ROUGHNESS CORRECTION MODEL FOR REFLECTION FROM PERFECTLY CONDUCTING SCATTERERS

by

W. Todd Doughty

A senior thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Bachelor of Science

Department of Physics and Astronomy

Brigham Young University

August 2008

BRIGHAM YOUNG UNIVERSITY

DEPARTMENT APPROVAL

of a senior thesis submitted by

W. Todd Doughty

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

Date	R. Steven Turley, Advisor
Date	Eric G. Hintz, Research Coordinator
Date	Ross L. Spencer, Chair

ABSTRACT

ROUGHNESS CORRECTION MODEL FOR REFLECTION FROM PERFECTLY CONDUCTING SCATTERERS

W. Todd Doughty

Department of Physics and Astronomy

Bachelor of Science

I modeled the reflectance from rough conductive surfaces for tranverse magnetic (TM) and tranverse electric (TE) polarization. The Nyström technique was applied in order to solve the Electric Field Integral Equation (EFIE) in the TM case and the Magnetic Field Integral Equation (MFIE) in the TE case. We studied 2.4 million sample surfaces with varied roughness heights and frequencies from various incident angles and compared the results to the predictions of the Debye-Waller Factor (DWF). As predicted, the attenuation is directly correlated with the qh factor, but differs from the DWF in both form and magnitude. There was also a significant dependance on both spatial frequency and polarization. We developed our own model by fitting the results to a cubic correction function. In addition to the predicted quadratic term, our simulations showed significant linear and cubic terms in the roughness correction function.

ACKNOWLEDGMENTS

I would like to thank R. Steven Turley who spent many hours with me helping me to accomplish this project and Jed Johnson for his work on the algorithms that have made my project possible. Also, I would especially like to thank the NASA Rocky Mountain Space Grant Consortium, BYU Office of Research and Creative Activities, and the BYU Department of Physics and Astronomy for providing funding in support of this research.

Contents

List of Figures

Chapter 1

Introduction

1.1 Significance

Light provides a powerful tool for probing the physical world, however the Extreme Ultraviolet (EUV) portion of the spectrum is one of the most difficult regions with which to work. Approximately spanning from 20-100 eV, EUV light is absorbed in air. All work done with these wavelengths must be done under vacuum, adding great difficulty to the work. Because of this difficulty, applications of EUV have only recently become a focus of attention. While there are many applications of EUV technology, two of the most important are in astronomy and computer manufacturing.

EUV astronomy is a relatively new field that will be able to probe unique features of the universe. The blackbody spectrum of the solar corona peaks in the EUV, so this region is the best suited to study that important feature of the sun. The solar and heliospheric observatory (SOHO) takes advantage of the EUV to image the heliosphere. Closer to home, the earth's magnetosphere traps many charged particles including ionized helium. The Lyman-α spectral line (304 Å) for ionized helium is in the EUV. Ionized helium can provide a beacon with which to study the nature

of the magnetosphere provided it can be isolated from the dominant light at 584 Å, the Lyman-α line for neutral helium [1]. Each of these applications requires precise mirror design.

Computers are constantly improving in speed and performance but are currently approaching a maximum limit for current production techniques. This limit stems from the size of the circuits which in turn are limited by the resolution of the production method: lithography. One method of decreasing the resolution size of optical lithography is to use wavelengths of light in the EUV [2]. In order to utilize these short wavelengths, new optical tools must be designed that can direct and control the high energy light of the EUV [3].

In order to reflect light in the EUV, each of these applications requires precisely engineered mirrors. The best approach is a series of alternating levels of thin films with correct thicknesses [4]. This approach exploits the wave nature of light by using constructive interference to amplify the reflected light. However, in order to correctly design the mirrors, the optical properties for the material must be known. One method for determining the index of refraction for a material is to measure the reflected light of various frequencies off of a surface from various incident angles [5]. The Fresnel equations (1.1,1.2) relate the reflected field to the index of refraction, n [6].

$$R_s = \left(\frac{n_i \cos(\theta_i) - n_t \cos(\theta_t)}{n_i \cos(\theta_i) + n_t \cos(\theta_t)} \right)^2 \tag{1.1}$$

$$R_p = \left(\frac{n_i \cos(\theta_t) - n_t \cos(\theta_i)}{n_i \cos(\theta_t) + n_t \cos(\theta_i)} \right)^2 \tag{1.2}$$

These equations, however, assume an ideal interface between the two layers (i.e. a perfectly flat surface). Any real surface has roughness that causes a decrease in reflectance. This in turn leads to errors in calculating the optical constants used in

designing mirrors. In order to correct this problem, the loss of light, or attenuation, due to roughness must be correctly understood and accounted for.

1.2 Roughness

As improvements continue in extreme-ultraviolet (EUV) optics, surface roughness remains one of the main obstacles to be overcome. Roughness causes an increase in scattering which in turn can cause a decrease in reflectivity [4,7]. Overall, the effect of roughness on a surface decreases the reflected field. For low frequency surface features (i.e. those larger than a wavelength), there can be specular scattering. Under this circumstance, the large features can act as angled facets reflected the incident wave away from the reflected angle predicted by Snell's Law (see Figure 1.1). For high frequencies, the effect of roughness stems from the randomization of the phase of the reflected wave. This leads to speckling and other effects due to the interference of the wavefronts (see Figure 1.2).

A common method for dealing with roughness is the use of the Debye-Waller correction factor (DWF) to correct for the attenuation [8]. Originally developed through the study of light scattering due to thermal vibration in crystals, Debye determined that the scattering of light by the crystals was based on the product of the momentum of the incident light and the amplitude of the vibrating atoms in the crystal lattice [9]. This product represents an uncertainty factor that can predict the attenuation. In reflection from scatterers, roughness acts in a way analogous to the thermal vibrations [8,10]. Surface roughness accounts for the positional variation in a relation similar to the vibrational amplitude, while the momentum of the light remains the same. Previous work has postulated that the factor used in calculating the attenuation due to scattering in vibrating crystals can accomplish the same role

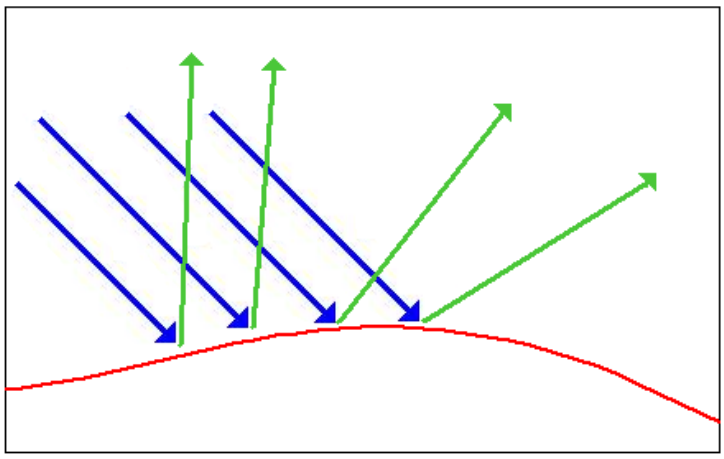

Figure 1.1 For low frequency roughness, the reflected field can be considered using Snell's Law reflection. The incident beams (blue) reflect at different locations on the surface (red) which have different orientations. The different slopes the surface lead to a diffuse spread of the light beams (green) and an overall decrease in reflection.

in scattering due to surface roughness [10]. This scalar correction factor can be calculated using a simple formula (1.3).

$$R = \exp(-2(qh)^2) \tag{1.3}$$

R is the attenuation, the ratio of the intensity of the field reflected by the rough scatterer with the theoretical maximum intensity of a perfectly flat mirror. The parameter, q, is the momentum of the light perpendicular to the surface of the scatterer; it is calculated using $q = 2\pi \sin\theta/\lambda$. Finally, h is the rms surface roughness height.

The qh factor is one important determinant in scattering because it represents a phase difference in the light. The momentum transfer, q, is the component of the

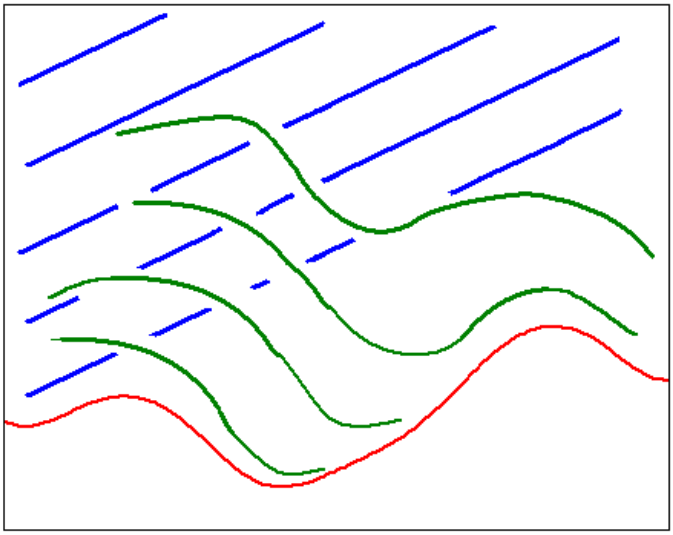

Figure 1.2 For high frequency roughness, the incident wave (blue) reflects from the surface (red) at different times leading to a randomization of the phases for the reflected field. This leads to decrease in specular reflectance as the reflected wave (green) interferes with itself.

wave vector, **k**, perpendicular to the surface. The ratio, qh, represents the number of wavelengths of light that the roughness represents and determines the form of the phase difference of the light scattered at different locations. This phase difference, in turn, determines interference effects on the surface that lead to attenuation. This means that the ratio represented by qh is important to a roughness correction.

The DWF assumes a gaussian distribution of vibrational motion to determine the the correction factor (1.3). While this is a good approximation for thermal vibration, it is ultimately non-physical for describing a rough surface. A gaussian distribution for roughness removes any correlation between atoms on a surface. Any two adjacent atoms would have no physical connection. In reality, there is physical correlation between atoms on a surface, so the DWF breaks down for calculations of scattering

due to roughness. The purpose of this paper is to determine the accuracy of the DWF and to develop a more accurate correction factor which accounts for the surface atom correlation using a form(1.4) motivated by the DWF (1.3).

$$R = \exp(f(\theta, h, \sigma)) \tag{1.4}$$

1.3 Previous Work

Our approach to determining the accuracy was to calculate the scattering from a 2.4 million rough surfaces with varying roughness heights and frequencies. Many methods exist for determining the scattering of light off of a surface. A common, simple approach based from principles taught introductory physics classes is the use of geometric optics, or ray tracing. This method is frequently utilized for similar problems [11–14]. At every point on a surface, the direction of the reflected ray is determined by the incident ray and the slope of the surface tangent using the law of reflection, $\theta_i = \theta_r$. Each of these rays represents and energy bundle that changes with the density of the rays. This approach is a simple, fast method for approximating the reflected field, however, it cannot predict wave phenomena of light such as diffusion. A more accurate model for calculating scattering is physical optics, also referred to as the Kirchhoff Approximation or Tangent Plane Approximation [13, 14]. This method involves computing the reflected field by calculating the current induced at the surface at every point by assuming an infinite tangent plane at every point. While this approach is more accurate than geometric optics, it still ignores the secondary effects due to currents induced by the fields reflected at nearby points. In addition, the Physical Optics doesn't include the effects of diffraction at edges and surface discontinuities.

In order to avoid the errors caused by the inaccuracies for either of these two methods, we will use the Field Integral Equations (FIE). The FIE are derived from Maxwell's equations using the boundary conditions for a scattering system. Much recent work has been done developing different methods to solve the FIE for random surfaces [15–18]. Many of these approaches have used the method of moments, which approximates the surface currents **J** and **K** with linear basis functions. For any discretization scheme, a linear approximation will be less accurate than a higher order approximation. In order to minimize calculation time and maximize accuracy while maintaining stability, we used a higher order discretization scheme, the Nyström method [19, 20], to solve for the reflected field.

There has been a lot of previous work in our research group to better understand the effects of roughness on reflectance. Jed Johnson, in his master's thesis [21], developed the algorithms used to calculate the reflectance for rough two-dimensional surfaces. Using Jed's derivations, Elise Martin, in a preliminary study of the reflectance of TM polarized light from rough surfaces, found the attenuation to be a function of qh, as predicted by DWF [22]. Continuing this study, this paper will discuss the effects of TE polarization in combination with a further study of the TM in order to develop a roughness correction function for the scattered field with arbitrarily polarized incident light. This paper will first discuss the setup for a perfect conductor, including the development of a physically realistic two-dimensional surface. Next, the results for test cases (i.e. a flat surface and a cylinder) are compared with the analytical results. Finally, the paper will show and discuss the results of the attenuation for both the TE and TM polarizations (see Chapter 3).

