SLAC-TN-07-018

Building X-ray Diffraction Calibration Software

Joshua Lande

Office of Science, Science Undergraduate Laboratory Internship (SULI)

Marlboro College

Stanford Linear Accelerator Center

Stanford, CA

August 24, 2007

Prepared in partial fulfillment of the requirements of the Office of Science, Department of Energy's Science Undergraduate Laboratory Internship under the direction of Dr. Samuel Webb at the Stanford Synchrotron Radiation Laboratory at the Stanford Linear Accelerator Center.

Participant: ______________________

Signature

Research Advisor: ______________________

Signature

Work supported in part by US Department of Energy contract DE-AC02-76SF00515

TABLE OF CONTENTS

ABSTRACT

Building X-ray Diffraction Calibration Software. JOSHUA LANDE (Marlboro College, Marlboro, VT 05344) DR. SAMUEL WEBB (Stanford Synchrotron Radiation Laboratory at the Stanford Linear Accelerator Center, Stanford, CA 94025)

X-ray diffraction is a technique used to analyze the structure of crystals. It records the interference pattern created when x-rays travel through a crystal. Three dimensional structure can be inferred from these two dimensional diffraction patterns. Before the patterns can be analyzed, diffraction data must be precisely calibrated. Calibration is used to determine the experimental parameters of the particular experiment. This is done by fitting the experimental parameters to the diffraction pattern of a well understood crystal. Fit2D is a software package commonly used to do this calibration but it leaves much to be desired. In particular, it does not give very much control over the calibration of the data, requires a significant amount of manual input, does not allow for the calibration of highly tilted geometries, does not properly explain the assumptions that it is making, and cannot be modified. We build code to do this calibration while at the same time overcoming the limitations of Fit2D. This paper describes the development of the calibration software and the assumptions that are made in doing the calibration.

INTRODUCTION

A particularly good method for probing the structure of crystals is x-ray diffraction–the process where x-rays interact with a particular crystal of interest. Because x-rays have wave-like properties, they scatter, or diffract, as they interact with atoms. The x-rays leaving each atom in a crystal interfere constructively or deconstructively as a function of angle. Because crystals have very regular structure, the interference minima and maxima are very regular. These diffraction patterns are measured with a detector. By looking at them, we infer general properties of a diffracted crystal such as the distance between atomic bonds.

X-rays preferentially diffract at certain angles. Because diffraction experiments image many small crystals at different orientations, diffraction patterns have a radial symmetry about the incoming beam. If the beam diffracts preferentially at a particular angle in one direction normal to the incoming beam, it will also diffract at that same angle for any other direction normal to the beam. Because of this symmetry, cones of x-rays emanate from the sample. When we detect this pattern, we measure intensity with a flat detector. When the cone of x-rays intersect the plane of this detector, we detect conic sections of high intensity. When the detector is perpendicular to the incoming rays, we see circles of high intensity. When the detector is tilted slightly, we see ellipses of high intensity. With more extreme tilts, we can see parabolas and hyperbolas. A particular diffraction pattern is shown in figure 1.

Previous literature has determined a set of geometric transformations for dealing with experimental data taken with a detector at an arbitrary angle to the x-rays[1]. This theory is useful because it allows us to infer from these images physically meaningful properties of the sample.

By measuring the parameters of an x-ray diffraction experiment, such as the particular tilt, we can use these transformations to calculate the properties of the crystal that we are interested in. But often, the reverse of this process is desired. We often do not precisely know the parameters of a diffraction experiment. This happens because it is difficult to accurately measure these parameters. A process of calibration can be used to precisely determine the experimental parameters. We first diffract a well understood sample. We then fit the experimental parameters to the diffraction pattern to figure out the precise experimental arrangement that was used. This process is done to calibrate other diffraction data. We first image a known crystal and use it to determine these experimental parameters. We then use the inferred experimental parameters to the analyze diffraction data that we are interested in.

