To the Graduate Council:

I am submitting herewith a thesis written by Kyle Peterson entitled, “Evaluation of Ellipsometric Monitors for Process Control of High Temperature Superconductors.” I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Physics.

Lloyd Davis, Major Professor

We have read this thesis and recommend its acceptance:

James W.L. Lewis

Chris Pariggers

Accepted for the Council:

Dr. C. W. Minkel

Associate Vice Chancellor and Dean of The Graduate School

(Original signatures are on file in the Graduate Admissions and Records Office.)
Evaluation of Ellipsometric Monitors for Process Control of High Temperature Superconductors

A Thesis

Presented for the Master of Science Degree

The University of Tennessee, Knoxville

Kyle John Peterson

May 2000
Acknowledgments

Several people have provided support and assistance on this project for which I am extremely grateful. First, I would like to thank my wife for her endless support, prayers, patience, and love throughout this endeavor. I would also like to thank Dr. Chris Parigger and Dr. Jim Lewis for serving as committee members. Dr. Parigger also provided assistance in computational aspects of this work through numerous discussions and implementation of a Condor high throughput computing loop. I would especially like to thank Dr. Lloyd Davis for serving as my advisor and committee chairman. Throughout this project, Dr. Davis provided support, council and numerous suggestions. I am especially indebted to the LI-COR and Sverdrup corporations for supporting me during this work through graduate research assistantships. Lastly, I give thanks and praise to God for giving me the will and ability to complete this thesis.
Abstract

Current methods of manufacturing high temperature thin film superconductors require increasingly more precise and accurate thin film diagnostics to ensure high quality superconductors. There are several diagnostic tools for monitoring thin films. However, due to the complex nature of superconducting thin film stacks, ellipsometric methods are well suited for this application. This study investigates various ellipsometric methods in simulated process control environments.

A computer program was developed to generate ellipsometric data for given film stack characteristics and experimental geometry, and to analyze simulated ellipsometric data to recover optical parameters of the given film stack. The program was used to investigate the effects of systematic errors in the angle of incidence, one of the primary sources of systematic error in a process control environment. It is found that for the given expected film characteristics, the experimental geometry can be selected to minimize the effects of angle of incidence error.
4.3 Angle of Incidence Error ........................................... 58

5 Summary and Suggestions for Future Work .......................... 81
  5.1 Ellipcalc Computer Code ........................................ 82

Bibliography .................................................................. 84

Appendix ...................................................................... 95

A Parameter Tables ....................................................... 96

B Ellipcalc ..................................................................... 97

C Input/Output Examples ................................................ 141

Vita ........................................................................... 145
List of Tables


Subscripts refer to the principle axes a, b, and c respectively. ........ 96

A.2 Drude-type dispersion parameters for YBCO. ................. 96
List of Figures

2.1 PCSA nulling ellipsometer arrangement. .......................... 10

3.1 S and p polarization states incident upon a surface. ........... 20

3.2 PCSA rotating analyzer configuration. ............................ 23

3.3 Ambient/isotropic thin film/isotropic substrate film stack model. . . . 29

3.4 Reflections in a multilayer film stack. ............................ 33

4.1 Ψ/∆ trajectory for YBCO with a 100Å buffer layer of SrTiO$_3$ on a Ni substrate for various angles of incidence. .................. 42

4.2 Ψ/∆ trajectory for SrTiO$_3$ on a Ni substrate for various angles of incidence. 43

4.3 Ψ/∆ trajectory for a uniaxial YBCO model with a 100Å buffer layer of SrTiO$_3$ on a Ni substrate for various incident angles. .... 44

4.4 Ψ/∆ trajectory for YBCO with a 100Å buffer layer of SrTiO$_3$ on a Ni substrate for various wavelengths. .......................... 45

4.5 Ψ vs. thickness for various angles of incidence for the YBa$_2$Cu$_3$O$_7$/SrTiO$_3$/Ni model. .................................................. 46
4.6 Ψ vs. thickness for various wavelengths for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model. ................................................................. 47

4.7 Dependence of ellipsometric parameter Ψ on incident angle and film thickness at a wavelength of 632.8 nm for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model. 48

4.8 Dependence of ellipsometric parameter Ψ on wavelength and film thickness at an incident angle of 60 degrees for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model. ................................................................. 49

4.9 Dependence of the ellipsometric parameter ∆ on film thickness for the uniaxial $YBa_2Cu_3O_7/SrTiO_3/Ni$ model at 632.8 nm for various incident angles. ................................................................. 51

4.10 Dependence of ellipsometric parameter ∆ on incident angle and film thickness at a wavelength of 632.8 nm for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model. ................................................................. 52

4.11 Dependence of the ellipsometric parameter ∆ on film thickness for the uniaxial $YBa_2Cu_3O_7/SrTiO_3/Ni$ model at an incident angle of 60 degrees for various wavelengths. ................................................................. 53

4.12 Dependence of ellipsometric parameter ∆ on incident angle and film thickness at an incident angle of 60 degrees for various wavelengths for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model. ................................................................. 54

4.13 Dependence of the ellipsometric parameter Ψ on film thickness for the $SrTiO_3/Ni$ model at 632.8 nm for various incident angles. .......................... 55
4.14 Dependence of the ellipsometric angle Ψ on film thickness and angle of incidence for the \( \text{SrTiO}_3/Ni \) model at 632.8 nm. .................. 56

4.15 Dependence of the ellipsometric parameter Δ on film thickness for the \( \text{SrTiO}_3/Ni \) model at 632.8 nm for various incident angles. ............ 57

4.16 Dependence of the ellipsometric angle Δ on film thickness and angle of incidence for the \( \text{SrTiO}_3/Ni \) model at 632.8 nm. .................. 58

4.17 Effect of angle of incidence error on the computed ellipsometric parameter Ψ at 632.8 nm for various angles of incidence. .................. 59

4.18 Effect of angle of incidence error on the computed ellipsometric angle Δ at 632.8 nm for various angles of incidence. .................. 60

4.19 Effect of angle of incidence error on the computed ellipsometric angle Ψ for various wavelengths at 60 degrees angle of incidence. ............ 61

4.20 Effect of angle of incidence error on the computed ellipsometric angle Δ for various wavelengths at 65 degrees angle of incidence. ............ 62

4.21 Effect of angle of incidence error on solution of film thickness at 632.8 nm for various angles of incidence. .................. 64

4.22 Effect of angle of incidence error on solution of film thickness for an angle of incidence of 60 degrees for various wavelengths. ............ 65

4.23 Effect of angle of incidence error on simplex-minimized solution of one dielectric tensor component at 632.8 nm for various angles of incidence. 66
4.24 Effect of angle of incidence error on simplex minimized solution of one dielectric tensor component at an angle of incidence of 60 degrees for various wavelengths. ........................................... 67

4.25 Effect of angle of incidence error on the error function when solving for thickness of film and one dielectric tensor component at 632.8 nm for various angles of incidence. ........................................... 68

4.26 Effect of angle of incidence error on the error function when solving for thickness of film and one dielectric tensor component at an incident angle of 60 degrees for various wavelengths. ........................................... 69

4.27 Effect of angle of incidence error on solution of one dielectric tensor component of a spectroscopic ellipsometer. ........................................... 71

4.28 Effect of angle of incidence error on solution of one dielectric tensor component of a multiple angle ellipsometer for various wavelengths. ..... 72

4.29 Effect of angle of incidence error on solution of film thickness of a spectroscopic ellipsometer. ........................................... 73

4.30 Effect of angle of incidence error on solution of film thickness of a multiple angle ellipsometer. ........................................... 74

4.31 Effect of angle of incidence error on the error function solving for thickness of film and one dielectric tensor component with a spectroscopic ellipsometer. ........................................... 75
4.32 Effect of angle of incidence error on the error function solving for thickness of film and one dielectric tensor component with a multiple angle ellipsometer. .......................................................... 76

4.33 Effect of angle of incidence error on solution of film thickness for a multiple angle spectroscopic ellipsometer. .......................................................... 77

4.34 Effect of angle of incidence error on solution of one dielectric tensor component for a multiple angle spectroscopic ellipsometer. ......................... 79
Chapter 1

Introduction

Until the discovery of high-temperature superconductors (HTSCs) in 1986 by Alex Müller and George Bednorz, superconductors were thought to have limited commercial use, largely because of the enormous refrigeration costs [1]. Now, HTSCs are beginning to make the transition from the laboratory to industrial applications.

In order for a HTSC to be practical, it must have a high critical current even in strong magnetic fields and be flexible enough to be wound into wires or coils [2]. It is believed that weak Josephson links limit critical current density in bulk high temperature superconductors and also make the material brittle and unworkable [3]. One solution to this problem is to deposit the superconductor crystal on a textured substrate. For the purposes of this paper, superconductors deposited in this manner will be referred to as a superconducting thin film. Critical currents exceeding one million amperes per square centimeter have been reported using $YBa_2Cu_3O_{7-x}$ (YBCO) deposited on a textured
substrate at a critical temperature of 77K [2].

As various manufacturing processes are being developed, it has become clear that diagnostics and control are essential for producing high quality HTSCs. The YBCO superconductors are biaxial crystals that yield high critical current densities only if the individual crystalline exhibit a high degree of alignment in the c-axis [4]. Hence, care must be taken in the manufacturing process to ensure good alignment of the YBCO crystals. It has also been shown that the thickness of individual layers of a superconducting thin film stack affects the quality of the superconductor [5]. Therefore, in order to produce high quality superconductors, it is also required that the film thickness of each layer in a film stack be monitored throughout the manufacturing process.

1.1 Superconducting Thin Film Stacks

In order to choose an appropriate method for the monitoring of film thicknesses in the superconductor manufacturing environment, a general understanding of the composition of a typical superconducting film stack is necessary.

The fabrication of superconducting film stacks is an evolutionary process. New materials and thin film combinations are being researched everyday. Hence, the monitoring of the thickness of each layer of a film stack must allow for different configurations. For example, simple films have superconducting crystals deposited on some form of textured substrate. More complicated films can include one or more buffer layers between the superconducting crystal and substrate. Buffer layers of lanthanum aluminate (LaAlO₃),
cerium oxide ($\text{CeO}_2$), barium zirconate ($\text{BaZrO}_3$), strontium titanium oxide ($\text{SrTiO}_3$), and yttria stabilized zirconia (YSZ) have been used on metal substrates \cite{2}. Superconducting film stacks can also have a protective layer on the top of the superconducting crystal. Unprotected YBCO thin films can interact with $\text{CO}_2$ in humid air to produce an overlayer of $\text{BaCO}_3$ \cite{6,7}.

HTSC crystals such as YBCO are biaxial crystals. This complicates optical methods for the determination of film thickness, because optical properties are anisotropic and vary with the polarization and angle of incidence of the radiation.

Ideally, a manufactured thin film of YBCO would consist of all crystals oriented in exactly the same direction. Unfortunately, even the best superconducting films contain increasing amounts of random crystal orientation as the thickness is increased \cite{4}. Because of this, the dielectric function of YBCO can vary considerably as a function of film thickness. Even in simple dielectric thin films, the complex index of refraction will vary slightly with film thickness. Therefore accurate determination of thin film thickness will only be achieved if the complex index of refraction, or complex dielectric tensor in the case of anisotropic materials, is also measured.

Since substrates generally used for superconducting tapes are metals with high coefficients of extinction, we will assume all substrates to be optically thick. Therefore only optical techniques based on reflection will be considered.
1.2 Overview

A brief review of several methods that can be used to monitor thickness in HTSC thin films is given in Chapter 2. At the conclusion of Chapter 2, a summary of the best methods for monitoring film thickness in a process control environment is presented.

Several computer simulations were conducted to investigate the performance of ellipsometric methods. The results of these simulations are given in Chapter 4. Since angle of incidence error is a primary source of systematic error in a process control environment, the primary focus of the simulations is to examine the effects of angle of incidence error on ellipsometric measurements. These simulations were completed using a new computer code written in C++ and listed in Appendix B. The equations used in the computer code are provided in a theoretical discussion of ellipsometry in Chapter 3. Finally, several suggestions for future work and an overall summary of the results obtained in Chapter 4 are given in Chapter 5.1.
Chapter 2

Thin Film Monitors

This chapter gives a brief review of several techniques that can be used to determine thin film thickness on an optically thick substrate. Concluding this overview, a brief summary is given as to the recommended methods for monitoring film thickness in a superconductor manufacturing environment.

2.1 Interference Methods

Interference methods measure interference fringe displacement to determine film thickness. Measurements are nondestructive, fast, and are capable of the highest accuracy of any optical technique [8]. Before the wide availability of coherent light sources, interferometric methods were rarely used for determining optical parameters of thin films.

There are several different types of optical interference methods that are capable of measuring thin film thickness. The Fizeau method, which uses monochromatic radia-
tion, and the fringes of equal chromatic order (FECO) method, which uses white light, are two common methods of measuring film thickness [9]. These techniques are not well suited for a typical manufacturing environment, because of stable setup requirements, sample preparation requirements, and the usual requirement of a separate method to measure the index of refraction. These methods require the index of refraction to be known \textit{a priori}.

Other interference methods, which are more suited to manufacturing environments and have the ability to measure the index of refraction, include variable angle monochromatic fringe observation (VAMFO) and constant angle reflection interference spectroscopy (CARIS) [10, 11, 12, 13, 14, 15]. In the case of homogeneous isotropic films, these methods have the ability to make multilayer measurements [16]. Automatic interference methods that utilize CCD detection and computerized interference fringe analysis have also been developed [17].

There are several disadvantages of interferometric techniques. One disadvantage is that absorbing films are difficult to measure. Measurement of optical parameters of very weakly absorbing films (coefficient of extinction, $k < 10^3$) has been demonstrated, but to do so requires transparent substrates [18]. Thickness and the indices of refraction of uniaxial films have been determined with the CATIS method [14] but this method also requires a transparent substrate. Despite these disadvantages, modern interferometric techniques are a good choice for simple film stack configurations.
2.2 Reflectance Methods

Reflection spectrophotometry, the most common reflectance method, measures the amplitude of reflected light at normal incidence over a range of wavelengths. The data is then analyzed in terms of the Fresnel equations to determine the film thickness. Films as thin as 30 Angstroms (Å) can be measured with good accuracy relative to other methods, but the best results are attained when films are greater than 200Å. If the film is relatively thick, other optical parameters such as index of refraction can be extracted from the data as well.

Another similar technique is variable-angle reflectance measurement. In contrast to reflection spectrophotometry, measurements are made at various angles at a single wavelength [19]. Multi-angle measurements are usually made simultaneously using a microscope objective rather than rotating the sample as was done in older techniques [19]. Because data is collected at multiple angles simultaneously, greater precision and repeatability than reflection spectrophotometry is achieved. The variable-angle reflectance technique has been expanded further to include spectroscopic measurements, which have been able to resolve the thickness of several layers simultaneously [20].

The primary advantage of reflection spectrophotometry is the low cost due to few optical components. However, this method is limited in certain situations. Film surfaces must be smooth, as rough surfaces will contribute significant error to the measurement. The technique can not accurately measure very thin films (< 30Å) and is generally limited to isotropic films. Intensity fluctuations of the light source must be kept to a
minimum as they can contribute a significant amount of systematic error [8]. However, despite these shortcomings, reflection spectrophotometry is an extremely robust and cost-effective technique.

2.3 Ellipsometric Methods

Ellipsometry was originally developed in the 1800s, but until the advent of computers, the calculations were too difficult and cumbersome for practical use. Now computers can extract the optical parameters of a thin film stack from ellipsometric data with ease. Ellipsometry measures the phase difference and ratio of magnitudes of the complex reflection coefficients between the p and s polarization states. More precisely, ellipsometry is a measurement of the complex ratio of the complex reflection coefficients for p and s polarization states. This is traditionally denoted by $\rho$ and can be represented by

$$\rho = \frac{R_p}{R_s} = \tan \Psi \exp(i\Delta),$$

(2.1)

where $\Psi$ is the measured ratio of magnitudes of the complex reflection coefficients for p and s polarization states, and $\Delta$ is the measured phase difference between p and s waves. After the ellipsometric angles ($\Psi$ and $\Delta$) are measured, the optical parameters of the sample can be determined from a model of the film stack. For complicated film stack models, ellipsometric parameters must be measured at multiple angles of incidence, multiple wavelengths, etc., to determine the sample’s optical parameters.
Just as with interference and reflectance methods, ellipsometry is a non-destructive optical technique. However, ellipsometry is more precise in optical parameter determination and less sensitive to surface irregularities than interference or reflectance methods. Ellipsometry is the most sensitive technique for accurately measuring film thicknesses as thin as one Angstrom. It can measure multiple layers of both transparent and absorbing thin films, and it can also simultaneously measure the complex index of refraction of a film. Ellipsometry is the only technique that can fully determine an anisotropic dielectric function at a particular wavelength [21]. This is of critical importance, because superconducting cuprates such as YBCO are strongly anisotropic.

2.3.1 Nulling Ellipsometry

Nulling or absolute ellipsometry is the most simple and basic ellipsometric technique. Here, angular measurements are made by measuring a null signal. The optical configuration would consist of a monochromatic light source, polarizer, compensator, analyzer, and photodetector. Light passing through the polarizer and compensator becomes elliptically polarized. The elliptically polarized light reflects from the sample and becomes linearly polarized. The analyzer is then rotated such that the reflected light is extinguished and the photodetector measures a null signal.

Figure 2.1 shows a typical nulling ellipsometer configuration. Usually the radiation source consists of a linearly polarized laser, which becomes elliptically polarized as it passes through a compensator. The light is then incident upon the sample at an oblique angle. Upon reflection, the radiation becomes linearly polarized. A photodetector is
then used to measure the reflected radiation while the azimuthal angle of the analyzer is oriented so as to extinguish the reflected radiation and produce a null signal. Once the null signal has been produced, the azimuthal orientation of the analyzer is recorded. Finally, Jones calculus and Fresnel equations can be used to measure the ellipsometric angles.

Although several automated nulling ellipsometry methods have been developed, the technique is slow and cumbersome, making it impractical for in situ measurements [22]. Shot noise may be significant because minimum light levels are being measured. One solution is to use higher intensity light sources. Nulling ellipsometry is also highly susceptible to noise if the light is not in a pure state of polarization.

Nulling ellipsometry does have some advantages over other ellipsometric techniques. As mentioned previously, it is the simplest ellipsometric technique to implement and the
ellipsometric equations are simplified, due in part to the null signal condition. Nulling ellipsometry is also capable of the highest accuracy of the ellipsometric techniques.

2.3.2 Photometric Ellipsometry

Unlike nulling ellipsometry in which the azimuthal angles of the optical components are adjusted for measurement of the ellipsometric angles, photometric ellipsometers measure the variation of detected light flux. Most photometric ellipsometers contain one or more optical components that modulate the polarization. This type of ellipsometers are called dynamic photometric ellipsometers. Dynamic ellipsometer systems offer substantially greater precision than nulling ellipsometer systems because most of the light is transmitted onto the photodetector so that the technique becomes limited by shot noise rather than detector noise as in nulling systems [23].

