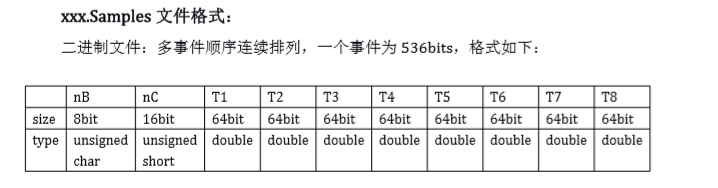
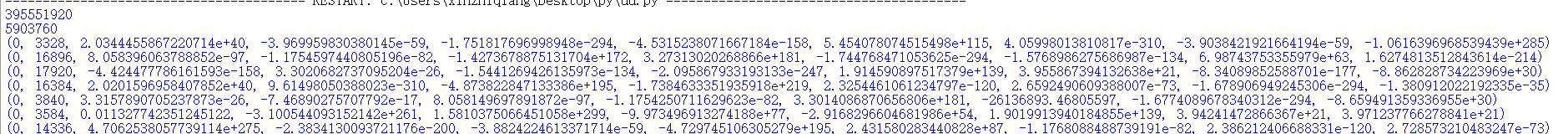
1、读取Samples文件数据：

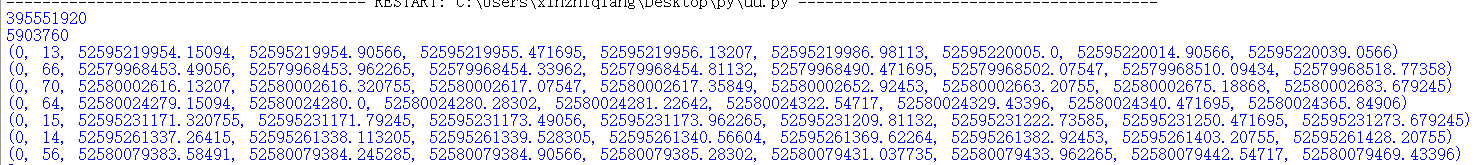
读取二进制文件计算长度，文件中共有395551920字节个数据，因每一帧（事件）占67字节，共有5903760个事件。



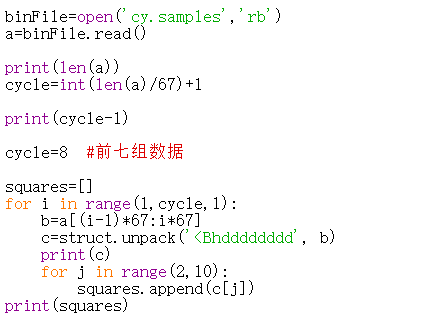
根据以上每一帧数据的存储格式，对应的恢复Format应为Bhdddddddd，起初我默认字节顺序为大端表示即最高位在前的形式进行恢复，发现恢复后的数据与老师发给我的不相符如图所示：



仔细观察发现前第二个数，发现转换成二进制后正好高低位颠倒，文件中的字节顺序为小端表示即最低为在前，修改后回复的数据完全吻合，如图所示：



程序如图：



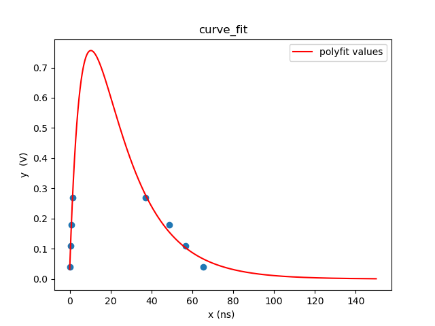
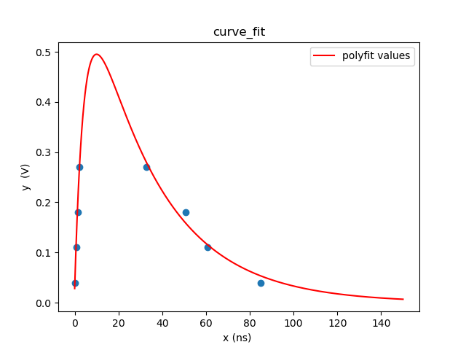
2、根据下面所示脉冲形状，选取一种拟合方法，对脉冲进行拟合：

