然后计算每个簇的均值，循环往复执行，直到满足聚类准则函数收敛为止。通常采用的是平方误差准则函数，这个准则函数试图使生成的k个结果簇尽可能的紧凑和独立。其具体的工作步骤如下:

```

输入: 初始数据集DATA和簇的数目k

输出:k个簇，满足平方误差准则函数收敛

1)任意选择k个数据对象作为初始聚类中心

2)Repeat

3)根据簇中对象的平均值，将每个对象赋给最类似的簇

4)更新簇的平均值，即计算每个对象簇中对象的平均值

5)计算聚类准则函数E

6)Until准则函数E值不再进行变化

```

2.2.3 k\_means++

k-means算法是一种基本的聚类算法，这个算法的先决条件是

1）必须选择最终结果需要聚为几类，就是k的大小。

2）初始化聚类中心点，也就是seeds。

当然，我们可以在输入的数据集中随机的选择k个点作为seeds，但是随机选择初始seeds可能会造成聚类的结果和数据的实际分布相差很大。既然选择初始的seeds这么重要，那有什么算法可以帮助选择初始的seeds吗？当然有，k-means++就是选择初始seeds的一种算法。

k-means++算法选择初始seeds的基本思想就是：初始的聚类中心之间的相互距离要尽可能的远。

wiki上对该算法的描述是如下:

从输入的数据点集合中随机选择一个点作为第一个聚类中心

对于数据集中的每一个点x，计算它与最近聚类中心(指已选择的聚类中心)的距离D(x)

选择一个新的数据点作为新的聚类中心，选择的原则是：D(x)较大的点，被选取作为聚类中心的概率较大

重复2和3直到k个聚类中心被选出来

利用这k个初始的聚类中心来运行标准的k-means算法

从上面的算法描述上可以看到，算法的关键是第3步，如何将D(x)反映到点被选择的概率上，一种算法如下(详见此地)：

先从我们的数据库随机挑个随机点当“种子点”

对于每个点，我们都计算其和最近的一个“种子点”的距离D(x)并保存在一个数组里，然后把这些距离加起来得到Sum(D(x))。

然后，再取一个随机值，用权重的方式来取计算下一个“种子点”。这个算法的实现是，先取一个能落在Sum(D(x))中的随机值Random，然后用Random -= D(x)，直到其<=0，此时的点就是下一个“种子点”。

重复2和3直到k个聚类中心被选出来

利用这k个初始的聚类中心来运行标准的k-means算法

可以看到算法的第三步选取新中心的方法，这样就能保证距离D(x)较大的点，会被选出来作为聚类中心了。

2.2.4 尝试其他聚类算法

Affinity Propagation(AP)算法

跟其他聚类算法的不同之处是，AP在开始时，将所有节点都看成潜在的聚类中心，然后通过节点之间的通信，去找出最合适的聚类中心，并将其他节点划分到这些中心下去，所以我们可以认为，AP算法所要做的事情就是去发现这些聚类中心。

AP的输入是一个节点间的相似度矩阵，S，其中S(i,j)表示节点i和节点j之间的相似度，也表明了，j作为i的聚类中心的合适程度，这个相似度的计算可以根据具体应用场景，这里未免误导不作相似度的假设。其中S(k,k)表示节点k作为k的聚类中心的合适程度，可以理解为，节点k成为聚类中心合适度，在最开始时，这个值是初始化的时候使用者给定的值，会影响到最后聚类的数量。

AP中节点间传递的消息为两类：吸引度和归属度。

首先，吸引度是节点i传递向节点k的信息，传达了节点k对节点i的吸引度，记为r(i,k)，那么如何来衡量这个吸引度，其实吸引度是一个相对的概念，先前我们有相似度矩阵记录了k成为i的聚类中心的合适程度，那么这里我们只需要证明k比其他节点更合适了就可以了，那么其他节点是否合适这个如何进行衡量呢，是否合适其实就是看这两个节点是否相互认可，对于其他节点k'我们有s(i,k')表示节点k'作为节点i的聚类中心的合适度，那么再定义一个a(i,k')表示i对节点k'的认可程度（归属度），这两个值相加，a(i,k') + s(i,k')，就可以计算出节点k'作为节点i的聚类中心的合适程度了，这里，在所有其他节点k'中，找出最大的a(i,k') + s(i,k')，即max{a(i,k’)+s(i,k')}，再使用s(i,k) - max{a(i,k’)+s(i,k')} 就可以得出k对i的吸引度了，也就是第一个公式：

r(i,k) = s(i,k) - max{a(i,k’)+s(i,k')} 其中k != k'