Multiple Angle Ellipsometry

Multiple Angle Ellipsometry (MAE) is a static photometric technique that is a logical extension to conventional nulling or photometric ellipsometry. MAE measures the ellipsometric angles over two or more incident angles. Although MAE can implement any ellipsometric technique, usually a configuration similar to the nulling configuration in Figure 2.1 is used. Frequent changes of the angle of incidence is undesirable for fast measurements, especially in a process control environment. One solution is to use two separate ellipsometers at different angles of incidence. A much better solution has recently been developed in which a microscope objective is used to simultaneously
measure multiple angles of incidence [24].

**Rotating Analyzer Ellipsometry**

One type of a dynamic photometric ellipsometry is called rotating analyzer ellipsometry (RAE). In this arrangement, all of the azimuthal settings of the optical components are fixed except the analyzer, which undergoes a constant rotation. The detected signal, which contains an AC component on a DC background, is then Fourier analyzed. The primary disadvantage of this technique is the mechanical complexity primarily due to the rotating element. Precision optimization of rotating analyzer ellipsometers has been studied by D.E. Aspnes and A. A. Studna [22, 23]. Systematic errors induced by component imperfections have also been studied [25]. Despite the mechanical complexity of this method, RAE provides an accurate and reliable technique to monitor film thickness in a process control environment [26].

**Phase Modulated Ellipsometry**

Another commonly used dynamic photometric technique, called phase modulated ellipsometry (PME), was developed by Jasperson and Schatterly in 1969 [27]. In this technique, a polarization modulation is induced by a strained birefringent crystal. By application of a controlled modulated high voltage to a piezoelectric, the birefringence of a amorphous silica crystal can be carefully controlled and thus the polarization modulated. Phase modulation ellipsometry is a much faster technique than rotating analyzer ellipsometry and it has the advantage of no moving components. The main
disadvantage of this technique is that the fluctuations in the modulation amplitude are another source of systematic error. This systematic error, as well as other systematic errors induced by component imperfections, have been thoroughly investigated elsewhere [28, 29, 30, 31, 32].

**Spectroscopic Ellipsometry**

Spectroscopic ellipsometry is the measurement of ellipsometric angles as a function of the wavelength of incident radiation. Implementation of the nulling technique here would be extremely tedious. RAE or PME are the two commonly used techniques in spectroscopic ellipsometry. In both of these techniques, a broad band radiation source is used and ellipsometric data is collected as a function of wavelength with a spectrometer. Phase modulated spectroscopic ellipsometry (PMSE) has the advantage of fast data collection, but rotating analyzer spectroscopic ellipsometry (RASE) is generally used because the birefringent crystal is wavelength dependent and contributes dispersive error to the measurement.

Spectroscopic ellipsometry has already proven to be an effective tool for monitoring film growth [33] and process control [26] in other applications. Measurements are fast and have a high degree of precision and repeatability. In addition to the capability for measurement of film thickness, spectroscopic ellipsometry can also provide insight about the quality of a thin film superconductor by measuring the pseudodielectric function [34], which is used to characterize the effective dielectric tensor. Spectroscopic ellipsometry has also been used in the laboratory to measure changes in crystal orientation [4].
These advantages make spectroscopic ellipsometry a good choice for the monitoring of film thickness in a superconductor process control environment.

**Multiple Angle Spectroscopic Ellipsometry**

Multiple angle spectroscopic ellipsometry is the extension of spectroscopic ellipsometry to one or more incident angles using a combination of the methods mentioned previously. In complicated film stacks, the collection of more ellipsometric data (various wavelengths, various angles) can be vital if an accurate solution for the desired optical parameter is to be obtained.

**Imaging Ellipsometry**

Imaging ellipsometry is another extension of other ellipsometric techniques that can provide a two dimensional surface map of optical parameters \[35, 36, 37, 38\]. Most often the rotating analyzer dynamic photometric technique is employed in this method. Instead of probing a point on the surface as is done in other ellipsometric methods, collimated light is incident on the surface to probe an area on the surface. The reflected light from the sample passes through the rotating analyzer and is detected by a CCD camera. The images captured by the CCD camera are fed to a computer which performs a pixel by pixel intensity analysis and generates ellipsometric data for each point. In this way, a complete two dimensional surface mapping can be achieved for a thin film.

A surface map of optical parameters of a thin film stack would be ideal for a production environment. Variations in thickness along the plane of the film could be easily
monitored. The primary disadvantage of this technique is the increased computer power required to perform quick measurements. Several algorithms for processing the data for imaging ellipsometers have been developed and evaluated [39]. The sensitivity and speeds of imaging ellipsometers have also been studied [40].

2.4 Acoustic Methods

The field of picosecond ultrasonics is a relatively new noncontact and nondestructive method of measuring thickness in thin films. In this technique, picosecond laser pulses are focused onto a region of a thin film. Absorption of the incident laser pulse creates a sudden thermal expansion, which produces an ultrasonic wave that propagates through the film. At each interface, some of the sound wave is reflected back towards the surface as an echo. When an echo reaches the surface it changes the reflectivity of the film. Thus the velocity of the sound wave can be measured by reflectance variations on the surface. From the velocity measurement, the thickness of the film can be determined. The method of picosecond ultrasonics is best suited for opaque/metal thin films, but has also been used to measure thickness in transparent thin films [41].

Measurements are fast and have a high degree of precision and repeatability. The technique is also capable of multiple layer thickness measurements. Although the technique is relatively new, commercial devices are currently available and have been employed in the semiconductor industry.
2.5 Other Methods

There are several other methods of measuring thin film thickness that can be used in certain situations. For highly transparent films, the index of refraction and thickness can be determined by using the thin film layer as a light guide, where light is coupled into the layer by a prism \[42\]. This method, called prism coupling, is better suited for thick films because it is generally limited to thickness measurements greater than 2000 Å.

X-ray reflectivity techniques are also commonly used in characterizing multilayer film structures \[43\]. These techniques are similar to the previous mentioned reflectivity techniques, but use is made of the absorption of X-rays \[9\]. X-ray diffraction can be used simultaneously with X-ray reflectivity for characterization of crystal phases, orientation and orientation distribution \[3\]. For films that contain heavy elements, thickness can be determined by the intensity of the material’s fluorescence radiation. As thickness increases, fluorescence from the thin film increases whereas that from the substrate decreases \[9\]. All of these techniques are more suited for \textit{ex situ} measurements and are impractical for process control.

2.6 Summary of Film Monitors

Several methods of monitoring film thickness have been presented. Interferometric methods are not well suited for process control of HTSC thin films because of stable setup requirements, the inability to measure absorbing films, and significant error induced
by surface roughness. While the method of picosecond ultrasonics may be a suitable method, it is expensive compared to other methods and currently more suited for opaque films.

Photometric ellipsometric methods are the recommended choice for thickness measurements on superconducting film stacks for a variety of reasons. Measurements are fast, nondestructive, accurate, and are relatively insensitive to surface roughness compared to other methods. If surface roughness is significant, it can be modeled by using ellipsometric techniques [44]. The types of thin films and various configurations of a thin film stack that can be characterized by ellipsometric methods are not as limited as with other methods. Ellipsometric methods are also the most sensitive techniques for thickness determination in very thin films (< 30Å). It should also be pointed out that ellipsometric methods have the advantage of measuring two independent parameters, whereas reflectance techniques measure just one. Reflectance methods are power measurements while ellipsometric measurements are intensity independent [8]. This makes ellipsometric techniques inheritantly more powerful in determining optical parameters of a thin film.
Chapter 3

Ellipsometry

In this chapter, ellipsometric methods are considered in detail. The purpose of this detailed review is to provide the theoretical basis and equations that are used in the computer model listed in Appendix B. From this computer model, several simulations were conducted. The results produced by the computer model are shown in Chapter 4. As previously mentioned, ellipsometric methods have several advantages over other techniques of monitoring thin film thickness and they are the recommended choice for current needs in superconductor manufacturing processes.

There are three primary components of an ellipsometric system. The first component is the physical measurement of the ellipsometric angles of a thin film stack. There are many different optical configurations and various techniques used in ellipsometers. Therefore the actual ellipsometer must be modeled to determine how the ellipsometric angles are measured. The second component is the theoretical model of the film stack,
from which ellipsometric equations are derived and calculated. The third component is the determination of the optical parameters of thin film stack from the ellipsometric angles. In general the ellipsometric equations cannot be inverted. Thus minimization algorithms are used to vary a set of calculated ellipsometric angles with the measured ellipsometric angles until a nonlinear least squares solution for the optical parameters in a film stack is found.

3.1 Jones Calculus

Jones calculus is typically used to model the ellipsometer. Each component in the optical system of an ellipsometer is represented by a particular Jones matrix. Matrix multiplication is then used to examine the effect on the incident radiation, which is represented by the Jones vector

\[
E_i = \begin{pmatrix} E_{ip} \\ E_{is} \end{pmatrix},
\]

(3.1)

where \(E_{ip}\) and \(E_{is}\) are the complex amplitudes projected onto the p and s polarization states, respectively [45, 46], as shown in Figure 3.1.

The ideal polarizer and analyzer, when aligned in the plane of incidence in the p-direction, are represented by [30]
Figure 3.1: S and p polarization states incident upon a surface.

\[
A = P = \begin{pmatrix}
1 & 0 \\
0 & 0
\end{pmatrix},
\]  
(3.2)

Similarly, the ideal phase modulator, when aligned in the p-direction, is represented by

\[
C = \begin{pmatrix}
e^{i\delta} & 0 \\
0 & 1
\end{pmatrix},
\]  
(3.3)

where \(\delta\) is the phase shift induced by the modulation \([27]\).

To obtain the Jones matrix for each of these optical components when aligned at an
azimuthal angle $\theta$ with respect to the P-direction, equations 3.2 and 3.3 are multiplied by the rotation matrix,

$$R(\theta) = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}.$$  

(3.4)

The sample, or film stack, being studied can be represented by the matrix

$$S = \begin{pmatrix} \tilde{R}_p & 0 \\ 0 & \tilde{R}_s \end{pmatrix}$$  

(3.5)

where $\tilde{R}_p = R_pe^{i\delta}$ is ratio of reflected and incident amplitude polarized in the p-direction, $\tilde{R}_s = R_se^{i\delta}$ is the ratio of the reflected and incident amplitude polarized in the s-direction.

### 3.2 Modeling Ellipsometers

#### 3.2.1 Nulling Ellipsometer Model

Consider the polarizer-compensator-sample-analyzer (PCSA) nulling ellipsometer arrangement as shown in Figure 2.1. In this configuration, the compensator is typically set to a fixed azimuthal angle while the azimuthal angles of the polarizer and analyzer are adjusted in order to produce a null signal. Upon multiplication of the appropriate Jones matrices and inverting the resultant equations, the ellipsometric angles $\Psi$ and $\Delta$ can be determined knowing the azimuthal angles and relative phase retardation at a
signal null according to the equation

$$\rho = \tan \Psi \exp(i\Delta) = - \tan A \frac{\tan C + e^{i\delta} \tan(P - C)}{1 - e^{i\delta} \tan C \tan(P - C)}$$  \hspace{1cm} (3.6)$$

where P,C,A are the azimuthal angles of the polarizer, compensator, and analyzer respectively.

### 3.2.2 Rotating Analyzer Ellipsometer Model

This technique has two typical configurations, the polarizer-compensator-sample-analyzer (PCSA) configuration or the polarizer-sample-compensator-analyzer (PSCA) configuration. This technique can also be applied without a compensator, in which case C=0 in equation 3.6. Figure 3.2 shows an example of a PCSA rotating analyzer ellipsometer. In this particular configuration, the polarizer and compensator are set at fixed azimuths, while the analyzer is rotated at a constant angular frequency. The ellipsometric angles are determined by Fourier analyzing the detected signal intensity, which is given by the equation

$$I = I_0 [1 + \alpha \cos 2A(t) + \beta \sin 2A(t)],$$  \hspace{1cm} (3.7)$$

where $\alpha$ and $\beta$ are the normalized Fourier coefficients of the sinusoidal components of the detected intensity signal, $A(t)$. This in turn is given by

$$A(t) = \omega t + A_c,$$  \hspace{1cm} (3.8)$$
where $\omega$ is the angular frequency of the rotating analyzer and $A_c$ is a constant phase offset [44].

If $N$ samples are recorded per analyzer period, let $I_\nu$ be the $\nu$-th data sample ($\nu=1,2,3...N$) of the incident flux. The Fourier coefficients can then be calculated from the from the equations [47]

$$I_0 = \sum_\nu \frac{I_\nu}{N}, \quad (3.9)$$
$$\alpha = \sum_\nu \frac{I_\nu \cos(2\pi\nu)}{\sum_\nu I_\nu/2}, \quad (3.10)$$
$$\beta = \sum_\nu \frac{I_\nu \sin(2\pi\nu)}{\sum_\nu I_\nu/2}. \quad (3.11)$$

Azzam and Bashara [46] showed that equation 3.6 can be written in terms of the
ellipsometric angles by

\[ \rho = \tan \Psi e^{i \Delta} = \frac{1 + \alpha}{\beta \pm i(1 - \alpha^2 - \beta^2)^{1/2}} \tan C + e^{i \delta} \tan(P - C). \]  \tag{3.12}

If we set \( C = 0 \) in equation 3.12, the equation simplifies and can be easily inverted to give \[ 23 \]

\[ \tan \Psi = \left[ \frac{(1 + \alpha)}{1 - \alpha} \right]^{1/2} \tan A, \]  \tag{3.13}

\[ \cos(\Delta - \delta) = \frac{\beta}{(1 - \alpha)^{1/2}}. \]  \tag{3.14}

### 3.2.3 Polarization Modulation Ellipsometric Model

Polarization modulation ellipsometry has three standard configurations – the polarizer-modulator-sample (\( PM_{\pi/4} S \)) configuration, the polarizer-modulator at 45 degrees azimuth-sample-analyzer at 45 degrees azimuth setting (\( PM_{\pi/4} S A_{\pi/4} \)) configuration, and the polarizer at 45 degrees azimuth-modulator-sample-analyzer at 45 degrees azimuth (\( P_{\pi/4} M S A_{\pi/4} \)).

For each of the configurations the detected intensities are \[ 32 \]

**Config: I (\( PM_{\pi/4} S \))**

\[ I = 1 + \rho^2 + (1 - \rho^2) \cos \delta. \]  \tag{3.15a}

**Config: II (\( P_{\pi/4} M S A_{\pi/4} \))**

\[ \]
\[ I = 1 + \rho^2 - 2\rho \sin(\delta_p - \delta_s) \sin \delta + 2\rho \cos(\delta_p - \delta_s) \cos \delta, \quad (3.15b) \]

Config: III \((PM_{\pi}SA_{\pi})\)

\[ I = 1 + \rho^2 + (1 - \rho^2) \cos \delta - 2\rho \sin(\delta_p - \delta_s). \quad (3.15c) \]

In these equations, \(\delta\) varies sinusoidally with time according to

\[ \delta = A_0 \sin \omega t, \quad (3.16) \]

where \(A_0\) is the phase amplitude, which is assumed to be proportional to the peak to peak voltage applied to the crystal divided by the wavelength of the incident radiation [27]. The sine and cosine terms can be Fourier analyzed by expanding into the following Bessel-function series:

\[ \sin(\delta) = \sin(A \sin \omega t) = 2 \sum_{k=0}^{\infty} J_{2k+1}(A) \sin[(2k + 1)\omega t], \quad (3.17a) \]

\[ \cos(\delta) = \cos(A \sin \omega t) = J_0(A) + 2 \sum_{k=1}^{\infty} J_{2k}(A) \cos[2k\omega t]. \quad (3.17b) \]

Measurement involves the isolation of the fundamental \((\omega)\) and second-harmonic \((2\omega)\) components of the detected signal. If we define the ratio of the AC components to the DC component of the signal as \(R_\omega = I_\omega/I_0\) and \(R_{2\omega} = I_{2\omega}/I_0\) we have
Config: I

\[ R_{2\omega} = [2J_2(A)] \frac{1 - \rho^2}{(1 + \rho^2)} \delta, \]  
(3.18a)

Config: II

\[ R_\omega = \sin \Delta [2J_1(A)] \frac{-2\rho}{(1 + \rho^2)} \delta \]  
(3.18b)
\[ R_{2\omega} = \cos \Delta [2J_2(A)] \frac{2\rho}{(1 + \rho^2)} \delta, \]  
(3.18c)

Config: III

\[ R_{2\omega} = 2J_2(A) \frac{1 - \rho^2}{(1 + \rho^2)} \delta, \]  
(3.18d)
\[ R_\omega = \sin \Delta [2J_1(A)] \frac{-2\rho}{(1 + \rho^2)} \delta. \]  
(3.18e)

A simple calibration procedure, which is unique to each configuration, is usually performed to eliminate calculation of the Bessel functions. Details of the calibration procedures can be found elsewhere \cite{30, 32, 46}. After calibration, the ellipsometric angles can be determined by

Config: I

\[ \frac{R_{2\omega}}{R_{2\omega}^{\text{cal}}} = \frac{(1 - \rho^2)}{(1 + \rho^2)} = \cos(2\psi), \]  
(3.19a)

Config: II

\[ \frac{R_\omega}{R_\omega^{\text{cal}}} = \frac{2\rho \sin(\delta_p - \delta_s)}{(1 + \rho^2)} = \sin(2\psi) \sin(\delta_p - \delta_s), \]  
(3.19b)

26
\[
\frac{R_{2\omega}}{R_{2\omega}^{\text{cal}}} = \frac{2\rho \cos(\delta_p - \delta_s)}{1 + \rho^2} = \sin(2\psi) \cos(\delta_p - \delta_s), \quad (3.19c)
\]

Config: III

\[
\frac{R_\omega}{R_\omega^{\text{cal}}} = \frac{2\rho \sin(\delta_p - \delta_s)}{1 + \rho^2} = \sin(2\psi) \sin(\delta_p - \delta_s), \quad (3.19d)
\]

\[
\frac{R_{2\omega}}{R_{2\omega}^{\text{cal}}} = \frac{(1 - \rho^2)}{1 + \rho^2} = \cos(2\psi). \quad (3.19e)
\]

### 3.2.4 Extension to Other Ellipsometric Models

As mentioned in an earlier chapter, spectroscopic ellipsometry employs either the rotating analyzer or polarization modulation techniques with a broadband radiation source. Therefore the same models are used with the only difference being that ellipsometric angles are measured at multiple wavelengths. Variable angle of incidence models utilize either the nulling model or any photometric ellipsometric model at one wavelength. Likewise, a multiple angle spectroscopic model uses any of the dynamic photometric ellipsometry models at multiple wavelengths and angles of incidence. The imaging ellipsometer is modeled by extending the rotating analyzer model to measurements in a X-Y planar array, which will provide a 2D surface mapping of ellipsometric angles.
3.3 Generating Ellipsometric Data

3.3.1 Single Layer Isotropic Film

Given a specific model of a film stack and the associated optical parameters, ellipsometric data can be readily generated. Consider a simple model of a substrate covered by a thin film in an ambient medium. Assume that the thin film and substrate are perfectly homogeneous and optically isotropic. As shown in Figure 3.3, light is incident onto the thin film at an angle $\theta_0$, and is refracted as it traverses into the film. The reflected light suffers a phase shift due to optical path length differences. This phase shift is known as the film phase thickness. In general, it is a complex number given by

$$\beta = 2\pi \left( \frac{d_1}{\lambda_0} \right) N_1 \cos \theta_1,$$  \hspace{1cm} (3.20)

where $N_1$ is the complex index of refraction of the film, $d_1$ is the thickness of the thin film, $\lambda_0$ is the wavelength of the light, and $\theta_1$ is the angle of the refracted light within the film. The complex index of refraction of the film can be separated into real and imaginary parts:

$$N_1 = n_1 - ik_1,$$  \hspace{1cm} (3.21)

where $n_1$ is the index of refraction and $k_1$ is the coefficient of extinction of the thin film. If the thin film layer were a dielectric material, the coefficient of extinction would
vanish and the complex index of refraction would be purely real. Using Snell’s law,

$$N_0 \sin \theta_0 = N_1 \sin \theta_1,$$

where the subscripts 0 and 1 represent the ambient and thin film mediums respectively, equation 3.20 can be rewritten as

$$\beta = 2\pi \left( \frac{d_1}{\lambda_0} \right) \left( N_1^2 - N_0^2 \sin^2 \theta_0 \right)^{\frac{1}{2}}.$$

For dielectric materials, the complex index of refraction $N_1$ is purely real. However, it is important to note that in the general case, the index of refraction $N_1$ is not purely real. Consequently, the angle $\theta_1$ will be complex and no longer has the simple physical meaning of angle of refraction [44]. Therefore, from here on, it will be referred to as the
complex angle of refraction.

