ELLIPSOMETRIC CHARACTERIZATION OF OXIDE FILMS ON SEMICONDUCTORS

by

Erik Benson

A senior thesis submitted to the faculty of

Brigham Young University - Idaho

in partial fulfillment of the requirements for the degree of

Bachelor of Science

Department of Physics

Brigham Young University - Idaho

December 2025

BRIGHAM YOUNG UNIVERSITY - IDAHO

DEPARTMENT APPROVAL

of a senior thesis submitted by

Erik Benson

This thesis has been reviewed by the research committee, senior thesis coordinator, and department chair and has been found to be satisfactory.

Date	Jon Paul Johnson, Advisor
Date	David Oliphant, Senior Thesis Coordinator
Date	David Oliphant, Committee Member
Date	Todd Lines, Committee Member
Date	Evan Hansen, Chair

ABSTRACT

ELLIPSOMETRIC CHARACTERIZATION OF OXIDE FILMS ON SEMICONDUCTORS

Erik Benson

Department of Physics

Bachelor of Science

The United States of America has recently made a greater push toward local technology production. This change necessitates an increased production of computer parts made from semiconductors and an improvement in the techniques to create them. This paper addresses the characterization of thin films on semiconductors, discusses code to analyze them, and presents suggestions on areas the research team can improve in the future.

ACKNOWLEDGMENTS

This paper never would've reached its conclusion without the support of many individuals. I first express my gratitude to the professors who guided me along this path, Prof. Datwyler, Prof. Hansen, Prof. Hill, Prof. Kelley, Prof. Johnson, Prof. Lines, Prof. Oliphant, and Prof. Hatt. Without your kind patience to answer all of my questions, I never would have made it this far.

I wish to express particular gratitude to Prof. Johnson, Prof. Lines, and Prof. Oliphant, who have actively provided feedback and suggestions that have allowed this paper to reach an acceptable state. I wouldn't understand half of what my paper is about without you.

Of course, I can't forget the family that supported me. My parents and grandparents have acted as a continual source of strength even when it felt hard enough to give up. My appreciation for your affection is endless, and my hope for the future is rooted in the pillars that you have helped me create in my life.

Finally, I thank you, dear reader. I hope the content of this thesis will bring you the benefit and growth you seek. I present it to the best of my effort and ability to you.

Contents

List of Figures

Chapter 1

Introduction

Semiconductors are rapidly becoming one of the most important components in production. They are used in practically every aspect of modern technology. However, approximately 60% of the global market is supplied by Taiwan [13]. This places The United States of America (USA), as well as the rest of the world, in a precarious position should something happen to the factories or supply lines there. The threat has spurred countries like the USA to push for an increase in local production. In response to the demand, this study sought to develop ellipsometry as a valid method for cost-effective characterization of semiconductors for undergraduate research teams. This method currently occupies relative popularity in industry, but the industry's way of using it requires expensive and dedicated machinery.

This thesis is a report on the efforts of our team in developing ellipsometry as a cost effective method for characterizing semiconductors in an undergraduate environment, without the use of the expensive and dedicated machinery. It is not the final goal of the research team's semiconductor research, but a support tool that will help the continuing efforts of the team in researching semiconductors and semiconductor devices.

1.1 Semiconductors

A semiconductor is a material used to create components of computers such as transistors that have the capability to rapidly switch between being on and off. This is how computers are able to communicate 0s and 1s. This allows them to form the basis for modern electronic components. Without them, modern computers would not exist, nor would even common technologies like Light Emitting Diodes (LEDs).

Silicon wafers are the material of choice for this study due to their properties as semiconductors and their relative affordability. There are other semiconducting materials that will work, but they were not addressed in this study. Part of the process of creating a semiconductor device is called doping. Doping the silicon wafer introduces or removes charge carriers using boron or phosphorus chemical solutions, after which the wafer is baked in an oven to oxidize it. Phosphorous creates what is called an N-type semiconductor, where it brings in an extra electron, making parts of the wafer more negative. Boron generates a P type and removes an electron to make parts of the wafer more positive. This creates sections that act like highways through the semiconducting material to control where the charge can flow.

The doping can be used to create wells of P or N type charge that are separated by gates. Gates are the term used to describe the sections of material between two wells and the distance between the wells is called the gate length. When supplied with a voltage, these gates allow the wells to be connected and for a current to flow between them. By selectively applying voltages, the on and off states can be interpreted as 0 and 1 to supply a computer with the foundation of its function [11].

1.2 Characterization

Some of the major physical components of the semiconducting silicon wafer include the substrate which is the original wafer, the thickness of the oxidation layer grown on the substrate, the structure of the molecular lattice, and the electrical properties make up the characteristics that define its function. Characterization is the act of measuring these components to describe the capabilities of a semiconductor.

There are numerous ways to achieve this characterization. Particular mention will be made of four primary methods. Three will be discussed in this section. The last is discussed in the next section.

1.2.1 Hall Effect

Passing a charge through a conductive material in a perpendicular magnetic field generates a potential difference that can be measured. This measurement can be used to determine holes or additional electrons in the lattice, as well as things like the electrical properties such as carrier density (the number of holes or extra electrons per unit volume), resistivity/conductivity (how strongly electrical flow is encouraged or opposed), and more. This method is both non-destructive and simple to set up and use, making it valuable [1] [7] [10].

1.2.2 Capacitance-Voltage

The Capacitance-Voltage method relies on the ability of a semiconductor to hold a charge. Typically expressed as $C = \frac{q}{V}$, this equation can be rewritten in terms of $V(C) = \frac{q}{C}$. Using a device to moderate the applied charge, the voltage is recorded to characterize the semiconductor. From this information, it is possible to calculate oxide thickness (thickness of the oxide layer on the substrate), doping density (dopants per unit volume), flat band capacitance (determines substrate doping density by finding the lowest limit of performance for a high-frequency metal-oxide capacitor), and flat band voltage (the minimum voltage to overcome the resistance of the gates). Ideally, this measurement is performed in a vacuum at high frequencies for greater accuracy, but it is possible to get a baseline in sub-optimal circumstances. It appears this method is not as broadly used as the Hall Effect or the other two methods. If necessary, this method may provide an alternative option for the team to try [5].

1.2.3 Scanning Electron Microscopy (SEM)

Though limited to substances of 1 μm in thickness, SEM can be used to characterize "...surface topography, crystalline structure, chemical composition and electrical behavior..." allowing for high accuracy understanding of the semiconductor. By firing an electron beam at the surface, the topography can be accurately mapped. This requires specific machinery, so it is unlikely to be affordable in an undergraduate environment, but it appears to be one of the most popular methods in industry. It is non-destructive and extremely detailed [12].

1.3 Ellipsometry

Ellipsometry uses the way that linearly polarized light is affected as it passes through the oxide layer, then reflects off the mirror-like silicon to calculate the thickness of the thin film and its index of refraction. It performs the calculations by measuring the way that the polarization and intensity of the refracted light vary. As the primary focus of this paper, it holds the advantages of being non-destructive and relatively inexpensive.