This process of calibration must be done before any sample is analyzed. Fit2D is a program which is commonly used to do this sort of calibration. It has many limitations. In particular, the software's curve fitting algorithm requires a lot of time for manual entry. Furthermore, it gives very little control over how the fitting is done. In particular, it cannot ignore overlapping diffraction rings. It can't account for highly tilted geometries. Also, Fit2D is a black box. We don't know what assumptions it is using to do the calibration, we don't know what type of calculations it is doing, and we cannot modify it. Because of these issues, we built a program to properly calibrating these diffraction images. Our new software is particularly useful because it can overcome the limitations of Fit2D and allow for better and more reliable calibration.

THEORY

Figure 2 shows the general setup of a diffraction experiment. Here, x-rays enter the sample from the left and diffract with various angles. Quantum theory predicts that most of the intensity of the outgoing beam would be found in discrete cones of light.[2] One particular cone is shown in the diagram. It is clear from the diagram that if the detector is perpendicular to the incoming x-ray source, then the detector will record a series of rings. The detector in a diffraction experiment is often placed at an angle to the incoming beam. This is sometimes caused by experimental error and is sometimes done deliberately to record diffraction patterns at higher scattering angles. Figure 3 shows geometrically how this works.

Previous literature has derived geometric transforms to deal with this complication[1]. I will summarize and then use their results.[1] The wavelength is denoted by λ. Suppose that our detector is at an angle to the incoming beam. We can characterize any tilting of the detector as two orthogonal rotations. We will call the rotation along the x axis α and the rotation along the y axis β. Figure 4 and 5 illustrate these rotations.

Our detector is made up of many pixels. Each pixel is addressed by a unique pair of coordinates. We will call the pixel location of some point of interest on the detector (x_d, y_d). We will call the pixel location where the x-rays would hit if they went straight through the sample as (x_c, y_c). The distance between the sample and this center is d. Figure 6 illustrates the situation. We will call the pixel scale of the image ps. This is the distance between each pixel (e.g. 1000 mm / pixel). Finally, we will call (x'',y'') the distance from the center of the image to our point of interest. It is easy to convert from pixel values to distances using the formula

$$x'' = (x_d - x_c) \times ps \qquad\qquad y'' = (y_d - y_c) \times ps. \tag{1}$$

(x'',y'') are easy to measure but do not directly tell us anything that is physically meaningful. We must use a transformation to get at physically meaningful quantities.

We can imagine another detector set up the same distance away from the detector. This new detector is perpendicular to the incoming beam as is shown in figure 7. We are interested in where the same x-rays that were detected at (x'',y'') would have been detected on this imaginary detector. We will call this new point (x,y).

[1]Note that I will stray from convention by labeling what literature calls γ by α. This is done to keep a more consistent notation.

The transform that we are interested in is taken from [1]:

$$x = \frac{dx'' \cos(\beta)}{d + y'' \sin(\alpha) + x'' \sin(\beta)} \qquad y = \frac{dy'' \cos(\alpha)}{d + y'' \sin(\alpha) + x'' \sin(\beta)}. \tag{2}$$

We can invert these formula:

$$x'' = \frac{yd}{d\cos(\beta) - x\sin(\beta) - y\cos(\beta)\tan(\alpha)} \quad y'' = \frac{xd\cos(\beta)/\cos(\alpha)}{d\cos(\beta) - x\sin(\beta) - y\cos(\beta)\tan(\alpha)}. \tag{3}$$

Physicists are interested angles instead of distances because they do not depend on detector placement. First, we define 2θ. It is the angle of scattering of the incoming beam. Refer to Figure 8. We define it as follows:

$$\tan(2\theta) = \frac{r}{d} \text{ with } r^2 = x^2 + y^2. \tag{4}$$

Physicists mainly care about the quantity Q and χ:

$$Q = \frac{4\pi \sin(2\theta/2)}{\lambda} \qquad \tan(\chi) = \frac{y}{x}. \tag{5}$$

χ is the azimuthal angle around the incoming beam. Q is particularly interesting because it is proportional to the change in momentum of the photons that arrive at the detector. Both theory and experiment have shown that the Q values for diffraction peaks are a material constant.

