TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

PROJEKTARBEIT

Festkörperphysik

ausgeführt am Institut für Festkörperphysik
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Univ.Prof. Dipl.-Ing. Dr.techn. Ernst Bauer
Projektass. Dipl.-Ing. Alexander Riss

durch

Nikolas Reumann

01425646

June 15, 2020

Abstract

This work presents a program and its graphical interace which can be used in the analysis of x-ray powder diffraction pattern. For identification of crystal structures it implements various algorithms based on comparison with a database. The usage of the interface is explained and the specific implementation of all algorithms is described.

Abstrakt

Diese Arbeit stellt ein Programm und seine grafische Oberfläche vor, welches bei der Analyse von Röntgenpulverdiffraktogrammen verwendet werden kann. Zur Identifizierung von Kristallstrukturen implementiert es verschiedene Algorithmen, die auf dem Vergleich mit einer Datenbank basieren. Die Verwendung der Schnittstelle wird erklärt und die spezifische Implementierung aller Algorithmen beschrieben.

Summary

The program presented in this work provides the possibility of interactive analysis of x-ray powder diffraction patterns. It consists of a graphical interface which utilizes gnuplot visualization of the data. Through the use of various algorithms a step by step analysis is possible. Because experimental data usually contains high amounts of background noise a few algorithms for removing them have been implemented. This noise reduction is either accomplished by application of a kernel density estimation or by linear approximation of background noise. After formatting and noise reduction is applied there exist two main features for crystal structure identification. The 'Peak Calculations' feature identifies peaks by implementing various peak detection algorithms and compares those peaks with a database of theoretically calculated x-ray patterns. The 'Graph*Database' function calculates another measure of identification by calculating the overlap of the data and each pattern in the database. This is accomplished by simple multiplication of each value after formatting properly. These features then determine the crystal structures most likely present in the given sample. While these approximations can not determine the exact composition they can provide help in manual analysis.

Contents

1 X-Ray Powder Diffraction

X-ray powder diffraction (XRD)[1][2] is a technique used for analysing the crystal structures and atomic spacing through the use of x-rays.

1.1 Theoretical Basics

XRD is based on constructive interference of monochromatic x-rays when diffracted in certain angles depending on the crystal structure. For visual reference refer to figure 1.1.
Contructive interference occurs in periodic lattices and is described by Bragg's law, which relates the wavelength to the diffraction angle and the lattice spacing. Bragg's law is given by

$$n\lambda = 2dsin\theta, \tag{1.1}$$

where n is a positive integer, λ the wavelength, d the gap between lattice planes and 2θ the angle between incoming and outgoing wave-vector.(see figure 1.1)

From the data received by measuring both angle and intensity of all outgoing waves from the sample, an intensity-angle diagram can be created, where each peak corresponds to a constructive interference. This diagram can then be further analyzed. The program presented here aims to help with this analysis.

The analysis of samples is based on the fact that different crystal structures have distinct peaks in their diffraction pattern which can be compared with theoretically calculated values.

Instead of single macroscopic crystals a powder is used as a sample because it contains crystals with randomly distributed orientations and therefore all possible diffraction directions can be measured simultaneously. Another problem with single crystals is the fact that only lattice planes parallel to the surface lead to proper Bragg reflection. In materials with macroscopic ordering of lattice orientation this can lead to incomplete capture of all existing Bragg reflections.

Ideal Bragg diffraction would theoretically result in infinitely sharp peaks, but due to both defects in the sample and imperfections in the experiment, the resulting peaks widen. The measured lattice parameters themselves also are intrinsically

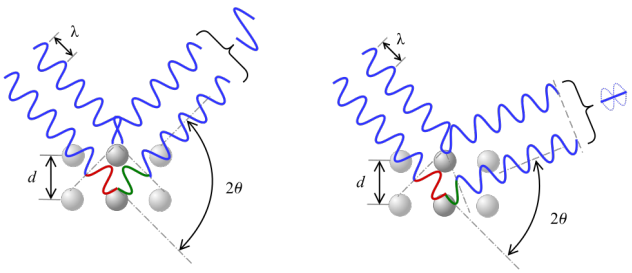

Figure 1.1: Visualisation of Bragg's law (source: [3])

only precise to a certain degree and are generally stochastically distributed around an average value. Because the samples contribution to this widening contains information about defects in the crystal, a careful analysis should provide information about those imperfections. However, because experimental contributions can not easily be distinguished, it is hard to perform in practice. A major factor in peak widening is the size of the diffracting crystal, more precisely the size of the coherently diffracting domains. Ideal Bragg reflection assumes infinite lattice planes and therefore infinite lattice reflections. In real samples very small crystals can result in significant peak widening. This relation is given by the so called Scherrer Equation

$$L = \frac{K \cdot \lambda}{b \cdot cos\theta}. \tag{1.2}$$

L: 'Real' crystallite size
K: Scherrer constant (shape factor)
λ: Wavelength
b: Peak width (in radians)
θ: Bragg angle