An ellipsometer is made up of two moving arms pointing at a clamp that can hold

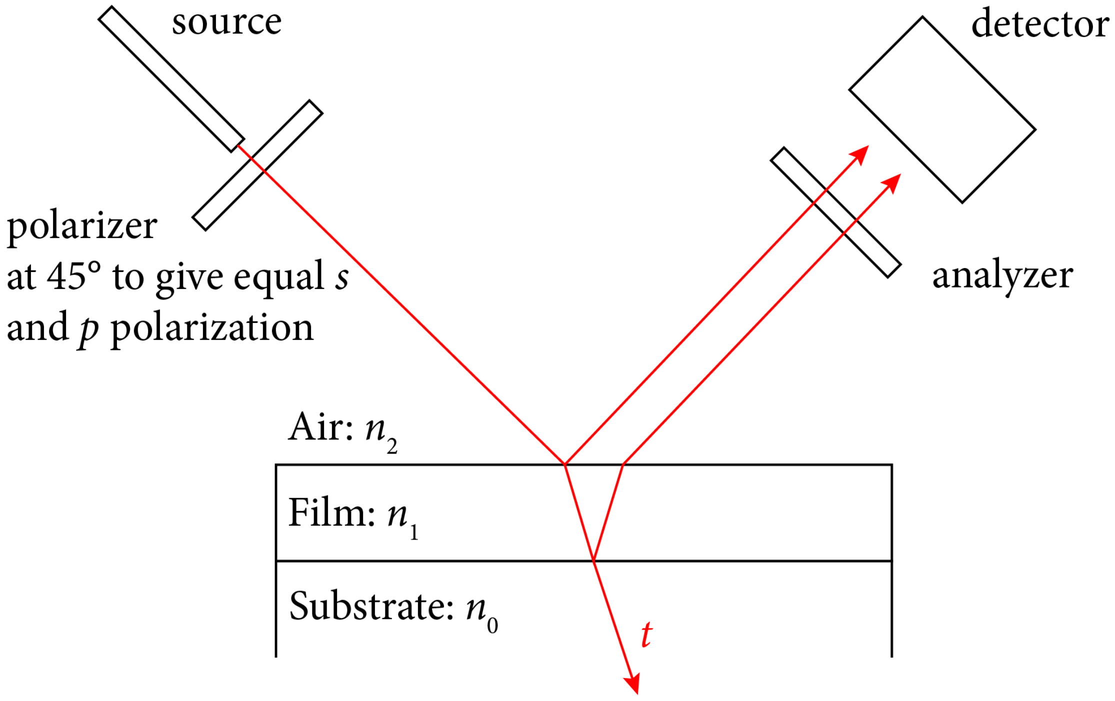


Figure 1.1 Ellipsometer

the semiconductor. One arm is set up with an incident laser that passes through a polarizer that is conventionally oriented at 45° to the surface before being reflected by the semiconductor. The other arm contains another polarizer called the analyzer and a photo detector at the receiving end, where data is collected, recording the intensity of the light. This process is repeated from several different angles of incidence. There are four different measurements for each of the analyzer angles necessary for the calculation. These four measurements are determined by the azimuthal angle of the analyzer: 0°, 90°, −45°, and 45°. The test data expects the incident light angles to be from 25° to 65°, with a measurement taken every 5°. Fewer can be taken, but accuracy will be impacted. The visual representation can be seen in figure 1.1.

This method requires software for the analysis. The software for this paper was written using a provided Excel document as the reference for the primary task for this research project. It will be discussed in detail in the methods section. Of particular note for this method is that it is prone to inaccuracies. Improper lasers, a dusty polarizer, or imprecise measurements can introduce significant variance in the results. It takes iterative experiments to reach a stable accuracy [6] [8].

Chapter 2

Index of Refraction

To properly understand ellipsometry, discussion of the mathematics of polarized light will be made here. As some important background, the complex index of refraction can be derived from the relative permittivity of the object. Starting with the definition,

$$n = \frac{c}{v} \tag{2.1}$$

where v can be defined as

$$v = \frac{1}{\sqrt{\mu\epsilon}}. \tag{2.2}$$

μ is permeability, and ϵ is permittivity. Permittivity is approximated by the equation:

$$\epsilon(\omega) = 1 + \frac{Ne^2}{2\epsilon_0 m} \sum_j f_j(\omega_j^2 - \omega^2 - i\omega\gamma_j)^{-1} \tag{2.3}$$

where f_j is the oscillator strength, N is the number of particles per unit volume, e is the charge of an electron and and m is electron mass, while ω_j is the natural frequency and γ_j is the damping constant [9].

With that relationship, n becomes

$$n = \sqrt{\frac{\mu\epsilon}{\mu_0\epsilon_0}} \tag{2.4}$$

Note the relationships

$$\frac{\mu}{\mu_0} \approx 1 \tag{2.5}$$

$$\epsilon_r = \frac{\epsilon}{\epsilon_0}. \tag{2.6}$$

This means that n can be approximated as

$$n = \sqrt{\epsilon_r} \tag{2.7}$$

where ϵ_r is the relative permittivity [9].

Further discussion of these relationships can be found in other materials. Sufficient for this paper is the understanding that the electrical property of relative permittivity creates a logical connection between light and the characteristics of a semiconducting material.

Using Snell's Law

$$n_i sin\theta_i = n_t sin\theta_t \tag{2.8}$$

where the subscript i represents the incident and t represents the transmitted. Additionally, using the angle at which polarized light is perfectly transmitted, known as Brewster's angle, to create the relationship $\theta_p + \theta_t = 90$ where θ_p is the polarization angle. This relationship allows for Snell's equation to take the form,

$$n_i sin\theta_p = n_t cos\theta_p \tag{2.9}$$

where the subscripts hold the same meaning as prior, which can be solved for:

$$tan\theta_p = \frac{n_t}{n_i}. \tag{2.10}$$

An inverse tangent will then give the θ [3].

Setting Snell's law aside for the moment, three waves need to be identified: the incident

$$\mathbf{E} = \mathbf{E}_0 e^{i\mathbf{k}\cdot\mathbf{x} - i\omega t} \tag{2.11}$$

$$\mathbf{B} = \sqrt{\mu\epsilon}\frac{\mathbf{k} \times \mathbf{E}}{k} \tag{2.12}$$

refracted

$$\mathbf{E}' = \mathbf{E}'_0 e^{i\mathbf{k}'\cdot\mathbf{x}-i\omega t} \tag{2.13}$$

$$\mathbf{B}' = \sqrt{\mu'\epsilon'}\frac{\mathbf{k}' \times \mathbf{E}'}{k'} \tag{2.14}$$

and the reflected

$$\mathbf{E}'' = \mathbf{E}''_0 e^{i\mathbf{k}''\cdot\mathbf{x}-i\omega t} \tag{2.15}$$

$$\mathbf{B}'' = \sqrt{\mu\epsilon}\frac{\mathbf{k}'' \times \mathbf{E}''}{k} \tag{2.16}$$

forming the three important waves. μ represents permeability, ϵ represents permittivity, while $\mathbf{k}$ is a wave vector with frequency ω[4].