根据脉冲形状和论文中所给出的公式，采用双指数函数对脉冲进行拟合，拟合公式如下：

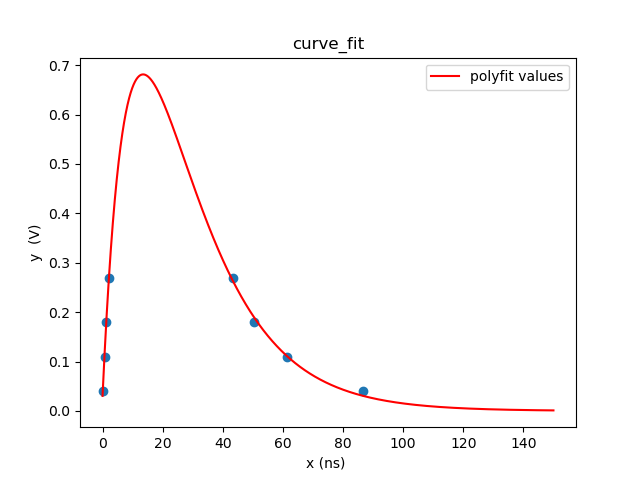
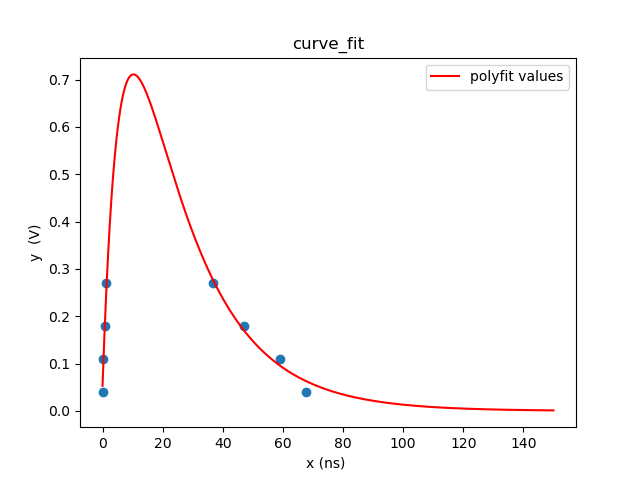


横坐标单位为ns，纵坐标单位为V，拟合后的曲线以横坐标间隔20ps在0-150ns间画出曲线脉冲形状，前七个脉冲拟合图形如下：

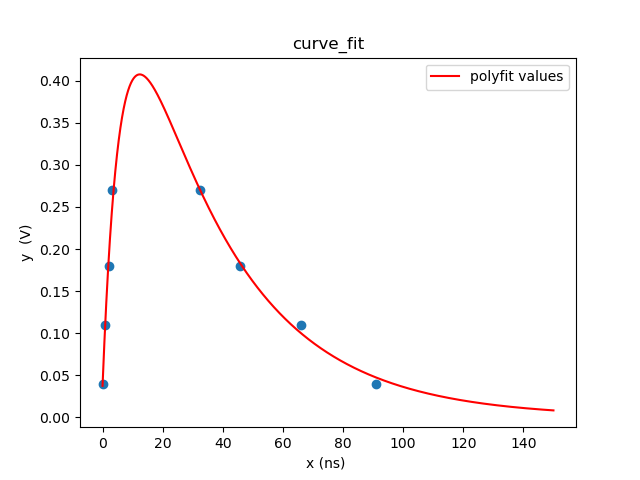
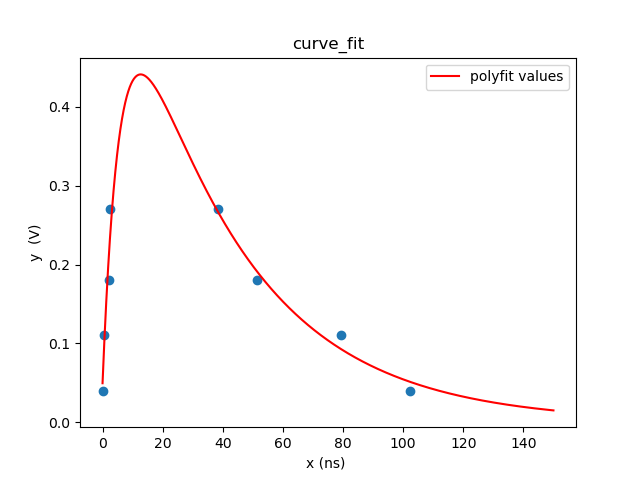
事件1 事件2