Chapter 2

Procedure

2.1 Problem Setup

A common approach to the scattering problem is the use of the FIE. Derived from the Helmholtz equation in the presence of sources, the FIE relates the scattered field, $\mathbf{E^s}$, to the surface currents $\mathbf{J}$ and $\mathbf{K}$. This outline follows the more complete derivation laid out by Peterson [23].

$$\mathbf{E^s} = -\frac{\nabla(\nabla \cdot \mathbf{A}) + \mathbf{k^2 A}}{\mathbf{i}\omega\epsilon_\mathbf{0}} - \nabla \times \mathbf{F} \tag{2.1}$$

$$\mathbf{H^s} = -\frac{\nabla(\nabla \cdot \mathbf{F}) + \mathbf{k^2 F}}{\mathbf{i}\omega\mu_\mathbf{0}} - \nabla \times \mathbf{A} \tag{2.2}$$

where $\mathbf{A}$ and $\mathbf{F}$ are found by applying the Green's function to the surface currents.

$$\mathbf{A}(\mathbf{x}) = \int_\mathbf{S} \mathbf{J}(\mathbf{x}')\mathbf{G}(\mathbf{x} - \mathbf{x}')\mathbf{d^2 x}' \tag{2.3}$$

$$\mathbf{F}(\mathbf{x}) = \int_\mathbf{S} \mathbf{K}(\mathbf{x}')\mathbf{G}(\mathbf{x} - \mathbf{x}')\mathbf{d^2 x}' \tag{2.4}$$

The Green's function for the Helmholtz equation in two dimensions free space is shown below.

$$G(x - x') = \frac{i}{4} H_0^{(1)}(k|x - x'|) \tag{2.5}$$

The scattered field can be calculated by solving the total field on the boundary using suitable boundary conditions for a perfect conductor. The total field is the superposition of the incident electric field, $\mathbf{E^{inc}}$, and the scattered field.

$$\mathbf{E} = \mathbf{E}^{\text{inc}} + \mathbf{E}^{\text{s}} \tag{2.6}$$

Applying the boundary conditions for the case of a perfect conductor greatly simplifies the problem. For the TM polarization, the EFIE reduces to (2.7), while the TE polarization reduces to (2.8).

$$E_z^{\text{inc}} = -ik\eta A_z \tag{2.7}$$

$$H_z^{\text{inc}} = -\frac{1}{2} J_t - \left[\frac{\partial A_y}{\partial x} - \frac{\partial A_x}{\partial y} \right] \tag{2.8}$$

2.2 Surface Setup

In order to provide a useful simulation basis, the model surfaces require an overall random distribution to account for the deposition procedure, but also a correlation between atoms that represents the physical association between atoms on a surface. We used a two-dimensional surface, in order to make the computation feasible. However, because of the polarized nature of light the results can be generalized to three-dimensions. We used a gaussian distribution to setup a random array of points. This led to a surface with roughness at the large scale. These points were then interpolated

with a cubic spline to create the full surface, satisfying both conditions for a realistic surface.

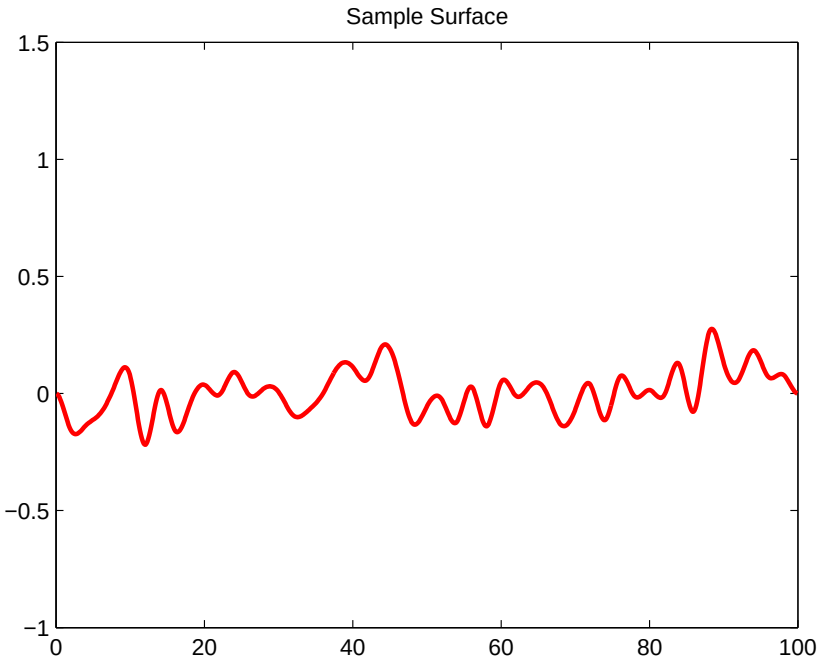

Figure 2.1 An example flat surface similar to those used to calculate the TM polarization. The surface parameters are length 100λ, height $.1\lambda$, and knot separation 2λ.

The surface generated by this method can be characterized by a cutoff frequency which is inversely related to the knot spacing, σ. The power spectrum (Figure 2.2) for the surface shown in Figure 2.1 shows this relationship. When using this correction method, the surface cutoff frequency and roughness height can be determined through various means, including atomic force microscopy.

The surface shown in Figure 2.1 is sufficient to solve for the reflected field of the TM polarized light. However, the calculation will break down for the TE polarization because of the form of the MFIE. The solution of the scattered field from the MFIE requires the derivative of the surface current to be known at all points of the surface. This form of the surface leads to singularities at the endpoints. In order to overcome

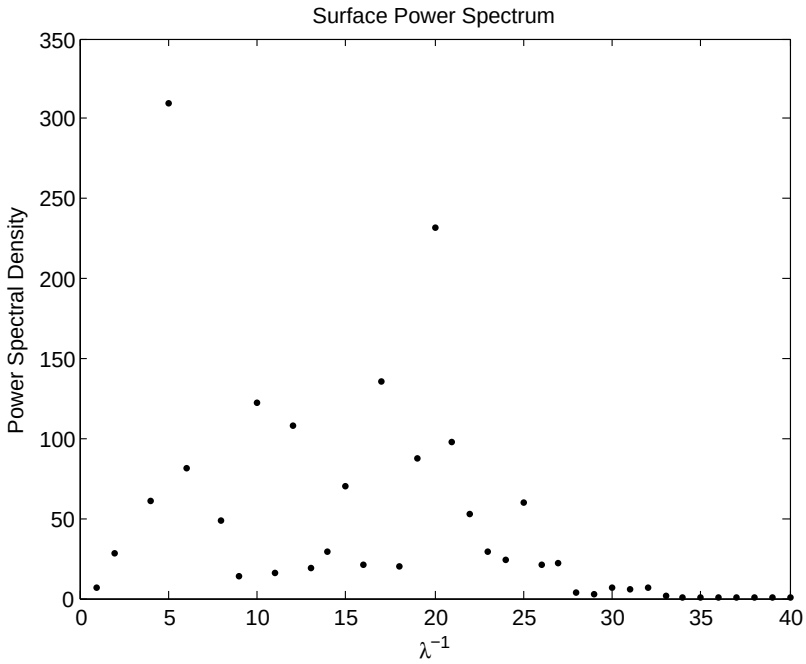

Figure 2.2 The power spectrum for the surface shown in Figure 2.1. The power spectrum shows a clear frequency cutoff that is related to the knot separation.

this problem, we chose to close the surface as shown in Figure 2.3. The endpoints of the surface were connected using hemispherical caps and a flat bottom surface. Both a continuous surface and a continuous slope are required in order to calculate the scattered field with the MFIE (Equation 2.8). Each of these conditions was incorporated into the interpolating spline used to complete the surface. While the setup of this surface adds a significant degree of difficulty to the solution of the problem, it will continue to apply in future studies and lend itself easily to the multilayer film problem common in EUV optics.

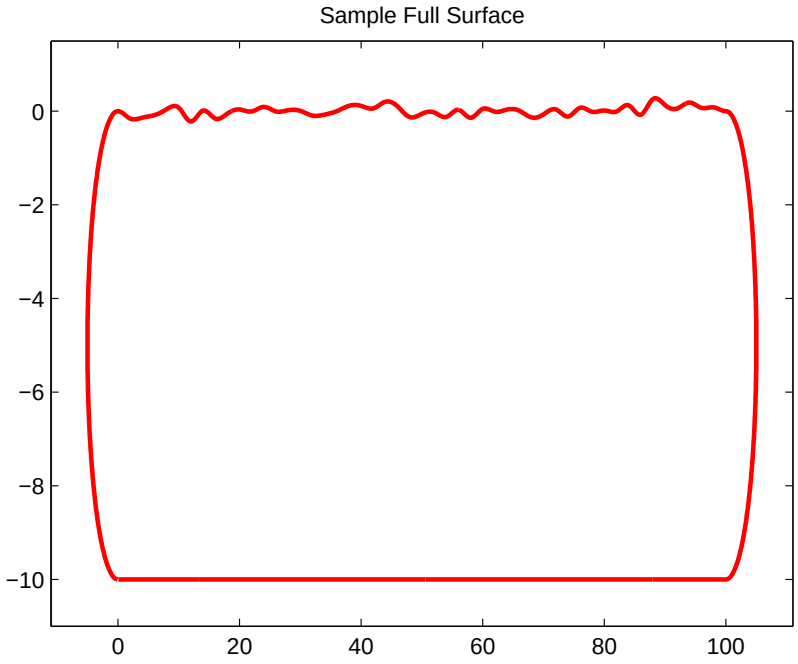

Figure 2.3 The same surface as shown in Figure 2.1 now shown using round end caps. This type of surface was used in calculating the TE reflected field. The scatterer thickness is 10λ.

2.3 Validation

The program has already been validated for the TM polarized light. For the results of this validation see Johnson's thesis [21]. For the TE polarization, we will compare the computed results for a cylinder and a flat plate to the expected analytical solution. These solutions represent relatively simple analytical calculations that can be used to check the more general algorithms used in this program.

2.3.1 Flat Plate

First we will compare the results of simulations with the field predicted by physical optics on a flat plate. The derivation for this section closely follows the derivation for TM polarization found in Johnson's thesis [21]. Applying the boundary condition that the Electric Field is zero on the surface, the MFIE is expressed

$$H_z^{\mathrm{S}} = \frac{e^{i\left(k\rho+\frac{\pi}{4}\right)}}{2\sqrt{2\pi k\rho}} \left\{ -ik \int_c \left[cos(\theta')sin(\phi) - sin(\theta')cos(\phi) \right] J_t(x', y') e^{-ik(cos(\phi)x' + sin(\phi)y')} ds' \right\} \tag{2.9}$$

The surface is a long smooth surface oriented along the $\hat{\mathbf{x}}$ with a normal in the $\hat{\mathbf{y}}$ (Figure 2.4). The symmetry axis of the situation is the $\hat{\mathbf{z}}$. We can solve for the current $\mathbf{J}$ using its definition.

$$\mathbf{J} = \hat{\mathbf{n}} \times \mathbf{H} \tag{2.10}$$

$$= \hat{\mathbf{y}} \times \mathbf{H^{inc}} + \mathbf{H^{s}} \tag{2.11}$$

$$J_t = \left(H^{\mathrm{inc}} + H^s \right) \hat{x} \tag{2.12}$$

$$= \left(e^{i\mathbf{p}\cdot\mathbf{x}} + e^{i\mathbf{q}\cdot\mathbf{x}} \right) \hat{x} \tag{2.13}$$

$$= \left[e^{i(p_x x + p_y y)} + e^{i(q_x x + q_y y)} \right] \hat{x}. \tag{2.14}$$

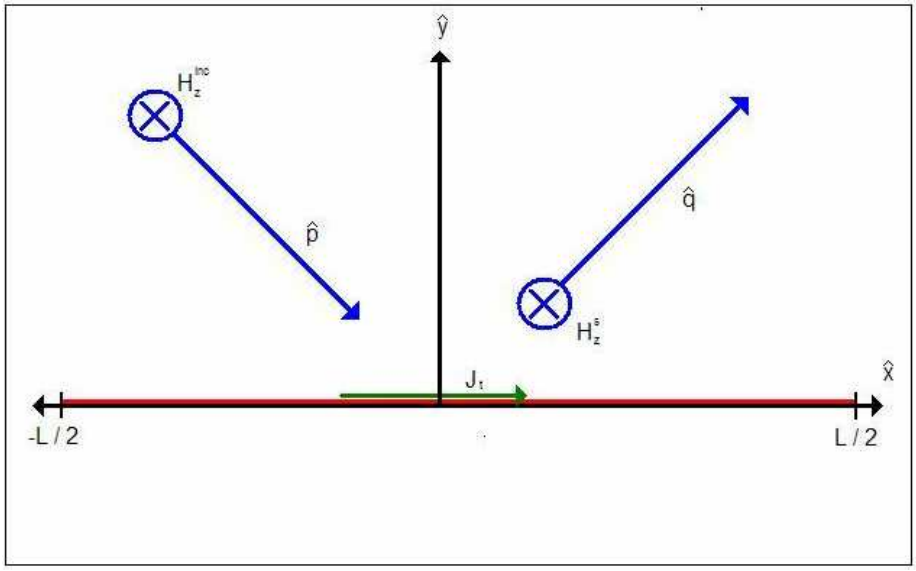

Figure 2.4 A diagram representing the reflection of a flat reflective plate. Both the incident and scattered Magnetic fields are polarized in the z direction (blue vector). A surface current (green) is induced on the surface (red).

