Guru Jambheshwar University of Science and Technology

Department of Physics

To study the XRD pattern of the sample and find it's parameters using Debye Scherrer formula and WH plot

Project of Material Science

Submitted to : Prof. Ashish Agarwal

Sourabh 220070720010

March,2024

X-Ray Diffraction

1.0.1 Introduction

It is technique use to find the structure of the given sample with using diffraction of X-rays with sample in bulk or powder form. It used for finding the crystal-lite size, inter-planer spacing, micro-strain and index of different plane of the given sample. Furthermore, it uses the diffraction properties of radiator in order to do this. And we use the X-ray because they have wavelength equal to the different lattice parameters.

1.0.2 Instrumentation : X-Ray Diffractometer

X-Ray Diffractometer is uses to find the diffraction pattern of a sample. Where we place the sample and then the start the Diffractometer and it record the X-rays diffracted from the sample with change in angle of falling on sample. And its construction and working is given below:

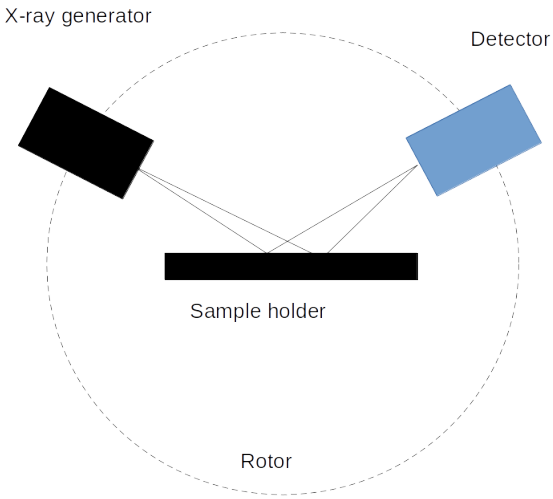
Construction:

All the parts of Diffractometer are given below:

- **X-Ray Source:** It is source of the X-rays that used to study the lattice parameters. It is a vacuum tube in which the electrons are generated by heating the tungsten filament, then are accelerated by using a high voltage and when it falls on the anode made up of Copper (Cu) (mainly) that generate X-rays by knocking of the K shell electron and, then L shell electron fill the release a photon of X-ray and then a filter is used to filter the X-ray and make monochromatic X-ray.
- **Sample Holder :** There are many types of sample holders based upon the sample and the reactivity of sample. And some of them cause the background interference and some them isolate the sample.

- **X-ray detector :** There are two type of X-ray detector can be used in the X-Ray Diffractometer these are Gas-filled Detector and Solid State Detector (Scintillation Detector and Silicon Strip Detector). And mostly Gas-filled Detector which utilized the ionizing power of X-rays and then collect the ions.
- **Rotor :** It is servo step motor that X-ray source and detector to change the angle of incident X-rays , and it change both the positions in exact way.

