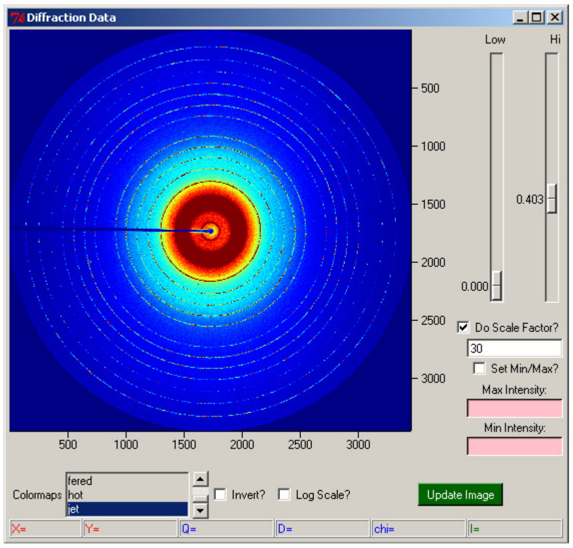


☑ Do Scale Factor? 30
☐ Set Min/Max?
☐ Invert? ☐ Log Scale?

The Area Diffraction Machine

A Program for Analysis of Two Dimensional Powder Diffraction Data

areadiffractionmachine.googlecode.com

By Joshua Lande (joshualande@gmail.com)
August 18, 2008

TABLE OF CONTENTS

CHAPTER 1

TIPS AND TRICKS

1.1 Calibration

The "`Calibration`" tab can also be used to load diffraction data into the program. The tab can be used to calibrate diffraction data to determine the parameters that characterize the experiment. Diffraction data can be loaded using the "`Data File:`" input. The program recognizes "`mar2300`", "`mar3450`", "`mccd`", "`tiff`", bruker "`gfrm`" and "`sfrm`" data, and "`edf`" data. Multiple files can be loaded into the program at the same time using the file selector and the sum image is loaded.

This program characterizes a diffraction experiment according to the parameters:

- "`xc:`", "`yc:`" - the (x, y) pixel coordinate on the detector where the incoming x-ray beam would have hit the detector were there no sample in the way (in pixels).
- "`d:`" – the distance from the sample to the detector (in mm).
- "`E:`" – the energy of the incoming beam (in eV).
- "`alpha:`", "`beta:`" "R" – 3 rotations of the detector (in degrees).
- "`pl:`" – the pixel length of the detector - the width of one pixel (in microns).
- "`ph:`" – The pixel height of the detector - the height of one pixel (in microns).

Before calibrating an image, three things must be done. First, the diffraction data must be loaded. Second, a Q data file with standard Q values for the sample must be loaded. Third, an initial guess of the calibration parameters must be loaded. A guess at the calibration parameters can sometimes be found in the header of the diffraction file. These values can be loaded into the program using the "`Get From Header`" button. The "`Do Fit`" button will perform the calibration and find a best guess at the real experimental parameters.

The "`Work in Lambda`" option in the "`Calibration`" menu can be used to make the program work with the x-ray's wavelength instead of its energy. The calibration parameter "`λ:`" will be used instead of "`E:`".

The "`Q Data:`" input will load into the program standard Q data files. This program stores several standard Q files that can be selected through the "`Standard Q`" option of the "`Calibration`" menu.

The calibration algorithm can be modified in a couple of ways. The program finds peaks in the diffraction data by running from the center of the image out. The number of radial slices that the program uses can be set with the "`Number of Chi?`" input. The "`Stddev?`" input tells the program what ratio higher a peak must be than the standard deviation of the background near it for the peak to be considered real. The higher the value, the more picky the program is about finding legitimate peaks.

If some of the experimental parameters are known exactly, the "`Fixed?`" check box associated with that variable will stop it from being refined when fitting. The pixel length and pixel height are never refined.

To see how good the current calibration parameters are, the "`Draw Q Lines?`" check box will make the program draw on the diffraction image lines of constant Q specified by the Q data file. The ΔQ ranges specified in the Q data file can be drawn using the "`Draw dQ Lines?`" check box. The diffraction peaks that were found while calibrating can be displayed using the "`Draw Peaks?`" check box.

The diffraction image can be zoomed into by left clicking on the image, dragging the mouse, and then releasing. The image can be unzoomed by right clicking on the image. The image can be panned across by shift clicking on the image and dragging. The image can be made bigger or smaller by resizing the window.

In the "`File`" menu, the "`Save Image`" option can be used to save the current diffraction file in several popular image formats. The image will be saved with the current zoom level and any Q lines, ΔQ lines, peaks, or masks drawn on it.

1.2 Masking

The program can ignore certain pixels in an image when performing diffraction analysis. This can be done using the "`Masking`" tab. Threshold masking can be used to ignore pixels above or below a certain value. All pixels larger than a certain value can be masked using the "`Do Greater Than Mask?`" check box and specifying the value in the "`(Pixels Can't Be) Greater Than Mask:`" input. All pixels less than a certain value can be ignored using the "`Do Less Than Mask?`" check box and specifying the value in the "`(Pixels Can't Be) Less Than Mask:`" input. The overloaded or underloaded pixels will show up as a different color on the diffraction and cake images. This color can be changed using the "`Color`" buttons near the other inputs. When a threshold mask is applied, masked pixels will not be used during an intensity integration.

The program can mask areas of the diffraction image using polygon masks. The "`Do Polygon Mask?`" check box will enable this. Any masks in the program will be displayed over the diffraction data and cake images. Any masked pixels will not be used during an intensity integration. The "`Add Polygon`" button can be used to draw new polygon masks. To draw a mask, simply push the button, left click all the nodes on the diffraction image except the last one, and then right click the final node. The "`Remove Polygon`" button can be used to remove polygons from the diffraction image. Simply push the button and then click on the polygon that should be removed. The "`Clear Mask`" button will remove all the polygons from the program. The "`Save Mask`" button and the "`Load Mask`" button will save and load polygons to and from a file.

1.3 Caking

A cake is a plot of diffraction data in Q vs. χ space. χ is a measure of the angle around the incoming x-ray beam. By convention, χ is equal to 0 to the right of the center of the image and increases in a counterclockwise direction. The program needs to know a range and bin size in Q and χ in order to make a caked plot. The "`Do Cake`" button will create a cake and open a new window with the caked data in it. The caked window can be interacted with just like the diffraction window. There is a button called "`AutoCake`" that picks a cake range with the whole diffraction image in it and then cakes the data. Any Q lines, ΔQ lines, and peaks that are drawn on the diffraction image will also be drawn on the caked image. The "`Save Data`" button will save the caked data to a file as plain text. The "`Save Image`" button will save the caked plot as a popular image format with any Q lines or peaks that are drawn on the caked plot saved on top of it.

The "`Do Polarization Correction?`" button will apply a polarization correction to the caked plot. The polarization of the incoming beam can be specified with the "`P?`" input. The formula for calculating the polarization correction is

$$I = Im/PF \tag{1.1}$$

$$PF = P(1-(\sin(2\theta)\sin(\chi-90))^2) + (1-P)(1-(\sin(2\theta)\cos(\chi-90))^2) \tag{1.2}$$

with Im the measured intensity.

1.4 Integrate

An intensity integration is a plot of average intensity vs. Q, χ, or 2θ. By default, the options available are to integrate in Q or χ. The "`Work in 2theta`" option in the menu bar can be used to make the program integrate in 2θ instead of Q. The program needs to know a range (both a lower and upper value) and a bin size in order to perform an intensity integration. When these values have been loaded, the "`Integrate`" button will perform the integration. A new window will open with a plot of data. By default, the integration will be over all possible values of the other variable. This can be changed using the constraint check boxes. For example, selecting the "`Constraint With Range on Right?`" check box and setting the "`Chi Lower?`" input to 0 and the "`Chi Upper?`" into to 90 will cause the integration in Q to be only of pixel's with χ between 0 and 90.

A polarization correction can be applied during an intensity integration. The "`Save Data`" button can be used to save the intensity data to a file as two column ASCII.

1.5 Macro

Macros can be used to speed up data analysis. The "`Start Record Macro`" option in the "`Macro`" menu will make the program start recording a macro. After the desired tasks have been recorded, the "`Stop Record Macro`" option will stop the recording and save the commands to a file. The "`Run Saved Macro`" option will run a macro file.

Small edits to a macro file can make them much more versatile. Most macro commands are just the name of the GUI item possibly followed by whatever the GUI would want (such as a filename or a number). The macro command to load a diffraction file is "`Data File:`" followed by a line with a filename. It can also be followed by a list of filenames, a directory containing diffraction data, or some combination of each. The program will run the subsequent macro on every file in the list and all diffraction files in folders in the list. The loop will end with a subsequent "`Data File:`" line, a "`END LOOP`" line, or the end of the macro file.

When looping over diffraction files, there is special markup which makes it easy to save files in a loop with useful names. Whenever the program finds "`BASENAME`" in a macro file, it will be replaced with the path of the current diffraction file that has been loaded. Similarly, "`FILENAME`" will be replaced with the filename of the current diffraction file. A diffraction file with the extension "`.mar3450`" could be recreate with the command "`PATHNAME/FILENAME.mar3450`". An example of the use of this would be the macro line "`Save Integration Data`" followed by the line "`PATHNAME/FILENAME_int.dat`". The program would always save the intensity integrated data right next to the diffraction file with the same filename but ending with "`_int.dat`".

CHAPTER 2

AN EXAMPLE

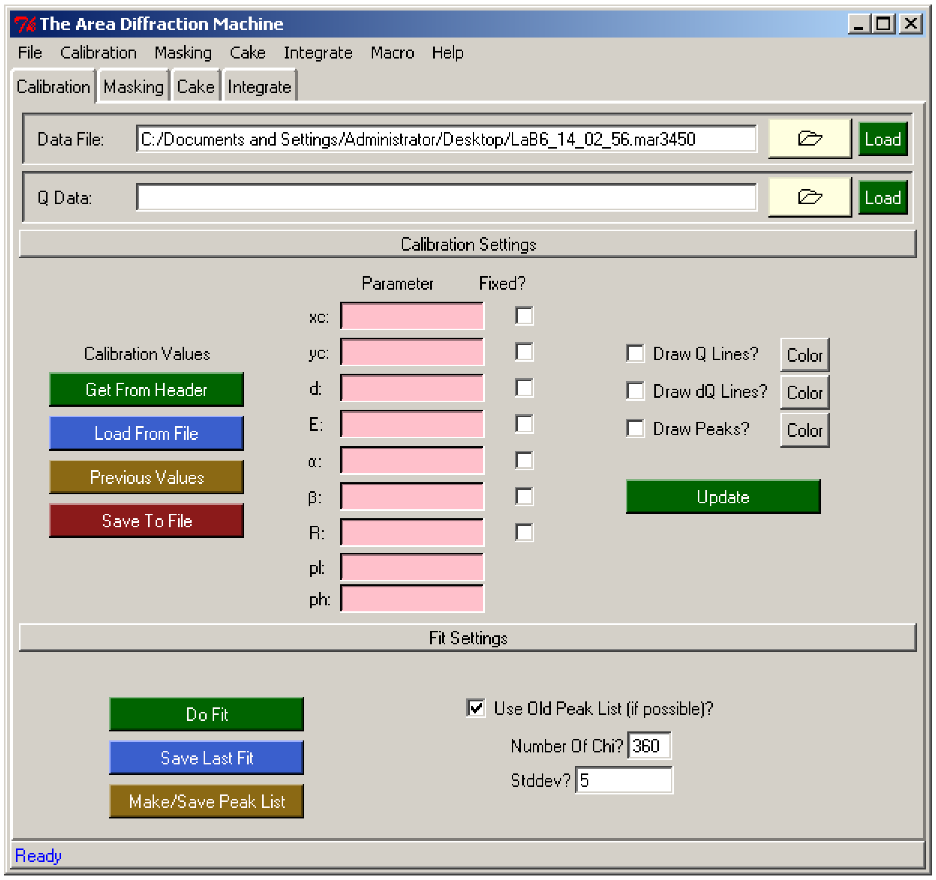
This section will present a pedagogically interesting example which demonstrates several of the program's important features. The purpose of this chapter is neither to be comprehensive nor to be particularly detailed. It will instead give a sense of the types of analysis that can be done with this program. It will act as a motivation for the rest of the manual. Further details and information on any of the things described below can be found in the appropriate sections of the manual.

David, a user of the program, was studying iron thin films using powder diffraction. He was particularly interested in measuring the shift in diffraction peaks of his sample. To realize this experimentally, he capture data of the calibration standard Lanthanum Hexaboride (LaB6). Without changing the experimental parameters, he then imaged many samples that he wanted to analyze.

The steps needed to do this analysis will be described. First, The diffraction detector needs to be calibrated. This will determine the precise experimental parameters that characterize the diffraction machine when the data was captured. Since the image of the standard was taken at the same time as the other images, the calibration parameters inferred from the standard crystal can be used to analyze the rest of data. Figure 2.1 shows what is first displayed when the Area Diffraction Machine is first opened. Using the "`Data File`" input, the LaB6 file can be loaded. Figure 2.2 shows a new window that opens up with the diffraction data. To calibrate the detector, the program must know the Q values for the standard crystal. Since LaB6 is so common, it is a preset default in the program. Figure 2.3 shows how to do this.

The program also needs an initial guess of the calibration parameters. Decent guesses at the experimental parameters are often stored in the diffraction data's header. The program can try to find these values with the "`Get From Header`" button. After the image, the Q values, and an initial guess at the parameters have been loaded, the detector can be calibrated.

But first, the quality of the initial guess can be checked with the "`Draw Q Lines?`" check



Data File: C:/Documents and Settings/Administrator/Desktop/LaB6_14_02_56.mar3450

Q Data:

xc:
yc:
d:
E:
α:
β:
R:
pl:
ph:

☐ Draw Q Lines?
☐ Draw dQ Lines?
☐ Draw Peaks?

☑ Use Old Peak List (if possible)?
Number Of Chi? 360
Stddev? 5

Figure 2.1: The calibration tab.

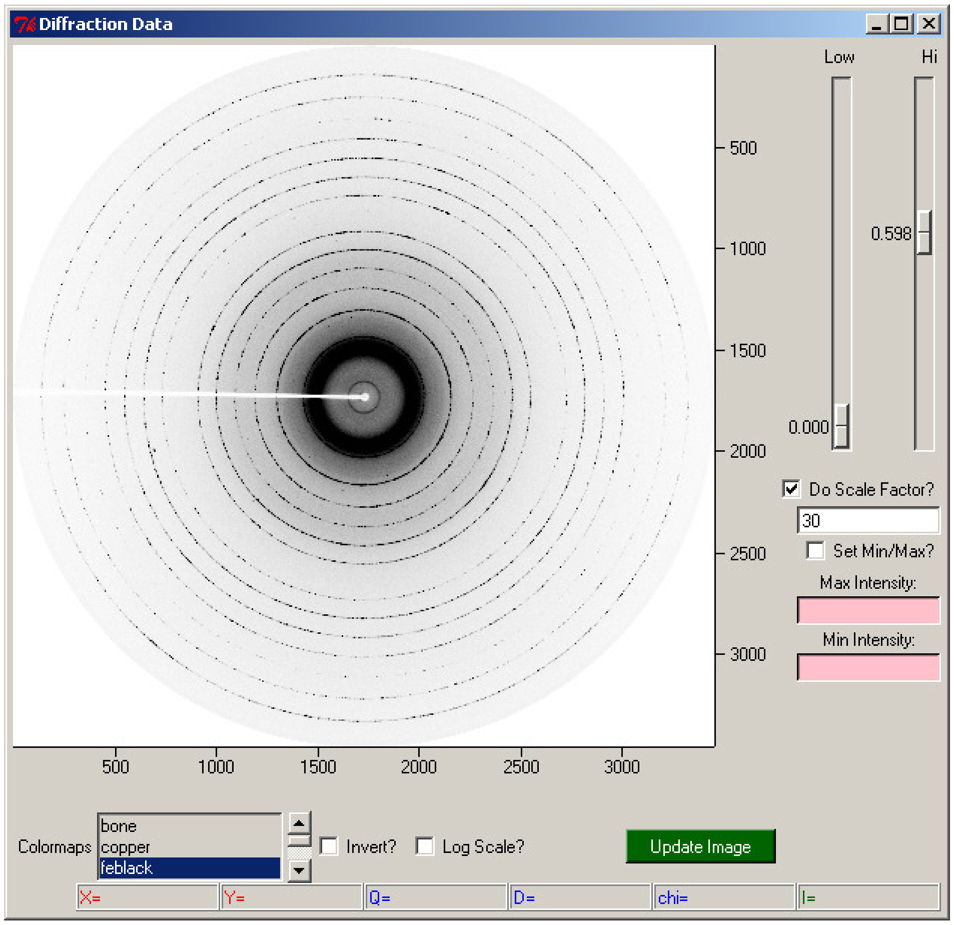


Figure 2.2: The diffraction data window.

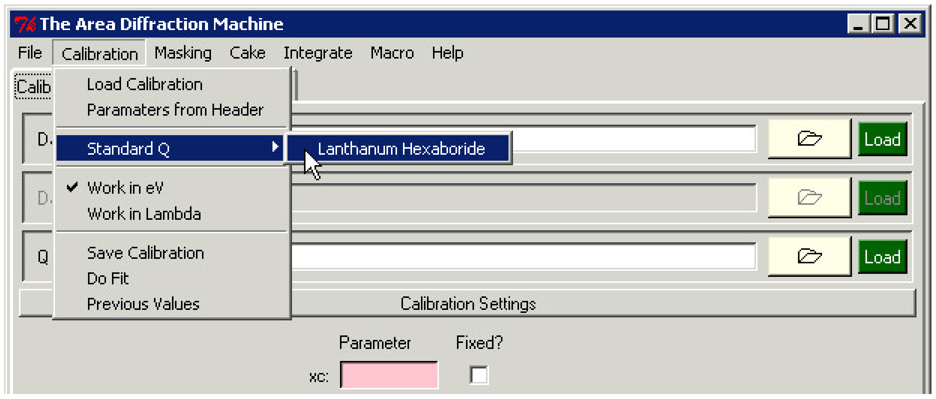


Figure 2.3: Loading a standard Q file. (More standard Q files might be added in the future).

box on the "`Calibration`" tab. The program will draw on top of the diffraction image what pattern should show up on the detector (for the given calibration parameters and Q values). Figure 2.4 shows an example. Of course, the initial guess isn't great.

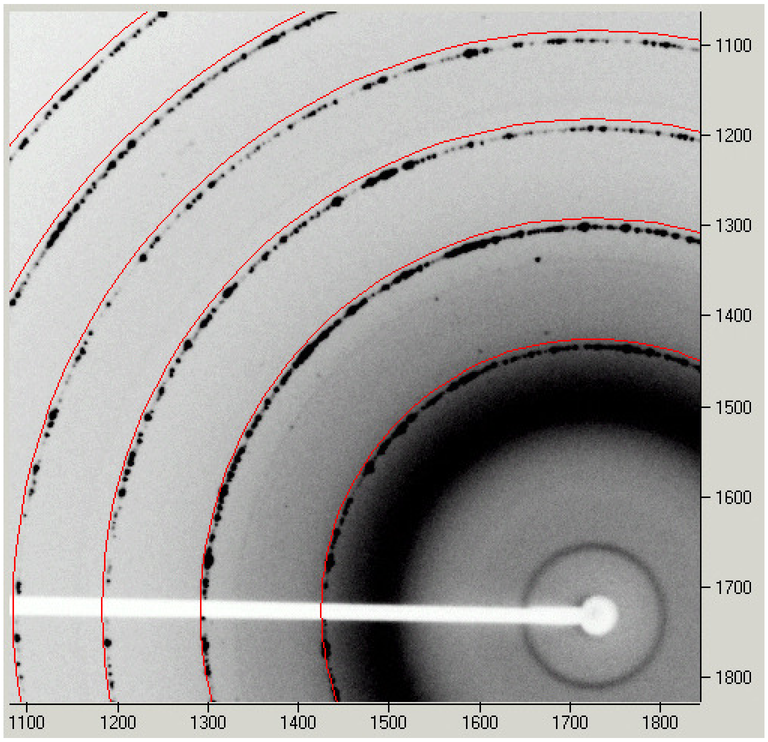


Figure 2.4: The diffraction image with constant Q lines displayed on it. These lines were calculated for the calibration parameters found in the image header and are not particularly accurate.

The diffraction data can be caked. If the calibration parameters are known exactly, the diffraction peaks will show up as many vertical lines. If the parameters are not exact, the diffraction peaks will have a distortion to them. The data can be caked on the "`Cake`" tab using the "`AutoCake`" button and a new window will open. this tab is shown in figure 2.5 and the cake window is shown in figure 2.6. The diffraction peaks on the caked data with the header calibration parameters are not very straight. They have a systematic wiggle. Figure 2.7 shows that this this distortion becomes very obvious when one peak is zooomed into.

Calibration can be done with the "`Do Fit`" button on the "`Calibration`" tab. If the calibration is good, the constant Q lines drawn on the diffraction image entirely overlap the real diffraction patter. A good calibration is shown in figure 2.8. The peaks on the caked image also become much straighter. Figure 2.9 shows the caked data after calibration Figure 2.10 shows that the peaks look good even when zoomed in. The calibration parameters can be saved to a file for later use using the "`Save to File`" button on the "`Calibration`" tab.

There is a beam stop in the image obstructing part of the data. It can be ignored with a polygon mask. Masking is done on the "`Masking`" tab. The masking tab is shown in figure 2.11. Figure 2.12 shows a rectangular mask covering the beam stop. The "`Save Mask`" button can be used to save the mask to a file. The mask can later be loaded into the

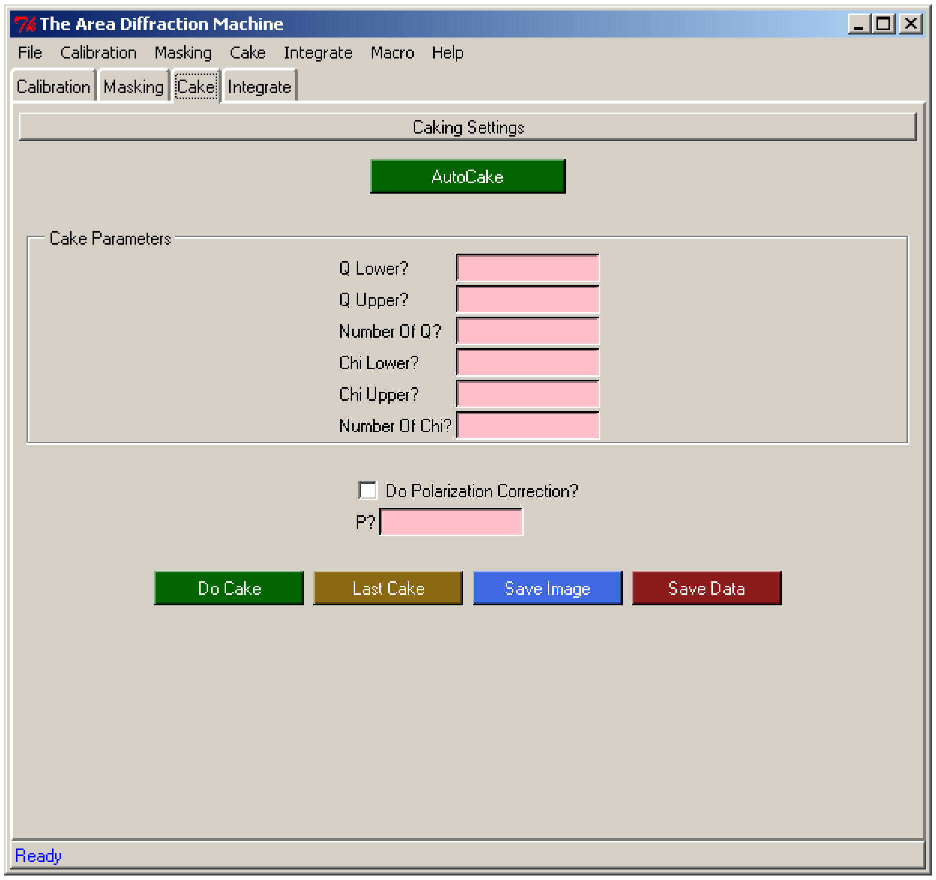


Figure 2.5: The calibration tab.