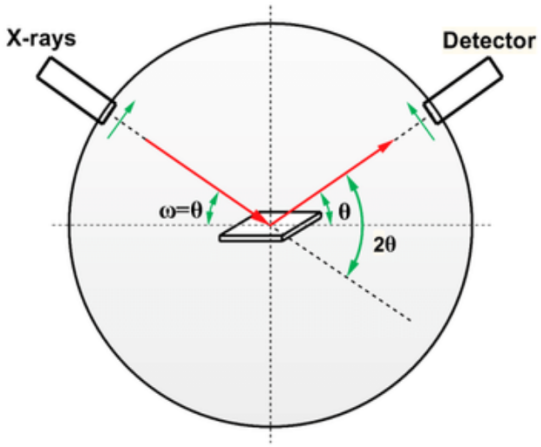

Figure 1.2: Basic experimental setup with Bragg-Brentano geometry (source: [4])

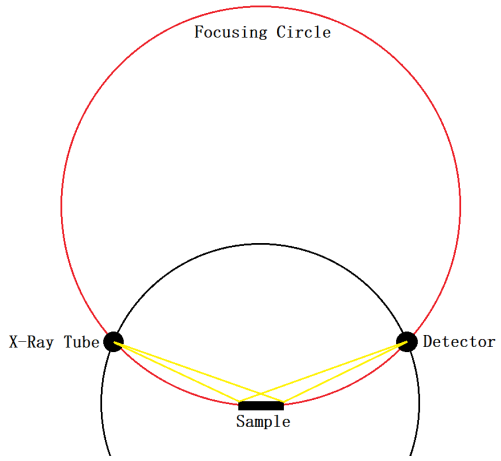

Figure 1.3: Bragg-Brentano geometry with focusing circle.

1.2 Experimental Setup

The experiment which created the data used in this work was based on the so called Bragg-Brentano geometry. (see figure 1.2) This setup is commonly used in powder diffraction experiments and consists of an x-ray tube, a detector and a flat sample. Both the tube and the detector are able to be rotated around the sample and are in equidistant to the sample. The angle of the detector and the tube are both set to θ with θ scanning the whole range of all existing Bragg reflexions. This specific geometry guarantees that nearly all X-rays reflected at a certain angle are captured by the detector. This results from the fact that the sample, tube and detector are all on the same focusing circle. (see figure 1.3)

2 Program Structure

First a short description of the software setup is provided then the program structure is presented. The general program is mainly structured into three parts with only the interface part being dependent on the other parts.

2.1 Software Setup

This document provides documentation for a program which aims to help with the interpretation of x-ray powder diffraction output and the analysis of peaks therein. A visual overview can be seen in figure 2.1.

Because this program requires gnuplot[5] for plotting the data, it is required to be installed on the system before using this software. Furthermore, gnuplot is accessed through simulating console input and therefore a PATH system variable has to be set up for this program to be able to communicate successfully with gnuplot. This can either be accomplished by checking the corresponding check box when installing gnuplot or manually adding the gnuplot bin folder to your systems PATH variables.

In case a binary of this program is not available, the source code can be found at[6]. The libraries 'boost'[7] and 'wxwidgets'[8] are needed in addition to the source-code provided to successfully compile this software. The boost library provides many utility functions and wxwidgets offers an easy implementation for graphical interfaces. Version 3.1.3 of wxwidgets was used in this work. Additionally, because Code::Blocks[9] was used, a project file is also included.

For further instruction on how to setup the database files which are needed for analysis see chapter Database.

2.2 General Graph Calculations

'Graph.cpp' and its header contain the code for all general calculations of graphs and are generally independent of its use in this specific application. They therefore