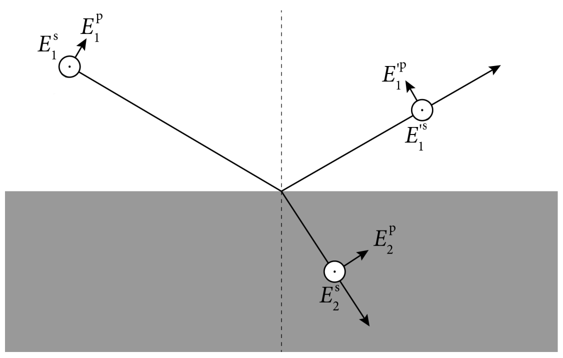
The above equations 2.11 to 2.16 can then be used to define the boundary conditions between two layers. This can be calculated from the E wave perpendicular to the surface, and E wave parallel to the surface. The two equations to represent the E perpendicular to the optical plane (s polarization)

$$R^s_{12} = \frac{E'_0}{E_0} = \frac{n_2 cos\theta_2 - n_1 cos\theta_1}{n_2 cos\theta_2 + n_1 cos\theta_1} \tag{2.17}$$

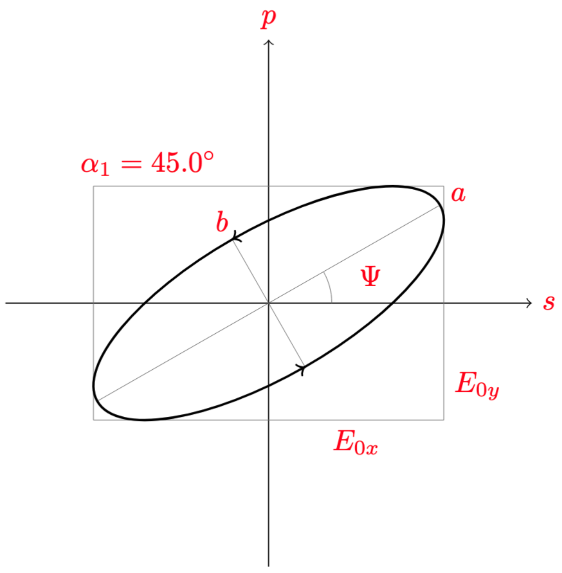
and the E parallel to the optical plane (p polarization)

$$R^p_{12} = \frac{E''_0}{E_0} = \frac{n_2 cos\theta_1 - n_1 cos\theta_2}{n_2 cos\theta_1 + n_1 cos\theta_2} \tag{2.18}$$

define the boundary condition between two layers. See the left image of Figure 2.1 for the visualization of this. The subscripts 12 represent going from air to the thin film, where 2 represents the incident. These equations take the same form going from the thin film to the substrate, but the subscript becomes 01 and the rest of the terms change accordingly. Full discussion of the derivations can be found in several of the references and many other optics textbooks [2][4][9].



Represents the s and p polarization. Note that s comes out of the page, and p is parallel to it. The orientation of this view is called the optical plane.



Visualization of the appearance of the polarized beam of light after reflection as seen by looking down the beam. The ratio $\frac{b}{a}$ is equivalent to Δ.

Figure 2.1 Visualization of s and p polarization.

Next,

$$R^s = \frac{r_{12}^s + r_{01}^s e^{-2i\beta}}{1 + r_{12}^s r_{01}^s e^{-2i\beta}} \tag{2.19}$$

$$R^p = \frac{r_{12}^p + r_{01}^p e^{-2i\beta}}{1 + r_{12}^p r_{01}^p e^{-2i\beta}} \tag{2.20}$$

give the Total Reflection of the s and p polarizations [4]. Where

$$\beta = 2\pi \frac{d_1}{\lambda} n_1 cos\theta_1 \tag{2.21}$$

is the Phase Film Thickness, which calculates the phase the wave is at when it reflects off the substrate in the thin film. λ is the wavelength of the incident light, d_1 is the thickness of the thin film as light passes through it, n_1 is the complex index of refraction of the thin film, and θ_1 is the angle of the beam in the thin film.

From there, R^p and R^s are used to create a new equation, the fundamental equation of ellipsometry, also known as the complex reflectance ratio.

$$\rho = \frac{R^p}{R^s} = tan\Psi e^{i\Delta} \tag{2.22}$$

where Ψ is the angle created by the ratio between the p and s polarizations, and Δ is the phase change between them. Ψ can be extracted using

$$\Psi = tan^{-1}(\sqrt{\rho^* \rho}) = tan^{-1}(|\rho|) \tag{2.23}$$

which removes the imaginary term by using the complex conjugate. The process for extracting Δ is a bit more complicated. Δ is the angle between the real and imaginary axis, so to calculate it, the relationship $\rho = Re + iIm$, will be the starting point. From there,

$$\rho + \rho^* = 2R \tag{2.24}$$

and

$$\rho - \rho^* = 2iI \tag{2.25}$$

meaning,

$$Re(\rho) = \frac{\rho + \rho^*}{2} \tag{2.26}$$

$$Im(\rho) = \frac{\rho - \rho^*}{2} \tag{2.27}$$

are the definitions of the real and imaginary components of ρ. Using these definitions, Δ can be extracted using

$$\Delta = tan^{-1}(\frac{\rho - \rho^*}{\rho + \rho^*}). \tag{2.28}$$

Note that this is mathematical background, and not directly implemented into the code. The easiest way to achieve this is by using the cmath python library or arctan2 from the numpy library, which is the method used by the code to calculate the Δ. Further there is another way to experimentally calculate Ψ and Δ which will be discussed in the Methods chapter.

Chapter 3

Code Function Calls

The purpose of this chapter is to provide a descriptive understanding of the calling order of the methods in the code. Description of the purpose of each method along with the proper order in which to call them will be presented here. Not all methods need to be called manually, and the proper call order is represented by its position in the following list.

Starting with a python class called Ellipsometer, the function calls follow as such:

1. data_entry(): guides the user through the process of entering in data to the .csv file.

2. view_data(): displays the contents of the .csv file for inspection.

3. Calculate(): Calculates Ψ and Δ from the .csv file.

   (a) Cal_Psi(): Calculates the Ψ term.

   (b) Cal_Delta(): Calculates the Δ term.

4. Model(): Calculates the Model Fitting. This is a theoretical ideal that will be compared to the collected data.

(a) to_rad():Converts degrees to radians.

(b) COS0(): Cos of the Angle of the Environment

(c) COS1(): Cos of the Angle in the Thin Film

(d) COS2(): Cos of the Angle in the Substrate

(e) P1_PI(): p polarization of r_{12}^{p} the atmosphere to thin film

(f) P1_SIG(): s polarization of r_{12}^{s} the atmosphere to thin film

(g) P2_PI(): p polarization of r_{01}^{p} the thin film to substrate

(h) P2_SIG(): s polarization of r_{01}^{s} the thin film to substrate

(i) Beta(): Film Phase Thickness

(j) P_PI(): Total Reflection of p polarization (R^p)

(k) P_Sigma(): Total Reflection of s polarization (R^s)

(l) parameter_P(): ρ term, calculated with $\frac{R^p}{R^s}$

(m) model_Psi_rads(): value of the modeled Ψ in radians

(n) model_Delta_rads(): value of the modeled Δ in radians

(o) to_deg_psi(): converts Ψ to degrees.

(p) to_deg_delta(): converts Δ to degrees.

5. Fit_Err(): Calculates the fitting error rate as a percentage

   (a) mem(): defines indexes of the matching values between the experimental and modeled data.

   (b) Err(): calculates the error between two values using the formula $\frac{|a-b|}{b} * 100$ where a is the experimental value and b is the modeled value.

   (c) SNE(): Sum of Normalized Errors

6. Plot_PD(): Plot only Psi and Delta

7. Plot(): Plot Psi, Delta, and the Error rate.

Note: methods 2, and 6 are optional. Method 1 can be optional depending on the user's coding experience. Discussed in methods. Figure 3.1 below is a simplified flowchart of the call order for readability.

