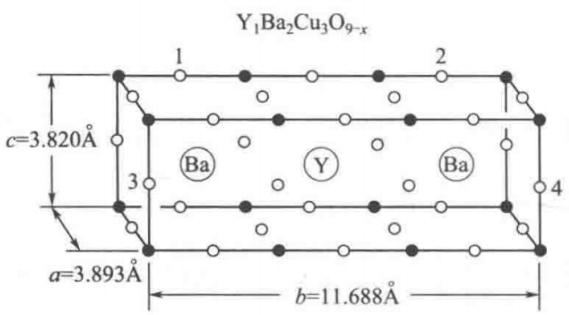
X 光编程作业

张锦程 材84 2018012082

编程语言：Python　version：3.7.7　date：2020/10/18　structure：$YBa_2Cu_3O_{9-x}(x \approx 2)$

本项目利用理论计算的方式计算超导材料钇钡铜氧的 X 射线衍射线的相对强度，假定样品为粉末状多晶体，X 射线采用 $Cu-K_\alpha$ 特征辐射（1.54178 Å）。

假定 $YBa_2Cu_3O_{9-x}(x \approx 2)$ 体系中 $x=2$，仅考虑 Ba 面或 Cu 面上缺氧的两种可能性，两种可能性如下图所示：



■ 图 5.15　$YBa_2Cu_3O_{9-x}$（$x \approx 2$）的相结构（1987 年 IBM 推荐）

Ba 面上缺氧即图中的 1、2 位置，Cu 面上缺氧即图中的 3、4 位置。

目录

粉末多晶样品 X 射线衍射强度计算

计算 X 射线散射强度首先要求 Y、Ba、Cu、O 等原子的散射因子 f_i（$\frac{该种原子产生的散射线振幅}{一个电子产生的散射线振幅}$），它是一个和散射角和 X 光波长都相关的量，具体数值可查相关晶体学用表进行插值估算，也可通过经验公式：$f_i=(sin\theta/\lambda)=\sum_{j=1}^{4}a_{j_i}exp[-b_{j_i}(sin\theta/\lambda)^2]+c_i$ 进行近似计算。

得到散射因子后可由 $F_{hkl}^2=\sum_{j}^{n}\sum_{k}^{n}f_jf_ke^{i2\pi\vec{s}\cdot r_{jk}}$ 计算结构因子 F_{hkl}^2，它的定义为 $\frac{单个晶格产生的散射线强度}{一个电子产生的散射线强度}$，$r_{jk}$ 为晶格中第 i、j 号原子的相对坐标，$\vec{s}$ 可理解为动量空间（也叫倒易空间）中波矢的改变量 $\Delta\vec{k}$。

散射线的强度和 F_{hkl}^2 成正比，同时也和散射角（布拉格角）θ 有关，具体来说，是和 $\frac{1+cos^2 2\theta}{sin^2\theta cos\theta}$ 成正比，上式也叫角因子（LP）。

除了结构因子 F_{hkl}^2 和 角因子（LP）之外，多晶粉末散射强度还和另一个叫做多重因子（P）的量成正比，简单说来，是因为我们计算的 X 射线衍射强度是针对特定晶面族 hkl 而言的，但每种晶面族所包含的晶面 (hkl) 数目不一样，所以需要乘上这个数目上的修正。对 $YBa_2Cu_3O_{9-x}$ 所属的正交晶系而言，hkl 都不为 0 时 P 为 8；当 h、k、l 中有一个为 0 时 P 为 4；当 h、k、l 中有两个为 0 时 P 为 2。

如果要计算多晶粉末衍射的绝对强度，则还要考虑试样厚度、温度等众多其它影响因素。然而如果只要求相对散射强度，则讨论到此为止，因为其他因素对同一测试条件下同一式样中不同晶面族的散射强度影响相同。最终结果为： $I_{相对} = F_{hkl}^2(LP)P$