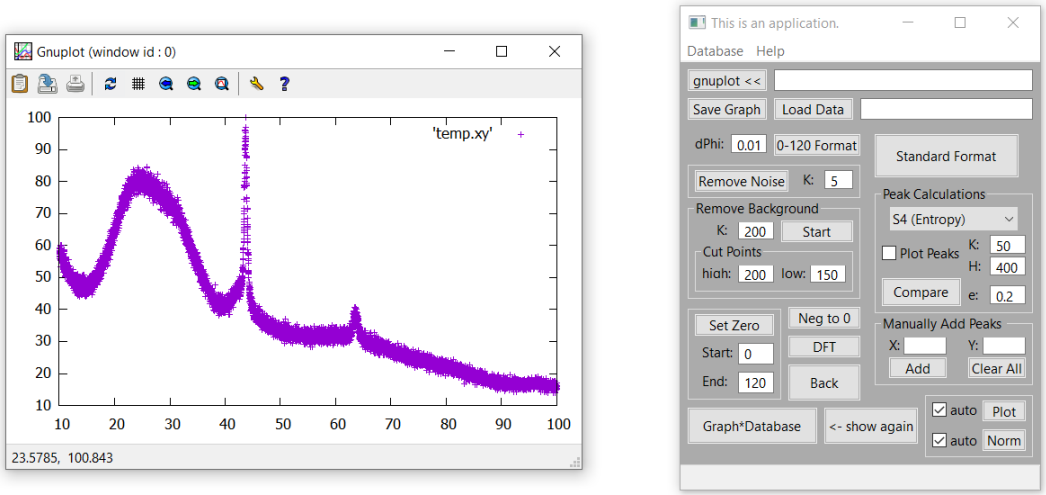

Figure 2.1: Overview of the an opened gnuplot window and the program interface.

contain none of the peak detection algorithms.

2.3 User Interface and Specific Algorithms

The code handling the user interface and the more specific algorithms is located in the files 'PA_WXApp.cpp', 'PA_WXMain.cpp' and their respective headers with most of the relevant code being in 'PA_WXMain.cpp'. The general structure of the layout was created by wxsmith[10], a Code::Blocks plugin for use with wxwidgets. Even though this plugin is not needed, it is highly recommended to use it for further alterations of the code.
These files contain most of the code for the graphical user interface and all peak and advanced graph calculations.

2.4 C++ Gnuplot Interface

'gnuplot-iostream.h' is a C++ interface for gnuplot created by Daniel Stahlke (dan@stahlke.org). This file has not been altered for this program. For further information see `http://www.stahlke.org/dan/gnuplot-iostream` (as of June 15, 2020).

3 Algorithms

What follows is a list of all algorithms used in this program and a description thereof.

3.1 0-120 Format

This algorithm creates a new graph with 2·theta values ranging from 0° to 120° in steps of the parameter dTheta. All values outside of the range of the original graph are then set to the average of the three first/last points of the original graph. Finally estimation of the graph at all other values of 2·theta is done by linear approximating in-between the nearest two points of the original graph.

3.2 Kernel Density Estimation

Kernel density estimation is a technique to estimate the probability density function of discrete distributions. It uses a so-called Parzen-Rosenblatt window[11] method to weight data points depending on their distance to the point being estimated. This method can also be used to reduce the noise in a given function by averaging values around a point and weighting those points according to the window function used.

There are two equivalent ways to imagine how this works. Either you widen each discrete peak by replacing it with an appropriately sized version of the window function (see figure 3.1) or you calculate each point in the density estimation by multiplying the data with the window function centered on the point being calculated.

For this program a discrete parabolic window function $P(n)_W$ was chosen. where 'W' is the parameter for the width of the window. An example of such a function can be seen in figure 3.2.

$$P(n)_W = 1 - (\frac{n}{W})^2, \quad -W < n < W \tag{3.1}$$

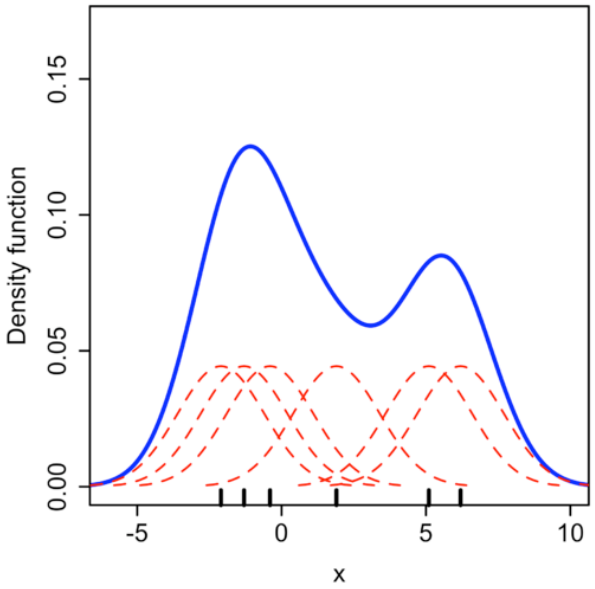

Figure 3.1: Application of the kernel density estimation. The discrete data points seen on the bottom are being weighted by a normal distribution, seen in red. Summation of all normal distribution results on the final estimation of the density function, seen in blue. (source: [12])