Diagram

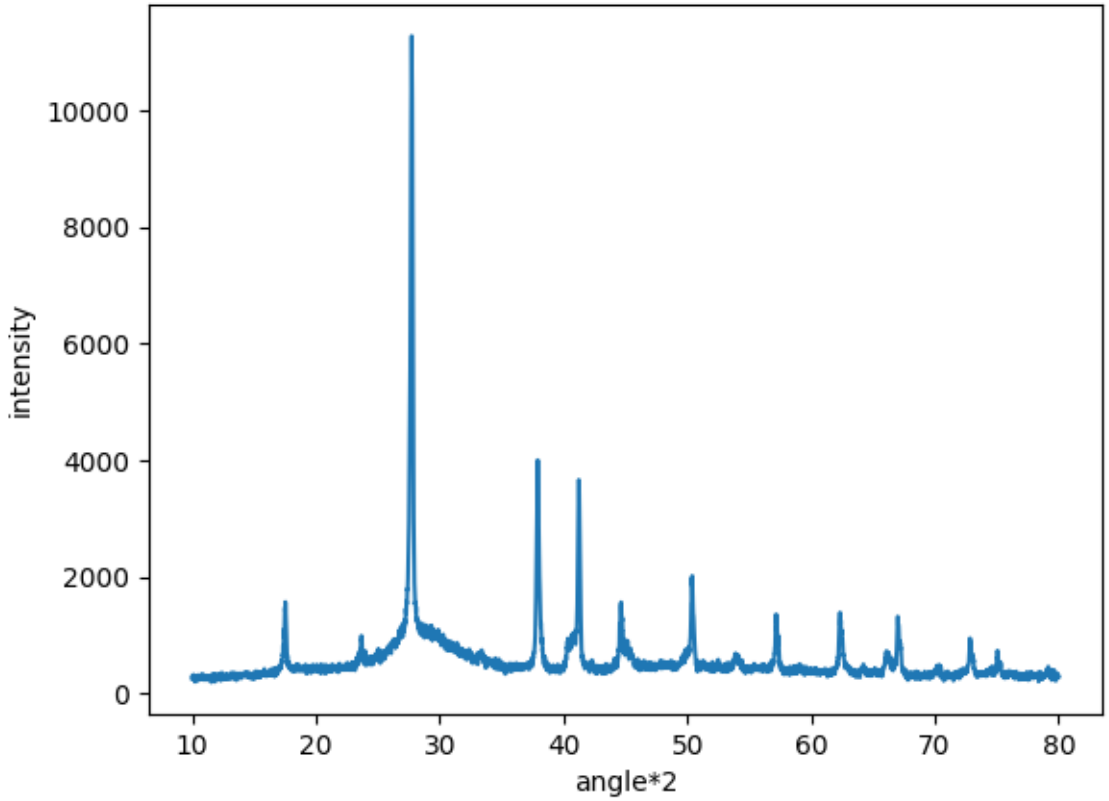


Working

First system is calibrated and stand on 10^o , then the filament is heated then it emits the thermal electrons, then they are accelerated by the high voltage and they collide with copper anode, then produce the x-rays then these rays are filtered by krypton filter. After that they fall on the sample and get diffracted then detector detected and record them into recorder and then angle is changed and repeat this process until angle reach to 80^o.

1.0.3 XRD Pattern

It is a graph between the angle and intensity of X-rays detected by detector. In this there are peak at specific angle for a material due to Bragg's Diffraction theory. And it shows as show in the below :



And this is pattern of the given sample which we are going to study

Debye Scherrer Theory

2.0.1 Introduction

As we see that in Bragg's law the all peaks are should be a delta peak function, but as we see in the real XRD pattern of a sample the peaks are like Gaussian or Cauchy Lorentz peak function. And Bragg's law doesn't explain this because it takes the crystal as perfect lattice with infinite dimension for simplicity and due to this Bragg's law can't explain it, and we use the Debye Scherrer Theory to explains this behavior of the crystalline materials.

2.0.2 Statement of Debye Scherrer Theory

The crystalline materials are form of crystallites, and these crystallites are finite in size and due to this at the boundary of these crystallites the inter-planner distance is more or less than it is inside the crystallite and due to this the peak is broadening is happening. And peaks change for delta peaks functions to Gaussian peak functions.
And we can find the average crystallite size using broadening of the XRD peaks.

2.0.3 Debye Scherrer Formula

$$\boldsymbol{D = \frac{K\lambda}{\beta cos(\theta)}} \tag{2.1}$$

Here , $\boldsymbol{D}$ is the average crystallite size,
$\boldsymbol{K}$ is a shape factor, that depend upon the shape of the crystallite,
$\boldsymbol{\lambda}$ is the wavelength of the X-ray we used in the XRD,
$\boldsymbol{\beta}$ is full width at half maxima of the XRD peak,
$\boldsymbol{\theta}$ is the angle at which the diffraction is happens.

