ANALYSIS OF X-RAY SCATTERING DATA ON AMORPHOUS AND CRYSTALLINE TRANSPARENT CONDUCTORS

Miko Stulajter

Spring 2018

Thesis submitted in completion of Honors Senior Capstone requirements
for the DePaul University Honors Program

Dr. Gabriela González Avilés, Physics

Dr. Anuj Sarma, Physics

Contents

ABSTRACT

The purpose of this research was to develop a methodology to analyze x-ray pair distribution function data for amorphous and crystalline complex oxides. The samples consisted of indium-based oxide transparent semiconductor thin films ranging from fully amorphous to fully crystalline. X-ray scattering data were collected at Argonne National Laboratory. We evaluated the films' lateral and depth uniformity by analyzing different sections of the data that were measured at different penetration depths. The data were analyzed with existing software packages and new Matlab code. The analysis enables the extraction of information about the short-range ordering of the atoms in the materials.

Chapter 1
Introduction

1.1 Background

Transparent conducting oxides (TCOs) were discovered in 1907, but their practical applications were not realized till the middle of the twentieth century [1]. Technologies developed that utilized crystalline TCOs as research into their properties and uses increased over the decades [2]. Since their discovery, TCOs have found their way into everyday life from our smartphone touchscreens to flat-panel televisions to solar panels. TCOs are unique in that they are semiconductors that have high electrical conductivity and are transparent in the visible region. Recently, manufacturers have started experimenting with using amorphous TCOs in their devices to create flexible products like bendable televisions and foldable smartphones. Amorphous TCOs have advantages over crystalline TCOs in that they can be manufactured at lower temperatures and can be deposited on plastic substrates that are flexible. Lower manufacturing temperatures can cut down on the cost of manufacturing, and the ability to deposit on different substrates allows for the creation of more durable products. The advantages that amorphous TCOs bring comes without significant loss to their electrical or optical properties, and in some cases, the electrical properties can be better than traditional crystalline TCOs [3].

Amorphous samples do not have a long-range ordering of atoms, therefore, typical x-ray diffraction techniques cannot provide structural information about these TCOs [2]. Extended x-ray absorption fine structure (EXAFS) is an x-ray technique used to study the short-range structure of amorphous and crystalline TCOs, and it is produced by backscattering of photoelectrons by an atom's nearest neighbors [3].

There have been numerous EXAFS studies probing crystalline and amorphous TCOs. These studies have experimentally measured the coordination numbers and bond lengths of indium oxide (IO) and zinc indium tin oxide (ZITO). The coordination number tells us at a given distance the number of atoms surrounding a particular atom. The first pairing of indium and oxygen atoms is referred to as the first shell. This closest shell is found around 2.18Å [2]. The second and third shells in IO correspond to pairings of indium to indium. The second, third, and fourth shells in ZITO correspond to pairings of indium to indium or indium to zinc or indium to tin. The studies that have been done on amorphous IO and ZITO typically concentrate on the first shell, because the EXAFS signal significantly decreases for higher-order shells like the second and third shells [2].

In this study, a 70 keV x-ray beam was used to obtain data to produce a pair-distribution function (PDF). The experimental setup allows for the measuring of the second and third shells. A radial-distribution function (RDF) is used to analyze the structure by determining the bond length, coordination number, and full width at half maximum (FWHM) for the first, second, and third shells.

1.2 RDF and PDF Theory

PDFs are measured to obtain structural information about materials. PDFs help us extract bond lengths and coordination numbers between atoms. Fourier analysis of total-scattering data is known as PDF analysis [4]. The scattering amplitude $\Psi(Q)$ is given by Eq. 1.1, where $\langle b \rangle$ is the compositional scattering amplitude average given by Eq. 1.2, and b_j defines the scattering amplitude for the atoms [4]. The position of the j^{th} atom is R_j, and N is the number of atoms [4].

$$\Psi(Q) = \frac{1}{\langle b \rangle} \sum_j b_j \tag{1.1}$$

$$\langle b \rangle = \frac{1}{N} \sum_j b_j e^{iQ*R_j} \tag{1.2}$$

The diffraction vector, Q, is the wavevector difference between the incoming beam, k_{init}, and the scattered beam, k_{final}, given as $Q = k_{init} - k_{final}$ [4]. Unfortunately, it is not possible to measure $\Psi(Q)$ directly, but we can measure the intensity of the diffracted beam, which is related to the square of the magnitude of $\Psi(Q)$ or $|\Psi(Q)|^2$ [4].

The intensity $I(Q)$ can be found from Eq. 1.3 [4].

$$I(Q) = \frac{1}{N} \sum_{j,l} b_j b_l e^{iQ(R_j - R_l)} \langle b \rangle^2 - \left\langle b^2 \right\rangle \tag{1.3}$$

Using Eq. 1.3, we get the total-scattering function $S(Q)$:

$$S(Q) = \frac{I(Q)}{\langle b \rangle^2} \tag{1.4}$$

The Fourier transform of the structure-function is the PDF [4]. From $S(Q)$ we can get the reduced-

structure function $F(Q)$, Eq. 1.5, which can be used to obtain the Reduced PDF (RPDF) or $G(r)$ given by Eq. 1.6.

$$F(Q) = Q[S(Q) - 1] \tag{1.5}$$

$G(r)$ is a function in real space where structural information can be extracted. $G(r)$ can also be written as Eq.1.7 where ρ_0 is the average number density, and $g(r)$ is the atomic pair distribution function [4].

$$G(r) = \int_{Q_{min}}^{Q_{max}} F(Q)\, sin(Qr)\, dQ \tag{1.6}$$

$$G(r) = 4\pi r \rho_0 \left[g(r) - 1\right] \tag{1.7}$$

The RDF, $R(r)$ can be obtained from the RPDF using Eq. 1.8 which can be combined with Eq. 1.7 to get Eq. 1.9.

$$R(r) = G(r)\, r + 4\pi r^2 \rho_0 \tag{1.8}$$

$$R(r) = 4\pi r^2 \rho_0 g(r) \tag{1.9}$$

The peak locations of the RDF give information about the bond length between atoms, and the coordination number can be extracted from the integration of the area under a peak. The coordination number N_c or the number of nearest neighboring atoms is proportional to the area underneath a peak through Eq. 1.10, where r_1 and r_2 define the peak corresponding to the coordination shell [4].

$$N_c = \int_{r_1}^{r_2} R(r)\, dr \tag{1.10}$$

1.3 Motivation

The main goal of researching TCOs is to find better materials with improved properties, such as electron mobility and mechanical flexibility. Higher electron mobility can be found in fully crystalline samples, but also in some amorphous samples, which also have improved mechanical properties. Amorphous TCOs are not rigid, so they can be deposited on flexible substrates like plastic which can bend more without cracking.

Through the study of different deposition temperatures and dopant for TCOs, the effects on the atomic structure can be seen, such as changes in bond length, coordination number, or FWHM. This study can help in the optimization of TCOs' macroscopic properties and guide their industrial synthesis. Studying multiple types of TCOs allows for the exploration of different compounds for diverse applications. Finding the best way to analyze PDF data of TCO samples will allow more samples to be analyzed faster. A clear procedure allows for the automation of certain parts of the analysis through coding, and allows for the standardization of the data since all data will be analyzed and normalized in the same manner. This standardization makes it is easier to compare different TCOs and see trends in the data. Therefore, a range of samples from amorphous to crystalline and doped and undoped were analyzed in order to find the best technique for analysis.

Chapter 2
Methodology

2.1 Data Collection

The samples used in this study were grown using pulsed-laser deposition (PLD) at Northwestern University by the Chang group [5, 6]. For amorphous samples to be made, the substrate was cooled in order to prevent the atoms from arranging into a lattice [2]. In this experiment, the substrate was fused silica [6]. The electrical properties of these samples were also measured at Northwestern by the Chang group. The physical and electrical properties are affected by the deposition temperature along with the thickness [6]. High energy x-ray scattering data of these samples were collected at the Advanced Photon Source at Argonne National Laboratory. Figure 2.1 shows the instrumental setup used for wide-angle x-ray scattering (WAXS) as well as PDF measurements [7].

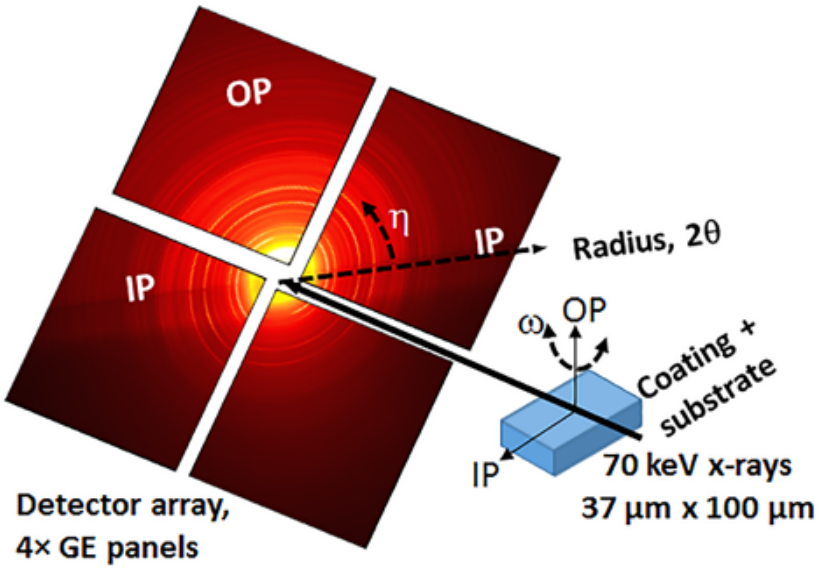


Figure 2.1: Schematic of the experimental setup showing the sample and the arrangement of the four-panel array of 2D detectors. IP stands for in-plane and OP stands for out-of-plane. The figure is taken from González, G. B., et al; "Relationship between electrical properties and crystallization of indium oxide thin films using ex-situ grazing-incidence wide-angle x-ray scattering"; *Journal of Applied Physics* 121.20 (2017): 205306 [7].

2.2 GSAS II

GSAS II [8] is an open-source Python project used for analysis in crystallographic studies. It can integrate a 2-dimensional image and create a 1-dimensional representation. We do this conversion in order to be able to get structural information about the material from the image. Before GSAS II can be used to integrate the 2-dimensional images collected from the samples, the positions and tilts of the detectors must be calibrated. Calibration must be done on the detector so that every pixel maps to a 2θ and azimuthal angle [8]. For the data used in this project, the detector was calibrated using lanthanum hexaboride (LaB6) 660a and 674b cerium oxide powder standards that were obtained from the National Institute for Standards and Technology (NIST) [2]. The specific sample-to-detector distance is also obtained from the calibration with the powder standards. These parameters are then used to integrate the images.