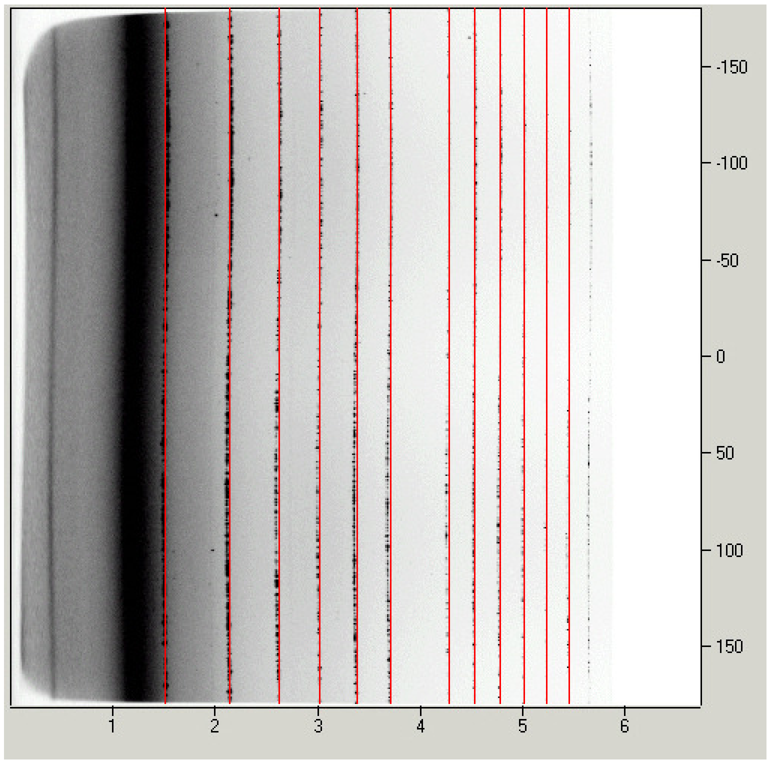


Figure 2.6: A cake with the calibration parameters found in the image header. These parameters are not particularly good and the diffraction peaks are not very straight.

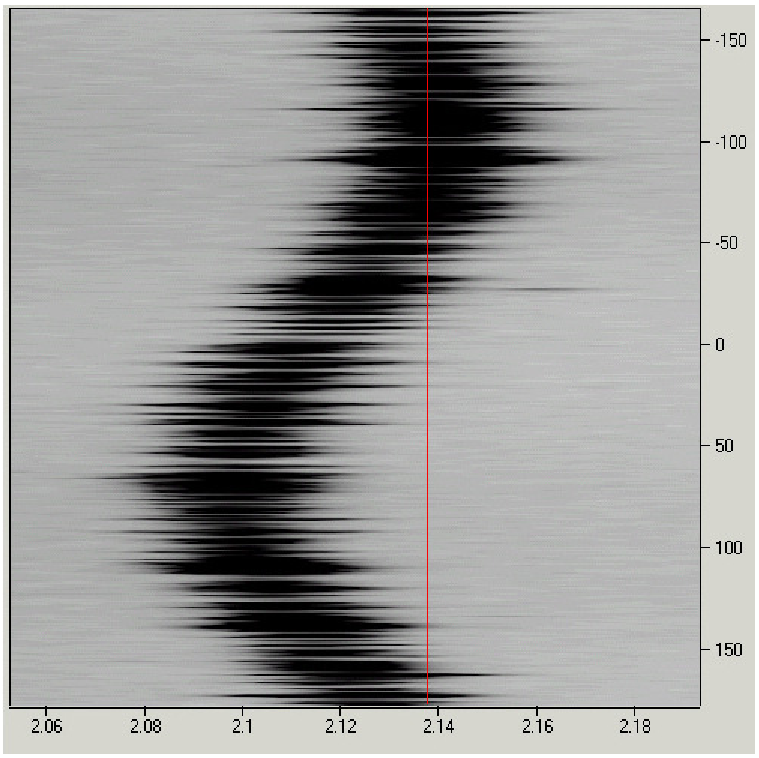


Figure 2.7: A zoom in of the cake shown in figure 2.6. The poor calibration becomes much more obvious.

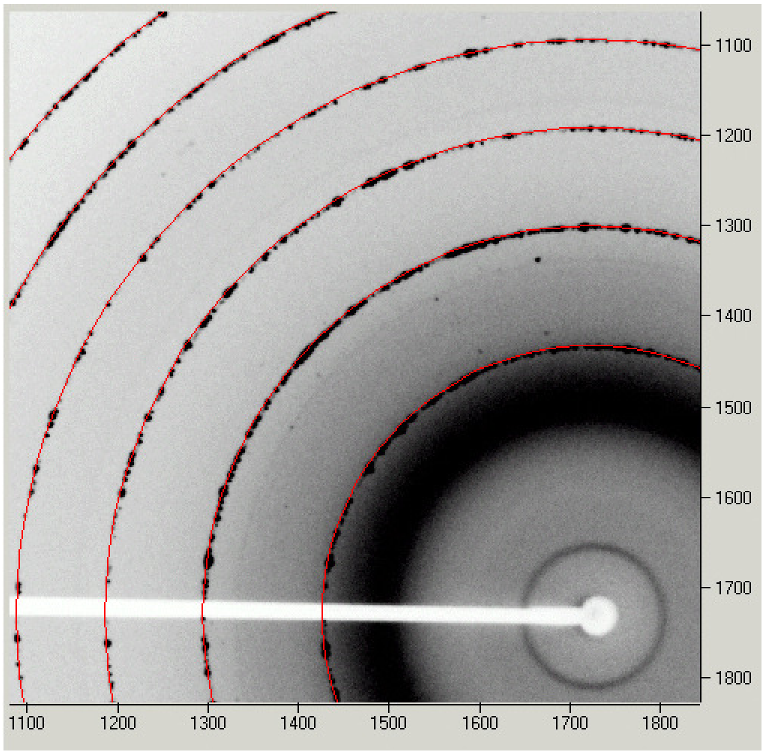


Figure 2.8: The diffraction window after calibration. The constant Q lines are on top of the diffraction peaks.

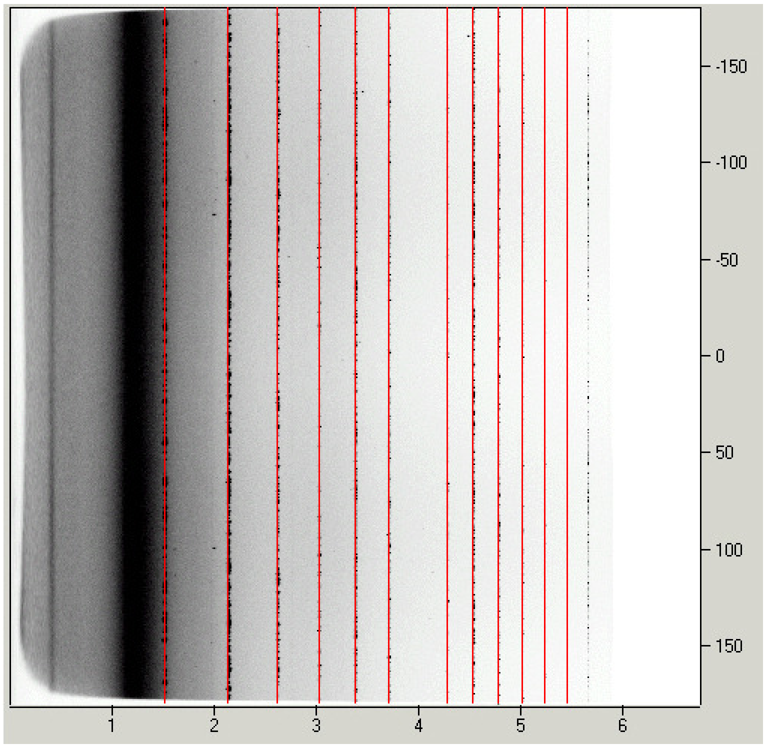


Figure 2.9: The cake window after calibration. The lines are much straighter than before calibration.

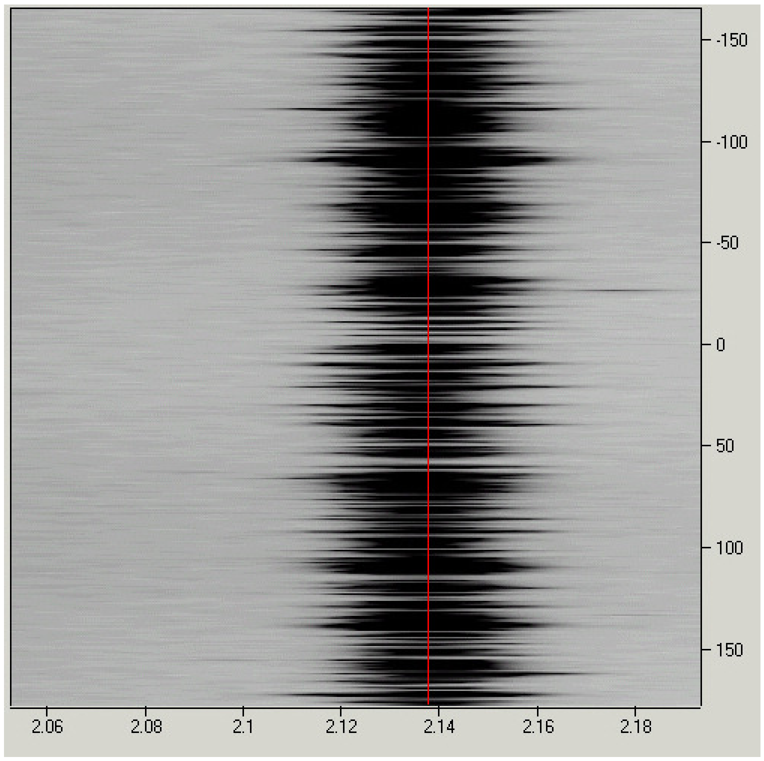


Figure 2.10: The same zoom as in figure 2.9. The line remains very straight.

program when performing the rest of the analysis.

After calibrating the detector and creating the beam stop mask, the rest of the data can be analyzed. This is done by performing an intensity integration on each of the other files. To perform the integration, the diffraction file must first be loaded. The calibration parameters determined earlier must be loaded and the beam stop mask must be loaded. The "`Do Polygon Mask?`" check box needs to be selected to ensure that the polygon masks are used in the analysis. Then, a $Q - I$ integration is done by pushing the "`AutoIntegrate`" button on the left side of the "`Integration`" tab shown in figure 2.13. Intensity data for one of the iron samples is shown in figure 2.14.

The intensity integrated data can be saved to a file using the "`Save Data`" button on the "`Integration`" tab. The data is saved out as two column ASCII. This same process could be done to all the files that need to be analyzed. This would be very time consuming if there were a lot of files . If all of the data was in the folder "`C:/Data/`", all the data it could all be analyzed using the macro

```
Data File:
        C:/Data/
Load From File
    C:/Data/LaB6_cal.dat
Load Mask
    C:/Data/beam_stop_mask.dat
Do Polygon Mask?
    Select
```

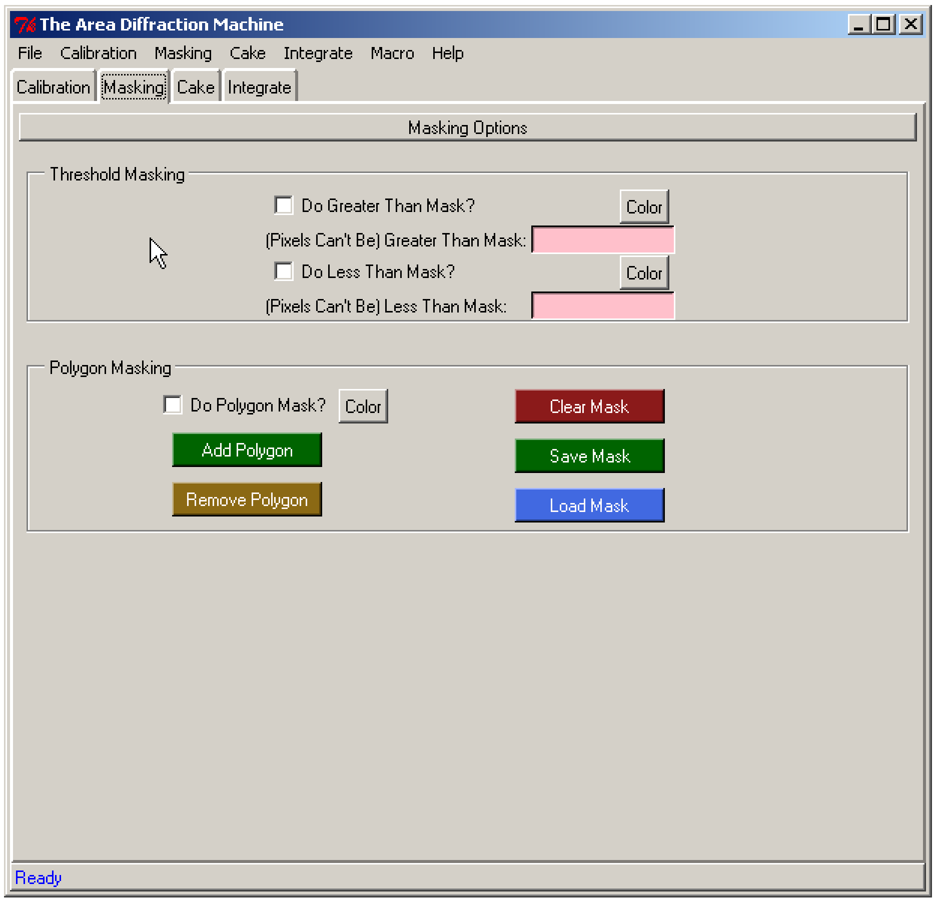

The Area Diffraction Machine
File Calibration Masking Cake Integrate Macro Help
Calibration Masking Cake Integrate
Masking Options
Threshold Masking
Do Greater Than Mask?
Color
(Pixels Can't Be) Greater Than Mask:
Do Less Than Mask?
Color
(Pixels Can't Be) Less Than Mask:
Polygon Masking
Do Polygon Mask?
Color
Clear Mask
Add Polygon
Save Mask
Remove Polygon
Load Mask
Ready

Figure 2.11: The pixel masking tab.

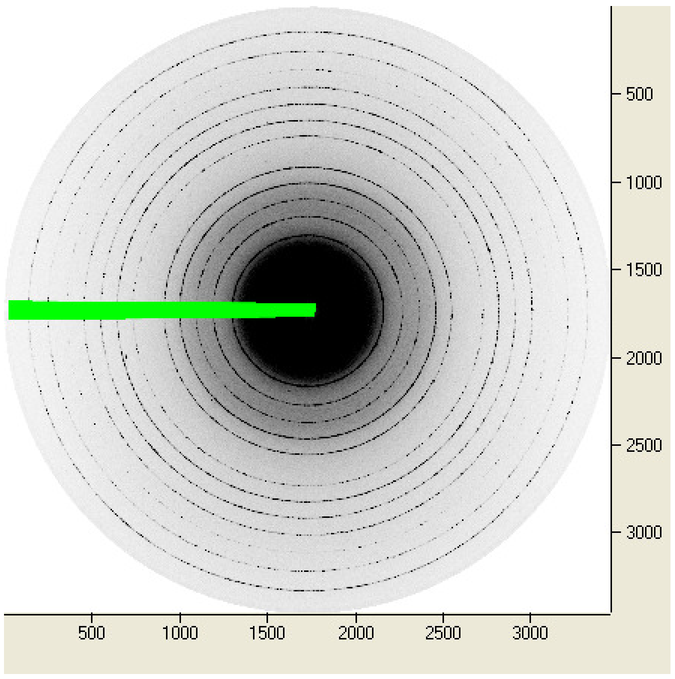


Figure 2.12: A mask is drawn over the beam stop. It will ensure that the beam stop will not be used in subsequent analysis.

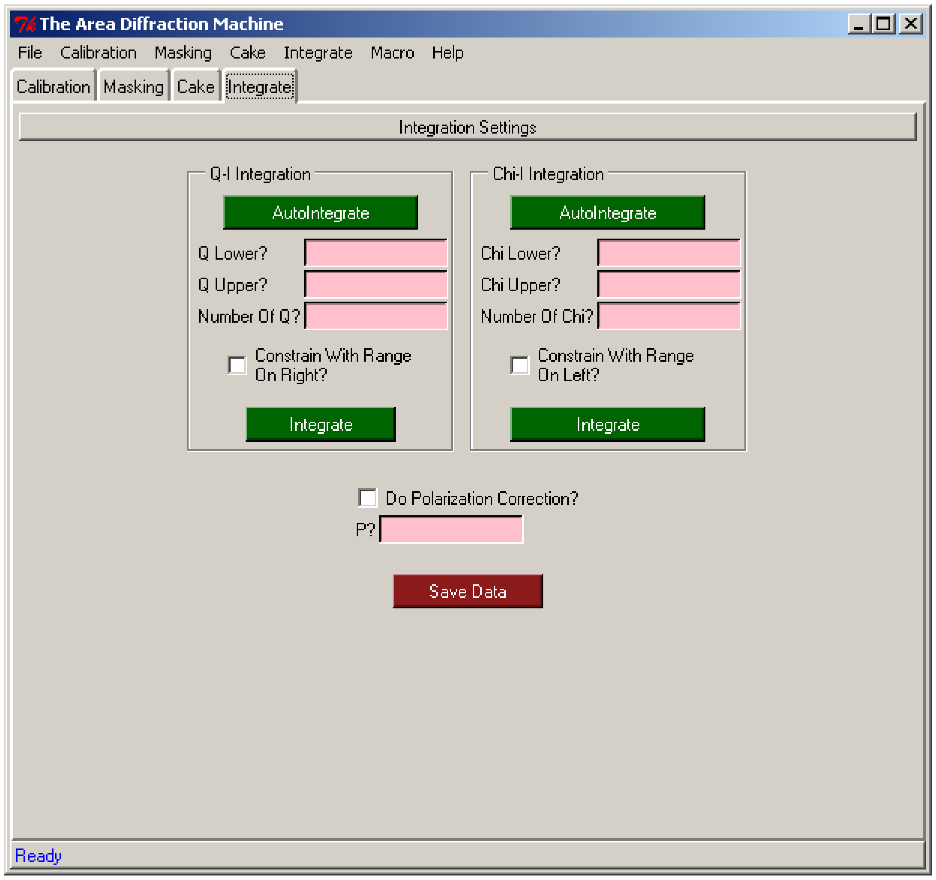

The Area Diffraction Machine
File Calibration Masking Cake Integrate Macro Help
Calibration Masking Cake Integrate
Integration Settings
Q-I Integration
AutoIntegrate
Q Lower?
Q Upper?
Number Of Q?
Constrain With Range On Right?
Integrate
Chi-I Integration
AutoIntegrate
Chi Lower?
Chi Upper?
Number Of Chi?
Constrain With Range On Left?
Integrate
Do Polarization Correction?
P?
Save Data
Ready

Figure 2.13: The integration tab.

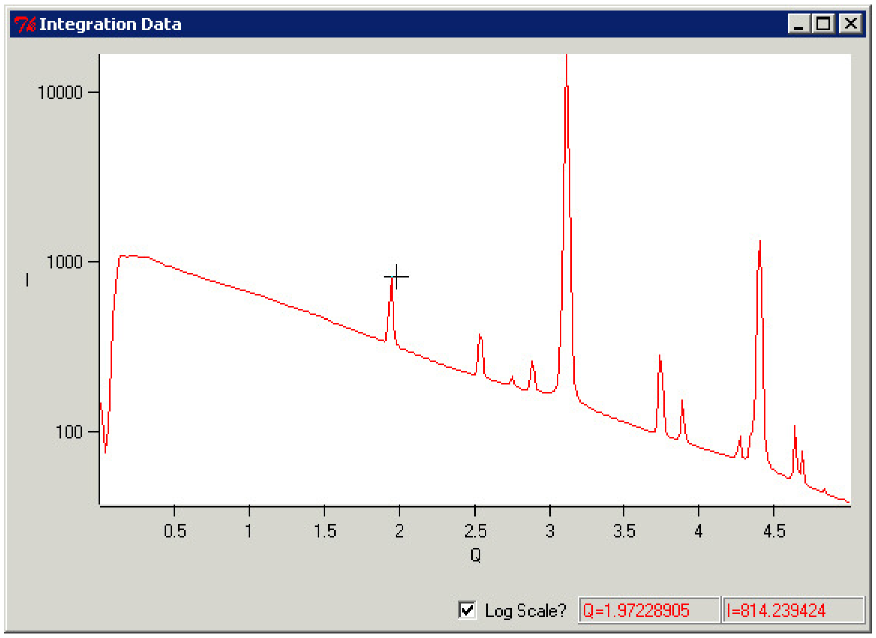


Figure 2.14: The intensity integration window for a particular iron sample.

```
AutoIntegrate Q-I
Save Integration Data
      PATHNAME/FILENAME_int.dat
```

The first command loads into the program all the diffraction files in the folder "`C:/Data/`" one at a time and runs the rest of the analysis on each of them. The macro loads in the calibration parameters and loads in the beam stop mask. It then does a $Q - I$ intensity integrated and saves the integration data to a file.

After the macro is run, the intensity integrated data can be open in Excel. The diffraction patterns of different data files can be plotted on the same graph to look for shifts in diffraction peaks. This data might look something like the graph in figure 2.15.

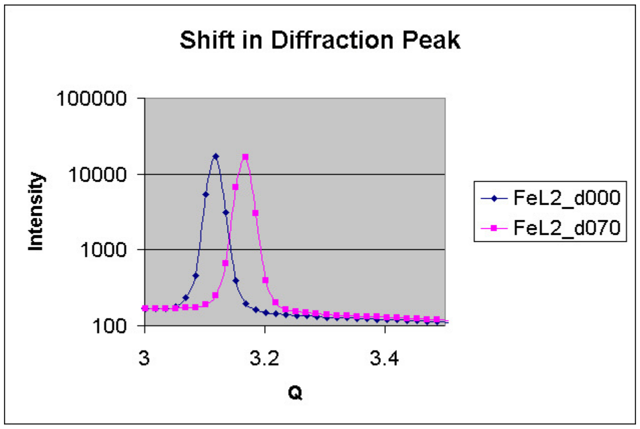


Figure 2.15: What the shift in diffraction peaks might look like when two diffraction patterns are plotted on top of one another in Excel.

CHAPTER 3

VIEWING DIFFRACTION DATA

Figure 3.1 shows the Area Diffraction Machine's "`Calibration`" tab. The "`Data File:`" input can be used to load diffraction data either by typing in the filename and pushing the load button or by clicking on the folder icon and using the file selector. After the file is loaded, the window shown in figure 3.2. will open with the diffraction data in it.