Using the Law of Reflection, $\theta_i = \theta_r$, we can simplify this form by recognizing the relationship between **p** and **q**.

$$p_{\mathrm{X}} = k\cos(\theta) = q_{\mathrm{X}} \tag{2.15}$$

$$p_{\mathrm{Y}} = k\sin(\theta) = -q_{\mathrm{Y}} \tag{2.16}$$

For a flat plate, $y' \to 0$ and $\theta' \to 0$. Once these results are applied to 2.9, the field becomes possible to evaluate analytically.

$$H_z^{\mathrm{S}} = \frac{e^{i\left(k\rho+\frac{\pi}{4}\right)}}{2\sqrt{2\pi k\rho}} \left\{ -iksin(\phi) \int_c \left[e^{i(kcos(\theta)x+ksin(\theta)y)} + e^{i(kcos(\theta)x-ksin(\theta)_y y)} \right] \cdot \left[e^{-ikcos(\phi)x'} \right] ds' \right\} \tag{2.17}$$

$$= \frac{e^{i\left(k\rho+\frac{\pi}{4}\right)}}{2\sqrt{2\pi k\rho}} \left\{ -2iksin(\phi) \int_{-L/2}^{L/2} e^{ik(cos(\theta)-cos(\phi))x'} dx' \right\} \tag{2.18}$$

$$= \frac{e^{ik\rho}}{2\sqrt{2\pi k\rho}} \left\{ e^{\frac{-\pi}{4}} \frac{4sin(\phi)sin(\frac{k\delta L}{2})}{\delta} \right\}. \tag{2.19}$$

where $\delta = cos(\theta) - cos(\phi)$.

The program has only considered the angular dependance of the reflection, so the analytical solution for the reflection of a flat plate is

$$f(\theta) = 4e^{\frac{-i\pi}{4}} \frac{sin(\phi)sin(\frac{k\delta L}{2})}{\delta} \tag{2.20}$$

The comparisons between the analytical and the computed intensity for three different incident angles are shown in Figures 2.5, 2.6, and 2.7. The computed value is very accurate for reflected light near normal incidence. However, at 30° incidence, the difference becomes noticeable. At low incident angles, the effect of the circular endcaps begins to cause a greater effect on the system; this is likely the cause of the minor discrepancy. The agreement between the analytical and computed values for a flat plate is an important check on the accuracy of the program.

2.3.2 Cylinder Conductor

A second important validation for the code is the other extreme condition: a perfectly conducting cylinder. The circular end caps were added in order to correct for the derivatives in the MFIE. In order to ensure that the end caps have the correct effect on the field, the cylinder should give the expected result. When the length of the

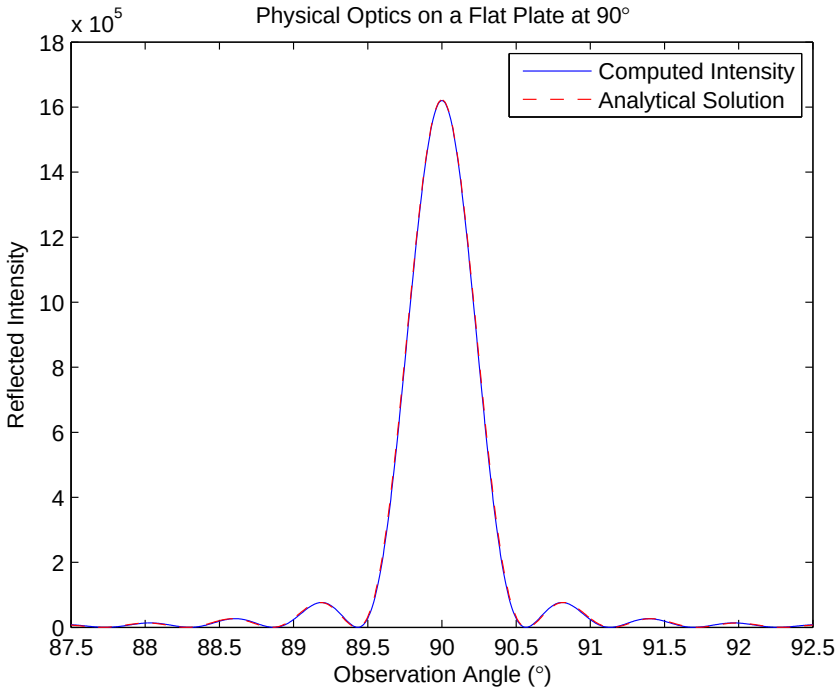

Figure 2.5 The scattered field calculated by the program for a flat surface with 90° incident light using the analytic solution (Equation 2.20).

scatterer is reduced to zero, the scatterer becomes a circle in two dimensions and projects as a cylinder in three. The intensity of the scattered field from a perfectly conducting infinite cylinder can be solved analytically and is represented using the Mie series. For the TE polarization, the angular dependance of the scattered field can be found using

$$f(\phi) = -4\sum_{n=0}^{\infty} \epsilon_n (-1)^n \frac{J_n'(ka)}{H_n^{(1)'}(ka)} cos(n\phi) \tag{2.21}$$

where $\epsilon_n = 1$ if $n = 0$, otherwise $\epsilon_n = 2$ [24].

To test the accuracy of the program, the length of the scatterer and the incident angle were set to zero and the intensity of the scattered field was sampled for all 360 °. The results are shown in Figures 2.8, 2.9, and 2.10. The results for the TE polarization were calculated with a scatterer thickness of 10λ, so it is essential that the effects due

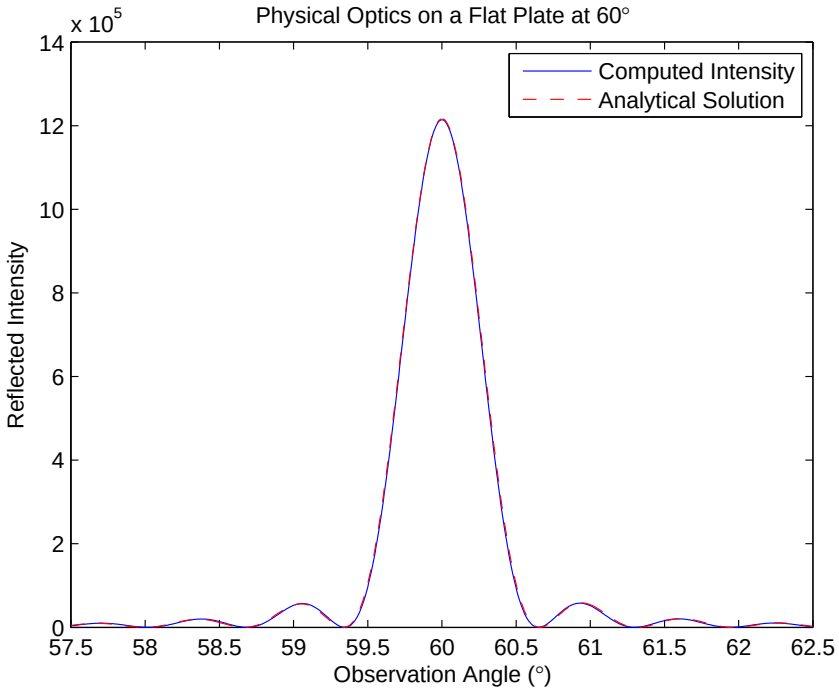

Figure 2.6 The scattered field calculated by the program for a flat surface with 60° incident light using the analytic solution (Equation 2.20).

to the circular endcaps match the expected value. For all three sample thicknesses, the calculated intensity very closely matches the analytical values predicted by the Mie series (Equation 2.21).

2.4 Sample Field

With the program matching the expected results for a cylinder and qualitative agreement with the scattered field expected for a flat surface, roughness can be added to the system to see its effects. Figure 2.11 shows a sample surface of length of 100λ, rms roughness height of 0.1λ, and surface knot separation of 10λ. Figure 2.12 shows the reflected field for this surface and the field from a smooth surface of the same size. A comparison of the two fields highlights some of the possible effects that roughness can have on reflected light. Most importantly the reflected light is diminished. The

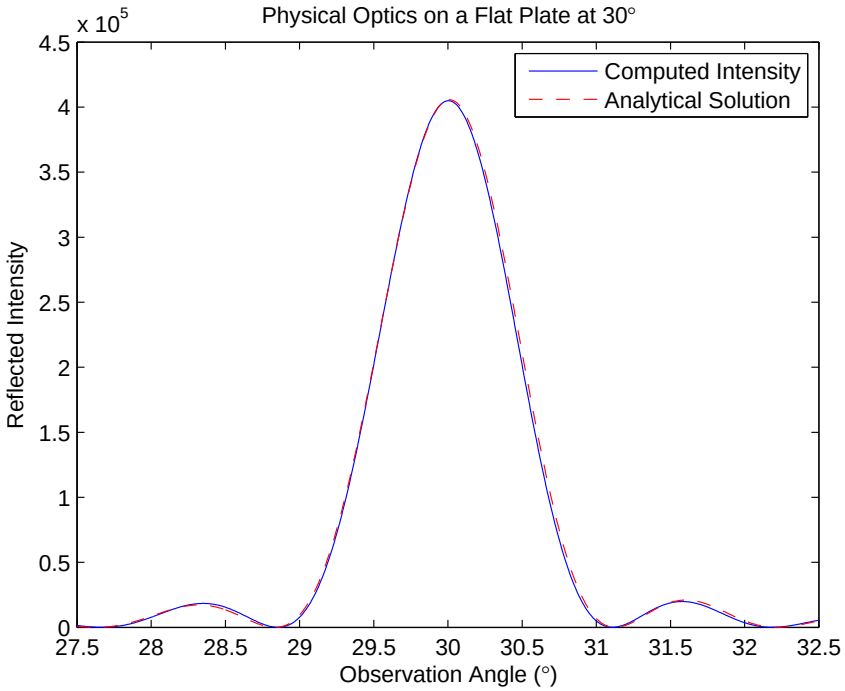

Figure 2.7 The scattered field calculated by the program for a flat surface with 30° incident light using the analytic solution (Equation 2.20).

primary maximum of the smooth field represents the light reflected at the incident angle. For a rough surface, the central peak is smaller because much of that light is scattered away from the central angle. This scattered light leads to secondary peaks much greater than those caused by diffraction. In addition, roughness can cause the central peak to shift away from the incident angle as shown below.

Because of the difference effects of roughness on the scattered field, there were many possible methods to define the attenuation. The approach we chose was to use the ratio of the maximum of the rough scattered field with respect to the maximum of the scattered field calculated from an equally sized smooth surface. This definition is both simple and realistic for the purpose of our application. In characterizing the indices of refraction for materials, only the light measured by the detector is considered. In our simulation, the light incident on the detector is the center of the

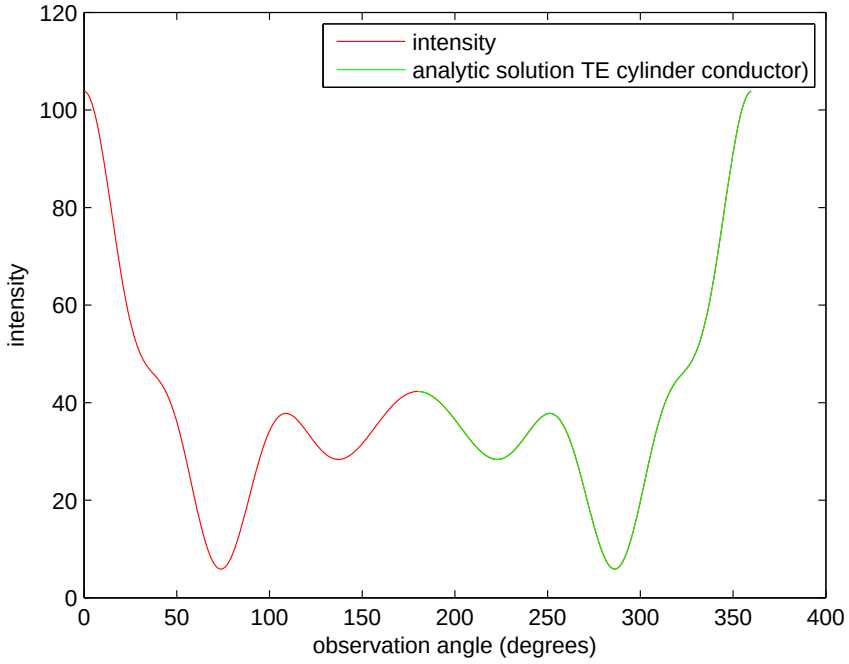

Figure 2.8 The scattered field calculated by the program and using the analytic solution for a cylinder of radius 1λ.

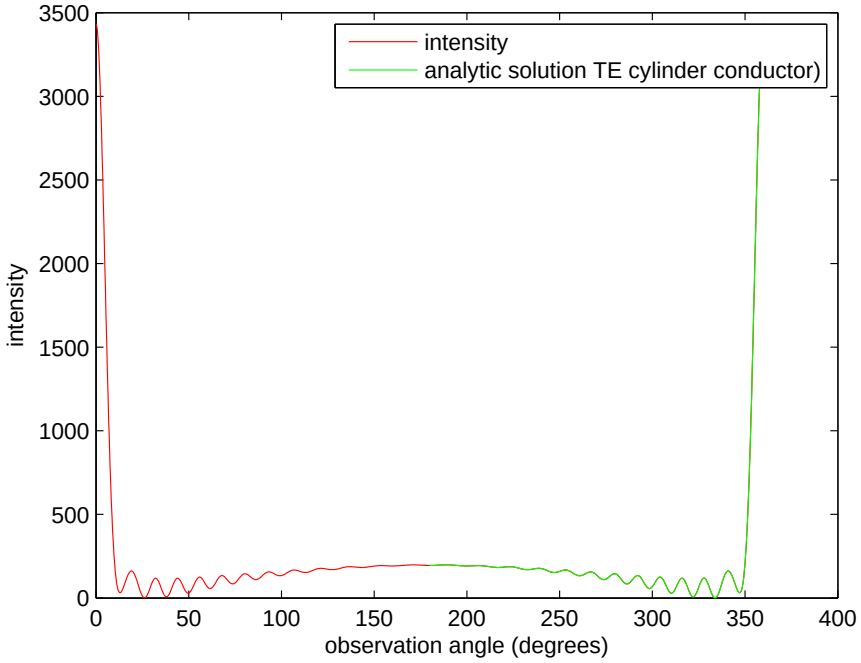

Figure 2.9 The scattered field calculated by the program and using the analytic solution for a cylinder of radius 5λ.