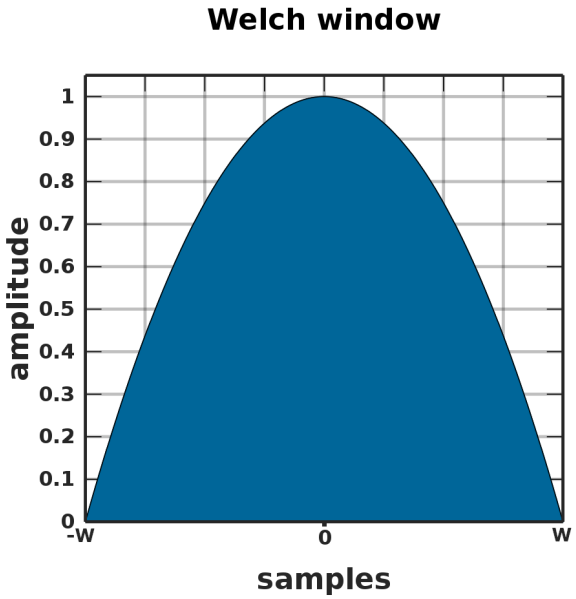

Figure 3.2: A parabolic window function, also called a welch window. (source, altered: [13])

The noise-reduced function $\rho(n)$ is then defined by

$$\rho(n) = \sum_{j=-W}^{W} P(j)_W \rho_0(n+j) \tag{3.2}$$

where $\rho_0(n)$ is the function to be noise reduced.

A good visual explanation can be found at `https://mathisonian.github.io/kde` (as of June 15, 2020).

3.3 Discrete Fourier Transform

This program uses a simple algorithm to calculate the discrete Fourier transform of a given discrete function.
The transformation is given by

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{3.3}$$

and its reverse transform by

$$x_n = \sum_{n=0}^{N-1} X_k e^{i2\pi kn/N} \tag{3.4}$$

where n and k are the indexes of the discrete points in the graph and its Fourier transform respectively. N is the number of discrete points in the graph.
For a more detailed explanation see [14].

Simple multi-threading optimization has also been implemented by splitting the calculation of all X_k into into multiple parts and assigning a thread to each of those parts.

3.4 Linear Background Approximation

For this function the input parameters 'K', 'High' and 'Low'. 'High' + 'Low' have to be smaller than the number of points in the graph.The additional parameter N represents the total number of data points in the graph.
The following calculations for each data point Y(n) from index 'K' to index N-1-'K' to calculate the expected value for the background are performed. For the 'K'

edge points the calculation is skipped.

All values from graph(n-'K') to graph(n+'K') are copied and sorted by their magnitude into a list. The highest 'High' and the lowest 'Low' values are removed from the list. After this only the middle N-'High'-'Low' values will therefore be considered. The average of this list is calculated and this value is then subtracted from the value of graph(n).

3.5 Peak Detection

- Peak values of the database are loaded by reading the database.txt file.
- Every point in the graph is iterated upon to determine if it is a peak.
- Only points where the points to the left/right are lower are considered. This step significantly improves performance.
- The peak significance function(S1, S2, S3 or S4) for each peak is evaluated and all points that have a significance value higher than parameter 'H' are saved.

'K' determines the size of the interval for the evaluation of a potential peak. The interval consists of the 'K' points to the left and the 'K' points to the right of each potential peak. Because of this, points on the edge are ignored when finding peaks.
List of different peak significance functions implemented [15]:

S1 - Max Difference Method

Calculates the maximum Y value difference between each point to the right of the potential peak, and the peak. The same process is repeated for the left side and returns the average of those two values. This function can be used when trying to find only global peaks or after removing most of the noise.

$$S_1(K,i,N) = \frac{max\{x_i - x_{i-1}, ..., x_i - x_{i-K}\} + max\{x_i - x_{i+1}, ..., x_i - x_{i+K}\}}{2} \tag{3.5}$$

K is the width parameter, i the index of the peak and N the sequence of all x_i points around the peak.

S2 - Average Difference Method

Returns the average of the difference between all surrounding Y values and the potential peak in the interval.

$$S_2(K, i, N) = \frac{1}{2K} \sum_{j=i-K}^{i+K} (x_i - x_j) \tag{3.6}$$

K is the width parameter, i the index of the peak and N the sequence of all x_i points around the peak.

S3 - Difference of Average Method

Calculates the average Y value of all points in the interval excluding the potential peak and returns the difference between the peak and the average Y value.

$$S_3(K, i, N) = x_i - \frac{1}{2K} \sum_{j=i-K, j\neq i}^{i+K} x_j \tag{3.7}$$

K is the width parameter, i the index of the peak and N the sequence of all x_i points around the peak.