The diffraction data window can be used to interact with diffraction data. The window can be used to:

- *Zoom into the data* – left click on the data and hold down on the mouse. When moving the mouse around, the program will create a resizable square. When the mouse is released, the program will zoom into the selected region.

- *Zoom out of the data* – right click on the data.

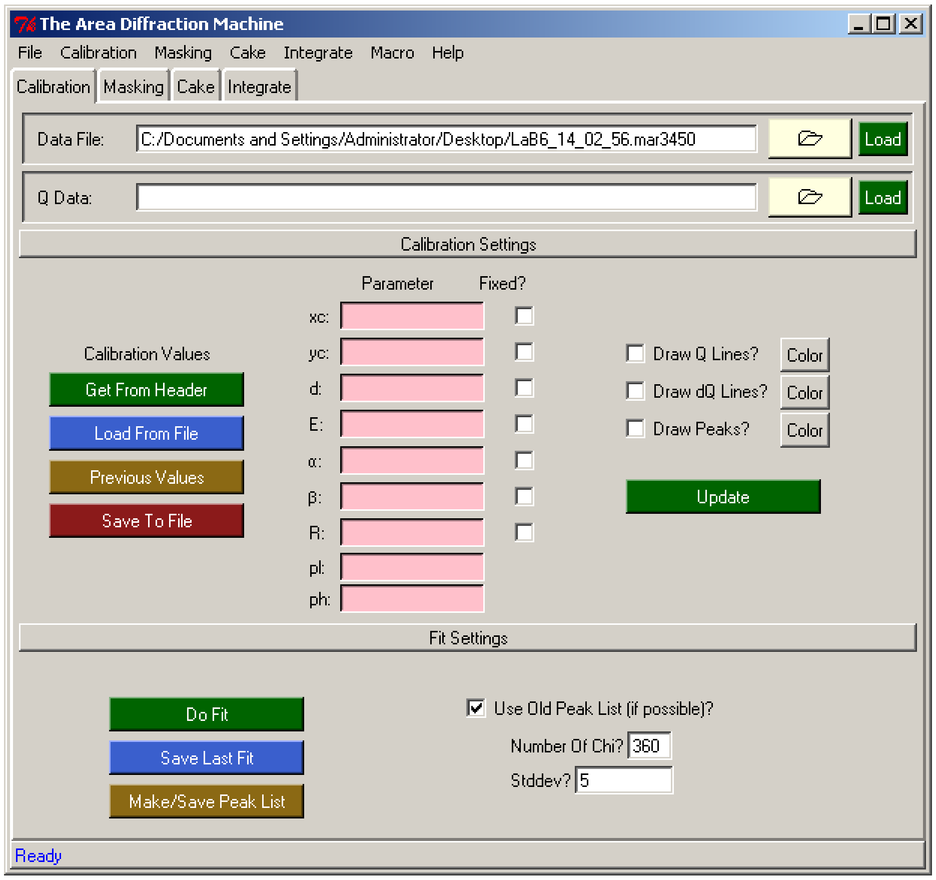
- *Pan across the data* – hold shift, push down either mouse button, and then move the mouse. The image will move with it. Let go of the mouse to stop panning.

- *Resize the window* – click on the bottom right corner of the window and drag. The window will resize just like any other window and the image will become larger or smaller.

- *Read coordinates for a selected point* – when mousing over the image, the x, y, Q, χ, and intensity values for the current pixel will be displayed at the bottom of the window. Q and χ will only be displayed if valid calibration data is loaded into the program. See chapter 5 for a discussion of calibration.

- *Change the Color Map* – the "`Colormaps`" selector can be used to change the particular color map used to display the data.

- *Invert the Color Map* – The "`Invert?`" check box can can be used to invert the colors of the color map.



Data File: C:/Documents and Settings/Administrator/Desktop/LaB6_14_02_56.mar3450

Q Data:

xc:
yc:
d:
E:
α:
β:
R:
pl:
ph:

[✓] Use Old Peak List (if possible)?

Number Of Chi? 360

Stddev? 5

Figure 3.1: The "`Calibration`" tab. This tab can be used to load diffraction data into the program. The "`Save Last Fit`" button was introduces for version 2 of the program.

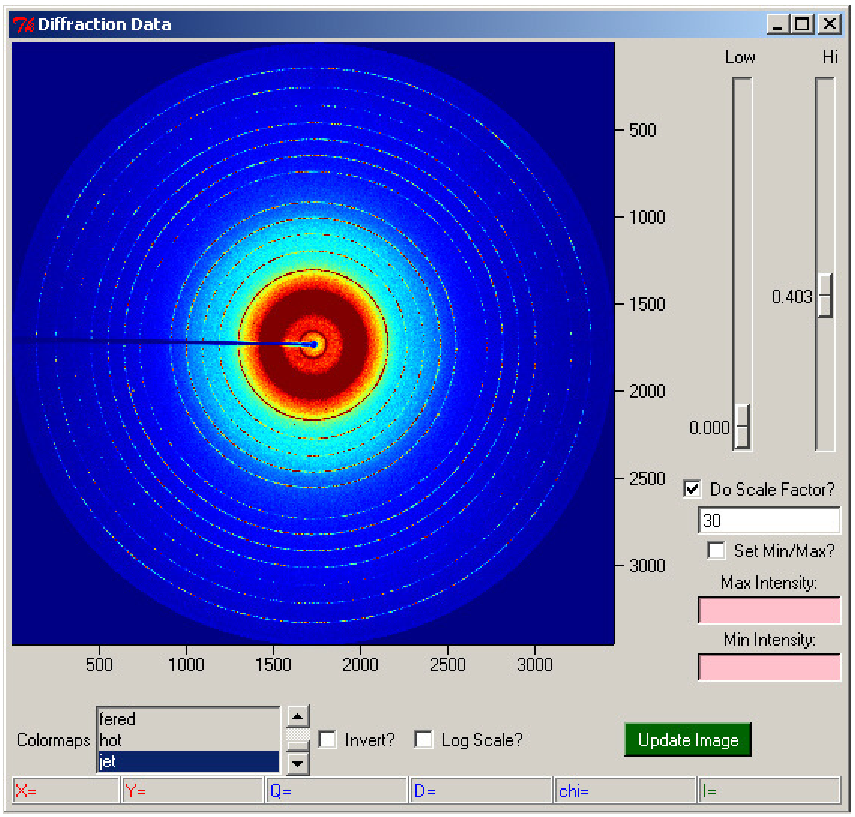


Figure 3.2: This diffraction data window will open after a file is loaded.

- *Log Scaling* - By default, intensity values are linearly mapped to colors in the color map. The "`Log Scale?`" check box can be used to instead apply a log scale mapping of the intensity values to the color map.

- *Low & Hi Pixels* – The sliders to the right of the image can be used to change the intensity scale of the image. The low value corresponds to the percentage of the most intensity pixel value that is mapped to the lowest part of the color map and the hi value corresponds to the percentage of the most intense pixel value that is mapped to the highest part of the color map. This feature is can make more visible certain intensity ranges in the image.

- *"Do Scale Factor" and "Set Min/Max?"* – Version 2 of the program introduced the feature "`Do Scale Factor?`" and "`Set Min/Max?`". These can be used to override the minimum and maximum pixel value used by the Low and Hi sliders. The scale factor will simply divide the maximum intensity by this value. So if the intensity scale by default goes from 0 to 10000, a scale factor of 10 will make the scale instead go from 0 to 1000. This feature is especially useful when just a few pixels have incredibly large values but the rest have very small values. The slides still let you pick a range between 0 and 1000, the scale factor simply change what maximum value the slider will correspond to. The "`Set Min/Max?`" option can be used instead to set directly the minimum and maximum intensity. To set these persistently, use the "`Set As`

`Initialization`" option described in section 9.3.

3.1 File Formats

The program can load the Mar data "`.mar2300`", "`.mar3450`", and the "`.mccd`" Mar CCD format It can load standard "`.tiff`" data. and the ESRF Data Format "`.edf`". It can load in Bruker data with extension "`.gfrm`" and "`.sfrm`".

The program can only deal with square data so whenever non-square data is loaded, the data will be padded out with blank values until the data is square.

3.2 Clipping High Pixels

The program stores the diffraction data inside the program as 4 byte integers and the largest value that can be stored as an integer is $2^{31} - 1 = 2147483647$. Any intensity values read from diffraction data with value larger then this number will be set to this number when stored in the program. In practice, this should never be a problem because intensities never range this high.

3.3 Loading Multiple Images

The "`Data File:`" input can be used to load multiple files at the same time. If multiple files are typed into the "`Data File:`" input and are separated by spaces, they will all be loaded in. Alternately, the diffraction data file selector can be used to select and load multiple files. The program will add the intensities of the images pixel by pixel and work with the combined image. The program can only add files of the same format.

3.4 Saving Diffraction Data

Diffraction data can be saved as several popular image formats. The data can be saved from the "`File`" menu using the "`Save Image`" option. The formats currently allowed are "`jpg`", "`gif`", "`eps`", "`pdf`", "`bmp`", "`png`", "`tiff`", and the ESRF data format "`edf`".

Images saved as a popular image format will contain whatever threshold masks, polygon masks, Q lines, ΔQ lines, and peaks are currently displayed and they will be saved at whatever the current zoom level is.[1] See chapter 5 for a discussion of the Q lines, ΔQ lines, and peaks. See chapter 6 for a discussion of threshold masks and polygon masks.

Because the program will pad any non-square data when it is loaded, the program will always save square images. If this is undesirable, the saved images will need to be cropped using another program.

[1]This is not the case with ESRF data. ESRF data will be saved unzoomed with none of the lines or masks on top of it.

CHAPTER 4

DETECTOR GEOMETRIES

X-ray diffraction can be modeled as in figure 4.1. Cones of light preferentially leave a crystal and Usually, the interesting thing to measure with x-ray diffraction is the scattering angle 2θ. If a detector was placed perpendicular to the incoming beam, the cones of light would be detected as circles of high intensity. If the distance d from the sample to the detector was and the distance r from the center of the detector to a particular ring were known, the scattering angle could be calculated as

$$\tan 2\theta = \frac{r}{d}. \tag{4.1}$$

This is shown in figure 4.2. Life is not always so simple. The detector is never exactly perpendicular to the incoming beam. In practice, the detector will always be slightly titled with respect to the incoming beam. Failing to account for this would introduce a systematic error in the angle measurements.

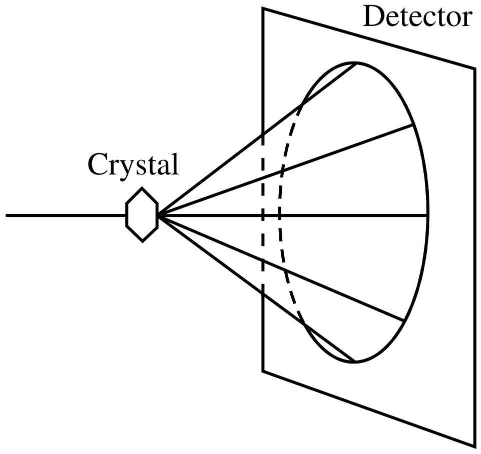


Figure 4.1: An X-Ray diffraction setup. X-rays scatter from a sample and are captured by a detector.

There is a need to analyze diffraction data on detectors that are not perpendicular to the incoming x-rays. We will present a theory of tilted detectors first developed by Abhik

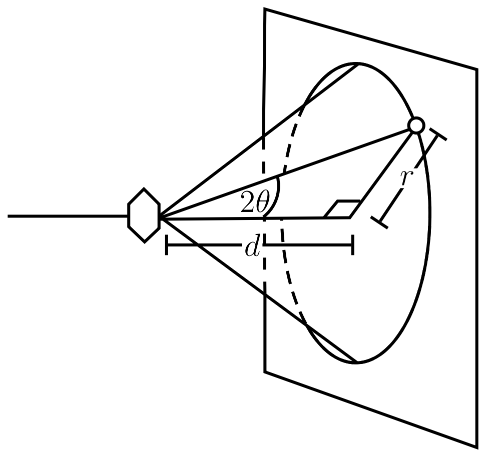


Figure 4.2: The same setup as in figure 4.1. 2θ is the scattering angle of the light, d is the distance from the crystal to the detector, and r is the distance from the center of the detector.

Kumar[4]. Our derivation will result in different formulas because of different assumptions about how the detector is tilted. This is actually a very important point because version 1 of the program uses Abhik's formulas for geometric transforms whereas the equations derived in the manual were only implemented in version 2 of the program. Section 4.8 discusses this situation in more detail.

We are interested in relating coordinates on a tilted detector to theoretically motivated quantities such as the scattering angle of the beam that hit the point. We must first work out the transformation of points on a tilted detector to points on an untilted detector. This is to say that we want to figure out where on an untilted detector a beam would have hit were it to hit the untilted detector instead of the tilted detector. We will call the point on the untilted detector as measured on the untilted detector (x, y) and the corresponding point on the tilted detector as measured on the tilted detector (x''', y'''). The reason for the three primes will become obvious shortly. This is shown in figure 4.3. Another way to think about this is to imagine putting your head at the crystal and looking directly at some point on a real detector. We want to figure out the corresponding point that would be seen be looking at the imagined untilted detector.

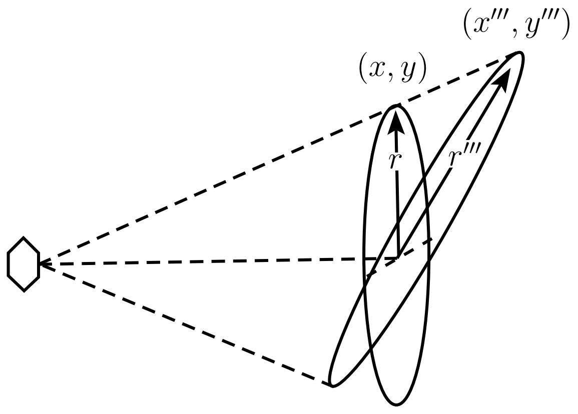
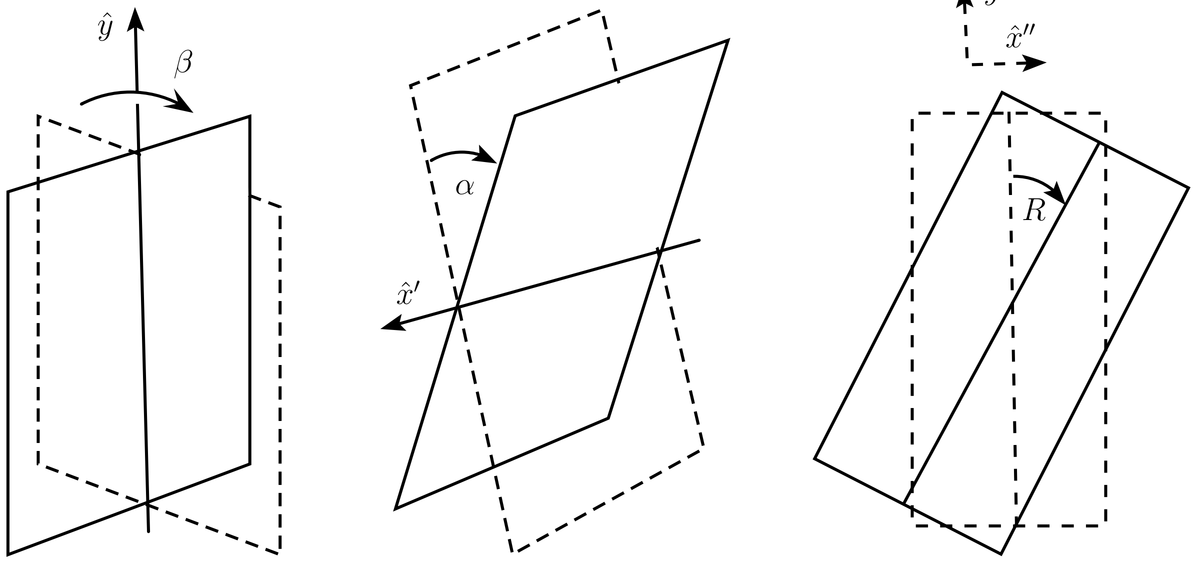


Figure 4.3: The detector is titled with respect to the incoming beam. We are interested in relating the point on the tilted detector (x''', y''') to the point (x, y) on the imagined untilted detector.

4.1 The Three Rotation Angels

In order to relate these points, we need to find a way to describe an arbitrary tilt. We will characterize a titled detector as an untitled detector that has 3 successive rotations applied to it. We will first rotate the detector about the detector's y axis. We will then rotate the detector about the detectors new x' axis. Finally, we will rotate the detector about a vector normal to the detector going through its center. These rotations are shown in figure 4.4. We can solve our original problem if we deal with each rotation separately.



(a) β characterizes a rotation around the $\hat{y}$ axis.

(b) α characterizes a subsequent rotation around the new $\hat{x}'$ axis.

(c) R characterizes the subsequent rotation about a vector normal to the plane.

Figure 4.4: Any detector tilt can be characterized by three successive rotations.

4.2 The β Rotation

We first apply to our untitled detector a rotation around $\hat{y}$ by angle β. We begin with a point (x, y) on an untilted detector and relate it to its corresponding point (x', y') on the rotated detector. A diagram of this is shown in figure 4.5. We can use the geometry of these diagrams to relate these coordinates. Using similar triangles:

$$\frac{x}{d} = \frac{x' \cos\beta}{d + x' \sin\beta}. \tag{4.2}$$

Or

$$\boxed{x = \frac{dx' \cos\beta}{d + x' \sin\beta}}. \tag{4.3}$$

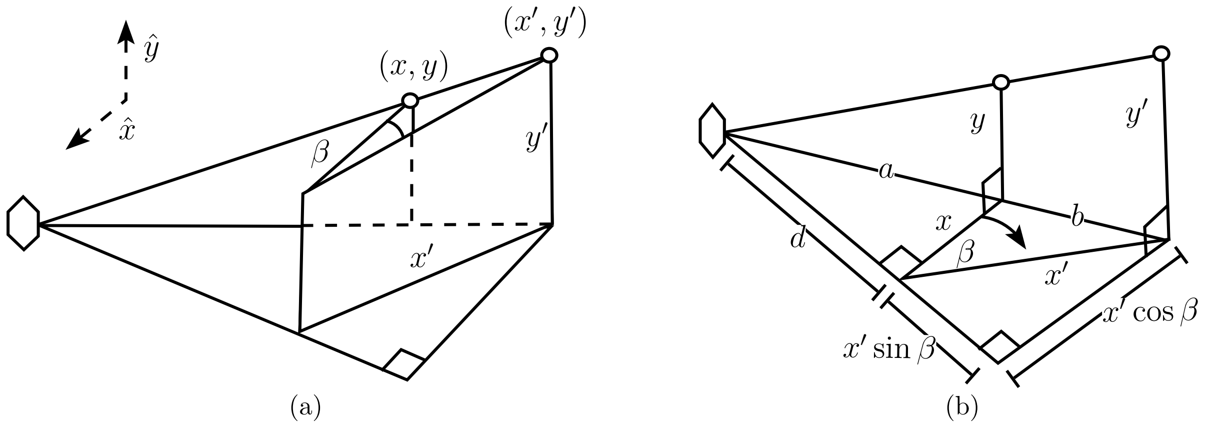


Figure 4.5: The untilted detector is rotated by β around $\hat{y}$.

Again, we have

$$\frac{y}{a} = \frac{y'}{a+b} \tag{4.4}$$

$$\frac{d}{a} = \frac{d + x' \sin\beta}{a+b}. \tag{4.5}$$

So

$$\boxed{y = \frac{dy'}{d + x' \sin\beta}}. \tag{4.6}$$

Equation 4.3 and 4.6 are the equations relating a point on an untilted detector to the corresponding point on a tilted detector.

4.3 The α Rotation

We take the point (x', y') on the plane rotated by β around $\hat{y}$ and relate it to the point (x'', y'') on a plane then rotated by α around $\hat{x}'$. Figure 4.6 is a diagram this plane. Figure 4.7 is a more geometrical diagram and figure 4.8 shows a cross section of the $y = 0$ plane.

From figure 4.8, we see that $f = y'' \sin\alpha \cos\beta$. From figure 4.7, we see that $h = y'' \cos\alpha$. Using similar triangles:

$$\frac{y}{a} = \frac{y'' \cos\alpha}{a+b+c}. \tag{4.7}$$

Again:

$$\frac{d}{a} = \frac{d + x'' \sin\beta + f}{a+b+c}. \tag{4.8}$$

We deduce that

$$\boxed{y = \frac{dy'' \cos\alpha}{d + x'' \sin\beta + y'' \sin\alpha \cos\beta}} \tag{4.9}$$

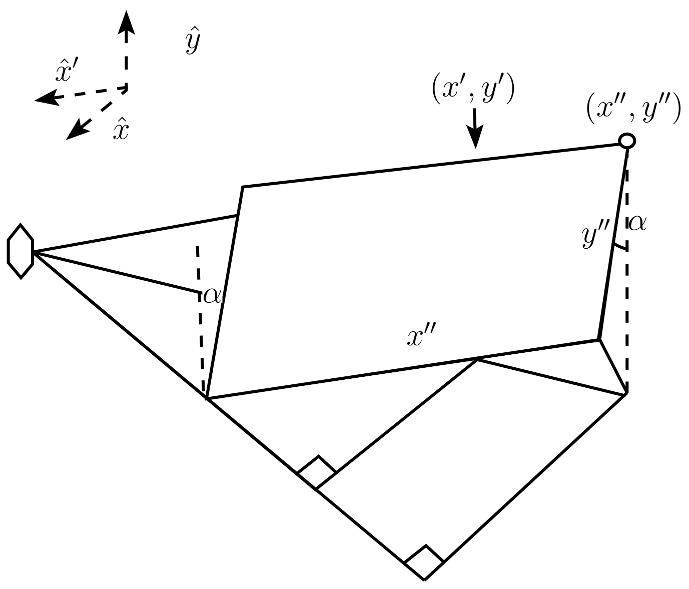


Figure 4.6: A diagram of a plane that has been rotate by angle β about the $\hat{y}$ axis and then by α about the $\hat{x}'$ axis.

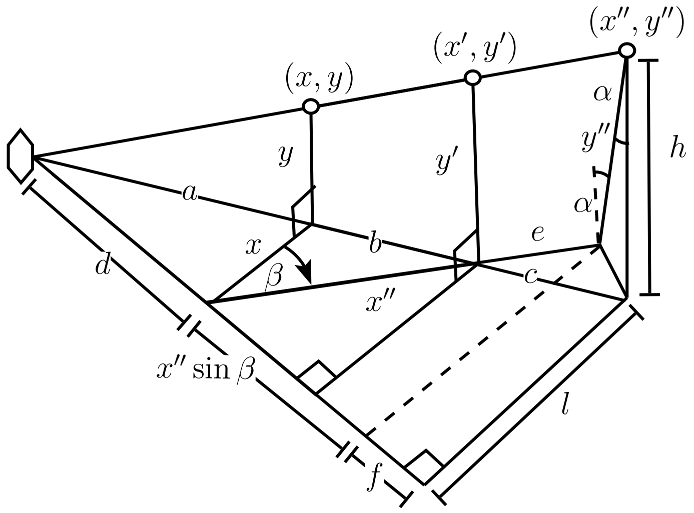


Figure 4.7: A more geometrical diagram of figure 4.6.

