科研实践报告

摘 要

晶体是我们计算物理研究的主要对象之一，晶体中存在晶面和晶向的定义，以及其对应的指数的定义。我们通过一个程序可以在命令行中轻松地将这两种指数相互转换。这个程序包含了Python中的ase, argparse, fractals等程序库。

**关键字**： 晶向；晶面；Python ASE(Atomic Simulation Environment) ；Python Argparse ；Python Fractals ；

**背景**:

晶体的结构有对称性和周期性，而且周期性的方向不一定是沿着直角坐标的三个坐标轴的方向，所以我们在研究晶体的时候要引入另外一组基矢，基矢的方向就是晶胞的三条棱的方向。

确定晶面指数有三个步骤：

1. 先求出该晶面与晶体中的基矢的轴相交的长度r,s,t;
2. 分别取它们的倒数, 并对这三个分数进行通分；
3. 通分后三个分数的分子就是这个晶面的晶面指数h,k,l。

确定晶向指数也有三个步骤：

1. 先做一条平行于该晶向的直线，并使其通过坐标原点；
2. 在这条直线上任取一点，求其原子坐标；
3. 化为最简整数比，即为晶向指数, 用[u v w]表示。[1]

通过以上的分析，我们可以对晶面指数和晶向指数进行转换，对应的晶面和晶向是垂直的。在此就引出这两种指数转换的公式：

给定一个晶面的系数h,k,l,将晶面与整个晶体的基矢,的交点分别记作A, B, C,将基矢的原点记作O，将与这个晶面垂直的晶向向量记作。那么,,, 其中c为常数。方便起见，不妨设c=1. 又因为 垂直晶面，而,属于晶面，所以垂直,。 又因为=u+vw, =, ，故

(u+vw)()=0

(u+vw)()=0

只有两个方程，但是有三个未知数，所以只能解出和：

=

=

观察到两个解的分母相同,故u : v : w = : 。

然后我们只要求出u,v,w的最简整数比就行了。

u:v:w的最简整数比就是他们各自除以u,v,w的最大公约数gcd(u,v,w)即可。

同理，已知u,v,w时，利用同一个方程组，可以解得

=

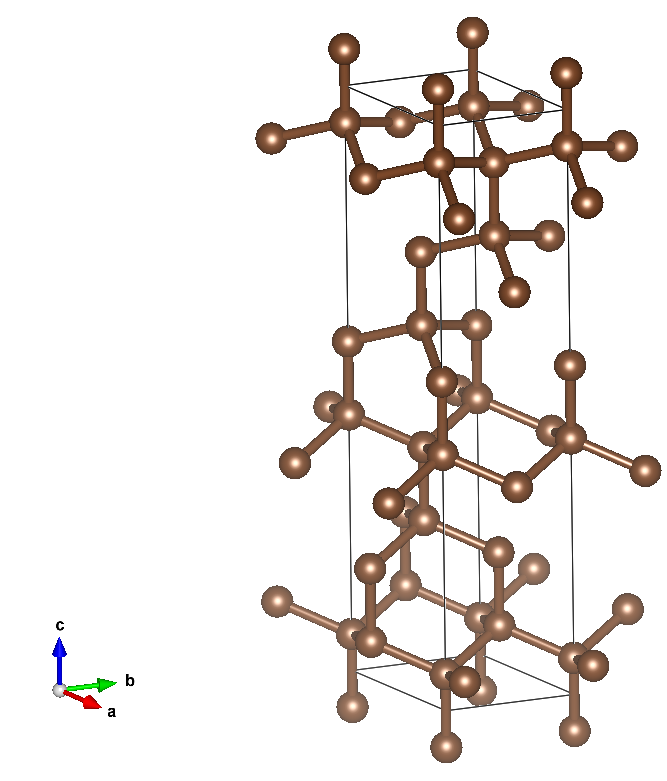
=

同理，它们的最简整数比就是它们解中的比例除以h,k,l的最大公约数。

通过以上的方法，我们就能设计出将晶面指数和晶向指数转换的程序了。

**工作内容:**

学习使用VESTA软件观察晶体的晶胞的情况。具体的测试的POSCAR文件如下图所示：



自学有关晶向晶面的内容。编写了一个用于转换晶体晶面和晶向指数的函数。

阅读Python的ase库的源代码[2]以及安装之后位于C:\Python27\Lib\site-packages\ase\io\vasp.py本地文件的源代码中的read\_vasp函数，我们可以看到get\_cell函数和read\_vasp函数可以配合从一个POSCAR文件中读取晶体的基矢的向量坐标参数。