To solve for the total reflection coefficients of this film stack model, we must sum all of the multiple reflections the light undergoes as it traverses through the film. The total reflected amplitude for this model is given by the infinite geometric series [46]

\[ R = r_{01} + (1 - r_{01})^2 r_{12} e^{-i2\beta} + (1 - r_{01})^2 r_{10}^2 r_{12} e^{-i4\beta} + ..., \]  

(3.24)

where \( r_{01} \) and \( r_{12} \) are the complex Fresnel amplitude reflection coefficients for the air/film interface and film/substrate interface respectively, and \( \beta \) equals the film phase thickness. Upon summation of the series, the total reflection coefficient for the system becomes

\[ \tilde{R} = \frac{r_{01} + r_{12} e^{-i2\beta}}{1 + r_{01} r_{12} e^{-i2\beta}}. \]  

(3.25)

Thus far we have not accounted for the p and s polarization states. In ellipsometry we must treat p and s polarization states independently, thus equation 3.25 must be written as

\[ \tilde{R}_p = \frac{r_{01p} + r_{12p} e^{-i2\beta}}{1 + r_{01p} r_{12p} e^{-i2\beta}}, \]  

(3.26a)

\[ \tilde{R}_s = \frac{r_{01s} + r_{12s} e^{-i2\beta}}{1 + r_{01s} r_{12s} e^{-i2\beta}}, \]  

(3.26b)

where the p and s subscripts represent the p and s polarization states respectively.

The Fresnel reflection coefficients are determined by the following well-known equa-
\begin{align*}
    r_{01p} & = \frac{N_1 \cos \theta_0 - N_0 \cos \theta_1}{N_1 \cos \theta_0 + N_0 \cos \theta_1}, \quad (3.27a) \\
    r_{01s} & = \frac{N_0 \cos \theta_0 - N_1 \cos \theta_1}{N_0 \cos \theta_0 + N_1 \cos \theta_1}, \quad (3.27b) \\
    r_{12p} & = \frac{N_2 \cos \theta_1 - N_1 \cos \theta_2}{N_2 \cos \theta_1 + N_1 \cos \theta_2}, \quad (3.27c) \\
    r_{12s} & = \frac{N_1 \cos \theta_1 - N_2 \cos \theta_2}{N_1 \cos \theta_1 + N_2 \cos \theta_2}, \quad (3.27d)
\end{align*}

where the subscripts 0, 1, and 2 denote the air, film, and substrate mediums respectively. As with the angle of refraction, it is important to note that in the general case, all values in the Fresnel equations are complex.

When solving for the ellipsometric parameters, it is convenient to write the total reflection coefficients in the complex phase notation

\begin{align*}
    \tilde{R}_p & = |R_p| e^{i\delta_p}, \quad (3.28a) \\
    \tilde{R}_s & = |R_s| e^{i\delta_s}, \quad (3.28b)
\end{align*}

where $|R_p|$ and $|R_s|$ represent the amplitude attenuation, and $\delta_p$ and $\delta_s$ represent the phase shifts in the p and s polarization states respectively [46].
The ellipsometric angles are then easily derived by the following equations:

\[ \rho_r = \frac{R_p}{R_s}, \]  
\[ \tan \psi_r = \frac{|R_p|}{|R_s|}, \]  
\[ \Delta_r = \delta_p - \delta_s. \]

Upon substitution of equations 3.26a and 3.26b into equation 3.29, the ellipsometric equation can be written more explicitly as

\[ \rho = \tan \psi e^{i\delta} = \left( \frac{r_{01} + r_{12} e^{-i2\beta}}{1 + r_{01} r_{12} e^{-i2\beta}} \right) \left( \frac{1 + r_{01} r_{12} e^{-i2\beta}}{r_{01} + r_{12} e^{-i2\beta}} \right). \]

### 3.3.2 Multi-Layered Isotropic Film Stacks

If the model is further generalized to multi-layered film stacks as shown in Figure 3.4, the amplitude reflection coefficients of equations 3.26a and 3.26b must be modified. Summation of all of the partial waves in the film stack would lead to an infinite series of an infinite series. However, under the approximation that multiple reflections are confined to individual film layers, the reflectivity coefficients can be determined recursively.
Figure 3.4: Reflections in a multilayer film stack.

by the following equations [20]

\[
R_p^n = \frac{r_p^{n,n+1} + R_p^{n+1}e^{-i2\beta}}{1 + r_p^{n,n+1}R_p^{n+1}e^{-i2\beta}}, \tag{33a}
\]

\[
R_s^n = \frac{r_s^{n,n+1} + R_s^{n+1}e^{-i2\beta}}{1 + r_s^{n,n+1}R_s^{n+1}e^{-i2\beta}}, \tag{33b}
\]

where the superscript \( n \) denotes the \( n \)-th layer, and \( r_{p,s}^{n,n+1} \) are the Fresnel coefficients between the \( n \) and \( n+1 \) layer, obtained by generalization of equations 3.27a, 3.27b, 3.27c, and 3.27d.
3.3.3 Anisotropic Films

Thus far, the model considered is valid only for optically isotropic films. The study of anisotropic films with ellipsometry has received considerable attention over the years [48, 49, 50, 51, 52, 53, 54]. The most general way to derive the appropriate ellipsometric equations is to use a 4x4 matrix formulism developed to study reflection, transmission and refraction in anisotropic crystals [55, 46]. The 4x4 matrix formulism is derived from a 6x6 matrix representation of Maxwell’s equations that includes Faraday rotation and optical activity [56]. Another less complicated approach is to modify the Fresnel equations [54].

For anisotropic media the phase thickness $\beta$ is no longer the same for both p and s polarization states. For biaxial anisotropic films the phase thickness becomes [46]

$$\beta_p = 2\pi \frac{d_1}{\lambda_0} \frac{N_{1x}}{N_{1z}} (N_{1z}^2 - N_0^2 \sin^2 \theta_0)^{1/2}, \quad (3.34a)$$

$$\beta_s = 2\pi \frac{d_1}{\lambda_0} (N_{1y}^2 - N_0^2 \sin^2 \theta_0)^{1/2}, \quad (3.34b)$$

where $N_{1x}$, $N_{1y}$, $N_{1z}$ are the complex indices of refraction for the anisotropic film in the principal axes x, y, and z respectively. For uniaxial films with the crystal axis aligned parallel or perpendicular to the film interface, $N_{1y}=N_{1z}$ or $N_{1x}=N_{1z}$ respectively. If $N_{1y}=N_{1z}=N_{1x}$, equations 3.34a and 3.34b reduce to equation 3.23, the phase thickness for isotropic thin films.
The total reflection coefficient for anisotropic crystals becomes a second ranked tensor which can be represented by

$$ R = \begin{pmatrix} R_{pp} & R_{ps} \\ R_{sp} & R_{ss} \end{pmatrix}. \quad (3.35) $$

If two of the three principal axis of the crystal are aligned parallel to the thin film interface and our optical axes is in the xz plane of the film, the reflection matrix becomes diagonal so that $R_{sp} = R_{ps} = 0 \,[46]$. The reflection coefficients are then given by

$$ \tilde{R}_p = \frac{r_{01p} + r_{12p}e^{-i2\beta_p}}{1 + r_{01p}r_{12p}e^{-i2\beta_p}}, \quad (3.36a) $$

$$ \tilde{R}_s = \frac{r_{01s} + r_{12s}e^{-i2\beta_s}}{1 + r_{01s}r_{12s}e^{-i2\beta_s}}, \quad (3.36b) $$

where $\beta_s$ is the phase thickness in the s polarization state and $\beta_p$ is the phase thickness in the p polarization state.

For a biaxial anisotropic film on an isotropic substrate in an ambient medium the Fresnel reflection coefficients can be shown to be $[54, 46]$

$$ r_{01pp} = \frac{N_{1x}N_{1z} \cos \theta_0 - N_0(N_{1z}^2 - N_0^2 \sin^2 \theta_0)^{1/2}}{N_{1x}N_{1z} \cos \theta_0 + N_0(N_{1z}^2 - N_0^2 \sin^2 \theta_0)^{1/2}}, \quad (3.37a) $$

$$ r_{01ss} = \frac{N_0 \cos \theta_0 - (N_{1y}^2 - N_0^2 \sin^2 \theta_0)^{1/2}}{N_0 \cos \theta_0 + (N_{1y}^2 - N_0^2 \sin^2 \theta_0)^{1/2}}, \quad (3.37b) $$
Again, the equations for uniaxial films can be determined by letting \( N_{1y} = N_{1z} \) or \( N_{1x} = -N_{1z} \). The equations can further be reduced to the isotropic case if \( N_{1x} = N_{1y} = N_{1z} \).

In the case of a biaxial anisotropic film on a biaxial anisotropic substrate, the Fresnel equations for the ambient/film interface are the same as equations 3.37a and 3.37b. The Fresnel equations for the film/substrate interface have been determined from an extension of a uniaxial model given in reference [46] to a biaxial model, and are given by

\[
 r_{12pp} = \frac{-N_{1x}N_{1z} \cos \theta_0 + N_2(N_1^2 - N_2^2 \sin^2 \theta_2)^{1/2}}{N_1 \cdot N_1 \cos \theta_0 + N_2(N_1^2 - N_2^2 \sin^2 \theta_2)^{1/2}},
\]

(3.37c)

\[
 r_{12ss} = \frac{-N_2 \cos \theta_2 - (N_2^2 - N_2^2 \sin^2 \theta_2)^{1/2}}{N_2 \cos \theta_2 + (N_2^2 - N_2^2 \sin^2 \theta_2)^{1/2}}.
\]

(3.37d)

where \( \theta_1e \) and \( \theta_2e \) are given by

\[
 \tan \theta_1e = (N_0 \cdot N_{1z}/N_{1x})(N_1^2 - N_0^2 \sin^2 \theta_0)^{-1/2},
\]

(3.40)

and

\[
 \tan \theta_2e = (N_0 \cdot N_{2z}/N_{2x})(N_2^2 - N_0^2 \sin^2 \theta_0)^{-1/2}.
\]

(3.41)
3.4 Parameter Determination

Only in a few specific cases can the ellipsometric equations be analytically inverted to solve for the various optical parameters of a thin film stack [57, 58]. In general the system of equations is analytically intractable and optical parameters are determined with nonlinear least squares minimization algorithms. The error function consists of calculated and measured ellipsometric angles and can be represented by

\[ X = \sum_{i=1}^{M} [ (\delta^c_i - \delta^m_i)^2 + (\psi^c_i - \psi^m_i)^2 ] , \]  

(3.42)

where the subscript \(i\) indicates measurements at different angles of incidence, the subscript \(j\) indicates measurements at different wavelengths, and the superscripts \(c\) and \(m\) represent calculated and measured values respectively.

One might assume that a sufficient number of measurements at different wavelengths and angles will completely overdetermine the system of ellipsometric equations. Unfortunately in most film stack models, measurements at different wavelengths and angles do not produce linearly independent systems of equations [46]. Complicating matters
further, several different minima may exist for the error function [59].

Several different minimization techniques have been used to solve for ellipsometric parameters [60, 61, 62, 59, 63, 64]. The minimization algorithm used for the simulations in Chapter 4 was the Nelder and Mead downhill simplex algorithm. It was chosen because of its robustness and because only the fitting function, and not its derivatives must be provided. It was implemented by utilizing the amoeba routine in the *Numerical Recipes in C* libraries [65].
Chapter 4

Computer Simulations

Two common situations occur when using ellipsometry to monitor film thickness in a manufacturing environment. One situation is where *in situ* ellipsometry is used to monitor film growth during various deposition techniques inside a vacuum chamber [33, 32]. The other situation is where thickness of a film stack or tape is monitored in a *reel-to-reel* type of continuous manufacturing scheme [2]. In the latter case, angle of incidence error is the predominant source of systematic error in ellipsometric measurement. The other major source of systematic error in both situations is azimuthal angle error of the optical components, which has been adequately studied by a number of authors [26, 46, 25, 32, 30, 29, 28].

The computer program listed in Appendix B, was developed to investigate the effect of angle of incidence error on different ellipsometric systems. This computer code simulates several ellipsometer arrangements for a broad range of film stack configurations.
and is sufficient to investigate angle of incidence errors for various superconducting thin film stacks.

The computer code consists of two main parts. Given a parameter file that consists of the theoretical model of the film stack to be studied, the program calculates and solves the ellipsometric angles exactly using equations presented in the previous chapter. Ellipsometric angles can be solved for any angle of incidence or wavelength using Cauchy or Drude dispersion models. Data can also be generated in a two dimensional array to simulate imaging ellipsometers. The program can also calculate ellipsometric angles for a series of films of increasing in thickness, which can then be used to plot the trajectories shown in Section 4.1, or the growth plots shown in Section 4.2. The other primary component of the program is the determination of optical parameters from ellipsometric angles. Parameters can be chosen to be either fixed or varied until a least squares solution can be found.

For most of the simulations in this chapter, a $YBa_2Cu_3O_7/SrTiO_3/Ni$ film stack model is used, as some of the best YBCO thin films have been deposited on $SrTiO_3$ [4]. $SrTiO_3$ crystals have a cubic perovskite structure and optical properties are therefore isotropic [8]. Sheriff et al. have shown maximum current densities of YBCO occur when film thickness is in the 800-1000Å region [5]. Thus the simulations will focus on this thickness region for YBCO thin films. All of the film stack parameters used in the simulations are listed in the tables in Appendix A.
4.1 Psi/Del Trajectories

Before the advent of computers, trajectories of the ellipsometric angles $\Psi$ and $\Delta$ were used to determine the thickness of a thin film [47]. Figure 4.1 shows the computed $\Psi/\Delta$ trajectory for a thin film of $YBa_2Cu_3O_{7-x}$ deposited on a 100Å layer thickness of $SrTiO_3$ on a $Ni$ substrate for different incident angles using the parameters listed in Table A.1. Dielectric tensor values of $YBa_2Cu_3O_{7-x}$ used for calculations were those obtained by Kircher et al\cite{66, 67}. Values for $SrTiO_3$ and $Ni$ were obtained from reference \cite{68}. From this trajectory, the growth of a $YBa_2Cu_3O_{7-x}$ thin film can be analyzed. At zero thickness, the $\Psi$ trajectory starts at $42^\circ$ while the $\Delta$ trajectory starts at $170^\circ$. The thickness of the $YBa_2Cu_3O_{7-x}$ layer is increased in 10Å increments. The trajectories start to spiral into values characteristic of bulk $YBa_2Cu_3O_{7-x}$. This spiral is a characteristic of the coefficient of extinction. As the coefficient of extinction increases, the trajectory will spiral in faster.

The growth of a thin layer of the buffer material $SrTiO_3$ on a $Ni$ substrate is shown in Figure 4.2. In the visible part of the spectrum, $SrTiO_3$ is transparent and has a very low coefficient of extinction. Therefore the trajectories of $\Psi$ and $\Delta$ show more periodicity and spiral to the bulk material value much more slowly.

Anisotropy can also be studied using $\Psi/\Delta$ trajectories. Due to the small size of $YBa_2Cu_3O_{7-x}$ crystals and films, it is impractical to use ellipsometric methods to directly measure the c-axis of the dielectric function, which is aligned normal to the surface \cite{69}. However, since $YBa_2Cu_3O_7$ crystals exhibit heavy a-b plane twinning, a
Figure 4.1: $\Psi/\Delta$ calculated at 632.8 nm for a thin film of $YBa_2Cu_3O_7$ deposited on a 100Å layer of $SrTiO_3$ on a $Ni$ substrate.
Figure 4.2: Ψ/Δ calculated at 632.8 nm for a thin film of SrTiO₃ deposited on a substrate of Ni for various angles of incidence.

uniaxial model can be assumed [66, 21].

Figure 4.3 shows the same trajectory as in Figure 4.1, except that the YBa₂Cu₃O₇₋ₓ thin film layer is calculated with uniaxial anisotropic crystal ellipsometric equations. Notice that all features of the trajectory are the same in this figure as in Figure 4.1. The only visible difference between the figures is that the trajectory has shifted up and to the right. This effect is consistent with observed anisotropy effects on other thin films [44]. From here on simulations will be made under the approximation that YBa₂Cu₃O₇₋ₓ is a uniaxial crystal.
Figure 4.3: Calculated $\Psi/\Delta$ trajectories at 632.8 nm for a thin film of $YBa_2Cu_3O_7$ deposited on a 100Å layer of $SrTiO_3$ on a $Ni$ substrate assuming a uniaxial model.

When a Drude-type dispersion relation is applied to the model, the trajectory for various wavelengths can be computed. The parameters used for the YBCO Drude dispersion calculations are listed in Table A.2. Figure 4.4 shows the result of the computed trajectories for various wavelengths.
Figure 4.4: $\Psi/\Delta$ calculated at an incident angle of 60 degrees for a thin film of $YBa_2Cu_3O_7$ deposited on a 100Å layer of $SrTiO_3$ on a $Ni$ substrate.