S4 - Entropy/KDE-Change Method

Returns the change of entropy in the sequence if the point in question is removed from the interval.
The entropy H of the interval is defined by summation of all points in the interval according to

$$H = \sum_{j=i-K}^{i+K} (-p(x_j))log(p(x_j)) \tag{3.8}$$

where $p(x_j)$ is the density calculated by the kernel density estimation at the point x_j and K the width parameter of the interval.
The value of significance is then given by

$$S_4(K, i, N) = H(N(K, i)) - H(N'(K, i)) \tag{3.9}$$

where K is the width parameter, i the index of the peak, N the sequence of all points around the peak and N' the same sequence excluding the peak point.

3.6 Peak Comparison

- All found peaks with the peak detection algorithms are recalculated.
- Found peaks are sorted in ascending order of X-values.
- All X and Y values of peaks in the database.txt file are imported.
- For each peak found by the peak detection algorithm the X-distance to each database peak is evaluated. If the distance is less than the value of 'e' then said peak is shown and saved to the output text file.

3.7 Graph*Database

- A temporary graph for the first file in the data/database sub-directory is created.
- Both the temporary and current graph are formated by calling the '0-120 format' function and normalizes the graphs so that Y ranges from 0 to 100.
- The sum of all products of the two graphs is calculated. The product of two graphs $A(n)$ and $B(n)$ is defined by

$$A \cdot B = \sum_n A(n) \cdot B(n) \tag{3.10}$$

  where $A(n)$ and $B(n)$ are all the data points at index n.
- The value of $A \cdot B$ is saved and this process is repeated for every file in the data/database sub-folder.
- The products are sorted and outputted in descending order. A copy of the output is also saved into a text file for later review. If the file is not renamed, the text file will be deleted the next time this function is called.

4 Graphical Interface

What follows are descriptions of all available operations and their use. Short descriptions of most features can also be found via tool-tips in the application. For a detailed explanation of used algorithms and functions see section 'Algorithms'. The buttons and input fields have been numbered for ease of reference in figure 4.1.

1 - Status Bar

Provides access to further features.

Database

'Open' opens the 'database.txt' file if it exists.
'Reload' starts the process of regenerating the database.txt file from all files located in the data/database sub-folder of the working directory. The cutoff-height parameter determines the minimum height peaks in the database have to be for them to be saved to the database.txt file. When setting the parameter to 50 for example all peaks with height lower than 50 will be ignored. For choosing the parameter, consider that all database files should be normalized to where the maximum peak has a value of 100.

Help

This provides help in various situations. For more detailed help and explanations, refer to this document.

2 - Direct Gnuplot Input

With this function gnuplot can be interacted with directly. The syntax is the same as if writing in a gnuplot command line. Use 'replot' to add things to the current graph for example.

3 - Save Graph

Opens a file dialog to save current graph to a .xy file.

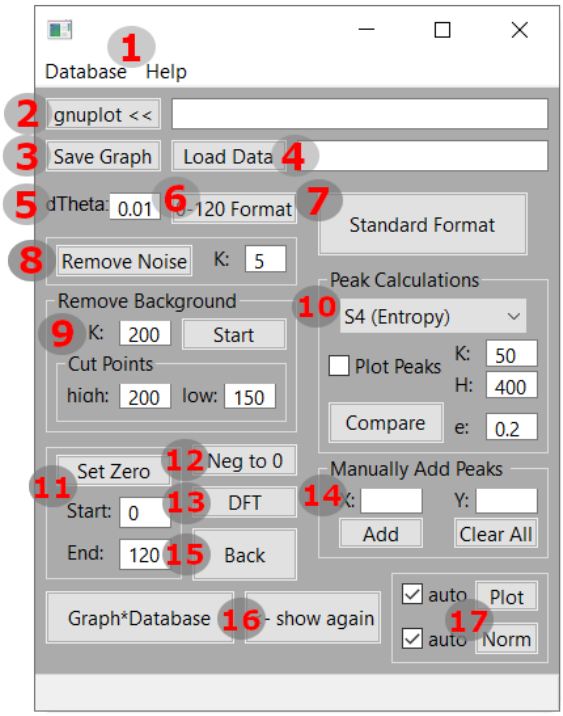

Figure 4.1: User interface with numbered buttons for ease of reference.