接下来计算上面提到的归属度a(i,k)，表示了节点i选择节点k作为它的聚类中心的合适程度，这里要考虑到的一个思想是：如果节点k作为其他节点i'的聚类中心的合适度很大，那么节点k作为节点i的聚类中心的合适度也可能会较大，由此就可以先计算节点k对其他节点的吸引度，r(i',k)，然后做一个累加和表示节点k对其他节点的吸引度：

∑max{0，r(i',k)}

ps.这里在r(i',k)跟0之间取一个大的原因是因为s(i',k)一般会初始化成负值，导致r(i',k)计算出来也有可能是负值，这样的好处是，最后可以方便找出合适的聚类中心在完成所有计算后。

然后再加上r(k,k)，这里为什么要加上r(k,k)，根据吸引度公式，我们可以看出，其实r(k,k)，反应的是节点k有多不适合被划分到其他聚类中心下去，这里的公式中，将k有多适合成为其他节点的聚类中心：∑max{0，r(i',k)}加上它有多不适合被划分到其他聚类中心下去：r(k,k) 就有了计算公式：

a(i,k)=min{0，r(k,k)+∑max{0，r(i',k)}}

为了不让这个值过大，影响整体结果，将这个值控制在0以下。

其中a(k,k)的定义稍微有些不一样，只用∑max{0，r(i',k)}就可以了，主要反映k作为聚类中心的能力。

DBSCAN

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。

DBSCAN中的几个定义：

```

Ε邻域：给定对象半径为Ε内的区域称为该对象的Ε邻域；

核心对象：如果给定对象Ε领域内的样本点数大于等于MinPts，则称该对象为核心对象；

直接密度可达：对于样本集合D，如果样本点q在p的Ε领域内，并且p为核心对象，那么对象q从对象p直接密度可达。

密度可达：对于样本集合D，给定一串样本点p1,p2….pn，p= p1,q= pn,假如对象pi从pi-1直接密度可达，那么对象q从对象p密度可达。

密度相连：存在样本集合D中的一点o，如果对象o到对象p和对象q都是密度可达的，那么p和q密度相联。

可以发现，密度可达是直接密度可达的传递闭包，并且这种关系是非对称的。密度相连是对称关系。DBSCAN目的是找到密度相连对象的最大集合。

Eg: 假设半径Ε=3，MinPts=3，点p的E领域中有点{m,p,p1,p2,o}, 点m的E领域中有点{m,q,p,m1,m2},点q的E领域中有点{q,m},点o的E领域中有点{o,p,s},点s的E领域中有点{o,s,s1}.

那么核心对象有p,m,o,s(q不是核心对象，因为它对应的E领域中点数量等于2，小于MinPts=3)；

点m从点p直接密度可达，因为m在p的E领域内，并且p为核心对象；

点q从点p密度可达，因为点q从点m直接密度可达，并且点m从点p直接密度可达；

点q到点s密度相连，因为点q从点p密度可达，并且s从点p密度可达。

```

DBSCAN算法描述:

```

输入: 包含n个对象的数据库，半径e，最少数目MinPts;

输出:所有生成的簇，达到密度要求。

(1)Repeat

(2)从数据库中抽出一个未处理的点；

(3)IF抽出的点是核心点 THEN 找出所有从该点密度可达的对象，形成一个簇；

(4)ELSE 抽出的点是边缘点(非核心对象)，跳出本次循环，寻找下一个点；

(5)UNTIL 所有的点都被处理。

DBSCAN对用户定义的参数很敏感，细微的不同都可能导致差别很大的结果，而参数的选择无规律可循，只能靠经验确定。

```

MeanShift

Mean Shift 这个概念最早是由Fukunaga等人于1975年在一篇关于概率密度梯度函数的估计（The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition ）中提出来的,其最初含义正如其名,就是偏移的均值向量,在这里Mean Shift是一个名词,它指代的是一个向量,但随着Mean Shift理论的发展,Mean Shift的含义也发生了变化,如果我们说Mean Shift算法,一般是指一个迭代的步骤,即先算出当前点的偏移均值,移动该点到其偏移均值,然后以此为新的起始点,继续移动,直到满足一定的条件结束.