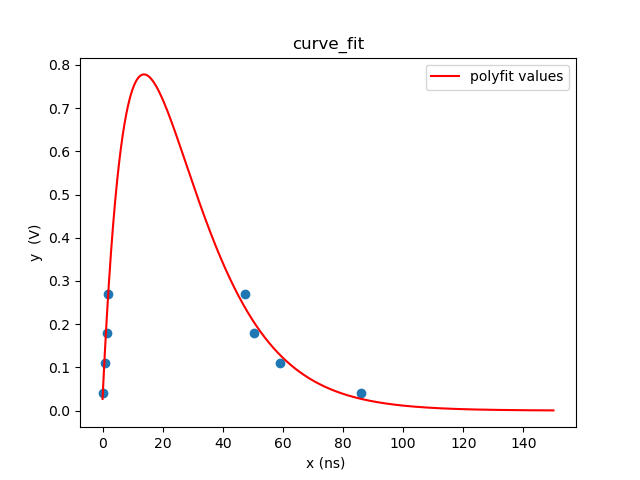
事件3 事件4



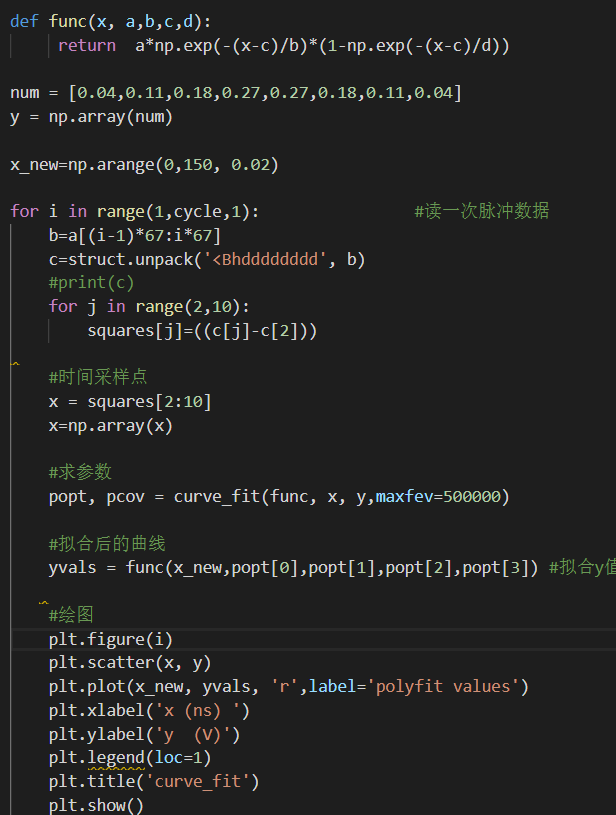
事件5 事件6



事件7



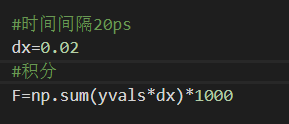
程序如下：



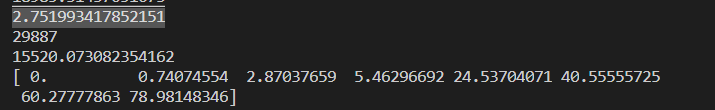
3、计算每个脉冲事件的能量（脉冲积分）：

根据第二步骤拟合出来的脉冲函数，和论文中描述一个脉冲的持续时间不超过150ns，因此脉冲积分的区间选择在脉冲起点到150ns间进行积分。

积分方式采用最简单的积分方式，将横坐标划分成若干个很小的片段作为小矩形的宽度，再以当前x值所对应的纵坐标y值作为小矩形的高，求取小矩形的面积并累加求和。



所得到的积分值在两万左右上下浮动，但多次观察发现，偶尔会出现积分值特别小的情况或出现负值的情况，如图所示：



即第29887个事件的积分值为2.751993417852151。将此采样数据拿出来，拟合曲线发现拟合形状与预计形状完全不同，所以是拟合错误所导致；经多次实验，对纵坐标进行不同放缩后可得到拟合脉冲，所以在程序中可通过脉冲积分值来判断拟合的成功与否，当能量积分超出正常区间外，换不同放缩的纵坐标再次拟合，再次计算脉冲积分，对应不同放缩在积分后也要进行不同加权。此例重新拟合计算后积分值为：15520.073082354162。