2.0.4 Shape Factor

$\boldsymbol{K}$ is factor that taking account of the shape of crystallite because as we see in Debye Scherrer theory the broadening of the XRD peaks is due the different inter-planner spacing at the surface of crystallite, and the spacing is depend upon the shape of the crystallite.
And the value for the shape factor can be different for different shapes and some of them are listed bellow:

- Spherical : $\boldsymbol{K \sim 0.9}$
- Rod shaped : $\boldsymbol{K \sim 1.0 - 1.3}$
- Plate shaped : $\boldsymbol{K \sim 0.7 - 0.8}$

And we mostly take $\boldsymbol{K = 0.9}$, because it workers for most of the materials.

Williamson-Hall Plot

3.0.1 Introduction

In Debye Scherrer Theory, we assume that the broadening is happening due to only crystallite size, and they don't taking account of other factors like micro-strain in the lattice that also cause the broadening in XRD peaks. Then Williamson and Hall come with there theory where they taking account of broadening that happens due to micro-strain and form that they find the average crystallite size and the micro-strain in the lattice.

3.0.2 Williamson-Hall Theory

It states that the total broadening in the XRD peak is sum of the broadening due to crystallite size and due to micro-strain in the lattice.

$$\boldsymbol{\beta_{total} = \beta_{crystallite} + \beta_{microstrain}} \tag{3.1}$$

3.0.3 Micro-Strain

We know that in real life lattices are not perfect, but they are having distortions and defects, and these cause the strain in lattice and the inter-planner distance is similar for similar planes. And that cause a broadening in the XRD peaks.
The formula for micro-strain as function of broadening is given below:

$$\boldsymbol{\epsilon = \frac{\beta}{4} cot(\theta)} \tag{3.2}$$

where, $\boldsymbol{\epsilon}$ is the micro-strain.

3.0.4 Williamson-Hall Equation

By using the equations **(2.1)**, **(3.1)** and **(3.2)** we get the equation:

$$\boldsymbol{\beta_{total} cos(\theta) = 4 sin(\theta) + \frac{K\lambda}{D}} \tag{3.3}$$

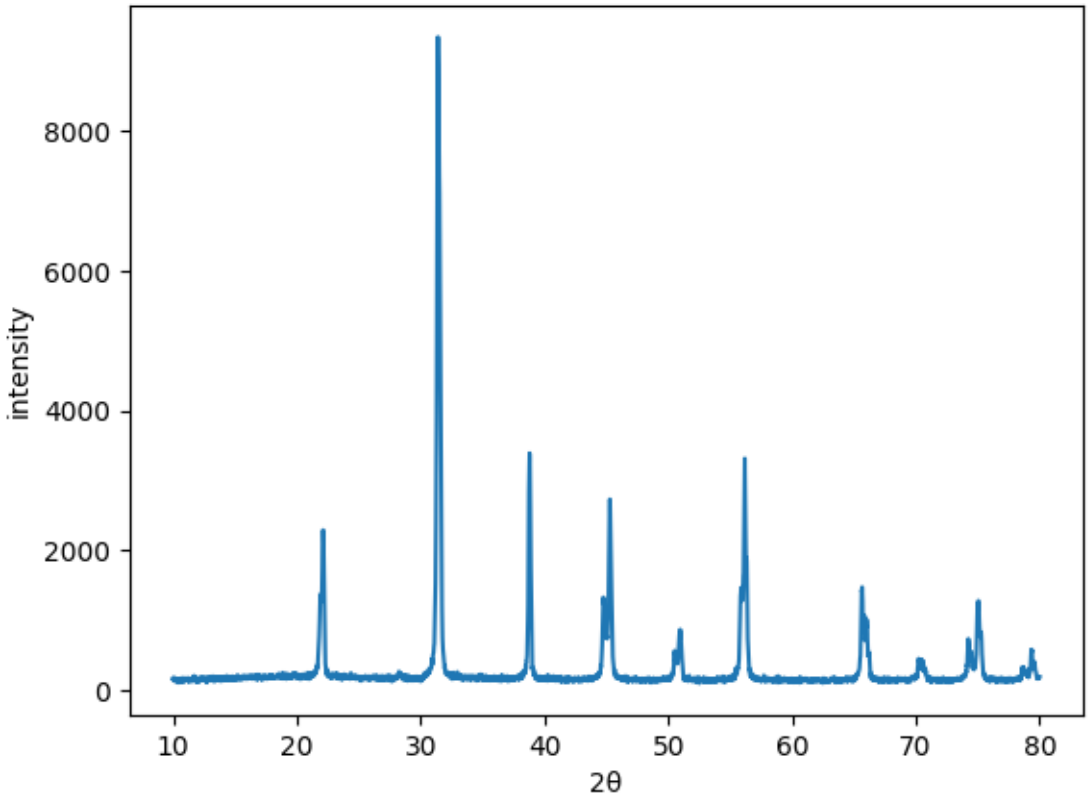
where, $\boldsymbol{\beta_{total}}$ is fwhm of XRD peak at angle $\boldsymbol{\theta}$