然而在以后的很长一段时间内Mean Shift并没有引起人们的注意,直到20年以后,也就是1995年,另外一篇关于Mean Shift的重要文献（Mean shift, mode seeking, and clustering ）才发表.在这篇重要的文献中,Yizong Cheng对基本的Mean Shift算法在以下两个方面做了推广,首先Yizong Cheng定义了一族核函数,使得随着样本与被偏移点的距离不同,其偏移量对均值偏移向量的贡献也不同,其次Yizong Cheng还设定了一个权重系数,使得不同的样本点重要性不一样,这大大扩大了Mean Shift的适用范围.另外Yizong Cheng指出了Mean Shift可能应用的领域,并给出了具体的例子。

Comaniciu等人在还（Mean-shift Blob Tracking through Scale Space）中把非刚体的跟踪问题近似为一个Mean Shift最优化问题,使得跟踪可以实时的进行。目前，利用Mean Shift进行跟踪已经相当成熟。

2.3 具体实现方法及步骤

2.3.1 从三维晶体中提取二维晶面

提取晶面基于Fortran代码实现。主要任务是从一个三维的立体晶体中提取等距的晶面，从而进行后面的相关计算。

2.3.2 聚类

聚类过程我们使用Python来实现自动化，我们这里采用了默认的KMeans算法，同时可供选择的算法有AffinityPropagation, DBSCAN, MeamShift等，通过对比我们发现K-Means算法有着更好的稳定性与更快的计算速度，但是我们仍然提供其他的算法接口供使用者调用。

关于代码更多细节请参考github: [https://github.com/zyoohv/generate-two-dimensional-materials-automatic](https://github.com/zyoohv/generate-two-dimensional-materials-automatic)

首先我们需要设置相关的参数

```Python

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument('-i', '--input', type=str, help='path of input file.', default='data/example/inp1')

parser.add\_argument('--log', type=str, help='log file.', default='data/log')

parser.add\_argument('-v', type=int, help='visual result.', default=3, choices=[1, 2, 3])

args = parser.parse\_args()

```

我们使用Python提供的argparse模块来为用户提供可供选择的参数接口，以上以及提供的所有参数，具体用法详情可见“第四章”。

考虑到我们已经手动输入input文件，因此我们只需调用可执行程序即可。注意这里需要重定向输入，我们使用“0<”命令重新定义标准输入为从文件输入。具体代码如下：

```Python

cmd = 'lib/find-2D-crystal/src/find2D.x 0<{} 1>{} 2>&1'.format(input\_file, log\_file)

os.system(cmd)

```

最后我们提取文件内容，并进行聚类算法。有关聚类的更多细节，请参考下一节。

```Python

from ActionTools.ClusterAtoms import cluster\_from\_file

for xsf\_file in xsf\_list:

os.system('mv {} {}/{}'.format(xsf\_file, input\_dir, xsf\_file))

print('Generate "{}/{}" Done.'.format(input\_dir, xsf\_file))

prefix, result = cluster\_from\_file('{}/{}'.format(input\_dir, xsf\_file), method='KMeans\_method', plot\_image=plot\_image)

with open('{}/{}\_split'.format(input\_dir, xsf\_file), 'w') as fout:

for line in prefix:

fout.write(line)

fout.write(' {} 1\n'.format(str(len(result))))

for line in result:

fout.write(' {} {:.6f} {:.6f} {:.6f}\n'.format(str(line[0]), float(line[1]), float(line[2]), float(line[3])))

print('Generate "{}/{}\_split" Done.'.format(input\_dir, xsf\_file))

```

2.3.3 生成计算文件

我们根据pwscf软件的输入文件格式的要求，编写了自动生成自洽计算的文件代码如下。

关于代码更多细节请参考github: [https://github.com/zyoohv/generate-two-dimensional-materials-automatic](https://github.com/zyoohv/generate-two-dimensional-materials-automatic)

第一步仍然是接受用的参数，我们来看看源码接受那些参数：

```Python

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument('-i', '--input', type=str, help='input file dir', default='data/example')

parser.add\_argument('-t', '--type', type=str, help='upf file type', default='None')

args = parser.parse\_args()

```

参数的具体解释可见第四章详解。

我们需要提取前一步生成的而文件，并摘取相关的信息：

```Python

# get atoms dict

atoms\_list = []

atoms\_quality = []

with open('data/atom\_search\_list', 'r') as fin:

for line in fin:

thisLine = [item for item in line.strip().split(' ') if item]

atoms\_list.append(thisLine[0])

atoms\_quality.append(float(thisLine[1]))

def getItemName(atom):

item = ''

for i in atom:

item += str(i)

return item

# get information from file

filelist = glob.glob(inputdir + '/\*.inp.xsf\_split')

for filename in filelist:

file\_content = []

with open(filename, 'r') as fin:

for line in fin:

file\_content.append(

[item for item in line.strip().split(' ') if item])

......