Suppose that we now consider a pixel value (x_d,y_d), we can use equation 1 to convert it to (x'',y''). We can use equation 2 to calculate (x,y) and then equation 5 to calculate Q and χ. There are 7 parameters used in this transformation. The pixel scale is fixed by the detector but the other six parameters are free to vary from experiment to experiment. The free parameters are x_c, y_c, α, β, d, and λ. They characterize a particular diffraction experiment. A common procedure when analyzing diffraction data is to take a diffraction pattern and calculate the Q values for each ring. Once the experimental parameters are known, these formulas give a straightforward way to do the calculations.

HANDLING DIFFRACTION DATA

Our calibration program is written in Python, which is particularly adept for building scientific applications. It is a high level languages which makes it robust, flexible, and easy to write. Python has many packages for handling scientific data. These packages allow for fast array operations and include powerful mathematical functions. It also has a powerful library for creating platform independent GUIs. Furthermore, Python acts as a powerful glue language which can interface well with code written in many other languages. This turned out to be very useful. The only practical drawback to Python is performance. It tends to run slower then other programming languages. We decided that our program would have an acceptable performance and we used several strategies to speed up our program.

In order to calibrate diffraction data, we first had to read the data into our program. The diffraction experiment at Stanford Linear Accelerator Center captures diffraction images using the Mar345 Image Plate Detector System. The Mar345 detector generates .mar3450 data files. A Mar diffraction file is made up of three sections as is specified by [3]. The first section is the header. It is a set of two column ascii. The header contains general information about the image, e.g. the *ps* value, the time that the experiment ran for, an estimate for the wavelength of the incoming x-rays, the pixel center of the image and the distance from the detector. The second section is a list of intensity values that were too high to store inside of the data array. This data is stored as uncompressed packed binary. The image itself is finally stored as compressed packed binary. The data is compressed using the lossless ”pck” compression algorithm developed by Jan P. Abrahams. The algorithm was developed specifically to compress x-ray diffraction data. It is particularly efficient because it exploits radial symmetry in the images. It achieves close to 70 percent compression. Code to decompress this format can be found at [4]. As far as I can tell, this is the only implementation of the decompression algorithm that has been written. It is written in the C programming language. To use the algorithm, we wrote a Python wrapper around the C code. Wrapping the C code was decidedly non-trivial but proved to be an elegant solution.

THE CALIBRATION ALGORITHM

Calibration is used to precisely determine the experimental parameters (x_c, y_c, d, λ, α, and β) of a particular diffraction experiment. These parameters can then be used to accurately and precisely analyze other diffraction data. It is easiest to first imagine an experiment where one already knows precisely the experimental parameters. If you collected a diffraction pattern, you can calculate Q and χ for any of the rings. We can do this procedure in reverse. Many crystals have well measured Q values. So, for any set of experimental parameters, we can use equations 1, 3, and 5 to calculate exactly what should show up on the detector. It should be easy to compute this with what is actually recorded. To calibrate an image, we vary the experimental parameters until the output image closely resembles the expected image. Calibration is a large fitting procedure. We fit the experimental parameters to the observed data. Presumably, the experimental parameters that fit best the observed data are the real experimental parameters.

The whola point of this process is to image a standard crystal before imaging important samples. The standard crystal gives you enough information to figure out how the experiment is set up. We can then use this information to precisely and accurately analize the rest of the data.

My calibration algorithm works as follows. It begins by finding a list of peaks around the image of maximum intensity (corresponding to the peaks on the diffraction rings). To get this list of peaks, the code needs a decent initial guess at what the experimental parameters should be. It also needs a list of the true Q value for the image. Furthermore, it requires a ranges dQ for all the Q values. These ranges do not tell where the true Q should be, but instead where in the image to look for the Q values. The program makes the assumption that when the the initial guess is good enough the Q ranges will entirely contain the acutal peak while at the same time not contaning any other peaks. With this assumption, the code picks a large number of χ values. For each χ slice and for each Q value, the code moves along the straight line from $Q - dQ$ to $Q + dQ$ and records all of the intensity values along the way. It uses a linear interpolation to get intensity values between real pixels. After it has recored all the intensity values, it fits a Gaussian curve to the peak. The center of the Gaussian becomes the Q value at the peak. The code converts this Q and χ pair into pixel values to generate a list of all of the pixel values where there were peaks. Figure 9 gives a general sense of how this works.