4.2 Growth Plots

4.2.1 $YBa_2Cu_3O_{7-x}$ Growth Plots

In Figures 4.5 and 4.6, the ellipsometric angle $\Psi$ is plotted independently as a function of thickness for various angles and wavelengths respectively. The same data is also plotted in Figures 4.7 and 4.8 three-dimensionally. From these figures it can seen that the value of $\Psi$ becomes nearly constant when the film thickness of the $YBa_2Cu_3O_{7-x}$ layer is greater than 3500Å. In this region there is significant absorption of the incident radiation in the $YBa_2Cu_3O_{7-x}$ layer and optical thickness measurements are impossible.
Figure 4.5: Dependence of the ellipsometric parameter $\Psi$ on film thickness for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model at 632.8 nm for various incident angles.
Figure 4.6: Dependence of the ellipsometric parameter $\Psi$ on film thickness for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model at an incident angle of 65 degrees.
Figure 4.7: Dependence of ellipsometric parameter $\Psi$ on incident angle and film thickness at a wavelength of 632.8 nm for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model.
Figure 4.8: Dependence of ellipsometric parameter $\Psi$ on wavelength and film thickness at an incident angle of 60 degrees for the $YBa_2Cu_3O_{7-x}/SrTiO_3/Ni$ model.

This also corresponds to the point at which ellipsometric measurements on the film stack are characteristic of bulk $YBa_2Cu_3O_{7-x}$.

If a particular range of thickness of $YBa_2Cu_3O_{7-x}$ is to be measured, either Figure 4.5 or 4.8 can be used to determine an appropriate incident angle for any fixed angle ellipsometer. For thickness below 1000Å, a low incident angle is preferred to obtain good convergence of the ellipsometric angle $\Psi$ when solving for optical parameters, whereas for thickness greater than 1000Å, higher incident angles are preferred. For a fixed wavelength ellipsometer, an appropriate wavelength can be chosen with the help
of Figure 4.6. Long wavelengths are better for thicker YBCO films whereas shorter wavelengths are preferred for thinner YBCO films.

In the same way that appropriate wavelengths and angles could be chosen for the ellipsometric angle Ψ, figures 4.9, 4.10, 4.11, and 4.12 can be used to determine the appropriate wavelengths and angles for the ellipsometric parameter Δ respectively. Notice in Figure 4.9 that for films thicker than 1000Å, larger incident angles are preferred. However, also note that an incident angle of 60 degrees and not 70 degrees is optimum for thicknesses greater than 1000Å. In fact, the incident angle of 70 degrees is quite poor for these films. It is also interesting to note that the curves do not flatten out until film thickness reaches approximately 4000Å indicating that a greater optical thickness can be determined from Δ than from Ψ.

Using Figure 4.11, the optimum wavelength for the ellipsometric parameter Δ can be determined. Just as long large incident angles should be used for a thick $YBa_2Cu_3O_7$ layer, long wavelengths are preferred for thick films and shorter wavelengths for films less than 1000Å. Note that for thick films the optimum wavelength is around 700 nm. The same data is also plotted in Figure 4.12 three-dimensionally.
Figure 4.9: Dependence of the ellipsometric parameter $\Delta$ on film thickness for the uniaxial $YBa_2Cu_3O_7/SrTiO_3/Ni$ model at 632.8 nm for various incident angles.
Figure 4.10: Dependence of ellipsometric parameter $\Delta$ on incident angle and film thickness at a wavelength of 632.8 nm for the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model.
Figure 4.11: Dependence of the ellipsometric parameter $\Delta$ on film thickness for the uniaxial $YBa_2Cu_3O_7/SrTiO_3/Ni$ model at an incident angle of 60 degrees for various wavelengths.
Figure 4.12: Dependence of ellipsometric parameter $\Delta$ on incident angle and film thickness at an incident angle of 60 degrees for various wavelengths for the $YBa_2Cu_3O_7$/SrTiO$_3$/Ni model.

4.2.2 $SrTiO_3$ Growth Plots

Now consider the epitaxial growth of the buffer layer $SrTiO_3$ on the $Ni$ substrate. In Figures 4.13 and 4.14, the ellipsometric parameter $\Psi$ is plotted as a function of thickness. Notice the strong periodicity of $\Psi$ due to the low coefficient of extinction in the visible spectrum. One interesting feature of this figure is that for low angles of incidence, the peaks becomes larger as thickness increases. For high angles of incidence the opposite is true.
Figure 4.13: Dependence of the ellipsometric parameter $\Psi$ on film thickness for the $SrTiO_3/Ni$ model at 632.8 nm for various incident angles.
Figure 4.14: Dependence of the ellipsometric angle $\Psi$ on film thickness and angle of incidence for the $SrTiO_3/Ni$ model at 632.8 nm.

Figures 4.15 and 4.16 show the ellipsometric parameter $\Delta$ as a function of thickness. As with $\Psi$, the strong periodicity of the function is easily seen. It can also be seen that $\Delta$ oscillates with a bigger amplitude for larger incident angles. This would improve minimization convergence and produce more accurate measurements.
Figure 4.15: Dependence of the ellipsometric parameter $\Delta$ on film thickness for the $SrTiO_3/Ni$ model at 632.8 nm for various incident angles.
4.3 Angle of Incidence Error

As previously mentioned, angle of incidence error is the primary source of systematic error for optical methods in process control. Several simulations were completed to examine the effects of this error on various parameters. First, simulations for the uniaxial $YBa_2Cu_3O_7/SrTiO_3/Ni$ model were performed to compare calculated ellipsometric angles with ellipsometric angles that contained angle of incidence error. Figures 4.17 and 4.18 show the errors induced in the ellipsometric angles $\Psi$ and $\Delta$ versus angle of incidence error, respectively, for a nulling or fixed wavelength, fixed angle ellipsometer.
Figure 4.17: Effect of angle of incidence error on the computed ellipsometric parameter \( \Psi \) at 632.8 nm for various angles of incidence.
Figure 4.18: Effect of angle of incidence error on the computed ellipsometric angle $\Delta$ at 632.8 nm for various angles of incidence.

Errors induced in $\Psi$ and $\Delta$ vary approximately linearly with angle of incidence error except for high angles of incidence. One important feature in Figures 4.17 and 4.18 is that low angles of incidence yield significantly less error than high angles of incidence in the calculated values for both $\Psi$ and $\Delta$.

In Figures 4.19 and 4.20, the results for the same simulation are shown but plotted for various wavelengths at an incident angle of 65 degrees. These graphs help determine the optimum wavelength for a nulling ellipsometer with this film stack model. Error induced on $\Psi$ is relatively small. Positive angle of incidence, however, does produce
Figure 4.19: Effect of angle of incidence error on the computed ellipsometric angle $\Psi$ for various wavelengths at 60 degrees angle of incidence.
Figure 4.20: Effect of angle of incidence error on the computed ellipsometric angle $\Delta$ for various wavelengths at 65 degrees angle of incidence.

more error than negative angle of incidence error. As can be seen in Figure 4.20, wavelength has a significant effect for angle of incidence error on computed values of $\Delta$. A wavelength choice of 500 nm produces substantially less error than other choices of wavelengths.

The induced error in the ellipsometric angles due to angle of incidence error is of little consequence when it is the optical parameters of thin films that are to be determined. However, they are directly related to each other. Depending on the nonlinearity of the ellipsometric equations, a small error in the ellipsometric angles could cause a minimization algorithm to converge to a local minima that is different from the actual
value. It should also be pointed out that the initial guess provided to these algorithms is also an important factor in converging to the actual parameters.

The next series of simulations investigates the effects of angle of incidence error on the determination of the optical parameters with the downhill simplex minimization algorithm [65]. Using the uniaxial $YBa_2Cu_3O_7/SrTiO_3/Ni$ film stack model, ellipsometric angles were calculated for an angle of incidence that excludes the given systematic error. Then, another set of ellipsometric angles were calculated for an angle of incidence that includes the error. Using these results, the downhill simplex algorithm was utilized to solve for the thickness and index of refraction in the a-axis of the $YBa_2Cu_3O_7$ film while keeping all other film stack parameters fixed at their actual values.

The angle of incidence error versus the determined film thickness for a simulated nulling or fixed angle, fixed wavelength ellipsometer is plotted in Figures 4.21 and 4.22. The horizontal line in the center of the figure represents the actual film thickness. As expected, for no angle of incidence error, all incident angles produce the correct value for film thickness. As the angle of incidence is increased, error in the determined value of film thickness increases. At seventy degrees incidence, a two degree error in angle of incidence will produce an approximately 50Å error in film thickness, whereas at a thirty degree angle of incidence, it would produce only a 10Å error. In Figure 4.22, it is evident that wavelengths of 500-600 nm produce substantially less error than other wavelengths. At 600 nm, a two degree error in angle of incidence will produce an approximately 20Å error in film thickness, whereas at a wavelength of 800nm, it would produce 75Å error.
Figure 4.21: Effect of angle of incidence error on solution of film thickness at 632.8 nm for various angles of incidence.
Figure 4.22: Effect of angle of incidence error on solution of film thickness for an angle of incidence of 60 degrees for various wavelengths.

Another interesting feature is the presence of the vertical line for a wavelength 700 nm. When angle of incidence error is greater than -0.2 degrees the algorithm converges to an extremely low value for film thickness.

In Figure 4.23, the effect of angle of incidence error on the determined a-axis index of refraction is plotted for various angles of incidence. From this plot it can be seen that generally low angles of incidence produce the greatest amount or error in determined index of refraction. An angle of incidence of between sixty and seventy degrees appears to be near the optimum.

Figure 4.24 shows the effect of angle of incidence error on the determined a-axis
Figure 4.23: Effect of angle of incidence error on simplex-minimized solution of one dielectric tensor component at 632.8 nm for various angles of incidence.
Figure 4.24: Effect of angle of incidence error on simplex minimized solution of one dielectric tensor component at an angle of incidence of 60 degrees for various wavelengths. Index of refraction for various wavelengths. A wavelength of 600 nm produces the least amount of error in this figure. Note that for a wavelength of 700 nm, the algorithm converged to huge values when error in the incident angles was greater than -0.4 degrees.

Figures 4.25 and 4.26 are plots of the error function when solving for the thickness and one dielectric tensor component of YBCO. The error function provides a measure of the accuracy of the determined solutions. For this simulation, the error function is nearly zero for all errors in angles of incidence. The minimization algorithm has high confidence that determined parameters are correct regardless of angle of incidence error. However, as seen previously in Figures 4.21 and 4.23, there is significant error in
Figure 4.25: Effect of angle of incidence error on the error function when solving for thickness of film and one dielectric tensor component at 632.8 nm for various angles of incidence.
Figure 4.26: Effect of angle of incidence error on the error function when solving for thickness of film and one dielectric tensor component at an incident angle of 60 degrees for various wavelengths.

determined parameters for angle of incidence error.

Spectroscopic and Multiple Angle Simulations

The next series of figures presents simulation results for the same film stack model but for multiple angle ellipsometers and spectroscopic ellipsometers. For the multiple angle simulations, ten angles of incidence are calculated (30 to 75 degrees in 5 degree increments), while twenty wavelengths were calculated for the spectroscopic simulations (400 to 875 nm in 25 nm increments). When minimization is performed in the simulations,
all of the parameters are minimized globally over all wavelengths in the spectroscopic simulations, and over all incident angles for the multiple angle simulations.

The effect of angle of incidence error on the determined value of the real part of the dielectric tensor of YBCO in the a-axis is shown in Figure 4.27 for the case of a spectroscopic ellipsometer, and in Figure 4.28 for the case of a multiple angle ellipsometer. As before, the horizontal line in the center on the graph represents the actual value of the real part of the dielectric tensor. In the case of the simulated spectroscopic ellipsometer, angles between fifty and sixty degrees appear to be the optimum angle of incidence. For the multiple angle ellipsometer, a wavelength of 600 nm yields the smallest amount of error in determined index of refraction when error in angle of incidence is present. A poor choice in wavelength can lead to substantial error in index of refraction determination. From Figures 4.27 and 4.28, it can be seen that index of refraction determination in spectroscopic ellipsometry is less susceptible to error than multiple angle ellipsometry.

Error in film thickness that results from angle of incidence error for spectroscopic ellipsometers and multiple angle ellipsometers is shown in Figures 4.29 and 4.30, respectively. For a spectroscopic ellipsometer, an angle of incidence of sixty degrees produces the least error in determined film thickness. Low angles of incidence should be avoided, as error in determined thickness is sufficiently greater for angle of incidence error. For the multiple angle ellipsometer, wavelengths between 500 and 600 nm are the best choice for minimizing error in film thickness.
Figure 4.27: Effect of angle of incidence error on solution of one dielectric tensor component of a spectroscopic ellipsometer.
Figure 4.28: Effect of angle of incidence error on solution of one dielectric tensor component of a multiple angle ellipsometer for various wavelengths.
Figure 4.29: Effect of angle of incidence error on solution of film thickness of a spectroscopic ellipsometer.
Figure 4.30: Effect of angle of incidence error on solution of film thickness of a multiple angle ellipsometer.

Figures 4.31 and 4.32 examine the effects of angle of incidence error on the minimization error function. Note that the values for the error function are much higher than those in the nulling or fixed angle, fixed wavelength ellipsometers in Figures 4.26 and 4.25. This is a result of the global minimization that occurs in multiple angle and spectroscopic ellipsometric techniques.

Multiple Angle of Incidence Spectroscopic Simulations

Multiple angle of incidence spectroscopic ellipsometers are simulated in the next series of graphs. Film thickness versus angle of incidence error is plotted in Figure 4.33.
Figure 4.31: Effect of angle of incidence error on the error function solving for thickness of film and one dielectric tensor component with a spectroscopic ellipsometer.
Figure 4.32: Effect of angle of incidence error on the error function solving for thickness of film and one dielectric tensor component with a multiple angle ellipsometer.
Figure 4.33: Effect of angle of incidence error on solution of film thickness for a multiple angle spectroscopic ellipsometer.
Compared with spectroscopic simulations in Figure 4.29 and multiple angle of incidence simulations in Figure 4.30, the combined multiple angle and spectroscopic method is much less susceptible to angle of incidence error in determining film thickness. Negative angle of incidence error only slightly affects thickness values. Negative three degrees in angle of incidence error only affects the determined thickness by two or three Angstroms. However, positive angle of incidence error does have a slightly greater affect on film thickness. The determined thickness is in error by approximately 45 Å when a positive three degree angle of incidence error is present.

Figure 4.34 shows the effect of angle on incidence error on the determined value of the real part of the YBCO dielectric tensor in the a-axis. This graph shows far less error in the determined dielectric tensor element than spectroscopic and multiple angle of incidence simulations in Figures 4.29 and 4.30, respectively.

The error function versus angle of incidence is plotted in Figure 4.35. The error function increases rapidly as angle of incidence error is increased. Examining figures 4.25 and 4.31, it is evident that as more data is used in global minimization, the error function becomes a better indicator of the amount of angle of incidence error that is present in a measurement.
Figure 4.34: Effect of angle of incidence error on solution of one dielectric tensor component for a multiple angle spectroscopic ellipsometer.
Figure 4.35: Effect of angle of incidence error on the error function for a multiple angle spectroscopic ellipsometer.
Chapter 5

Summary and Suggestions for Future Work

From the evaluation discussed in Chapter 2, photometric ellipsometry is the recommended method of measuring and monitoring thin film thickness in a superconductor process control environment. Using the computer code listed in Appendix B, simulations have shown several ellipsometric methods to be effective techniques to monitor film thickness and other optical parameters of superconducting thin film stacks in manufacturing environments.

Several simulations were completed to determine effects of angle of incidence error on thickness and index of refraction on a $YBa_2Cu_3O_7/SrTiO_3/Ni$ film stack. Angle of incidence error was not found to be a factor in affecting the minimization algorithm to converge to unreasonable values for thickness. It was also found that error in determined
values of thickness and index of refraction can be made relatively insensitive to angle of incidence error if an appropriate wavelength and angle of incidence is chosen for the target film characteristics.

Multiple angle spectroscopic ellipsometry produces the smallest error in film thickness determination when angle of incidence error is present. However, an appropriate choice of incidence angle in spectroscopic ellipsometry and appropriate choice of wavelength in multiple angle ellipsometry can also be nearly as good in minimizing the effect of angle of incidence error. Even in the simple nulling or basic photometric ellipsometric technique, an appropriate choice of wavelength and incidence angle can significantly reduce error in determined thickness caused by angle of incidence error.

5.1 Ellipcalc Computer Code

One way that computer code could be significantly improved is to add random error to the ellipsometric data sets. This would allow for studies of random experimental error, which was not considered in any of the simulations presented. Random experimental error could be implemented into the computer code by the addition of error to ellipsometric angles using a Monte Carlo Gaussian distributed random number generator before minimization routines are executed.

The computer code could be extended in a variety of ways. Systematic error correction could be incorporated, as in Bermudez’s ellips program [31]. The imaging ellipsometer routines in the code could make analysis of real experimental data easier
if routines were added to extract data from acquired CCD images. Better and faster minimization routines could be implemented to provide faster optical parameter determination. An algorithm could also be developed to implement minimization routines in a more effective and efficient way. With the algorithm used in this source code and the $YBa_2Cu_3O_7/SrTiO_3/Ni$ film stack model, only two parameters could be varied and have good convergence to the actual solutions. Depending on the initial starting points given to the minimization algorithm, three parameters could be solved accurately with varying amounts of success. Convergence to the actual results when four parameters were varied could not be obtained with this model. The minimization algorithm developed by Frank K. Urban would also help in accurate convergence for absorbing films \[64\]. Anisotropic routines could be further generalized in the 4x4 matrix technique as described by \[53, 56, 70\]. Algorithms for special film stack cases could also be included that allow analytic/semianalytic inversion of the ellipsometric equations.
Bibliography
Bibliography


Appendix
Appendix A

Parameter Tables

Table A.1: Parameters for simulations using the $YBa_2Cu_3O_7/SrTiO_3/Ni$ model. Subscripts refer to the principle axes a,b, and c respectively.

<table>
<thead>
<tr>
<th>Substrate</th>
<th>$SrTiO_3$</th>
<th>$YBa_2Cu_3O_7$</th>
<th>ambient</th>
</tr>
</thead>
<tbody>
<tr>
<td>thickness</td>
<td>n/a</td>
<td>100Å</td>
<td>varied</td>
</tr>
<tr>
<td>$n_a$</td>
<td>1.97</td>
<td>2.437</td>
<td>1.739</td>
</tr>
<tr>
<td>$k_a$</td>
<td>3.72</td>
<td>0.02</td>
<td>0.4727</td>
</tr>
<tr>
<td>$n_b$</td>
<td>1.97</td>
<td>2.437</td>
<td>1.39788</td>
</tr>
<tr>
<td>$k_b$</td>
<td>3.72</td>
<td>0.02</td>
<td>0.6893</td>
</tr>
<tr>
<td>$n_c$</td>
<td>1.97</td>
<td>2.437</td>
<td>2.196</td>
</tr>
<tr>
<td>$k_c$</td>
<td>3.72</td>
<td>0.02</td>
<td>0.569</td>
</tr>
</tbody>
</table>

Table A.2: Drude-type dispersion parameters for YBCO.