```

接下来我们调用相关算法进行计算，并生成分割文件：

```Python

import sys

sys.path.append('.')

from ActionTools.getmodel import getscfin

from ActionTools import \*

# get information from file

filelist = glob.glob(inputdir + '/\*.inp.xsf\_split')

for filename in filelist:

......

vec = []

vec.append([float(i) for i in file\_content[2]])

vec.append([float(i) for i in file\_content[3]])

vec.append([float(i) for i in file\_content[4]])

atom = []

atomq = []

posi = []

numofAtoms = int(file\_content[10][0])

for i in range(11, 11 + numofAtoms):

atom.append(atoms\_list[int(file\_content[i][0]) - 1])

atomq.append(atoms\_quality[int(file\_content[i][0]) - 1])

posi.append(file\_content[i][1:])

item = getItemName(atom)

getscfin(item, atom, atomq, vec, posi, scf=True, outputdir=outputdir)

# getscfin(item, atom, atomq, vec, posi, scf=False, outputdir=outputdir)

```

2.3.4 计算并保存结果

这里我们使用迭代计算的方法，来计算经过改变沿某一方向拉伸后能量的变化。由于某些位置的不可计算性质，我们使用历史最大值来代替缺失值，实验证明这种方法不会影响最后的实验结果，同时我们可以顺利的进行其他位置的测验。

关于代码更多细节请参考github: [https://github.com/zyoohv/generate-two-dimensional-materials-automatic](https://github.com/zyoohv/generate-two-dimensional-materials-automatic)

首先仍然是接受参数的部分源码：

```Python

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument('-i', '--input', type=str, help='input file dir', default='data/example')

parser.add\_argument('--disnum', type=int, help='split distance', default=50)

args = parser.parse\_args()

```

各参数的具体使用及意义详见第四章。

接下来我们首先需要提取文件中我们需要的信息，比如把原子的原子序数转换为元素的名称。这里我们在data文件夹下面维护了一个字典，因此当需要查询例如8号原子时，我们只需家在字典，即可查到其对应的是O原子。同样，我们也维护了一个upf文件列表的字典，这样可以使我们最终的软件主题更小。当我们需要某个文件时，我们只需使用linux下的wget命令即可下载到对应目录中。

```Python

# we will read scf.in file such as SSMoMoSS.scf.in, so don't put any this kind of

# files in which directory.

scf\_in\_file = glob.glob(args.input + '/\*.scf.in')[0]

def getprimvec(file\_content):

index = np.where(file\_content == 'CELL\_PARAMETERS\n')[0][0]

vec = []

for i in range(index + 1, index + 4):

line = [float(i) for i in file\_content[i].strip().split(' ') if i]

vec.append(line)

# print('vec : ', vec)

return np.array(vec)

# we need calculate with them

# type : np.array

file\_content = []

vec = []

with open(scf\_in\_file, 'r') as fin:

file\_content = np.array([line for line in fin])

# print('file\_content : \n', file\_content)

vec = getprimvec(file\_content)

```

最后我们进行晶格参数的调整，并调用pw.x命令进行相关的计算。注意这里我们使用python提供的matplotlib库进行绘图，同时我们对缺省的值进行了相关处理。我们的策略是选取目前为止的最大值来代替缺省值，事实证明这种做法并不影响最后的结论：

```Python

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

# some config here.

mullow = 0.6

mulhigh = 3

tmp\_scf\_in = 'scf.in.tmp'

tmp\_scf\_out = 'scf.out.tmp'

energy = []

dislist = np.linspace(mullow, mulhigh, args.disnum)

def generatefile(origin\_file, vec):

# print('generatefile from origin : \n', origin\_file)

......

def generatevec(dismul, vec):

......

for dismul in dislist:

......

energy = np.array(energy)

# print('energy : ', energy)

fig = plt.figure()

ax = fig.add\_subplot(111)

print('X = \n', energy[:, 0])

print('Y = \n', energy[:, 1])

ax.plot(energy[:, 0], energy[:, 1], 'k-')

plt.savefig('result.jpg')

```

2.4 本章小结

本章我们主要讨论了二维材料相关计算的理论基于与部分的代码实现。

首先我们列出了部分关于二维材料的相关基本理论与概念，接下来我们介绍了关于统计学习方法中的聚类方法，并对每种算法的具体操作过程给出一个大致的讲解。最后我们讲解了实验的部分代码，对实验的大致思路也进行了大致的阐述。