Once the code has calculated the peak list, it then defines a goodness of fit function as follows. For any set of experimental parameters, it takes the entire peak list and converts that list into Q_{fit} and χ_{fit} values. The particular Q and χ values will depend upon the experimental parameters. We know the true Q value for the peak also since that was one of the user inputs. Presumably, once we picked the true experimental parameters, these Q values should be exactly the same. We can quantify how close these experimental parameters are to the true values as follows:

$$\text{Residual}(x_c,y_c,d,\lambda,\alpha,\beta) = \sum_{All\ pairs} (Q_{fit} - Q_{real})^2. \tag{6}$$

The closer the experimental parameters are to the true parameters, the lower a residual. Ideally, the residual should be 0 for a perfect fit. I had my code numerically minimized this function of 6 variables. To do so, I used an existing non-linear minimization package called levmar[5]. levmar performs the Levenberg-Marquardt least-squares minimization procedure to minimize an arbitrary function of n dimensions. By minimizing the residual function, my code will calibrate x-ray diffraction data.

FUTURE WORK

There is quite a bit of work to be done on this project. Ideally, this program could develop into a full fledged x-ray diffraction data reduction package. It could be able to handle all of the standard diffraction computations. In this respect, there are many fetures which can be added to the program. A more refined graphical user interface could then be developed to make the program easy to use.

ACKNOWLEDGMENTS

I would like to acknowledge the U. S. Department of Energy, Office of Science for creating, organizing, and funding the SULI program. I would like to thank the Stanford Linear Acceleartor Center for hosting me this summer. I would like to thank David Bronfenbrenner and Dr. Apurva Mehta for their suggestions, help, and support of my project. Most importantly, I would like to personally thank my mentor Dr. Samuel Webb. Sam's vision and insight propelled this project forward and and his guidance challenged me to excel in the research.

REFERENCES

[1] A. Kumar, "Analysis Strategy of Powder Diffraction Data with 2-D detector," Office of Science, SULI Program, Stanford Linear Accelerator Center, Menlo Park, CA, Tech. Rep., Dec. 2005.

[2] B. Warren, *X-Ray Diffraction.* New York: Dover Publications, Inc., 1990.

[3] *mar345_formats*, X-ray Research G.m.b.H. Std. 1, 1997. [Online]. Available: http://www.mar-usa.com/support/downloads/mar345_formats.pdf

[4] D. C. Klein. (1995, Oct.) pck.c. [Online]. Available: http://www.ccp4.ac.uk/ccp4bin/viewcvs/ccp4/lib/DiffractionImage/MAR/pck.c

[5] M. Lourakis. (2006, June) levmar: Levenberg-marquardt nonlinear least squares algorithms in c/c++. [Online]. Available: http://www.ics.forth.gr/~lourakis/levmar/