<table>
<thead>
<tr>
<th>Axis</th>
<th>$\epsilon_\infty$</th>
<th>$\hbar\omega_p(eV)$</th>
<th>$\hbar\gamma(eV)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3.53 + i1.51</td>
<td>1.71</td>
<td>0.36</td>
</tr>
<tr>
<td>b</td>
<td>2.79 + i1.70</td>
<td>2.28</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Table A.2: Drude-type dispersion parameters for YBCO.
Appendix B

Ellipcalc

1 // //////////////////////////////////////////////////////////////////////
2 // Name: ellipcalc.cc (v0.95b)
3 // Date: 3/11/2000
4 // Author: Kyle Peterson
5 // Desc: Ellipsometric data generator and Optical Parameter solver
6 // for multi-layer film stack
7 // from ellipsometric data. Parameters allow for any
8 // number of thin film stack models.
9 // This program also can solve for imaged ellipsometer
10 // arrays.
11 // Notes: Requires "Numerical Recipes in C" libraries!
12 // Specifically (amoeba.o amotry.o nrutil.o)
13 // License restrictions prevent me from including these
14 // except in compiled executable.
15 //
16 // Compilation: Designed for Linux kernel 2.2 using glibc 2.1 libraries
17 // Should compile on any standard ANSI C++ system
18 //
19 // Example: g++ ellipcalc.cc -oellipcalc -g amotry.o amoeba.o nrutil.o
20 //
21 // //////////////////////////////////////////////////////////////////////
22
23 #include <stdlib.h>
24 #include <math.h>
25 #include <complex>
26 #include <iostream>
27 #include <fstream>
28 #include <streambuf.h>
29
30 #define PI 3.141592654
31 #define MAX_LAYERS 5
/* Initialize Global Variables*/

int wavelengths, angles;
int nnp,mmp;
int vpixels=1;
int hpixels=1;
int verbose=0;
int layers;

int nnflag[MAX_LAYERS];
int nznflag[MAX_LAYERS];
int nnyflag[MAX_LAYERS];
int kkflag[MAX_LAYERS];
int kkzflag[MAX_LAYERS];
int kkyflag[MAX_LAYERS];
int ttflag[MAX_LAYERS];
float thickness[MAX_LAYERS];
int anisoflag[MAX_LAYERS];
int imageflag, dispersionflag;

float (**phicalc);
float (**deltacalc);
float (**phimeasure);
float (**deltameasure);
float (**phidata);
float (**deltadata);
float (**wavedata);
float (**angledata);

float *wavelength;
float *angle;
float **p;
float **guess;

float ftol;
float nn,kk;

complex<double> N[MAX_LAYERS];
complex<double> NA1[MAX_LAYERS];
complex<double> NA2[MAX_LAYERS];

complex<double> NZ[MAX_LAYERS];
complex<double> NZA1[MAX_LAYERS];
complex<double> NZA2[MAX_LAYERS];

complex<double> NY[MAX_LAYERS];
complex<double> NYA1[MAX_LAYERS];
complex<double> NYA2[MAX_LAYERS];

complex<double> NO[MAX_LAYERS];
complex<double> NZO[MAX_LAYERS];

complex<double> NZO[MAX_LAYERS];
complex<double> NYO[MAX_LAYERS];
complex<double> imaginary(0.0,1.0);

// Declare functions //
int menu(); int ellip_menu();
float err_func(float x[]);
void solvedata(char *argv[]);
void make_parameters(char *argv[]);
void make_datafile(char *argv[]);
void output_phidelta(char *argv[]);
void output_growthcurves(char *argv[]);
void solvedata_ae(char *argv[]);
void get_parameters(char *argv[]);
void reset_parms();
void standard_output(int nfunc);
void get_data(char *argv[]);
void allocate_datamemory();
void phidel_aoie(char *argv[]);
void allocate_matrixmemory();
void amoeba(float **p, float y[], int ndim, float ftol, float (*funk)(float []), int *nfunk);
complex<double> cauchy_dispersion(float lambda, complex<double> z, complex<double> z1, complex<double> z2);
complex<double> drude_dispersion(float lambda, complex<double> z, complex<double> z1, complex<double> z2);
complex<double> CCasin(const complex<double> x);
complex<double> CCatan(const complex<double> z);
float calculate_delta(complex<double> rho);
float calculate_phi(complex<double> rho);
complex<double> calculate_rho(float inc_angle, float lambda);
void null_ellip(int config, float lambda, float inc_angle, float x[]);
void ellip(int config, float lambda, float inc_angle, float x[]);
void elliprot(int config, float lambda, float inc_angle, float x[]);
void make_phideltadata(char *argv[]);
void print_parms(char *argv[]);
void extrap_data(int i, int j);
void csv_parmoutput(int nfunc, int xx, int yy);

main(int argc, char *argv[])
{
    int choice;
    if (argc<=2)
    {
        cout <<" USAGE: imaging [parameterfile] [datafile] {flags} \n";
        return (0);
    }
}
if (argc == 3) choice = menu();
else choice = atoi(argv[3]);

switch (choice)
{
  // Solve Data //
  case 1: solvedata(argv);
          break;

  // Make Parameter File //
  case 2: make_parameters(argv);
          break;

  // Make Ellipsometer Data File //
  case 3: make_datafile(argv);
          break;

  // Output Phi, Delta Calculations //
  case 4: output_phidelta(argv);
          break;

  // Output Phi, Delta Growth Curve //
  case 5: output_growthcurves(argv);
          break;

  // Solve Parameters with Angle of Incidence Error //
  case 6: solvedata_aoie(argv);
          break;

  // Print Parameter File Data and Exit //
  case 7: print_parms(argv);
          break;

  // Make Phi/Delta Data File //
  case 8: make_phideltadata(argv);
          break;
  case 9: phidel_aoie(argv);
          break;

  default: cout << "\n ERROR: Undefined choice or file flag\n"
            return 0;
}

void get_parameters(char *argv[])
{
  int i, l, m;
  mmp=0;
  nnp=0;

  // Additional code...
// Open file Stream //
ifstream infile(argv[1]);

infile >> ftol >> layers >> dispersionflag >> imageflag;

if (imageflag==1)
{
    infile >> hpixels >> vpixels;
}

// Read incident Medium Complex Index of Refraction //
infile >> N[0];

// Read film layers Complex Index of Refraction //
for (i=1; i<=layers; i++)
{
    /* Read flags to determine whether or not parameter is
determined or to be varied */

    infile >> anisoflag[i];
    infile >> nnflag[i];
    if (nnflag[i] == 1) nnp++;
    if (anisoflag[i] >= 1) infile >> nnzflag[i];
    if (nnzflag[i] == 1) nnp++;
    if (anisoflag[i] == 2) infile >> nnyflag[i];
    if (nnyflag[i] == 1) nnp++;
    infile >> kkflag[i];
    if (kkflag[i] == 1) nnp++;
    if (anisoflag[i] >= 1) infile >> kkzflag[i];
    if (kkzflag[i] == 1) nnp++;
    if (anisoflag[i] == 2) infile >> kkyflag[i];
    if (kkyflag[i] == 1) nnp++;

    infile >> ttflag[i];
    if (ttflag[i] == 1) nnp++;
    infile >> N[i];
    NO[i]=N[i];
    if (anisoflag[i] == 1) // i.e. uniaxial
    {
        infile >> NZ[i];
        NZO[i]=N[i];
    }
    if (anisoflag[i] == 2) // i.e. biaxial
    {
        infile >> NZ[i];
        NZO[i]=N[i];
        infile >> NY[i];
    }
NYO[i]=N[i];

if (dispersionflag >=1)
{
    // Read in Dispersion Coefficients //
    infile >> NA1[i];
    infile >> NA2[i];
    if (anisoflag[i] ==1 )
    {
        infile >>NZA1[i];
        infile >>NZA2[i];
    }
    if (anisoflag[i] ==2 )
    {
        infile >>NZA1[i];
        infile >>NZA2[i];
        infile >>NYA1[i];
        infile >>NYA2[i];
    }
    infile >> thickness[i];
}

// Read Substrate Complex Index of Refraction //
infile >> N[layers+1];

// Define Columns for parameter matrix //
mmp=nnp+1;
allocate_matrixmemory();

// Read in Initial Guesses //
for (l=1;l<=mmp;l++)
{
    for (m=1;m<=nnp;m++)
    {
        infile >> guess[l][m];
    }
}
// Close File Stream //
infile.close();
void make_parameters(char *argv[])
{
    int i,l,m;
    nnp=0;
    mmp=0;
    // Open file Stream for writing //
    ofstream infile(argv[1]);
    cout << "Enter Tolerance:";
    cin >> ftol;
    cout << "Enter Number of Layers:";
    cin >> layers;
    cout << "Enter Dispersion Flag:";
    cin >> dispersionflag;
    cout << "Enter Imaging Flag:";
    cin >> imageflag;
    if (imageflag != 1) imageflag=0;
    infile << ftol << "\n" << layers << "\n" <<
    dispersionflag << "\n" << imageflag << "\n";
    if (imageflag == 1)
    {  
        cout << "Enter hpixels:";
        cin >> hpixels;
        cout << "Enter vpixels:";
        cin >> vpixels;
        infile << hpixels << "\n" << vpixels << "\n" ;
    }
    // Read incident Medium Complex Index of Refraction //
    cout << "Enter N[0] (Complex Index of Refraction):";
    cin >> N[0];
    infile << N[0] << "\n" ;
    for (i=1; i<=layers; i++)
    {
        cout << "LAYER[<<i<<"] Anisotropicflag->:";
        cin >> anisoflag[i];
        infile << anisoflag[i] << "\n" ;
        cout << "LAYER[<<i<<"] Solve for n flag->:";
        cin >> nnflag[i];
        infile << nnflag[i] << "\n" ;
        if (nnflag[i] == 1) nnp++;
        if (anisoflag[i] >=1)
{
  cout << "LAYER[<<i<<] Solve for nz flag->:";
  cin >> nnzflag[i];
  infile << nnzflag[i] << "\n" ;
  if (nnzflag[i] == 1) nnp++;
  if (anisoflag[i] ==2) {
    cout << "LAYER[<<i<<] Solve for ny flag->:";
    cin >> nnyflag[i];
    infile << nnyflag[i] << "\n" ;
    if (nnyflag[i] == 1) nnp++;
  } 
  cout << "LAYER[<<i<<] Solve for k flag->:";
  cin >> kkflag[i];
  infile << kkflag[i] << "\n" ;
  if (kkflag[i] == 1) nnp++;
  if (anisoflag[i] >=1) {
    cout << "LAYER[<<i<<] Solve for kz flag->:";
    cin >> kkzflag[i];
    infile << kkzflag[i] << "\n" ;
    if (kkzflag[i] == 1) nnp++;
    if (anisoflag[i] ==2) {
      cout << "LAYER[<<i<<] Solve for ky flag->:";
      cin >> kkyflag[i];
      infile << kkyflag[i] << "\n" ;
      if (kkyflag[i] == 1) nnp++;
    }
  }
  cout << "LAYER[<<i<<] Solve for thickness flag->:";
  cin >> ttflag[i];
  infile << ttflag[i] << "\n" ;
  if (ttflag[i] == 1) nnp++;
  if (anisoflag[i] ==1) nnp=2*nnp;
  cout << "LAYER[<<i<<] Complex Index of Refraction ->:";
  cin >> N[i];
  infile << N[i] << "\n" ;
  if (anisoflag[i] ==1) {
    cout << "LAYER[<<i<<] Complex Index of Refraction (Other Principle axis) ->:";
    cin >> NZ[i];
    infile << NZ[i] << "\n" ;
  }
  if (anisoflag[i] ==2)
{ cout << "LAYER[<<i<<"] Complex Index of Refraction ab plane#1 ->:";
    cin >> NZ[i];
    cout << "LAYER[<<i<<"] Complex Index of Refraction t ab plane#2 ->:";
    cin >> NY[i];
    infile << NZ[i] << "\n" ;
    infile << NY[i] << "\n" ;
}

if (dispersionflag>=1)
{
    // Read in Dispersion Coefficients //
    cout << "LAYER[<<i<<"] Dispersion Coefficient #1 ->:";
    cin >> NA1[i];
    cout << "LAYER[<<i<<"] Dispersion Coefficient #2 ->:";
    cin >> NA2[i];
    infile << NA1[i] << "\n" ;
    infile << NA2[i] << "\n" ;

    if (anisoflag[i] == 1 )
    {
        cout << "LAYER[<<i<<"] Dispersion Coefficient ab plane#1 ->:";
        cin >> NZA1[i];
        cout << "LAYER[<<i<<"] Dispersion Coefficient ab plane#2 ->:";
        cin >> NZA2[i];
        infile << NZA1[i] << "\n" ;
        infile << NZA2[i] << "\n" ;
    }

    if (anisoflag[i] == 2 )
    {
        cout << "LAYER[<<i<<"] Dispersion Coefficient c plane#1 ->:";
        cin >> NYA1[i];
        cout << "LAYER[<<i<<"] Dispersion Coefficient c plane#2 ->:";
        cin >> NYA2[i];
        infile << NYA1[i] << "\n" ;
        infile << NYA2[i] << "\n" ;
    }
}

cout << "LAYER[<<i<<"] Thickness ->:";
    cin >> thickness[i];
    infile << thickness[i] << "\n" ;
}

// Read Substrate Complex Index of Refraction //
    cout << "Substrate Complex Index of Refraction ->:";
    cin >> N[layers+1];
(infile << N[layers+1] << "\n" ;
// Define Columns for parameter matrix //
mmp=nnp+1;
allocate_matrixmemory();
// Read in Initial Guesses //
for (l=1;l<=mmp;l++)
{
    for (m=1;m<=nnp;m++)
    {
        cout << "Parameter Matrix ["<<l<<"]"<<m<<"] Initial Starting guess ->: ";
        cin >> guess[l][m];
        infile << guess[l][m] << "\n" ;
    }
}
// Close File Stream //
infile.close();
}
void allocate_matrixmemory()
{
    int i;
    // Allocate memory for matrices //
p = (float **) malloc((mmp+1) * sizeof(float *));
    for (i=1;i<=mmp;i++)
    {
        p[i]= (float *) malloc((nnp+1) * sizeof(float));
    }
    guess = (float **) malloc((mmp+1) * sizeof(float *));
    for (i=1;i<=mmp;i++)
    {
        guess[i]= (float *) malloc((nnp+1) * sizeof(float));
    }
    if ( guess == NULL || p == NULL ){
        cout << "Memory allocation error.";
        exit(0);
    }
void allocate_datamemory()
{
    int i,k,j;
    // Allocate memory for data matrices //
    wavelength = (float *) malloc((wavelengths+1) * sizeof(float));
    angle = (float *) malloc((angles+1) * sizeof(float));
    phimeasure = (float **)malloc((wavelengths+1) * sizeof(float *));
    for (i=1;i<=wavelengths;i++)
    {
        phimeasure[i]= (float *)malloc((angles+1)*sizeof(float));
    }
    deltameasure = (float **) malloc((wavelengths+1) * sizeof(float *));
    for (i=1;i<=wavelengths;i++)
    {
        deltameasure[i]= (float *)malloc((angles+1)*sizeof(float));
    }
    phicalc = (float **)malloc((wavelengths+1) * sizeof(float *));
    for (i=1;i<=wavelengths;i++)
    {
        phicalc[i]= (float *)malloc((angles+1)*sizeof(float));
    }
    deltacalc = (float **) malloc((wavelengths+1) * sizeof(float *));
    for (i=1;i<=wavelengths;i++)
    {
        deltacalc[i]= (float *)malloc((angles+1)*sizeof(float));
    }
    phidata = (float ****)malloc((hpixels+1) * sizeof(float ***));
    for (i=1;i<=hpixels;i++)
    {
        phidata[i]= (float ****)malloc((vpixels+1) * sizeof(float **));
        for (j=1;j<=vpixels+1;j++)
        {
            phidata[i][j]= (float **)malloc(((wavelengths+1) * sizeof(float *));
            for (k=1;k<=wavelengths;k++)
            {
                phidata[i][j][k]= (float *)malloc((angles+1)*sizeof(float));
            }
        }
    }
    deltadata = (float ****) malloc((hpixels+1) * sizeof(float ***));
    for (i=1;i<=hpixels;i++)
    {

deltadata[i] = (float ***)malloc((vpixels+1)*sizeof(float**));
for (j=1;j<=vpixels+1;j++)
{
    deltadata[i][j] = (float **)malloc((wavelengths+1)*sizeof(float*));
    for (k=1;k<=wavelengths;k++)
    {
        deltadata[i][j][k] = (float *)malloc((angles+1)*sizeof(float));
    }
}

angledata = (float ***)malloc((hpixels+1) * sizeof(float **));
for (i=1;i<=hpixels;i++)
{
    angledata[i] = (float **)malloc((vpixels+1)*sizeof(float*));
    for (j=1;j<=vpixels;j++)
    {
        angledata[i][j] = (float *)malloc((angles+1)*sizeof(float));
    }
}

wavedata = (float ***)malloc((hpixels+1) * sizeof(float **));
for (i=1;i<=hpixels;i++)
{
    wavedata[i] = (float **)malloc((vpixels+1)*sizeof(float*));
    for (j=1;j<=vpixels;j++)
    {
        wavedata[i][j] = (float *)malloc((wavelengths+1)*sizeof(float));
    }
}

if (wavedata == NULL || angledata == NULL || phidata == NULL ||
    deltadata == NULL ){
    cout << "Memory allocation error.";
    exit(0);
}
if (wavelength == NULL || angle == NULL || phimeasure == NULL ||
    deltameasure == NULL || deltacalc == NULL || phicalc == NULL ){
    cout << "Memory allocation error.";
    exit(0);
}

void get_data(char *argv[])
{
    int i,j,m,l;
    ifstream infile(argv[2]);
infile >> wavelengths >> angles;
allocate_datamemory();

for (i=1;i<=hpixels;i++){
  for(j=1;j<=vpixels;j++){
    for (l=1;l<=wavelengths;l++)
      {
        infile >> wavedata[i][j][l];
        for (m=1;m<=angles;m++)
          {
            infile >> angledata[i][j][m];
            infile >> phidata[i][j][l][m] >> deltadata[i][j][l][m];
          }
    }
  }
}