第三章 安装操作系统

3.1 debian简介

我们选用的debian网址首页如下图

![debian](image/debian.png)

我们选用debian的原因：

1.软件包管理

Debian中的软件包由dpkg和apt-get来管理。如今，它们管理依赖项的能力不再很独特；几个对应工具(比如dnf和urpmi)有一两个选项是dpkg和apt-get所没有的。然而，它们仍是一大批用来查看和处理软件包的实用工具的核心。没有对应工具提供那么多的选项来解决安装冲突。这么多年后，dpkg和apt-get仍然独具一格。

2.风险的选择

Debian的三个主要软件库是Stable(稳定)、Testing(测试)和Unstable(不稳定)。这些名称体现了每一个软件库中的软件包在测试这条道路上走得有多远，让用户可以根据自己的情况来兼顾最新软件包和稳定性程度。

如果你主要关注的是稳定性，那可以坚持使用Stable。相比之下，如果你想要最新的软件，可以使用Unstable，不过面临这种风险：有更多的软件错误，不相兼容。Testing通常介于这两个极端之间。

当然了，风险程度是相对的。许多基于Debian的发行版(包括Ubuntu)使用Testing或者Unstable软件包，不过在发布之前进行自己的一番测试。除非准备进行技术方面的重大变化，比如上一个版本改用systemd，否则Unstable通常来说足够安全，如果你借鉴的对象仅限于非核心元素(比如桌面环境)，更是如此。

3.自由程度的选择

Debian软件库分为三个部分：Main、Contrib和Non-Free。Main完全包括免费许可证的软件，Contrib包括本身免费许可证的软件，但是依赖其他的非免费软件，而Non-Free包括采用专有许可证发布的软件。

Debian安装时只启用了Main，所以该项目的偏好显而易见。不过，添加另外两个部分只需要五分钟的时间来编辑/etc/apt/sources.list。我满足于坚持使用默认设置，不过我很欣赏这一点：Debian提供了一种选择，然后放心让用户做出正确的选择。

4.不一样的说明文档

Debian的说明文档散布于无数个网站，并不经常被讨论。然而，这些年来，我发现，如果我将“Debian”添加到我的互联网搜索，选择近些年来的结果，几乎总是会找到一个网页给出详细的逐步说明，帮助解决我要解决的那个问题。

5.迅速修复错误

我没听说过另外哪个发行版像Debian这么迅速地应对安全或者技术问题。无论Debian维护人员是单枪匹马还是团队合作，他们的响应时间表明了勤勤恳恳的工作态度，值得用户依赖。如果说Debian有时似乎比其他发行版更频繁地更新，那不是由于它有更多的错误，而是由于被其开发人员疏忽的错误比较少。

6.控制安装环节

Debian很早以前就取得了长足发展，戳穿了它很难安装这个流言。目前的安装程序是市面上最灵活的安装程序之一。如果你接受默认设置、安装它建议的一群软件包，它就会进行安装，安装时间与Fedora的Anaconda大致一样。然而如果你自行选择，安装和选择一个个软件包所花的时间要长得多。这就是为什么率先推出快速安装程序的Ubuntu建议使用快速解决问题的Debian安装程序版本。

7.Debian社区

除了技术方面外，Debian还拥有免费软件领域最庞大、最创新的社区之一。该项目的邮件列表上热烈地讨论政策和技术选择；重大问题在Debian维护人员当中进行投票表决，包括谁是下一任项目负责人。所有投票取决于Condorcet方法，这是最公正的表决方法之一。在过去，该社区常常不欢迎女性，但这种情况在改善；总体上来讲，Debian在力求包容方面所做的工作与其技术素养一样出名。