3.0.5 WH Plot

It is a plot of the equation **(3.3)** in which we use plot the graph between $\boldsymbol{\beta_{total} cos(\theta)}$ and $\boldsymbol{4 sin(\theta)}$ and it is straight line.
By fitting this line we can get the micro-strain and the crystallite size from the slope and the intercept.

ABCD Data Analysis

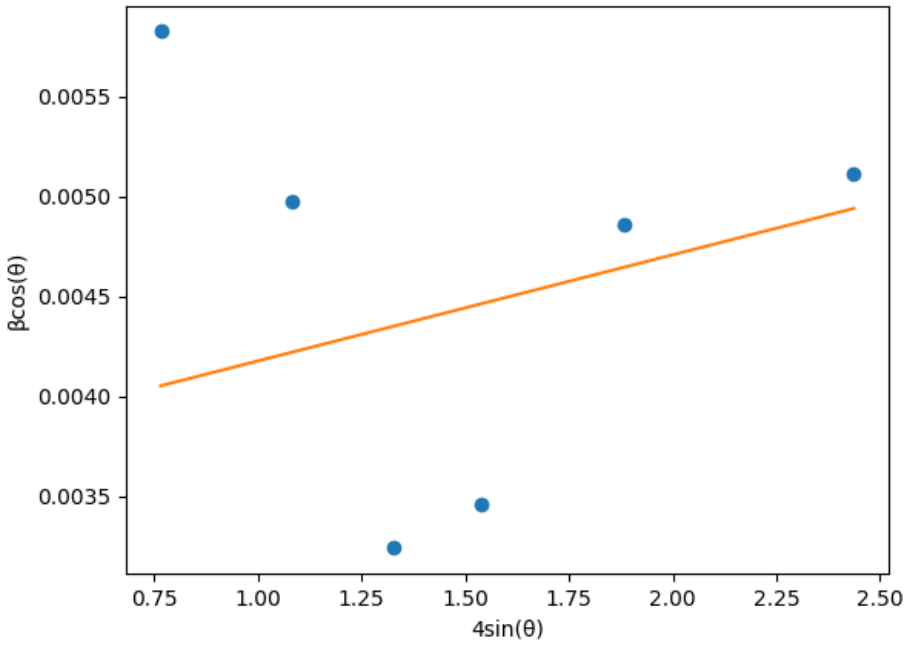
4.0.1 XRD Plot



4.0.2 Extracted Data

2θ	$\beta\ (fwhm)$	$crystallite\ size\ (nm)$	$4sin(\theta)$	$\beta cos(\theta)$
22.0636482	0.00593724	23.79294294	0.765416785	0.005827526
31.43689615	0.005168592	27.86841332	1.083641596	0.00497531
38.79930557	0.003435427	42.78943684	1.328621664	0.003240379
45.28855593	0.003745554	40.11026577	1.540034645	0.003456821
56.17597669	0.005513616	28.50468387	1.883307757	0.004864253
75.07582497	0.006452684	27.09854656	2.437145023	0.005116658