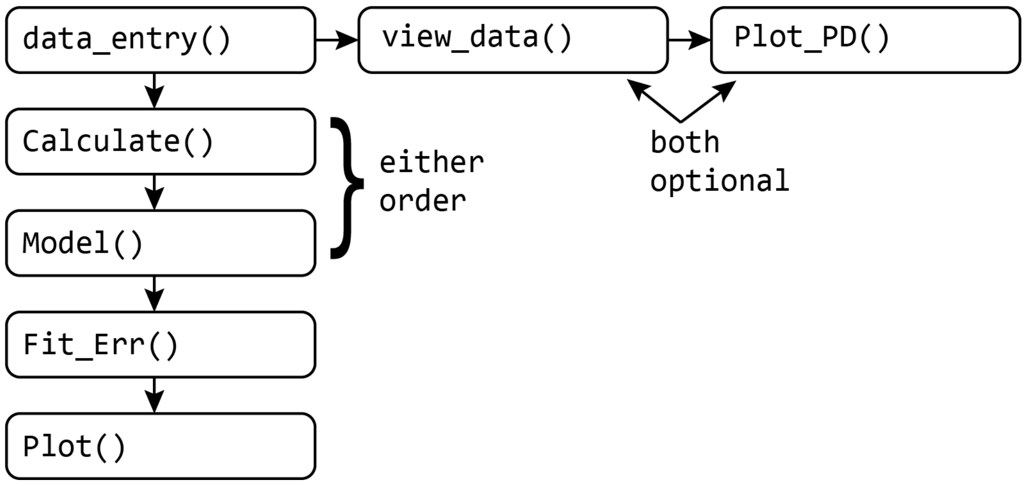


Figure 3.1 Function Calls Flowchart

Chapter 4

Methods

At the beginning of the project term, the goal was twofold: the first was successfully performing photolithography, the second was to replicate a 3D-printed ellipsometer using the designs from the paper: Optical measurements on a budget: A 3D-printed ellipsometer[8]. Neither the team's photolithography nor doping results will be discussed in this paper. The ellipsometer was already 3D printed, but the software was incomplete. The python code for the software will be fully discussed below.

The original Excel file was broken down into 4 tabs: Instructions, Exp.Data Collecting, Exp.Data Processing, and Model Fitting. Where Exp. stands for Experimental. Following convention, the same format will be utilized while skipping the instructions. See appendix A for the full code and the link to the GitHub.

4.1 Handling the Data

In an effort to consider all skill levels, multiple methods were included for convenience. The data is held in a .csv file, which can be directly manipulated by more experienced users, allowing quick input of the collected data. For the less experienced users of

.csv files, the method: data_entry() was included to guide the user step by step to enter the data. When downloading this code to use, the path to the .csv file needs to be adjusted to the user's needs.

Full discussion of the derivations can be found in Bixby [8], for our purposes, the important values are Ψ and Δ. Ψ "...is an angle that describes the anisotropic reflection of p and [s] polarization components...". Δ "...is the phase shift between the [s] and p components acquired upon reflection from the thin-film system." It is not possible to measure Ψ and Δ directly, as they are complex numbers with real and imaginary components, which necessitates this code for the calculations.[8]

As per the Bixby paper [8] the formula for Ψ takes the form

$$\Psi = \frac{1}{2}\cos^{-1}\left(\frac{I(90^\circ) - I(0^\circ)}{I(90^\circ) + I(0^\circ)}\right) \tag{4.1}$$

where $I(x)$ indicates the intensity measurement at the given polarizer angle x. Δ is

$$\Delta = \cos^{-1}\left(\frac{\left(\frac{I(45^\circ)-I(-45^\circ)}{I(45^\circ)+I(-45^\circ)}\right)}{2\Psi}\right) \tag{4.2}$$

The paper also lists an alternative form of equation 4.2 that is used with only three measurements, without the -45° but we made the decision to standardize using all 4 for accuracy, and it is restricted to an $\alpha_1 = \pm 45^\circ$, which is the angle of the initial polarizer. These are the values entered in to the .csv. See Appendix A for the specific layout of the data.

For each angle of incidence, measurements are taken at the 4 given analyzer angles, meaning that each angle has its own Ψ and Δ.

4.2 Model Initial Conditions

Before discussing the following parameters, it is important to recognize that n and k are specific values of the complex index of refraction for the given parts of the material.

$$\tilde{n} = n + ik \tag{4.3}$$

The original documentation recommends using the website https://refractiveindex.info to get this information for each part of the material. Optionally, the result from equation 2.7 could be used instead, but sufficient for the purposes of the code, looking up the known values is simpler.

The key details of creating a model to compare to the experimental data comes from the initial conditions. The first to know are n_2 and k_2, which are the index of refraction of the atmosphere. Next are n_1 and k_1, which are the index of refraction for the thin film, and d_1 the thickness of the substrate. This will be a guess, and the code can adjust to find a best fit. The next group is the substrate index of refraction n_0 and k_0. Finally, the incident wavelength λ. See Fig 1.1 for the visual representation of this.

The parameters for the index of refraction of the atmosphere are known and applicable unless the measurement is taken in a vacuum, which causes a slight difference in the value.

$$n_2 = 1 \tag{4.4}$$

$$k_2 = 0 \tag{4.5}$$

For the thin film, the model uses known quantities and a guess. n_1 and k_1 are pulled from the above website, but d_1 occupies an interesting spot, as it is a guess in units of Angstroms.

$$d_1 = 1485\text{Å} \tag{4.6}$$

is a good guess for this value, as the expected result would be roughly that much. This parameter can be adjusted for a closer fit.

n_0 and k_0 are the index of refraction of the substrate, so we follow the same process to determine what they are. λ is dependent on the laser used for the measurement and we assume that the utilized laser is monochromatic.

4.3 Model Fitting

The following are a host of equations used as the intermediate parameters to calculate the model Ψ and Δ. See chapter 3, item 4 for the list of methods each parameter is assigned to. Note: angles are always calculated in terms of radians, but the expected input of the code is degrees. The code handles the conversion. Further note: the subscripts of this section and previous sections have been retooled to match the proper convention. The code was written before knowledge of this convention, so it treats the atmosphere as n_0 as opposed to n_2, while the paper does the reverse for convention. It is better convention to make the substrate n_0 so that additional layers can be placed on top of each other. This paper is only concerned with a single layer.

1. **Cosine of Angle in Ambient**:

$$\cos\phi_2 = \sqrt{1 - \sin^2\phi_2}$$

This is taken from Snell's Law, which is defined in terms of sin instead of cos, so the identity $sin^2\theta + cos^2\theta = 1$ was used to rewrite in terms of cos.

2. **Cosine of Angle in Film**:

$$\cos\phi_1 = \frac{\sqrt{n_1^2 - n_2^2\sin^2\phi_2}}{n_1}$$

3. **Cosine of Angle in Substrate**:

$$\cos\phi_0 = \frac{\sqrt{n_0^2 - n_2^2\sin^2\phi_2}}{n_0}$$

4. **Interface Reflection (Ambient-Film, p-pol)**:

$$r_{12}^p = \frac{n_1 \cos \phi_2 - n_2 \cos \phi_1}{n_1 \cos \phi_2 + n_2 \cos \phi_1}$$

See equation 2.12.

5. **Interface Reflection (Ambient-Film, s-pol)**:

$$r_{12}^s = \frac{n_2 \cos \phi_2 - n_1 \cos \phi_1}{n_2 \cos \phi_2 + n_1 \cos \phi_1}$$

See equation 2.11.