一些其他方面的说明

在倒易点阵的计算中，有的参考书为了方便进行 Fourier 变换而引入了 2π 的常量，本计算中相关常数取值为 1 ，也就是说 $\vec{a}_i \cdot \vec{b}_j = \delta_{ij}$

单位制方面，无特殊说明，以下计算距离单位为：Å；角度取弧度制；X射线散射强度参考电子衍射强度，且为相对值。

带解析源码部分：

```
# 引入计算库
import numpy as np
import matplotlib.pyplot as plt
import openpyxl
```

引入基本的模型输入数据：

```
# 正交晶系晶格基矢
a1 = 3.893 * np.array([1,0,0])
a2 = 11.688* np.array([0,1,0])
a3 = 3.820 * np.array([0,0,1])
a = a1+a2+a3

wav_len = 1.54178 # 所用 X 射线的波长

# 诸原子的坐标
coor_Ba = {'Y':[[1/2,1/2,1/2]],'Ba':[[1/2,1/6,1/2],[1/2,5/6,1/2]],
           'Cu':[[0,0,0],[0,1/3,0],[0,2/3,0]],'O':[[1/2,0,0],[0,0,1/2],[1/2,1/3,0],[0,1/3,1/2],
                 [0,1/2,0],[1/2,2/3,0],[0,2/3,1/2]]}
coor_Cu = {'Y':[[1/2,1/2,1/2]],'Ba':[[1/2,1/6,1/2],[1/2,5/6,1/2]],
           'Cu':[[0,0,0],[0,1/3,0],[0,2/3,0]],'O':[[0,1/6,0],[1/2,1/3,0],[0,1/3,1/2],
                 [0,1/2,0],[1/2,2/3,0],[0,2/3,1/2],[0,5/6,0]]}

# 查表得到的 f_i 计算式中的诸常数数值
#############        a1       b1       a2       b2       a3       b3       a4        b4         c
fi_consts={'Y' :[17.9268,1.35417,9.15310,11.21450,1.76795,22.6599,-33.108,-0.01319,40.2602],
           'Ba':[20.1807,3.21367,19.1136,0.283310,10.9054,20.0558,0.77634,51.74600,3.02920],
           'Cu':[11.8168,3.37484,7.11181,0.244078,5.78135,7.98760,1.14523,19.89700,1.14431],
           'O' :[4.19160,12.8573,1.63969,4.172360,1.52673,47.0179,-20.307,-0.01404,21.9412]}
```

定义计算子程序

通过经验式计算各原子的散射因子 f_i：

```
def cal_fi(name,X):  # name='Y','Ba','Cu','O'; X=sin(theta)/lambda
    cs = fi_consts[name]
    fi = cs[-1]   # c
    for i in range(4):
        fi += cs[2*i]*np.exp(-cs[2*i+1]*X**2)
    return fi
```

对 $f_i = f(\frac{sin\theta}{\lambda})$ 经验公式进行检验如下：

```
# 经多组数据验算，用以上参数计算得到的 fi 貌似和《X射线衍射技术》（潘峰版）书后附表 10 所列各元素参考值有小出入，但是差别不大，普遍在 1 以内，可能是参照版本不同导致
cal_fi('Y',0.1),cal_fi('Ba',0.1),cal_fi('Cu',0.1),cal_fi('O',0.1)
```

结果：

```
(34.42508090192148, 51.01752454025608, 25.939560823721287, 7.8439522512047475)
```