Once calibration is done, multiple images for each sample were summed and integrated using GSAS II. The limits for integration were the inner and outer 2θ angles and the azimuthal angle. Figure 2.2 shows these limits for a measurement that consists of multiple detectors. The 2θ angle corresponds to the radially outward direction on the detector while the azimuthal angle is the angle in the ϕ direction or the angle that sweeps across the detectors. Once the integrations are done, they are exported so that they can be used in PDFgetX3 to get a PDF [9]. The exported file is a text file that contains a header and two columns of data. The first column is the scattering angle, 2θ, in degrees, and the second column contains the corresponding intensities at those angles normalized per unit solid angle [8].

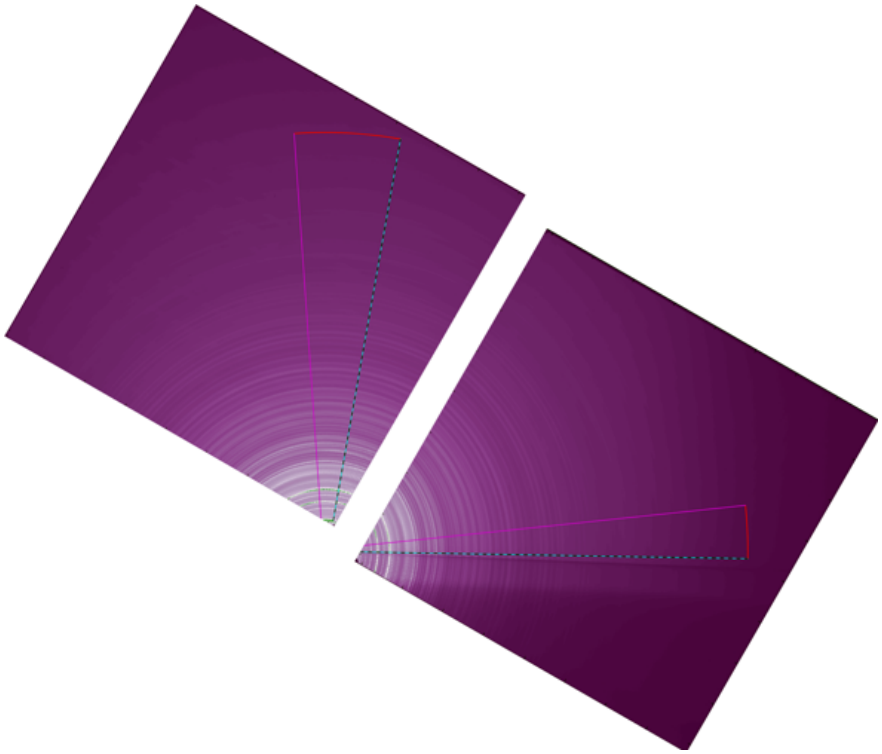


Figure 2.2: This is the output image of two detectors. Each detector has limits of integration. The red lines are the upper 2θ limits while the purple lines and green dotted lines are the azimuthal angle limits. The figure is taken from Thomas Bsaibes; "Radial-Distribution Function of a Transparent Conducting Oxide"; DePaul University, 2017 [2].

2.3 PDFgetX3

PDFgetX3 calculates the reduced-structure function, Eq. 1.5. The reduced-structure function is easier to calculate than the standard structure function given by Eq. 2.11 where $I_c(Q)$ is the coherent scattering and $f(Q)$ is the atomic scattering factor. The angled brackets indicate an average [9].

$$S(Q) = \frac{I_c(Q) - \langle f(Q)^2 \rangle + \langle f(Q) \rangle^2}{\langle f(Q) \rangle^2} \tag{2.11}$$

The reduced-structure function is Fourier transformed into real space so that a PDF is created. The Fourier transform is done from a Q_{min} close to 0Å^{-1} to a Q_{max} close to 20Å^{-1}. A wide range of Q values is used in the Fourier transform so that a high-resolution PDF can be obtained [2]. The output of PDFgetX3 is a text file that contains a header and two columns of data. The first column is a distance in Å and the second column contains the corresponding PDF values in

$Å^{-2}$.

PDFgetX3 does several corrections to the experimental data. It corrects for air scattering and the contribution due to the sample holder with the $bgscale$ parameter. The program also smooths the data with the $rpoly$ parameter, which is the r-limit for the maximum frequency in the reduced-structure function correction polynomial [9].

All sample data were normalized so that the peaks reached the same height in intensity at ~ 2.2Å, corresponding to the first shell. The parameters were adjusted so that the height of the first peak was around $3Å^{-2}$. This was done in order to be able to compare values obtained across different samples. The r_{poly} value used was 1.49Å unless the peak was not able to attain $3Å^{-2}$ in which case the value was lowered. Figure 2.3 shows a session of PDFgetX3 and the resulting graphs with multiple samples overlaid on top of each other to show how the peaks match up at ~ 2.2Å.

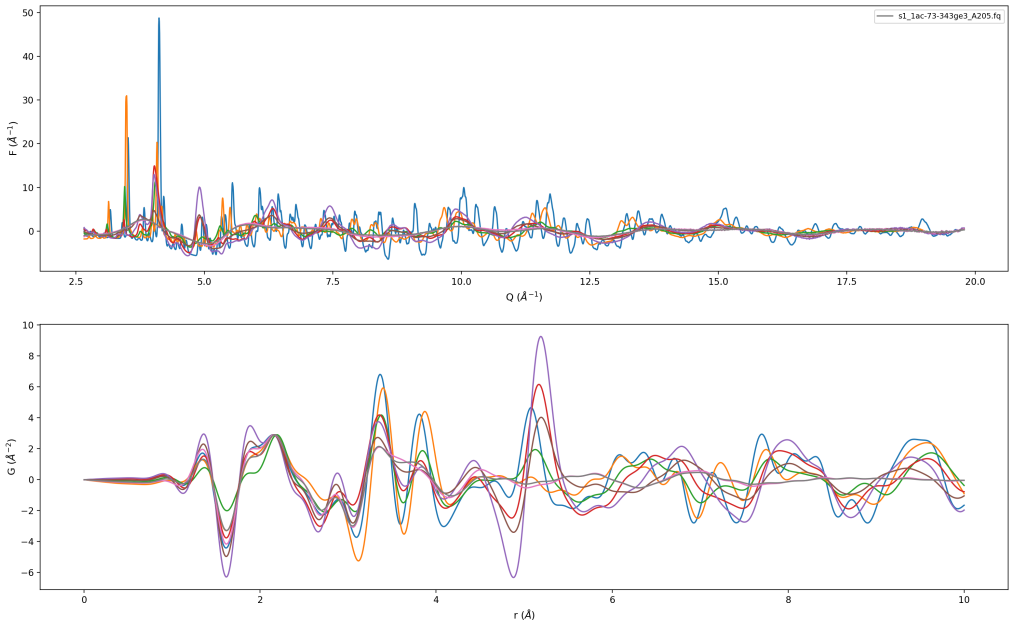


Figure 2.3: This is a session from PDFgetX3. The top graph is the reduced-structure function while the bottom graph is the PDF. The different colored curves correspond to different samples plotted on top of each other. The peaks were overlapped at ~ 2.2Å in order to normalize them to each other. Peaks correspond to coordination shells, and information about the local atomic structure can be obtained from the peak position, integrated area, and width of each peak.

2.4 MATLAB

MATLAB scripts were used to convert the PDFs created by PDFgetX3 to RDFs. This was done by utilizing Eq. 1.7 and Eq. 1.9. Eq. 1.7 can be rewritten as Eq. 2.12 so that we can solve for $g(r)$ since we know $G(r)$ from PDFgetX3.

$$g(r) = \frac{G(r)}{4\pi r \rho_0} + 1 \tag{2.12}$$

After conversion to an RDF, the data is fitted in the Curve Fitting App, which gives parameters a, b, and c. These parameters correspond to the gaussian function given by Eq. 2.13.

$$f(x) = ae^{-(\frac{x-b}{c})^2} \tag{2.13}$$

Here a is the amplitude, b is the centroid, and c is related to the peak width [10]. MATLAB outputs a 95% confidence interval in its Curve Fitting App for each fitting allowing for easier error propagation.

The parameter b corresponds to the bond length. Eq. 2.14 was used to obtain the FWHM from the c parameter.

$$\text{FWHM} = 2\sqrt{2\ln(2)}\frac{c}{\sqrt{(2)}} \tag{2.14}$$

The error also has to be propagated for the FWHM, which results in Eq. 2.15.

$$\text{FWHM}_{error} = \frac{d\text{FWHM}}{dc}c_{error} = 2\sqrt{2\ln(2)}\frac{c_{error}}{\sqrt{(2)}} \tag{2.15}$$

The variable c_{error} is the minimum or maximum c value obtained from the Curve Fitting App. This equation is identical to Eq. 2.14 except that we have a different variable.

The integrated area was calculated using Eq. 2.16 which is obtained through integrating Eq. 2.13. In the equation, y is the integrated area.

$$y = \int_{-\infty}^{\infty} ae^{-(\frac{x-b}{c})^2} dx \tag{2.16}$$

Once the integrated area was calculated, the error was propagated. Using the error that MATLAB provides in its gaussian fits, the error for each integrated area was calculated using the following equations:

$$\Delta a = a - a_{error} \tag{2.17}$$

$$\Delta b = b - b_{error} \tag{2.18}$$

$$\Delta c = c - c_{error} \tag{2.19}$$

$$y = \int_{-\infty}^{\infty} \sqrt{\left[\frac{2ae^{\left[-\left(\frac{x-b}{c}\right)^2\right]}(x-b)\,\Delta b}{c^2}\right]^2 + \left[\frac{2ae^{\left[-\left(\frac{x-b}{c}\right)^2\right]}(x-b)^2\,\Delta c}{c^3}\right]^2 + \left[e^{\left[-\left(\frac{x-b}{c}\right)^2\right]}\Delta a\right]^2}\, dx \tag{2.20}$$

Here a_{error}, b_{error}, and c_{error} are either the upper or lower errors given by the gaussian fit in MATLAB. The difference between the calculated value and the error are given by Δa, Δb, and Δc.