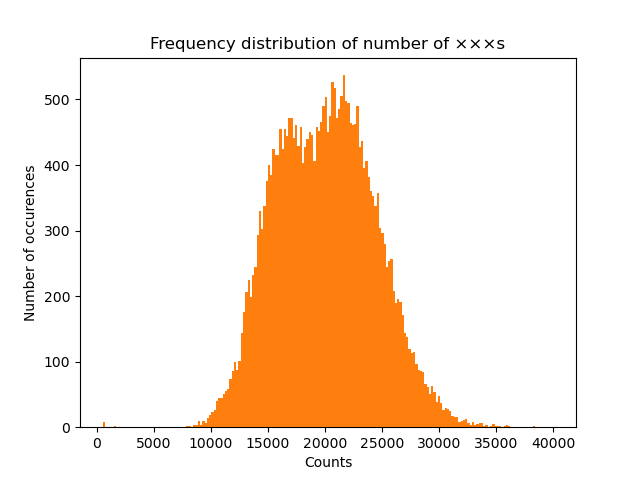
4、画出所有事件能谱：

刚开始我以为能谱类似于信号的能谱，对其做傅里叶变换取模的平方再除以采样时间Ts，如此公式所示：

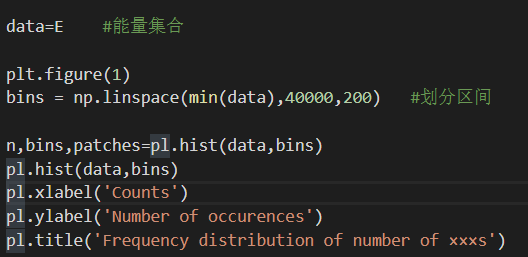
。

读完论文后，发现并非如此，能谱是对所有事件积分后，对其类似于直方图统计，得到的频数图。

因为文件中的数据太大，有5903760个事件，电脑跑完需要很长的时间，所以我只计算了前三万个时间的脉冲积分并得到其能谱。如图所示：

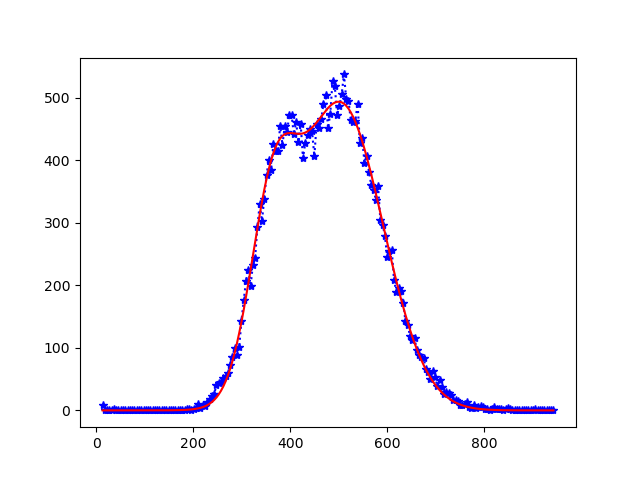


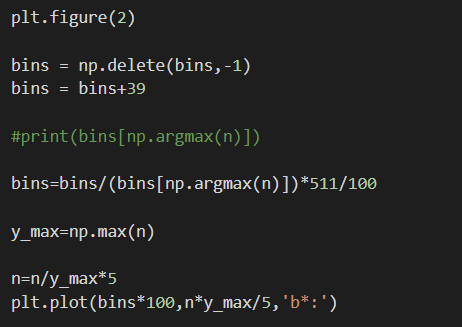
频数统计方式为将脉冲积分值每相差79划分为一段，落在此段内的能量均算为此段能量的中间点所对应的能量并进行统计，利用plt.hist()函数，返回值有每个区间内样值个数n，和区间的左端点bins，并画图。



5、对能谱进行归一化，将最高峰归一到511kev处：

根据上一步获得的能量谱，首先需要先将横坐标整体右移39个单位，因为上一步所得到的横坐标是每个分割段的左端点，我们要的是中间点；之后找出能量谱中的最高峰所对应的横坐标值，将横坐标整体除以此值并乘以511，再以星号虚线的形式描绘出来。如图中蓝色线条所示所示：