6. **Interface Reflection (Film-Substrate, p-pol)**:

$$r_{01}^p = \frac{n_0 \cos \phi_1 - n_1 \cos \phi_0}{n_0 \cos \phi_1 + n_1 \cos \phi_0}$$

See equation 2.12. Note the change in subscripts represents the light passing through the thin film to the substrate.

7. **Interface Reflection (Film-Substrate, s-pol)**:

$$r_{01}^s = \frac{n_1 \cos \phi_1 - n_0 \cos \phi_0}{n_1 \cos \phi_1 + n_0 \cos \phi_0}$$

See equation 2.11. Note similar change in subscripts as the Interface Reflection (p-pol).

8. **Film Phase Thickness**:

$$\beta = 2\pi \frac{d_1}{\lambda} n_1 \cos \phi_1$$

See equation 2.14.

9. **Total Reflection (p-pol)**:

$$R^p = \frac{r_{12}^p + r_{01}^p e^{-2i\beta}}{1 + r_{12}^p r_{01}^p e^{-2i\beta}}$$

10. **Total Reflection (s-pol)**:

$$R^s = \frac{r_{12}^s + r_{01}^s e^{-2i\beta}}{1 + r_{12}^s r_{01}^s e^{-2i\beta}}$$

11. **Rho (Complex Ratio)**:

$$\rho = \frac{R^p}{R^s} = tan\Psi e^{i\Delta}$$

See equation 2.22.

12. **Modeled Psi (rad)**:

$$\Psi = \arctan(|\rho|)$$

See equation 2.23.

13. **Modeled Delta (rad)**:

$$\Delta = cmath.phase(\rho)$$

See equations 2.18 to 2.22 for the full derivation.

This can optionally be converted back to degrees. This code was written to take inputs and outputs in terms of degrees, so it has taken that additional step. A python loop is recommended as the previous calculations need to be made from 1° to 89°. While this many calculations are optional (only the ones in common from experimental angles of incidence are necessary) having this many allows for neat and clear plotting.

4.4 Error Fitting

Using the equation

$$\rho = tan(\Psi)e^{i\Delta} \tag{4.7}$$

and restricting it to the angles of incidence used in the experimental data, we can project Ψ and Δ into the complex-space which can be used to see how much of a shift occurs. Reminder: ρ is complex.

This code does not take advantage of this potential comparison, neither does the original reference document. It is present as an option to the user to implement if desired.

To calculate the error (E), the equations

$$E_\Psi = \frac{|\Psi_{exp} - \Psi_{model}|}{\Psi_{model}} \tag{4.8}$$

$$E_\Delta = \frac{|\Delta_{exp} - \Delta_{model}|}{\Delta_{model}} \tag{4.9}$$

are used, which can then be plotted for a visual representation of the error. The figure takes the form seen in Fig 4.2 using the provided test data.

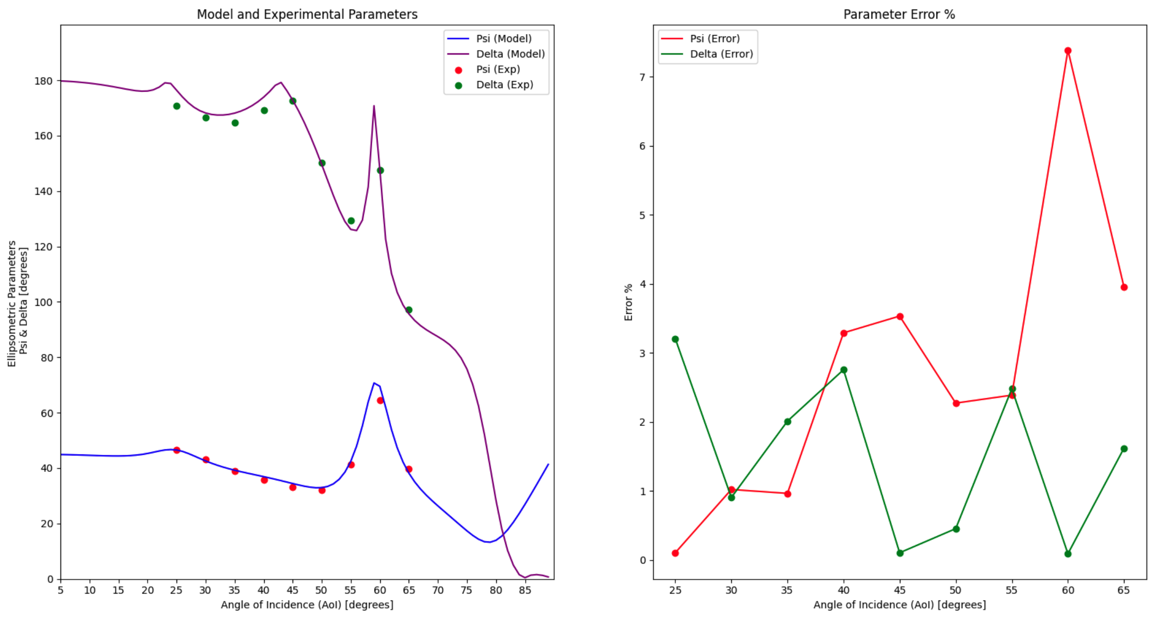


Figure 4.1 Code outputs with the reference data.

The left graph compares the model calculated Psi and Delta to the experimentally calculated one. The right graph represents the absolute percentage error between the values.
Note: the y axis of the left plot says "Ellipsometric Properties Psi & Delta (degrees)." y for the right plot is "Error %." x is the same for both, stating "Angle of Incidence (degrees)."

Chapter 5

Discussion

The purpose of this section is to discuss the project as a whole, with particular mention to its shortcomings at the time of research. The code was successfully completed, but the experimental method is flawed and there is more to be done.

5.1 State of the Code

In its current form, the code is feature complete. There is room for improvement, including a better optimization parameter for the model to reduce the error, as well as more potentially efficient ways to write the code. This code was written with its primary purpose being legibility for future researchers. Inefficiencies were intentionally included to improve readability.

Though the code matches the outputs given by the Bixby [8] paper with the reference data, there are limitations that have not been made clear in the documentation. Some of these limitations were only discovered while testing the code. An example of the code's tendency to blow up when exceeding certain hidden bounds. This became especially clear when testing our experimental data which will be discussed in the next section.

Researchers on the team do consider the code to be sufficient for our purposes. The data we are collecting theoretically exists within the limits of the code's capacity. Due to the current state of the experimental methods the team was using, the values are too unreliable to properly test the code. Compare 4.1 and 5.1, where 4.1 is the reference data used to test the code, and 5.1 is the data that our team collected.

5.2 State of the Experimental Data Collection

The method we had used for collecting the data is flawed. There are issues in practically every aspect of the device, which shall be broken down one at a time below. The primary reason these issues remained unsolved came down to time. We took most of the available time just getting the code ready and working, leaving no time to properly inspect the physical device and leading to the following problems.

5.2.1 The Laser and Polarizer

The laser we used was a cheap one bought online, it had high beam divergence, creating an astigmatism of the light, and an imprecise wavelength (λ) somewhere in the range 625nm to 750nm, the red spectrum, which was not properly identified with a spectrometer. Similar issues were shared by the polarizer, where its exact parameters were unknown, necessitating a test following Malus's law which was not performed during the research period. Worsening the problem, this polarizer had been frequently handled and as a result, was dirty and warped. This creates potential inconsistencies in the data and can affect the parameters of the polarizer - particularly the intensity of the light. The intensity was recorded with the program Logger Pro.