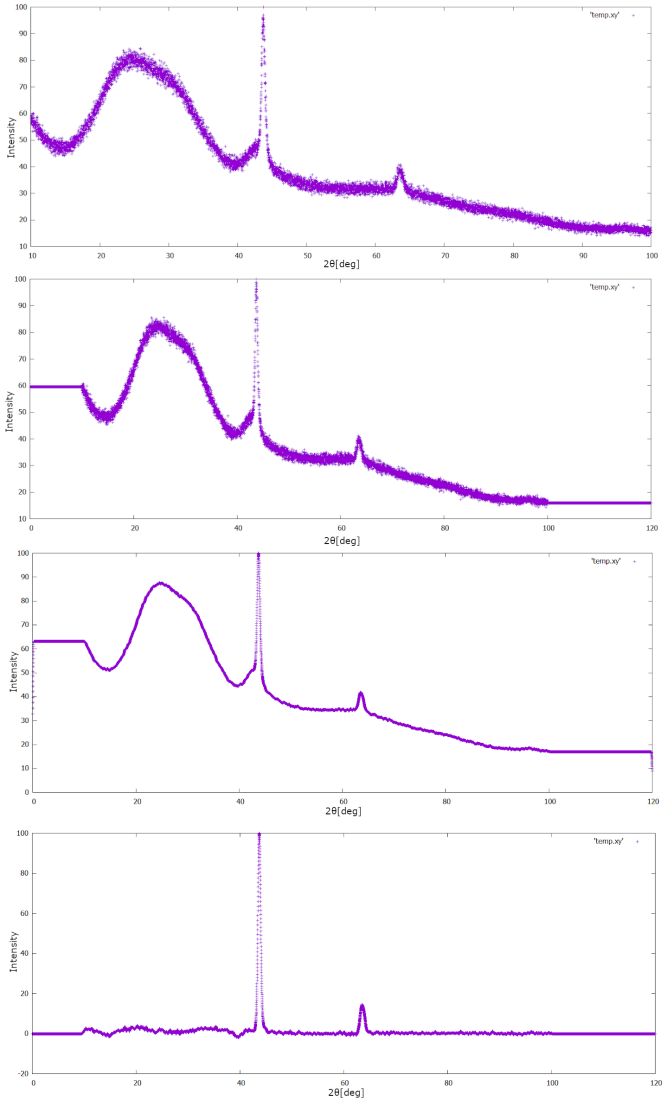

Figure 4.2: An example of formatting a graph step by step. Functions used in this example from top to bottom:
'0-120 format': 'dTheta'=0.01
'Remove Noise': 'K'=20
'Remove Background': 'K'=200 'High'=200 'Low'=150

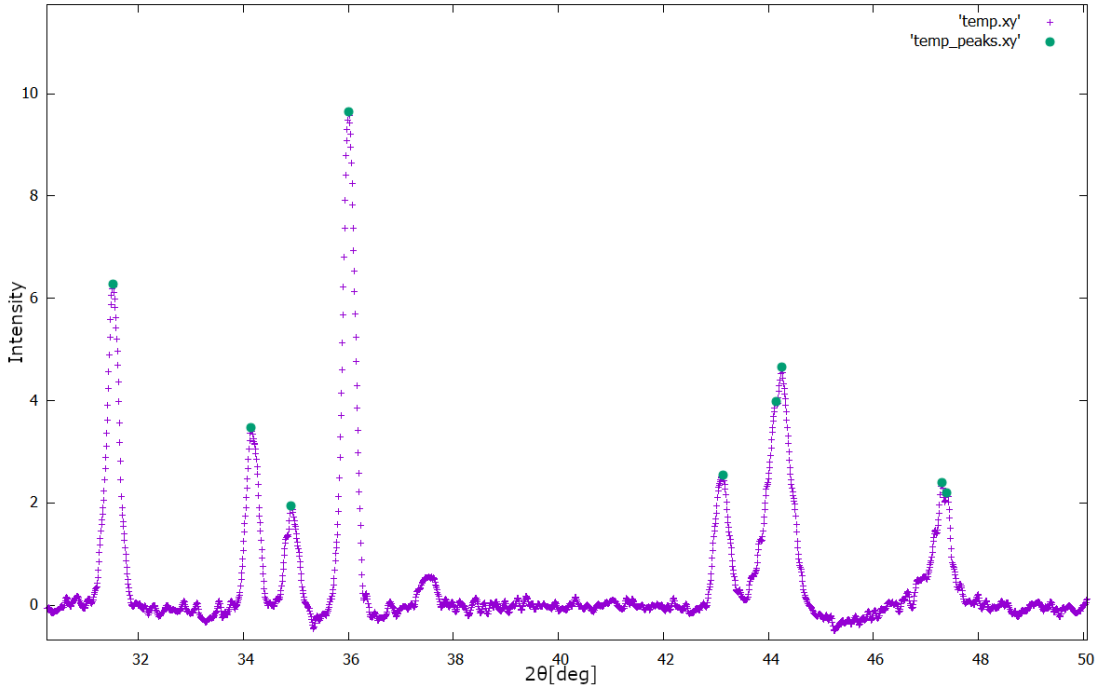

Figure 4.3: Example of an application of the S4 peak detection algorithm.

4 - Load Data

Loads Data from a '.xy' file. Other formats can be used but the data must be formated appropriately. The program searches in the current folder of the program and in the data sub-folder. The '.xy' ending can be omitted when entering the filename. If the input text field is empty this opens a file dialog to choose the file manually.