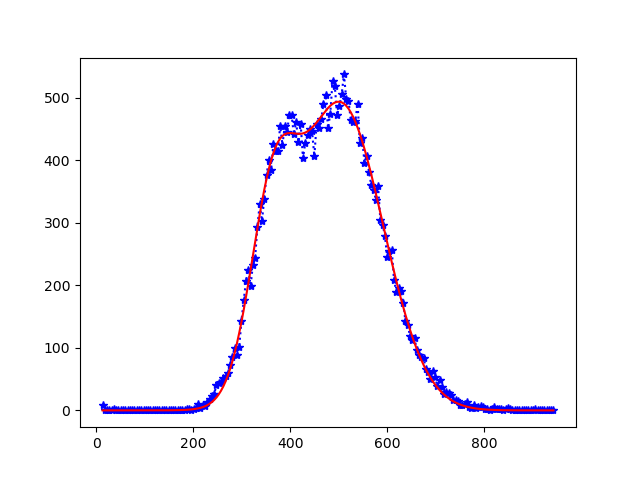
6、对此峰进行高斯拟合：

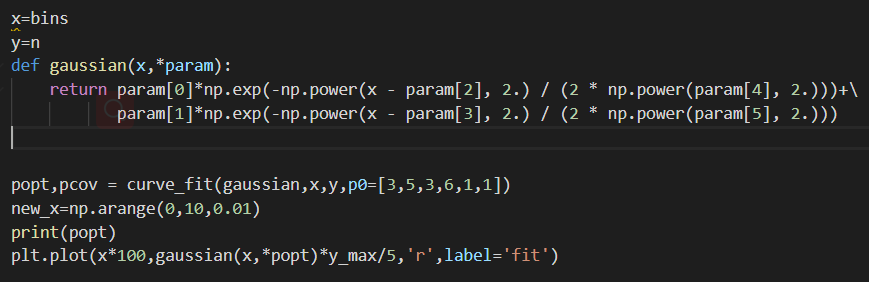
经过上一步归一化能谱后，我们得到了所对应能谱的横纵坐标，并描绘了离散点，通过观察归一化后的能谱，可以明显的看到能谱存在着两个峰，因为要做高斯拟合，所以应当选取双高斯函数进行拟合。

拟合形状公式如下：



通过对参数设定，和坐标相应的放缩并还原得到拟合的能谱如图中红色曲线所示：





7、计算能量分辨率

能量响应是指对入射光子所产生的脉冲能谱分布。能量分辨率则定义为脉冲能谱分布得到半高宽与入射伽马光子的能量比。它表明了PET系统对散射符合计数的甄别能力，PET的能量分辨率主要取决于所用晶体的光产额和光阻止能力及光电倍增管的性能。

通过上一步对能谱进行了高斯拟合，我们得到了对应横纵坐标的数据，因此我们可从纵坐标数据中找出大于能谱半高度(max\_y/2)数据所在的位置(num\_y)，并从横坐标中找出对应位置所占据的能量宽度(W)，最后用能谱半高宽(W)除以能谱最高点对应的能量值(E\_x)即为所求得的能量分辨率，如下图所示：



从结果上看，所求得的能量分辨率远高于文章中的能量分辨率，我猜想原因可能如下：

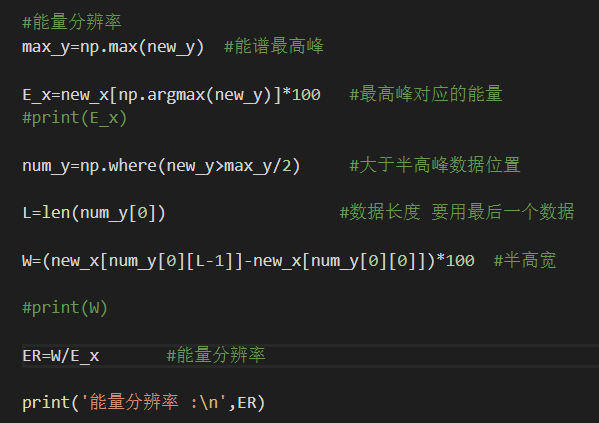
（1）所采用的事件数据太少，文件中有五百万多数据我们只采用了三万个数据；

（2）脉冲拟合效果不好；

（3）所用的积分形式过于简单，存在误差。

（4）采样数据本身原因（可能性微乎其微）

程序如下：



程序：

import struct

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import curve\_fit

import pandas as pd

import math

import random

import csv

import matplotlib

from sympy import \*

import pylab as pl

import matplotlib.mlab as mlab

from scipy.stats import norm

binFile=open('cy.samples','rb')

a=binFile.read()

print(len(a))             #字节数

cycle=int(len(a)/67)+1    #事件数+1

print(cycle-1)

cycle=30000  #前七组数据

#拟合形状 双指数

def func(x, a,b,c,d):

     return  a\*np.exp(-(x-c)/b)\*(1-np.exp(-(x-c)/d))

#电压阈值40 110 180 270 mV

num2 = [0.4,1.1,1.80,2.70,2.70,1.80,1.10,0.4]

num1 = [0.004,0.011,0.018,0.027,0.027,0.018,0.011,0.004]

num = [0.04,0.11,0.18,0.27,0.27,0.18,0.11,0.04]