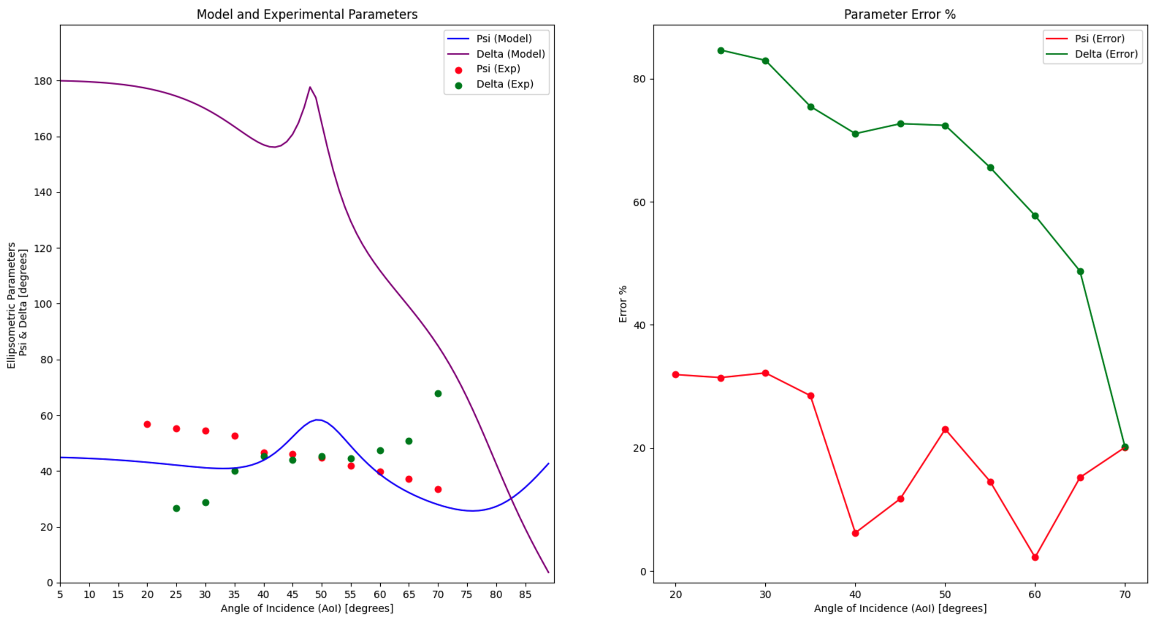


Figure 5.1 Code output with the team's collected data.

The team's attempt to collect data with the ellipsometer. Note the high error rate of the right graph, reaching as high as 80%.
Note: the y axis of the left plot says "Ellipsometric Properties Psi & Delta (degrees)." y for the right plot is "Error %." x is the same for both, stating "Angle of Incidence (degrees)."

5.2.2 The Ellipsometer

As outlined in the Bixby[8] paper, we had a 3-D printed ellipsometer to use for the experiment. This ellipsometer was inherited from a previous research team, and had suffered significant damage. Dials were broken off, clamps no longer worked, and the polarizer mount seemed broken. Even prior to the damage, the previous team was not able to get this ellipsometer working. For all these reasons, the ellipsometer did not meet the needs of the team.

5.3 Future Suggestions

The first priority needs to be fixing the data collection of the experiment. Discussions with the supervising professor indicated that the biggest issue was likely the data collection method. The absolute error would likely still be high until the other issues are resolved, but fixing the data collection method would likely have the biggest impact, potentially greater than any other suggestion made. The current state of the data collection renders any output from the code unreliable. Until the team refines the data collection method, a proper laser and polarizer is found, and the ellipsometer is fixed, no progress can be made. Once the data collection is done, the code can be adjusted to precisely meet the teams needs. The research team has acquired a goniometer, with hopes that it may allow the team to resolve many of these issues.

Bibliography

[1] Marco Crescentini, Sana Fatima Syeda, and Gian Piero Gibiino. “Hall-Effect Current Sensors: Principles of Operation and Implementation Techniques”. In: *IEEE Sensors Journal* 22.11 (2022), pp. 10137–10151. DOI: 10.1109/JSEN.2021.3119766.

[2] Grant R. Fowles. *Introduction to Modern Optics*. 2nd ed. New York: Dover Publications, 1989. ISBN: 0-486-65957-7.

[3] Eugene Hecht. *Optics*. 4th ed. Addison-Wesley, 2002. ISBN: 0-8053-8566-5.

[4] John David Jackson. *Classical Electrodynamics*. 3rd ed. New York: John Wiley & Sons, 1999. ISBN: 0-471-30932-X.

[5] J. Joy et al. “Capacitance-voltage profiling of MOS capacitors: A case study of hands-on semiconductor testing for an undergraduate laboratory”. In: *American Journal of Physics* 86.10 (Oct. 2018), pp. 787–796. ISSN: 1943-2909. DOI: https://doi.org/10.1119/1.5052360.

[6] Jesper Jung et al. *Ellipsometry*. Semester Project Report. Aalborg, Denmark: Aalborg University, 2004. URL: https://homes.nano.aau.dk/kp/Ellipsometry/main.pdf.

[7] Andrzej Kowalewski et al. “Semiconductor contact layer characterization in a context of hall effect measurements”. In: *Metrology and Measurement Systems*

(Jan. 2019), pp. 109–114. ISSN: 2300-1941. DOI: https://doi.org/10.24425/mms.2019.126324.

[8] Matthew Mantia and Teresa Bixby. "Optical measurements on a budget: A 3D-printed ellipsometer". In: *American Journal of Physics* 90.6 (June 2022), pp. 445–451. ISSN: 1943-2909. DOI: https://doi.org/10.1119/10.0009665.

[9] David Oliphant. "Characterization of Uranium, Uranium Oxide and Silicon Multilayer Thin Films". MA thesis. Brigham Young University, 2000. URL: https://physics.byu.edu/docs/thesis/953.

[10] E.H. Rhoderick. "The hall effect — an important diagnostic tool". In: *III-Vs Review* 13.3 (May 2000), pp. 46–51. ISSN: 0961-1290. DOI: https://doi.org/10.1016/S0961-1290(00)88881-X.

[11] R. P. Smith et al. "Introduction to semiconductor processing: Fabrication and characterization of p-n junction silicon solar cells". In: *American Journal of Physics* 86.10 (2018), pp. 740–746. DOI: 10.1119/1.5046424.

[12] K.D. Vernon-Parry. "Scanning electron microscopy: an introduction". In: *III-Vs Review* 13.4 (July 2000), pp. 40–44. ISSN: 0961-1290. DOI: https://doi.org/10.1016/S0961-1290(00)80006-X.

[13] World Population Review. *Semiconductor Manufacturing by Country 2025*. Retrieved October 8, 2025. 2025. URL: https://worldpopulationreview.com/country-rankings/semiconductor-manufacturing-by-country (visited on 10/08/2025).