4.0.3 WH Plot



4.0.4 Results

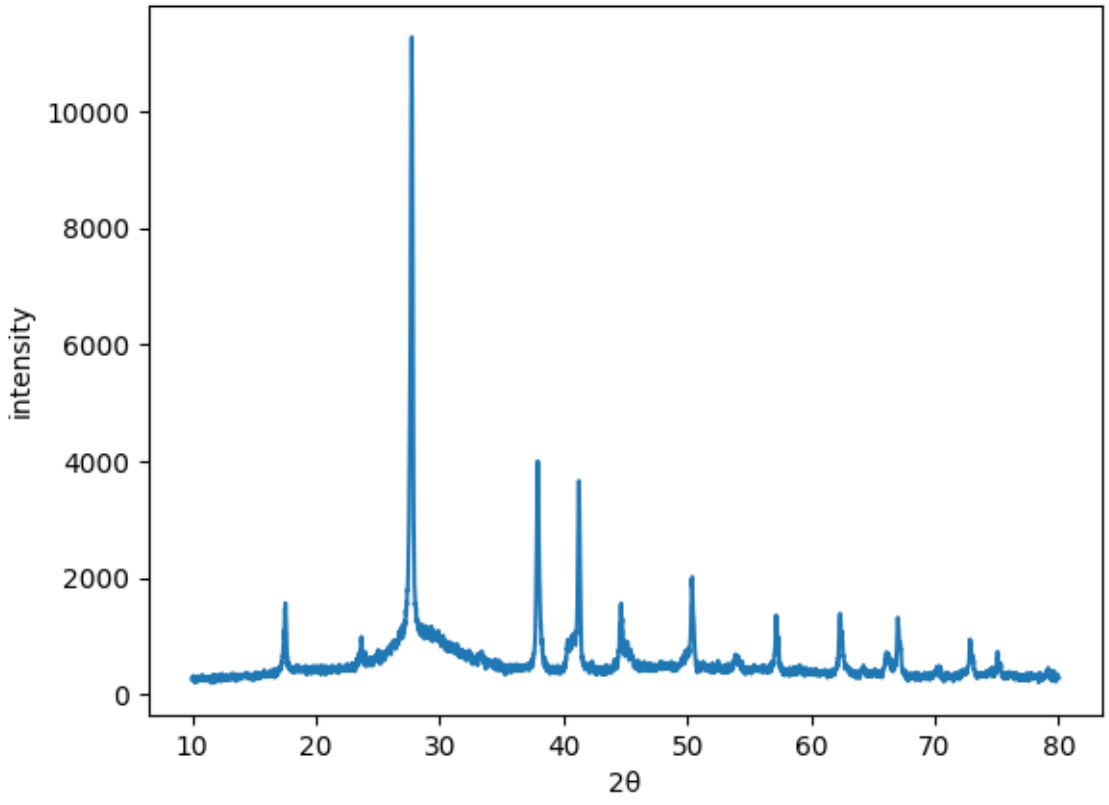
Crystallite size by debye scherrer formula is **31.694048217604717** nm
Crystallite size by wh plot is **38.03285786114954** nm
Micro strain is **0.000531528851930141**

4.0.5 Indexing of XRD Data

2θ	$sin^2(\theta)/sin^2(\theta_{min})$	$h^2+k^2+l^2$	hkl	$LatticeParameter$
22.12	1	1	100	0.086326582
31.4	1.989724535	2	110	0.172209069
38.8	2.998013352	3	111	0.258893981
45.28	4.026442133	4	200	0.346445776
50.62	4.966380297	5	210	0.430179324
50.96	5.028850131	5	201	0.432876384
56.18	6.024426184	6	211	0.519012734
56.3	6.048083213	6	211	0.520030778
65.66	7.986786381	8	220	0.690042074
65.84	8.025703183	8	220	0.691721198
66.02	8.064674868	8	220	0.693398614
74.24	9.896254532	10	310	0.858776146
74.54	9.964769032	10	310	0.86174379
75.04	10.07917958	10	310	0.866676729
75.26	10.12960535	10	310	0.868842001
79.36	11.0778802	11	311	0.952948035
				0.578132701