When two gaussians were used, the error was calculated using:

$$y = \sqrt{\left(\left|y_1 - y_{1,error}\right|\right)^2 + \left(\left|y_2 - y_{2,error}\right|\right)^2} \tag{2.21}$$

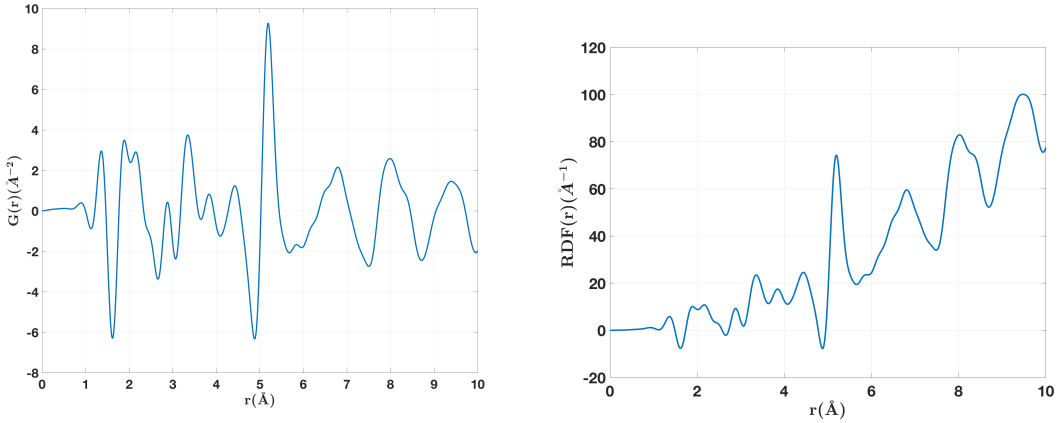
where a subscript of one indicates gaussian one and a subscript of two indicates gaussian two.

These equations were used in order to develop a code that can automatically calculate these values. The code is explained in the next chapter in greater detail.

Chapter 3
Implementation of the Code

3.1 MATLAB

The first code called Gtog(), which can be found in Appendix 7.1.1, converts the PDF to an RPDF using Eq. 2.12. The next code used is called gtoR(), which can be found in Appendix 7.1.2, which converts the RPDF to an RDF using Eq. 1.9. Figure 3.1 shows the original PDF, Figure 3.1a, and the PDF converted to an RDF, Figure 3.1b.



(a) Original PDF before conversion to an RDF. **(b)** RDF after conversion from a PDF.

Figure 3.1: Conversion from PDF to RDF.

A code called savevariables(), which can be found in Appendix 7.1.3, is used next in order to save variables r and $G(r)$ produced by gtoR(). These variables are saved to the workspaces so that they can be used in the Curve Fitting App in MATLAB. In the Curve Fitting App, we use gaussians to fit the first, second, and third shells and sometimes the fourth shell, if needed. Three gaussians are used for the first shell. Two peaks are used for the second and third shells, which are fitted together because the peaks are formed through the convolution of two gaussians. If there appears to be a fourth peak that is of interest, then shells two, three, and four are fitted together using three gaussians. In the Curve Fitting App, data points can be included or excluded. This allows for the fitting of individual peaks of interest. When fitting a shell or a set of shells, data points that do not contribute to the shell or shells are excluded. Figure 3.2 shows a fitting of the second and third shells.

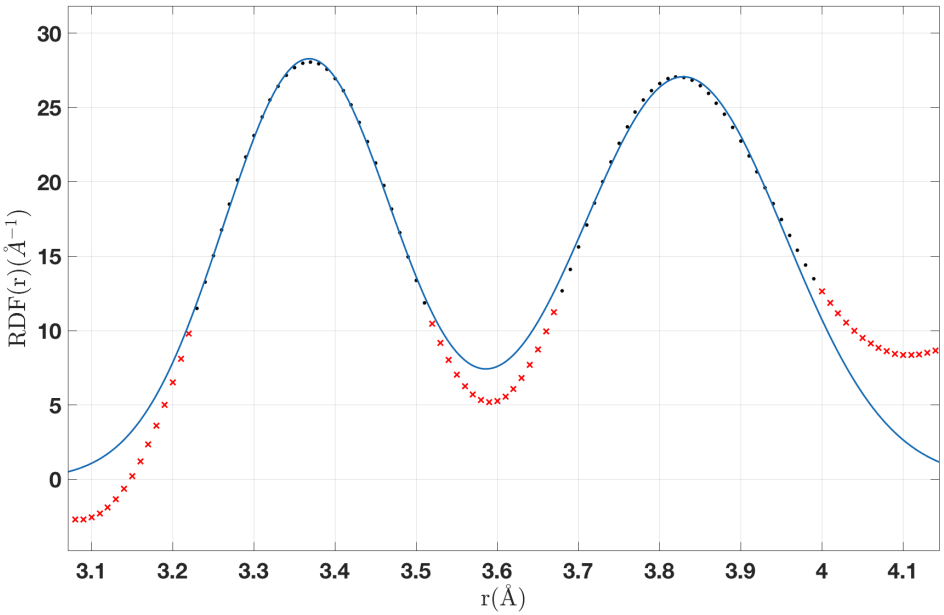


Figure 3.2: This is a session from the Curve Fitting App used to fit shells two and three. The red stars are excluded data points while the black dots are the included data points. The blue line is the convolution of two gaussians which are used to fit these shells.

As Figure 3.2 shows, it is sometimes necessary to exclude data points on the sides or in the middle. This is done in order to match up the peak with the gaussian as well as possible so that the bond length will match up with the peak. The overcalculation in the integrated area caused by excluding these points can be corrected while the bond length not matching up is harder to correct. The second and third shell peaks were more convoluted in amorphous samples than in crystalline samples. In all samples, convolution could be seen in the peaks.

The code filetoarray(), which appears in Appendix 7.1.4, converts the MATLAB parameters from the Curve Fitting App to a matrix. The output from the App needs to be pasted into a text file and then filetoarray() has to be called to obtain the matrix used in later codes.

For the first shell, a different integrating analysis was also used, but the method of using gaussians gave better results. In this other method, the peak area was found by integrating the area underneath the curve created by the data points from ~ 1.7Å to ~ 2.7Å depending on the sample. The bond length corresponded to the center of mass of this integrated area, and the FWHM was obtained by finding the range from side to side on the peak at halfway up the peak. Figure 3.3 shows an example of the center of mass, FWHM, and integrated area. This method did not give numbers consistent with EXAFS [5]. The error was also more difficult to propagate and estimate in this method.

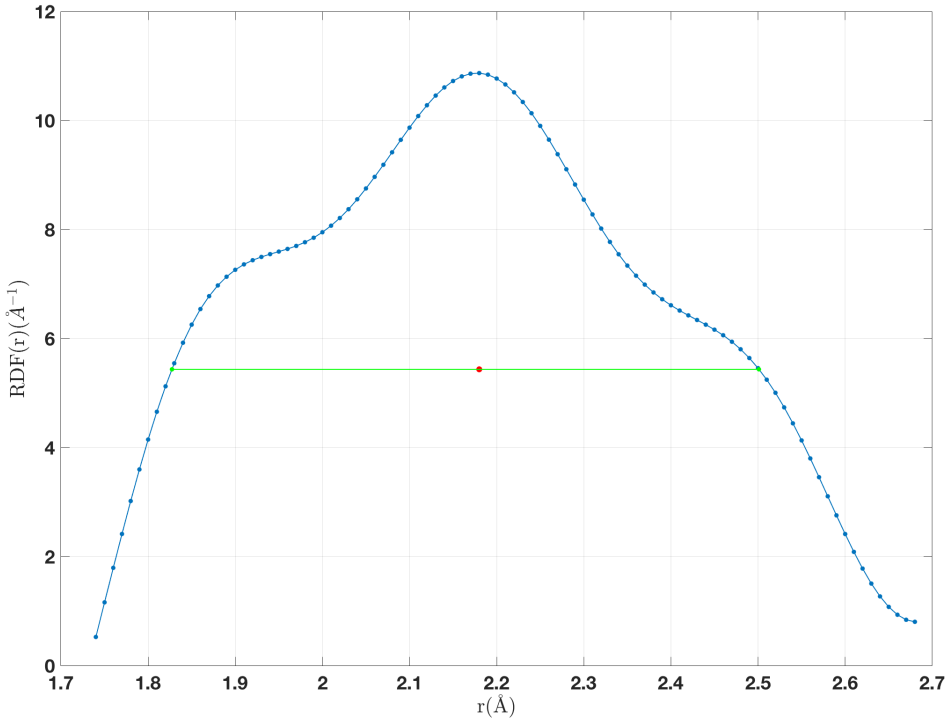


Figure 3.3: This is a fit of the first peak where the blue dots correspond to the raw data, and the blue line is those dots connected. The red dot corresponds to the center of mass, and the green line corresponds to the FWHM of the peak.

3.1.1 Bond Length

The bond length was obtained from the parameter b of the gaussian fit of the Curve Fitting App. Error propagation, in this case, is fairly easy, since MATLAB outputs confidence interval in its Curve Fitting App.

The bond length did not always match with what was expected. The gaussians chosen by MATLAB would sometimes be wrong or would make the integrated area negative, which does not make physical sense. Figure 3.4 shows an instance where MATLAB initially fitted the peaks incorrectly.

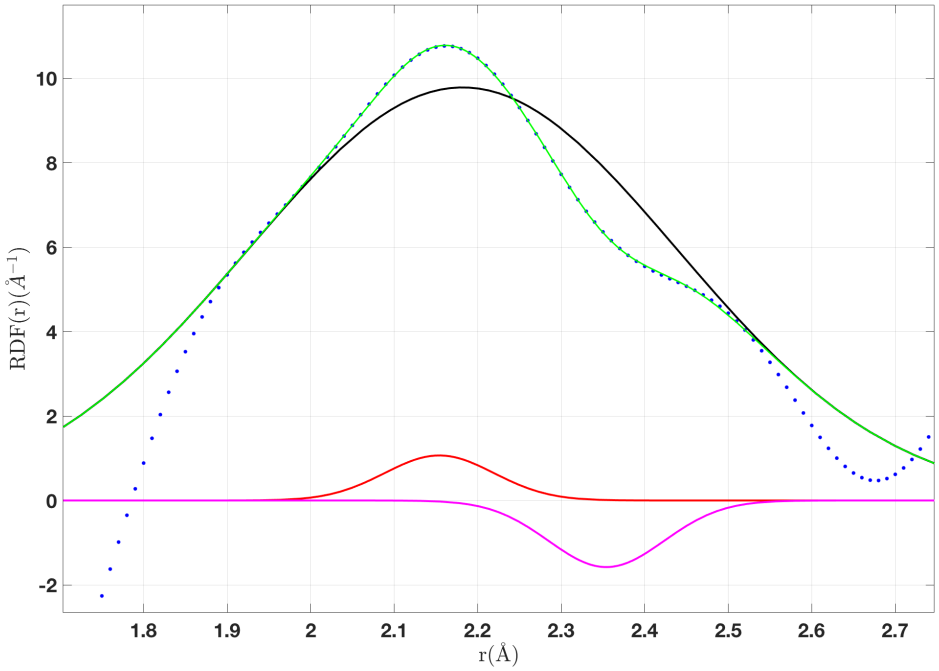


Figure 3.4: This is a fit of a sample's first shell. The blue dots correspond to the raw data. The green fit corresponds to the convolution of the gaussians. The red, black, and pink curves correspond to the individual peaks used in the fit. The pink curve gives a negative area and the black curve overcalculates the main peak and shifts the bond length to the right.