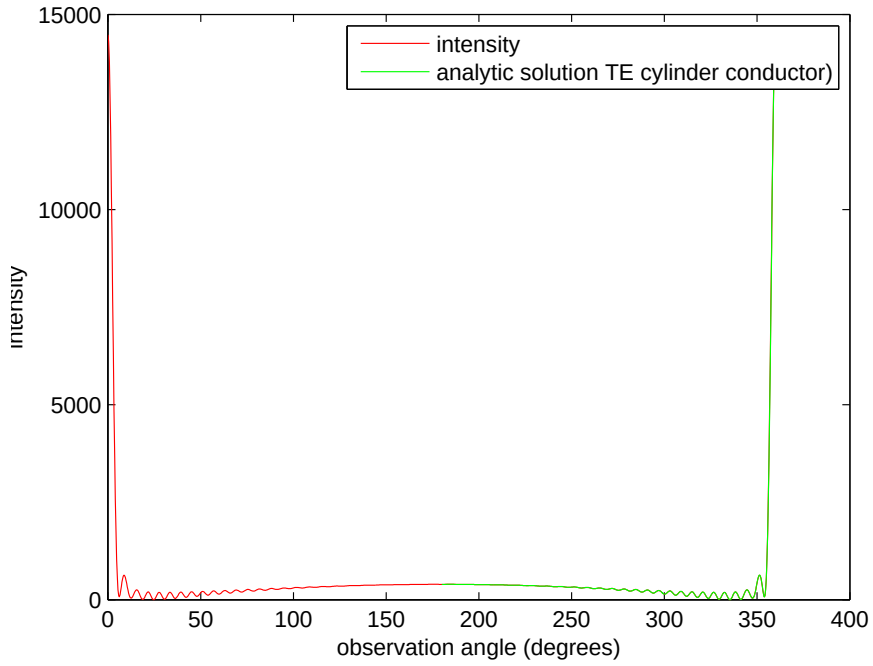

Figure 2.10 The scattered field calculated by the program and using the analytic solution for a cylinder of radius 10λ.

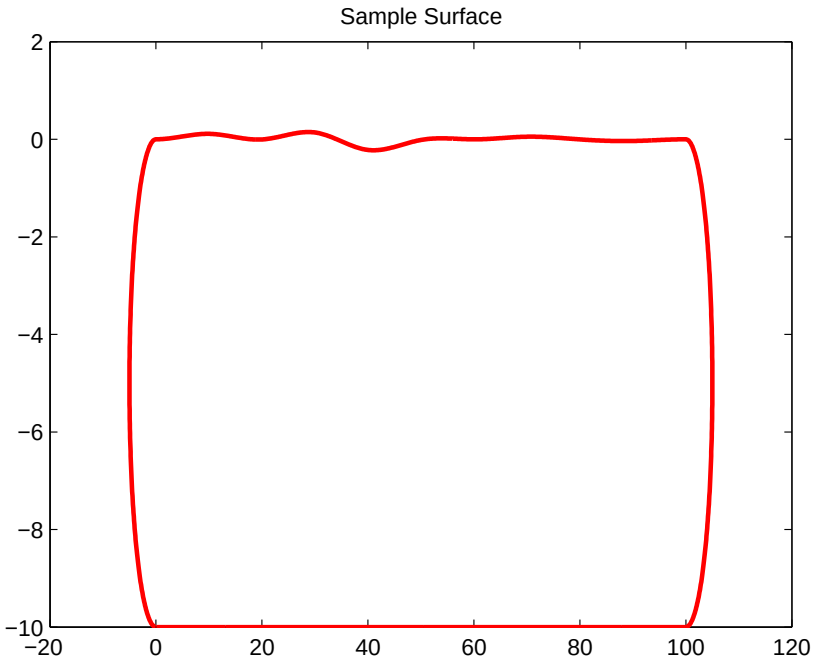

Figure 2.11 A randomly generated surface with length 100λ, thickness 10λ, spacing 10λ, and height 0.1λ.

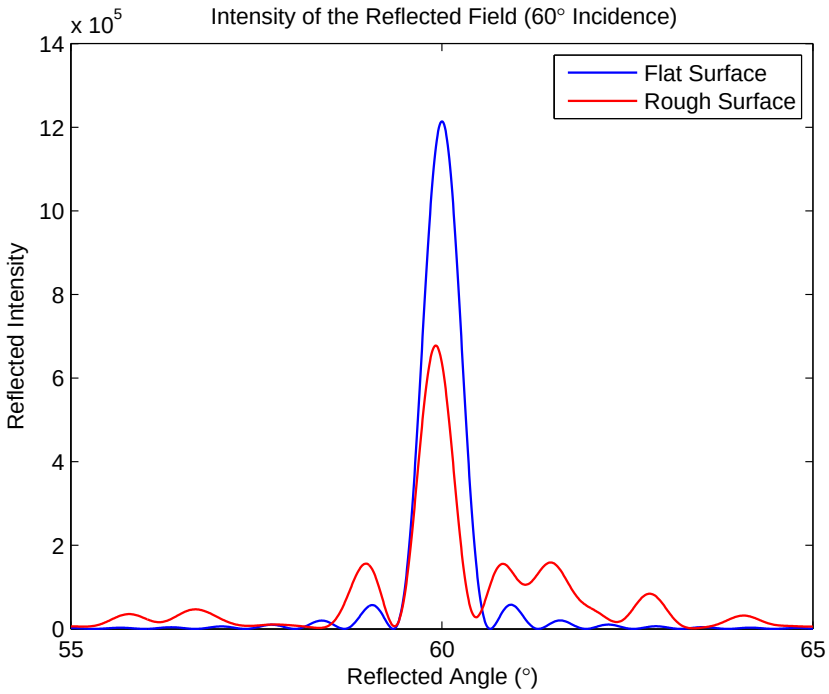

Figure 2.12 The intensity of the field scattered from the rough surface shown above with an incident angle of 30°. The field was calculated with 200 patches on the flat surface and 40 tpatches for the round endcaps.

primary peak along with 5° on either side. Calculating the ratio of the light in this region is similar to the method of measuring light in most experiments.

2.5 Computation

A useful correction model applies over a wide range of angles and roughness values. For this project, the surface was fixed at an 100λ length and a 10λ thickness. The incident angle varied from 15°-90° using 30 separate values. The roughness height varied from 0.05λ to 0.1λ with 20 different values. Finally, the knot separation varied from 0.5λ to 20λ using 40 values. Each combination of these three parameters was repeated for 100 random surfaces to provide a statistically significant sample size for a total of 2.4 million data points. In order to be computationally feasible, the program was run using the parallel processing capabilities of Marylou4. This is the supercomputer maintained by BYU. It is Dell 1955 Linux cluster composed of 630 nodes each with two 2.6 GHz dual-core processors. For more information, please see the support website (http://www.marylou.edu). The TM polarization sample took approximately 24 hours on 96 processors while the TE simulation took approximately 20 hours on 192 processors.

Chapter 3

Results

3.1 Attenuation Function

Originally applied due to an apparent similarity between two types of scattering, the DWF predicts that the attenuation due to roughness is a function of the qh factor. Because of this analogy, the form of our fitting model is motivated by (1.4). Isolating the correction function, f, in terms of the calculated attenuation yields

$$f(qh) = -ln(R) \tag{3.1}$$

The correction function f, was plotted with respect to qh for each value of the roughness knot separation. Figure 3.1 shows the results for a knot separation of 9λ as an example. The plot shows a clear correlation between the correction function and qh for both polarizations of light. There appears to be a similar relationship for both polarizations with this knot separation, but the actual value predicted by the DWF (shown in red) is inaccurate. The correlation implies that concept used in order to apply the DWF to roughness scattering is accurate. However, the inaccuracy shows that the exact form used by the DWF is not appropriate. This is likely due to the

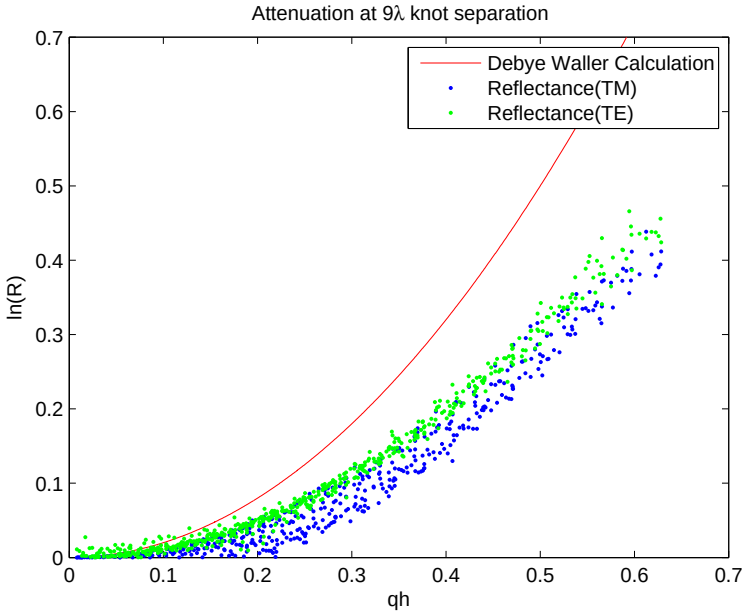

Figure 3.1 The attenuation function shows a clear correlation with the qh factor for both TE(green) and TM(blue). The function shows approximately the same form for both polarizations. For comparison, the DWF(red) prediction is also shown.

surface correlation between atoms on the surface that causes the gaussian distribution to break down.

A second issue with the DWF is that it does not take into consideration the knot separation of the surface. Our simulations showed a clear and significant relationship between the attenuation of the light and spatial frequency of the surface. In order to provide an accurate correction factor, this effect must be accounted for. One example of this is found at high spatial frequency roughness for TE. Corresponding to small knot separation, the size of the roughness approaches the wavelength of the incident light. To calculate the reflection of TE polarized light, we used the MFIE (2.8), which includes a derivative in the calculation. As the roughness frequency increases, the effect of this derivative begins to dominate and the correlation found at lower roughness frequencies becomes less obvious (see Figure 3.2). The plots for many of

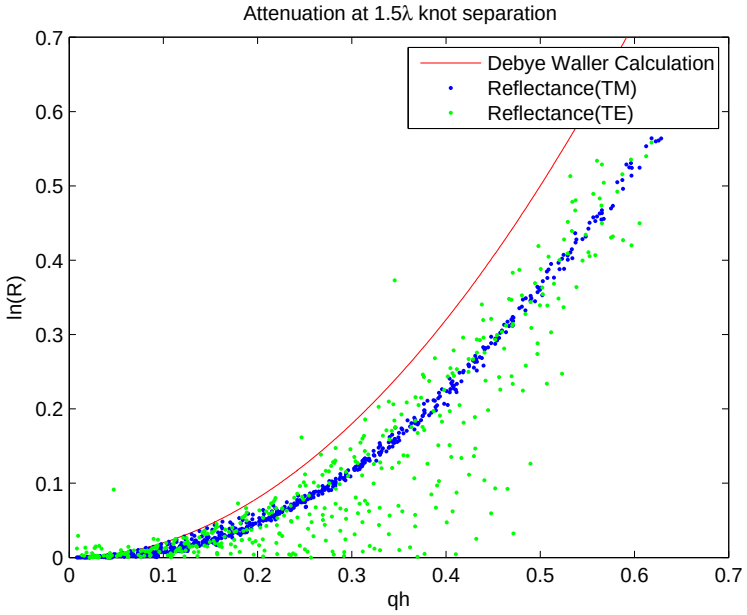

Figure 3.2 The attenuation function for high spatial frequency roughness(1.5λ knot separation). While the TM(blue) still shows a strong correlation with qh, the TE(green) polarized light's correlation is much less clear. Again the DWF(red) is shown for comparison.

the other knot separations are displayed in Appendix A.

3.2 Correction Coefficients

The original DWF predicts a quadratic correction function, but this form is unable to correctly model the attenuation. Due to the symmetry of the situation of the scattering problem, the constant term is expected to be zero. The correction function, f, was fit to a cubic polynomial form (3.2) with the constant set to zero.

$$f = \alpha(qh)^3 + \beta(qh)^2 + \gamma(qh) \tag{3.2}$$

In addition, the first two points in each data set were ignored in the fit. This is due to the effect of high frequency roughness on the correlation function. While both polarizations show a similar correlation with qh, there is a definite distinction

between the two. For TM polarization, the coefficients so a significant change in behavior in this regime. On the other hand, the TE polarization function has low correlation in this regime. As the roughness frequency increases, the derivative terms in the function become much more significant and the clear correlation breaks down.