// Close File Stream //
infile.close();

void extrap_data(int i, int j)
{
  int m,l;

  for (l=1;l<=wavelengths;l++)
    {
      wavelength[l] = wavedata[i][j][l];
      for (m=1;m<=angles;m++)
        {
          angle[m] = angledata[i][j][m];
          phimeasure[l][m] = phidata[i][j][l][m];
          // cout << "\n" << phimeasure[l][m];//
          deltameasure[l][m] = deltadata[i][j][l][m];
        }
    }
}

void standard_output(int nfunc)
{
  int i,j;

  float x[MAX_LAYERS*3];

  cout << "\nNumber of evaluations:" << nfunc << "\n";
  for (i=1;i<=nnp;i++) cout << "Parameter\t";
  cout << "ERROR\n";
  for (i=1;i<=mmp;i++)
    {
      for (j=1;j<=nnp;j++)
```c++
{
    cout << p[i][j] << "\t";
    x[j] = p[i][j];
}
cout << err_func(x) << "\n";
}
}
}

void reset_parms()
{
    int l, m;
    for (l=1; l<=mmp; l++)
    {
        for (m=1; m<=nnp; m++)
        {
            p[l][m] = guess[l][m];
            //cout <<"\n" << p[l][m]; //
        }
    }
}

complex<double> cauchy Dispersion(float lambda, complex<double> z, complex<double> z1, complex<double> z2)
{
    complex<double> temp;
    temp = z;
    z = temp + (z1 / (lambda*lambda)) + (z2 / (lambda*lambda*lambda*lambda));
    return z;
}

complex<double> drude Dispersion(float lambda, complex<double> z, complex<double> z1, complex<double> z2)
{
    complex<double> temp;
    float pe;
    //Convert wavelength to photon energy //
    pe = 1.98645e-25 / (lambda * 1.6e-19);
    temp = z;
    z = sqrt(temp - ((z1 * z1) / (pe * (pe + imaginary*z2))));
    temp = z;
    z = real(temp) + (-imaginary*imag(temp));
    return z;
}

float err_func(float x[])
{
    int var=1;
}
```
int neato;
int a,i,j,m;
double chi=0.0;
complex<double> temp;
complex<double> rho;

for(i=1;i<=wavelengths;i++)
{
    var=1;
    for (a=1;a<=layers;a++)
    {
        if (nnflag[a]==1)
        {
            N[a]=x[var]+(imaginary*imag(NO[a]));
            //cout <<"\n"<<N[a];//
            var++;
        }
        if (anisoflag[a] >= 1 && nnzflag[a]==1 )
        {
            NZ[a]=x[var]+(imaginary*imag(NZO[a]));
            var++;
        }
        if (anisoflag[a] == 2 && nnyflag[a]==1)
        {
            NY[a]=x[var]+(imaginary*imag(NYO[a]));
            var++;
        }
        if (kkflag[a]==1)
        {
            N[a]=real(NO[a])+(imaginary*x[var]);
            var++;
        }
        if (anisoflag[a] >= 1 && kkzflag[a]==1 )
        {
            NZ[a]=real(NZO[a])+(imaginary*x[var]);
            var++;
        }
        if (anisoflag[a] == 2 && kkyflag[a]==1)
        {
            NY[a]=real(NYO[a])+(imaginary*x[var]);
            var++;
        }
        if (nnflag[a]==0 && kkflag[a]==0) N[a]=NO[a];
    }
}
if (nnzflag[a]==0 && kkzflag[a]==0) NZ[a]=NZ0[a];
if (nnyflag[a]==0 && kkyflag[a]==0) NY[a]=NYO[a];
//if (nnflag[a]==1 && kkflag[a]==0) N[a]=real(N[a])+imag(NO[a]);
//if (nnzflag[a]==1 && kkzflag[a]==0) NZ[a]=real(NZ[a])+imag(NZO[a]);
//if (nnyflag[a]==1 && kkyflag[a]==0) NY[a]=real(NY[a])+imag(NYO[a]);
//if (nnflag[a]==0 && kkflag[a]==1) N[a]=real(NO[a])+imag(N[a]);
//if (nnzflag[a]==0 && kkzflag[a]==1) NZ[a]=real(NZO[a])+imag(NZ[a]);
//if (nnyflag[a]==0 && kkyflag[a]==1) NY[a]=real(NYO[a])+imag(NY[a]);
if (ttflag[a]==1)
{
    thickness[a]=x[var];
    var++;
}
if (dispersionflag==1)
{
    for ( m=1; m<=layers; m++ )
    {
        N[m]=cauchy_dispersion(wavelength[i],N[m],NA1[m],NA2[m]);
        if ( anisoflag[m] >= 1 ) NZ[m]=cauchy_dispersion(wavelength[i],
            NZ[m],NZA1[m],NZA2[m]);
        if ( anisoflag[m] == 2 ) NY[m]=cauchy_dispersion(wavelength[i],
            NY[m],NYA1[m],NYA2[m]);
    }
}
if (dispersionflag==2)
{
    for ( m=1; m<=layers; m++ )
    {
        //cout <<"\n in->" << N[m];//
        N[m]=drude_dispersion(wavelength[i],N[m],NA1[m],NA2[m]);
        //cout <<"\n out->" << N[m];//
        if ( anisoflag[m] >= 1 ) NZ[m]=drude_dispersion(wavelength[i],
            NZ[m],NZA1[m],NZA2[m]);
        if ( anisoflag[m] == 2 ) NY[m]=drude_dispersion(wavelength[i],
            NY[m],NYA1[m],NYA2[m]);
    }
}
for(j=1;j<=angles;j++)
{
    //cout <<"\n wavelength->" << wavelength[i]<< " angle->"<< //
    //angle[j]+aoie;//
    rho=calculate_rho((angle[j]),wavelength[i]);
    phicalc[i][j] = calculate_phi(rho);
    deltacalc[i][j] = calculate_delta(rho);
if (deltacalc[i][j] < 0) deltacalc[i][j] = deltacalc[i][j] + 360;
if (real(rho) < 0 && abs(imag(rho)) < 1.0e-6) deltacalc[i][j] = 180;

// Calculate error Function //
chi = chi + ((phicalc[i][j] - phimeasure[i][j]) * (phicalc[i][j] - phimeasure[i][j]) +
(phimeasure[i][j] - deltameasure[i][j]) *
(deltacalc[i][j] - deltameasure[i][j]));
//cout << "\n" <<x[1] << "\t" <<phicalc[i][j] << "\t" <<phimeasure[i][j] << "\t" << deltameasure[i][j]; //
//cout << "\n" <<x[1]; //
}

chi = chi / ((wavelengths + 1) * (angles + 1) * 1.0);
//cout << "chi->" << chi; //
//exit(0); //
if (neato < 1 || neato > 8) neato = 1;
else neato++;
switch (neato)
{
    case 1: cout << "/\" << flush; break;
    case 2: cout << "-\" << flush; break;
    case 3: cout << "\\\" << flush; break;
    case 4: cout << "|\" << flush; break;
    case 5: cout << "/\" << flush; break;
    case 6: cout << "-\" << flush; break;
    case 7: cout << "\\\" << flush; break;
    case 8: cout << "|\" << flush; break;
}
return chi;

complex<double> calculate_rho(float inc_angle, float lambda)
{
    int k;
    complex<double> rho;
    complex<double> beta[MAX_LAYERS];
    complex<double> betas[MAX_LAYERS];
    complex<double> betap[MAX_LAYERS];
}
complex<double> theta[MAX_LAYERS];
complex<double> thetad[MAX_LAYERS];
complex<double> arg[MAX_LAYERS];
complex<double> r01p[MAX_LAYERS];
complex<double> r12p[MAX_LAYERS];
complex<double> r01s[MAX_LAYERS];
complex<double> r12s[MAX_LAYERS];
complex<double> X[MAX_LAYERS];
complex<double> Xs[MAX_LAYERS];
complex<double> Xp[MAX_LAYERS];
complex<double> RP[MAX_LAYERS];
complex<double> RS[MAX_LAYERS];

theta[0]=inc_angle*PI/180;

// Substrate & ambient cannot be anisotropic... //
anisoflag[ layers+1]=0;
anisoflag[0]=0;
NY[0]=NZ[0]=N[0];
NY[ layers+1]=NZ[ layers+1]=N[ layers+1];
for( k=1;k<=layers+1;k++ ){
    // Calculate Complex Angles of Refraction //
    arg[k]=(N[k-1]/N[k])*sin(theta[k-1]);
    theta[k]=CCasin(arg[k]);
}

for( k= layers; k>=1;k--){
    if (anisoflag[k] == 0) {
        NZ[k]=NY[k]=N[k];
    } else if (anisoflag[k] == 1) {
        NY[k]=N[k];
    }

    beta[k]=4*PI*(thickness[k]/lambda) * sqrt((N[k]*N[k]-N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1])));
    betas[k] = 4*PI*(thickness[k]/lambda) * sqrt((NY[k] * NY[k]-N[k-1]*N[k-1] * sin(theta[k-1])*sin(theta[k-1])));
    betap[k] = 4*PI*(thickness[k]/lambda) * (N[k]/NZ[k])*sqrt((NZ[k]*NZ[k]-NZ[k-1]*NZ[k-1]*sin(theta[k-1])) *
if (anisoflag[k] >= 1 \&\& anisoflag[k-1] >= 1 \&\& anisoflag[k+1] >= 1)
{
    cout << "\n Sorry, only support for 2 anisotropic layers in a row is currently supported."
    exit(0);
}

if (anisoflag[k] >= 1 \&\& anisoflag[k-1] == 0)
{
    r01p[k] = (N[k]*NZ[k]*cos(theta[k-1]) - N[k-1] * sqrt(NZ[k]*NZ[k] - (N[k]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1]))) / (N[k]*NZ[k]*cos(theta[k-1]) + N[k-1] * sqrt(NZ[k]*NZ[k] - (N[k]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1]))));

    r01s[k] = (N[k-1]*NZ[k]*cos(theta[k-1]) - (sqrt(NY[k]*NY[k] - N[k]*N[k] * sin(theta[k-1]) * sin(theta[k-1])))) / (N[k-1]*NZ[k]*cos(theta[k-1]) + (sqrt(NY[k]*NY[k] - N[k]*N[k] * sin(theta[k-1]) * sin(theta[k-1]))));
}

else if (anisoflag[k] >= 1 \&\& anisoflag[k-1] == 1)
{
    thetae[k] = CCatan( ((N[k-1]*NZ[k]) / N[k]) * sqrt(NZ[k]*NZ[k] - (N[k-1]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1]))));

    thetae[k+1] = CCatan( ((N[k-1]*NZ[k+1]) / N[k+1]) * sqrt(NZ[k+1]*NZ[k+1] - (N[k-1]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1]))));

    r01p[k] = (-N[k]*NZ[k-1]*sin(thetae[k]) * sin(thetae[k]) * sqrt(NZ[k]*NZ[k] - N[k-1]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1])) + (N[k]*NZ[k]*sin(thetae[k]) * sin(thetae[k]) * sqrt(NZ[k]*NZ[k] - N[k-1]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1])))) / (N[k]*NZ[k]*sin(thetae[k]) * sin(thetae[k]) * sqrt(NZ[k]*NZ[k] - N[k-1]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1])) + (N[k]*NZ[k]*sin(thetae[k]) * sin(thetae[k]) * sqrt(NZ[k]*NZ[k] - N[k-1]*N[k-1] * sin(theta[k-1]) * sin(theta[k-1]))));

    r01s[k] = (N[k]*NZ[k]*sin(theta(k)) - N[k-1]*N[k]*sin(theta(k)) * sin(theta(k)) * (sqrt(NY[k]*NY[k] - N[k]*N[k] * sin(theta(k)) * sin(theta(k)))) / (sqrt(NY[k]*NY[k] - N[k]*N[k] * sin(theta(k)) * sin(theta(k))) + (sqrt(NY[k]*NY[k] - N[k+1]*N[k+1] * sin(theta(k+1)) * sin(theta(k+1)))) * (N[k]*N[k] + N[k]*N[k+1] * sin(theta(k+1)) * sin(theta(k+1))));
}
else
{
    r01p[k]=(N[k] * cos(theta[k-1]) - N[k-1] * cos(theta[k])) /
            (N[k] * cos(theta[k-1]) + N[k-1] * cos(theta[k]));
    r01s[k]=(N[k-1] * cos(theta[k-1]) - N[k] * cos(theta[k])) /
            (N[k-1] * cos(theta[k-1]) + N[k] * cos(theta[k]));
}

if (anisoflag[k] >= 1 && anisoflag[k+1] == 0)
{
    r12p[k] = (-N[k] * NZ[k] * cos(theta[k]) + N[k+1] *
                 sqrt(NZ[k] * NZ[k] - (N[k] * N[k] * sin(theta[k]) * sin(theta[k])))) / (N[k] * NZ[k] * cos(theta[k]) + (sqrt(NY[k] * NY[k] - N[k] * N[k] * sin(theta[k]) * sin(theta[k]))));
    r12s[k] = (-N[k+1] * NZ[k+1] * cos(theta[k]) + (sqrt(NY[k] * NY[k] - N[k-1] * N[k-1] * sin(theta[k]) * sin(theta[k])))) / (sqrt(NY[k] * NY[k] - N[k-1] * N[k-1] * sin(theta[k]) * sin(theta[k])));
}

if (anisoflag[k] >= 1 && anisoflag[k+1] == 1)
{
    thetae[k] = CCatan( (N[k-1] * NZ[k]) / N[k] ) * sqrt(NZ[k] * NZ[k] - (N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1])));
    thetakp[k] = CCatan( (N[k] * NZ[k]) / N[k-1] ) * sqrt(NZ[k] * NZ[k] - (N[k] * N[k] * sin(theta[k]) * sin(theta[k])));
    r12p[k] = (N[k] * NZ[k+1] * sin(thetae[k]) * sin(thetae[k]) *
                 sqrt(NZ[k] * NZ[k] - (N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1]))) - (N[k+1] * NZ[k] * sin(thetae[k])- sin(thetae[k])) * sqrt(NZ[k] * NZ[k] - (N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1])))) / (N[k] * NZ[k] * sin(thetae[k]) * sin(thetae[k]) * sqrt(NZ[k] * NZ[k] - (N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1])) + (N[k]+1) * NZ[k+1] * sin(thetae[k])+ sin(thetae[k]) * sqrt(NZ[k] * NZ[k] - (N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1]))));
    r12s[k] = (sqrt(NY[k] * NY[k] - N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1])) - sqrt(NY[k] * NY[k+1] - N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1])) * (sqrt(NY[k] * NY[k] - N[k] * N[k] * sin(theta[k]) * sin(theta[k])) + sqrt(NY[k] * NY[k+1] - N[k] * N[k] * sin(theta[k]) * sin(theta[k])))) / (sqrt(NY[k] * NY[k] - N[k-1] * N[k-1] * sin(theta[k-1]) * sin(theta[k-1])) + sqrt(NY[k] * NY[k+1] - N[k] * N[k] * sin(theta[k]) * sin(theta[k])));}

else
{

}
r12p[k]=(N[k+1]* cos(theta[k]) - N[k]* cos(theta[k+1])) / 
(N[k+1] * cos(theta[k])+N[k]*cos(theta[k+1]));

r12s[k]=(N[k]* cos(theta[k]) - N[k+1]* cos(theta[k+1])) / 
(N[k] * cos(theta[k])+N[k+1]*cos(theta[k+1]));

if (anisoflag[layers] == 0)
{
    RP[layers] = (r01p[layers]+r12p[layers]*X[layers]) / 
    (1+r01p[layers]*r12p[layers]*X[layers]);
    RS[layers] = (r01s[layers]+r12s[layers]*X[layers]) / 
    (1+r01s[layers]*r12s[layers]*X[layers]);

    for(k=layers-1;k>=1;k--){
        RP[k]=(r01p[k]+RP[k+1]*X[k])/(1+r01p[k]*RP[k+1]*X[k]);
        RS[k]=(r01s[k]+RS[k+1]*X[k])/(1+r01s[k]*RS[k+1]*X[k]);
    }
}

if (anisoflag[layers] >= 1)
{
    RP[layers] = (r01p[layers]+r12p[layers]*Xp[layers]) / 
    (1+r01p[layers]*r12p[layers]*Xp[layers]);
    RS[layers] = (r01s[layers]+r12s[layers]*Xs[layers]) / 
    (1+r01s[layers]*r12s[layers]*Xs[layers]);

    for(k=layers-1;k>=1;k--){
        RP[k] = (r01p[k]+RP[k+1]*Xp[k]) / (1+r01p[k]*RP[k+1]*Xp[k]);
        RS[k] = (r01s[k]+RS[k+1]*Xs[k]) / (1+r01s[k]*RS[k+1]*Xs[k]);
    }
}

return (RP[1]/RS[1]);

float calculate_phi(complex<double> rho) 
{

float calculate_delta(complex<double> rho)
{
    return atan2(imag(rho),real(rho))*180/PI;
}

int menu()
{
    int i;
    cout << "\nEllip-Sim ";
    cout << "\n-----------------------------------------------";  
    cout << "\n Select Choice:";    
    cout << "\n 1. Solve Optical Parameters";
    cout << "\n 2. Make a Parameter File";
    cout << "\n 3. Make a Ellipsometer Simulation Datafile";
    cout << "\n 4. Generate Phi, Delta and Output to stdout";
    cout << "\n 5. Generate Phi, Delta Film Growth Curve";
    cout << "\n 6. Solve Optical Parameters with Angle of Inc. Error";
    cout << "\n 7. Display Parameters";
    cout << "\n 8. Make Phi, Delta Raw Datafile";
    cout << "\n 9. Make Phi, Delta Error Datafile\n";
    cout << "\n-----------------------------------------------\n\n";
    cin >> i;
    return i;
}

void solvedata_aoie(char *argv[])
{
    int i,j,l,m,n;
    int nfunc;
    int xx,yy;
    int output;
    float max_error,error_inc;
    float aoie;
    complex<double> rho;
    float y[MAX_LAYERS*3];
    float x[MAX_LAYERS*3];
    cout << "\n 1. Standard Output format ";
    cout << "\n 2. Tab Delimited output to file output ";
    cout << "\n 3. CSV output to stdout";
    cin >> output;
    cout << "\n Enter Max Angle of Incidence Error->";
    cin >> max_error;
cout << "Enter Angle of Incidence Error Increment->";
cin >> error_inc;