This was corrected manually by setting where MATLAB looks for the peaks in the Curve Fitting App. The initial values for a, b, and c can be set or fixed. In this case, the initial value was set, and MATLAB was then able to fit the peaks better and with more realistic values that would not produce a negative integrated area. Figure 3.5 shows the same curve after setting the initial values, which results in a better fit.

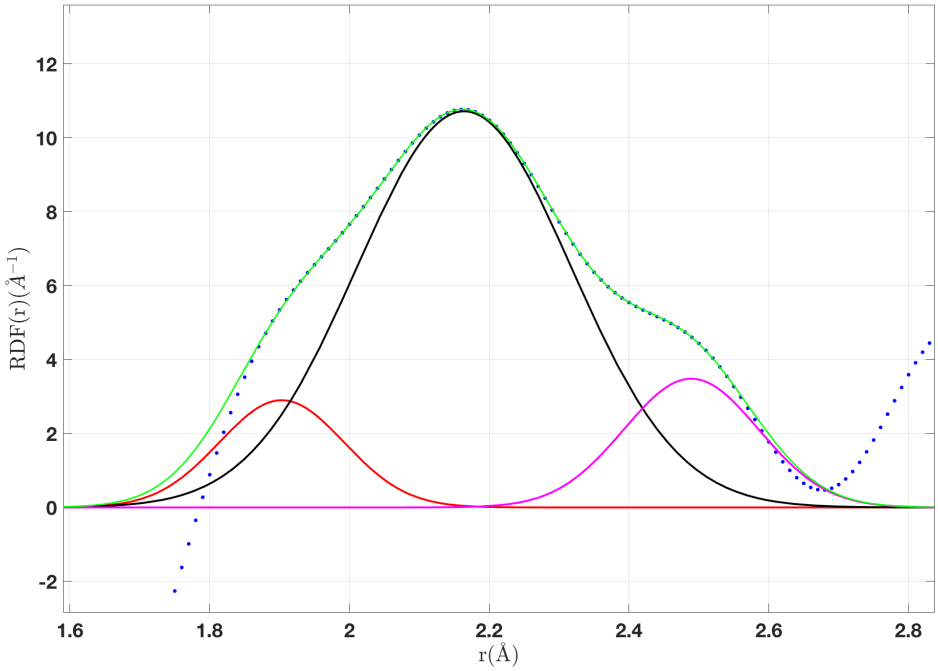


Figure 3.5: This is a fit of a sample's first shell. The blue dots correspond to the raw data. The green fit corresponds to the convolution of the gaussians. The red, black, and pink curves correspond to the individual peaks used in the fit. The black fit corresponds to the first peak which is used to get the bond length.

3.1.2 Full Width at Half Maximum

The full width at half maximum was obtained from the Gaussian fit of the Curve Fitting App by using the variable c and Eq. 2.14. The FWHM is calculated just for the peak of interest. Figure 3.6 shows the FWHM calculated for the second and third shells.

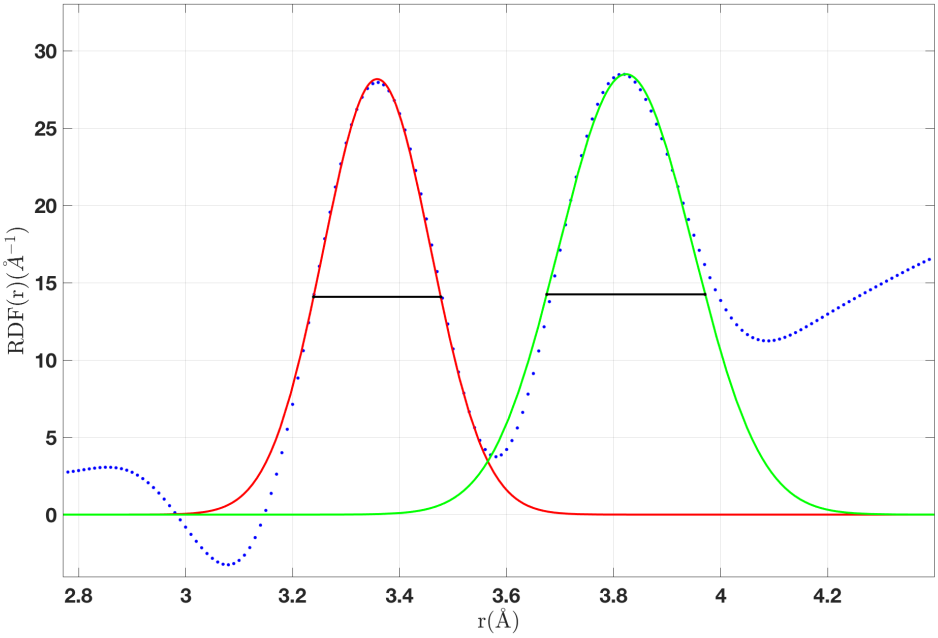


Figure 3.6: This is a fit of the second and third shells where the blue dots correspond to the raw data. The red curve is the second shell gaussian and the green curve is the third shell gaussian. The black lines correspond to the FWHM.

The FWHM is a good measure of how much the bond lengths vary. A smaller FWHM value corresponds to more uniform bond lengths in the structure. Once the FWHM was calculated, the error was propagated using Eq. 2.15. Larger FWHM values were obtained for amorphous samples. As the crystallinity in the samples increased, the peaks became narrower, due to an increased order in the atomic structure.

3.1.3 Integrated Area

The integrated area was found by calculating the area of the individual gaussian peaks. Figure 3.7 shows how the first shell was fitted using gaussians. In the first shell, two approaches were used to get the integrated area. The first approach used only the main middle peak, and the second approach used the strongest middle peak and the peak to the left. The second approach gave numbers that were more consistent with previous EXAFS experiments that found that the coordination number for the first In-O shell was around six [5].

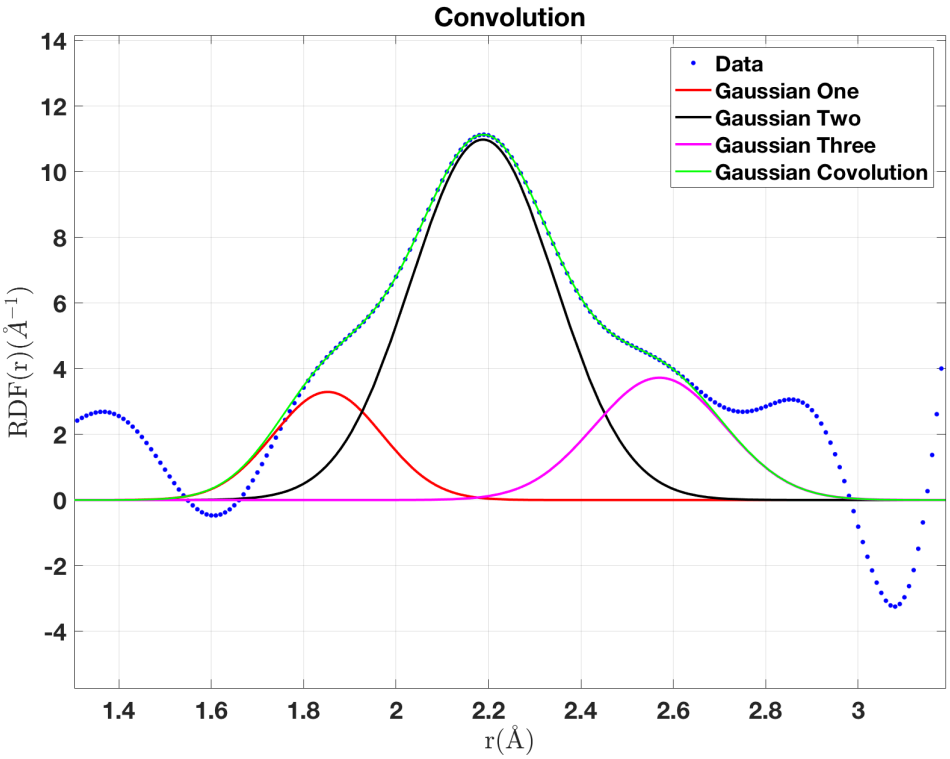


Figure 3.7: This is a fit of a sample's first shell. The blue dots correspond to the raw data. The green fit corresponds to the convolution of the gaussians. The red, black, and pink curves correspond to the individual peaks used in the fit. The black fit corresponds to the first peak which is used to get the bond length and FWHM. The area under the black fit and red fit are used to get the integrated area.

For the second and third shells, one gaussian per shell was used to find the integrated area. For some samples, the fit did not perfectly pass through all the data points on the left of the second shell and/or in-between the second and third shells, leaving a gap which can be seen in Figure 3.8. Therefore, a method was developed to correct this error. For the overcalculation on the left of the second shell, the overcalculated area was calculated and then subtracted from the second shell. For the overcalculation in-between the second and third shells, the intersection point of the two gaussians was found first. This point was used to decide what percentage of the overcalculated area should be subtracted from each gaussian. The overcalculation to the left of the intersection point was subtracted from the gaussian under the second shell while everything to the right was subtracted from the gaussian under the third shell. This gives a more accurate value for the integrated area for each shell.