The DWF predicts that the only significant term in the correction function in quadratic and that the coefficient is two. This stems from an approximation that the surface roughness is gaussian in distribution. Both TE and TM polarized light show a significant quadratic term that is nearly two. This implies that the underlying assumptions are fairly accurate. The large error bars in the high frequency results of the TE quadratic term (see Figure 3.3) is mainly due to the greater error inherent in derivatives at high frequencies and the error in the coefficient drops as the frequency decrease. The TM quadratic term (Figure 3.4), without the derivative, has strong correlation for all frequencies.

The cubic (Figures 3.5 and 3.6) and linear (Figures 3.7 and 3.8) terms are expected to be zero in the DWF. While the cubic term for the TE polarization was consistent with zero (zero is entirely within the confidence bounds), both terms for TM polarization and the linear TE term are significant. Because the surface can not be fully described with a gaussian distribution, the DWF is incorrect. The functional forms for correction coefficients that account for the true nature of surfaces must take into account not only the surface roughness heights and incident angles, as DWF does, but also the spatial frequency and polarization of the light. For the correction coefficients see the following table.

Correction Coefficient Functions		
	TE Polarization	TM Polarization
$\alpha(\sigma)$		$(-0.084 \pm 0.003)\sigma + 0.21 \pm 0.01$
$\beta(\sigma)$	$(-0.025 \pm 0.007)\sigma + 1.82 \pm 0.09$	$(0.027 \pm 0.001)\sigma + 1.50 \pm .02$
$\gamma(\sigma)$	$(0.007 \pm 0.002)\sigma - (0.20 \pm 0.02)$	$(-0.011 \pm 0.001)\sigma - 0.08 \pm 0.01$

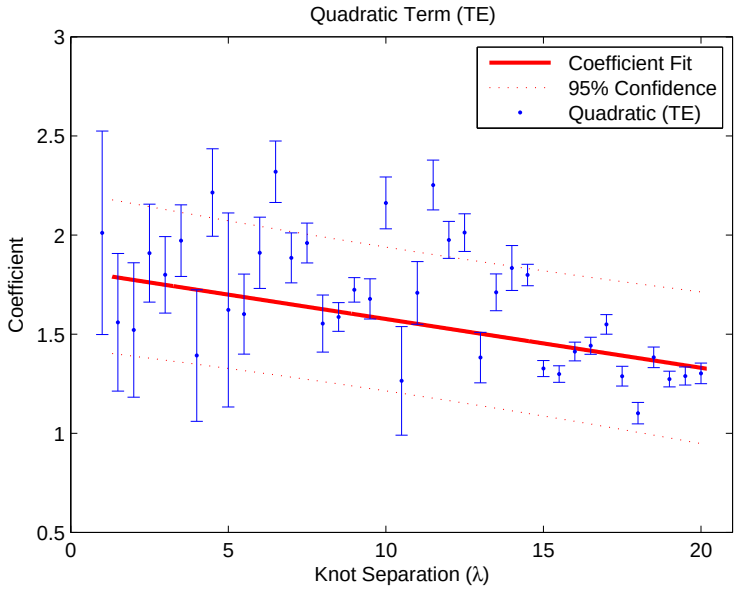

Figure 3.3 The quadratic coefficient for the correction function, f, with TE polarization.

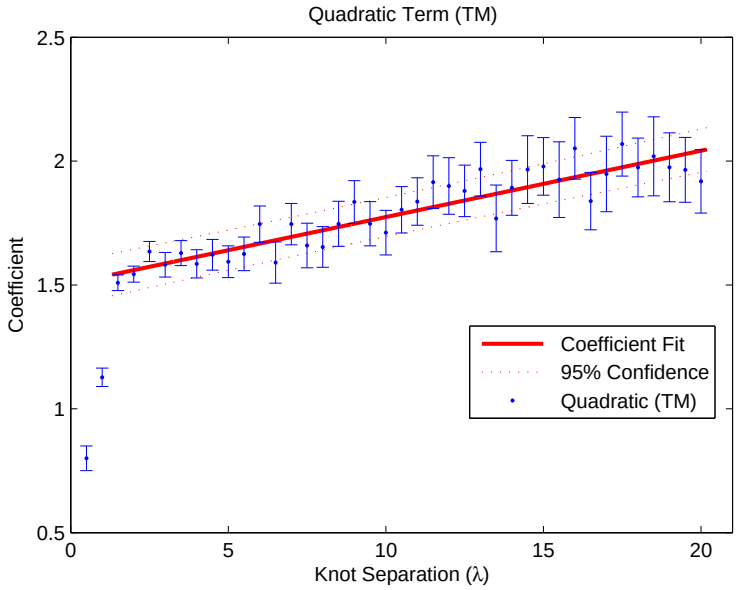

Figure 3.4 The quadratic coefficient for the correction function, f, with TM polarization.

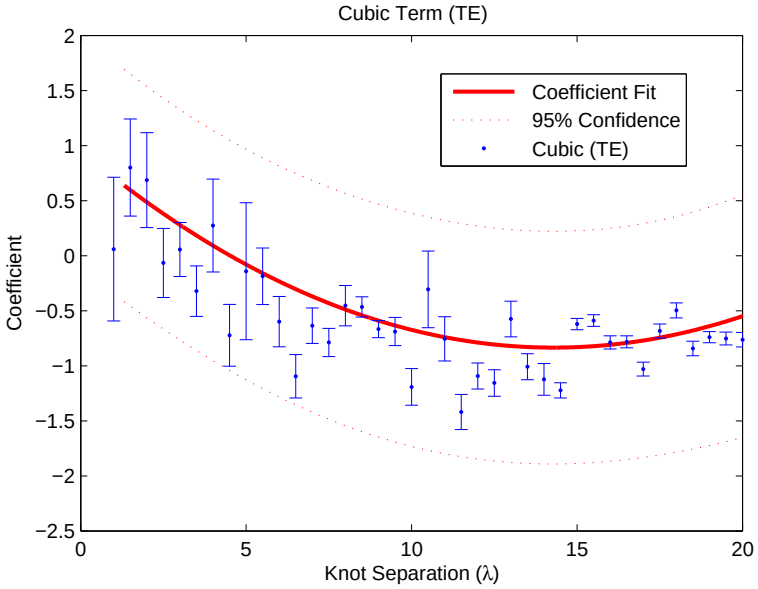

Figure 3.5 The cubic coefficient for the correction function, f, with TE polarization.

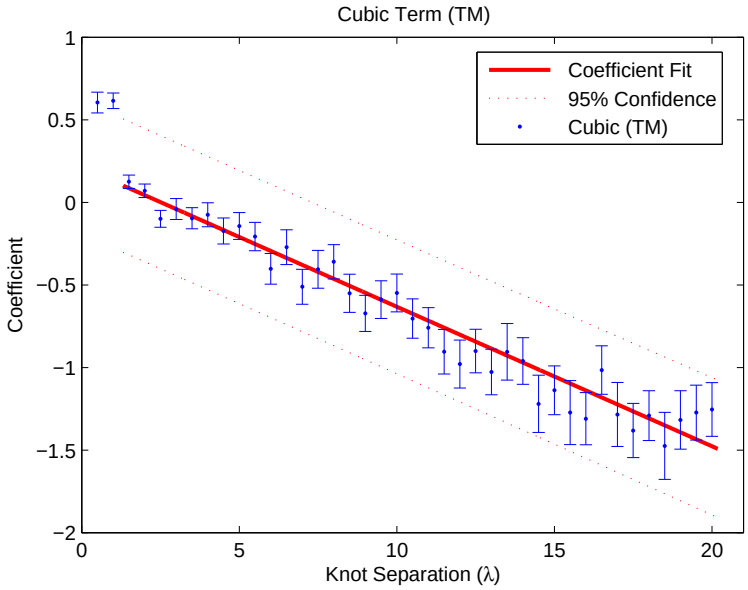

Figure 3.6 The cubic coefficient for the correction function, f, with TM polarization.

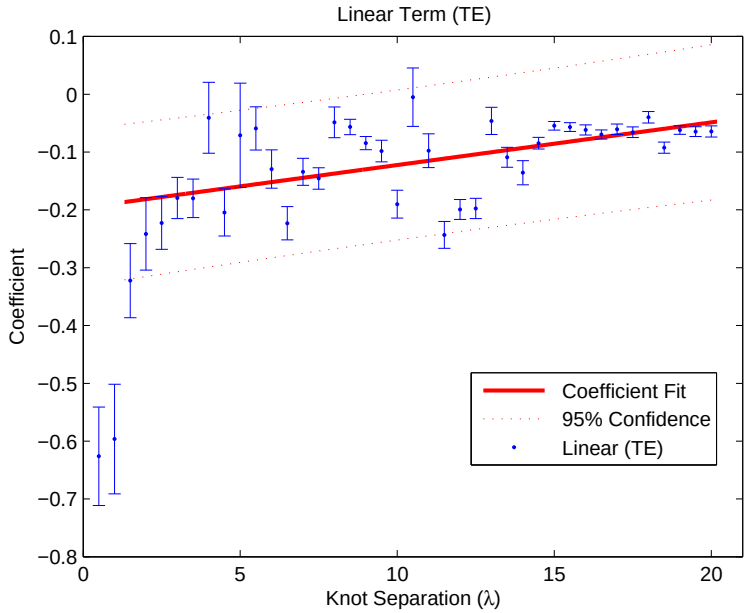

Figure 3.7 The linear coefficient for the correction function, f, with TE polarization.

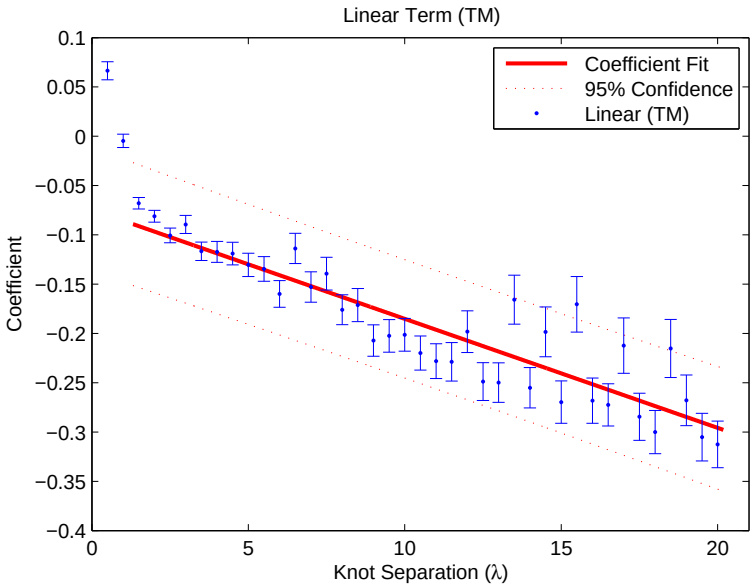

Figure 3.8 The linear coefficient for the correction function, f, with TM polarization.

Bibliography

[1] D. D. Allred, R. S. Turley, and M. B. Squires, "Dual-function EUV multilayer mirrors for the IMAGE mission," In *EUV, X-Ray, and Neutron Optics and Sources*, C. A. MacDonald, K. A. Goldberg, J. R. Maldonado, H. H. Chen-Mayer, and S. P. Vernon, eds.,**3767,** 280–287 (Bellingham, WA, 1999).

[2] T. E. Jewell, J. M. Rodgers, and K. P. Thompson, "Reflective systems design study for soft x-ray projection lithography," J. Vac. Sci. Technol. B **8,** 1519–1523 (1990).

[3] A. M. Hawryluk and L. G. Seppala, "Soft x-ray projection lithography using an x-ray reduction camera," J. Vac. Sci. Technol. B **6,** 2162–2166 (1988).

[4] D. G. Stearns, "X-ray scattering from interfacial roughness in multilayer structures," J. Appl. Phys. **71,** 4286–4298 (1992).

[5] N. Farnsworth, "Thorium-based Mirrors in the Extreme-Ultraviolet,", 2005.

[6] E. Hecht, *Optics*, 3 ed. (Addison-Wesley, Reading, MA, 1998).

[7] D. G. Stearns and E. M. Gullikson, "Nonspecular scattering from extreme ultraviolet multilayer coatings," Physica B **283,** 84–91 (2000).

[8] D. E. Savage and et al., "Determination of roughness correlations in multilayer films for x-ray mirrors," J. Appl. Phys. **69,** 1411–1424 (1991).

[9] P. Debye, "Interference of Rontgen Rays and Heat Motion," J. Appl. Phys. 15 (1913).

[10] V. Holy, J. Kubena, and I. Ohlidal, "X-ray reflection from rough layered systems," Phys. Rev. B **47,** 896–903 (1993).

[11] N. C. Bruce, "Scattering of light from surfaces with one-dimensional structure calculated by the ray-tracing method," J. Opt. Soc. Am. A **14,** 1850–1858 (1997).

[12] K. F. Warnick and D. V. Arnold, "Generalization of the geometrical-optics scattering limit for a rough conducting surface," J. Opt. Soc. Am. A **15,** 2355–2361 (1998).

[13] I. Sassi and M. S. Sifaoui, "Comparison of geometric optics approximation and integral method for reflection and transmission from microgeometrical dielectric surfaces," J. Opt. Soc. Am. A **24,** 451–462 (2007).