安装与配置

待添加。。。。。。

3.2 搭建开发环境

3.2.1 build-essential

我们首先要安装可能需要的编译环境及类库，这里我们直接使用apt提供的安装办法：

```

# apt install build-essential

```

3.2.2 Anaconda3

我们这里会用到数学计算的相关python库，比如numpy，matplotlib等，这里我们使用基于Python3.5的Anaconda3。

安装过程如下：

```

# ls

Anaconda3-4.2.0-Linux-x86\_64.sh

# ./Anaconda3-4.2.0-Linux-x86\_64.sh

Welcome to Anaconda3 4.2.0 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license

agreement.

Please, press ENTER to continue

>>>

================

Anaconda License

================

Copyright 2016, Continuum Analytics, Inc.

……

Do you approve the license terms? [yes|no]

>>> yes

Anaconda3 will now be installed into this location:

/home/yonghui/anaconda3

- Press ENTER to confirm the location

- Press CTRL-C to abort the installation

- Or specify a different location below

[/home/yonghui/anaconda3] >>>

PREFIX=/home/yonghui/anaconda3

installing: python-3.5.2-0 ...

installing: \_license-1.1-py35\_1 ...

installing: conda-build-2.0.2-py35\_0 ...

Python 3.5.2 :: Continuum Analytics, Inc.

creating default environment...

installation finished.

Do you wish the installer to prepend the Anaconda3 install location

to PATH in your /home/yonghui/.bashrc ? [yes|no]

[no] >>> yes

Prepending PATH=/home/yonghui/anaconda3/bin to PATH in /home/yonghui/.bashrc

A backup will be made to: /home/yonghui/.bashrc-anaconda3.bak

For this change to become active, you have to open a new terminal.

Thank you for installing Anaconda3!

Share your notebooks and packages on Anaconda Cloud!

Sign up for free: https://anaconda.org

```

3.2.3 fortran

由于我们的部分功能由fortran实现，因此我们需要安装一个fortran的编译器，这里我们选用的为gfortran。它的安装十分方便：

```

# apt install gfortran

```

3.2.4 git

Git是一款免费、开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。

![git](image/git.png)

Git是一个开源的分布式版本控制系统，可以有效、高速的处理从很小到非常大的项目版本管理。Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

Torvalds 开始着手开发 Git 是为了作为一种过渡方案来替代 BitKeeper，后者之前一直是 Linux 内核开发人员在全球使用的主要源代码工具。开放源码社区中的有些人觉得BitKeeper 的许可证并不适合开放源码社区的工作，因此 Torvalds 决定着手研究许可证更为灵活的版本控制系统。尽管最初 Git 的开发是为了辅助 Linux 内核开发的过程，但是我们已经发现在很多其他自由软件项目中也使用了 Git。例如 很多 Freedesktop 的项目迁移到了 Git 上。

linux上安装git比较方便

```

# apt install git

# git config --global user.name "zyoohv"

# git config --global user.email zyoohv@gmail.com

```

3.2.5 cmake

CMake是一个跨平台的安装（编译）工具，可以用简单的语句来描述所有平台的安装(编译过程)。他能够输出各种各样的makefile或者project文件，能测试编译器所支持的C++特性,类似UNIX下的automake。只是 CMake 的组态档取名为 CMakeLists.txt。Cmake 并不直接建构出最终的软件，而是产生标准的建构档（如 Unix 的 Makefile 或 Windows Visual C++ 的 projects/workspaces），然后再依一般的建构方式使用。这使得熟悉某个集成开发环境（IDE）的开发者可以用标准的方式建构他的软件，这种可以使用各平台的原生建构系统的能力是 CMake 和 SCons 等其他类似系统的区别之处。

在Linux下安装cmake的代码如下：

```

# apt install cmake

```

3.3 安装相关软件

3.3.1 pwscf

`pwscf`现改名为`Quantum ESPRESSO`，我们需要从官网下载源码进行安装。

![pwscf](image/pwscf.png)

```

# tar -zxvf qe-6.1.tar.gz

# cd qe-6.1/

# ./configure

# make all -j4

```

3.3.2 xcrysden

xcrysden为我们提供了一个可以直观的观察原子分布的方法。

![xcrysden](image/xcrysden.png)

我们选用xcrysden-1.5.60-linux\_x86\_64-semishared.tar.gz文件进行解压即可。

```

# tar -zxvf xcrysden-1.5.60-linux\_x86\_64-semishared.tar.gz

```

![xcrysdenrun](image/xcrysdenrun.png)

3.4 本章小结

本章我们首先在pc上介绍安装了debian版本的linux操作系统，用来构建我们的实验环境。我们介绍了我们因为稳定，功能强大等选用debian的原因。接下来我们初步对操作系统进行了配置，使之满足我们实验要求的最基本环境，我们主要是安装了`python`, `build-essential`, `gfortran`等基本编译环境。最后，我们编译安装了`pwscf`来进行原子计算的软件，以及成功部署了`xcrsden`软件。这样，我们最基本的实验环境即已经搭建成功。