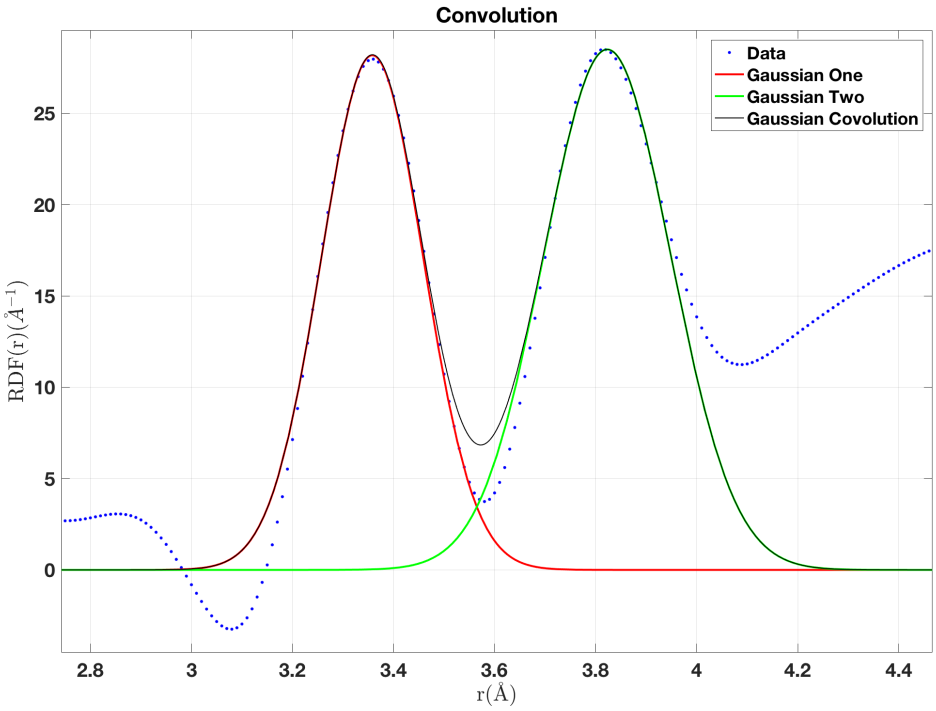


Figure 3.8: This is a fit of a sample's second (red fit) and third (green fit) shells. The blue dots correspond to the raw data. The black fit corresponds to the convolution of the gaussians. The blue dots and black fit show the gap between the fit and experimental data. The red fit and green fit show the individual gaussians used to find the FWHM, bond length, and integrated area.

The integrated area was calculated using Eq. 2.16, and the error was propagated using 2.20 when one gaussian was used and 2.21 when two gaussians were used.

When the area was subtracted the error was propagated using percentage errors. The percentage difference in the error of the integrated area divided by original integrated area was calculated first. This percentage was then multiplied by the calculated subtracted area to get the error in the subtracted area.

In MATLAB, a script was created to automatically find the integrated area and propagate the error. Three different codes were created to account for the different situations. The first code called split(), found in Appendix 7.4.1, finds the integrated area and propagates the error when there is no overcalculation. The second code called GaussianAddition(), in Appendix 7.4.4, finds the integrated area of two gaussians, adds them together, and propagates the error. The last code called Subtractanderror(), found in Appendix 7.4.5, finds the integrated area of the second and third shells, subtracts the overcalculated area, and propagates the errors. Each code saves all the important values to a text file, such as integrated area and gaussian parameter values. This was

done so that these values can be copied to an excel file. These three codes call GaussianArea() and GaussianArea1() to calculate the integrated areas once all the preprocessing is done. These codes can be seen in Appendix 7.4.2 and Appendix 7.4.3 respectively.

3.1.4 Graphing

The data collected and analyzed have to be graphed for interpretation, so a set of codes were developed to plot all graphs needed in one run. The analyzed data, which include bond length, FWHM, and integrated area, are kept in an excel file. This allows for data to be easily updated or viewed if necessary. The graphing code reads from this excel file. There is a main function that calls sub graphing functions which can also be called on their own. This main function plots all-new graphs or a specific subsection of the data can be graphed by calling individual functions. There are three main graphing code categories for each peak. The categories are bond length, FWHM, and integrated area. Each is then separated into peak one, peak two, and so forth for all peaks that are being examined. The excel file contains the best-fit values along with the minimum and maximum values that are acquired through error propagation.

There are multiple graphs created by each function, and each graph contains in-plane, out-of-plane, and average data along with error bars for each. In some cases, the upper and lower error bars are different. Surf plots are also created when two different variables are being studied at the same time, for example, temperature and incident angle. The codes automatically create filenames depending on data in the excel files and save the graphs as PNGs (Portable Network Graphics) and FIGs (MATLAB figure files). This is useful when hundreds of graphs need to be made at one time. The graphing codes automate graphing and save the user time.

Chapter 4
Conclusion

PDF analysis reveals the local atomic structure of both amorphous and crystalline samples. This analysis of TCOs help in the quest to find better materials with improved properties, such as electron mobility and mechanical flexibility. A better methodology to analyze PDF data of TCO samples was developed to speed up the analysis of multiple samples. This methodology is currently being used to study different series of amorphous and crystalline TCOs. A clear procedure allowed for the automation of certain parts of the analysis through coding making it easier to compare different TCOs and see trends in the data. Codes were developed to easily calculate the FWHM, bond length, and coordination number. The codes have been used to analyze the local atomic structure in amorphous and crystalline TCO samples. Trends and differences in bond lengths and coordination numbers can be correlated to measured electrical properties. The developed procedure will help in the analysis of further doped IO thin films. This analysis along with future samples studied can help guide the synthesis of indium-based TCOs with improved properties. The methodology developed in this project will be applied to several ongoing and future projects involving different types of TCOs.

Chapter 5
Acknowledgements

The NSF-MRSEC grant number DMR-1121262 funded the project on amorphous and crystalline TCOs. The Office of Science, Office of Basic Energy Sciences in the U.S. Department of Energy, under contract # DE-AC02- 06CH11357 supported the use of the Advanced Photon Source. I would like to acknowledge Dr. John Okasinski from Argonne National Laboratory, Dr. Bruce Buchholtz, and Prof. Robert Chang from Northwestern University. They provided the samples, measured their electrical properties, and helped with the x-ray measurements. I would also like to acknowledge my thesis director Professor Gabriela González Avilés and my faculty reader Professor Anuj Sarma. I am grateful to all of those with whom I have had the pleasure to work with during this project.

Chapter 6
Bibliography

[1] Lewis, Brian G., and David C. Paine. "Application and Processing of Transparent Conducting Oxides." *MRS Bulletin*, 25.8 (2000): 22-27.

[2] Bsaibes, Thomas. "Radial-Distribution Function of a Transparent Conducting Oxide." DePaul University, 2017.

[3] González Gabriela B. "Investigating the Defect Structures in the Transparent Conducting Oxides Using X-ray and Neutron Scattering Techniques." *Materials,* 5.5 (2012): 818-850.

[4] Egami, Takeshi, and Simon JL Billinge. *Underneath the Bragg Peaks: Structural Analysis of Complex Materials*. Elsevier, 2012.

[5] Buchholz, D. Bruce, et al. "The Structure and Properties of Amorphous Indium Oxide." *Chemistry of Materials* 26.18 (2014): 5401-5411.

[6] Buchholz, D. Bruce, et al. "Differences between amorphous indium oxide thin films". *Progress in Natural Science: Materials International* 23.5 (2013): 475-480.

[7] González, G. B., et al. "Relationship between electrical properties and crystallization of indium oxide thin films using ex-situ grazing-incidence wide-angle x-ray scattering." *Journal of Applied Physics* 121.20 (2017): 205306.

[8] Toby, Brian H., and Robert B. Von Dreele. "GSAS-II: the genesis of a modern open-source all purpose crystallography software package." *Journal of Applied Crystallography* 46.2 (2013): 544-549.

[9] Juhás, Pavol, et al. "PDFgetX3: a rapid and highly automatable program for processing powder diffraction data into total scattering pair distribution functions." *Journal of Applied Crystallography* 46.2 (2013): 560-566.

[10] "Documentation" MathWorks, (2018). (R2018a). The MathWorks, Inc., Natick, MA.

Chapter 7
Appendix

7.1 Conversion

7.1.1 Gtog Function

```
function output = Gtog(PDF,avgNumDens)
%Converts from G to g
%output is r and g
%PDF = Data output of pdfgetx3, which is G(r) file
%avgNumDens = average number density of sample
pdf = fopen(PDF);
data = textscan(pdf,'%f %f','HeaderLines',28); %data is a 1 by 2 cell array
r = data1,1; %data1,1 = r values
G = data1,2; %data1,2 = intensities
g = (G./(4.*pi.*r.*avgNumDens))+1;
output = [r,g];
fclose('all');
end
```

7.1.2 gtoR Function

```
function output = gtoR(input,avgNumDens)
%Converts PDF g(r) to RDF R(r)
%output is g and R
%input = Data output of pdfgetx3, which is G(r) file
%avgNumDens = average number density of sample
g = Gtog(input,avgNumDens);%Calls Gtog function
R = 4.*pi.*g(:,1).^2.*avgNumDens.*g(:,2);%Converts PDF g(r) to RDF R(r)
output = [g(:,1),R];
end
```

7.1.3 savevariables Function

```
function savevariables(input,avgNumDens)
%Saves g and R for use in Curve Fitting app
%No output
%input = Data output of pdfgetx3, which is G(r) file
%avgNumDens = average number density of sample
[g,R]=gtoR(input,avgNumDens);%Calls gtoR function
R(1,1)=0; %Reassigns first value from NaN to 0
%% Assign for curvefitting
assignin('base','x',(g(:,1)));%Assigns g to x in workspace
assignin('base','y',(R));%Assigns R to y in workspace
end
```

7.1.4 filetoarray Function

```
function finaltext=filetoarray(file)
%put file of gaussian variables a,b,c into array
%finaltext(:,1) is values,finaltext(:,2) is mins, finaltext(:,3) is maxes
%file = text file with gaussian variables a,b,c from Curve Fitting app
fileID = fopen(file);
text1 = textscan(fileID,'%q');
fclose(fileID);
text = [text1:];
text=text.';
text = strrep(text,'(','');
text = strrep(text,')','');
text = strrep(text,',','');
text(strncmp( text, '=',1 )) = [];
text(strncmp( text, 'a',1 )) = [];
text(strncmp( text, 'b',1 )) = [];
text(strncmp( text, 'c',1 )) = [];
text=reshape(text,3,length(text)/3);
finaltext=text.';
end
```

7.2 Bond Length

7.2.1 BL Function

```
function BL(array,minarray,maxarray)
%prints out BL with minimum and maximum values
%array = gaussian variables a,b,c from Curve Fitting app
%minarray = minimum gaussian variables a,b,c from Curve Fitting app
%maxarray = maximum gaussian variables a,b,c from Curve Fitting app
for i=1:length(array)%go through all Gaussians
        if(mod(i,3)==2)
                disp('BL');
                disp(array(i));
                disp('BLmin');
                disp(minarray(i));
                disp('BLmax');
                disp(maxarray(i));
        end
end
end
```