FIGURES AND TABLES

FIGURES

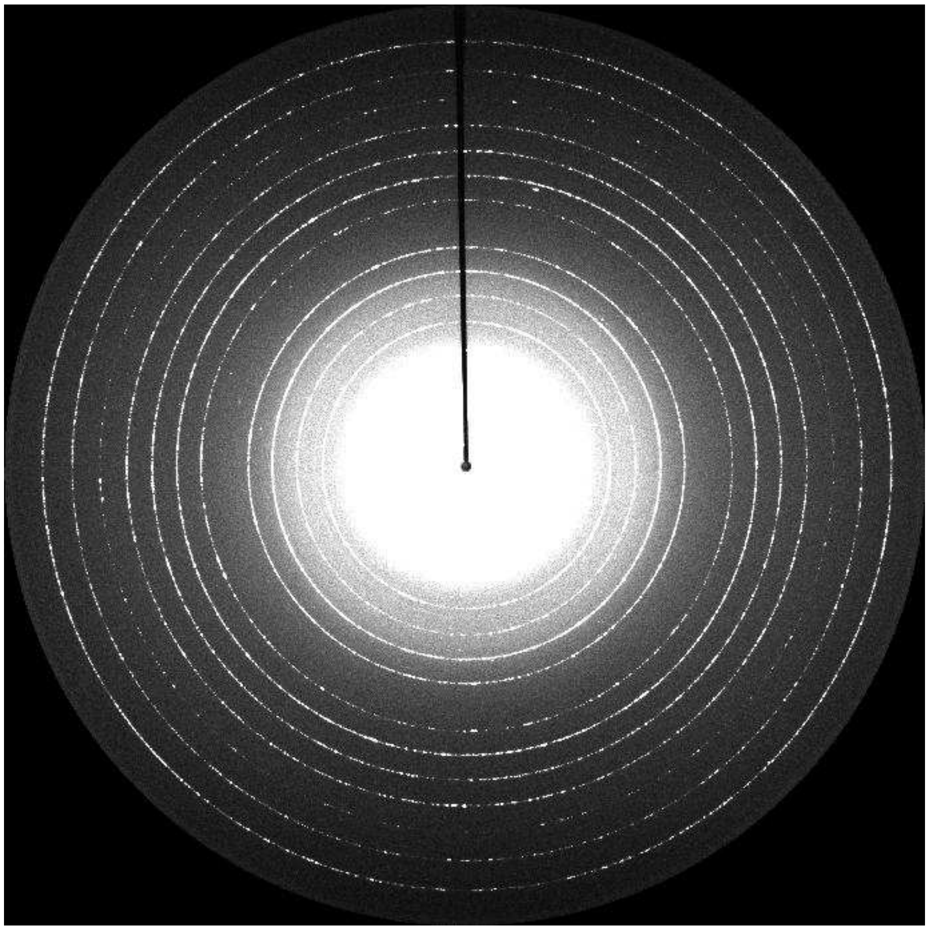

Figure 1: Here is a diffraction pattern of Lanthanum Hexaboride. it is commonly used to calibrate diffraction experiments. Notice the discrete diffraction rings.

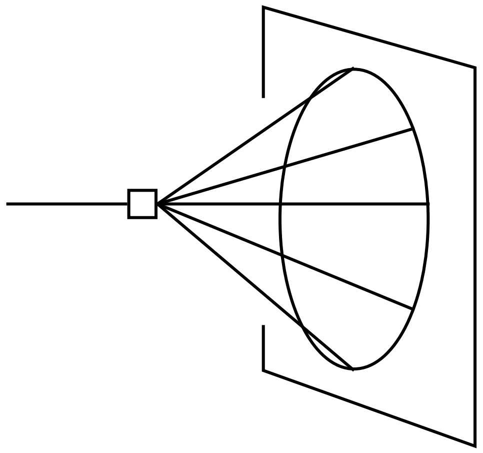

Figure 2: An X-Ray diffraction setup. X-rays scatter from a 3-D sample and are captured by a 2-D detector.

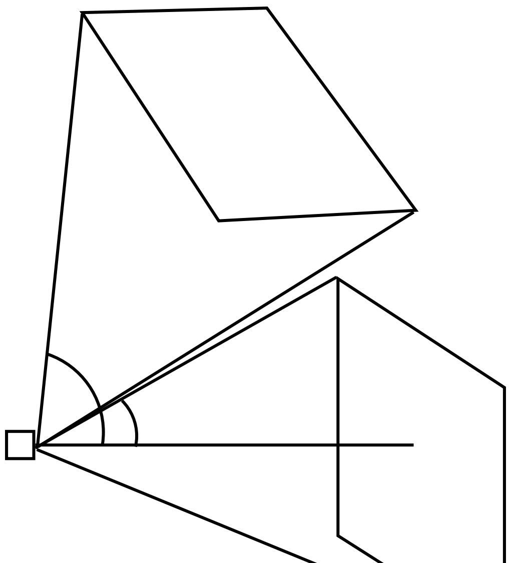

Figure 3: This diagram illustrates how tilted geometries allow for the collection of diffraction data at more extreme angles without the need for a larger detector.