[14] N. Pinel and C. Bourlier, "Scattering from very rough layers under the geometric optics approximation: further investigation," J. Opt. Soc. Am. A **25,** 1293–1306 (2008).

[15] X. Nie, L. W. Li, and N. Yuan, "Precorrected-FFT Algorithm for Solving Combined Field Integral Equations in Electromagnetic Scattering," J. Electromagn. Waves Appl. **16,** 1171–1187 (2002).

[16] R. J. Adams, "Combined Field Integral Equation Formulations for Electromagnetic Scattering from Convex Geometries," IEEE Trans. Antennas Propag. **52,** 1294–1303 (2004).

[17] R. J. Adams and N. J. Champagne, "A Numerical Implementation of a Modified Form of the Electric Field Integral Equation," IEEE Trans. Antennas Propag. **52,** 2262–2266 (2004).

[18] A. Colliander and P. Yla-Oijala, "Electromagnetic Scattering from Rough Surfaces Using Single Integral Equation and Adaptive Integral Method," IEEE Trans. Antennas Propag. **55,** 3639–3646 (2007).

[19] L. F. Canino and et al., "Numerical Solution of the Helmholtz Equation in 2D and 3D Using a High-Order Nystrom Discretization," J. Comp. Phys. **146,** 627–663 (1998).

[20] A. A. Nosich and et al., "Numerical analysis and synthesis of 2D quasi-optical reflectors and beam waveguides based on an integral-equation approach with Nystroms discretization," J. Opt. Soc. Am. A **24,** 2831–2836 (2007).

[21] J. Johnson, Master's thesis, Brigham Young University, 2006.

[22] E. Martin, "Surface-Roughness Corrections to Extreme Ultraviolet Thin-Film Reflectance Measurements,", 2007.

[23] A. F. Peterson and et al., *Computational Methods for Electromagnetics* (IEEE Press, New York, 1998).

[24] J. Bowman, T. B. A. Senior, and P. L. E. Uslenghi, *Electromagnetic and Acoustic Scattering by Simple Shapes* (North-Holland Publishing Co., Amsterdam, 1969).

Appendix A

Program Results

This appendix contains the attenuation functions $f = -ln(R)$ for all the different spatial frequency roughness parameters tested. The higher frequency (small knot separation) samples show very little correlation with qh for TE polarized light while TM shows a clear correlation. This is likely due to the derivative found in solving the MFIE. For high spatial frequency roughness, this effect becomes more pronounced and the results are more unpredictable. However, for lower frequency roughness (large knot separation) the correlation for qh is clear for both polarizations. The TE has a clearer correlation for this regime.