通过阅读Python的argparse文档，将原来的程序包装成一个可以读取命令行参数的程序，通过输入-m,-i和POSCAR文件的名字可以分别在晶向和晶面指数之间相互转换[3]。

最终程序的源代码：

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 15 18:44:03 2017

@author: 93137

"""

import ase

import ase.io.vasp as vp

import numpy as np

import argparse

tolerance = 1e-2

parser = argparse.ArgumentParser(description='Convert miller index.')

parser.add\_argument('index', metavar='INDEX', type=str,

help='the index can either be miller index or orientation index')

parser.add\_argument('poscar', metavar='POSCAR', type=str,

help='poscar file where basis vectors come from')

parser.add\_argument('-m', dest='miller', action='store\_true',

help='convert miller index to orientation index')

parser.add\_argument('-i', dest='orientation', action='store\_true',

help='convert orientation index to miller index')

args = parser.parse\_args()

abc = ase.Atoms.get\_cell(vp.read\_vasp(args.poscar))

def gcd(a,b):

'''Calculate the Greatest Common Divisor of a and b

'''

while (b != 0) :

c = a % b

a = b

b = c

return a

def stridx2intidx(stridx):

'''Convert the index like '1,m3,1' to '1,-3,1'

'''

strarray = stridx.split(',')

intidx = [0,0,0]

for i in range(3):

if strarray[i][0] == 'm':

intidx[i] = -1 \* int(strarray[i][1:])

else:

intidx[i] = int(strarray[i])

return intidx

def ratioconvert(numerator,denominator,tolerance):

'''Convert a ratio of two floats into the ratio of two integers.

Tolerance means the tolerance of the bias bewteen these two ratios.

'''

prec = 1e-3

multiple = int(round(np.log10(1 / tolerance)))

scale = int(round(np.log10(abs(float(numerator) / float(denominator)))))

scale\_num = int(round(np.log10(abs(float(numerator)))))

intden = 10\*\*multiple

if (abs(numerator) <= prec)&(abs(denominator) <= prec):

return [0,0]

elif abs(numerator) <= prec:

return [0,intden]

elif abs(denominator) <= prec:

intnum = int(round(float(numerator),multiple - scale\_num) \* intden)

return [intnum,0]

else:

intnum = int(round(float(numerator) / float(denominator),multiple - scale) \* intden)

return [intnum,intden]

def absvec(abc):

'''Make the coordinates of vector positive

'''

n = 0

for i in range(3):

if (abc[i] < 0):

n = n + 1

if n >= 2:

for i in range(3):

abc[i] = (-1)\*abc[i]

return abc

def hkl2uvw(hkl,abc,tolerance):

'''Convert the miller index h,k,l into its related orientation index u,v,w

'''

h,k,l = hkl[0],hkl[1],hkl[2]

a,b,c = abc[0],abc[1],abc[2]

aa,bb,cc,ab,ac,bc = np.dot(a,a),np.dot(b,b),np.dot(c,c),np.dot(a,b),np.dot(a,c),np.dot(b,c)

u1 = (l\*bc-k\*cc)\*(k\*ab-h\*bb)-(k\*ac-h\*bc)\*(l\*bb-k\*bc)

v1 = (l\*bc-k\*cc)\*(h\*ab-k\*aa)-(k\*ac-h\*bc)\*(k\*ac-l\*ab)

w1 = (k\*ab-h\*bb)\*(k\*ac-l\*ab)-(l\*bb-k\*bc)\*(h\*ab-k\*aa)

#We have already solve the value of v/u and w/u

#Code below aims at turn u,v,w into their smallest integers

u2 = ratioconvert(u1,v1,tolerance)[0]

v2 = ratioconvert(u1,v1,tolerance)[1]

w2 = ratioconvert(w1,v1,tolerance)[0]

gcd\_uvw = gcd(gcd(abs(u2),abs(v2)),abs(w2))

u = u2 / gcd\_uvw

v = v2 / gcd\_uvw

w = w2 / gcd\_uvw

return absvec([u,v,w])

def uvw2hkl(uvw,abc,tolerance):