由晶体学的相关理论可以证明，当 $\vec{a}_i$ 和 $\vec{b}_i$ 乘积为 1 时，$\{hkl\}$ 晶面族的晶面间距为同名倒易向量 $\vec{g}_{hkl}$ 欧几里得测度的倒数,计算过程如下：

```
# 计算倒易矢量
V = np.dot(np.cross(a1,a2),a3)
b1= np.cross(a2,a3)/V
b2= np.cross(a3,a1)/V
b3= np.cross(a1,a2)/V

def c_scale(A): # 求解 A 中各数组的 K1 测度（欧几里得测度）
    A = np.array(A)
    n_dim = A.shape[1]; n_size = A.shape[0]
    scale2 = np.zeros(n_size)
    for j in range(n_dim):
        for i in range(n_size):
            scale2[i] += A[i,j]**2
    scale = scale2 ** 0.5
    return scale

# 通过倒易矢量求解 {hkl} 晶面族的晶面间距
def cal_dist(index,b1,b2,b3):                # index=[h,k,l], bi为倒易点阵基矢
    g = index[0]*b1+index[1]*b2+index[2]*b3  # 同名倒易矢量
    distance = 1/c_scale([g])[0]             # 晶面间距
    return distance
```

由基础的 X 射线衍射知识，hkl 晶面衍射线的结构因子为：$F_{hkl}^2 = \sum_j^n \sum_k^n f_j f_k e^{i2\pi \vec{s}\cdot r_{jk}}$ ，其中 n 为晶胞中原子的个数，计算过程如下：

```
def cal_f2(plane,atom_Y,atom_Ba,atom_Cu,atom_O):
    # 将所有的原子信息浓缩到一个列表中以方便进行加和运算
    atom_coors = np.concatenate((atom_Y.coor,atom_Ba.coor,atom_Cu.coor,atom_O.coor))
    atom_fis   = np.concatenate((atom_Y.fi*np.ones(len(atom_Y.coor)),atom_Ba.fi*np.ones(len(//
      atom_Ba.coor)),atom_Cu.fi*np.ones(len(atom_Cu.coor)),atom_O.fi*np.ones(len(atom_O.coor))))
    f2 = 0   # 初始值为 0
    for j in range(len(atom_coors)):
        for k in range(len(atom_coors)):
            f2 += atom_fis[j]*atom_fis[k]*np.cos(2*np.pi*(np.dot(np.array(plane.index),
                        (np.array(atom_coors[j])-np.array(atom_coors[k])).T)))
    return f2
```

由于 $YBa_2Cu_3O_{9-x}$ 体系为正交晶格，对它而言，h、k、l 都不为 0 时 P 为 8；当 h、k、l 中有一个为 0 时 P 为 4；当 h、k、l 中有两个为 0 时 P 为 2：

```
def cal_p(index):
    count = 0
    for i in index:
        if i == 0:
            count+=1
    if count == 0:
        return 8
    if count == 1:
        return 4
    if count == 2:
        return 2
    return 0
```

```
# 定义几个类：晶面 Plane；原子：Atom；初始值取 0（这几乎不可能）以便 debug
class Plane:
    def __init__(self, index):
        self.index = index       # 接受一个 list：[h,k,l] 作为晶面的密勒指数
        self.name  = '{000}'     # 该晶面族的名称
        self.dist  = 0           # 该晶面的晶面间距
        self.theta = 0           # 该晶面的衍射角（布拉格角）
        self.lp    = 0           # 该晶面所产生的衍射线的角因子 LP
        self.f2    = 0           # 该晶面所产生的衍射线对应的结构因子 F**2
        self.p     = 0           # 该晶面所对应的多重因子（等效晶面有多少个）
        self.i     = 0           # 该晶面对应的衍射强度

class Atom:
    def __init__(self, name):
        self.name  = name        # 该原子的名称
        self.coor  = 0           # 该类原子在晶格中的坐标位置
        self.fi    = 0           # 某种情况下该原子的散射因子 fi
```