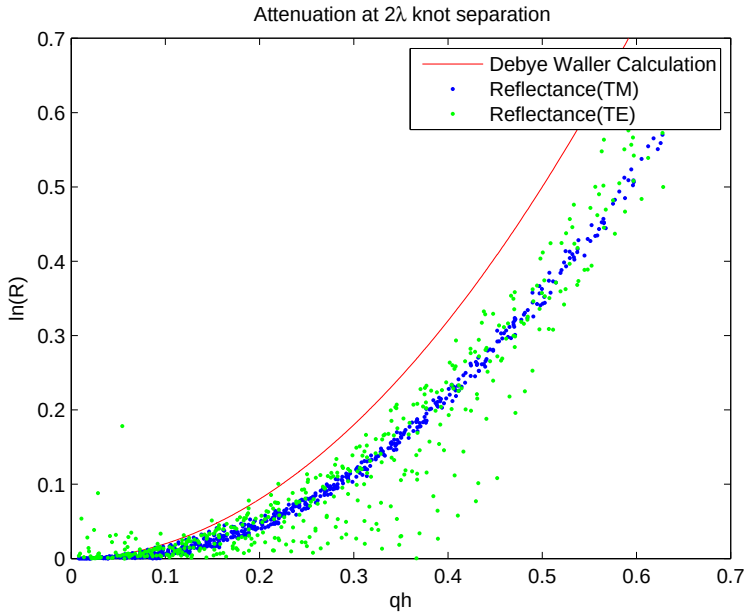

Figure A.1

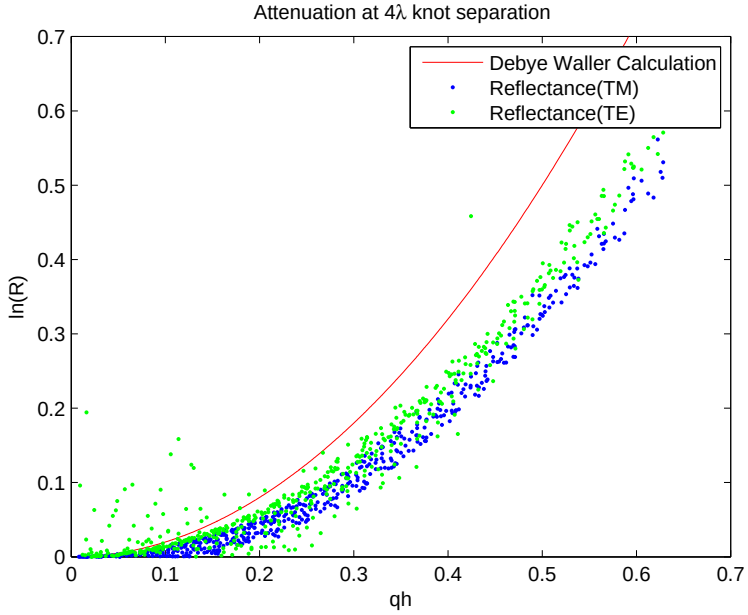

Figure A.2

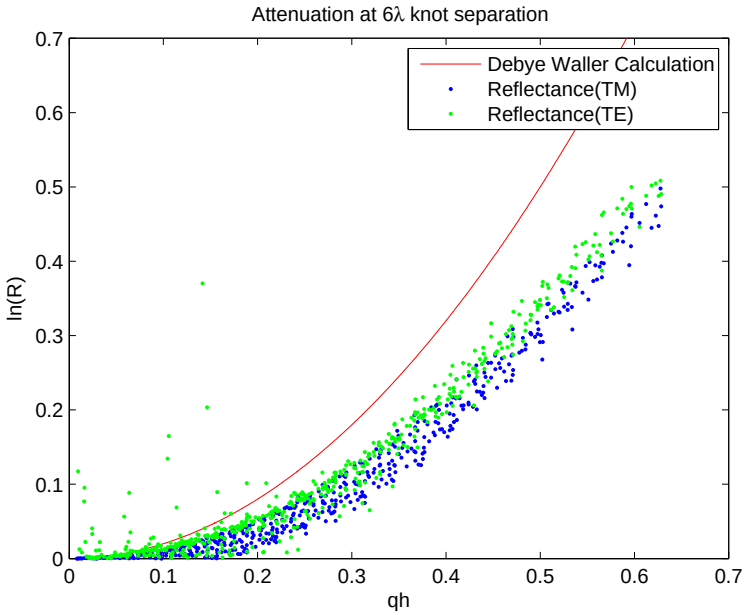

Figure A.3

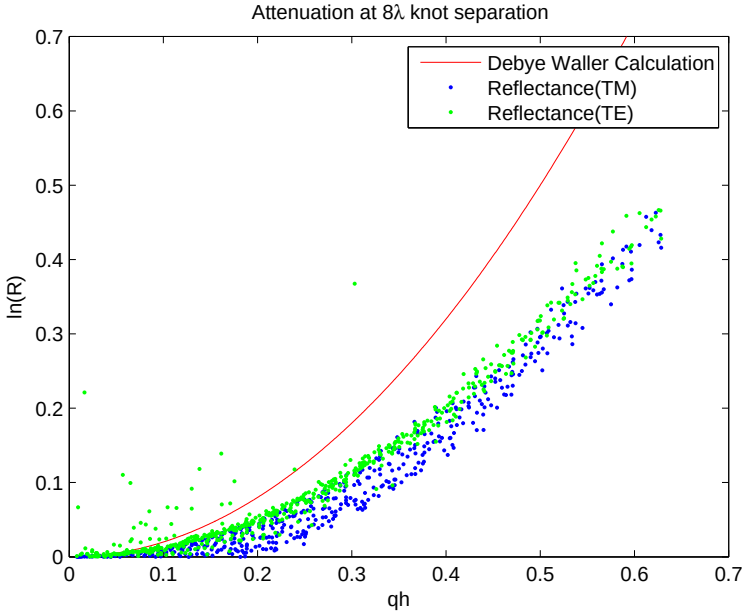

Figure A.4

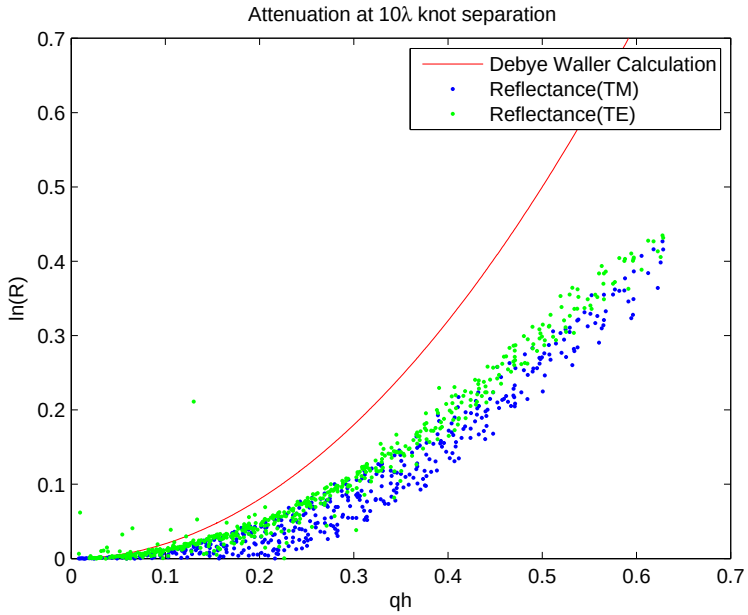

Figure A.5

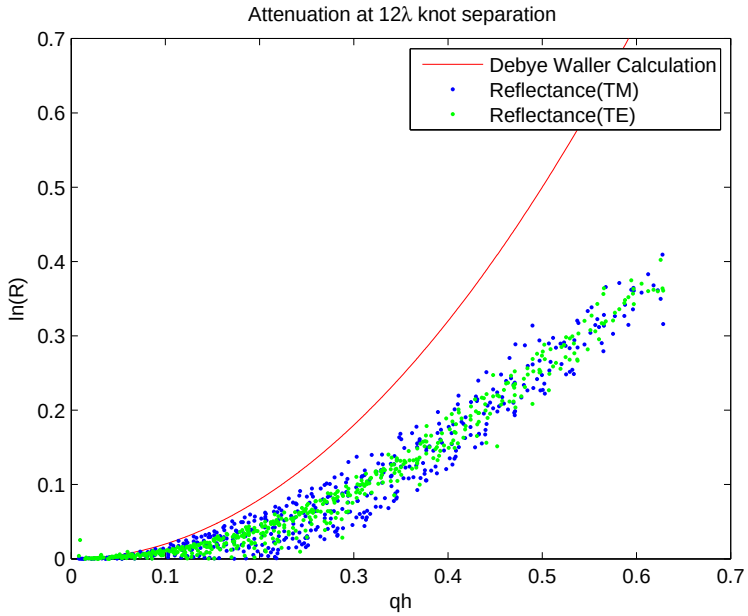

Figure A.6

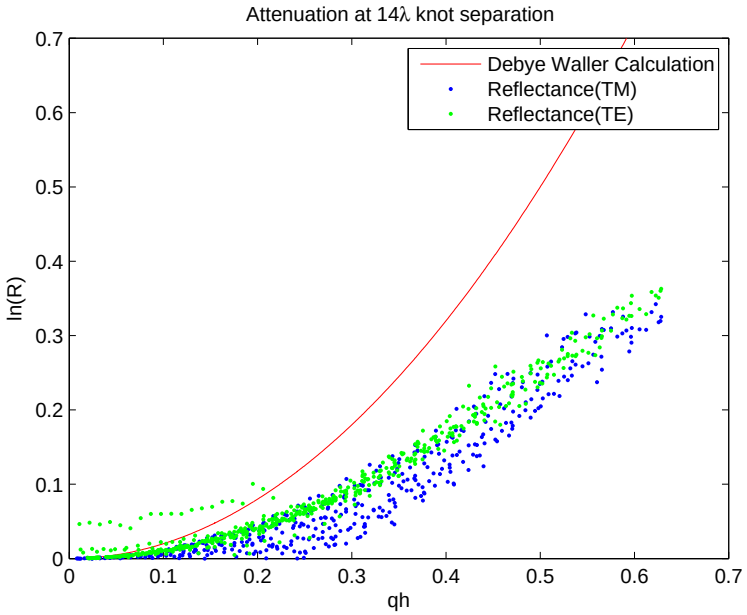

Figure A.7

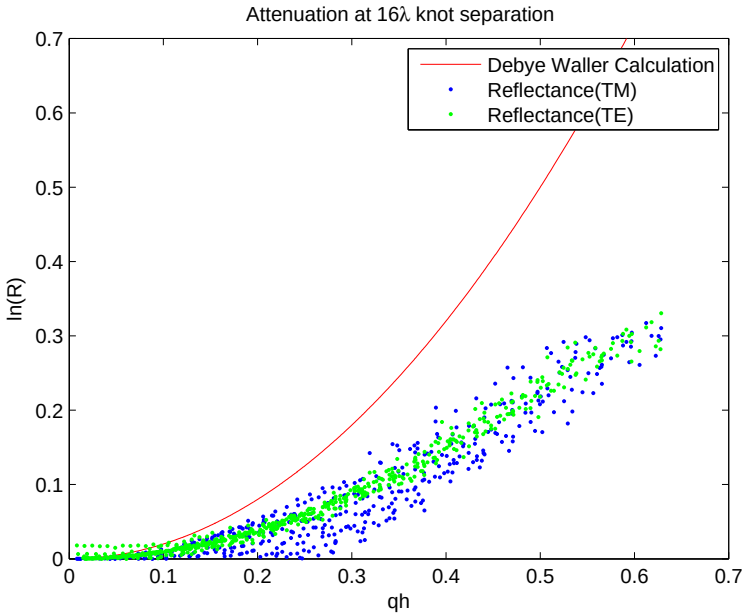

Figure A.8

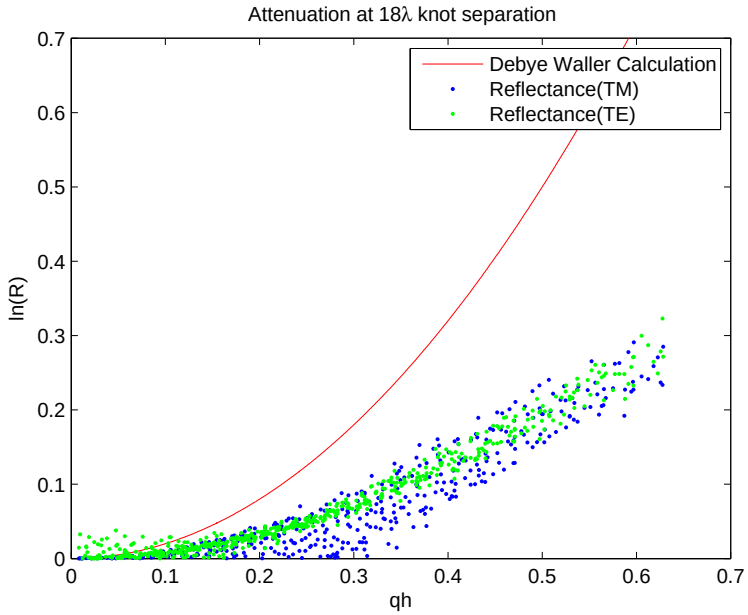

Figure A.9

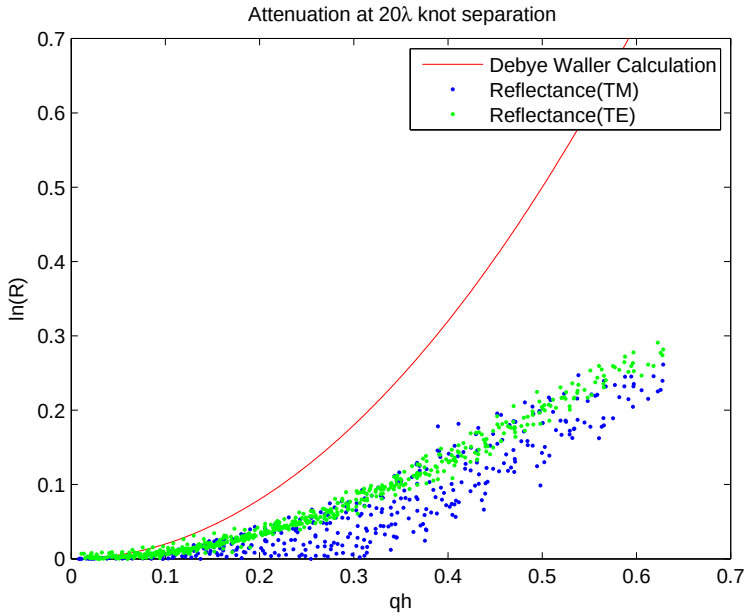

Figure A.10

Appendix B

Fortran Code

B.1 ECS.f90

This is the master slave process that uses RoughTE.f90 to calculate the scattered field for a wide range of incident parameters. The reflection from a flat surface is first calculated and the attenuation, R, is the ratio of the maximum of the ideal scattered field to the maximum of the rough scattered field.

```
!*****************************************************************************
!
!  PROGRAM: ecs
!
!  PURPOSE:  Entry point for perfect conductor p-pol scattering program
!  AUTHOR: Todd Doughty
!  DATE: June 30, 2007
!
!*****************************************************************************

program ecs
  use mpi
  use quad, only: qd_init,qd_free
  use params, only: init, k2, knots, angles, heights, samples, pprint
  implicit none
```

```
integer::err, rank, size, i, j, k, proc
integer::status(MPI_Status_size), count
real,allocatable::data(:),rdata(:,:,:,:)
integer, parameter::dtag=55, ctag=56, dest=0
character(len=9)::fname
integer, allocatable::seed(:)
integer,parameter::pDataSize=3
real::pData(pDataSize)

call MPI_Init(err)
if(err /= MPI_Success)then
   write(6,'("Error initializing MPI")')
   stop
end if
call init  ! initialize parameters
allocate(data(samples+3))
call MPI_Comm_Rank(MPI_Comm_World, rank, err)
if(rank == 0) then
   ! master process
   write(6,'(a/)') 'Computing reflection from perfect conductor (TM)'
   call pprint  ! print the parameters
   call MPI_Comm_Size(MPI_Comm_World, size, err)
   write(fname,'("ecs",i2.2,".txt")')size
   ! I'll assume that I have enough to do to keep every processor
   ! busy.  On marylou4, there is about a 1.3 ms time for interprocess
   ! control communication.
   allocate(rdata(knots,angles,heights,samples))
   proc=1
   do i=1,knots
      pData(1)=i;
      do j=1,angles
         pData(2)=j
         do k=1,heights
            pData(3)=k
            if(proc<size)then
               ! more initial processors are still available
               call MPI_Send ( pData, pDataSize, MPI_Real, proc, ctag,&
                    MPI_Comm_World, err)
               proc=proc+1
```

```
            else
               ! Wait for a processor to finish and use it again
               call MPI_Recv( data, samples+3, MPI_Real, MPI_Any_source,&
                    dtag, MPI_Comm_World, status, err)
               rdata(data(1), data(2), data(3), :)=data(4:)
               call MPI_Send ( pData, pDataSize, MPI_Real,&
                    status(MPI_Source), ctag, MPI_Comm_World, err)
            end if
         end do
      end do
   end do
   ! Terminate slave processes
   pData(1)=-1.0
   do i=1,proc-1
      call MPI_Recv( data, samples+3, MPI_Real, MPI_Any_source,&
           dtag, MPI_Comm_World, status, err)
      rdata(data(1), data(2), data(3), :)=data(4:)
      call MPI_Send ( pData, pDataSize, MPI_Real, i, ctag,&
           MPI_Comm_World, err)
   end do
   ! Save the data in a file
   open(unit=7,name=fname,status='replace')
   write(7,fmt='(f12.8)')rdata(:,:,:,:)
   close(unit=7)
   deallocate(rdata)
else
   ! slave process
   ! Init random number for this process
   call random_seed(size=k)
   allocate(seed(k))
   call random_seed(get=seed)
   if(k.GT.1)then
      ! Create a random seed related to the clock time
      call system_clock(seed(1))
      seed(2)=rank*123
   else
      call system_clock(seed(1))
      seed(1)=seed(1)+123*rank
   end if
```

```
      call random_seed(put=seed)
      deallocate(seed)

      ! One time initialization calls
      call qd_init

      ! Multiple computations as driven by master processor
      do
         ! Get the call parameters from master process
         call MPI_Recv(pData, pDataSize, MPI_Real, MPI_Any_source,&
              ctag, MPI_Comm_World, status, err)
         if(pData(1)==-1.0)exit  ! termination signal
         call refl(pData(1), pData(2), pData(3), data(4:))
         ! copy call parameters to output
         data(1:3)=pData
         ! send back the data
         call MPI_Send (data,samples+3,MPI_Real,dest,dtag,MPI_Comm_World,err)
      end do

      call qd_free
   end if
   deallocate(data)
   call MPI_Finalize(err)

contains

   subroutine refl(isep, iiangle, iheight, data)
     use RoughTE, only: Efield, NyFill, NyClean
     use params, only: k2, thpoints
     use timer, only:tm_mark, tm_summary
     implicit none
     real,intent(in)::isep,iiangle,iheight
     real(kind=k2)::sep, iangle, height
     real(kind=k2),allocatable::oangles(:)
     complex(kind=k2),allocatable::field(:)
     integer::seed1, seed2, i, j, k, element
     real(kind=k2):: peak, speak
     real(kind=k2)::xbar,xsqbar
     real(kind=k2)::stdev, sigma
```

```
      real,intent(out)::data(:)

      sep=isep*(20.0/knots)  !knot separation
      iangle=15.0+(iiangle-1)*(75.0/(angles-1)) !incident angle
      height=iheight*(0.1/heights)
      write(6,*)'sep=',sep
      write(6,*)'iangle=',iangle
      write(6,*)'height=',height
      allocate(field(thpoints),oangles(thpoints))
      sigma=0.0
      call tm_mark('start')
      call NyFill(sigma, sep)
      call Efield(iangle, field, oangles)
      call NyClean
      speak=maxval(abs(field))
      call tm_mark('end')
      call tm_summary
      do i=1,samples
         ! Use a different surface at each angle.  This is
         ! undoubtedly slower, but it will offer better statistics
         call NyFill(height, sep)
         call Efield(iangle, field, oangles)
         call NyClean
         peak=maxval(abs(field))
         data(i)=peak/speak
      end do ! loop over samples
      deallocate(field, oangles)
    end subroutine refl

  end program ecs
```

B.2 RoughTE.f90

The main work of the program is done in this module. The subroutine NyFill fills the Nystrom matrix which uses the MFIE to relate the incident field to the surface

