Analysis of X-ray Diffraction Patterns with Python

CME 502: MATHEMATICAL ANALYSIS & MODELING

Wenbo Du

Department of Materials Science and Chemical Engineering, Stony Brook University

May 2, 2023

Introduction

1. Benchtop X-ray diffraction(XRD) techniques are extensively used in measuring structural properties of crystalline powders and sintered block samples.
2. In each scanning pattern, 5 to 9 main peaks and several minor peaks are made of 7000 to 10000 steps of scanning.
3. From a series of scanning, the trend for crystal transformation under certain situation can be detected.

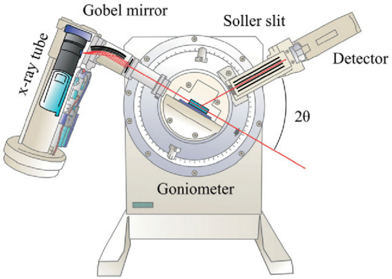


Figure 1: Schematic Benchtop X-ray Diffractometer

Introduction

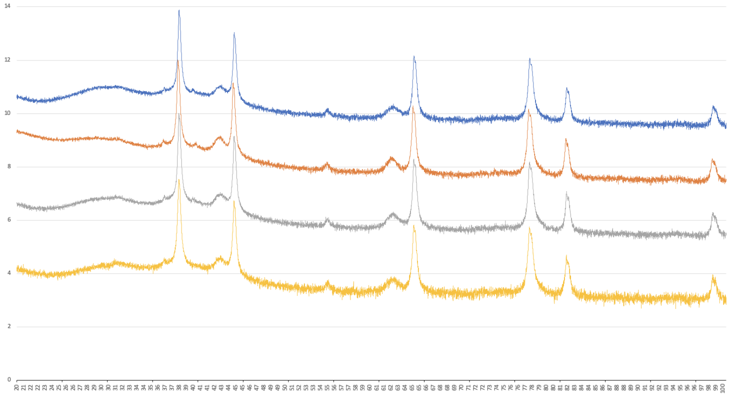


Figure 2: XRD Scanning Patterns for as-sintered Al-nano-alloy samples

Introduction

Structural properties can be calculated by scanning pattern analysis:

1. Lattice parameter is related to peak poistion by Bragg's Law
2. Crystalline size can be calculated by Scherrer Equation with peak broadening β or full witdh at half maximum
3. Mean inhomogeneous strain ϵ to the peak broadening β's approx. relation can also derived from Bragg's Law

Goal

1. Main Goals
    - Apply background subtraction for patterns of as-sintered sample's top surface
    - Fit peaks of main phase with Gaussian and Lorentz distribution, and extract the parameters from the best fits
    - Calculate the structrual properties with the extracted parameters
    - Make Williamson-Hall Plot based on the parameters.
2. Extra Goals
    - Apply the analysis on several patterns of other sections at the as-sintered samples and conclude the relation between the structural properties and section positions

Background Subtraction

1. Import data from .csv files of background scanning and as-sintered sample scanning with Pandas Read CSV
2. As-sintered pattern's data points minus data points from background pattern
3. plot the subtracted patterns with functions from matplotlib, and find out the best fitted one

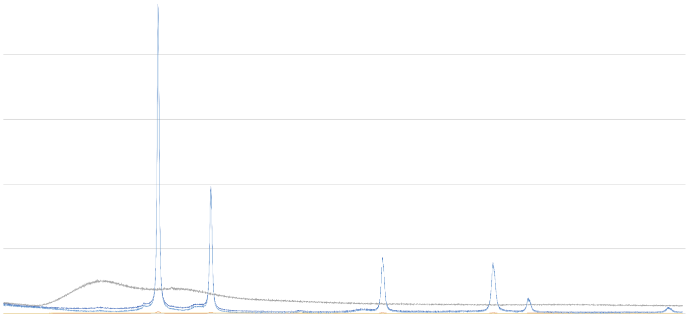


Figure 3: As-sintered scanning pattern subtracted by 0.1 times background pattern