定义各晶面计算循环结构

以下开始分情况（Cu 面或者 Ba 面缺氧）循环求解，考虑到原子数最多的 a_2 方向上一个晶胞共 6 个原子，所以考虑 h、k、l 的取值范围为 $[0,6]$ 内的整数：

```
# 诸原子坐标 coor；X射线波长 wav_len；经验方程参数 fi_consts
def cal_i(coor, wav_len,fi_consts,b1,b2,b3):
    atom_Y  = Atom('Y');  atom_Y.coor  = coor[atom_Y.name]
    atom_Ba = Atom('Ba'); atom_Ba.coor = coor[atom_Ba.name]
```

```
    atom_Cu = Atom('Cu'); atom_Cu.coor = coor[atom_Cu.name]
    atom_O  = Atom('O');  atom_O.coor  = coor[atom_O.name]
    hkls = []; doub_thetas = []; ds = []; Is = []
    lps = []; f2s = []; ps = []
    for h in range(7):
        for k in  range(7):
            for l in range(7):
                plane = Plane([h,k,l])
                plane.name  = '{'+str(h)+str(k)+str(l)+'}'
                plane.dist  = cal_dist(plane.index,b1,b2,b3)
                plane.theta = np.arcsin(wav_len*0.5/plane.dist)
                plane.lp    = (1+np.cos(2*plane.theta)**2)/
                                    (np.sin(plane.theta)**2*np.cos(plane.theta))
                atom_Y.fi   = cal_fi(name=atom_Y.name, X=np.sin(plane.theta)/wav_len)
                atom_Ba.fi  = cal_fi(name=atom_Ba.name,X=np.sin(plane.theta)/wav_len)
                atom_Cu.fi  = cal_fi(name=atom_Cu.name,X=np.sin(plane.theta)/wav_len)
                atom_O.fi   = cal_fi(name=atom_O.name, X=np.sin(plane.theta)/wav_len)
                plane.f2    = cal_f2(plane,atom_Y,atom_Ba,atom_Cu,atom_O)
                plane.p     = cal_p(plane.index)
                plane.i     = plane.f2 * plane.lp * plane.p
                #print([atom_Y.fi,atom_Ba.fi,atom_Cu.fi,atom_O.fi])
                # 储存数据
                hkls.append(plane.name);doub_thetas.append(2*plane.theta)
                ds.append(plane.dist); Is.append(plane.i)
                lps.append(plane.lp); f2s.append(plane.f2); ps.append(plane.p)
    return hkls,doub_thetas,ds,Is,lps,f2s,ps

hkls1,doub_thetas1,ds1,Is1,lps1,f2s1,ps1 = cal_i(coor_Ba, wav_len,fi_consts,b1,b2,b3)
hkls2,doub_thetas2,ds2,Is2,lps2,f2s2,ps2 = cal_i(coor_Cu, wav_len,fi_consts,b1,b2,b3)
```

程序数据导出及处理

上面得到得数据还没有按照 2θ 大小进行排序，将它导入到 excel 表格中以进行下一步的数据整理：

```
from openpyxl import Workbook
wb1 = Workbook(); wb2 = Workbook()# 激活 worksheet
ws1 = wb1.active; ws2 = wb2.active

doub_thetas1 = np.nan_to_num(doub_thetas1); Is1 = np.nan_to_num(Is1);
doub_thetas2 = np.nan_to_num(doub_thetas2); Is2 = np.nan_to_num(Is2)

doub_thetas_h1=[]; doub_thetas_h2=[]; Is_r1=[]; Is_r2=[]
for i in range(len(doub_thetas1)):
    doub_thetas_h1.append(doub_thetas1[i]*180/np.pi)
    doub_thetas_h2.append(doub_thetas2[i]*180/np.pi)
    Is_r1.append(Is1[i]*100/np.max(Is1))
    Is_r2.append(Is2[i]*100/np.max(Is2))

ws1.append([     'hkl',        '2θ',           '2θ/(°)',      'd/Å',    'I',     'I相对'])
ws2.append([     'hkl',        '2θ',           '2θ/(°)',      'd/Å',    'I',     'I相对'])
for i in range(len(hkls1)):
    ws1.append([hkls1[i], doub_thetas1[i], doub_thetas_h1[i], ds1[i], Is1[i], Is_r1[i]])
    ws2.append([hkls2[i], doub_thetas2[i], doub_thetas_h2[i], ds2[i], Is2[i], Is_r2[i]])