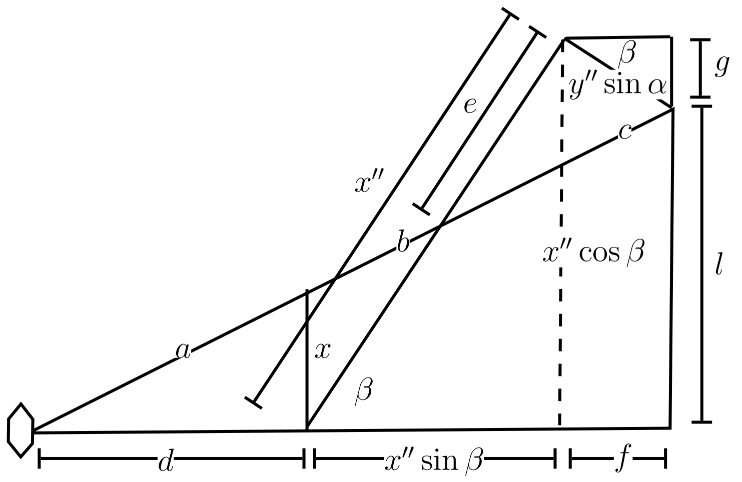


Figure 4.8: A cross section of the $y = 0$ plane.

Figure 4.8 shows that $g = y'' \sin\alpha \sin\beta$ and that $x'' \cos\alpha = l + g$. Using similar triangles again

$$\frac{x}{d} = \frac{l}{d + x'' \sin\beta + y'' \sin\alpha \cos\beta} \tag{4.10}$$

Plugging in and simplifying

$$\boxed{x = \frac{d(x'' \cos\beta - y'' \sin\alpha)}{d + x'' \sin\beta + y'' \sin\alpha \cos\beta}} \tag{4.11}$$

4.4 The R Rotation

The final rotation is by angle R about a vector normal to the plane. We relate the coordinate (x'', y'') on the previously rotated detector to the point (x''', y''') on a detector then rotated about its center:

$$x'' = x''' \cos R + y''' \cos R \tag{4.12}$$

$$y'' = y''' \cos R - x''' \cos R \tag{4.13}$$

Applying equation 4.12 and 4.13 to equation 4.11 and 4.9 gives us the needed relationship between a point on an untilted detector and the corresponding point on a tilted detector.

4.5 Pixel Coordinates

(x''', y''') represents the distance between the center and a point on the real detector. We do not actually measure this distance. What we experimentally measure are pixel coordinates on a detector. There is some pixel coordinate corresponding to the center of the detector[1]. We will call it (x_c, y_c). We will call (x_d, y_d) the pixel coordinate on the detector that corresponds to the point (x''', y'''). We want to relate (x_d, y_d) to (x''', y'''). There is some material property of the detector called the pixel length (pl) that describes the width of each pixel (e.g. 1000 mm/pixel). There is a corresponding quantity ph that is the height of each pixel. We can relate the pixel coordinates to the physical distances by

$$x''' = (x_d - x_c) \times pl, \qquad y''' = (y_d - y_c) \times ph. \tag{4.14}$$

4.6 Inverting the Equations

We can invert these formula to learn what x'' and y'' are in terms of x and y. We have:

$$x'' = \frac{dx}{d\cos\beta - x\sin\beta - \cos\beta(x\cos\beta + d)/(\frac{x}{y}\cot\alpha + 1)} \tag{4.15}$$

[1]This is the point that the x-rays would hit were they not to diffract off the crystal

and

$$y'' = \frac{dx\cos\beta/(\frac{x}{y}\cos\alpha + \sin\alpha)}{d\cos\beta - x\sin\beta - \cos\beta(x\cos\beta + d)/(\frac{x}{y}\cot\alpha + 1)}. \tag{4.16}$$

4.7 Q, 2θ, and χ

We now have a way to relate (x''', y'''), a point on a detector rotated by angle β, α, and R to a point (x, y) on an untilted detector. We now relate (x, y) to theoretically motivated quantities. Figure 4.9 shows that 2θ, the angle of scattering of a beam, is

$$\tan 2\theta = \frac{r}{d} = \frac{\sqrt{x^2 + y^2}}{d}. \tag{4.17}$$

χ, the angle around the beam, is calculated as[2]

$$\tan\chi = \frac{y}{x}. \tag{4.18}$$

Q is calculated as

$$Q = 4\pi\sin(2\theta/2)/\lambda. \tag{4.19}$$

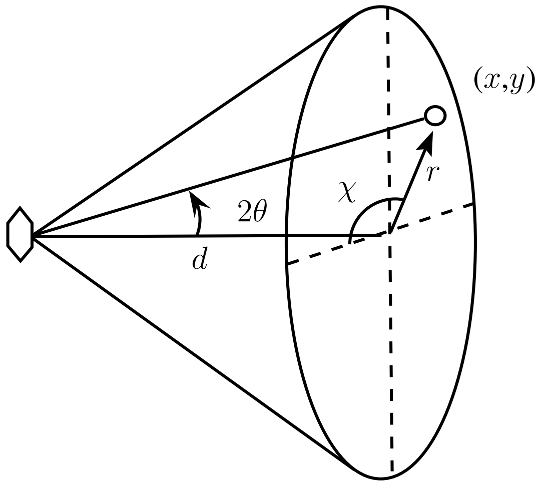


Figure 4.9: For a particular point (x, y) on an untilted detector, we define 2θ as the angle of scattering of the beam and χ as the azimuthal angle of the scattered light around the beam.

We could use incoming beam's energy instead of its wavelength using De Broglie's formula

$$E = hc/\lambda. \tag{4.20}$$

[2]Actually, it is a little more complicated then just this formula. Really, when viewing the diffraction pattern from where the sample is, we want χ to equal 0 pointing directly to the right and we want for it to increase in a counterclockwise direction. This is just like the definition of θ in Cartesian coordinates. To make $\chi = 0$ to the right, we need to account for the fact that a rotation by angle R will offset where χ equals 0 by adding the value of R to χ. Then, our previous definition of χ makes χ increase in a clockwise direction. (The reason is because we define y to increase as you go down the detector). To make χ increase in the counterclockwise direction, we need to multiply by negative 1. When we do this, our real χ is related to the chi derived above by $-1 \times (\chi + R)$ It would probably be cleaner to get rid of this final multiplication by changing the definition so that y increased as you go up the detector, but this way is good enough.

Some people use the quantity D instead of Q

$$D = 2\pi/Q. \tag{4.21}$$

If we knew x_c, y_c, pl, ph, d, λ, α, β, and R for an experiment, could relate pixel coordinates (x_d, y_d) read directly off of a detector to the theoretically motivated quantities Q and χ. A discussion of how these detector parameters can be determined experimentally is given in section 5.

4.8 A Note About Versions

Version 1 of the program uses the equations derived by Abhik in [4]. For reference, Abhik's derivation is different in that equation 4.9 and 4.11 are replaced with the equations

$$x = \frac{dx'' cos\beta}{d + y'' \sin\alpha + x'' \sin\beta}, \qquad y = \frac{dy''' \cos\alpha}{d + y'' \sin\alpha + x'' \sin\beta}. \tag{4.22}$$

These equations must also be inverted to replace 4.15 and 4.16. But other that, the rest of the discussion above is exactly the same using Abhiks' method.

Beginning with version 2 of the program, equation 4.9 and 4.11 derived earlier are used for the transforms. So technically, version 1 and version 2 of the program are not interchangeable and certain quantities calculated with one of the versions of the program, such as calibration parameters, can not be used in the other version of the program. But in practice, both sets of equations reduce to each other in the small α and β limit which is almost always the case when performing powder diffraction. So it is very unlikely that this switch between version 1 and version 2 would cause any difficulty.

4.9 Implementation In Code

For reference, this section present the C source code used in the computer program to perform the transformation from pixel values on a tilted detector to (Q, χ) and back again. This code uses the equations derived in this manual for the geometric transforms and the code was introduced in version 2 of the program. Version 1 uses different code using Abhik's formulas for the geometric transform.

Below is the function "`getTwoThetaChi()`" which converts real pixel coordinates (x_d, y_d), called "`xPixel`" and "`yPixel`" on the detector, into the values 2θ and χ, called "`twoTheta`" and "`chi`" using the equations described above.

```
/*
 * This transformation is done using Josh's equations derived
 * in the software manual. The notation in the manual
 * corresponds to the variable names by:
 *   xMeasured = x'''
 *   yMeasured = y'''
 *   xPhysical = x_d
 *   yPhysical = y_d
 */
void getTwoThetaChi(double xCenter,double yCenter,
        double distance,double xPixel,double yPixel,
        double pixelLength,double pixelHeight,
        double rotation,double cos_beta,double sin_beta,
        double cos_alpha,double sin_alpha,
        double cos_rotation,double sin_rotation,
        double *twoTheta,double *chi) {

    double pixelLength_mm,pixelHeight_mm;
    double xMeasured,yMeasured;
    double bottom;
    double xPhysical,yPhysical;

    // pixellength comes in in units of micron
    // We convert pixelLength & pixelHeight into mm units so
    // that they are comparable with distance (in units of
    //  millimeters)
    pixelLength_mm = pixelLength/1000.0;
    pixelHeight_mm = pixelHeight/1000.0;

    xMeasured = (xPixel-xCenter)*pixelLength_mm;
    yMeasured = (yPixel-yCenter)*pixelHeight_mm;

    bottom = distance+(xMeasured*cos_rotation+
            yMeasured*sin_rotation)*sin_beta+
            (-xMeasured*sin_rotation+
            yMeasured*cos_rotation)*sin_alpha*cos_beta;

    // calculate the x y coordinates on the imaginary
    // detector using fancy math
    xPhysical = distance*((xMeasured*cos_rotation+
            yMeasured*sin_rotation)*cos_beta
            -(-xMeasured*sin_rotation+
            yMeasured*cos_rotation)*sin_alpha)/bottom;

    yPhysical = distance*(-xMeasured*sin_rotation+
            yMeasured*cos_rotation)*cos_alpha/bottom;

    *twoTheta = atan2(sqrt(xPhysical*xPhysical+
            yPhysical*yPhysical),distance);
    // Convert to radians
    *twoTheta = *twoTheta * 180.0/PI;

    // explicitly convert chi to degrees
    *chi=atan2(yPhysical,xPhysical)*180.0/PI;

    // then add rotation to it so that chi always points
    // to the right. Also, we have to multiply chi by -1
    // because we have been defining our angles the inverse
    // of the way they should be. There is a probably a
    // better way to do this if I really thought through
    // exactly how chi is defined. For the moment, through,
    // this does exactly the right thing.
    *chi = (*chi + rotation)*(-1);

    // make sure that chi is b/n 0 and 360
    *chi = mod(*chi, 360.0);
}
```

As described above, the program must be given the values x_c, y_c, ps, d, α, β, and R in order to perform the transformations. For x_c and y_c, the function takes in the variables *xCenter* and *yCenter*. The program takes in "pixelLength" and "pixelHeight" for pl and ph. The program takes in the sines and cosines of α, β, and R so that the sin of α can be cached for increased efficiency. This function calculates 2θ instead of Q or D but it is trivial to do the conversion.

The listing below presents the function "getXY()" which does the inverse transformation of the above function. It uses the same terminology for parameters above:

```
/*
 * This transformation is done using Josh's equations derived
 * in the software manual. The notation in the manual
 * corresponds to the variable names by:
 *   xMeasured = x'''
 *   yMeasured = y'''
 *   xPhysical = x_d
 *   yPhysical = y_d
 */
void getXY(double xCenter,double yCenter,double distance,
        double energy,double q,double chi,double pixelLength,
        double pixelHeight,double rotation,double cos_beta,
        double sin_beta,double cos_alpha,double sin_alpha,
        double cos_rotation,double sin_rotation,
        double * xPixel,double * yPixel) {

    double wavelength;
    double twoTheta;
    double xPhysical,yPhysical;
    double bottom;
    double pixelLength_mm,pixelHeight_mm;
    double xMeasured,yMeasured;

    double tan_chi;

    wavelength = 12398.4172/energy;

    // rotate chi back to point whatever way it is supposed
    // to point
    chi = chi*(-1) - rotation;

    chi = mod(chi, 360.0);

    // explicitly convert chi to radians
    chi*=PI/180.0;

    twoTheta = 2.0*asin(wavelength*q/(4.0*PI));

    tan_chi = tan(chi);
    xPhysical = fabs(distance*tan(twoTheta)/sqrt(1.0+
            tan_chi*tan_chi));

    // one must determine explicitly the sign of xPhysical
    // by inspecting the diagram. This is b/c at one point
    // we take a sqrt in our derivation.
    if (chi>PI/2.0 && chi<3.0*PI/2.0)
        xPhysical = -1.0*xPhysical;

    yPhysical=fabs(xPhysical*tan_chi);

    // set the sign of y explicitly
    if (chi > PI && chi < 2.0*PI)
        yPhysical = -1.0*fabs(yPhysical);

    // I should worry about cos_alpha being 0
    bottom = distance*cos_beta-xPhysical*sin_beta-
            cos_beta*(xPhysical*cos_beta+distance)/(
            (xPhysical*cos_alpha)/(yPhysical*sin_alpha)+1);

    xMeasured = (distance*xPhysical*cos_rotation-
            distance*xPhysical*cos_beta*sin_rotation/
            (xPhysical*cos_alpha/yPhysical+sin_alpha))/
            bottom;

    yMeasured = (distance*xPhysical*sin_rotation+
            distance*xPhysical*cos_beta*cos_rotation/
            (xPhysical*cos_alpha/yPhysical+sin_alpha))/
            bottom;

    // convert pixelLength & pixelHeight into mm units so
    // that they are comparable with distance (in units of
    // millimeters)
    pixelLength_mm = pixelLength/1000.0;
    pixelHeight_mm = pixelHeight/1000.0;

    *xPixel = xMeasured/pixelLength_mm + xCenter;
    *yPixel = yMeasured/pixelHeight_mm + yCenter;
}
```

CHAPTER 5

CALIBRATION

Area Diffraction data can only be analyzed with a knowledge of how the diffraction experiment was set up. Calibration is the process used to find these values.

5.1 The Calibration Algorithm

In principle, all the calibration values could be measured directly from a diffraction experiment. In practice, they can not be measured to an acceptable level of precision. Instead, a calibration procedure is used to infer these values from the diffraction data. Calibration is done by imaging a calibration standard before imaging an interesting sample. Assuming that the diffraction machine was not altered between the imaging of the calibration standard and the imaging of the unknown sample, the calibration parameters for both sets of data will be the same. If the calibration values could be determined from the pattern of the standard crystal, this information could be used to analyze the other data. What it means to use a standard crystal is to know in advance its Q values of preferential scattering. With these and with the calibration parameters, The diffraction pattern that should show up on the detector can be calculated by, for each Q value, varying χ and calculating the corresponding (x_d, y_d) pair. The program can do this. If a set of Q values and calibration parameters is enter into the program, the "`Draw Q Values?`" check box will draw on top of the diffraction image what pattern should show up on the detector. This feature described in section 5.8.

In order to perform the calibration, the calibration parameters are varied until they make the calculated pattern match the pattern that was captured. The calibration parameters are fit to the diffraction pattern.

5.2 Fitting

In order for the fitting algorithm to work, the program must already have an initial guess at the calibration parameters, a list of the known Q values, and a range for each of these Q values. Inside of this Q range (as calculated by the initial calibration values) there must uniquely be the diffraction peak. An example of this is shown in figure 10.1.

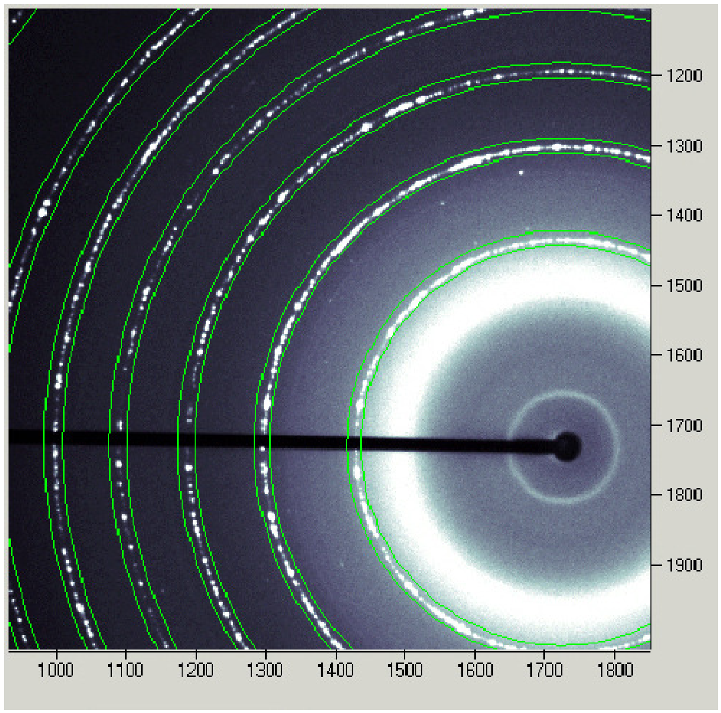


Figure 5.1: The diffraction image is divided into Q ranges. Each diffraction peak falls uniquely into one Q range.

The fitting algorithm first finds (x, y) coordinates for many diffraction peaks. To do so, it picks a χ value and spread radially out from the center of the diffraction image. The program makes an array of the intensity values in each range. It then fits a Gaussian to the ranges and the peak of the Gaussian is taken as the diffraction peak peak. Figure 5.2 shows a diagram of this. This is done for many evenly spaced χ values. Section 5.5 describes how the number of χ slices can be set by the user.

There is not always a consistent diffraction ring all the way around the image. Some of the Gaussian fits are not real peaks and should be ignored. To remove these spurious peaks, the program applies several tests to each peak. The first test is that the Gaussian fit was not too close to the edge of the image. Any peak whose fit center plus or minus twice the fit's standard deviation is outside of the Q range is considered too close to the edge of the image. Next, the program tests if the height of the peak divided by the standard deviation of the data outside of the peak is smaller than a certain value. If so, the peak is not considered to be significant enough. This value is called "`Stddev`" and can be specified by the user. See section 5.5. The higher the value of "`Stddev`", the more picky the program is about allowing peaks.

After making a list of peaks, the program defines a residual function which describes how

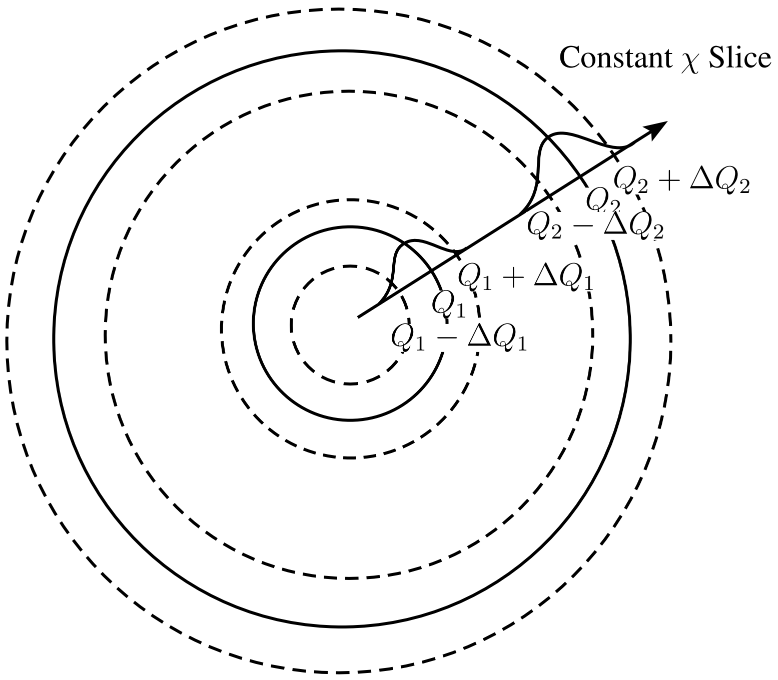


Figure 5.2: This is a diagram of the peak finding algorithm. The solid black circles represent diffraction peaks. The dotted lines represent the Q ranges. The radial line represents the program picking a particular χ value and looking for peaks along it. The program fits a Gaussian to the intensity profile inside each of the ranges to find diffraction peaks.

good the current calibration parameters are. It converts the (x, y) coordinate of each peak into a $(Q^{\text{peak}}, \chi^{\text{peak}})$ pair. Since each peak has a known Q value (which we will call Q^{exp}), we define the residual function as

$$\text{Residual}(x_c, y_c, d, \lambda, \alpha, \beta, R) = \sum_{x,y \text{ pairs}} (Q^{\text{peak}} - Q^{\text{exp}})^2 \tag{5.1}$$

The functional dependence comes from calculating Q^{peak} for particular calibration parameters. The smaller the residual, the closer the calibration parameters are to a perfect calibration.

The program takes this function of 7 variables (the calibration parameters) and minimizes it. Ideally, the calibration parameters that minimize the residual are the best guess at the calibration parameters. The program uses the Levenberg-Marquardt nonlinear least squares minimization algorithm to minimize the residual. The particular implementation used by the program is Manolis Lourakis's levmar library[5].

5.3 Calibrating With the Program

Calibration is done with the "`Calibration`" tab shown in figure 3.1 on page 20. An image can be calibrated after a diffraction image of a standard crystal has been loaded, the Q values for that crystal have been loaded, and an initial guess at the calibration parameters have been loaded.

The "`Do Fit`" button can be used to refine the calibration parameters using the calibration algorithm described in section 5.1. The best guess for the calibration parameters will be put in the inputs on the "`Calibration`" tab.

The program will print to the console some useful things while it does the calibration. The program will calculate the residual function (equation 5.1) before and after the calibration and print the value to the console[1]. The output will look like

```
 - Before fitting, the calculated residual is 5.336138e-04
 - Doing the fitting
 - After fitting, the calculated residual is 6.532131e-06
```

The program will display the reason why the fitting algorithm decided to quit. For example, the program might print out

```
 - Reason for quitting the fit: 2-stopped by small gradient J^T e
```

These reason are told to the program by the fitting algorithm and they are described on the levmar website (http://www.ics.forth.gr/~lourakis/levmar/). That site says that the different reasons why the fitting can stop are:

- stopped by small gradient J^T e
- stopped by small Dp
- stopped by itmax
- start from current p with increased \mu
- no further error reduction is possible. Restart with increased mu
- stopped by small ||e||_2[5]

Stopped by small gradient means that the program found its way to the bottom of the hill and is convinced that it did its best job minimizing the function. Stopped by itmax means that the program was forced to quit by a hard coded limit on the number of loops through the fitting. The fit should probably be done again to find the best guess. I don't know enough about the levmar fitting algorithm to know what the other messages really mean. If you need to know, you should read the fitting algorithm's documentation.

The fitting algorithm also displays a covariance matrix that it calculates while fitting. I do now know how it calculates this and I do not know what it means physically. The program print it out after the fitting.

```
Covariance Matrix:
[[ 9.47e-04 -1.76e-04  6.11e-05  3.75e-03  6.37e-06 -3.01e-05  2.45e-02]
 [-1.76e-04  1.24e-03 -1.66e-04 -1.02e-02 -2.58e-05 -5.24e-05 -1.24e-02]
 [ 6.11e-05 -1.66e-04  2.15e-04  1.44e-02  9.19e-06  3.28e-06 -1.52e-03]
 [ 3.75e-03 -1.02e-02  1.44e-02  9.85e-01  5.90e-04  2.10e-04 -1.03e-01]
 [ 6.37e-06 -2.58e-05  9.19e-06  5.90e-04  9.44e-06  1.85e-07 -4.43e-03]
 [-3.01e-05 -5.24e-05  3.28e-06  2.10e-04  1.85e-07  6.70e-06  6.77e-05]
 [ 2.45e-02 -1.24e-02 -1.52e-03 -1.03e-01 -4.43e-03  6.77e-05  4.00e+00]]
```

[1]Actually, the program calculates the residual per peak. It is a more useful quantity because it is independent of the number of peaks found.