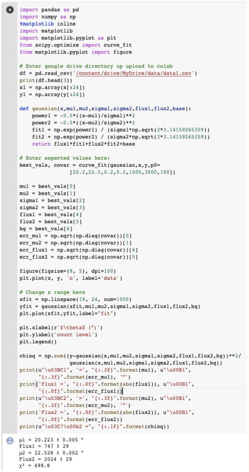
Curve fitting

1. Build functions for Gaussian and Lorentz distribution with Numpy and Math.

$$I_{Gaussian}(2\theta) = I_{max} e^{-\pi(\frac{2\theta - 2\theta_0}{\beta})^2}$$

$$I_{Lorentzian} = \frac{FWHM^2}{FWHM^2 + (2\theta - 2\theta_0)^2}$$

2. fit the subtracted pattern with both distributions by functions, such as "curve_fit()", from library scipy.optimize
3. Calculate R square and Chi square to find out the best fit



```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from matplotlib.pyplot import figure

# Enter google drive directory up upload to colab
df = pd.read_csv('/content/drive/MyDrive/data/data1.csv')
print(df.head(3))
x1 = np.array(x[:24])
y1 = np.array(y[:24])

def gaussian(x,mu1,mu2,sigma1,sigma2,flux1,flux2,base):
    power1 = -0.5*((x-mu1)/sigma1)**2
    power2 = -0.5*((x-mu2)/sigma2)**2
    fit1 = np.exp(power1) / (sigma1*np.sqrt(2*3.14159265359))
    fit2 = np.exp(power2) / (sigma2*np.sqrt(2*3.14159265359))
    return flux1*fit1+flux2*fit2+base

# Enter expected values here:
best_vals, covar = curve_fit(gaussian,x,y,p0=
                [20.2,22.5,0.2,0.2,1000,3000,100])

mu1 = best_vals[0]
mu2 = best_vals[1]
sigma1 = best_vals[2]
sigma2 = best_vals[3]
flux1 = best_vals[4]
flux2 = best_vals[5]
bg = best_vals[6]
err_mu1 = np.sqrt(np.diag(covar))[0]
err_mu2 = np.sqrt(np.diag(covar))[1]
err_flux1 = np.sqrt(np.diag(covar))[4]
err_flux2 = np.sqrt(np.diag(covar))[5]

figure(figsize=(8, 5), dpi=100)
plt.plot(x, y, 'o', label='data')

# Change x range here
xfit = np.linspace(19, 24, num=1000)
yfit = gaussian(xfit,mu1,mu2,sigma1,sigma2,flux1,flux2,bg)
plt.plot(xfit,yfit,label="fit")

plt.xlabel(r'$\theta$ (°)')
plt.ylabel('count level')
plt.legend()

chisq = np.sum((y-gaussian(x,mu1,mu2,sigma1,sigma2,flux1,flux2,bg))**2/
                gaussian(x,mu1,mu2,sigma1,sigma2,flux1,flux2,bg))
print(u"\u03BC1", '=', "{:.3f}".format(mu1), u"\u00B1",
      "{:.3f}".format(err_mu1), '°')
print('Flux1 =', "{:.0f}".format(abs(flux1)), u"\u00B1",
      "{:.0f}".format(err_flux1))
print(u"\u03BC2", '=', "{:.3f}".format(mu2), u"\u00B1",
      "{:.3f}".format(err_mu2), '°')
print('Flux2 =', "{:.0f}".format(abs(flux2)), u"\u00B1",
      "{:.0f}".format(err_flux2))
print(u"\u03C7\u00b2 =", "{:.1f}".format(chisq))
```

```
μ1 = 20.223 ± 0.005 °
Flux1 = 747 ± 29
μ2 = 22.528 ± 0.002 °
Flux2 = 2024 ± 29
χ² = 498.8
```