// First Determine New measured Ellipsometric angles with Error //
for(aoie=-max_error;aoie<max_error;aoie=aoie+error_inc)
{
    get_parameters(argv);
    get_data(argv);
    cout <<"thickness->"<<thickness[1]<<"t;
    for(xx=1;xx<=hpixels;xx++)
    {
        for(yy=1;yy<=vpixels;yy++)
        {
            extrap_data(xx,yy);
            for(i=1;i<=wavelengths;i++)
            {
                if (dispersionflag==1)
                {
                    for ( m=1; m<=layers; m++ )
                    {
                        if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion
                        (wavelength[i],NZO[m],NZA1[m],NZA2[m]);
                        N[m]=cauchy_dispersion(wavelength[i],NO[m],NA1[m],NA2[m]);
                        if ( anisoflag[m] == 2 )
                        {
                            NZ[m]=cauchy_dispersion(wavelength[i],NZO[m],NZA1[m],
                            NZA2[m]);
                            NY[m]=cauchy_dispersion(wavelength[i],NYO[m],NYA1[m],
                            NYA2[m]);
                        }
                    }
                    if (dispersionflag==2)
                    {
                        for ( m=1; m<=layers; m++ )
                        {
                            //cout << wavelength[i] << NO[m]<<NA1[m]<<NA2[m]; //
                            //cout << "\n->" <<N[m]; //
                            N[m]=drude_dispersion(wavelength[i],NO[m],NA1[m],NA2[m]);
                            //cout << "\nout->" <<N[m]; //
                            if ( anisoflag[m] >= 1 ) NZ[m]=drude_dispersion(wavelength[i],
                            NZO[m],NZA1[m],NZA2[m]);
                            if ( anisoflag[m] == 2 ) NY[m]=drude_dispersion(wavelength[i],
                            NYO[m],NYA1[m],NYA2[m]);
                        }
                    }
                    for(j=1;j<=angles;j++)
                    {

rho = calculate_rho(angle[j]+aoie, wavelength[i]);

// cout << "\nbefore->" "<< phimeasure[i][j] <<\\t" << //
// deltameasure[i][j]; //
phimeasure[i][j] = calculate_phi(rho);
deltameasure[i][j] = calculate_delta(rho);

if (deltameasure[i][j] < 0) deltameasure[i][j] =
deltameasure[i][j]+360;
if (real(rho)<0 && abs(imag(rho))<1.0e-6)
deltameasure[i][j]=180;
// cout << "nafter->" "<< phimeasure[i][j] <<\\t" << //
// deltameasure[i][j]; //
}
reset_parms();

for (i=1;i<=mmp;i++)
{
  for (j=1;j<=nnp;j++)
  {
    x[j]=p[i][j];
  }
  y[i]=err_func(x);
}

// Call Downhill Simplex Least Squares Minimization Algorithm
cout << "n Iterating..." << flush;
amoeba(p,y,nnp,ftol,err_func,&nfunc);
cout << "n Pixel Array [" << xx << "][" << yy << "]\n";
if (output == 2)
{
  ofstream outfile("parmsolveerror",ios::app);
  if (!outfile)
  {
    cout << "Unable to open output for appending.\n";
    exit(0);
  }

  outfile << "n" << aoie << "\t" << xx << "\t" << yy << "\n";
  outfile << "\t" << nfunc << "\t";
  for (i=1;i<=mmp;i++)
  {
    outfile << p[i][i] << "\t";
  }
  outfile << y[1] << "\t";
  outfile.close();
  }
if (output == 3)
{
  cout << "n" << xx << "," << yy << "," << nfunc << ",";
}
for (i=1;i<=mmp;i++)
{
        cout <<p[1][i]<< ",";
}
cout << y[1] << ",";
}
else standard_output(nfunc);
nfunc=0;
}
for ( m=1; m<=layers; m++ )
{
        N[m]=NG[m];
        NZ[m]=NZO[m];
        NY[m]=NYO[m];
}
for(aoie=-max_error;aoie<max_error;aoie=aoie+error_inc)
{     errordatwave << "\n"<< aoie;
}
for (i=1;i<=wavelengths;i++)
{
    if (dispersionflag==1)
    {
        for ( m=1; m<=layers; m++ )
        {
            if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion(wavelength[i], NZO[m],NZA1[m],NZA2[m]);
            N[m]=cauchy_dispersion(wavelength[i],NO[m],NA1[m],NA2[m]);
            if ( anisoflag[m] == 2 )
                {NZ[m]=cauchy_dispersion(wavelength[i],NZO[m],NZA1[m],NZA2[m]);
                    NY[m]=cauchy_dispersion(wavelength[i],NYO[m],NYA1[m],NYA2[m]);
                }
        }
    }
    if (dispersionflag==2)
    {
        for ( m=1; m<=layers; m++ )
        {
            N[m]=drude_dispersion(wavelength[i],NO[m],NA1[m],NA2[m]);
            if ( anisoflag[m] >= 1 ) NZ[m]=drude_dispersion(wavelength[i], NZO[m],NZA1[m],NZA2[m]);
            if ( anisoflag[m] == 2 ) NY[m]=drude_dispersion(wavelength[i], NYO[m],NYA1[m],NYA2[m]);
        }
    }
    rho=calculate_rho(angle[1]+aoie,wavelength[i]);
    phicalc_error = calculate_phi(rho);
    deltagcalc_error = calculate_delta(rho);
    if (deltagcalc_error <0) deltagcalc_error=deltagcalc_error+360;
    if (real(rho)<0 && abs(imag(rho)<1.0e-6)) deltagcalc_one=180;
    errordatwave << "\t" << (phicalc_error-phimeasure[i][1])
    << "\t" << (deltagcalc_error-deltameasure[i][1]);
}
if (dispersionflag==1)
{
    for ( m=1; m<=layers; m++ )
    {
        if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion(wavelength[1], NZO[m],NZA1[m],NZA2[m]);
        N[m]=cauchy_dispersion(wavelength[1],NO[m],NA1[m],NA2[m]);
        if ( anisoflag[m] == 2 )
            {NZ[m]=cauchy_dispersion(wavelength[1],NZO[m],NZA1[m],NZA2[m]);
            NY[m]=cauchy_dispersion(wavelength[1],NYO[m],NYA1[m],NYA2[m]);
        }
    }
NY[m]=cauchy Dispersion(wavelength[1],NYO[m],NYA1[m],NYA2[m]);
}
}
if (dispersionflag==2)
{
    for ( m=1; m<=layers; m++ )
    {
        NY[m]=drude Dispersion(wavelength[1],NYO[m],NYA1[m],NYA2[m]);
        if ( anisoflag[m] >= 1 ) NZ[m]=drude Dispersion(wavelength[1],NZO[m],NZA1[m],NZA2[m]);
        if ( anisoflag[m] == 2 ) NY[m]=drude Dispersion(wavelength[1],NYUO[m],NYA1[m],NYA2[m]);
    }
}
errordatang << "\n"<< aoie ;
for (j=1;j<=angles;j++)
{
    rho=calculate_rho(angle[j]+aoie,wavelength[1]);
    phicalc_error = calculate_phi(rho);
    deltcalc_error = calculate_delta(rho);
    if (deltcalc_error <0) deltcalc_error=deltcalc_error+360;
    if (real(rho)<0 & & abs(imag(rho)<1.0e-6)) deltcalc_one=180;
    errordatang << "\t" << (phicalc_error-phimeasure[1][j])
    << "\t" << (deltcalc_error-deltameasure[1][j]);
}
aoie=0;
errordatang.close();
errordatwave.close();
void solvedata(char *argv[])
{
    int i,j,l,m,n;
    int nfunc;
    int xx,yy;
    int output;
    float y[MAX_LAYERS*3];
    float x[MAX_LAYERS*3];
    cout << "\n 1. Standard Output format ";
    cout << "\n 2. CSV output to file ouput ";
    cout << "\n 3. CSV output to stdout";
cin >> output;
get_parameters(argv);
//cout <<" ok" << flush; //
get_data(argv);
cout <<" ok" << flush;
// Start Solving //
for (xx=1;xx<=hpixels;xx++)
{
    for (yy=1;yy<=vpixels;yy++)
    {
        extrap_data(xx,yy);
        // cout <<" ok" << flush; //
        // Create Parameter Matrix //
        reset_parms();
        // Evaluate Ellipsometric Function //
        for (i=1;i<=mmp;i++)
        {
            for (j=1;j<=nnp;j++)
            {
                x[j]=p[i][j];
            }
            y[i]=err_func(x);
        }
        // Call Downhill Simplex Least Squares Minimization Algorithm
        cout << " Iterating..." << flush;
        amoeba(p,y,nnp,ftol,err_func,&nfunc);
        cout << " Pixel Array [" << xx << "][" << yy << "]n";
        if (output ==2)
        {
            ofstream outfile("output",ios::app);
            if (!outfile)
            {
                cout << "Unable to open output for appending.\n";
                exit(0);
            }
            outfile << "\n" << xx << "," << yy << "," << nfunc << ",";
            for (i=1;i<=nnp;i++)
            {
                outfile << p[i][j] << ",";
            }
            outfile << y[1] << ",";
            outfile.close();
        }
        if (output ==3)
        {
            ofstream outfile("output",ios::app);
            if (!outfile)
            {
                cout << "Unable to open output for appending.\n";
                exit(0);
            }
            outfile << "\n" << xx << "," << yy << "," << nfunc << ",";
            for (i=1;i<=nnp;i++)
            {
                outfile << p[i][j] << ",";
            }
            outfile << y[1] << ",";
            outfile.close();
        }
    }
}
if (output ==3)
cout << "\n" << xx << "," << yy << "," << nfunc << ",";
for (i=1;i<=nnp;i++)
{
    cout <<p[1][i]<< ",";
}
    cout << y[1] << ",";
}
else standard_output(nfunc);
    nfunc=0;
}

void make_datafile(char *argv[])
{
    int choice,config;
    float lambda,inc_angle;
    float y[MAX_LAYERS*3];
    float x[MAX_LAYERS*3];
    wavelengths=1;
    angles=1;
    allocate_datamemory();
    null_ellip(config,lambda,inc_angle,x);
    break;
    case 2: cout << "\n Input Configuration (1,2,3)->";
    cin >> config;
    cout << "\n Wavelength->";
    cin >> lambda;
    cout << "\n Incident Angle->";
    cin >> inc_angle;
    wavelengths=1;
    angles=1;
    allocate_datamemory();
    null_ellip(config,lambda,inc_angle,x);
    break;
allocate_datamemory();
ellip(config,lambda,inc_angle,x);
break;
case 3: cout << "\n Input Configuration (1,2,3)->";
cin >> config;
cout << "\n Wavelength->";
cin >> lambda;
cout << "\n Incident Angle->";
cin >> inc_angle;

wavelengths=1;
angles=1;
allocate_datamemory();
null_ellip(config,lambda,inc_angle,x);
break;
case 4: cout << "\n Input Configuration (1,2,3)->";
cin >> config;
cout << "\n Wavelength->";
cin >> lambda;
cout << "\n Starting Incident Angle->";
cin >> inc_angle;
cout << "\n Number of Angles (5 Degree Increments)";
cin >> angles;

wavelengths =1;
allocate_datamemory();
null_ellip(config,lambda,inc_angle,x);
break;
case 5: cout << "\n Input Configuration (1,2,3)->";
cin >> config;
cout << "\n Starting Wavelength->";
cin >> lambda;
cout << "\n Number of Wavelengths-> (5 nm increments)";
cin >> wavelengths;
cout << "\n Incident Angle->";
cin >> inc_angle;

angles=1;
allocate_datamemory();
ellip(config,lambda,inc_angle,x);
break;
case 6: cout << "\n Input Configuration (1,2,3)->";
cin >> config;
cout << "\n Starting Wavelength->";
cin >> lambda;
cout << "\n Number of Wavelengths (5 nm increments)";
cin >> wavelengths;
cout << "\n Starting Incident Angle->";
cin >> inc_angle;
cout << "Number of Angles (5 Degree Increments)";
cin >> angles;
allocate_datamemory();
ellip(config,lambda,inc_angle,x);
break;
case 9: break;
default: cout << " ERROR: Unknown option ";
exit (0);
}
}
int ellip_menu()
{
    int i;
cout << "Make-Data Menu";
cout << "--------------------------------------------------------";
cout << "Select Choice:"
    cout << "1. Nulling Ellipsometer"
    cout << "2. Polarization Modulation Ellipsometer"
    cout << "3. Rotating Analyzer Ellipsometer"
    cout << "4. Multiple Angle Ellipsometer"
    cout << "5. Spectroscopic Ellipsometer"
    cout << "6. Multiple Angle Spectroscopic Ellipsometer"
    cout << "9. Exit"
    cout << "--------------------------------------------------------\n\n";
cin >> i;
return i;
}
void null_ellip(int config, float lambda, float inc_angle, float x[])
if (dispersionflag==2) {
    for ( m=1; m<=layers; m++ )
        N[m]=drude_dispersion(wavelength[i],N[m],NA1[m],NA2[m]);
    if ( anisoflag[m] == 1 ) NZ[m]=drude_dispersion(wavelength[i],NZ[m],NZA1[m],NZA2[m]);
    if ( anisoflag[m] == 2 ) NY[m]=drude_dispersion(wavelength[i],NY[m],NYA1[m],NYA2[m]);
}

for(j=1;j<=angles;j++)
    angle[j]=inc_angle+(5*j);
    rho=calculate_rh(o(angle[j],wavelength[l]);
    phicalc[l][j] = calculate_phi(rho);
    deltacalc[l][j] = calculate_delta(rho);
    if (deltacalc[l][j] <0) deltacalc[l][j]=deltacalc[l][j]+360;
    if (real(rho)<0 && abs(imag(rho))<1.0e-6) deltacalc[l][j]=180;
    A=atan(abs(rho));
    P=(arg(rho)-PI/2)/2;
    P2=(arg(rho)+PI/2)/2;
    cout << "\n " << A *180/PI << " " << P*180/PI << " " << P2*180/PI << "\n"; void ellip(int config, float lambda, float inc_angle, float x[])
wavelength[i]=lambda+(i*5e-9);

if (dispersionflag=1)
{
    for ( m=1; m<=layers; m++ )
    {
        temp=wavelength[i];
        if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion(wavelength[i],NZ[m], NZA1[m],NZA2[m]);
        N[m]=cauchy_dispersion(wavelength[i],N[m],NA1[m],NA2[m]);
        if ( anisoflag[m] == 2 )
            {
                NZ[m]=cauchy_dispersion(wavelength[i],NZ[m], NZA1[m],NZA2[m]);
                NY[m]=cauchy_dispersion(wavelength[i],NY[m], NYA1[m],NYA2[m]);
            }
    }
}

for(j=1;j<=angles;j++)
{
    angle[j]=inc_angle+(5*j);
    rho=calculate_rho(angle[j],wavelength[i]);
    phicalc[i][j] = calculate_phi(rho);
    deltacalc[i][j] = calculate_delta(rho);
    if (deltacalc[i][j] <0) deltacalc[i][j]=deltacalc[i][j]+360;
    if (real(rho)<0 && abs(imag(rho))<1.0e-6) deltacalc[i][j]=180;
}

if (config == 1)
{
    // P - M - S Configuration (Initial P at 0 Degrees) //
    // P - M - A - S Calibration //
    NN=cos(2*(phicalc[i][j]*PI/180));
    cout <<"\t" << N <<"\n";
}
else if (config == 2)
{
    // P - M - S - A Configuration (Initial P at 45 Degrees) //
    // P - M - C - A - S Calibration //
    C=cos(2*(phicalc[i][j]*PI/180))*cos(deltacalc[i][j]*PI/180);
    S=sin(2*(phicalc[i][j]*PI/180))*sin(deltacalc[i][j]*PI/180);
    cout <<"\t" << S << "\t" << C <<"\n";
}
else if (config == 3)
{

// P - M - S - A Configuration (Initial P at 0 Degrees) //
// P - M - C - A - S Calibration //

NN=cos(2*(phicalc[i][j]*PI/180));
S=sin(2*(phicalc[i][j]*PI/180))*sin(deltacalc[i][j]*PI/180);
cout <<"\t" << S << "\t" << NN <<"\n";

} else
{
cout <<"\nERROR: Configuration input error\n";
exit (0);
}
}

void elliprot(int config, float lambda, float inc_angle, float x[])
{
    int i,m,j;
    float temp,alpha,beta;
    complex<double> rho;

    for(i=1;i<=wavelengths;i++)
    {
        wavelength[i]=lambda+(i*5e-9);

        if (dispersionflag=1)
        {
            for ( m=1; m<=layers; m++ )
            {
                temp=wavelength[i];

                if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion(wavelength[i],NZ[m], NZA1[m],NZA2[m]);
                N[m]=cauchy_dispersion(wavelength[i],N[m],NA1[m],NA2[m]);

                if ( anisoflag[m] == 2 )
                {
                    NZ[m]=cauchy_dispersion(wavelength[i],NZ[m],NZA1[m],NZA2[m]);
                    NY[m]=cauchy_dispersion(wavelength[i],NY[m],NYA1[m],NYA2[m]);
                }
            }

        }

        for(j=1;j<=angles;j++)
        {
            angle[j]=inc_angle+(5*j);

            rho=calculate_rho(angle[j],wavelength[i]);
            phicalc[i][j] = calculate_phi(rho);
            deltalcalc[i][j] = calculate_delta(rho);
if (deltacalc[i][j] < 0) deltacalc[i][j] = deltacalc[i][j] + 360;
if (real(rho) < 0 && abs(imag(rho)) < 1.0e-6) deltacalc[i][j] = 180;

if (config == 1)
{
    // P - M - S Configuration (Initial P at 0 Degrees) //
    // P - M - A - S Calibration //
    alpha = 1 - 2/(1 + atan(phicalc[i][j])*atan(phicalc[i][j]));
    beta = acos(deltacalc[i][j])*sqrt(1-alpha*alpha);
    cout <<"\t" << alpha <<"\t" << beta <<"\n";
}

void make_phideltadata(char *argv[])
{
    int i,j,m,xx,yy;
    float lambda, inc_angle;
    float vary,temp;
    complex<double> rho;
    get_parameters(argv);
    cout << "Enter Wavelengths->";
    cin >> wavelengths;
    cout << "Enter Starting Wavelength->";
    cin >> lambda;
    cout << "Enter Angles->";
    cin >> angles;
    cout << "Enter Starting Angle->";
    cin >> inc_angle;
    allocate_datamemory();

    ofstream infile(argv[2]);
    infile << wavelengths << "\n" << angles << "\n" ;
    if (imageflag==1){
        cout << "\n Random thickness variation:";
        cin >> vary;
    }
    for(xx=1;xx<=hpixels;xx++)
    {
        for(yy=1;yy<=vpixels;yy++)
        {
            if (imageflag==1)
    // Randomize layer one thickness
    temp=thickness[1];
}
for(i=1;i<=wavelengths;i++)
{
    wavelength[i]=lambda+((i-1)*25e-9);
    infile << wavelength[i] << "\n" ;
    if (dispersionflag==1)
    {
        for ( m=1; m<=layers; m++ )
        {
            if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion(wavelength[i], NO[m],NZA1[m],NZA2[m]);
            N[m]=cauchy_dispersion(wavelength[i],N[m],NA1[m],NA2[m]);
            if ( anisoflag[m] == 2 )
            {
                NZ[m]=cauchy_dispersion(wavelength[i],NZO[m],NZA1[m],NZA2[m]);
                NY[m]=cauchy_dispersion(wavelength[i],NYO[m],NYA1[m],NYA2[m]);
            }
        }
    }
    if (dispersionflag==2)
    {
        for ( m=1; m<=layers; m++ )
        {
            cout << "in->" <<N[m];
            N[m]=drude_dispersion(wavelength[i],NO[m],NA1[m],NA2[m]);
            cout << "out->" <<N[m];
            if ( anisoflag[m] >= 1 ) NZ[m]=drude_dispersion(wavelength[i], NZO[m],NZA1[m],NZA2[m]);
            if ( anisoflag[m] == 2 ) NY[m]=drude_dispersion(wavelength[i], NYO[m],NYA1[m],NYA2[m]);
        }
    }
    for(j=1;j<=angles;j++)
    {
        angle[j]=inc_angle+(5*(j-1));
        infile << angle[j] << "\n";
        rho=calculate_rho(angle[j],wavelength[i]);
        phicalc[i][j] = calculate_phi(rho);
        deltacalc[i][j] = calculate_delta(rho);
        if (deltacalc[i][j] <0) deltacalc[i][j]=deltacalc[i][j]+360;
        if (real(rho)<0 && abs(imag(rho))<1.0e-6) deltacalc[i][j]=180;
        infile << phicalc[i][j] << "\n";
        infile << deltacalc[i][j] << "\n";
if (imageflag=1) thickness[1]=temp;