The rows (from top to bottom and from left to right) correspond to "`xc`", "`yc`", "`d`" "`E`", "`alpha`", "`beta`", and "`rotation`". I think that the square root of the diagonal elements of the covariance matrix are supposed to correspond to uncertainties, but I do not know enough about the minimization algorithm to really be comfortable saying that these values are meaningful. The program print out the square root of the diagonals.

```
Root of the diagonal of the covariance matrix (I think these are uncertainties)
 - xc         0.0307807389
 - yc         0.0352730417
 - d          0.0146853032
 - E          0.9928960497
 - alpha      0.0030735910
 - beta       0.0025888303
 - R          2.0017867847
```

The program can undo to the previous calibration parameters using the "`Previous Values`" input.

5.4 The "`Save Last Fit`" Button

Often times, it is desirable to save to a file all of the information about a particular calibration that has been done. Version 2 of the program introduces a convenient button called "`Save Last Fit`" that does just that. After calibrating, pushing this button and then selecting an output file name will save all of the information about the previous calibration to a file. This includes the residual before and after, and the covariance matrix, the initial and final calibration parameters, and the Q list used when calibrating. An example is:

```
Fit done of: C:/work/LaB6_14_02_56.mar3450

Initial Guess at calibration parameters:
xc       1725.0000000000 0
yc       1725.0000000000 0
D         125.2960000000 0
E       12735.3957721000 0
alpha       0.0000000000 0
beta        0.0000000000 0
R           0.0000000000 0
pl        100.0000000000
ph        100.0000000000

Before fitting, the calculated residual per peak is 0.000533613563184

Refined calibration parameters:
xc       1723.2670576589 0
yc       1729.0275947978 0
D         123.9263391635 0
E       12712.8416444082 0
alpha       0.0043942864 0
beta        0.1124327684 0
```

```
R          52.4701069289 0
pl        100.0000000000
ph        100.0000000000

After fitting, the calculated residual per peak is 6.52739855241e-006

Reason for quitting the fit: 2-stopped by small gradient J^T e

Covariance Matrix:
[[ 9.47e-04 -1.76e-04  6.11e-05  3.75e-03  6.37e-06 -3.01e-05  2.45e-02]
 [-1.76e-04  1.24e-03 -1.66e-04 -1.02e-02 -2.58e-05 -5.24e-05 -1.24e-02]
 [ 6.11e-05 -1.66e-04  2.15e-04  1.44e-02  9.19e-06  3.28e-06 -1.52e-03]
 [ 3.75e-03 -1.02e-02  1.44e-02  9.85e-01  5.90e-04  2.10e-04 -1.03e-01]
 [ 6.37e-06 -2.58e-05  9.19e-06  5.90e-04  9.44e-06  1.85e-07 -4.43e-03]
 [-3.01e-05 -5.24e-05  3.28e-06  2.10e-04  1.85e-07  6.70e-06  6.77e-05]
 [ 2.45e-02 -1.24e-02 -1.52e-03 -1.03e-01 -4.43e-03  6.77e-05  4.00e+00]]

Root of the diagonal of the covariance matrix (I think these are uncertainties)
xc          0.0307807389
yc          0.0352730417
d           0.0146853032
E           0.9928960497
alpha       0.0030735910
beta        0.0025888303
R           2.0017867847

Q Data used when calibrating:
            Q          Delta Q
  1.5115438090      0.0500000000
  2.1376468230      0.0500000000
  2.6181029660      0.0500000000
  3.0230876190      0.0500000000
  3.3798737530      0.0500000000
  3.7025252250      0.0500000000
```

5.5 “Number of Chi?” and “Stddev?”

The calibration algorithm requires starting at the center and moving across the image in constant χ slices (see section 5.1). The number of these slices can be set using the “Number Of Chi?” input. The default value is 360.

Section 5.2 describes the “Stddev” parameter. It can be set using the “Stddev?” input. The default value is 5. The higher the value, the more likely it is that the program will ignore bad peaks but the more likely it is that the program will ignore valid peaks.

5.6 Work in λ

Often, it is desirable to work with the wavelength of the beam instead of its energy. Of course, these values are related by

$$E = hc/\lambda. \tag{5.2}$$

The "`Work in Lambda`" radio select in the menu bar can be used to work with wavelength in units of nanometers, instead of energy, in units of electron volts. This can be changed back using the "`Work in eV`" input. When the program switches to working with wavelength, the energy input "`E`" will be relabeled "λ" and the value in the input will be converted. When working with wavelength, the wavelength will be saved to the file instead of the energy when saving experimental parameters.

5.7 Fixing Calibration Parameters

When fitting calibration parameters, it is not always desirable to allow the program to vary all of the parameters. For example, the energy of the beam might be well known already so it would be desirable to keep it fixed. The "`Fixed?`" check boxes next to the parameter inputs can be used to fix the corresponding parameter so that it doesn't vary. The pixel length and pixel height can not be fixed because they are never refined.

5.8 Displaying Constant Q Lines

After a diffraction file, a list of Q values, and calibration parameters have been loaded into the program, the program can display on top of the diffraction data the diffraction pattern that should show up for the particular Q lines and the particular calibration parameters. This can be done with the "`Draw Q Lines?`" button on the "`Calibration`" tab. Figure 5.3 shows an example. The way that this is done is described in section 5.1. Constant Q lines can also be drawn on top of the caked data. This is described in section 7.4. The color of the Q lines can be changed using the "`Color`" button next to the "`Draw Q Lines?`" button.

5.9 Displaying Constant ΔQ Lines

The program can also display on top of the diffraction data the Q range. The use of this range is described in section 5.2. This can be done with the "`Draw dQ Lines?`" button. The color of these lines can be changed with the corresponding "`Color`" button. Figure 5.4 shows an example. Constant ΔQ lines can also be displayed on top of caked data. This is described in section 7.4.

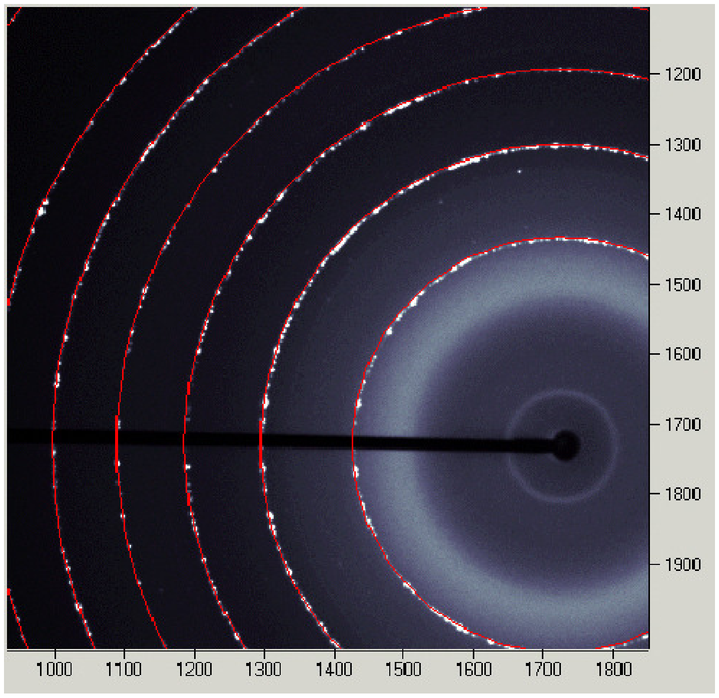


Figure 5.3: A diffraction image with constant Q lines displayed on it.

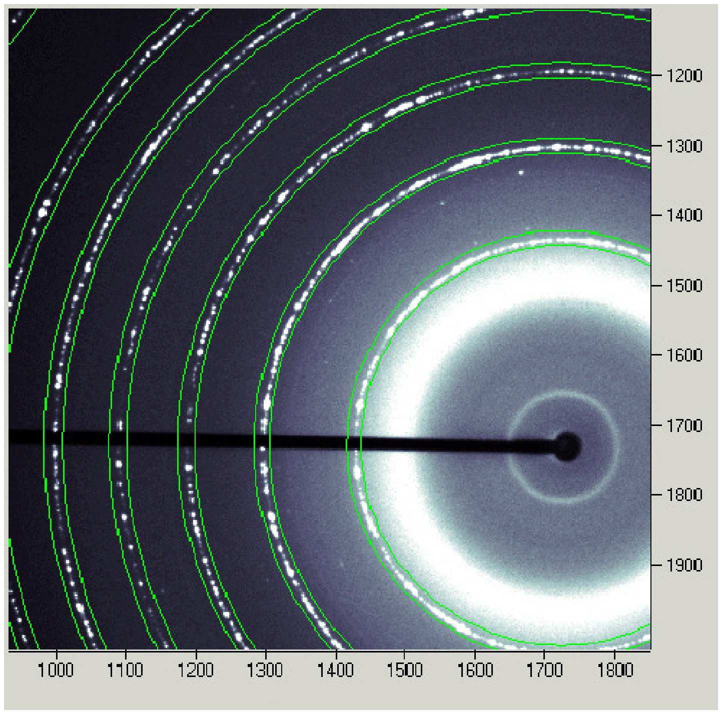


Figure 5.4: A diffraction image with the ΔQ range displayed on it.

5.10 Displaying Peaks

Section 5.2 describes how the program finds a list of diffraction peaks when it performs the image calibration. After the program finds the peaks, it can display them on top of the diffraction image. The "`Draw Peaks?`" check box can be used to display the peaks on the diffraction image as crosses. Figure 5.5 shows an example of this. This feature can be used to see if the program is finding real peaks. The color of the peaks can be changed with the corresponding "`Color`" button. Peaks can also be drawn on top of the caked data. This is described in section 7.4.

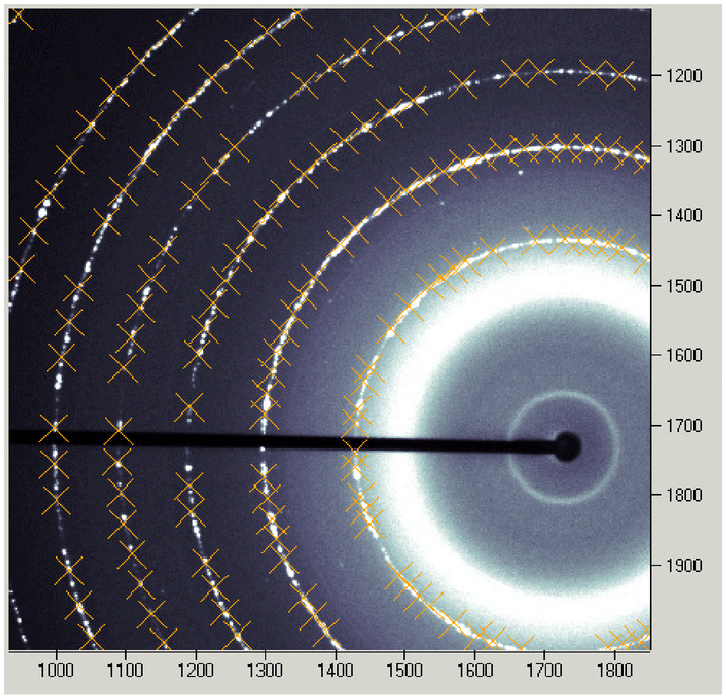


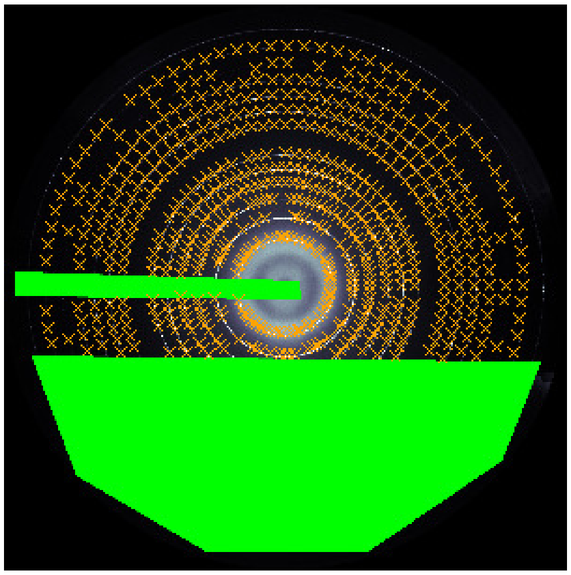
Figure 5.5: A diffraction image with diffraction peaks displayed on it.

5.11 Masking Peaks

Polygon masks can be used to stop the program from using peaks found in certain regions of the image. See chapter 6 for a discussion of masking. No peaks found within polygon masks will be used when calibrating the diffraction image. Figure 5.6a shows an example of this. Figure 5.6b shows the same effect on on a caked plot. In particular, see section 7.5 for more information on displaying peaks on a caked image. This feature was added in version 2.0.0.

5.12 Saving the Peak List

The program can make a list of diffraction peaks and save them to a file. If a diffraction image, a list of standard Q values, and calibration parameters are loaded in the program,


(a) A diffraction image with diffraction peaks and two polygon masks displayed on it.

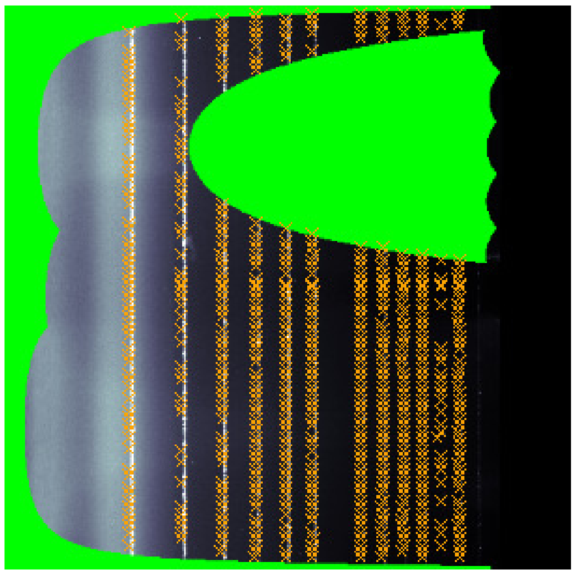

(b) A caked plot with diffraction peaks and two polygon masks displayed on top of it.

Figure 5.6: When a polygon mask is in the program, none of the peaks found in the mask will be used while calibrating the image.

the “`Make/Save Peak List`” button can be used to save out the peak list. A typical peak list file looks like

```
# Diffraction Data: C:/data/LaB6_14_02_56.mar3450
# Calculated on Mon Jun 09 23:41:58 2008
# Calibration data used to find peaks:
#   xc       1723.2670576600 pixels
#   yc       1729.0275948000 pixels
#   D         123.9263391640 mm
#   E       12712.8416444000 eV
#   alpha       0.0043942864 degrees
#   beta        0.1124327684 degrees
#   R          52.4701069289 degrees
#   pl         100.0000000000 microns
#   ph         100.0000000000 microns
# Q data used to find peaks:
# Peaks:
#                 Q            Delta Q
#     1.5115438090      0.0500000000
#     2.1376468230      0.0500000000
#     2.6181029660      0.0500000000
#     3.0230876190      0.0500000000
#     3.3798737530      0.0500000000
#     3.7025252250      0.0500000000
#     4.2751481980      0.0500000000
#     4.5346314280      0.0500000000
#       4.7799051400        0.0500000000
#       5.0133130990        0.0500000000
#       5.2360313900        0.0500000000
#       5.4498961800        0.0500000000
#                  x                     y                Real Q      ...
  2019.7834513826      1728.4442537319           1.5115438090      ...
  2019.9785942029      1707.6396996094           1.5115438090      ...
```

The file contains the calibration parameters used to generate the peaks and the Q list used to generate the peaks. Each peak gets its own line in the file. Each peak is a list of space separated numbers.[2] The first two numbers are the x and y pixel coordinate corresponding of the peak. The next number is the Q value found in the Q list. The next number is the Q value calculated at the peak using the calibration parameters. The next number is the χ value calculated from the pixel coordinate using the calibration parameters. The next number is the intensity value at the peak. The final number is the 2θ value calculated at peak using the calibration parameters.

5.13 Handling Calibration Data

There are inputs on the "`Calibration`" tab for the calibration parameters. "`xc`" is for the x pixel coordinate of the center of the image, "`yc`" is for the y pixel coordinate of the center of the image, "`d`" is for the distance from the sample to the detector, "`E`" (or "λ`:`") is for the energy or wavelength of the beam. The α, β, and R inputs are for the three rotation angles. "`pl`" is for the pixel length and "`ph`" is for the pixel height. The calibration parameters can be directly entered using the inputs or they can be loaded and saved using the "`Load From File`" and "`Save To File`" buttons. The format for a calibration data file is

```
# Calibration Parameters
xc            1725.0000000000 0
yc            1725.0000000000 0
D              125.2960000000 0
E            12735.3957721000 0
alpha            0.0000000000 0
beta             0.0000000000 0
R                0.0000000000 0
pl             100.0000000000
ph             100.0000000000
```

Comment lines beginning with a # and are ignored. Each of the parameters is on its own line. Each line is the parameter's name followed by spaces or tabs and the value. The value can be followed by an optional second number that is whether that parameter should be fixed or not. 1 means fix the parameter and 0 means to let it very. If there is no number, the parameter will not be fixed. *pl* and *ph* can not be followed by their fixed value since they

[2]In version 1 of the program, the numbers were tab separated, but this changed with version 2.

can not be varied. Instead of energy, the wavelength of the incoming beam can be stored in a calibration file. The wavelength line looks like "`wavelength 0.973540`". When the program is in wavelength mode, the program will wavelength instead of energy as one of the calibration parameters The program will load files containing either and do the conversion. See section 5.6 for a discussion of the program working with wavelength instead of energy.

5.14 The "`Select The Center`" option

Version 2 of the Area Diffraction Machine introduces a convenient option in the "`Calibration`" menu called "`Select The Center`". It can be used to select on the diffraction data the center of the image (the "`xc:`" and "`yc:`" value. To do so, pus the "`Select The Center`" option and then click on the diffraction data where the center should be. This will place the desired coordinates into the "`xc:`" and "`yc:`" inputs.

5.15 Handling Q Data

Q data is loaded into the program with the "`Q Data:`" input on the "`Calibration`" tab. The filename can either be entered by hand and then loaded or picked from a file selector by pushing the button with the folder icon on it. Below is an example of the Q data file format:

```
# This is Q Data for Lanthanum Hexaboride
Q    dQ
1.511543809 .05
2.137646823 .05
2.618102966 .05
3.023087619 .05
3.379873753 .05
3.702525225 .05
...
```

Comment lines beginning with a # are ignored. The first line in the file should be "`Q dQ`" or "`Q delta Q`". The rest of the are lines of Q values followed a ΔQ range. All Q values must be larger than 0. None of the Q ranges can overlap. Instead of Q values, the program can input D values If the first line is instead "`D dD`" or "`D delta D`" the rest of the file will be taken to be D and ΔD values. The values will be convert using equation 4.21.

5.16 The "`Standard Q`" Menu

This program stores several common standard Q files as defaults. They can be selected through the "`Standard Q`" option of the "`Calibration`" menu. Selecting one of them will simply put the path of the Q file into the "`Q Data:`" input and hit the load for you. This menu shown on figure 2.3 on page 10.

You can actually add your own Q data files to this menu by going into the "`StandardQ/`" folder inside of the Area Diffraction Machine's folder and moving your Q data file. It will show up in the menu the next time you run the program. It is a little more complicated to do with the Mac distribution because the whole program is bundled up into one "`.app`" file. To do so, you have to right click on the "`.app`" file, which is probably in your "`Applications/`" folder, and select the "`Show Package Contents`" option. In the "`Contents/`" folder, there is a "`Resources/`" folder, which contains the "`StandardQ/`" folder where you can put the Q data files.

5.17 The "`Get From Header?`" Input

Sometimes calibration parameters are stored in the header of a diffraction file. The "`Get From Header`" button will make the program try to find these values and load them into the program.

CHAPTER 6

PIXEL MASKING

When analyzing diffraction data, not all of the pixels in an image should be used in the analysis. In order to make the program ignore certain pixels, this program allows for two types of pixel masking: threshold masking and polygon masking. You can apply either of these from the "`Masking`" tab shown in figure 6.1.

6.1 Threshold Masking

The top half of the "`Masking`" tab is devoted to threshold masking. Threshold masks allows all pixels, either above a certain intensity or below a certain intensity, to be ignored when doing the diffraction analysis. The "`Do Greater Than Mask?`" check box can be used to apply a mask that will cause all pixels greater than a certain value to be ignored. The "`(Pixel's Can't Be) Greater Than Mask`" input can be used to specify the maximum pixel value. The "`Do Less Than Mask`" check box can be used to make the program ignore all pixels below a certain value. The particular value can be specified with the "`Less Than Mask`" input.

When applying a threshold mask, the pixels above or below the threshold will be colored on the diffraction and cake image. The color can be set with the "`Color`" button next to the greater than and less than masks. Figure 6.2 shows diffraction image when all pixels with intensity above 5000 are colored green and all pixels below 30 are colored red.

When caked data is saved, any of the pixels that are larger than the greater than mask are saved as -2. Any of the pixels smaller than the less than mask are saved as -3. This behaviour needs to be accounted for when analyzing caked data outside the program.

When an integrating intensity, any of the too high or too low pixels are simply ignored when calculating average intensity.

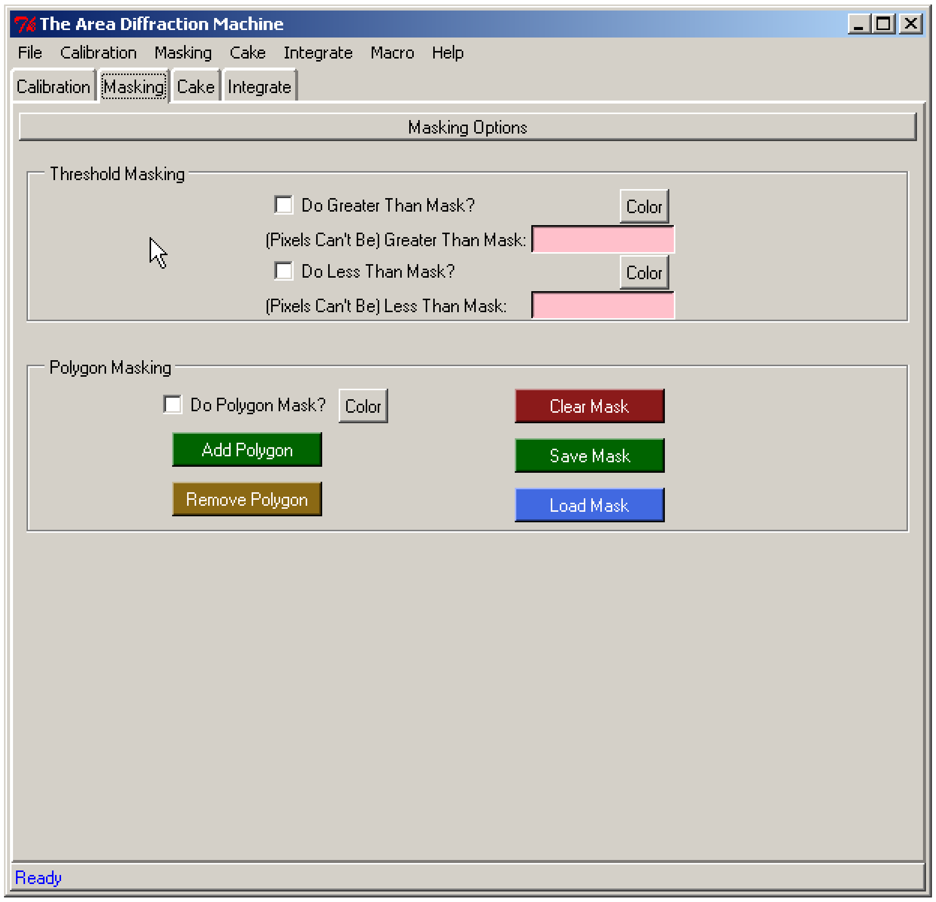

The Area Diffraction Machine
File Calibration Masking Cake Integrate Macro Help
Calibration Masking Cake Integrate
Masking Options
Threshold Masking
Do Greater Than Mask?
Color
(Pixels Can't Be) Greater Than Mask:
Do Less Than Mask?
Color
(Pixels Can't Be) Less Than Mask:
Polygon Masking
Do Polygon Mask?
Color
Clear Mask
Add Polygon
Save Mask
Remove Polygon
Load Mask
Ready

Figure 6.1: The pixel masking tab can be used to add a threshold mask or a polygon masking.