```

```
wb1.save('results/钇钡铜氧Ba面缺氧.xlsx'); wb2.save('results/钇钡铜氧Cu面缺氧.xlsx')
```

经过处理的部分结果如下图所示，完整结果参考目录中的 Excel 文件。

表一：钇钡铜氧 Ba 面缺氧

hkl	2θ	2θ/°	d/Å	计算强度 I	归一化强度 I′	参考值 I0
{010}	0.1320	7.5635	11.6880	73353.5736	2.2685	1.0000
{020}	0.2646	15.1601	5.8440	17401.5055	0.5381	2.0000
{030}	0.3984	22.8245	3.8960	38754.6269	1.1985	12.0000
{100}	0.3987	22.8423	3.8930	231693.8623	7.1652	12.0000
{001}	0.4064	23.2849	3.8200	221176.1620	6.8399	2.0000
{120}	0.4805	27.5292	3.2399	62078.8017	1.9198	5.0000
{021}	0.4870	27.9020	3.1975	59785.3376	1.8489	5.0000
{040}	0.5340	30.5940	2.9220	3638.3776	0.1125	2.0000
{130}	0.5675	32.5125	2.7538	3233613.7792	100.0000	58.0000
{031}	0.5731	32.8337	2.7276	3151078.3836	97.4476	100.0000
{101}	0.5733	32.8465	2.7266	2385937.0085	73.7855	100.0000
{111}	0.5891	33.7544	2.6553	11529.3032	0.3565	2.0000
{121}	0.6346	36.3583	2.4709	9649.9177	0.2984	3.0000
{050}	0.6721	38.5107	2.3376	2100.3338	0.0650	26.0000
{140}	0.6723	38.5218	2.3370	22990.8491	0.7110	26.0000
{041}	0.6772	38.7993	2.3209	22482.3402	0.6953	5.0000
{131}	0.7047	40.3747	2.2339	829071.2523	25.6392	18.0000
{051}	0.7939	45.4893	1.9939	13561.7663	0.4194	1.0000
{060}	0.8137	46.6235	1.9480	880918.0877	27.2425	37.0000
{200}	0.8144	46.6616	1.9465	878655.2147	27.1725	37.0000
{002}	0.8309	47.6078	1.9100	824611.2748	25.5012	15.0000
{151}	0.8987	51.4923	1.7747	4162.3370	0.1287	6.0000
{160}	0.9168	52.5286	1.7421	51290.6542	1.5862	4.0000
{230}	0.9173	52.5547	1.7413	17227.5340	0.5328	4.0000
{061}	0.9206	52.7467	1.7354	50716.6819	1.5684	4.0000
{201}	0.9212	52.7815	1.7343	50626.0966	1.5656	4.0000
{211}	0.9321	53.4049	1.7155	16078.1591	0.4972	3.0000
{032}	0.9324	53.4232	1.7150	16751.1821	0.5180	3.0000
{102}	0.9326	53.4318	1.7147	48973.4394	1.5145	3.0000
{221}	0.9642	55.2459	1.6627	14388.6608	0.4450	1.0000
{122}	0.9752	55.8766	1.6454	13863.8926	0.4287	1.0000
{161}	1.0158	58.2027	1.5850	969700.6731	29.9881	44.0000
{231}	1.0163	58.2271	1.5844	1194306.0844	36.9341	44.0000
{132}	1.0269	58.8369	1.5695	1156223.7431	35.7564	23.0000