7.3 Full Width at Half Max

7.3.1 FWHM Function

```
function FWHM(array,minarray,maxarray)
%prints out FWHM with minimum and maximum values
%array = gaussian variable c from Curve Fitting app
%minarray = minimum gaussian variable c from Curve Fitting app
%maxarray = maximum gaussian variable c from Curve Fitting app
for i=1:length(array)%go through all Gaussians
        if(mod(i,3)==0)
                %calculate fwhm from c
                fwhm=2*sqrt(2*log(2))*(array(i)/sqrt(2));
                fwhmmin=2*sqrt(2*log(2))*(minarray(i)/sqrt(2));
                fwhmmax=2*sqrt(2*log(2))*(maxarray(i)/sqrt(2));
```

```
                disp('FWHM');
                disp(fwhm);
                disp('FWHMmin');
                disp(fwhmmin);
                disp('FWHMmax');
                disp(fwhmmax);
        end
end
end
```

7.4 Integrated Area

7.4.1 split Function

```
function split(file,llength)
%finds the area with minimum and maximum areas for error
%file = text file with gaussian variables a,b,c from Curve Fitting app
%llength = array of how many Gaussians fitted per group with max of 3
fileID = fopen(file);
text1 = textscan(fileID,'%q');
fclose(fileID);
otext = [text1:];
%below gets rid of unnecessary characters in text file
otext=otext.';
otext = strrep(otext,'(','');
otext = strrep(otext,')','');
otext = strrep(otext,',','');
otext(strncmp( otext, '=',1 )) = [];
otext(strncmp( otext, 'a',1 )) = [];
otext(strncmp( otext, 'b',1 )) = [];
otext(strncmp( otext, 'c',1 )) = [];
otext=reshape(otext,3,length(otext)/3);
textt=otext.';
previndex=1;
for ll=1:length(llength)%go through all groups of Gaussians
```

```
currindex=(llength(ll))*3+previndex-1;
text=textt(previndex:currindex,:);
previndex=currindex+1;
%put in order
sort=zeros(1,length(text)/3);
for i=1:length(text)
        if(mod(i,3)==2)
                sort(((i+1)/3))=strjoin(string(cell2mat(text(i))));
        end
end
sort=sort.';
sort=sortrows(sort);
sort=sort.';
text1=zeros(length(text),3);
for i=1:length(text)
        text1(i,1)=strjoin(string(cell2mat(text(i,1))));
        text1(i,2)=strjoin(string(cell2mat(text(i,2))));
        text1(i,3)=strjoin(string(cell2mat(text(i,3))));
end
sorted=zeros(length(text),3);
k=1;
for j=1:length(sort)
        [row,col] = find(double(text1)==sort(j));
        col=col(1);
        row=row(1);
        sorted(k,1)=text1(row-1,col);
        sorted(k+1,1)=text1(row,col);
        sorted(k+2,1)=text1(row+1,col);
        sorted(k,2)=text1(row-1,col+1);
        sorted(k+1,2)=text1(row,col+1);
        sorted(k+2,2)=text1(row+1,col+1);
        sorted(k,3)=text1(row-1,col+2);
        sorted(k+1,3)=text1(row,col+2);
        sorted(k+2,3)=text1(row+1,col+2);
        k=k+3;
end
```

```
        if llength(ll)==2
                sorted(7,:)=0;
                sorted(8,:)=0;
                sorted(9,:)=0;
                llength(ll)=llength(ll)+1;
        end
        if llength(ll)==1
                sorted(4,:)=0;
                sorted(5,:)=0;
                sorted(6,:)=0;
                llength(ll)=llength(ll)+1;
        end
        %call GaussianArea to calculate areas
        GaussianArea(sorted(:,1),sorted(:,2),sorted(:,3));
        fileID = fopen('Area.txt', 'a');%text file for area
        fprintf(fileID,'\n');%add new line to separate next area written
        fclose(fileID);
end
end
```

7.4.2 GaussianArea Function

```
function GaussianArea(array,minarray,maxarray)
%calculates the area with minimum and maximum areas for error
%array = gaussian variables a,b,c from Curve Fitting app
%minarray = minimum gaussian variables a,b,c from Curve Fitting app
%maxarray = maximum gaussian variables a,b,c from Curve Fitting app
llength=3;
for i=1:length %setting variables from arrays
        k=i*3;
        if k==3
                a1=array(k-2);
                a1min=minarray(k-2);
                a1max=maxarray(k-2);
                b1=array(k-1);
                b1min=minarray(k-1);
```

```
                b1max=maxarray(k-1);
                c1=array(k);
                c1min=minarray(k);
                c1max=maxarray(k);
        elseif k==6
                a2=array(k-2);
                a2min=minarray(k-2);
                a2max=maxarray(k-2);
                b2=array(k-1);
                b2min=minarray(k-1);
                b2max=maxarray(k-1);
                c2=array(k);
                c2min=minarray(k);
                c2max=maxarray(k);
        else
                a3=array(k-2);
                a3min=minarray(k-2);
                a3max=maxarray(k-2);
                b3=array(k-1);
                b3min=minarray(k-1);
                b3max=maxarray(k-1);
                c3=array(k);
                c3min=minarray(k);
                c3max=maxarray(k);
        end
end
for ii=1:length(c2max)
        %% Find minimum and maximum area with error propagation
        deltaa1min=a1(ii)-a1min(ii);
        deltab1min=b1(ii)-b1min(ii);
        deltac1min=c1(ii)-c1min(ii);
        f =  @(x) sqrt((((2*a1(ii).*exp(-((x-b1(ii))./c1(ii)).^2).*...
        (x-b1(ii))).(c1(ii).^2)).*deltab1min).^2+(((2*a1(ii).*...
        exp(-((x-b1(ii))./c1(ii)).^2).*(x-b1(ii)).^2)./(c1(ii).^3))*...
        deltac1min).^2+deltaa1min.*exp(-((x-b1(ii))./c1(ii)).^2));
        % Peak 1 minimum area
```

```
area1min=integral(f,-Inf,Inf);
deltaa1max=a1max(ii)-a1(ii);
deltab1max=b1max(ii)-b1(ii);
deltac1max=c1max(ii)-c1(ii);
f =  @(x) sqrt((((2*a1(ii).*exp(-((x-b1(ii))./c1(ii)).^2).*...
(x-b1(ii))).^2)).*deltab1max).^2+(((2*a1(ii).*exp(-((x-...
b1(ii))./c1(ii)).^2).*(x-b1(ii)).^2)./(c1(ii).^3))*deltac1max)....
^2+deltaa1max.*exp(-((x-b1(ii))./c1(ii)).^2));
% Peak 1 maximum area
area1max=integral(f,-Inf,Inf);
deltaa2min=a2(ii)-a2min(ii);
deltab2min=b2(ii)-b2min(ii);
deltac2min=c2(ii)-c2min(ii);
f =  @(x) sqrt((((2*a2(ii).*exp(-((x-b2(ii))./c2(ii)).^2).*(x-...
b2(ii))).^2)).*deltab2min).^2+(((2*a2(ii).*exp(-((x-...
b2(ii))./c2(ii)).^2).*(x-b2(ii)).^2)./(c2(ii).^3))*deltac2min)....
^2+deltaa2min.*exp(-((x-b2(ii))./c2(ii)).^2));
% Peak 2 minimum area
area2min=integral(f,-Inf,Inf);
deltaa2max=a2max(ii)-a2(ii);
deltab2max=b2max(ii)-b2(ii);
deltac2max=c2max(ii)-c2(ii);
f =  @(x) sqrt((((2*a2(ii).*exp(-((x-b2(ii))./c2(ii)).^2).*(x-...
b2(ii))).^2)).*deltab2max).^2+(((2*a2(ii).*exp(-((x-...
b2(ii))./c2(ii)).^2).*(x-b2(ii)).^2)./(c2(ii).^3))*deltac2max)....
^2+deltaa2max.*exp(-((x-b2(ii))./c2(ii)).^2));
% Peak 2 maximum area
area2max=integral(f,-Inf,Inf);
deltaa3min=a3(ii)-a3min(ii);
deltab3min=b3(ii)-b3min(ii);
deltac3min=c3(ii)-c3min(ii);
f =  @(x) sqrt((((2*a3(ii).*exp(-((x-b3(ii))./c3(ii)).^2).*(x-...
b3(ii))).^2)).*deltab3min).^2+(((2*a3(ii).*exp(-((x-...
b3(ii))./c3(ii)).^2).*(x-b3(ii)).^2)./(c3(ii).^3))*deltac3min)....
^2+deltaa3min.*exp(-((x-b3(ii))./c3(ii)).^2));
% Peak 3 minimum area
```

```
area3min=integral(f,-Inf,Inf);
deltaa3max=a3max(ii)-a3(ii);
deltab3max=b3max(ii)-b3(ii);
deltac3max=c3max(ii)-c3(ii);
f =  @(x) sqrt((((2*a3(ii).*exp(-((x-b3(ii))./c3(ii)).^2).*(x-...
b3(ii)))./(c3(ii).^2)).*deltab3max).^2+(((2*a3(ii).*exp(-((x-...
b3(ii))./c3(ii)).^2).*(x-b3(ii)).^2)./(c3(ii).^3))*deltac3max)....
^2+deltaa3max.*exp(-((x-b3(ii))./c3(ii)).^2));
% Peak 3 maximum area
area3max=integral(f,-Inf,Inf);
% Find area
originalareaP1=GaussianArea1([a1(ii),b1(ii),c1(ii)]);
originalareaP2=GaussianArea1([a2(ii),b2(ii),c2(ii)]);
originalareaP3=GaussianArea1([a3(ii),b3(ii),c3(ii)]);
if isnan(area3min)
        area3min=0;
end
if isnan(area3max)
        area3max=0;
end
fileID = fopen('Area.txt', 'a');
fprintf(fileID,strcat(num2str(a1(ii),'%.4f'),',',num2str(b1(ii),...
'%.4f'),',',num2str(c1(ii),'%.4f'),','));
fprintf(fileID,strcat(num2str(a2(ii),'%.4f'),',',num2str(b2(ii),...
'%.4f'),',',num2str(c2(ii),'%.4f'),','));
fprintf(fileID,strcat(num2str(a3(ii),'%.4f'),',',num2str(b3(ii),...
'%.4f'),',',num2str(c3(ii),'%.4f'),','));
fprintf(fileID,',    ,');
fprintf(fileID,num2str(originalareaP1),'%.4f');% Peak 1 area
fprintf(fileID,',');
fprintf(fileID,num2str(originalareaP2),'%.4f');% Peak 2 area
fprintf(fileID,',');
fprintf(fileID,num2str(originalareaP3),'%.4f');% Peak 3 area
fprintf(fileID,',    ,');
fprintf(fileID,strcat(num2str(a1min(ii),'%.4f'),',',num2str(...
b1min(ii),'%.4f'),',',num2str(c1min(ii),'%.4f'),','));
```

```
        fprintf(fileID,strcat(num2str(a2min(ii),'%.4f'),',',num2str(...
        b2min(ii),'%.4f'),',',num2str(c2min(ii),'%.4f'),','));
        fprintf(fileID,strcat(num2str(a3min(ii),'%.4f'),',',num2str(...
        b3min(ii),'%.4f'),',',num2str(c3min(ii),'%.4f')));
        fprintf(fileID,',    ,');
        % Peak 1 min area
        fprintf(fileID,num2str(originalareaP1-area1min),'%.4f');
        fprintf(fileID,',');
        % Peak 2 min area
        fprintf(fileID,num2str(originalareaP2-area2min),'%.4f');
        fprintf(fileID,',');
        % Peak 3 min area
        fprintf(fileID,num2str(originalareaP3-area3min),'%.4f');
        fprintf(fileID,',    ,');
        fprintf(fileID,strcat(num2str(a1max(ii),'%.4f'),',',num2str(...
        b1max(ii),'%.4f'),',',num2str(c1max(ii),'%.4f'),','));
        fprintf(fileID,strcat(num2str(a2max(ii),'%.4f'),',',num2str(...
        b2max(ii),'%.4f'),',',num2str(c2max(ii),'%.4f'),','));
        fprintf(fileID,strcat(num2str(a3max(ii),'%.4f'),',',num2str(...
        b3max(ii),'%.4f'),',',num2str(c3max(ii),'%.4f')));
        fprintf(fileID,',    ,');
        % Peak 1 max area
        fprintf(fileID,num2str(originalareaP1+area1max),'%.4f');
        fprintf(fileID,',');
        % Peak 2 max area
        fprintf(fileID,num2str(originalareaP2+area2max),'%.4f');
        fprintf(fileID,',');
        % Peak 3 max area
        fprintf(fileID,num2str(originalareaP3+area3max),'%.4f');
        fprintf(fileID,',    ,');
        fclose(fileID);
end
end
```