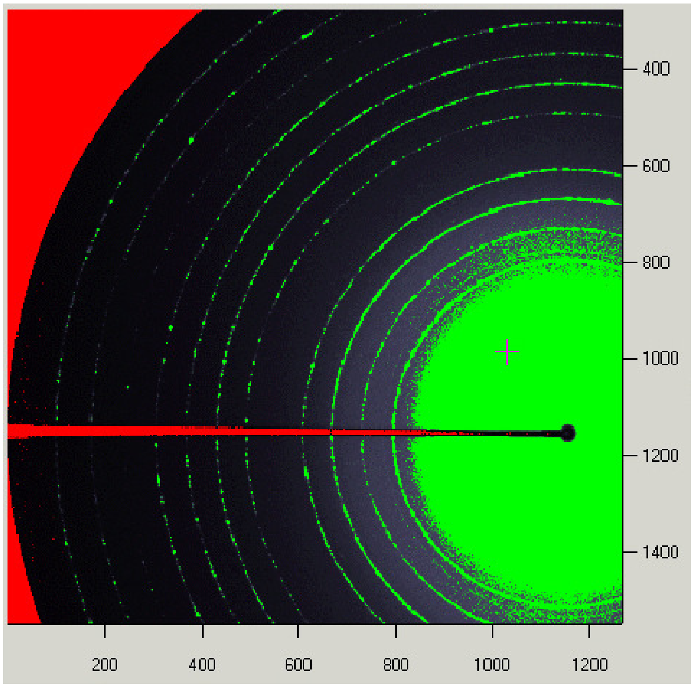


Figure 6.2: A diffraction image with a greater than mask and less than mask. All pixels with intensity greater than 5000 have been colored green and all pixels with intensity less than 30 have been colored red. Applying an intensity mask can be used to see if a detector's pixels have been overloaded. They can also be a used to ensure that overloaded pixels are not used in the data analysis.

6.2 Polygon Masking

Sometimes, large areas of a diffraction image should not be included in the data analysis. For example, often a beam stop blocks part of the detector and the pixels behind the beam stop should be ignored. This can be done with a polygon masks. Polygons can be drawn on the diffraction image and those parts of the image will not be used in subsequent analysis. This program can handle multiple polygons at the same time.

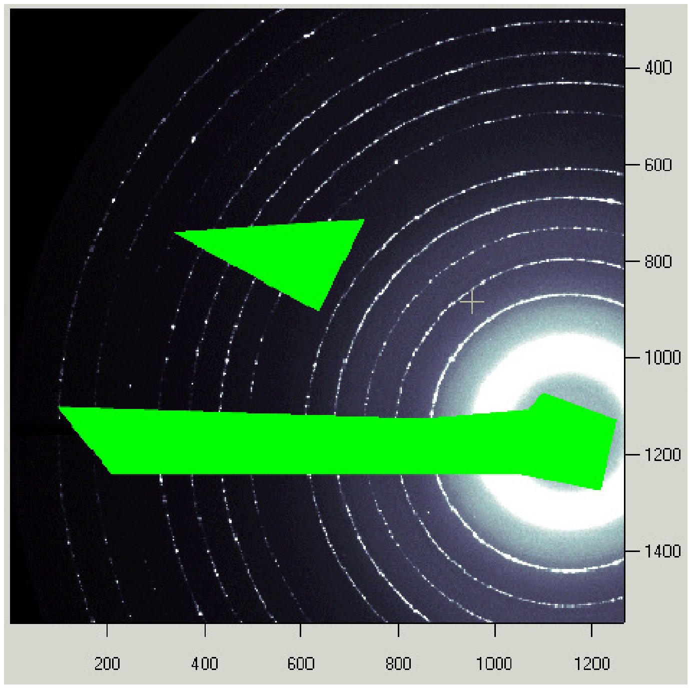


Figure 6.3: Two polygon masks that have been applied to this diffraction image. One of them covers the beam stop.

So long as the "`Do Polygon Mask?`" check box is selected, the polygon masks will be used when performing subsequent analysis. As figure 6.3 shows, the polygons will be colored on the diffraction and cake image. The color of the polygon masks can be set with the "`Color`" button next to the "`Do Polygon Mask?`" check box. When caked data is saved, any pixels inside polygon masks will be given an intensity value of -4. During an intensity integration masked pixels will be ignored.

A polygon mask can be added to the image with the "`Add Polygon`" button on the "`Masking`" tab. This button will stay down when pushed and puts the program in polygon drawing mode. In this mode, the diffraction image can no longer be zoomed or panned. Instead, left clicking on the diffraction image will make the program draw a polygon mask. The first left click adds the first vertex. Each success left click add another vertex. The drawing can be finished and the final vertex added by right clicking. The program return to its normal state and add the polygon into the program. Multiple polygons can be added using the "`Add Polygon`" button. Figure 6.4 shows the program when a polygon is being drawn. A half drawn polygon can be aborted without finishing it by unpushing the "`Add Polygon`" button.

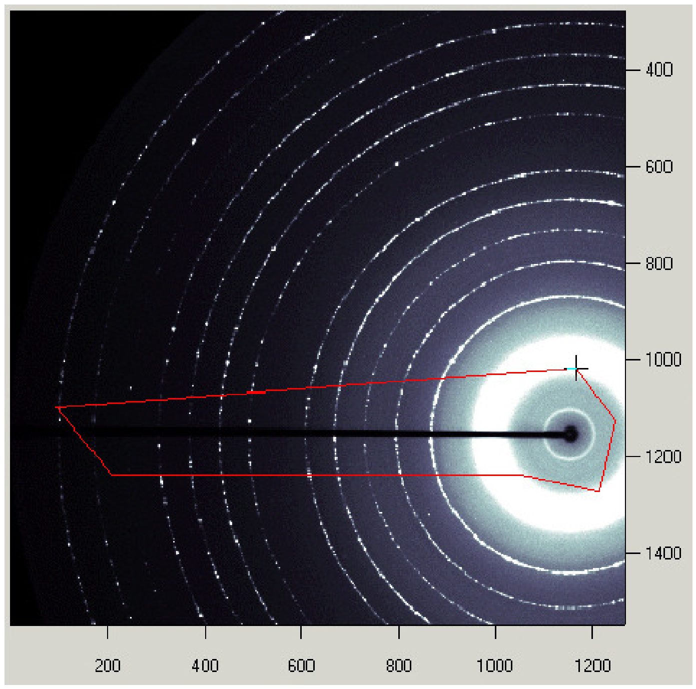


Figure 6.4: A new polygon mask is being added to the program. This particular mask will cover the beam stop.

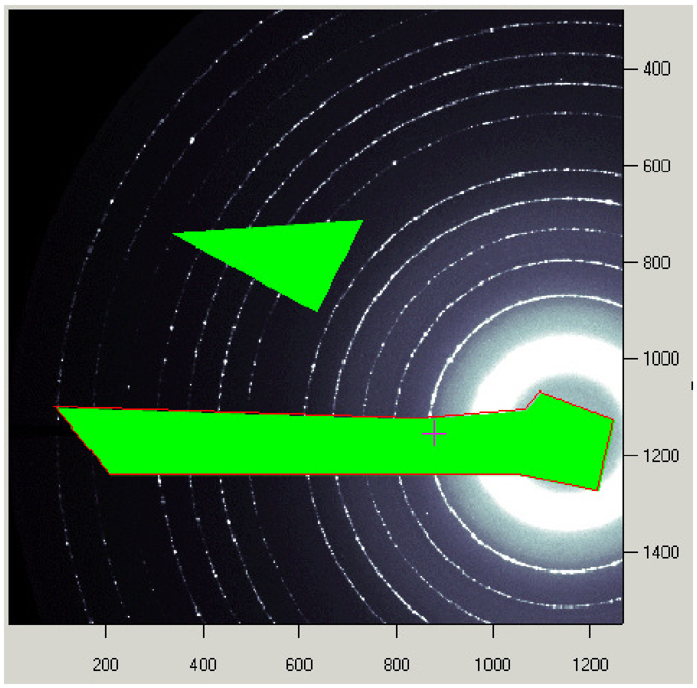


Figure 6.5: A polygon is about to be removed. When mousing over a polygon to remove it, the program will display a red border around it.

The "`Remove Polygon`" button can be used to remove a polygon from the program. Like the "`Add Polygon`" button, this button will stay pushed and change the behavior of the diffraction image. After pushing the "`Remove Polygon`" button, clicking over a particular polygon will remove it. After the polygon is removed, the program will return to its normal state. Figure 6.5 shows what the diffraction window looks like when a polygon is about to be removed. The program can return to its normal state without removing a polygon by unpushing the "`Remove Polygon`" button.

The "`Clear Mask`" button can be used to remove all the polygons at once. The "`Save Mask`" button can be used to save all the polygons to a file. A file of polygons can be added to the program using the "`Load Mask`" button. A polygon file is very simple. The polygons in figure 6.3 were saved as

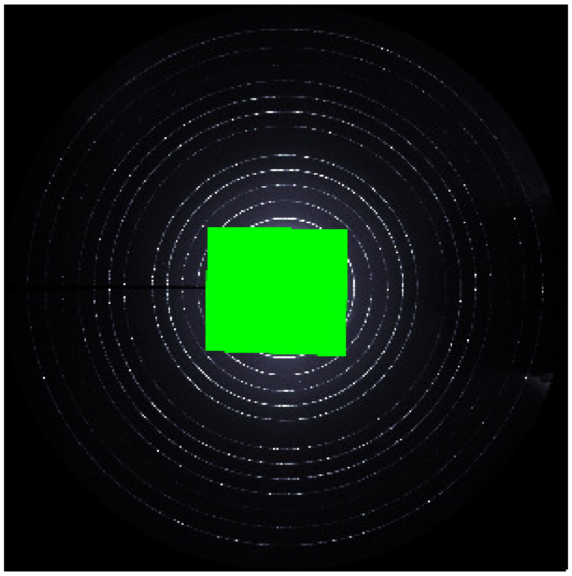
```
# Polygon(s) drawn on Thu Feb 07 00:00:21 2008
93.140587183    1098.06704199
208.013978042   1237.77792276
1052.48863517   1237.77792276
1213.93231962   1271.92947139
1248.08386825   1126.00921814
1095.95424252   1067.02017959
1064.90738013   1104.27641447
847.579343365   1122.9045319

332.201427619   737.923438212
633.355992844   902.471808902
729.601266267   709.981262058
```

Each line is an (x,y) coordinate for one of the nodes of the polygon. The coordinates are separated by spaces. Each polygon is separated by a newline. Comment lines beginning with # are ignored.

6.3 Masking Caked Plots

Any polygon masks or threshold masks will also show up on the caked plot. As figure 6.6 shows, polygons on a diffraction image can look very distorted on a caked plot.


(a) A rectangular polygon mask in the middle of a diffraction image

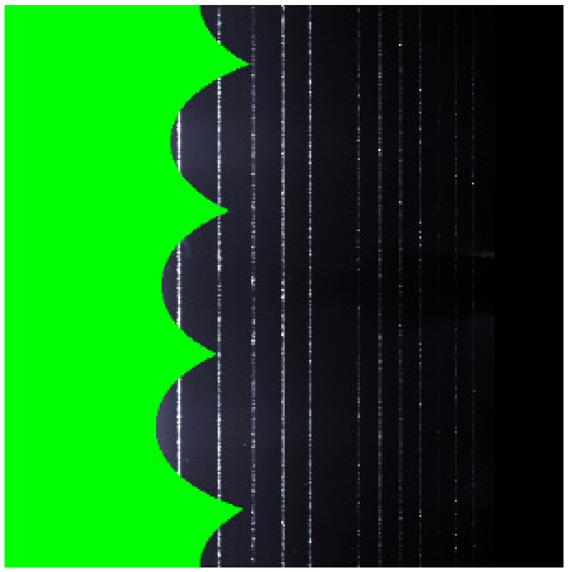

(b) The same rectangular mask on a caked plot

Figure 6.6: A relatively simple shape on a diffraction image can look very different on a caked plot.

CHAPTER 7

CAKING

7.1 The Caking Algorithm

A caked plot is similar to a radial (r vs. θ) plot of the area diffraction data as it would appear if it were captured on an untilted detector. A radial plot will turn circles of constant r into straight lines. Caking is important because diffraction peaks will also become straight lines. A caked plot is actually a plot of Q vs. χ. Although the relationship is not linear, Q increase as r increases and therefore Q is similar to r. χ corresponds to the angle radially around the x-ray beam. So a cake is analogous to a radial plot.

Cakes are calculated as follows. Q and χ space is first binned. Since each bin has some particular Q and χ value[1] the corresponding (x''', y''') pair can be calculate for each bin using equation 4.15 and 4.16. The intensity value at the pixel coordinate (x''', y''') is put into the bin. (x''', y''') is generally between a pixel coordinate so a bilinear interpolation of the intensity is used to get a best estimate.

Pixel masks can be used to ignore parts of the caked data. Whenever the program finds an intensity value that should be masked (either because it is too large, too small, or in a polygon mask), it fills that part of the caked array with a particular negative value. When the caked data is displayed, these negative values are shown a with special color.

Finally, in order to insure there is not an ambiguity between negative intensity values and these special negative values, the program will always set the intensity of all bins whose intensity is less then zero to equal zero. If the diffraction data contains pixels with negative values, this could cause problems. But this is rarely the case. This clipping of negative values was added in version 2 of the program.

The program can perform a polarization correction of the caked data. The polarization

[1]Technically, each bin has a Q and χ range. The middle of the bin is taken to be the particular Q and χ value.

correction is

$$I = Im/PF \tag{7.1}$$
$$PF = P(1-(\sin(2\theta)\sin(\chi-90))^2)+(1-P)(1-(\sin(2\theta)\cos(\chi-90))^2) \tag{7.2}$$

with Im the measured intensity. The 2θ and χ values are for to the particular value that is being corrected. All pixels have their intensity corrected by this formula before they are binned.

7.2 Caking With the Program

Caking is done with the "`Caking`" tab shown in figure 7.1 The program can only cake data after diffraction data and calibration parameters have been loaded into the program. In order to cake, this program needs to know the range in Q and χ space that should be caked. This can be specified with the "`Q Lower?`", "`Q Upper?`" "`Chi Lower?`", and "`Chi Upper?`" inputs. The program also needs to know how many Q and χ bins to create when caking. This can be set with the "`Number of Q?`" and "`Number of Chi?`" inputs. The "`Do Cake`" button can then be used to cake the data.

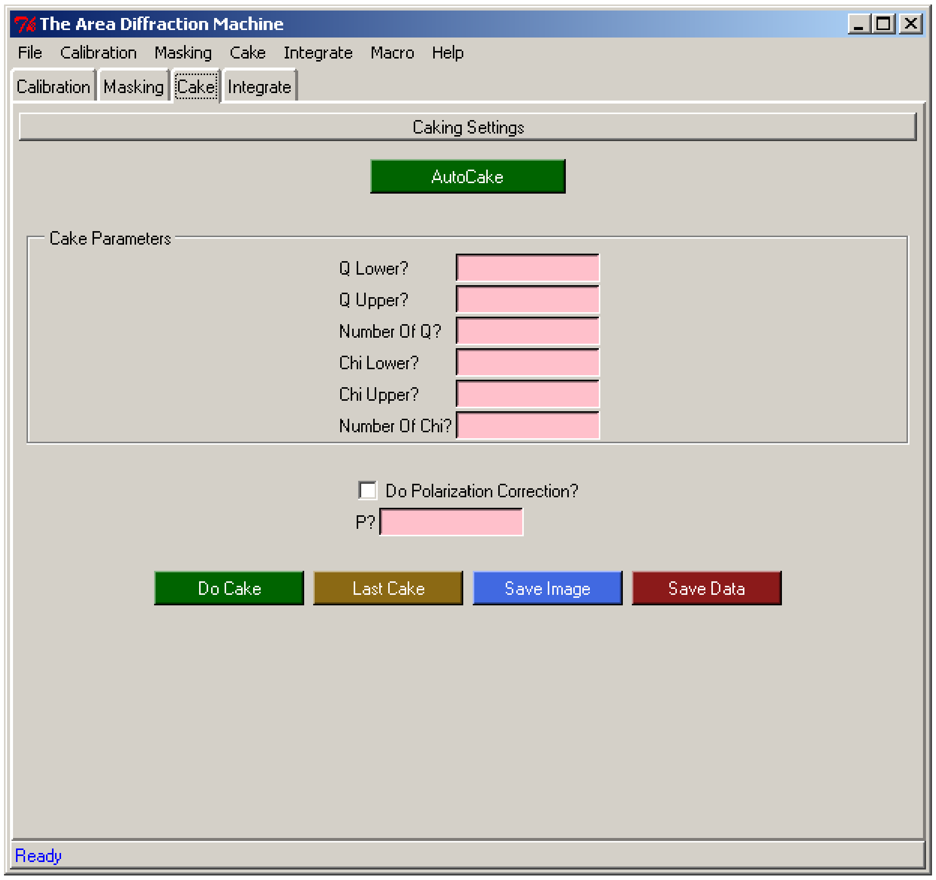


Figure 7.1: Caking is done with the "`Caking`" tab.

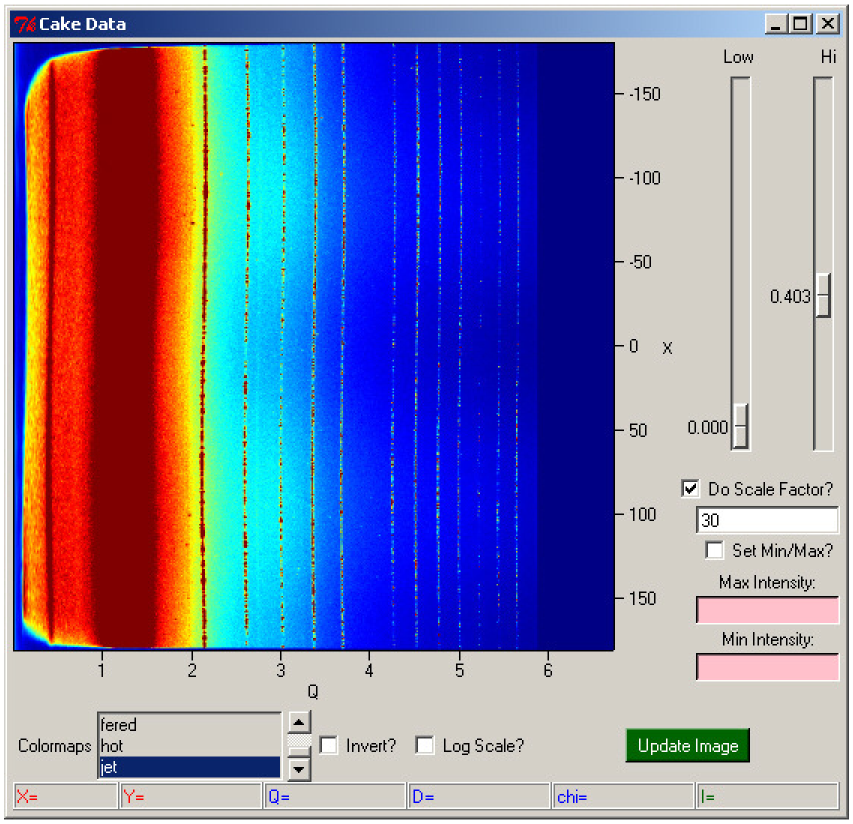


Figure 7.2: The caked data window will open after the data is caked.

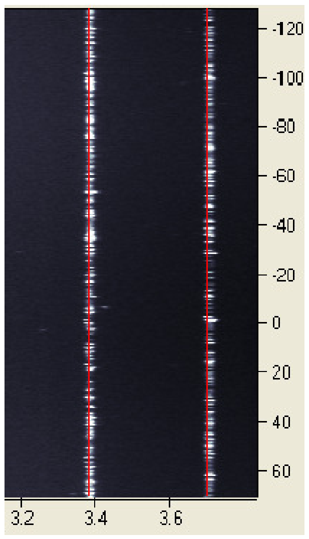
After the cake finishes, the program will open a caked data window that displays the cake interactively. The caked data window acts just like the diffraction data window so everything in Chapter 3 carries over. The only difference is that whenever the caked data is zoomed into, the program will take the selected zoom range and put it into the inputs on the cake tab and then recake the image. The caked can be taken to the previous zoom range with the "`Last Cake`" button

7.3 AutoCake

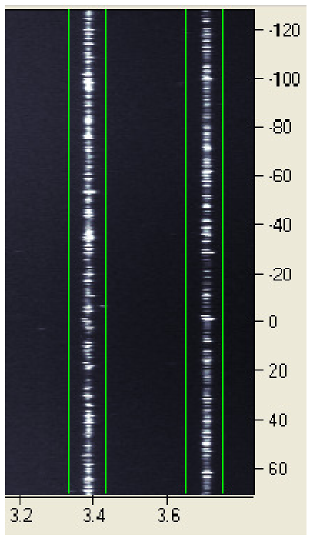
"`AutoCake`" is a button on the "`Cake`" tab that will guess a good range of Q and χ values, put them into the inputs, and then push the "`Do Cake`" button automatically. It can be used to create a cake without much work. "`AutoCake`" will pick a range that puts the entire diffraction image into the cake. It will pick a bin sizes so that each displayed pixel correspond to one bin. This will ensure that the cake looks as sharp as the computer can display it.

7.4 Displaying Q and ΔQ Lines

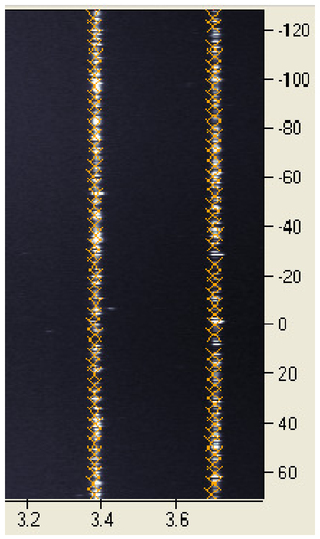
If a Q list has been loaded into the program, constant Q lines or ΔQ lines can be displayed on top of the cake. Constant Q lines on the diffraction image are vertical lines on the caked plot. The program will display constant Q lines or ΔQ lines on the caked plot whenever they should be displayed on the diffraction image. See section 5.8 and section 5.9 for a discussion of displaying constant Q lines on diffraction data. Figure 7.3a shows constant Q lines displayed on a caked plot and figure 7.3b shows constant ΔQ lines displayed on a caked plot.