Curve fitting

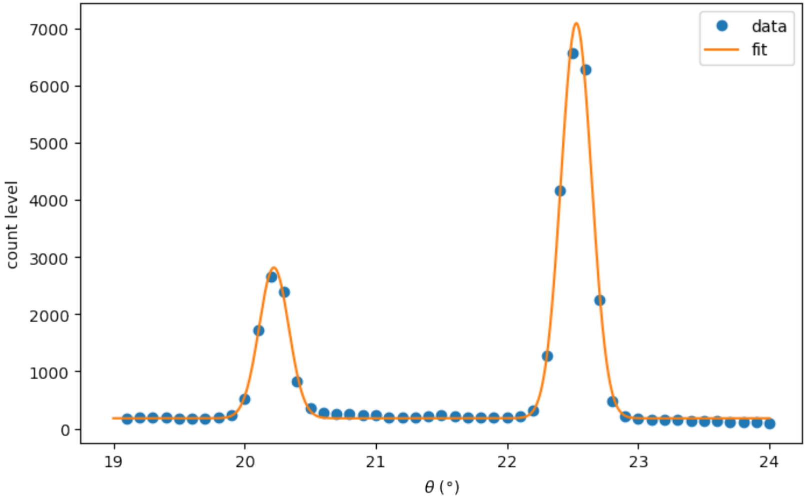


Figure 4: Curve fit for a rough scanning pattern of LiF crystal

Equations for finding Structural Properties

1. Bragg's Law (Lattice Parameter)

$$n\lambda = 2dsin\theta$$

2. Scherrer Equation (crystalline size)

$$L = \frac{K\lambda}{\beta_L * cos\theta}$$

3. Inhomogeneous strain

$$\beta_e = C\epsilon tan\theta$$

Williamson-Hall Plot

Approximate formulae for size broadening β_L, and strain broadening β_e

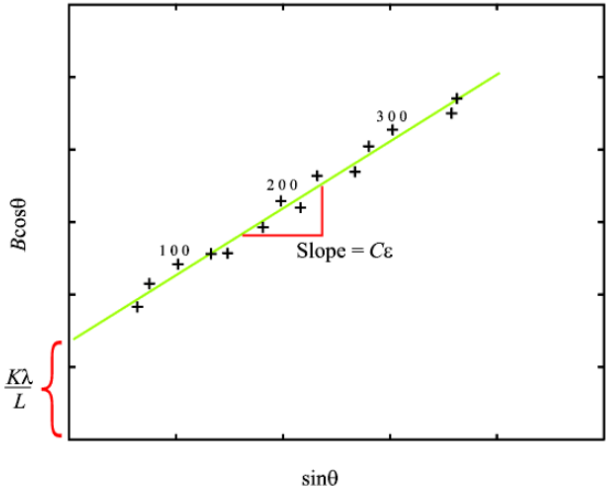
$$\beta_L = \frac{K\lambda}{L cos\theta} \ and \ \beta_e = C\epsilon tan\theta$$

can be rewritten as

$$\beta_{L\&e} cos\theta = \beta_{total} cos\theta = C\epsilon sin\theta + \frac{K\lambda}{L}$$

where and the integral breadth $\beta = 0.5 * FWHM\sqrt{\frac{\pi}{log_e 2}}$ for gaussian distribution, $\beta = \frac{\pi FWHM}{2}$ for Lorentz distribution. Hence, crystalline size and microstrain can be calculated via above equation. And the figure of $\beta_{total} cos\theta$ versus $sin\theta$ is known as Williamson-hall plot.

Williamson-Hall Plot



Parameters form fitted curves will be used to plot this figure with funcitions from matplotlib. From the plotted figure, microstrain and crystalline size can be calculated from the slope and Y intercept.

Results and Discussions

1. To check the accuracy, correlation and goodness of fitted curves, value of r square, Pearson correlation coefficient and Chi square will be calculated.
2. To find out the accuracy of the strcutural properities, can be compared with the analysis results from Topas or Maud.