5 - dTheta

This is the value for the distance between points on the X axes, when using '0-120 format' or 'Remove Noise'. The default value of 0.01 has been chosen on the basis that all database files provided were in this format.

6 - 0-120 Format

This formats the graph into increments of dTheta from 0° to 120°. In case 'Standard Format' is not used, this function should be applied at the very start to format all data into the range and dTheta increment of the database.

7 - Standard Format

Performs operations in this order:
'0-120 format': 'dTheta'=0.01
'Remove Noise': 'K'=5

'Remove Background': 'K'=200, 'High'=200, 'Low'=200
Activates peak calculation.

This formats the graph in a way that works in many cases or provides a good starting point for others.

8 - Remove Noise

Before analyzing the peaks, the noise often has to be removed or at least lowered. This is accomplished with a kernel density estimation using a parabolic kernel with a width of 2·'K'·'dTheta'. The choice of the kernel was mainly based on empirical testing and the fact that the kernel function should be efficient to calculate.
An example of this can be seen in figure 4.2.

9 - Remove Background

This operation removes background noise by approximating it as linear and by assuming that for ranges of 0 to 2'K' points, most of the values are only the result of background noise. The exact algorithm is explained in section 'Algorithms'.

'K' represents the size of the interval around each point in which the approximation of the background is done. For this reason 'K' has to be big enough so that the interval is significantly larger that the width of individual peaks. The size of the used interval is 'dTheta'·2· 'K'. The background does not actually have to be perfectly linear but only has to be well approximated within the chosen interval, therefore 'K' has to be as small as possible in those cases while still being bigger than the width of peaks.

'High' and 'Low' are the number of highest/lowest points that are ignored when calculating the background for each point. For most cases where 'K' is appropriately chosen it works to only consider the median of all points in the interval. To do that set both 'High' and 'Low' equal to 'K'. This leaves just the middlemost point.

Multiple processes with different values are recommended if one fitting set of parameters cannot be easily found.

An example done with the default parameter of this can be seen in figure 4.2.

10 - Peak Calculations

With this algorithm it is possible to calculate peaks in the current graph and

compare those found peaks with peaks in the database.

The method to determine which points are peaks can be chosen through the drop-down list. Parameters 'K' and 'H' for those functions can be changed here. Exact explanations what 'K' and 'H' represent and how the algorithms work are shown in the 'Algorithms' section. Essentially 'K' represents the size of the interval in which it is decided if a peak truly is a peak while 'H' represents the threshold value for determining a peaks significance. A higher value of 'H' generally means more peaks found.

The default value of those parameters ('K'=50 and 'H'=400) have been found to work in many test cases but the value of 'H' had to be changed drastically in many cases, especially when changing the type of peak determination function. In most test cases 'S4'(Entropy) worked best while varying the value of 'H'.

When the check mark is checked, peaks will be actively searched and shown whenever the graph is plotted. Because peaks have to be recalculated with each change in the graph and calculating it every time can be relatively time intensive with high values of 'K', it is therefore recommended to only enable it when the graph is formated. However, on most modern hardware this should most certainly not pose a big problem.

One of the most important functionalities is the ability to compare found peaks with database peaks to guess what crystal structures those peaks originate from. The program simply checks each peak saved in the database text file against each peak found in the database. The only parameter here is 'e', which represents the maximum distance on the x axis a peak can have to a peak in the database to still identify it. Input 'e' in units of the X-axis, usually in degrees. All peaks are returned sorted by their position and all matches are both shown on-screen and saved to a file named "found_peaks.txt". An example done with the default parameter of this can be seen in figure 4.2.

11 - Set Zero

Sets all data points between 'Start' and 'End' to 0. This can be used to isolate peaks of interest by setting everything else to 0 or manually removing unwanted peaks or noise.

12 - Neg to 0

Sets all negative values in the graph to zero.

13 - Fourier Transform

Performs a Fourier transform on the graph. This is usually not used but has been added to the program for additional optional functionality.

14 - Manual Peak Input

Manually adds a peak to the list of all found peaks. The inputted peak does not actually have to be a peak to be added to the list. This can either be used to add peaks that have not been found by the algorithm or to add peaks that should be there or were removed for some reason.

15 - Back

Reverts last change to the current graph. This only works once because the program only saves the last state of the graph.

16 - Graph*Database

Tries to estimate a probability for each crystal structure file in the database. This is by far the most time-intensive algorithm and should only be used when the background of the data has been removed significantly. The result depends on many factors and should only be a guide to help identify the actual underlying structures and elements.

17 - Auto Settings

Enables or disables automatic plotting and/or normalization of the graph after every change of data or use of a function.