表二：钇钡铜氧 Cu 面缺氧

hkl	2θ	2θ/°	d/Å	计算强度 I	归一化强度 I′	参考值 I0
{010}	0.1320	7.5635	11.6880	38.3770	0.0012	0.5000
{020}	0.2646	15.1601	5.8440	246480.3395	7.8301	6.0000
{030}	0.3984	22.8245	3.8960	232131.0666	7.3743	9.0000
{100}	0.3987	22.8423	3.8930	114944.3968	3.6515	9.0000

hkl	2θ	2θ/°	d/Å	计算强度 I	归一化强度 I′	参考值 I0
{001}	0.4064	23.2849	3.8200	110103.0630	3.4977	3.0000
{110}	0.4205	24.0941	3.6935	42272.6205	1.3429	1.0000
{011}	0.4279	24.5153	3.6310	40437.1478	1.2846	1.0000
{040}	0.5340	30.5940	2.9220	38915.9426	1.2363	0.2000
{130}	0.5675	32.5125	2.7538	2827276.7436	89.8160	48.0000
{031}	0.5731	32.8337	2.7276	2756485.6400	87.5671	100.0000
{101}	0.5733	32.8465	2.7266	3147854.8680	100.0000	100.0000
{111}	0.5891	33.7544	2.6553	115509.2263	3.6695	2.0000
{121}	0.6346	36.3583	2.4709	907.5878	0.0288	0.3000
{050}	0.6721	38.5107	2.3376	228.3627	0.0073	0.3000
{140}	0.6723	38.5218	2.3370	11709.1159	0.3720	0.3000
{041}	0.6772	38.7993	2.3209	11467.4114	0.3643	0.3000
{131}	0.7047	40.3747	2.2339	829071.2523	26.3377	16.0000
{051}	0.7939	45.4893	1.9939	7177.1180	0.2280	0.8000
{060}	0.8137	46.6235	1.9480	880918.0877	27.9847	28.0000
{200}	0.8144	46.6616	1.9465	878655.2147	27.9128	28.0000
{002}	0.8309	47.6078	1.9100	824611.2748	26.1960	13.0000
{151}	0.8987	51.4923	1.7747	28780.2865	0.9143	0.8000
{160}	0.9168	52.5286	1.7421	32002.3740	1.0166	1.0000
{230}	0.9173	52.5547	1.7413	51221.3710	1.6272	1.0000
{061}	0.9206	52.7467	1.7354	31692.4789	1.0068	1.0000
{201}	0.9212	52.7815	1.7343	31643.5377	1.0052	1.0000
{211}	0.9321	53.4049	1.7155	8888.2694	0.2824	2.0000
{032}	0.9324	53.4232	1.7150	48994.8095	1.5565	2.0000
{102}	0.9326	53.4318	1.7147	30749.0253	0.9768	2.0000
{221}	0.9642	55.2459	1.6627	8033.4067	0.2552	0.3000
{122}	0.9752	55.8766	1.6454	7766.4274	0.2467	0.3000
{161}	1.0158	58.2027	1.5850	1195864.6901	37.9898	35.0000
{231}	1.0163	58.2271	1.5844	1078439.9824	34.2595	35.0000
{132}	1.0269	58.8369	1.5695	1044658.1265	33.1863	16.0000

绘图区域

上面已经得到了我们想要的结果，在整理数据的同时，我们可以使用 matplotlib 库来对 $I-2\theta$ 关系进行分析

```
plt.rcParams['figure.figsize'] = (16.0, 10.0)
plt.rcParams['figure.dpi'] = 300
plt.rcParams['font.size'] = 16

fig, [ax1, ax2] = plt.subplots(2, 1, sharex=True)

ax1.stem(doub_thetas_h1, Is_r1, linefmt='k', markerfmt='k.', basefmt='k')
ax1.set_title('Ba 面缺氧的情况')
ax1.set_ylabel('相对强度 (I)');ax1.set_xlabel('布拉格角'+r'($2\theta$)')
ax2.stem(doub_thetas_h2, Is_r2, linefmt='k', markerfmt='k.', basefmt='k')
ax2.set_title('Cu 面缺氧的情况')
ax2.set_ylabel('相对强度 (I)');ax2.set_xlabel('布拉格角'+r'($2\theta$)')
```

```
13  plt.tight_layout()  #设置默认的间距
14
15  plt.savefig('img/X射线衍射强度分布.png')
```

绘出的数据图像如下所示：