最后，我们从`github`上下载了我们实验的源码，并进行相关的路径配置，使之可以正常工作。接下来，我们就可以在此基础上顺利的进行我们的实验环节了。

第四章 实验

4.1 从三维晶体中寻找二维晶面

4.1.1 构建输入文件

我们的三维晶体结构的输入文件详述如下。

首先第一行一个数字表示一个晶胞中的原子的个数。注意这个数字必须是整数，否则可能会发生无法预知的错误。

```

6 # atoms

```

接下来是三个晶矢，以空格隔开，每行分别代表a，b，c三个向量。注意我们所讨论的是在三维直角坐标系中为基础，单位是Angstrom。

```

3.170841 0.016158 -0.219126

-0.011973 5.508152 0.218269

-0.293847 0.558833 7.265195

```

接下来我们会设定搜索计算的相关参数。从第一行开始，分别代表的意思是：相邻晶面的最小距离，不用平面中原子的最小距离，原子之间新的距离，以及最大磁性，最小精度。

具体参数样例如下：

```

4 # minimal distance between crystal planes

3 # minimal distance between atoms of different planes

15 # new distance between atoms

20 # maximal magnitude of lattice vectors

0.01 # eps\_scale

```

4.1.2 调用计算

算法已经由Fortran实现，因此只需调用代码处理文件即可。

由于我们的代码已经对相关实现进行了封装，因此此处并不需要实验人员直接调用Fortran生成的可执行文件，只需调用我们事先写好的python脚本即可。

但是如果阅读我们的python源码，仍然可以在此处看到我们相关的实现细节。

代码位于 `g2dma/tools/find2D.py`

```python

input\_dir, \_ = os.path.split(input\_file)

cmd = 'lib/find-2D-crystal/src/find2D.x 0<{} 1>{} 2>&1'.format(input\_file, log\_file)

os.system(cmd)

......

```

4.2 聚类

4.2.1 scikit-learn

自2007年发布以来，scikit-learn已经成为Python重要的机器学习库了。scikit-learn简称sklearn，支持包括分类、回归、降维和聚类四大机器学习算法。还包含了特征提取、数据处理和模型评估三大模块。

sklearn是Scipy的扩展，建立在NumPy和matplotlib库的基础上。利用这几大模块的优势，可以大大提高机器学习的效率。

sklearn拥有着完善的文档，上手容易，具有着丰富的API，在学术界颇受欢迎。sklearn已经封装了大量的机器学习算法，包括LIBSVM和LIBINEAR。同时sklearn内置了大量数据集，节省了获取和整理数据集的时间。

4.2.2 安装sklearn

由于我们使用的Anaconda版本的Python，其已经自带sklearn库。如果使用其他版本的Python，或者需要更新sklearn版本，可使用python-pip管理工具进行安装。

```

# pip install sklearn

```

4.2.3 准备阶段-设计模式

为了方便拓展及维护，我们首先定义了一个公共父类，用来实现所有算法都需要的数据预处理、画图、数据增强等功能，我们稍后会详细的讲解这几个功能的实现方法。

我们的父类的核心代码如下：

```Python

class clusterNormal(object):

"""docstring for cluster

"""

def \_\_init\_\_(self, axis, atom, posi):

""" for example:

axis : [vec\_a, vec\_b, vec\_c]

atom : [[42], [16], [16]]

posi : [x1, x2, ...]

"""

...

def augAtoms(self):

"""augment dataset.

"""

...

def paint2D(self):

"""draw 2D image.

"""

...

def paint3D(self):

"""draw 3D image.

"""

...

def selectAtom(self):

"""slect atoms which we want.

"""

...

```

4.2.4 实现算法接口

有了父类代码，我们就可以简单的继承父类，同时只要简单的实现我们的算法部分就可以了。

我们已经实现了DBSCAN, MeanShift, AffinityPropagation, KMeans等算法，如果实验人员有其他需要，只需按照如下写法，简单的继承并重写`run`方法即可。

```Python

class DBSCAN\_method(clusterNormal):

"""docstring for DBSCAN\_method"""

def run(self):

...

class MeanShift\_method(clusterNormal):

"""docstring for MeanShift\_method"""

def run(self):

...

class AffinityPropagation\_method(clusterNormal):

"""docstring for AffinityPropagation\_method"""

def run(self):

...

class KMeans\_method(clusterNormal):

"""docstring for KMeans\_method"""

def run(self):

...