SZBT-2 Data Analysis

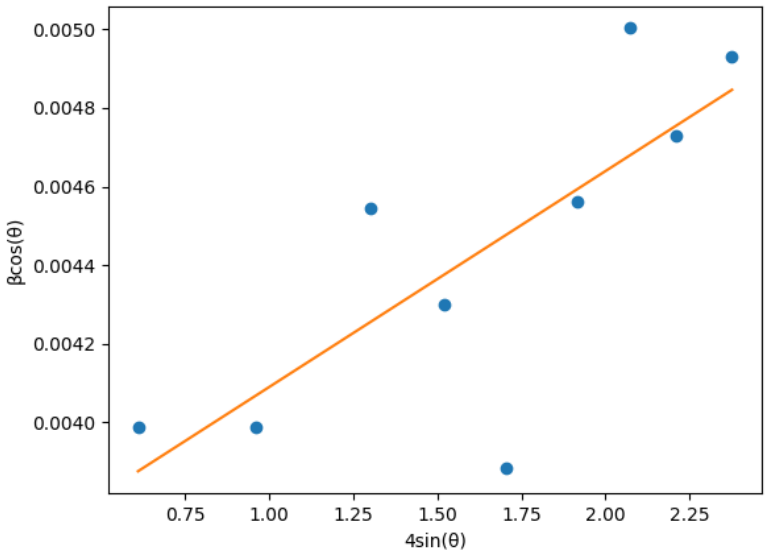
5.0.1 XRD Plot



5.0.2 Extracted Data

2θ	$\beta\ (fwhm)$	$crystallite\ size\ (nm)$	$4sin(\theta)$	$\beta cos(\theta)$
17.51900534	0.004034379	34.77370193	0.609149227	0.003987324
27.73102359	0.004107899	34.76607033	0.958574772	0.003988199
37.93257644	0.004851558	30.21991417	1.30004709	0.004588167
44.66163201	0.004646945	32.25674623	1.51981488	0.00429845
50.3989391	0.004291504	35.70717065	1.703083658	0.003883086
57.22018857	0.005246376	30.10430735	1.915386129	0.004605786
62.38317709	0.005849039	27.71138395	2.071605719	0.005003503
67.05035038	0.005674085	29.31273675	2.209213332	0.004730162
72.89582687	0.006128271	28.12703054	2.376367077	0.004929564

5.0.3 WH Plot



5.0.4 Results

Crystallite size by debye scherrer formula is **31.50783976576545** nm
Crystallite size by wh plot is **39.17098509561332** nm
Micro strain is **0.0005498292167665724**

5.0.5 Indexing of XRD Data

2θ	$sin^2(\theta)/sin^2(\theta_{min})$	$2(sin^2(\theta)/sin^2(\theta_{min}))$	$h^2+k^2+l^2$	hkl	$LatticeP$
17.54	1	2	2	110	0.7144882
27.74	2.471997552	4.943995104	5	210	0.7185236
37.92	4.541101195	9.082202391	9	221/300	0.7112475
41.24	5.335082383	10.67016477	11	311/221	0.7254473
44.66	6.209715634	12.41943127	12	222	0.7023197
50.4	7.798471644	15.59694329	16	400	0.7236613
57.18	9.850865776	19.70173155	20	420	0.7198763
67	13.10446881	26.20893762	26	431	0.7116346
					0.7158998