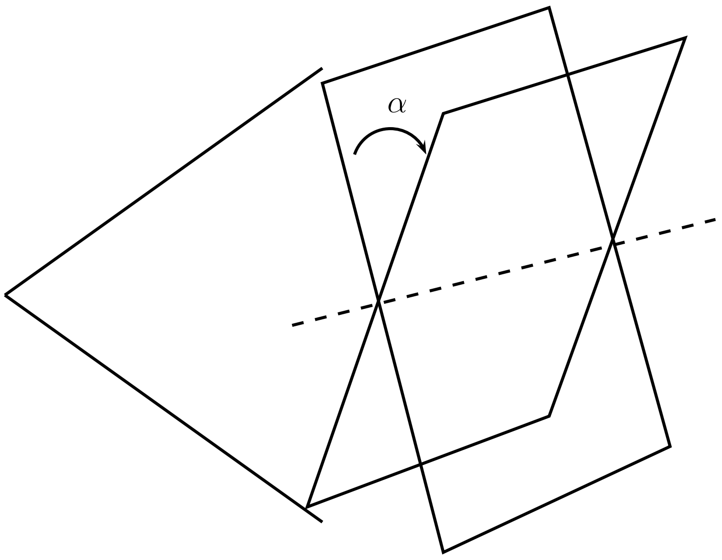

Figure 4: The rotation angle α.

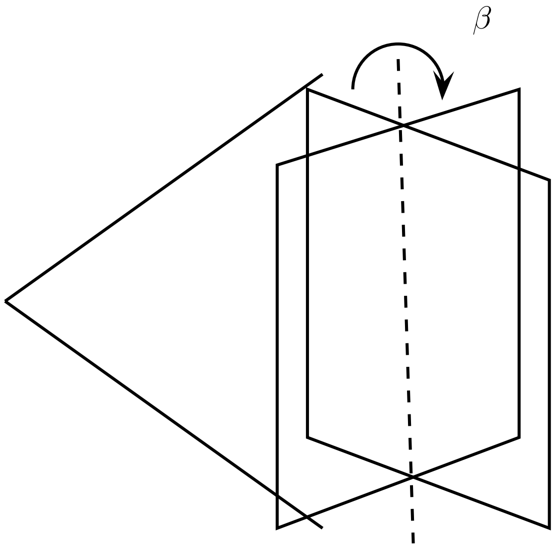

Figure 5: The rotation angle β. Any detector rotation can be characterized as a rotation by both α and β.

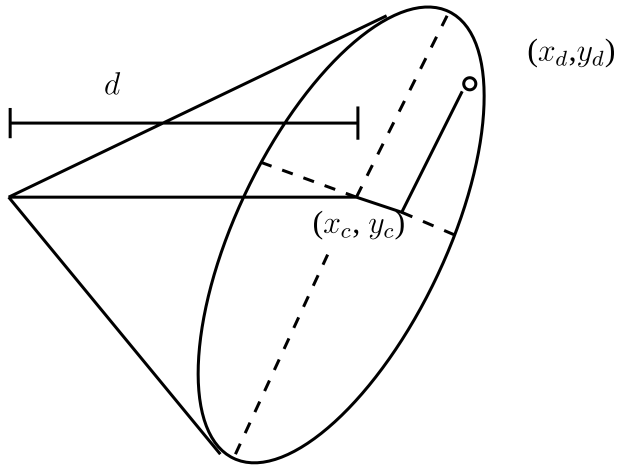

Figure 6: The setup of the experiment. Here, the detector is titled by an angle with respect to the experiment. The image that is recorded on the detector will therefore look distorted.