Appendix A

Code

https://github.com/EnriqueBenny/Ellipsometry_BYUI

```
import numpy as np
import cmath as cm
import matplotlib.pyplot as plt
import csv


class Ellipsometer:
    def __init__(self):
        """
            Initializes the Ellipsometer class.

            variables:
                self.alpha_1:   Polarizer Azimuth Angle (alpha 1) in degrees (int)
                self.alpha_2:   Analyzer (alpha 2) angles in degrees (array of int)
                self.Aoi:       Angle of Incidence (Aoi) in degrees (array of int)
                self.model_Aoi: Angle of Incidence (Aoi) in degress for the model calculation (array of int)
                self.psi:       Psi value (array of float)
                self.delta:     Delta value (array of float)
                self.exp_P:     Our P-Value as determined by the experiment (array of float)
                self.wavel:     Wavelength of Light from the laser (float)

                The comments below, from self.n0 to self.k2
                   need to be fixed.
                It is not clear what purpose they serve, so
                   the comments are currently general.

                self.n0:        Real Index of Refraction for
                   Air (int)
                self.k0:        Imaginary Index of Refraction
                   for Air (int)
                self.n1:        Real Index of Refraction for
                   the Thin Film (float)
                self.k1:        Imaginary Index of Refraction
                   for the Thin Film (float)
                self.d1:        Estimation of the Thickness
                   of the Thin Film (needs optimization
                   method) (float)
                self.n2:        Real Index of Refraction for
                   the Substrate (float)
                self.k2:        Imaginary Index of Refraction
                   for the Substrate (float)
        """

        self.alpha_1 = 45
        self.alpha_2 = [45,90,-45,0]
        self.Aoi = [20,25,30,35,40,45,50,55,60,65,70,75]
        self.model_Aoi = np.arange(1,90,1)
        self.psi = []
        self.delta = []
        self.exp_P = []
        #The following parameters may be potentially broken.
        self.wavel = 6530.0
        # Index of Refraction Solver Inputs
        self.n0 = 1
        self.k0 = 0
        self.n1 = 1.4830
        self.k1 = 0.0000
        self.d1 = 14857 # Angs
        self.n2 = 3.844
        self.k2 = 0.015793

    def view_data(self):
        '''
            Inspect the current data in the .csv file.
            Can be used to check proper updates while the
               code is running.
            The first line in the csv gives the formatting.
            Uncomment the 'next(data)' call to get rid of it.

            Parameters:
                None
        '''

        with open('Ellipsometry_BYUI/ellipsometry_data.csv')
```

```
            as data:
            #next(data)
            reader = csv.reader(data)
            for i in reader:
                print(i)

    def data_entry(self):
        '''
            Enter data into the .csv file. Mistakes are
               easier to correct in the .csv file.
            This function can be skipped, just enter the data
               directly into the
            .csv, but ensure that proper formatting is
               maintained.
            The line, writer.writerow(['Aoi',45,90,-45,0])
               exists to provide formatting.
        '''

        with open('Ellipsometry/ellipsometry_data.csv','w',
           newline='') as data:

            writer = csv.writer(data)
            writer.writerow(['Aoi',45,90,-45,0]) #Adds the
               formatting to the csv file.
            print('If data for an Aoi has not been collected,
               type "skip".')