#num = [40,110,180,270,270,180,110,40]

y = np.array(num)

y1= np.array(num1)

y2= np.array(num2)

#时间取值范围0-150ns

x\_new=np.arange(0,150, 0.02)

#存贮每个事件的八个采样点

squares=[0,0,0,0,0,0,0,0,0,0,0]

#能量

E = []

for i in range(1,cycle,1):                #读一次脉冲数据

    b=a[(i-1)\*67:i\*67]

    c=struct.unpack('<Bhdddddddd', b)

    #print(c)

    for j in range(2,10):

        squares[j]=((c[j]-c[2]))

    #时间采样点

    x = squares[2:10]

    x=np.array(x)

    #print('x is :\n',x)

    #求参数

    popt, pcov = curve\_fit(func, x, y,maxfev=500000)

    #拟合后的曲线

    yvals = func(x\_new,popt[0],popt[1],popt[2],popt[3]) #拟合y值

    #时间间隔20ps

    dx=0.02

    #积分

    F=np.sum(yvals\*dx)\*1000

    print(F)

    #当能量很小 判定拟合失败 换不同放缩坐标重新拟合

    if F<100:

        print(i)

        popt, pcov = curve\_fit(func, x, y1,maxfev=500000)

        yvals = func(x\_new,popt[0],popt[1],popt[2],popt[3])

        F=np.sum(yvals\*dx)\*10000

        print(F)

        print(x)

        if F<100:

            print(i)

            popt, pcov = curve\_fit(func, x, y2,maxfev=500000)

            yvals = func(x\_new,popt[0],popt[1],popt[2],popt[3])

            F=np.sum(yvals\*dx)\*100

            print(F)

            print(x)

    if F>100 and F<50000:

        E.append(int(F))

    #绘图

    #plt.figure(i)

    #plt.scatter(x, y)

    #plt.plot(x\_new, yvals, 'r',label='polyfit values')

    #plt.xlabel('x (ns) ')

    #plt.ylabel('y  (V)')

    #plt.legend(loc=1)

    #plt.title('curve\_fit')

    #plt.show()

n\_x=np.array(E)

data=E    #能量集合

#画能谱

plt.figure(1)

bins = np.linspace(min(data),40000,200)   #划分区间

n,bins,patches=pl.hist(data,bins)

pl.hist(data,bins)

pl.xlabel('Counts')

pl.ylabel('Number of occurences')

pl.title('Frequency distribution of number of ×××s')

#能谱归一化

plt.figure(2)

bins = np.delete(bins,-1)

bins = bins+39

print(bins[np.argmax(n)]) #峰值对应的积

#归一化并放缩100倍

bins=bins/(bins[np.argmax(n)])\*511/100

y\_max=np.max(n)

n=n/y\_max\*5

#绘制归一化能量谱

plt.plot(bins\*100,n\*y\_max/5,'b\*:')

#高斯拟合峰值

x=bins

y=n

def gaussian(x,\*param):

    return param[0]\*np.exp(-np.power(x - param[2], 2.) / (2 \* np.power(param[4], 2.)))+\

           param[1]\*np.exp(-np.power(x - param[3], 2.) / (2 \* np.power(param[5], 2.)))

popt,pcov = curve\_fit(gaussian,x,y,p0=[3,5,3,6,1,1])

new\_x=np.arange(0,10,0.01)

print(popt)

new\_y=gaussian(new\_x,\*popt)\*y\_max/5

plt.plot(new\_x\*100,new\_y,'r',label='fit')

#能量分辨率

max\_y=np.max(new\_y)

E\_x=new\_x[np.argmax(new\_y)]\*100

print(E\_x)

num\_y=np.where(new\_y>max\_y/2)

L=len(num\_y[0])

W=(new\_x[num\_y[0][L-1]]-new\_x[num\_y[0][0]])\*100

print(W)

ER=W/E\_x

print('能量分辨率 :\n',ER)

'''

plt.figure(3)

countDict = dict()

y=[]

for i in np.sort(list(set(E))):

    countDict[i] = E.count(i)

    y.append(countDict[i])

x=np.sort(list(set(E)) )

y=np.array(y)

x=np.array(x)

x=x/(x[np.argmax(y)])\*511

plt.bar(x, y)

#plt.plot(x, y, 'r',label='polyfit values')

'''

plt.show()