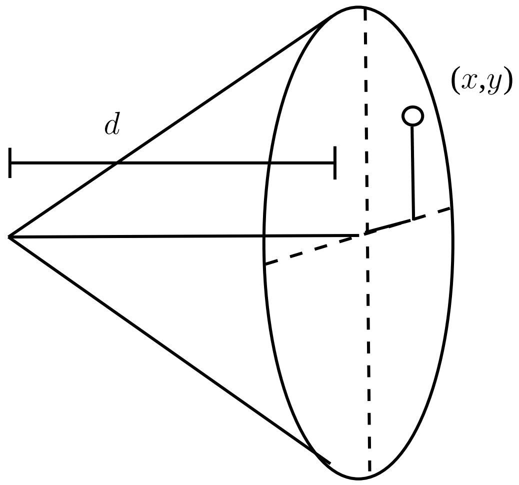

Figure 7: One can imagine another diffraction experiment where the detector is perpendicular to the experiment. (x,y) is the point that the x-ray that arrived at (x'',y'') would have landed at if it arrived at this detector instead.

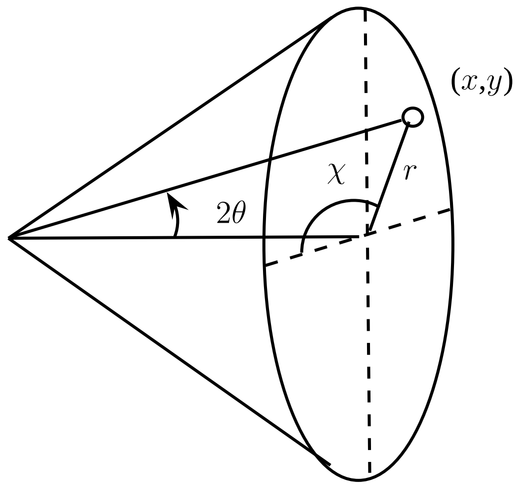

Figure 8: Above is the angle 2θ for a particular point (x,y). 2θ is the angle of scattering of the beam. χ is the azimuthal angle.

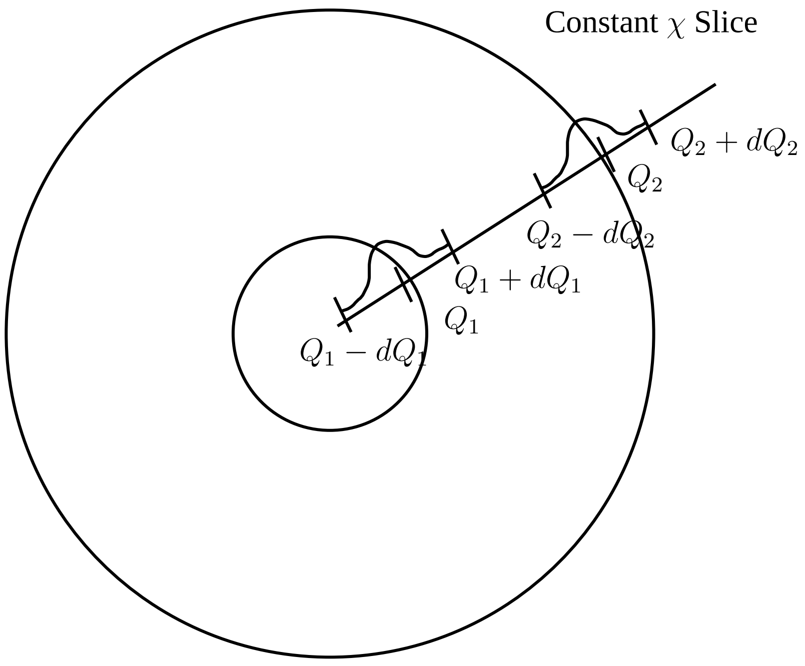

Figure 9: Here is a schematic diagram of the peak finding algorithm. For a particular χ slice, my code fits Gaussians along the line to find the peaks.