7.4.3 GaussianArea1 Function

```
function area=GaussianArea1(array)
%calculates the area of Gaussian
%outputs an area
%array = gaussian variables a,b,c from Curve Fitting app
a=array(1);
b=array(2);
c=array(3);
f =  @(x) a.*exp(-((x-b)./c).^2);
area=integral(f,-Inf,Inf);
k=k+3;
end
```

7.4.4 GaussianAddition Function

```
function GaussianAddition()
%Adds two Gaussians together and propagates error
%prints out area with minimum and maximum values
y= xlsread('filename','Sheet','xlRange');
y1= xlsread('filename','Sheet','xlRange');
ymax= xlsread('filename','Sheet','xlRange');
y1max= xlsread('filename','Sheet','xlRange');
ymin= xlsread('filename','Sheet','xlRange');
y1min= xlsread('filename','Sheet','xlRange');
y(isnan(y)) = [];
y1(isnan(y1)) = [];
ymin(isnan(ymin)) = [];
y1min(isnan(y1min)) = [];
ymax(isnan(ymax)) = [];
y1max(isnan(y1max)) = [];
ytot=y+y1;
ytotmin=sqrt((y-ymin).^2+(y1-y1min).^2);%propagation of error
ytotmax=sqrt((ymax-y).^2+(y1max-y1).^2);%propagation of error
disp('Area');
disp(ytot);
```

```
disp('Areamin');
disp(ytotmin);
disp('Areamax');
disp(ytotmax);
end
```