```

4.2.5 实验结果

我们使用经典的Kmean++算法进行聚类，初始核心数量为3。

实验结果我们用不同的颜色表示不同的类别，如图片所示：

![origin1](image/cluster\_origin1.png)

![cluster1](image/cluster1.png)

![origin2](image/cluster\_origin2.png)

![cluster2](image/cluster2.png)

4.3 调整晶距

4.4 自动化

到目前为止，我们可以输出显示一下三个过程相关主程序的使用方法：

find2D.py

```

stu@lele:~/zyoohv/Documents/g2dma$ python tools/find2D.py --help

usage: find2D.py [-h] [-i INPUT] [--log LOG] [-v {0,1,2,3}]

optional arguments:

-h, --help show this help message and exit

-i INPUT, --input INPUT

path of input file.

--log LOG log file.

-v {0,1,2,3} visual result.

```

如上所示，find2D文件的运行参数一共4个。其中-i参数全名为--input，代表输入文件的路径。--log参数表示记录文件的路径，此路径一般不需我们修改。-v参数表示我们现实聚类结果的方法：0表示不显示，1表示以一维的方式现实，2表示以二维的形式现实，3表示以三维的形式显示。

get\_scf\_file.py

```

stu@lele:~/zyoohv/Documents/g2dma$ python tools/get\_scf\_file.py --help

usage: get\_scf\_file.py [-h] [-i INPUT] [-t TYPE]

optional arguments:

-h, --help show this help message and exit

-i INPUT, --input INPUT

input file dir

-t TYPE, --type TYPE upf file type

```

如上所示，get\_scf\_file文件的运行参数一共有三个。其中-i参数全名为--input，代表输入文件所在目录（注意这里为目录，与find2D路径区别）。-t参数表示指定使用UPF赝势文件的类型，其接受字符串的形式作为输入，如-t "pbe-mt\_fhi"表示使用pbe-mt\_fhi作为中缀的赝势文件。其运行后会现在目标目录搜索需要的赝势文件是否存在，如果存在则验证成功，如果不成功则在赝势文件库中下载。如果赝势文件库中存在此类型的赝势文件，则下载到相关目录，否则报错目标赝势文件不存在。

calculatedis.py

```

stu@lele:~/zyoohv/Documents/g2dma$ python tools/calculatedis.py --help

usage: calculatedis.py [-h] [-i INPUT] [--disnum DISNUM]

optional arguments:

-h, --help show this help message and exit

-i INPUT, --input INPUT

input file dir

--disnum DISNUM split distance

```

如上所示，calculatedis文件的运行参数一共有三个。其中-i参数同样表示输入文件所在的目录（注意这里为目录，与find2D路径区别）。--disnum表示我们分开的间隔数量，此参数越大，则我们的计算约准确，但同时会更加的消耗时间。一般我们设置此参数为20~50即可。

第五章 全文总结与展望

5.1 全文总结

本文以二维材料的自动生成方法为中心进行了研究。首先在人们对二维的材料的基础上，总结了二维材料的有关基本概念。接着本文提出了一种从三维晶体自动生成二维材料的新的方法，并对该方法进行了详细的论述。

接着，本文介绍了从一台电脑上从安装操作系统开始，进一步进行环境的搭建，软件的安装配置的全过程。到此我们已经具备了实验的环境，接下来下载本文提供的源码进行实验过程。

本文在“从三维晶体中寻找二维晶面”，“聚类”，“调整晶距”，“自动化”四个方面对实验过程进行论述。第一步从三维晶体中寻找二维晶面，即从三维常见晶体中提取出我们需要的有关的二维晶面的所有信息。但是从第一步提取的信息是杂乱、冗余的，我们使用聚类的方法剔除了不相关的晶面信息，使我们的注意力集中到我们需要的单个晶胞之中。调整晶距过程实际上就是所谓的寻找适当的二维晶面结构，我们通过调整晶距的方法，计算费米能，根据计算结果判断二维材料理论上是否可行。最后，我们使用Python脚本使上述步骤自动实现，即二维材料的自动生成方法。

5.2 后续工作展望

我们使用提取信息-生成模型-调整参数的操作步骤来对二维材料进行探索与验证。但是我们仍然希望有更多，更灵活的寻找新的二维材料的方法，可以更加准确的搜索新的二维材料。

同时我们目前的实验大部分在命令行下进行，我们希望可以结合图形界面，给科研工作者一个更加直观的解决方案。同时我们也希望提供更多的参数来方便科研工作者的操作的过程。

关于效率问题，我们在比较有限的时间内，为了功能的尽快实现，采用了Python语言进行实现。但是我们会涉及大量的计算，因此我们往往希望程序更快一点，因为后期我们尝试使用C/C++来实现相关的功能。同时我们也希望可以利用CUDA技术来加速计算，利用GPU来提速我们的计算过程。