(a) A cake with constant Q lines drawn on top of it.



(b) A cake with constant ΔQ lines drawn on top of it.



(c) A cake with diffraction peaks drawn on top of it.

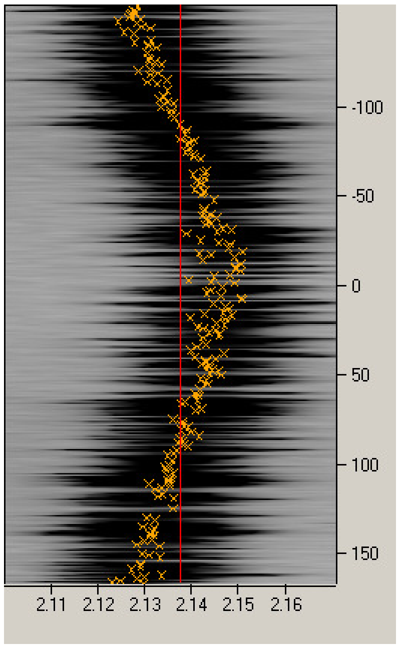
Figure 7.3:

7.5 Displaying Peaks

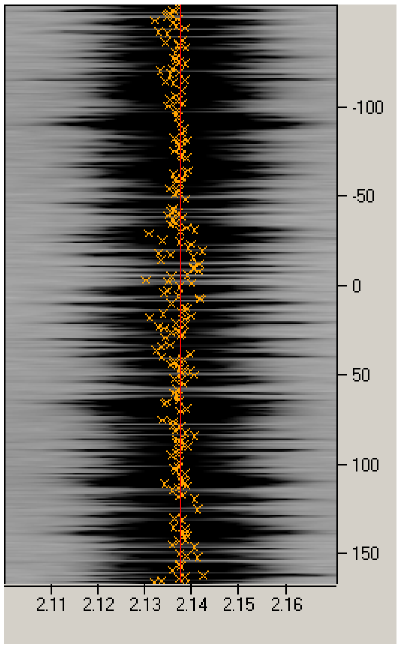
Any peaks that the program finds when calibrating can be displayed on top of the caked data as crosses. Peaks will be displayed on the caked plot whenever they are displayed on the diffraction image. Figure 7.3c shows peaks displayed on a caked plot. Being able to display Q lines and peaks on caked data can be very useful for checking if the calibration was done properly. Figure 7.3 illustrates this principle.

7.6 Polarization Correction

The "`Do Polarization Correction?`" check box can be used to apply a polarization correction to the cake and the polarization value can be set with the "`P?`" input.



(a) A bad calibration



(b) A good calibration

Figure 7.4: Displaying peaks and constant Q lines on top of a cake of a calibration standard can be used to tell if the data was properly calibrated. If the calibration is bad, the diffraction peaks will have a systematic distortion. If the calibration is good, all the peaks will cluster very close to their particular Q value and there will be no systematic distortion.

7.7 Working in 2θ

Caked plots can have 2θ instead of Q as one of the axes. The program can change to 2θ mode by going into the file menu and selecting the "`Work in 2theta`" option. When this is selected, all the names in the program will change from Q to 2θ. The program will have "2θ `Lower`", "2θ `Upper`", and "`Number of` 2θ" as inputs. The allowed range will then be in 2θ and the program will display the cake image with 2θ as an axis. The corresponding "`Work in Q`" option can be used to return the program to caking in Q. This feature was introduced in version 2.0.0.

7.8 Saving Cake Data

Caked data can be saved as a plain text using the "`Save Data`" button. The format for cake files is a long comment string followed by the data in rows of numbers:

```
# Cake of: N:/data/LaB6_14_02_56.mar3450
# Data Caked on Wed Mar 12 21:30:55 2008
# Calibration data used to make the cake:
#   x center:     1725.0000000 pixels
#   y center:     1725.0000000 pixels
#   distance:      125.2960000 mm
#   energy:      12735.3957721 eV
#   alpha:           0.0000000 degrees
#   beta:            0.0000000 degrees
#   rotation:        0.0000000 degrees
#   pixel length:      100.0000000 microns
#   pixel height:      100.0000000 microns
# A Polarization correction was applied
#   P = 0.500000
# A greater than mask was applied
#   Greater than mask = 1000.000000
# A Less Than Mask was applied
#   Less than mask = 10.000000
# Polygon mask(s) were applied
# Polygon(s) used in the analysis:
#   2400.10912343       1073.5706619
#   962.511627907       2282.88014311
#   2850.51520572       2572.86762075
#
#   1573.33631485       1215.47942755
#   1820.13416816       2893.70483005
#   2906.04472272       1573.33631485
# Cake range:
#   Q Lower = 0.000000
#   Q Upper = 6.726544
#   Number of Q = 560.000000
#   Q Step = 0.012012
#   chi Lower = -180.000000
#   chi Upper = 180.000000
#   Number of Chi = 560.000000
#   chi Step = 0.642857
# Note: pixels outside the diffraction image are saved as -1
#   Pixels greater than the greater than mask are saved as -2
#   Pixels less than the less than mask are saved as -3
#   Pixels inside of a polygon masks are saved as -4
# chi increased down. Q increases to the right
```

the header describes what state the program was in when the cake was done. It first lists the name of the diffraction file(s) that were caked. It then lists the calibration parameters, the polarization correction, the threshold masks, and the polygon masks used when caking. It then lists the pixel coordinates of any polygons used. It then lists the range of the cake and the number of bins that were used. The program tries to be smart about the comment string. So if no masks were used, the comment string would instead contains

```
# No greater than mask was applied
# No less than mask was applied
# No polygon masks were applied
```

If the program is working in 2θ, the comment string would instead say

```
#    2theta Lower = 0.000000
#    2theta Upper = 62.814525
#    number of 2theta = 560.000000
#    2theta Step = 0.112169
```

Then comes the data. Each line is of constant χ and contains many numbers separated by spaces. Each column is of constant Q. χ increases down and Q increases to the right. The program sets certain bins in the data to special values. Bins outside the diffraction image are saved as -1. Bins masked because they are too large are saved as -2. Bins masked because they are too small are saved as -3. Bins inside a pixel mask are saved as -4. This is also noted in the comment string.

7.9 Saving Cake Images

Caked data can be saved as one of many popular image formats. The program can save caked images as "`jpg`", "`gif`", "`eps`", "`pdf`", "`bmp`", "`png`", or "`tiff`". A caked image will be saved with whatever threshold masks, polygon masks, Q lines, ΔQ lines, and peaks were displayed over the caked data.

CHAPTER 8

INTENSITY INTEGRATION

8.1 The Integration Algorithm

An intensity integration is a plot of average intensity as a function either Q, 2θ, or χ. The calibration values for the diffraction data must be known before the integration is done. A range and bin size for the integration must be give. For example, a $Q - I$ integration might have a range from 2 to 5 with 100 bins.

The algorithm for performing an intensity integration is as follows: loop over every pixel in the image. Add its intensity to a bin if its Q, 2θ, or χ value falls within the bin's range. Remember that the calibration parameters are used to calculate these from the pixel coordinates. After binning all the pixels, the bins are then averaged.

This program can constrain the integration range. This can be used, for example, to integrate in Q only those pixels with some particular χ range or to integrate in χ for only a particular Q range. This could be used, for example, to perform a χ integration of only one diffraction peak. The program applies the constraint by only binning pixels allowed by the constraint.

The program can perform a polarization correction by correcting all pixel before they are binned by

$$I = Im/PF \tag{8.1}$$

$$PF = P(1 - (\sin(2\theta)\sin(\chi - 90))^2) + (1 - P)(1 - (\sin(2\theta)\cos(\chi - 90))^2) \tag{8.2}$$

with Im the measured intensity. The 2θ and χ values correspond to the particular value that is being corrected.

Finally, if no pixels were put into a particular intensity bin, the bin's intensity is set to -1 as a flag that the bin doesn't contain any data. To avoid ambiguity, the program will set all bins with negative intensities to have an intensity of zero. If the diffraction data contains pixels with negative values, this could cause problems. But this is rarely the case. This clipping of negative values was added in version 2 of the program.

8.2 Integrating with the Program

The program requires one or more diffraction images and calibration parameters to be loaded into the program before an intensity integration is done. Figure 8.1 shows the “`Integrate`” tab where integration is done. There are two sets of inputs on the tab. The inputs on the left are titled “`Q-I Integration`” and are used for performing Q integration. The integration range in Q and the number of Q bins can be specified with the “`Q Lower?`”, “`Q Upper?`”, and “`Number of Q?`” inputs. The “`Integrate`” button on the left does the Q integration.

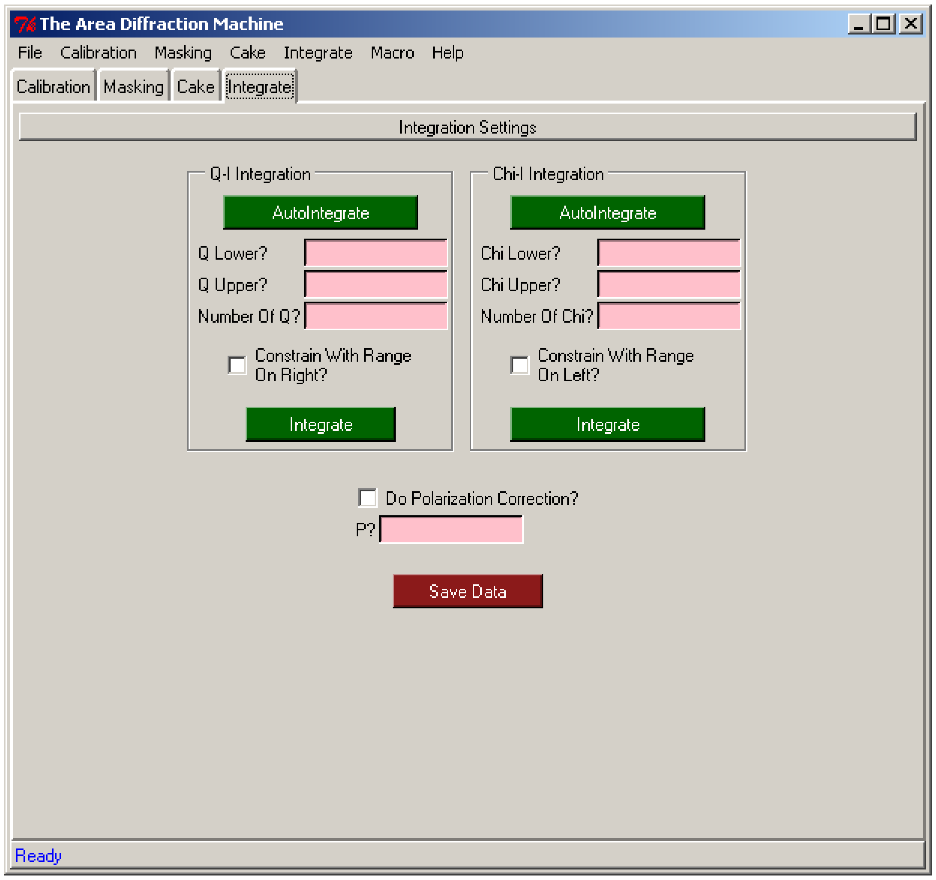

The Area Diffraction Machine
File Calibration Masking Cake Integrate Macro Help
Calibration Masking Cake Integrate
Integration Settings
Q-I Integration
AutoIntegrate
Q Lower?
Q Upper?
Number Of Q?
Constrain With Range On Right?
Integrate
Chi-I Integration
AutoIntegrate
Chi Lower?
Chi Upper?
Number Of Chi?
Constrain With Range On Left?
Integrate
Do Polarization Correction?
P?
Save Data
Ready

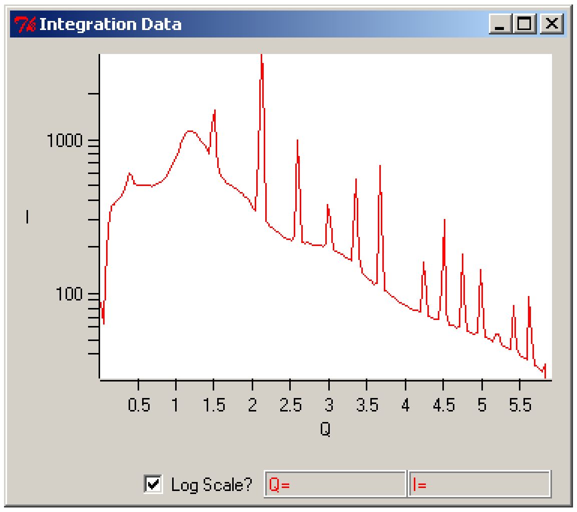
Figure 8.1: The integration tab is used for intensity integration.

The inputs on the right are titled “`Chi-I Integration`” and can be used for performing a χ integration.

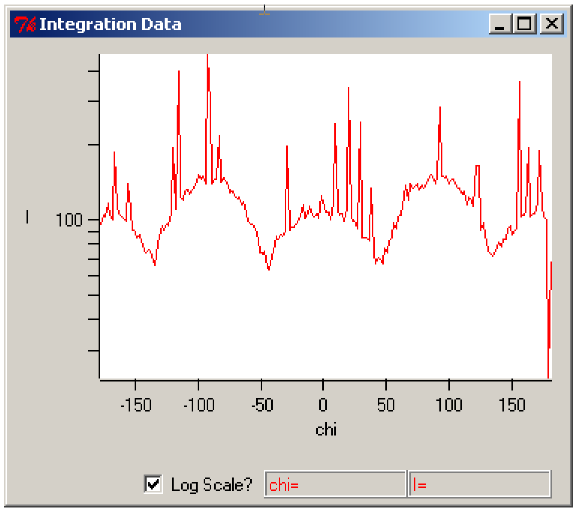
The integration range in χ and the number of χ bins can be specified with the “`Chi Lower?`”, “`Chi Upper?`”, and “`Number of Chi?`” inputs. The “`Integrate`” button on the right does the χ integration.

8.3 The Integration Window

A line plot of the integrated data will be displayed in a new window after the program finishes integrating. Figure 8.2a shows the integration window displaying $Q - I$ integrated



(a) A $Q - I$ integration.



(b) A $\chi - I$ integration.

Figure 8.2: The integration window opens after an intensity integration.

data and figure 8.2b shows the window displaying $\chi - I$ integrated data. You can:

- *Zoom into the data* – left click on the plot and hold down on the mouse. The program will create a resizing rectangle that follows the mouse. When the mouse is released, the program will zoom into the selected range.
- *Zoom out of the data* – right click on the plot.
- *Resize the window* – click on the bottom right corner of the window and drag. The window will resize just like any other window and the plot will become larger or smaller.
- *Read coordinates for a selected point* – when mousing over the plot, the selected Q, χ or 2θ and intensity value will be displayed at the bottom of the window.
- *Log Scaling* – the "`Log Scale?`" check box will toggle whether to display the intensity axis as a log scale.

8.4 Working in 2θ

This program can integrate in 2θ instead of Q. The "`Work in 2theta`" option in the menu bar will make the label on the left to say "`2`θ`-I Integration`". The inputs will change to "2θ `Lower`", "2θ `Upper`", and "`Number of` 2θ". The "`Integrate`" button will then perform an integrate in 2θ. The diffraction window will display average intensity as a function of 2θ. If there are any values in the "`Q Lower`" or "`Q Upper`" inputs, they will be convert from Q to 2θ values when the program switches. The "`Work in Q`" option in the menu bar can be used to change the program back to working with Q and any 2θ values will be converted back.

8.5 AutoIntegrate

The "`AutoIntegrate`" button on the left will guess a nice range of Q (or 2θ) and then do the Q (or 2θ) integration. It will always make the lower Q (or 2θ) value 0 and the upper value large enough to include all the data. It will set the number of Q (or 2θ) to 200. The "`AutoIntegrate`" button on the right will guess a nice range of χ and do the χ integration. It will always set "`Chi Lower`" to -180, "`Chi Upper`" to 180, and "`Number of Chi`" to 200.

8.6 Constraining the Inputs

As was described in section 8.1, a Q or 2θ integration can be done only of values in a particular χ range. A χ integration can only be done only of a particular Q or 2θ range. The integration can be constrained with the "`Constrain With Range On Right?`" and "`Constrain With Range On Left?`" check boxes. When "`Constrain With Range On Right?`" is selected, the Q or 2θ integration will be constrained in χ by the chi range specified by "`Chi Lower`" and "`Chi Upper`". When "`Constrain With Range On Left?`" is selected, the χ integration will be constrained by either the Q range specified by "`Q Lower?`" and "`Q Upper?`" or the 2θ range specified by "`2θ Lower?`" and "`2θ Upper?`".

8.7 Masking

The program allows for masking of certain pixels while integrating. Masking is done whenever the "`Do Greater Than Mask?`", "`Do Less Than Mask?`", or "`Do Polygon Mask?`" check boxes are selected. Whenever the program finds an intensity value that should should be masked (either because it is too large, too small, or in a polygon mask), the program will ignore the pixel and not bin it. Refer to Chapter 6 for a discussion of masking.

8.8 Saving Integrated Data

The intensity integrated data can be saved to a file using the "`Save Data`" button. A typical integration file looks like:

```
# Q vs I Intensity Integration
# Intensity integration of: C:/data/LaB6_14_02_56.mar3450
# Data Integrated on Fri Mar 21 17:59:16 2008
# Calibration data used:
#   x center:    1725.0000000 pixels
#   y center:    1725.0000000 pixels
#    distance:      125.2960000 mm
#    energy:      12735.3957721 eV
#    alpha:           0.0000000 degrees
#    beta:            0.0000000 degrees
#    rotation:        0.0000000 degrees
#    pixel length:      100.0000000 microns
#    pixel height:      100.0000000 microns
# A polarization correction was applied
#    P = 1.000000
# A greater than mask was applied
#    Greater than mask = 10000.000000 (All pixels above 10000.000000 were
# A Less Than Mask was applied.
#    Less than mask = 50.000000 (All pixels below 50.000000 were ignored)
# Polygon mask(s) were applied
# Polygon(s) used in the analysis:
#    647.844364937       1369.72808587
#    1449.93738819       3226.88193202
#    2535.84794275       1449.93738819
#
#    1258.66905188       641.674418605
#    1215.47942755       999.531305903
#    1505.46690519       1116.76028623
#    1653.54561717       777.413237925
# Integration performed with a chi constraint
#    chi constraint lower: 90.000000
#    chi constraint upper: 270.000000
# Integration Range:
#    Q Lower = 0.000000
#    Q Upper = 6.726544
#    Number of Q = 200.000000
#    Q Step = 0.033633
# Q     Avg Intensity
0.016901        0.000000
0.050703        0.000000
0.084504        0.000000
0.118306        0.000000
0.152108        0.000000
0.185910        0.000000
...
```

The header describes the state of the program when the integration was performed. The fist line describes what type of integration was performed. If a $\chi - I$ integration was performed, the header file would say "`# Chi vs I Intensity Integration`". The header then contains the name(s) of the diffraction files that were integrated, the calibration parameters that were used when integrating, any polarization correction that was used, and any threshold

It describes any constrains on the integration and the integration range. Following the header is the line "`# Q Avg Intensity`" (or "`# Chi Avg Intensity`" or "`# 2theta Avg Intensity`") followed by the data. Each line contains two numbers. The first number is the middle Q (or χ or 2θ value) in the bin and the second number is the average intensity.

CHAPTER 9

MACROS

This program is almost fully automatable with macros. Macros can be used to quickly analyze data. This program is capable of recording and running macros and it is easy to write and modify macro files.

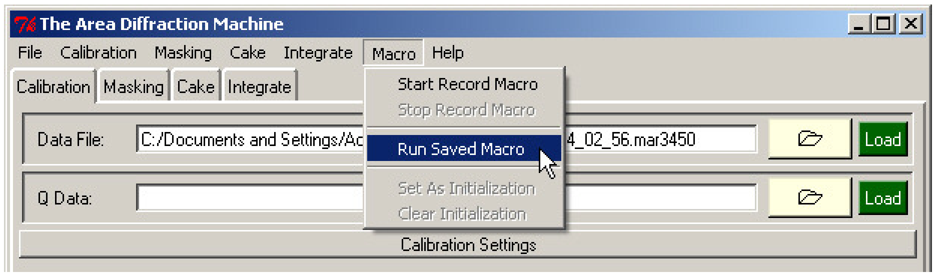


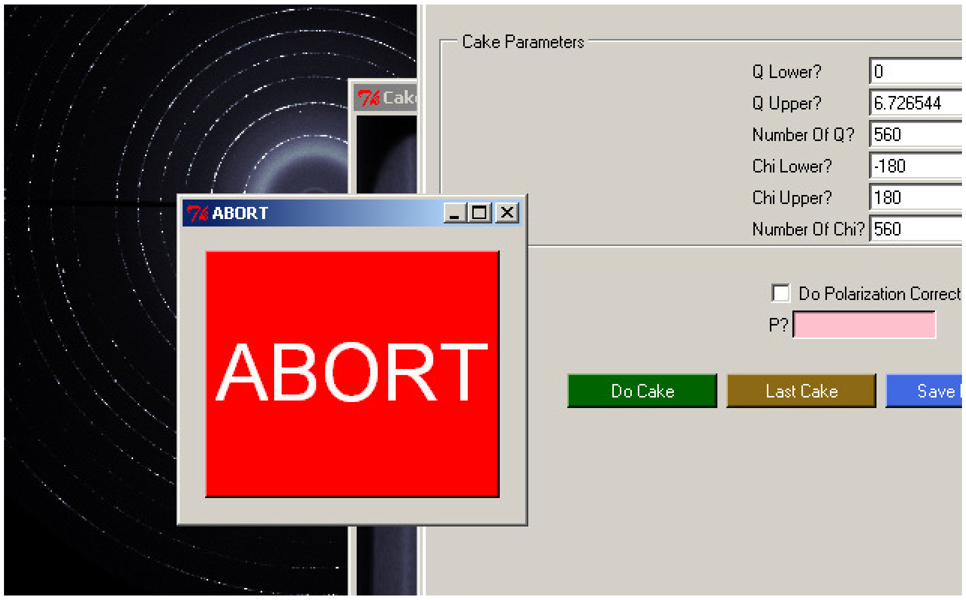
Figure 9.1: The “`Macro`” menu bar is used to record and run macros.

9.1 Record and Run Macros

The easiest way to create a macro is to record it by selecting the “`Start Record Macro`” option in the “`Macro`” menu bar shown in figure 9.1. After all of the desired steps are finished, the “`Stop Record Macro`” option will save the macro to a file. The “`Run Saved Macro`” option in the “`Macro`” menu will run all the commands in a saved macro.

9.2 The “`Abort`” Button

Version 2 of the program introduces an abort button which can be used to abort out of running a macro in the middle of running it. Figure 9.1 shows a screen shot of this button. This can be useful if part way through running a long macro, you realize that the macro isn't doing the right thing. It is sometimes a little difficult to push the button and have it actually stop the program, especially when the program gets frozen up running long calculations like an intensity integration, but if you are diligent and push it a couple of times, it will eventually stop the macro.


Cake Parameters

Q Lower? 0
Q Upper? 6.726544
Number Of Q? 560
Chi Lower? -180
Chi Upper? 180
Number Of Chi? 560

☐ Do Polarization Correct
P?