Source Code

This data is analyzed by a python program that written by me named xanalyser. And it a part of my big project not given code name yet and you can get the project on `github.com/sourabh945`

```
from scipy.optimize import curve_fit
import numpy as np
from math import sqrt,pi,cos,sin
from matplotlib import pyplot as plt
from csv import writer as wr
import gi
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk, Gdk, GObject, GLib

def load_data(filename:str) -> list[list,list]:
    x_return , y_return = [], []
    with open(filename,"r") as file:
        file_content = file.readlines()
    file.close()
    for line in file_content:
        if line[0].isnumeric() is True:
            x_return.append(float(line.split("\t")[0]))
            y_return.append(float(line.split("\t")[1]))
    return [x_return ,y_return]

def peak_separator(data:tuple[list,list],range:tuple[float,float]) -> list[list,list]:
    left_index = np.searchsorted(data[0],range[0],side='left')
    right_index = np.searchsorted(data[0],range[1],side='right')
    return data[0][left_index:right_index], data[1][left_index:right_index]

def gaussian(x,*args) -> any:
    """
    Parameter:
    ----------
    x: x value for the gaussian function
```

```
    data: list contain the tuples with tuples are as (y0,
    amplitude,mean,sigma)
    """
    sum = (args[0]) + (args[1]/(args[3]*sqrt(2*pi)))*(np.exp
    ((-1/2)*(((x-args[2])/args[3])**2)))
    for i in range(1,int(len(args)/4)):
        y0 = float(args[4*i])
        amplitude = float(args[4*i+1])
        mean = float(args[4*i+2])
        sigma = float(args[4*i+3])
        sum = sum  + (y0 + (amplitude/(sigma*sqrt(2*pi)))*(np.
    exp((-1/2)*(((x-mean)/sigma)**2))))
    return sum

def fit_curve(data:tuple[list,list]) -> tuple[float,float,float,
    float]:
    _angle,_intensity = data
    mean = _angle[np.argmax(_intensity)]
    y0 = min(_intensity)
    a = max(_intensity)
    sigma = min(mean - _angle[0],_angle[-1] - mean)
    popt,pcov = curve_fit(gaussian,_angle,_intensity,p0=[y0,a,
    mean,sigma],bounds=[(0,0,0,0),(np.inf,np.inf,np.inf,5)])
    return tuple(popt)

def data_exporter(data:list[tuple[float,float,float,float]],
    filename:str,header:list[str]) -> None:
    with open(filename,"w") as file:
        writer = wr(file)
        writer.writerow(header)
        for i in data:
            writer.writerow(i)
    file.close()

def crystallite_size_by_debye_seherrer(fitted_data) -> float:
    sum = 0
    for i in fitted_data:
        beta = 2.35*i[-1]*(pi/180)
        sum = sum + 0.9*0.15406/(beta*cos((pi/180)*(i[2]/2)))
    return sum/len(fitted_data)

def whplot(fitted_data) -> tuple[list,list]:
    x,y = [], []
    for i in fitted_data:
        beta = 2.35*i[-1]*(pi/180)
        x.append(4*sin((pi/180)*(i[2]/2)))
        y.append(beta*cos((pi/180)*(i[2]/2)))
    return x,y
```

```
74 def st_line(x,m:float,c:float) -> any:
       return m*x + c
76
   def fit_whplot(x,y,csfds:float) -> tuple[float,float]:
78     csfds = 0.9*0.15406/csfds
       popt,pcov = curve_fit(st_line,x,y,p0=[0.001,csfds/1.3],
       bounds=((0,csfds/1.4),(np.inf,csfds/1.2)))
80     return popt

82 def cal_crystallite_size_and_microstrain(m:float,c:float) ->
       tuple[float,float]:
       return 0.9*0.15406/c , m
84
   def indexing_helper(filename,data):
86     _x = []
       _y = []
88     for i in data:
           _x.append(sin(i[2]*pi/360))
90     minimum = min(_x)
       for i in range(len(_x)):
92         _y.append(_x[i]/minimum)
       with open(filename,"w") as file:
94         writer = wr(file)
           for i in range(len(_x)):
96             writer.writerow([_x[i],_y[i]])

98 if __name__ == "__main__":
       angle , intensity = load_data("SZBT-2.TXT")
100    number_of_peaks = int(input("Enter the number of peaks : "))
       result = []
102    for i in range(number_of_peaks):
           left = float(input(f"Enter the left index of peak {i+1}
       :: "))
104        right = float(input(f'Enter the right index of peak {i
       +1} :: '))
           result.append(fit_curve(tuple(peak_separator((angle,
       intensity),(left,right)))))
106    data_exporter(result,"fitted data.csv",["y0",'a','$2\theta$'
       ,'sigma'])
       csbds = (crystallite_size_by_debye_seherrer(result))
108    print(csbds)
       _x , _y = whplot(result)
110    _m , _c = fit_whplot(_x ,_y,csbds)
       csbwhp , micro_strain = cal_crystallite_size_and_microstrain
       (_m,_c)
112    plt.plot(_x,_y,'o')
       __y = []
114    for j in _x:
           __y.append(st_line(j,_m,_c))
```

```
    plt.plot(_x,__y)
    plt.xlabel("4sin($\theta$)")
    plt.ylabel("$\beta$cos($\theta$)")
    plt.show()
    with open("result.txt","w") as file:
        file.writelines([f'crystallite size by debye scherrer formula is {csbds}\n',f'crystallite size by wh plot is {csbwhp}\n',f'Micro strain is {micro_strain}\n'])
    print(csbwhp , micro_strain)
    indexing_helper("SZBT-2_indexing.csv",result)
```