5 Database

For all analysis and identification of samples and their peaks a database of different materials is needed. The database contains the information of various crystal structures and their theoretically calculated x-ray powder diffraction pattern. It consists of various files in the data/database sub-directory containing the whole output from x-ray powder diffraction formatted into X and Y columns. There is also a more condensed version of the database containing only the height and position of peaks. This file has to be updated by reloading the database each time its content changes. This condensed file, named 'database.txt' located in the main folder, is used for faster calculation and different cut-off heights for peaks can be chosen upon creation of the file in the program.

Database Files Creation Even though files could be added from other sources, as long as they are formatted correctly, all files used were created with VESTA[16], using its powder diffraction pattern function with different crystalline structures. Appropriate wavelength parameters corresponding to the parameters used in experiments must be used. Then the data should be exported into an '.xy' file.

Adding New Files New files can be added by simply putting correctly formatted files into the data/database sub-folder and then reloading the database with the corresponding function in the program.

6 Ways to Improve the Program

In this chapter some thoughts about potential improvements to the code and other ideas on how to add further utility to the program will be presented.

Precalculated Kernel As of now the kernel used in all Kernel Density Estimations is always the same and is always calculated. This is not really necessary and could be improved for faster calculations.

//! Marker There are various locations in the code marked with '//!' that have minor problems which might be necessary to solve in case this program will ever be expanded upon. Reviewing the code by reading and trying to solve all problems is recommended.

Choice of Kernel Only a parabolic kernel has been implemented. The implications of a different kernel in the density estimation has not been examined thoroughly. A change of kernel could certainly lead to improvements.

Automatic Y Finding in Manual Peak Input For easier manual input of peaks, the Y value could be estimated by interpolating the function at value X. Of course, this should not prevent the input of arbitrary values of Y, so one would have to expand the interface and add options for automatic detection.

Automatically Set PATH Variables In the current version it is necessary to properly set up gnuplot by setting its executable folder as a system PATH variable. There should exist away to open gnuplot without prior setup.

Linear Background Remove - Edge Cases As of this version the 'linear background remove' method does not properly handle cases where the edge of the data is not pure background. In most cases this is not an issue but might become one if those datapoints are of special interest in future analysis. As there is no definite way this should be handled, further consideration is needed.

Bibliography

[1] Carleton College, X-Ray Powder Diffraction (XRD)
https://serc.carleton.edu/research_education/geochemsheets/techniques/XRD.html (as of June 15, 2020)

[2] Fritz-Haber-Institut, X-Ray Powder Diffraction (XRD)
http://www.fhi-berlin.mpg.de/acnew/department/pages/teaching/pages/teaching__wintersemester__2015_2016/frank_girgsdies__peak_profile_fitting_in_xrd__151106.pdf (as of June 15, 2020)

[3] https://en.wikipedia.org/wiki/Bragg%27s_law#/media/File:Braggs_Law.svg (as of June 15, 2020)

[4] https://www.researchgate.net/profile/Mohsin_Raza10/publication/315142103/figure/fig30/AS:472893919371277@1489758104999/4-Schematic-of-a-Bragg-Brentano-th-2th-and-b-GIXRD-geometry.png (as of June 15, 2020)

[5] Gnuplot
http://www.gnuplot.info/ (as of June 15, 2020)

[6] Sourcecode on github
https://github.com/SkipXX/PA_WX (as of June 15, 2020)

[7] C++ boost library
https://www.boost.org/ (as of June 15, 2020)

[8] C++ wxwidgets library
https://www.wxwidgets.org/ (as of June 15, 2020)

[9] Code::Blocks
http://www.codeblocks.org/ (as of June 15, 2020)

[10] wxsmith
http://wiki.codeblocks.org/index.php/WxSmith_plugin (as of June 15, 2020)

[11] Parzen-Rosenblatt window method
https://sebastianraschka.com/Articles/2014_kernel_density_est.html (as of June 15, 2020)

[12] https://en.wikipedia.org/wiki/Kernel_density_estimation#/media/File:Comparison_of_1D_histogram_and_KDE.png (as of June 15, 2020)

[13] https://en.wikipedia.org/wiki/Window_function#/media/File:Window_function_and_frequency_response_-_Welch.svg (as of June 15, 2020)

[14] Discrete Fourier Transformation
https://www.allaboutcircuits.com/technical-articles/an-introduction-to-the-discrete-fourier-transform/ (as of June 15, 2020)

[15] Palshikar, Girish. (2009).
Simple Algorithms for Peak Detection in Time-Series

[16] VESTA
http://jp-minerals.org/vesta/en/ (as of June 15, 2020)

[17] Akab V. Oppenheim; Ronald W. Schafer. (2014).
Discrete - Time Signal Processing