Figure 9.2: The "`Abort`" button can be used to abort out of a running macro.

9.3 The "`Set As Initialization`" Option

Version 2 of the Area Diffraction Machine introduces the feature "`Set As Initialization`". It can be used to run a particular Macro file ever time the Area Diffraction Machine is opened. This is a useful way to make the program always open with certain options selected and certain inputs set to particular values (like setting the "`Stddev?`" input to 7). The best way to do this is to open the program, start recording a macro set the program in the desired state, and instead of pushing the "`Stop Record Macro`" button push the "`Set As Initialization`" option. This will save the recorded macro to the file "`InitializationMacro.dat`" in the folder "`Preferences/`" next to the executable.[1]. If desired, the "`InitializationMacro.dat`" file can be modified by hand to change the macro file.

To get the program to no longer run a macro whenever the program opens, either push the "`Clear Initialization`" option next to the "`Set As Initialization`" option. Alternately, push the "`Start Record Macro`" option and then immediately push the "`Set As Initialization`" option. Alternately, simply delete the "`InitializationMacro.dat`" file.

[1]It is a little bit harder to find this file on the Macintosh. To find it, right click on the application and select the "`Show Package Contents`" option. Inside of the "`Contents/`" folder is the folder "`Resources/`" which contains the "`Preferences/`" folder where you can find the file

9.4 The Macro File Format

A macro file is a list of commands, each on their own line, that tells the program what to do. The syntax for macros is straightforward. Macro commands are the text corresponding to the part of the GUI that does the command. For example, the macro command "`Get From Header`" will get the calibration data from the header of the image.[2] Things get more interesting when the GUI item is more complicated then a button. The "`Draw Q Data?`" check box needs to know if it is selected or deselect. The macro command is

```
Draw Q Data?
    Select
# Or, Deslect to not display them
```

Entering numbers is similar:

```
beta:
    5.23
```

So are filenames:

```
Save Caked Image
    C:/cake_output.jpg
```

9.5 Looping Over Diffraction Data

Diffraction data is loaded with the command

```
Data File:
    C:/foo.mar3450
```

Diffraction files in a list will be looped over and the same analysis can be performed on all of them. For example,

```
Data File:
    C:/foo.mar3450 C:/bar.mar3450
# ...
```

will run the subsequent macro lines on the file "`C:/foo.mar3450`" and then on the file "`C:/bar.mar3450`". The loop will end when one of three things happens: a subsequent line in the macro file is "`END LOOP`", more diffraction data is loaded using the "`Data File:`" command, or the macro file ends.

Directories can also be given. The program will (non-recursively) look for all diffraction files in the directory and include them in the list. If the folder "`C:/data`" contained the file "`foo.mar3450`" and "`bar.mar3450`", these files could be looped over with the command

[2]Because of the ambiguity of some GUI items, some macro command have different names. For example, the "`Number Of Chi`" input shows up multiple places in the program so the macro commands are instead "`Fit Number of Chi?`", "`Integrate Number Of Chi?`", and "`Cake Number Of Chi?`". When in doubt, section 9.11 has a list of all macro commands.

```
Data File:
    C:/data/
# ...
```

Multiple folders and multiple files can be in the list. They must be on the same line and separated by spaces.

9.6 PATHNAME and FILENAME

When a diffraction file is loaded by the macro, any subsequent lines in the file will have the string "`PATHNAME`" replaced with the path of the current diffraction file and "`FILENAME`" replaced with the filename of the current diffraction file. For the file "`C:/data/first.mar3450`", "`PATHNAME`" would be replaced by "`C:/data`" and "`PATHNAME`" would be replaced by "`first`". A "`.mar3450`" file could be reconstructed as "`FILENAME/PATHNAME.mar3450`"

While looping over multiple files, these commands can be used to save things in useful places with useful names. It would be easy, for example, to save intensity integrated data with the macro:

```
Save Integration Data
    FILENAME/PATHNAME\_int.dat
```

This would save the intensity data for "`C:/data/foo.mar3450`" as "`C:/data/foo_int.dat`" and the intensity data for "`C:/data/bar.mar3450`" as "`C:/data/bar_int.dat`", etc.

9.7 Loading Multiple Images

Multiple diffraction files can be loaded and their intensities added. The macro command to do this is "`Multiple Data Files`" followed by a list of filenames enclosed with [and] brackets. "`C:/data/foo.mar3450`" and "`C:/data/bar.mar3450`" could be loaded at the same time with the macro

```
Multiple Data Files:
    [C:/data/foo.mar3450 C:/data/bar.mar3450]
# ...
```

All of the files that are added together must be in the same folder so that that the "`PATHNAME`" syntax remains meaningful. These sets of files can be looped over by putting several of these bracketed lists on a line. For example:

```
Multiple Data Files:
    [C:/a.mar3450 C:/b.mar3450] [C:/c.mar3450 C:/d.mar3450]
# ...
```

will separately loop over "`a.mar3450`" and "`b.mar3450`" added together and then "`c.mar3450`" and "`d.mar3450`" added together. Alternately, all of the files that need to be analyzed can be grouped into subfolders. If each subfolder contains only files that should be added together, the "`Multiple Data Files`" command followed by the base folder will loop over each of the subfolders and add all of the files in each subfolder together. For example, the folder "`C:/data`" contained the sub folder "`A`" containing the files "`a.mar3450`" and "`b.mar3450`" and suppose "`C:/data`" also contained the sub folder "`B`" containing the files "`c.mar3450`" and "`d.mar3450`". The macro command

```
Multiple Data Files:
    C:/data
# ...
```

would first load "`a.mar3450`" and "`b.mar3450`" and run the through loop. It would then load "`c.mar3450`" and "`d.mar3450`" and run the through loop. Multiple Lists of files and multiple folders containing subfolders can be put after the "`Multiple Data Files`" line. They must all be on the same line and separated by spaces. Since the all the files added together must come from the same folder, "`PATHNAME`" is unambiguous. "`FILENAME`" is replaced by the string "`MULTIPLE_FILES`" to avoid ambiguity.

9.8 FOLDERPATH and FOLDERNAME

To facilitate writing macros that load and add together diffraction files in a loop, macros allow the "`FOLDERNAME`" and "`FOLDERPATH`" syntax. "`FOLDERNAME`" will be replaced by the name of the folder containing the current diffraction file(s). "`FOLDERPATH`" will be replaced by the path leading to the folder containing the file. A "`mar3450`" file can be reconstructed as "`FOLDERPATH/FOLDERNAME/FILENAME.mar3450`". "`FOLDERNAME`" is useful because if multiple files have been loaded at the same time, output can be saved with the enclosing folder's name. If multiple subfolders are looped over, the output of the program for each set of files can be given a meaningful name using the subfolder's name.

9.9 Setting Colors in a Macro

There are several places in the program where a color can be selected. Colors can be set from inside a macro but the names of the colors is a little confusing. The easiest way to figure out the macro command to set a color is to record a macro and then copy and paste. Technically, the program will accept any color string that the tk GUI framework accepts. tk will accept quite a few colors by their English name (such as "`red`").[3] Tk also accepts colors by their RGB value. A color is written as # followed by the color's RGB value in hexadecimal. Each red, green, and blue value ranges from 0 to 255 in decimal and 00 to ff in hexadecimal. For example, #ff0000 is red.

[3] `http://wiki.tcl.tk/16166` contains a list of all the allowed color names.

9.10 Little Tidbits

- Macro commands are not case sensitive. The command "`GeT fRoM hEaDeR`" is just as valid as "`Get From Header`".
- White spaces at the beginning and end of a line are ignored. In the preceding examples, the spaces separating macro commands from input values was only there to increase readability.
- New lines in a macro file are ignored.
- Comment lines beginning with # are ignored.
- The program will automatically move between tabs as it executes different commands on different tabs.
- Usually, macro commands will not be recorded when the macro item is selected through the menu bar. The only exceptions are items that can only be accessed that way.
- The program will create folders if necessary in order to save files out into folders that are specified but that do not yet exist.

9.11 List of Macro Commands

Table 9.1: Macro Commands

Command	Followed By	Effect
	Program State Commands	
`Work In eV`	None	Changes the state of the program to work with the beam's energy.
`Work in Lambda`	None	Changes the state of the program so that the beam's wavelength.
`Work in 2theta`	None	Changes the state of the program so that caking and intensity integration are done in 2θ instead of Q.
`Work in Q`	None	Changes the state of the program so that caking and intensity integration are done in Q instead of 2θ.
	Calibration Commands	
`Data File:`	Files & Directories	Opens a diffraction file and possibly loops over doing this.

Continued on next page...

Table 9.1 – continued from previous page

Command	Followed By	Effect
`Multiple Data Files"`	Files & Directories	Loads several several diffraction files and adding them together and possibly loops over doing this.
`Q Data:`	Filename	Load in a Q data file.
`Standard Q`	Q File	Loads in a standard Q files. This command is followed by the name in the standard Q menu of one of the standards.
`Get From Header:`	None	Gets the calibration data from the image header.
`Load From File:`	Filename	Loads a calibration data file.
`Previous Values`	None	Loads the previous calibration parameters.
`Save To File`	Filename	Saves the calibration data to a file.
`xc:`	Number	Sets the x center.
`xc Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the x center.
`yc:`	Number	Set the y center.
`yc Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the y center.
`d:`	Number	Set the detector distance.
`d Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the distance.
`E:`	Number	Sets the energy. If the program is in λ mode, it will switch to eV mode.
`E Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the energy. If the program is in λ mode, it will switch to eV mode.
`lambda:`	Number	Sets the wavelength. If the program is in eV mode, it will switch to λ mode.
`lambda Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the wavelength. If the program is in eV mode, it will switch to λ mode.
`alpha:`	Number	Sets the α angle.
`alpha Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the α angle.
`beta:`	Number	Sets the β angle.
`beta Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the β angle.
`R:`	Number	Sets the rotation angle.
`R Fixed:`	`Select` or `Deselect`	Sets whether or not to fix the rotation angle.
`pl`	Number	Sets the pixel length.
`ph`	Number	Sets the pixel height.

Continued on next page...

Table 9.1 – continued from previous page

Command	Followed By	Effect
`Draw Q Lines?`	`Select` or `Deselect`	Sets whether or not to draw constant Q lines.
`Draw Q Lines Color?`	color	Sets the color of the constant Q lines.
`Draw dQ Lines?`	`Select` or `Deselect`	Sets wheter or not to draw constant ΔQ lines.
`Draw dQ Lines Color?`	color	Sets the color of the ΔQ lines.
`Draw Peaks?`	`Select` or `Deselect`	Sets wheter or not to draw the fit peaks.
`Draw Peaks Color?`	color	Sets the color of the peaks.
`Update`	None	Refreshes the diffraction image.
`Save Calibration`	Filename	Saves the calibration parameters to a file.
`Do Fit`	None	Fits the calibration parameters.
`Save Last Fit`	Filename	Introduces in version 2 of the program. Will save information about the previous calibration to a file.
`Make/Save Peak List`	Filename	Creates a peak list and saves it to a file.
`Use Old Peak List (if possible)?`	`Select` or `Deselect`	Sets whether or not to use previous peak lists when fitting.
`Fit Number of Chi?`	Number	Sets the number of χ slices around the diffraction image used when calibrating.
`Stddev`	Number	The threshold for allowing peaks.
Diffraction Display Commands		
`Diffraction Data Colormaps`	A colormap name	Sets the colormap used to display the diffraction data.
`Diffraction Data Invert?`	`Select` or `Deselect`	Inverts the colormap used to display the diffraction data.
`Diffraction Data Log Scale?`	`Select` or `Deselect`	Sets whether or not to use a log of the colormap.
`Diffraction Data Low?`	Number from 0 to 1	Sets the percentage of maximum intensity that is mapped to the lowest part of the colormap.
`Diffraction Data Hi?`	Number from 0 to 1	Sets the percentage of maximum intensity that is mapped to the highest part of the colormap.
`Diffraction Data Do Scale Factor?`	`Select` or `Deselect`	Whether or not to apply a scale factor to the intensity range.

Continued on next page...

Table 9.1 – continued from previous page

Command	Followed By	Effect
`Diffraction Data Scale Factor?`	Real number larger then 0	The scale factor to apply to the intensity range.
`Diffraction Data Set Min/Max?`	`Select` or `Deselect`	Whether or not to set the minimum and maximum intensity explicitly for the intensity range.
`Diffraction Data Min Intensity`	Real number	The minimum intensity for the intensity range.
`Diffraction Data Max Intensity`	Real number	The maximum intensity for the intenisty scale
`Save Diffraction Image`	Filename	Saves the diffraction image to a file.
Masking Commands		
`Do Less Than Mask?`	`Select` or `Deselect`	Sets whether or not to apply a less than mask to the diffraction data.
`(Pixels Can't Be) Less Than Mask:`	Number	Sets the less than mask.
`Less Than Mask Color?`	color	Sets the color of the less than mask.
`Do Greater Than Mask?`	`Select` or `Deselect`	Sets whether or not to apply a greater than mask to the diffraction data.
`(Pixels Can't Be) Greater Than Mask:`	Number	Sets the greater than mask.
`Greater Than Mask Color?`	color	Sets the color of the greater than mask.
`Do Polygon Mask?`	`Select` or `Deselect`	Sets whether or not to use polygon masks.
`Polygon Mask Color?`	color	Sets the color of the polygon masks.
`Save Mask`	Filename	Saves all polygon masks to a file.
`Load Mask`	Filename	Loads all polygon masks from a file.
`Clear Mask`	None	Removes ll polygon masks.
Cake Commands		
`AutoCake`	None	Picks a nice Q and χ range and cakes.
`Cake Q Lower?`	Number	Sets the lower Q value used when caking. If the program is in 2θ mode, it will switch to Q mode.
`Cake Q Upper?`	Number	Sets the upper Q value used when caking. If the program is in 2θ mode, it will switch to Q mode.

Continued on next page...

Table 9.1 – continued from previous page

Command	Followed By	Effect
`Cake Number Of Q?`	Number	Sets the number of Q bins used when caking. If the program is in 2θ mode, it will switch to Q mode.
`Cake 2theta Lower?`	Number	Sets the lower 2θ value used when caking. If the program is in Q mode, it will switch to 2θ mode.
`Cake 2theta Upper?`	Number	Sets the upper 2θ value used when caking. If the program is in Q mode, it will switch to 2θ mode.
`Cake Number Of 2theta?`	Number	Sets the number of 2θ bins used when caking. If the program is in Q mode, it will switch to 2θ mode.
`Cake Chi Lower?`	Number	Sets the lower χ value used when caking.
`Cake Chi Upper?`	Number	Sets the upper χ value used when caking.
`Cake Number Of Chi?`	Number	Sets the number of χ values used when caking.
`Do Cake`	None	Performs the cake and displays that caked data.
`Last Cake`	None	Goes back to the previous cake values.
`Save Caked Image`	Filename	Saves the cake as a popular image format. The extension of the file tells the program what format to save the image as.
`Save Caked Data`	Filename	Saves the cake to a file.
`Cake Do Polarization Correction?`	`Select` or `Deselect`	Sets whether or not to apply a polarization correction when caking.
`Cake P?`	Number from 0 to 1	Sets the value of the polarization used when caking.
Cake Display Commands		
`Cake Data Colormaps:`	Colormap	Sets the colormap used to display the caked data.
`Cake Data Invert?`	`Select` or `Deselect`	Sets whether or not to invert the colormap used to display the caked data.
`Cake Data Log Scale?`	`Select` or `Deselect`	Sets whether or not to use a log scale of the colormap.

Continued on next page...

Table 9.1 – continued from previous page

Command	Followed By	Effect
`Cake Data Low?`	Number from 0 to 1	Sets the percentage of maximum intensity that is mapped to the lowest part of the colormap.
`Cake Data Hi?`	Number from 0 to 1	Sets the percentage of maximum intensity that is mapped to the highest part of the colormap.
`Cake Data Do Scale Factor?`	`Select` or `Deselect`	Whether or not to apply a scale factor to the intensity range.
`Cake Data Scale Factor?`	Real number larger then 0	The scale factor to apply to the intensity range.
`Cake Data Set Min/Max?`	`Select` or `Deselect`	Whether or not to set the minimum and maximum intensity explicitly for the intensity range.
`Cake Data Min Intensity`	Real number	The minimum intensity for the intensity range.
`Cake Data Max Intensity`	Real number	The maximum intensity for the intenisty scale
Intensity Integration Commands		
`Integrate Q Lower?`	Number	Sets the lower Q value used when integrating. If the program is in 2θ mode, it will switch to Q mode.
`Integrate Q Upper?`	Number	Sets the upper Q value used when integrating. If the program is in 2θ mode, it will switch to Q mode.
`Integrate Number Of Q?`	Number	Sets the number of Q bins used when integrating. If the program is in 2θ mode, it will switch to Q mode.
`Integrate 2theta Lower?`	Number	Sets the lower 2θ value used when integrating. If the program is in Q mode, it will switch to 2θ mode.
`Integrate 2theta Upper?`	Number	Sets the upper 2θ value used when integrating. If the program is in Q mode, it will switch to 2θ mode.
`Integrate Number Of 2theta?`	Number	Sets the number of 2θ bins used when integrating. If the program is in Q mode, it will switch to 2θ mode.
`Integrate Chi Lower?`	Number	Sets the lower χ value used when integrating.
`Integrate Chi Upper?`	Number	Sets the upper χ value used when integrating.

Continued on next page...

Table 9.1 – continued from previous page

Command	Followed By	Effect
`Integrate Number Of Chi?`	Number	Sets the number of χ bins used when integrating.
`Integrate Q-I`	None	Performs a Q integration. If the program is in 2θ mode, it will switch to Q mode.
`AutoIntegrate Q-I`	None	Picks a good range of Q values and does a Q integration. If the program is in Q mode, it will switch to 2θ mode.
`Integrate 2theta-I`	None	Performs a 2θ integration. If the program is in Q mode, it will switch to 2θ mode.
`AutoIntegrate 2theta-I`	None	Picks a good range of 2θ values and does a 2θ integration. If the program is in Q mode, it will switch to 2θ mode.
`Integrate chi-I`	None	Performs a χ integration of the diffraction data.
`AutoIntegrate chi-I`	None	Picks a good range of χ values and does a χ integration.
`Save Integration Data`	Filename	Saves the integrated data to a file.
`Constrain With Range On Right?`	`Select` or `Deselect`	Sets whether or not to constrain the Q or 2θ integration with the χ range.
`Constrain With Range On Left?`	`Select` or `Deselect`	Sets whether or not to constraint the χ integration with the Q or 2θ range.
`Integrate Do Polarization Correction?`	`Select` or `Deselect`	Sets whether or not to use a polarization correction when integrating.
`Integrate P?`	Number from 0 to 1	Sets the polarization use when integrating.
`Integration Data Log Scale?`	`Select` or `Deselect`	Sets whether or not to display integration data with a log scale.

9.12 What Macros Can't Do

- There is no way with a macro to zoom into the diffraction data, the cake data, or the intensity integrated data
- polygon masks can not be drawn or individually removed from the program. Polygons

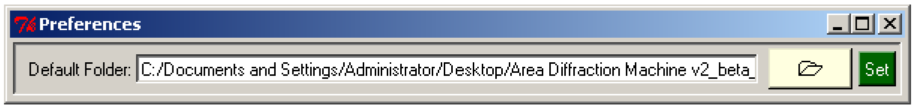
can be loaded from a file with the "`Load Mask`" command and they can all be removed with the "`Clear Mask`" command.

- You can not "`Select The Center`" of the diffraction image from inside of a macro. If you know the coordinates, you can explicitly set them using the "`xc:`" and "`yc:`" commands.
- Diffraction data can only be loaded if it has a standard file extension.

CHAPTER 10

THE PREFERENCES PAGE

In the "File" menu is a "Preferences" option which will open a "Preferences" page.



Default Folder: C:/Documents and Settings/Administrator/Desktop/Area Diffraction Machine v2_beta_

Figure 10.1: The "Preferences" page.

10.1 Setting the Default Folder

Currently, the "Preferences" page contains only one input "Default Folder" that can be used to set a default folder to look in when selecting files. Either type in the folder name and push the "Set" button or simply select the folder from the folder selector by pushing the button with the folder icon on it. The default folder will be persistent. The next time the program is opened, the same default folder will be in the input and the program will look in the same default folder. It does so by saving the folder name in a file named "DefaultDirectory.dat" in the folder "Preferences/". To make the program not use a default folder, simply clear out the input and push the "Set" button. You can also clear out the default folder by deleting the "DefaultDirectory.dat" file in the "Preferences/" folder.

CHAPTER 11

SOFTWARE LICENSING

This program is released under the GNU General Public License (GPL) version 2. The license can e found at http://www.gnu.org/licenses/old-licenses/gpl-2.0.html. For the most part, you are free to use and distribute this software. You are free to make any modifications to the code under the condition that any modifications are clearly stated and that the modifications are released under the GPL version 2.

This software manual is also licensed under the GPL. This is a bit unconventional. I decided to do so after reading several discussions online. Following Nathanael Nerode's article *Why You Shouldn't Use the GNU FDL*, I include the clarification "for the purpose of applying the GPL to this document, I consider 'source code' to refer to the texinfo source and 'object code' to refer to the generated info, tex, dvi, [pdf] and postscript files."[6]

This program uses the software package levmar for performing Levenberg-Marquardt nonlinear least squares minimization. It is released under the GPL. That package can be found at http://www.ics.forth.gr/~lourakis/levmar/.[5]

This program uses the function get_pck() from the CCP4 package DiffractionImage to uncompress Mar data. It written by Dr. Claudio Klein.[3] This prgoram also uses the file marccd_header.h from the DiffractionImage package. It released under the GPL and can be found at http://www.ccp4.ac.uk/ccp4bin/viewcvs/ccp4/lib/DiffractionImage/.[1]

This program uses the EdfFile library (EdfFile.py) for reading and writing files of the ESRF Data Format. It is is part of the PyMCA library and is licensed under the GNU GPL version 2.[7]

This program also uses W. Randolph Frankin's pnpoly() function for performing a point inclusion in polygon test. This code can be found at http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html We are in compliance with his software license which is reproduced below[2]:

BIBLIOGRAPHY

[1] G. Winter F. Remacle. Diffraction Image: a new CCP4 library.

[2] W. Randolph Franklin. PNPOLY - Point Inclusion in Polygon Test. [web page] http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html, Oct. 2005. [Accessed on 06 Feb. 2008.].

[3] Dr. Claudio Klein. pck.c. [web page] http://www.ccp4.ac.uk/ccp4bin/viewcvs/ccp4/lib/DiffractionImage/, Oct. 1995. [Accessed on 06 Feb. 2008.].

[4] Abhik Kumar. Analysis Strategy of Powder Diffraction Data with 2-D Detector. Tech. Rep., Office of Science, SULI Program, Stanford Linear Accelerator Center, Menlo Park, CA, December 2005.

[5] M.I.A. Lourakis. levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. [web page] http://www.ics.forth.gr/~lourakis/levmar/, Jul. 2004. [Accessed on 06 Feb. 2008.].

[6] Nathanael Nerode. Why You Shouldn't Use the GNU FDL. [web page] http://home.twcny.rr.com/nerode/neroden/fdl.html, Sept. 2003. [Accessed on 23 Feb. 2008.].

[7] M. Cotte Ph. Walter J. Susini V.A. Sol, E. Papillon. A multiplatform code for the analysis of energy-dispersive x-ray fluorescence spectra. *Spectrochim. Acta Part B*, 62:63–68, 2007.

INDEX