```

```python
            for i in self.Aoi:
                line = [i]
                for j in self.alpha_2:
                    inp = input(f'For Aoi {i}, enter in {j} Intensity: ')
                    if inp.lower() == 'skip':
                        break
                    line.append(inp)
                print(f'Appending: {line}')
                writer.writerow(line)

        self.view_data()

    def Calculate(self): # Currently has issues with calculating Psi
        '''
            Calculates the Psi and Delta values.
            Pages 446 and 447 of the Mantia Bixby paper explain what Psi and Delta are.
            Contains the sub-functions Cal_Psi and Cal_Delta.
            Data must be entered prior to this function call for it to work.
        '''

        def Cal_Psi(nt, zr): # Mysterously Problematic
            '''
                Calculates Psi for a single line from the csv.
                This one must be called before Cal_Delta as Psi is used in calculating delta.
                The code curently does this automatically.

                Parameters:
                    self.psi = a class array (float) psi will be appended to.
                    nt = the ninety degree measurement
                    zr = the zero degree measurement
            '''
            a = np.radians(self.alpha_1)

            self.psi.append(np.arctan(np.tan(a)*\
                              np.tan(np.arccos((nt-zr)/(nt+zr))/2)))

        def Cal_Delta(ff,nff):
            '''
                Calculates Delta for a single line from the csv.
                This one must be called after Cal_Psi as Psi is needed for these calculations.
                The code currently does this automatically.

                Parameters:
                    self.delta = a class array (float) delta will be appended to.
                    ff = The Forty-Five degree measurement
```

```
                    nff = The Negative Forty-Five degree measurement
            '''
            a = np.radians(self.alpha_1)

            self.delta.append(np.arccos(((ff-nff)/(ff+nff))/\
                np.sin(2*np.arctan(np.tan(self.psi[-1])/np.tan(a)))))

        with open('Ellipsometry_BYUI/ellipsometry_data.csv','r') as data:
            reader = csv.reader(data)
            self.Aoi = [] # To keep the number of variables lower, this one is being reused to overwrite the
                          # initialized version.

            for line in reader:

                try: # The try/except exist to prevent issues ocurring if the formatting line is removed.
                    error_test = int(line[0])
                except:
                    continue

                if len(line) > 1: # This line exists to handle skipped measurements.
```

```
                    aoi = int(line[0]) # Angle of Incidence
                    ff = int(line[1]) # Forty-Five degree angle
                    nt = int(line[2]) # Ninety degree angle
                    nff = int(line[3]) # Negative Forty-Five degree angle
                    zr = int(line[4]) # Zero Degree Angle

                    self.Aoi.append(aoi)
                    Cal_Psi(nt,zr)
                    Cal_Delta(ff,nff)

            def to_deg(value):
                return np.degrees(value)
            self.psi = list(map(to_deg,self.psi))
            self.delta = list(map(to_deg,self.delta))
            #for i in range(len(self.psi)):
            #    print(self.psi[i])

    def Model(self):
        '''
            Calculates the Model Fitting. I used the map() function as a quick way to
            apply repeated processes while avoiding loops. This is going to be a
            very difficult section to read, as a warning.
        '''
```

```python
        # N Parameters
        N0 = complex(self.n0, self.k0)
        N1 = complex(self.n1, self.k1)
        N2 = complex(self.n2, self.k2)


        def to_rad(deg):
            return np.radians(deg)
        model_rads = list(map(to_rad, self.model_Aoi)) # Create an array from the model_Aoi of radians.

        # All following functions are for intermediate parameters.
        # See columns T to AD in the third tab on the excel document.
        def COS0(rads):
            return cm.sqrt(1.0 - cm.sin(rads) * cm.sin(rads))
        cos0 = list(map(COS0, model_rads))

        def COS1(value):
            return cm.sqrt(N1**2 - N0**2 * (cm.sin(value)**2)) / N1
        cos1 = list(map(COS1, model_rads))

        def COS2(value):
            return cm.sqrt(N2**2 - N0**2 * (cm.sin(value)**2)) / N2
        cos2 = list(map(COS2, model_rads))

        def P1_PI(value1, value2):
            return  (N1*value1 - N0 *
               value2)/(N1*value1+N0*value2)
        p1_pi = list(map(P1_PI, cos0, cos1))

        def P1_SIG(value1,value2):
            return (N0*value1 -
               N1*value2)/(N0*value1+N1*value2)
        p1_sig = list(map(P1_SIG, cos0, cos1))

        def P2_PI(value1, value2):
            return (N2*value1-N1*value2)/(N2*value1+N1*value2)
        p2_pi = list(map(P2_PI, cos1, cos2))

        def P2_SIG(value1,value2):
            return (N1*value1-N2*value2)/(N1*value1+N2*value2)
        p2_sig = list(map(P2_SIG, cos1, cos2))

        def Beta(value):
            return 2 * np.pi * (self.d1 / self.wavel) * N1 *
               value
        beta = list(map(Beta, cos1))

        def P_PI(value1, value2, value3):
            exp = cm.exp(-2j*value3)
            return (value1 + value2 * exp)/(1 + value1 *
               value2 * exp)
        p_pi = list(map(P_PI, p1_pi, p2_pi, beta))

        def P_Sigma(value1, value2, value3):
            exp = cm.exp(-2j*value3)
            return (value1+value2*exp)/(1+value1*value2*exp)
        p_sigma = list(map(P_Sigma, p1_sig,p2_sig,beta))

        def parameter_P(value1,value2):
            return value1/value2
        P = list(map(parameter_P, p_pi, p_sigma))

        # The following has been made available for the
           entire class.
        def model_Psi_rads(value):
            return np.arctan(np.abs(value))
        self.Psi_rads = list(map(model_Psi_rads,P))

        def model_Delta_rads(value):
            return cm.phase(value)
        self.Delta_rads = list(map(model_Delta_rads,P))

        def to_deg_psi(value):
            return np.degrees(value)
        self.Psi_deg = list(map(to_deg_psi,self.Psi_rads))

        def to_deg_delta(value):
            return np.degrees(np.abs(value))
        self.Delta_deg =
           list(map(to_deg_delta,self.Delta_rads))

    def Model_P(self):
        '''
            Calculates P values from the Model.
            The self.Model() function must be run for this
               function
            to work, so it was included to prevent user error.
            This is an optional method that a later team can
               build upon.
            Neither this document or the original excel make
               use of it.
        '''
        self.Model()

        def P(value1, value2):
            return cm.tan(value1) * cm.exp(1j * value2)
        self.mod_P =
           list(map(P,self.Psi_rads,self.Delta_rads))

    def Experiment_P(self):
        '''
            Taken from the second tab of the referenced Excel
               Document,
            this function calculates our P values from the
               experimental data.
            The model P is calulated in Model_P.
            This is an optional method that a later team can
               build upon.
            Neither this document or the original excel make
               use of it.
        '''
        def to_rads(value):
            return np.radians(value)
        psi_rad = list(map(to_rads,self.psi))
        delta_rad = list(map(to_rads,self.delta))

        def Psi_Cal(value):
            return cm.tan(value)
        psi_cal = list(map(Psi_Cal,psi_rad))

        def Delta_Cal(value):
            return cm.exp(cm.rect(1, value))
        delta_cal = list(map(Delta_Cal,delta_rad))

        def Exp_P(value1, value2):
            return value1*value2
        self.exp_P = list(map(Exp_P,psi_cal,delta_cal))

    def Fit_Err(self):
        '''
```

```
            Calculates the fitting error rate as a %. Must be
                run after Model().
        '''
        # This function is here to identify the index values
            in common between the model and experimental Aoi.
        def mem():
            index = []
            aoi = set(self.Aoi)
            for i in range(len(self.model_Aoi)):
                if self.model_Aoi[i] in aoi:
                    index.append(i)
            return index
        index = mem()

        comp_Psi = []
        comp_Delta = []
        for i in index:
            comp_Psi.append(self.Psi_deg[i])
            comp_Delta.append(self.Delta_deg[i])

        def Err(value1,value2):
            return (np.abs(value1-value2)/value2)*100
        self.psi_err = list(map(Err,self.psi,comp_Psi))
        self.delta_err = list(map(Err,self.delta,comp_Delta))

        def SNE(value1,value2):
            return ((value1-value2)/value2)**2
        self.Psi_sse_norm =
           sum(list(map(SNE,self.psi,self.Psi_deg)))
        self.Delta_sse_norm =
           sum(list(map(SNE,self.delta,self.Delta_deg)))
        self.Avg_see_norm =
           (self.Psi_sse_norm+self.Delta_sse_norm) / 2

    def Plot_PD(self):
        '''
            Plot Aoi vs Psi and Aoi vs Delta.
            Must be called after the Calculate method.
            This is a function to give a quick check on Psi
               and Delta. It contains no other data.
        '''

        plt.scatter(self.Aoi,self.psi)
        plt.scatter(self.Aoi,self.delta)
        plt.xlim(5,90)
        plt.xticks(np.arange(5,90,5))
        plt.ylim(0,200)
        plt.yticks(np.arange(0,200,20))
        plt.show()

    def Plot(self):
        '''
            Complete Plot statement. Mimics the plot from the
               second tab of the Excel document.
        '''
        fig = plt.figure(figsize = (40,10))

        ax1 = fig.add_subplot(1,2,1)
        ax2 = fig.add_subplot(1,2,2)

        ax1.set_title('Model and Experimental Parameters')
        ax1.plot(self.model_Aoi,self.Psi_deg, color = 'blue',
           label='Psi (Model)')
        ax1.plot(self.model_Aoi,self.Delta_deg, color =
           'purple', label='Delta (Model)')
        ax1.scatter(self.Aoi, self.psi, color='red',
           label='Psi (Exp)')
        ax1.scatter(self.Aoi, self.delta, color='green',
           label = 'Delta (Exp)')
        ax1.legend()
        ax1.set_xlabel('Angle of Incidence (AoI) [degrees]')
        ax1.set_ylabel('Ellipsometric Parameters \n Psi &
           Delta [degrees]')

        ax1.set_xlim(5,90)
        ax1.set_xticks(np.arange(5,90,5))
        ax1.set_ylim(0,200)
        ax1.set_yticks(np.arange(0,200,20))

        ax2.set_title('Parameter Error %')
        ax2.plot(self.Aoi,self.psi_err, color = 'red',
            label='Psi (Error)')
        ax2.plot(self.Aoi,self.delta_err, color = 'green',
            label='Delta (Error)')
        ax2.scatter(self.Aoi,self.psi_err, color = 'red')
        ax2.scatter(self.Aoi,self.delta_err, color = 'green')
        ax2.legend()
        ax2.set_xlabel('Angle of Incidence (AoI) [degrees]')
        ax2.set_ylabel('Error %')

        plt.show()

# Class Call
E = Ellipsometer()

# Method Calls
#E.data_entry() # Keep commented if entering data via csv
#E.view_data() # Optional
E.Calculate() # Call after the data is entered
E.Model() # Call after Calculate
E.Fit_Err() # Call after Model
#E.Plot_PD() # Call after Calculate (Optional)
E.Plot() # Call last

''' A copy of the test data for the .csv.
Aoi,45,90,-45,0
25,1,60,137,67
30,3,101,202,88
```

```
35,11,211,373,137
40,18,310,501,160
45,25,357,524,153
50,49,295,402,117
55,50,145,219,113
60,29,31,141,135
65,167,225,214,155
'''
'''A copy of data collected by the team for the .csv
20,2392,797,47,1876
25,4690,1454,422,3049
30,5488,2111,516,4128
35,8068,3143,1220,5441
40,10131,5206,1782,5816
45,12383,6895,2017,7458
50,13368,7411,2345,7317
55,12054,7129,2064,5722
60,10319,7598,2064,5253
65,9662,7739,2345,4456
70,7083,7223,3424,3189
'''
```

Listing A.1 Ellipsometry_Data_Processing.py