xanalyser.py

And the code for indexing is also made by self , and it has written in python.

```
import numpy as np
from csv import writer as wr
from math import sin , radians

def load_data(filename:str) -> list[list,list]:
    x_return , y_return = [], []
    with open(filename,"r") as file:
        file_content = file.readlines()
    file.close()
    for line in file_content:
        if line[0].isnumeric() is True:
            x_return.append(float(line.split("\t")[0]))
            y_return.append(float(line.split("\t")[1]))
    return [x_return ,y_return]

def all_peak_separator(data,tols:tuple[float,float]) -> list[tuple[list,list]]:
    """
    Parameter
    ---------
    tols: ( min_height , base_line )
    min_height : is the minimum height of the peak you take
    base_line : it the base level of the signal
    """
    peak__ = []
    min_height , base_line = tols
    result__angle= []
    result__intensity = []
    peak_is_on = False
    for i in range(0,len(data[1])):
        if data[1][i] > base_line:
            result__angle.append(data[0][i])
            result__intensity.append(data[1][i])
            peak_is_on = True
        if data[1][i] <= base_line and peak_is_on:
```

```
            if result__intensity[np.argmax(np.array(
    result__intensity))] > min_height:
                peak__.append((np.array(result__angle),np.array(
    result__intensity)))
            result__angle = []
            result__intensity = []
            peak_is_on = False
    return peak__

def peaks_in_interval(data,tols:tuple[float,float]) -> list[
    float]:
    _angle,_intensity = data
    min_heigh, width = tols
    peak_is_on = False
    min_height = min_heigh*max(_intensity)/100
    result = []
    for i in range(0,len(_angle)):
        if i + width == len(_angle):
            break
        elif _intensity[i] > min_height:
            if _intensity[i] > _intensity[i-1] and peak_is_on is
     False:
                peak_is_on = True
            elif _intensity[i] < _intensity[i-1] and peak_is_on
    and _intensity[i] > max(_intensity[i+1:i+width]):
                result.append(_angle[i-1])
                peak_is_on = False
    return result

def indexing(filename:str,peaks_data:list) -> None:
    min_angle = peaks_data[np.argmin(peaks_data)]
    min_angle = sin(radians(min_angle/2))**2
    with open(filename,"w") as file:
        writer = wr(file)
        for i in peaks_data:
            value = sin(radians(i/2))**2/min_angle
            writer.writerow([i,value,2*value,3*value,4*value])

if __name__ == "__main__":
    angle,intensity = load_data("ABCD.TXT")
    peaks = all_peak_separator([angle,intensity],(500,300))
    peaks_angle = []
    for j in peaks:
        for k in peaks_in_interval(j,(50,5)):
            peaks_angle.append(k)
    print(peaks_angle)
    indexing("ABCD-fitting.csv",peaks_angle)
```

indexing.py