7.4.5 Subtractanderror Function

```
function Subtractanderror(avgNumDens)
%subtracts overcalculation in gaussians and propagates error
%avgNumDens = average number density of sample
%% Read in data
% file name
[~,filestemp,~]= xlsread('filename','Sheet','xlRange');
filestemp=cell2str(filestemp);
files=filestemp([3:12,18:26,32:38,40,46:52,54],:);
types=filestemp([1,16,30,44],:);
% coefficients a1,b1,c1,a2,b2,c2
a1=xlsread('filename','Sheet','xlRange');
a1(isnan(a1)) = [];
b1=xlsread('filename','Sheet','xlRange');
b1(isnan(b1)) = [];
c1=xlsread('filename','Sheet','xlRange');
c1(isnan(c1)) = [];
a2=xlsread('filename','Sheet','xlRange');
a2(isnan(a2)) = [];
b2=xlsread('filename','Sheet','xlRange');
b2(isnan(b2)) = [];
c2=xlsread('filename','Sheet','xlRange');
c2(isnan(c2)) = [];
% Min coefficients a1,b1,c1,a2,b2,c2
a1min=xlsread('filename','Sheet','xlRange');
a1min(isnan(a1min)) = [];
b1min=xlsread('filename','Sheet','xlRange');
b1min(isnan(b1min)) = [];
c1min=xlsread('filename','Sheet','xlRange');
```

```
c1min(isnan(c1min)) = [];
a2min=xlsread('filename','Sheet','xlRange');
a2min(isnan(a2min)) = [];
b2min=xlsread('filename','Sheet','xlRange');
b2min(isnan(b2min)) = [];
c2min=xlsread('filename','Sheet','xlRange');
c2min(isnan(c2min)) = [];
% Max coefficients a1,b1,c1,a2,b2,c2
a1max=xlsread('filename','Sheet','xlRange');
a1max(isnan(a1max)) = [];
b1max=xlsread('filename','Sheet','xlRange');
b1max(isnan(b1max)) = [];
c1max=xlsread('filename','Sheet','xlRange');
c1max(isnan(c1max)) = [];
a2max=xlsread('filename','Sheet','xlRange');
a2max(isnan(a2max)) = [];
b2max=xlsread('filename','Sheet','xlRange');
b2max(isnan(b2max)) = [];
c2max=xlsread('filename','Sheet','xlRange');
c2max(isnan(c2max)) = [];
for ii=1:length(c2max)
  %% Subtract Area
  if ii<11
    type=types(1,:);
  elseif ii<20
    type=types(2,:);
  elseif ii<28
    type=types(3,:);
  else
    type=types(4,:);
  end
  strfile=(strcat('filepath/',type,'/',files(ii,:)));
  g = Gtog(strfile,avgNumDens);%Calls Gtog function input is G(r) file
  R = 4.*pi.*g(:,1).^2.*avgNumDens.*g(:,2);%Converts PDF g(r) to RDF R(r)
  R(1,1)=0;
  gg=g(:,1);
```

```
h= @(x) (a1(ii).*exp(-((x-b1(ii))./c1(ii)).^2))-(a2(ii).*exp(-((x...
-b2(ii))./c2(ii)).^2));
firstzero=fzero(h,3.5); %intersection point
xs2 = linspace(3,4,1000);
Gauss =  (a1(ii).*exp(-((xs2-b1(ii))./c1(ii)).^2))+(a2(ii).*exp(-((xs2...
-b2(ii))./c2(ii)).^2));
[~,y]=findpeaks(R,gg);
[~,ys2]=findpeaks(Gauss*-1,xs2);
%% if not NaN
if ~isnan(firstzero)
  for i=1:length(y)
    if y(i)>=3.1&&y(i)<3.5
      z=y(i);
    end
    if y(i)>=3.5&&y(i)<4
      zz=y(i);
      break;
    end
  end
  difpoints=[];
  difx=[];
  difpoints2=[];
  index=1;
  for i=2:length(gg)
    if(gg(i)>=z && gg(i)<=firstzero)
      value=(a1(ii)*exp(-((gg(i)-b1(ii))/c1(ii))^2))+(a2(ii)*exp(-((...
      gg(i)-b2(ii))/c2(ii))^2));
      if R(i)<value
        difpoints(index)=R(i);
        difx(index)=gg(i);
        difpoints2(index)=value;
        index=index+1;
    end
    end
  end
  if ~isempty(difx)
```

```
  area1=trapz(difx,difpoints);
  area2=trapz(difx,difpoints2);
else
  area1=0;
  area2=0;
end
subtracted1=area2-area1;
% area to subtract from peak 2
% Original area of Peak 2
originalareaP1=GaussianArea1([a1(ii),b1(ii),c1(ii)]);
newarea1=originalareaP1-subtracted1;
difpoints=[];
difx=[];
difpoints2=[];
index=1;
for i=2:length(gg)
  if(gg(i)>=firstzero && gg(i)<=zz)
    value=(a1(ii)*exp(-((gg(i)-b1(ii))/c1(ii))^2))+(a2(ii)*exp(...
    -((gg(i)-b2(ii))/c2(ii))^2));
    if R(i)<value
      difpoints(index)=R(i);
      difx(index)=gg(i);
      difpoints2(index)=value;
      index=index+1;
    end
  end
end
if ~isempty(difx)
  area1=trapz(difx,difpoints);
  area2=trapz(difx,difpoints2);
else
  area1=0;
  area2=0;
end
subtracted2=area2-area1;
% area to subtract from peak 3
```

```
% Original area of Peak 3
originalareaP2=GaussianArea1([a2(ii),b2(ii),c2(ii)]);
newarea2=originalareaP2-subtracted2;
difpoints=[];
difx=[];
difpoints2=[];
index=1;
for i=2:length(gg)
  if(gg(i)>=2 && gg(i)<=z)
    value=(a1(ii)*exp(-((gg(i)-b1(ii))/c1(ii))^2))+(a2(ii)*exp(-((...
    gg(i)-b2(ii))/c2(ii))^2));
    if R(i)<value
      difpoints(index)=R(i);
      if R(i)<0
        difpoints(index)=0;
      end
      difx(index)=gg(i);
      difpoints2(index)=value;
      index=index+1;
    end
  end
end
area1=trapz(difx,difpoints);
area2=trapz(difx,difpoints2);
subtracted3=area2-area1;
% area peak 2
newarea4=newarea1-subtracted3;
%% Find error
deltaa1min=a1(ii)-a1min(ii);
deltab1min=b1(ii)-b1min(ii);
deltac1min=c1(ii)-c1min(ii);
f =  @(x) sqrt((((2*a1(ii).*exp(-((x-b1(ii))./c1(ii)).^2).*(x-...
b1(ii))).*(c1(ii).^2)).*deltab1min).^2+(((2*a1(ii).*exp(-((x-...
b1(ii)).*c1(ii)).^2).*(x-b1(ii)).^2).*(c1(ii).^3))*deltac1min)...
.^2+deltaa1min.*exp(-((x-b1(ii)).*c1(ii)).^2));
area1min=integral(f,-Inf,Inf); % Peak 2 min
```

```
deltaa1max=a1max(ii)-a1(ii);
deltab1max=b1max(ii)-b1(ii);
deltac1max=c1max(ii)-c1(ii);
f =  @(x) sqrt((((2*a1(ii).*exp(-((x-b1(ii))./c1(ii)).^2).*(x...
-b1(ii))).^2)).*deltab1max).^2+(((2*a1(ii).*exp(-((x...
-b1(ii))./c1(ii)).^2).*(x-b1(ii)).^2)./(c1(ii).^3))*deltac1max)...
.^2+deltaa1max.*exp(-((x-b1(ii))./c1(ii)).^2));
area1max=integral(f,-Inf,Inf); % Peak 2 max
deltaa2min=a2(ii)-a2min(ii);
deltab2min=b2(ii)-b2min(ii);
deltac2min=c2(ii)-c2min(ii);
f =  @(x) sqrt((((2*a2(ii).*exp(-((x-b2(ii))./c2(ii)).^2).*(x...
-b2(ii))).^2)).*deltab2min).^2+(((2*a2(ii).*exp(-((x...
-b2(ii))./c2(ii)).^2).*(x-b2(ii)).^2)./(c2(ii).^3))*deltac2min)...
.^2+deltaa2min.*exp(-((x-b2(ii))./c2(ii)).^2));
area2min=integral(f,-Inf,Inf); % Peak 3 min
deltaa2max=a2max(ii)-a2(ii);
deltab2max=b2max(ii)-b2(ii);
deltac2max=c2max(ii)-c2(ii);
f =  @(x) sqrt((((2*a2(ii).*exp(-((x-b2(ii))./c2(ii)).^2).*(x...
-b2(ii))).^2)).*deltab2max).^2+(((2*a2(ii).*exp(-((x...
-b2(ii))./c2(ii)).^2).*(x-b2(ii)).^2)./(c2(ii).^3))*deltac2max)...
.^2+deltaa2max.*exp(-((x-b2(ii))./c2(ii)).^2));
area2max=integral(f,-Inf,Inf); % Peak 3 max
% percentages
area1maxperc=area1max/originalareaP1;
area1minperc=area1min/originalareaP1;
area2maxperc=area2max/originalareaP2;
area2minperc=area2min/originalareaP2;
P2min=-newarea4*area1minperc+newarea4;
P2max=newarea4*area1maxperc+newarea4;
P3min=-newarea2*area2minperc+newarea2;
P3max=newarea2*area2maxperc+newarea2;
%% save
fileID = fopen('area.txt', 'a');
fprintf(fileID,num2str(firstzero));     % intersection
```

```
    fprintf(fileID,',    ');
    % left of intersection (area from peak 2)
    fprintf(fileID,num2str(subtracted1));
    fprintf(fileID,',    ');
    % right of intersection (area from peak 3)
    fprintf(fileID,num2str(subtracted2));
    fprintf(fileID,',    ,');
    fprintf(fileID,num2str(subtracted3));   % leftside (area from peak 2)
    fprintf(fileID,',    ,');
    % Peak 2 min area with subtraction
    fprintf(fileID,num2str(P2min));
    fprintf(fileID,',    ');
    fprintf(fileID,num2str(newarea4));      % Peak 2 area with subtraction
    fprintf(fileID,',    ');
    % Peak 2 max area with subtraction
    fprintf(fileID,num2str(P2max));
    fprintf(fileID,',    ');
    % Peak 3 min area with subtraction
    fprintf(fileID,num2str(P3min));
    fprintf(fileID,',    ');
    fprintf(fileID,num2str(newarea2));      % Peak 3 area with subtraction
    fprintf(fileID,',    ');
    % Peak 3 max area with subtraction
    fprintf(fileID,num2str(P3max));
    fprintf(fileID,'\n');
    fclose(fileID);
  else
  %% if is NaN
    for i=1:length(y)
      if y(i)>=3.1&&y(i)<3.5
        z=y(i);
      end
      if y(i)>=3.5&&y(i)<4
        zz=y(i);
        break;
      end
```

```
end
firstzero=ys2;
difpoints=[];
difx=[];
difpoints2=[];
index=1;
for i=2:length(gg)
  if(gg(i)>=z && gg(i)<=firstzero)
    value=(a1(ii)*exp(-((gg(i)-b1(ii))/c1(ii))^2))+(a2(ii)*exp(-((...
    gg(i)-b2(ii))/c2(ii))^2));
    if R(i)<value
      difpoints(index)=R(i);
      difx(index)=gg(i);
      difpoints2(index)=value;
      index=index+1;
    end
  end
end
if ~isempty(difx)
  area1=trapz(difx,difpoints);
  area2=trapz(difx,difpoints2);
else
  area1=0;
  area2=0;
end
subtracted1=area2-area1;
% area to subtract from peak 2
% Original area of Peak 2
originalareaP1=GaussianArea1([a1(ii),b1(ii),c1(ii)]);
newarea1=originalareaP1-subtracted1;
difpoints=[];
difx=[];
difpoints2=[];
index=1;
for i=2:length(gg)
  if(gg(i)>=z && gg(i)<=zz)
```

```
    value=(a1(ii)*exp(-((gg(i)-b1(ii))/c1(ii))^2))+(a2(ii)*exp(-((...
    gg(i)-b2(ii))/c2(ii))^2));
    if R(i)<value
      difpoints(index)=R(i);
      difx(index)=gg(i);
      difpoints2(index)=value;
      index=index+1;
    end
  end
end
if ~isempty(difx)
  area1=trapz(difx,difpoints);
  area2=trapz(difx,difpoints2);
else
  area1=0;
  area2=0;
end
subtracted2=area2-area1;
% area to subtract from peak 3
% Original area of Peak 3
originalareaP2=GaussianArea1([a2(ii),b2(ii),c2(ii)]);
newarea2=originalareaP2-subtracted2;
difpoints=[];
difx=[];
difpoints2=[];
index=1;
for i=2:length(gg)
  if(gg(i)>=2 && gg(i)<=z)
    value=(a1(ii)*exp(-((gg(i)-b1(ii))/c1(ii))^2))+(a2(ii)*exp(-((...
    gg(i)-b2(ii))/c2(ii))^2));
    if R(i)<value
      difpoints(index)=R(i);
      if R(i)<0
        difpoints(index)=0;
      end
      difx(index)=gg(i);
```

```
      difpoints2(index)=value;
      index=index+1;
    end
  end
end
area1=trapz(difx,difpoints);
area2=trapz(difx,difpoints2);
subtracted3=area2-area1;
% area peak 2
newarea4=newarea1-subtracted3;
%% Find error
deltaa1min=a1(ii)-a1min(ii);
deltab1min=b1(ii)-b1min(ii);
deltac1min=c1(ii)-c1min(ii);
f =  @(x) sqrt((((2*a1(ii).*exp(-((x-b1(ii))./c1(ii)).^2).*(x...
-b1(ii)))./(c1(ii).^2)).*deltab1min).^2+(((2*a1(ii).*exp(-((x...
-b1(ii))./c1(ii)).^2).*(x-b1(ii)).^2)./(c1(ii).^3))*deltac1min)...
.^2+deltaa1min.*exp(-((x-b1(ii))./c1(ii)).^2));
area1min=integral(f,-Inf,Inf); % Peak 2 min
deltaa1max=a1max(ii)-a1(ii);
deltab1max=b1max(ii)-b1(ii);
deltac1max=c1max(ii)-c1(ii);
f =  @(x) sqrt((((2*a1(ii).*exp(-((x-b1(ii))./c1(ii)).^2).*(x...
-b1(ii)))./(c1(ii).^2)).*deltab1max).^2+(((2*a1(ii).*exp(-((x...
-b1(ii))./c1(ii)).^2).*(x-b1(ii)).^2)./(c1(ii).^3))*deltac1max)...
.^2+deltaa1max.*exp(-((x-b1(ii))./c1(ii)).^2));
area1max=integral(f,-Inf,Inf); % Peak 2 max
deltaa2min=a2(ii)-a2min(ii);
deltab2min=b2(ii)-b2min(ii);
deltac2min=c2(ii)-c2min(ii);
f =  @(x) sqrt((((2*a2(ii).*exp(-((x-b2(ii))./c2(ii)).^2).*(x...
-b2(ii)))./(c2(ii).^2)).*deltab2min).^2+(((2*a2(ii).*exp(-((x...
-b2(ii))./c2(ii)).^2).*(x-b2(ii)).^2)./(c2(ii).^3))*deltac2min)...
.^2+deltaa2min.*exp(-((x-b2(ii))./c2(ii)).^2));
area2min=integral(f,-Inf,Inf); % Peak 3 min
deltaa2max=a2max(ii)-a2(ii);
```

```
deltab2max=b2max(ii)-b2(ii);
deltac2max=c2max(ii)-c2(ii);
f =  @(x) sqrt((((2*a2(ii).*exp(-((x-b2(ii))./c2(ii)).^2).*(x...
-b2(ii)))./(c2(ii).^2)).*deltab2max).^2+(((2*a2(ii).*exp(-((x...
-b2(ii))./c2(ii)).^2).*(x-b2(ii)).^2)./(c2(ii).^3))*deltac2max)...
.^2+deltaa2max.*exp(-((x-b2(ii))./c2(ii)).^2));
area2max=integral(f,-Inf,Inf); % Peak 3 max
% percentages
area1maxperc=area1max/originalareaP1;
area1minperc=area1min/originalareaP1;
area2maxperc=area2max/originalareaP2;
area2minperc=area2min/originalareaP2;
P2min=-newarea4*area1minperc+newarea4;
P2max=newarea4*area1maxperc+newarea4;
P3min=-newarea2*area2minperc+newarea2;
P3max=newarea2*area2maxperc+newarea2;
%% save
fileID = fopen('area.txt', 'a');
fprintf(fileID,num2str(firstzero));     % intersection
fprintf(fileID,',    ');
% left of intersection (area from peak 2)
fprintf(fileID,num2str(subtracted1));
fprintf(fileID,',    ');
% right of intersection (area from peak 3)
fprintf(fileID,num2str(subtracted2));
fprintf(fileID,',    ,');
fprintf(fileID,num2str(subtracted3));   % leftside (area from peak 2)
fprintf(fileID,',    ,');
% Peak 2 min area with subtraction
fprintf(fileID,num2str(P2min));
fprintf(fileID,',    ');
fprintf(fileID,num2str(newarea4));      % Peak 2 area with subtraction
fprintf(fileID,',    ');
% Peak 2 max area with subtraction
fprintf(fileID,num2str(P2max));
fprintf(fileID,',    ');
```

```
    % Peak 3 min area with subtraction
    fprintf(fileID,num2str(P3min));
    fprintf(fileID,',    ');
    fprintf(fileID,num2str(newarea2));      % Peak 3 area with subtraction
    fprintf(fileID,',    ');
    % Peak 3 max area with subtraction
    fprintf(fileID,num2str(P3max));
    fprintf(fileID,'\n');
    fclose(fileID);
  end
end
end
```