'''Convert the orientation index h,k,l into its related miller index u,v,w

'''

u,v,w = uvw[0],uvw[1],uvw[2]

a,b,c = abc[0],abc[1],abc[2]

aa,bb,cc,ab,ac,bc = np.dot(a,a),np.dot(b,b),np.dot(c,c),np.dot(a,b),np.dot(a,c),np.dot(b,c)

h1 = u\*aa + v\*ab + w\*ac

k1 = u\*ab + v\*bb + w\*bc

l1 = u\*ac + v\*bc + w\*cc

#We have already solve the value of h/k and l/k

#Code below aims at turn h,k,l into their smallest integers

h2 = ratioconvert(h1,k1,tolerance)[0]

k2 = ratioconvert(h1,k1,tolerance)[1]

l2 = ratioconvert(l1,k1,tolerance)[0]

gcd\_hkl = gcd(gcd(abs(h2),abs(k2)),abs(l2))

h = h2 / gcd\_hkl

k = k2 / gcd\_hkl

l = l2 / gcd\_hkl

return absvec([h,k,l])

if args.miller :

print hkl2uvw(stridx2intidx(args.index),abc,tolerance)

if args.orientation :

print tuple(uvw2hkl(stridx2intidx(args.index),abc,tolerance))

**困难**:

虽然整个晶面和晶向转换的程序原理看起来简单，但是在程序运行的过程中需要考虑的因素有很多，比如分母为零的情况，输出的结果的坐标大部分是负数的情况。我们还要考虑怎么将几个整数值比转化为最简整数比，

在将晶体的晶面指数（Miller指数）转换为晶向的指数的函数的时候，由于刚开始只考虑到了基矢, 的各个坐标的值是整数的情况,但实际上基矢的各个坐标可以是任意的小数,所以在原来的只能处理int类型的变量的情况下要增加处理float类型变量的情况。为了增加这种情况，我们要将两个float类型变量的除法转换为两个int类型变量的除法。

但是这种转换的方法并不简单，可能会遇到各种各样的问题，而且由于浮点数的不精确性（因为计算机记录浮点数是以2的次方形式记录的），所以其实浮点数在小于一定的数值（比如1e-15）的情况下就可以认为是零了。所以还要加入一定的零的判断语句。

但是这里存在一个精确度和合理性的矛盾，如果把精确度调整得太高，那么可能会得出一些指数比例相差巨大的解，比如[3800000, 1, 237000],这种解一般不是我们期望的解，我们可以将其转换为[1460, 0, 91]这样的结果，或者更不精确一点就是[50, 0, 3]。

从中我们可以看到，随着结果的简洁性越来越好，结果的精确度也越来越差，所以我们要在简洁性和精确性中进行取舍。

在学长的指导下，我知道了有Python的Fractals库，这个库提供了将float的除法转换为int的除法，相当于可以替换我所写的源代码中的ratioconvert函数，不过这个库的具体的使用方法我目前还没有掌握。

**后续工作计划**:

深入了解有关晶向、晶面等晶体相关的内容。

继续完善晶向晶面指数转换的py文件，优化代码，同时提升精确度和准确性。

研究脚本作图 (示意图) 软件

[Solides usuels](http://asy.marris.fr/asymptote/Solides/index.html)  ( <http://asy.marris.fr/asymptote/Solides/index.html> )

[Surfaces 3D](http://asy.marris.fr/asymptote/Surfaces_3D/index.html) ( <http://asy.marris.fr/asymptote/Surfaces_3D/index.html> )

<http://asymptote.sourceforge.net/gallery/2D%20graphs/index.html>

**参考文献**：

[1] 第四章 晶向、晶面等概念, 吉林大学微纳传感与器件实验室. Retrieved at March 25, 2017, from <http://smdlab.jlu.edu.cn/background/attachments/crystallography/chapter4.pdf>

[2] Source code for ase.atoms, Atomic Simulation Environment. Retrieved at March 25, 2017, from <https://wiki.fysik.dtu.dk/ase/_modules/ase/atoms.html#Atoms.get_celldisp>

[3] argparse — Parser for command-line options, arguments and sub-commands. Retrieved at March 19, 2017, from <https://docs.python.org/2/library/argparse.html>