// Close File Stream //
infile.close();

void print_parms( char *argv[])
{
    int i, l, m;

    get_parameters(argv);
    cout << "\nTolerance:" << ftol;
    cout << "\nNumber of Layers:" << layers;
    cout << "\nDispersion Flag:" << dispersionflag;
    cout << "\nImaging Flag:" << imageflag;
    if (imageflag != 1) imageflag=0;
    if (imageflag == 1)
    {
        cout << "\nhpixels:" << hpixels;
        cout << "\nv pixels:" << vpixels;
    }

    // Read incident Medium Complex Index of Refraction //
    cout << "\nN[0] (Complex Index of Refraction):";
    cout << N[0];

    for (i=1; i<=layers; i++)
    {
        cout << "\nLAYER["<<i<<"] Anisotropicflag->:" << anisoflag[i];
        cout << "\nLAYER["<<i<<"] Solve for n flag->:" << nnflag[i];
        if (anisoflag[i]>=1)
        {
            cout << "\nLAYER["<<i<<"] Solve for nz flag->:" << nnzflag[i];
            if (anisoflag[i]==2) cout << "\nLAYER["<<i<<"] Solve for ny flag->:";
            << nnyflag[i];
        }
        cout << "\nLAYER["<<i<<"] Solve for k flag->:" << kkflag[i];
        if (anisoflag[i]>=1)
        {
            cout << "\nLAYER["<<i<<"] Solve for kz flag->:" << kkzflag[i];
            if (anisoflag[i]==2) cout << "\nLAYER["<<i<<"] Solve for ky flag->:";
            << kkyflag[i];
        }
        cout << "\nLAYER["<<i<<"] Solve for thickness flag->:" << ttflag[i];
        cout << "\nLAYER["<<i<<"] Complex Index of Refraction ->:" << N[i];
if (anisoflag[i] == 1)
{  
cout << "\nLAYER["<<i<<"] Complex Index of Refraction
   (Other Principle axis)->:" << NZ[i];
}
if (anisoflag[i] == 2)
{
  cout << "\nLAYER["<<i<<"] Complex Index of Refraction a axis ->:"  
    << NZ[i];
  cout << "\nLAYER["<<i<<"] Complex Index of Refraction b axis ->:" 
    << NY[i];
}
if (dispersionflag>=1)
{
  cout << "\nLAYER["<<i<<"] Dispersion Coefficient #1 ->:" << NA1[i];
  cout << "\nLAYER["<<i<<"] Dispersion Coefficient #2 ->:" << NA2[i];
  if (anisoflag[i] == 1 )
  {
   cout << "\nLAYER["<<i<<"] Dispersion Coefficient
      ab plane#1 ->:" << NZA1[i];
   cout << "\nLAYER["<<i<<"] Dispersion Coefficient
      ab plane#2 ->:" << NZA2[i];
  }
  if (anisoflag[i] == 2 )
  {
   cout << "\nLAYER["<<i<<"] Dispersion Coefficient
      ab plane#1 ->:" << NZA1[i];
   cout << "\nLAYER["<<i<<"] Dispersion Coefficient
      ab plane#2 ->:" << NZA2[i];
   cout << "\nLAYER["<<i<<"] Dispersion Coefficient
      c plane#1 ->:" << NYA1[i];
   cout << "\nLAYER["<<i<<"] Dispersion Coefficient
      c plane#2 ->:" << NYA2[i];
  }
  cout << "\nLAYER["<<i<<"] Thickness ->:" << thickness[i] << " Meters";
}
cout << "\nSubstrate Complex Index of Refraction ->:" << N[layers+1];
for (l=1;l<=mmp;l++)
{
  for (m=1;m<=nnp;m++)
  {
    cout << "\nParameter Matrix ["<<l<<"]["<<m<<"] Initial Starting
         guess ->:" << guess[l][m];
  }
}
cout << "\n Absorption coefficients at 632.8 nm ...";
cout << "\n Ambient->"<<4*PI*imag(-N[0])/(632.8e-9);
for (l=1;l<=layers;l++)
{
cout << "\n Layer["<<l<<"]->"<<4*PI*imag(-N[l])/(632.8e-9);
}
cout << "\n Substrate->"<<4*PI*imag(-N[layers+1])/(632.8e-9);
cout << "\n";

} 

void output_growthcurves(char *argv[])
{
  int output,m;
  float i;
  float thickness_max;
  float increment;
  float wave,ang;
  float phicalc_one, deltalcalc_one;
  complex<double> rho;
  get_parameters(argv);

  cout << "\nEnter Max thickness value->";
  cin >> thickness_max;
  cout << "\nEnter increment thickness value->";
  cin >> increment;
  if (dispersionflag>0)
  {
    cout << "\nEnter Angle of Incidence to evaluate->";
    cin >> ang;
  }
  else
  {
    cout << "\nEnter Wavelength to evaluate->";
    cin >> wave;
  }

  cout << "\nOutput (0 screen /1 file 'growthdat'/2 3D growdat file)";
  cin >> output;

  if (output==1)
  {
    ofstream growth("growthdat");
    if (dispersionflag>0) growth <<"\n Various wavelength mode
2008 400,500,600,700,800,900nm";
2009 else growth << "\n Various angle mode 30,40,50,60,70 degrees";
2010 for (i=0;i<thickness_max;i=i+increment)
2011 {
2012   thickness[i]=i;
2013   growth << "\n" << i;
2014   if (dispersionflag>0)
2015   {
2016     for(wave=400e-9;wave<=900e-9;wave+=100e-9)
2017     {
2018       if (dispersionflag==1)
2019         {
2020           for ( m=1; m<=layers; m++ )
2021             {
2022               if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion(wave,NZO[m],
2023                      NZA1[m],NZA2[m]);
2024               N[m]=cauchy_dispersion(wave,NO[m],NA1[m],NA2[m]);
2025               if ( anisoflag[m] == 2 )
2026                 {
2027                   NZ[m]=cauchy Dispersion(wave,NZO[m],NZA1[m],NZA2[m]);
2028                   NY[m]=cauchy_dispersion(wave,NYO[m],NYA1[m],NYA2[m]);
2029                 }
2030             }
2031         }
2032     }
2033     if (dispersionflag==2)
2034     {
2035         for ( m=1; m<=layers; m++ )
2036             {
2037               N[m]=drude_dispersion(wave,NO[m],NA1[m],NA2[m]);
2038               if ( anisoflag[m] >= 1 ) NZ[m]=drude Dispersion(wave,NZO[m],
2039                  NZA1[m],NZA2[m]);
2040               if ( anisoflag[m] == 2 ) NY[m]= drude Dispersion(wave,NYO[m],
2041                  NYA1[m],NYA2[m]);
2042             }
2043     }
2044     }
2045     rho=calculate_rho(ang,wave);
2046     phicalc_one=calculate_phi(rho);
2047     deltacalc_one=calculate_delta(rho);
2048     if (deltacalc_one<0) deltacalc_one=deltacalc_one+360;
2049     //cout << "\n" << abs(imag(rho))<< "\t deltacalc_one-> "//
2050     //<< deltacalc_one;///<
2051     if ( abs(imag(rho)) < 1.0e-6 && real(rho) < 0 ) deltacalc_one=180;
2052     growth << "\t" << phicalc_one << "\t" << deltacalc_one;
2053   }
2054 if(dispersionflag==0)
2055   {
2056     
2057   }
2058 if(dispersionflag==0)
for(ang=30;ang<=70;ang=ang+10)
{
    rho=calculate_rho(ang,wave);
    phicalc_one=calculate_phi(rho);
    deltacalc_one=calculate_delta(rho);
    if (deltacalc_one <0) deltacalc_one=deltacalc_one+360;
    //cout << "\n" << abs(imag(rho))<< "\t deltacalc_one-> "//
    //<< deltacalc_one;;
    if ( abs(imag(rho)) < 1.0e-6 && real(rho) < 0 ) deltacalc_one=180;
    growth << "\t" << phicalc_one << "\t" << deltacalc_one;
}
}
}
growth.close();
}
if (output==2)
{
    ofstream growth("growthdat3D");
growth << " TITLE="3D plots "\n";
growth << " VARIABLES="x" "y" "z" "w" " n";
growth << " ZONE I=100 J=100 F=POINT \n";
for (i=0;i<=thickness_max;i=i+increment)
{
    thickness[1]=i;
    if (dispersionflag>0)
    {
        for(wave=400e-9;wave<=900e-9;wave=wave+5e-9)
        {
            if (dispersionflag==1)
            {
                for ( m=1; m<=layers; m++ )
                {
                    if ( anisoflag[m] == 1 ) NZ[m]=cauchy_dispersion(wave,NZO[m],
NZA1[m],NZA2[m]);
                    N[m]=cauchy_dispersion(wave,NO[m],NA1[m],NA2[m]);
                    if ( anisoflag[m] == 2 )
                    {
                        NZ[m]=cauchy_dispersion(wave,NZO[m],NZA1[m],NZA2[m]);
                        NY[m]=cauchy_dispersion(wave,NO[m],NYA1[m],NYA2[m]);
                    }
                }
            }
            if (dispersionflag==2)
            {
                for ( m=1; m<=layers; m++ )
                {
...
N[m]=drude_dispersion(wave,NO[m],NA1[m],NA2[m]);
cout << "\n" "NO[m]" "\t" "N[m];
if ( anisoflag[m] >= 1 ) NZ[m]=drude_dispersion(wave,NZO[m],
NZA1[m],NZA2[m]);
if ( anisoflag[m] == 2 ) NY[m]= drude_dispersion(wave,NYO[m],
NYA1[m],NYA2[m]);
}
}
}
rho=calculate_rho(ang,wave);
phicalc_one=calculate_phi(rho);
deltacalc_one=calculate_delta(rho);
if (deltacalc_one <0) deltacalc_one=deltacalc_one+360;
//cout << "\n" << abs(imag(rho))w "t deltacalc_one-> " //
// << deltacalc_one://
if ( abs(imag(rho)) < 1.0e-6 && real(rho) < 0 ) deltacalc_one=180;
growth << "\n" << i << "\t" << wave << "\t" << phicalc_one <<
"\t" << deltacalc_one;;
}
if(dispersionflag==0)
{
for(ang=30;ang<=70;ang=ang+.4)
{
    rho=calculate_rho(ang,wave);
    phicalc_one=calculate_phi(rho);
    deltacalc_one=calculate_delta(rho);
    if (deltacalc_one <0) deltacalc_one=deltacalc_one+360;
    //cout << "\n" << abs(imag(rho))w "t deltacalc_one-> " //
    // << deltacalc_one://
    if ( abs(imag(rho)) < 1.0e-6 && real(rho) < 0 ) deltacalc_one=180;
    growth << "\n" << i << "\t" << ang << "\t" << phicalc_one << "\t"
    << deltacalc_one;
}
}
else
{
    for (i=0;i<=thickness_max;i=i+increment)
    {
        thickness[i]=i;
        rho=calculate_rho(wave,ang);
        phicalc_one=calculate_phi(rho);
        deltacalc_one=calculate_delta(rho);
        if (deltacalc_one <0) deltacalc_one=deltacalc_one+360;
if (real(rho)<0 && abs(imag(rho))<1.0e-6) deltacalc_one=180;

cout << "\n" << i << "," << phicalc_one << "," << deltacalc_one;
}

void output_phidelta(char *argv[])
{
    int i,j,m,xx,yy;
    float lambda, inc_angle;
    float vary;
    complex<double> rho;
    float phicalc_one;
    float deltacalc_one;
    get_parameters(argv);
    cout << "Enter Wavelength->";
    cin >> lambda;
    cout << "Enter Angle->";
    cin >> inc_angle;
    if (dispersionflag==1)
    {
        for ( m=1; m<=layers; m++ )
        {
            if ( anisoflag[m] == 1 )
                NZ[m]=cauchy_dispersion(lambda,NZO[m],NZA1[m],NZA2[m]);
            else
                N[m]=cauchy_dispersion(lambda,NO[m],NA1[m],NA2[m]);
            if ( anisoflag[m] == 2 )
                
                NZ[m]=cauchy_dispersion(lambda,NZO[m],NZA1[m],NZA2[m]);
                NY[m]=cauchy_dispersion(lambda,NYO[m],NYA1[m],NYA2[m]);
                
        }
    }
    if (dispersionflag==2)
    {
        for ( m=1; m<=layers; m++ )
        {
            N[m]=drude_dispersion(lambda,NO[m],NA1[m],NA2[m]);
            if ( anisoflag[m] >= 1 )
                NZ[m]=drude_dispersion(lambda,NZO[m],NZA1[m],NZA2[m]);
            else
                NY[m]=drude_dispersion(lambda,NYO[m],NYA1[m],NYA2[m]);
            
        }
    }
    rho=calculate_rho(inc_angle,lambda);
phi_calc_one = calculate_phi(rho);
deltacalc_one = calculate_delta(rho);
if (deltacalc_one < 0) deltacalc_one = deltacalc_one + 360;
if (real(rho) < 0 && abs(imag(rho)) < 1.0e-6) deltacalc_one = 180;
cout << " Phi=" << phi_calc_one << " Delta=" << deltacalc_one << "\n";
}

void csv_parmoutput(int nfunc, int xx, int yy)
{
complex<double> CCasin(const complex<double> z)
{
    return -imaginary*log(imaginary*z+sqrt(1-z*z));
}

complex<double> CCatan(const complex<double> z)
{
    return (log(1+imaginary*z)-log(1-imaginary*z)) / (2*imaginary);
}
Appendix C

Input/Output Examples

COMMAND → ./ellipcalc parmfile1 datafile1 7

Ellip-Sim

Select Choice:

1. Solve Optical Parameters
2. Make a Parameter File
3. Make a Ellipsometer Simulation Datafile
4. Generate Phi, Delta and Output to stdout
5. Generate Phi, Delta Film Growth Curve
6. Solve Optical Parameters with Angle of Inc. Error
7. Display Parameters
8. Make Phi, Delta Raw Datafile
9. Make Phi, Delta Error Datafile

Enter Wavelengths->
Tolerance:1e-06
Number of Layers:2
Dispersion Flag:2
Imaging Flag:0
N[0] (Complex Index of Refraction):(1,0)
LAYER[1] Anisotropicflag->:1
LAYER[1] Solve for n flag->:1
LAYER[1] Solve for nz flag->:0
LAYER[1] Solve for k flag->:0
LAYER[1] Solve for kz flag->:0
LAYER[1] Solve for thickness flag->:1
LAYER[1] Complex Index of Refraction ->:(3.53,1.51)
LAYER[1] Complex Index of Refraction
(Other Principle axis)->:(2.79,1.7)
LAYER[1] Dispersion Coefficient #1 ->:(1.71,0)
LAYER[1] Dispersion Coefficient #2 ->:(0.36,0)
LAYER[1] Dispersion Coefficient
ab plane#1 ->:(2.28,0)
LAYER[1] Dispersion Coefficient
ab plane#2 ->:(0.34,0)
LAYER[2] Anisotropicflag->:0
LAYER[2] Solve for n flag->:0
LAYER[2] Solve for k flag->:0
LAYER[2] Solve for thickness flag->:0
LAYER[2] Complex Index of Refraction ->:(6.7596,-0.104)
LAYER[2] Dispersion Coefficient #1 ->:(0,0)
LAYER[2] Dispersion Coefficient #2 ->:(0,0)
Substrate Complex Index of Refraction ->:(1.97,-3.72)
Parameter Matrix [1][1] Initial Starting guess ->:3.2
Parameter Matrix [1][2] Initial Starting guess ->:8e-08
Parameter Matrix [2][1] Initial Starting guess ->:3.8
Parameter Matrix [2][2] Initial Starting guess ->:8e-08
Parameter Matrix [3][1] Initial Starting guess ->:3.2
Parameter Matrix [3][2] Initial Starting guess ->:1.2e-07
Absorption coefficients at 632.8 nm ...
Ambient->-0
Layer[1]->-2.99681e+07
Layer[2]->2.06527e+06
Substrate->7.38731e+07

FILE→ parmfile1

1e-06
2
2
0
(1,0)
1
1
0
0
0
1
(3.53, 1.51)
(2.79, 1.70)
(1.71, 0)
(0.36, 0)
(2.28, 0)
(0.34, 0)
10e-08
0
0
0
0
(6.7596, -0.104)
(0.0, 0.0)
(0.0, 0.0)
1e-08
(1.97, -3.72)
3.2
8e-8
3.8
8e-8
3.2
12e-8

FILE → datafile1

2
2
3e-07
30
33.549
173.97
35
29.5233
170.051
3.25e-07
30
35.3551
169.952
35
32.1324
165.539
COMMAND→ ./ellipcalc parmfile1 datafile1 1

Ellip-Sim

Select Choice:

1. Solve Optical Parameters
2. Make a Parameter File
3. Make a Ellipsometer Simulation Datafile
4. Generate Phi, Delta and Output to stdout
5. Generate Phi, Delta Film Growth Curve
6. Solve Optical Parameters with Angle of Inc. Error
7. Display Parameters
8. Make Phi, Delta Raw Datafile
9. Make Phi, Delta Error Datafile

1

1. Standard Output format
2. CSV output to file output
3. CSV output to stdout

Iterating...
Pixel Array [1][1]

Number of evaluations: 127
Parameter Parameter ERROR
3.52997 1.00001e-07 7.97809e-09
3.52997 1.00001e-07 7.97809e-09
3.52997 1.00001e-07 7.97809e-09

144
Vita

Kyle John Peterson was born in Chicago, Illinois on September 8, 1975. He graduated from Fort Calhoun High School in Fort Calhoun, Nebraska in May of 1994. In August of 1994, Kyle entered North Park University in Chicago, Illinois where he received a Bachelor of Science degree in Physics in May of 1998. In the fall of that same year, he enrolled in the Master’s program in Physics at the University of Tennessee Space Institute. Two years later, Kyle received his Master’s degree in May of 2000.

Kyle is presently pursuing a Doctorate of Philosophy in Physics at the University of Tennessee Space Institute.