currents. In subroutine Efield, the matrix is inverted to solve for the surface currents for a given incident field. Those currents are than used to solve for the scattered field.

```
Module RoughTE
  use params, only:k2
  implicit none
  private
  public NyFill, Efield, NyClean
  complex(kind=k2),allocatable::A(:,:)
contains

  subroutine NyClean
    use fullsurface, only:surf_clean
    implicit none
    if(allocated(A)) deallocate(A)
    call surf_clean
  end subroutine NyClean

  subroutine NyFill(sigma, spacing)
    use params, only:k2, patches, tpatches, pi, i, zero, len
    use fullsurface, only:surf_setup, ct, cb, xp, pl, dx, dtheta, r, x, tl, theta
    use bessel, only: h11
    use timer, only: tm_mark
    implicit none
    real(kind=k2), intent(in):: sigma, spacing
    real(kind=k2)::singpt,offset
    integer::j,l,pv,pvl,m,pvn,pvm,n
    complex(kind=k2)::W0,W1,W2,W3
    real(kind=k2)::C1,C2,C1t,C2t

    call surf_setup(sigma,spacing)
    call tm_mark('surface setup')

    C1 = 13*dx/12
    C2 = 11*dx/12
    C1t = 13*dtheta*r/12
    C2t = 11*dtheta*r/12
```

```
    allocate(A((patches+tpatches)*8,(patches+tpatches)*8))



! Bottom Surface
    if (len /= 0) then
      do j = 1,2*(patches+tpatches)
        pv = 4*(j-1)
        do l = 1,patches
          pvl = 4*(l-1)
          offset = (l-1)*pl
          if(j==l) then
            do m = 1,4
              pvm = pv + m
              singpt = x(m)
              W0 = cartTE(zero,pl,singpt,0,cb,offset,xp);
              W1 = cartTE(zero,pl,singpt,1,cb,offset,xp)/dx;
              W2 = cartTE(zero,pl,singpt,2,cb,offset,xp)/dx**2;
              W3 = cartTE(zero,pl,singpt,3,cb,offset,xp)/dx**3;
              A(pvm,pv+1) = (13.125*W0-17.75*W1+7.5*W2-W3)/6;
              A(pvm,pv+2) = (-4.375*W0+11.75*W1-6.5*W2+W3)/2;
              A(pvm,pv+3) = (2.625*W0-7.75*W1+5.5*W2-W3)/2;
              A(pvm,pv+4) = (-1.875*W0+5.75*W1-4.5*W2+W3)/6;
              A(pvm,pvm) = A(pvm,pvm)-0.5
            end do !m
          else
            do n=1,4
              pvn = pv+n
              A(pvn,pvl+1) = C1*flatdiff(pvn,pvl+1)
              A(pvn,pvl+2) = C2*flatdiff(pvn,pvl+2)
              A(pvn,pvl+3) = C2*flatdiff(pvn,pvl+3)
              A(pvn,pvl+4) = C1*flatdiff(pvn,pvl+4)
            end do !n
          end if
        end do !l
      end do !j
    end if

! Right Surface
    do j = 1,2*(patches+tpatches)
```

```
        pv = 4*(j-1)
        do l = patches+1,patches+tpatches
          pvl = 4*(l-1)
          offset = (l-patches-1)*tl-pi/2;
          if(j==l) then
            do m = 1,4
              pvm = pv + m
              singpt = theta(4*patches+m)+pi/2
              W0 = polTE(zero,tl,singpt,0,r)
              W1 = polTE(zero,tl,singpt,1,r)/dtheta
              W2 = polTE(zero,tl,singpt,2,r)/dtheta**2
              W3 = polTE(zero,tl,singpt,3,r)/dtheta**3
              A(pvm,pv+1) = -(13.125*W0-17.75*W1+7.5*W2-W3)/6
              A(pvm,pv+2) = -(-4.375*W0+11.75*W1-6.5*W2+W3)/2
              A(pvm,pv+3) = -(2.625*W0-7.75*W1+5.5*W2-W3)/2
              A(pvm,pv+4) = -(-1.875*W0+5.75*W1-4.5*W2+W3)/6
              A(pvm,pvm) = A(pvm,pvm)-0.5
            end do !m
          else
            do n = 1,4
              pvn = pv + n
              A(pvn,pvl+1) = C1t*rounddiff(pvn,pvl+1)
              A(pvn,pvl+2) = C2t*rounddiff(pvn,pvl+2)
              A(pvn,pvl+3) = C2t*rounddiff(pvn,pvl+3)
              A(pvn,pvl+4) = C1t*rounddiff(pvn,pvl+4)
            end do !n
          end if
        end do !l
      end do !j

  ! Top Surface
      if (len /= 0) then
        do j = 1,2*(patches+tpatches)
          pv = 4*(j-1)
          do l = patches+tpatches+1,2*patches+tpatches
            pvl = 4*(l-1)
            offset = (2*patches+tpatches-l)*pl
            if(j==l) then
              do m = 1,4
```

```
            pvm = pv + m
            singpt = x(5-m)
            W0 = cartTE(zero,pl,singpt,0,ct,offset,xp)
            W1 = cartTE(zero,pl,singpt,1,ct,offset,xp)/dx
            W2 = cartTE(zero,pl,singpt,2,ct,offset,xp)/dx**2
            W3 = cartTE(zero,pl,singpt,3,ct,offset,xp)/dx**3
            A(pvm,pv+1) = (13.125*W0-17.75*W1+7.5*W2-W3)/6
            A(pvm,pv+2) = (-4.375*W0+11.75*W1-6.5*W2+W3)/2
            A(pvm,pv+3) = (2.625*W0-7.75*W1+5.5*W2-W3)/2
            A(pvm,pv+4) = (-1.875*W0+5.75*W1-4.5*W2+W3)/6
            A(pvm,pvm) = A(pvm,pvm)-0.5
          end do
        else
          do n = 1,4
            pvn = pv + n
            A(pvn,pvl+1) =-C1*flatdiff(pvn,pvl+1)
            A(pvn,pvl+2) =-C2*flatdiff(pvn,pvl+2)
            A(pvn,pvl+3) =-C2*flatdiff(pvn,pvl+3)
            A(pvn,pvl+4) =-C1*flatdiff(pvn,pvl+4)
          end do !n
        end if
      end do !l
    end do !j
  end if

! Left Surface
  do j = 1,2*(patches+tpatches)
    pv = 4*(j-1)
      do l = 2*patches+tpatches+1,2*(patches+tpatches)
        pvl = 4*(l-1)
        offset = (l-2*patches-tpatches-1)*tl+pi/2
        if(j==l) then
          do m = 1,4
            pvm = pv + m
            singpt = theta(4*patches+m)+pi/2
            W0 = polTE(zero,tl,singpt,0,r)
            W1 = polTE(zero,tl,singpt,1,r)/dtheta;
            W2 = polTE(zero,tl,singpt,2,r)/dtheta**2;
            W3 = polTE(zero,tl,singpt,3,r)/dtheta**3;
```

```
            A(pvm,pv+1) = -(13.125*W0-17.75*W1+7.5*W2-W3)/6;
            A(pvm,pv+2) = -(-4.375*W0+11.75*W1-6.5*W2+W3)/2;
            A(pvm,pv+3) = -(2.625*W0-7.75*W1+5.5*W2-W3)/2;
            A(pvm,pv+4) = -(-1.875*W0+5.75*W1-4.5*W2+W3)/6;
            A(pvm,pvm) = A(pvm,pvm)-0.5
          end do !m
        else
          do n = 1,4
            pvn = pv + n
            A(pvn,pvl+1) = C1t*rounddiff(pvn,pvl+1)
            A(pvn,pvl+2) = C2t*rounddiff(pvn,pvl+2)
            A(pvn,pvl+3) = C2t*rounddiff(pvn,pvl+3)
            A(pvn,pvl+4) = C1t*rounddiff(pvn,pvl+4)
          end do !n
        end if
      end do !l
    end do !j

end subroutine NyFill

function flatdiff(a,b)
   use params, only:k2,pi,i
   use fullsurface, only:sterm,y,x,dy,dx
   use bessel, only:h11
   implicit none
   complex(kind=k2)::flatdiff
   integer,intent(in)::a,b

   flatdiff = -sterm(b)*i*pi/2*h11(2*pi*sqrt((x(a)-x(b))**2+(y(a)-y(b))**2))*&
              cos(atan(dy(b))+(pi/2)-atan2(y(a)-y(b),x(a)-x(b)));
end function flatdiff

function rounddiff(a,b)
   use params, only:k2,pi,i
   use fullsurface, only:x,y,phi,dtheta,r,sterm
   use bessel, only:h11
   implicit none
   complex(kind=k2)::rounddiff
   integer,intent(in)::a,b
```

```
   rounddiff = i*pi/2*h11(2*pi*sqrt((x(b)-x(a))**2+&
               (y(b)-y(a))**2))/sqrt((x(b)-x(a))**2+(y(b)-y(a))**2)*&
               ((y(b)-y(a))*cos(phi(b))-(x(b)-x(a))*sin(phi(b)));
end function rounddiff

function cartTE(a,b,sing,n,c,offset,xp)
  use params, only: k2,pi
  use quadFunc, only: cartNfunc, setParams
  use quad, only:llquadz, linlogOrder
  implicit none
  complex(kind=k2)::cartTE
  complex(kind=k2)::k=2*pi
  real(kind=k2),intent(in)::a,b,sing,c(:,:),offset,xp(:)
  integer,intent(in)::n

  call setParams(n,sing,c,offset,xp,k)
  cartTE=llquadz(cartNfunc, linlogOrder(b-sing,n), sing, b, k)-&
         llquadz(cartNfunc, linlogOrder(sing-a,n), sing, a, k)
end function cartTE

function polTE(a,b,sing,n,r)
  use params, only: k2,pi
  use quadFunc, only: polNfunc, setPolParams
  use quad, only:llquadz, linlogOrder
  implicit none
  complex(kind=k2)::polTE
  complex(kind=k2)::k=2*pi
  real(kind=k2),intent(in)::a,b,sing,r
  integer,intent(in)::n

  call setPolParams(n,sing,k,r)
  polTE=llquadz(polNfunc,linlogOrder(b-sing,n), sing, b, k)-&
        llquadz(polNfunc,linlogOrder(sing-a,n), sing, a, k)
end function polTE

subroutine Efield(iangle, field, oangles)
  use pconst, only: dpi
  use params, only: k2, thpoints, thrange, len, patches, tpatches, i, k0
```

```
use fullsurface, only: x,y,sterm,dx,yp,dy,r,dtheta
use timer, only: tm_mark
use mkl95_precision, only: wp=>dp
use mkl95_lapack, only: gesv
use displot, only: plot
implicit none
real(kind=k2), intent(in)::iangle
complex(kind=k2), intent(out)::field(:)
real(kind=k2), intent(out)::oangles(:)
real(kind=k2)::thetar,T  ! incident angle in radians
real(kind=k2)::dth  ! spacing between output angle points
real(kind=k2)::xphase, yphase, pi=dpi
integer::ith,v,q,pvq,ii,j,k,l
complex(kind=k2),allocatable::rhs(:)
complex(kind=k2),allocatable::B(:,:)
character(len=*),parameter::titles(1)=(/"surface current"/)
complex(kind=k2)::C3,C4,C3t,C4t,test1,test2,test3,test4
real(kind=k2),parameter::zero=0
real,allocatable::Intensity(:)

! Compute right-hand side and solve for surface current
! NOTE:  You may want to change this to allow for multiple rhs
! without recomputing the Nystrom matrix

allocate(rhs(2*(patches*4+tpatches*4)))
thetar = iangle*pi/180 ! switching to radians
xphase=2*pi*cos(thetar);
yphase=-2*pi*sin(thetar);
rhs = exp(cmplx(zero,xphase*x+yphase*y));

! This next step is the bottleneck.  It can probably be speeded up
! a lot by using an optimized BLAS.  MATLAB is actually faster on
! this step then this program

call gesv(A,rhs) ! rhs is now the current
call tm_mark('Solve for surface current')

dth=thrange/(thpoints-1)
oangles=(/(ith*dth+iangle-thrange/2,ith=0,thpoints-1)/)*pi/180
```

```
C3 = 13*dx/12;
C4 = 11*dx/12;
C3t = 13*dtheta*r/12
C4t = 11*dtheta*r/12

allocate(B(thpoints,4*(2*(patches+tpatches))))

do v=1,thpoints
  if (len/=0) then
  do q=1,patches
    pvq = 4*(q-1)
    B(v,pvq+1)=-C3*rhs(pvq+1)*flatfield(pvq+1,oangles(v))
    B(v,pvq+2)=-C4*rhs(pvq+2)*flatfield(pvq+2,oangles(v))
    B(v,pvq+3)=-C4*rhs(pvq+3)*flatfield(pvq+3,oangles(v))
    B(v,pvq+4)=-C3*rhs(pvq+4)*flatfield(pvq+4,oangles(v))
  end do
  end if

  do q=patches+1,patches+tpatches
    pvq = 4*(q-1)
    B(v,pvq+1)=-C3t*rhs(pvq+1)*roundfield(pvq+1,oangles(v))
    B(v,pvq+2)=-C4t*rhs(pvq+2)*roundfield(pvq+2,oangles(v))
    B(v,pvq+3)=-C4t*rhs(pvq+3)*roundfield(pvq+3,oangles(v))
    B(v,pvq+4)=-C3t*rhs(pvq+4)*roundfield(pvq+4,oangles(v))
  end do

  ! Far Field - Top Surface
  if (len/=0) then
  do q=patches+tpatches+1,2*patches+tpatches
    pvq = 4*(q-1)
    B(v,pvq+1)=C3*rhs(pvq+1)*flatfield(pvq+1,oangles(v))
    B(v,pvq+2)=C4*rhs(pvq+2)*flatfield(pvq+2,oangles(v))
    B(v,pvq+3)=C4*rhs(pvq+3)*flatfield(pvq+3,oangles(v))
    B(v,pvq+4)=C3*rhs(pvq+4)*flatfield(pvq+4,oangles(v))
  end do
  end if

  ! Far Field - Left Surface
```

```
    do q=2*patches+tpatches+1,2*(patches+tpatches)
      pvq = 4*(q-1)
      B(v,pvq+1)=-C3t*rhs(pvq+1)*roundfield(pvq+1,oangles(v))
      B(v,pvq+2)=-C4t*rhs(pvq+2)*roundfield(pvq+2,oangles(v))
      B(v,pvq+3)=-C4t*rhs(pvq+3)*roundfield(pvq+3,oangles(v))
      B(v,pvq+4)=-C3t*rhs(pvq+4)*roundfield(pvq+4,oangles(v))
    end do
    field(v) = sum(B(v,:))*i*sqrt(i)*k0
  end do

  allocate(Intensity(size(field)))
  Intensity=abs(field)**2;
  deallocate(Intensity)
  call tm_mark('Solve for far field')
  deallocate(rhs)
  deallocate(B)
  oangles=oangles*180/pi  ! convert to degrees
end subroutine Efield

function roundfield(a,b)
  use params, only: pi, i, k0
  use fullsurface, only: phi, x, y, sterm
  implicit none
  integer,intent(in)::a
  real(kind=k2),intent(in)::b
  complex(kind=k2)::roundfield
  complex(kind=k2)::expnt,test
  real(kind=k2)::front

  expnt = exp(-i*k0*(x(a)*cos(b)+y(a)*sin(b)))
  roundfield = sterm(a)*sin(b-phi(a))*expnt
end function roundfield

function flatfield(a,b)
  use params, only: pi, i, k0
  use fullsurface, only:dx, dy, sterm, y, x
  implicit none
  integer,intent(in)::a
  real(kind=k2),intent(in)::b
```

```
    complex(kind=k2)::flatfield
    complex(kind=k2)::expnt

    expnt = exp(-i*k0*(x(a)*cos(b)+y(a)*sin(b)))
    flatfield = sterm(a)*sin(b-atan(dy(a)))*expnt
  end function flatfield

end module RoughTE
```

B.3 fullsurface.f90

In order to apply the MFIE to solve for the reflected TE field, the surface needed to be closed to avoid instabilities caused by derivatives at the endpoints of the surface. This module uses make_surface to randomly generate surface knots to simulate roughness based on the parameters of surface frequency and rms roughness height and connects them using a spline. The subroutine surf_setup then uses the calculated splines to interpolate values for the surface for the given discretization scheme.

```
! This module allocates dynamic memory.  It is wise to call surf_clean
! in order to free it up at program exit.
module fullsurface
  use spline
  use params, only:k2, externSurface
  implicit none
  private
  ! surf_clean: call this when done with surface to free allocated memory
  ! surf_setup: call to setup up surface points
  ! x,y: coordinates of surface points
  ! sterm: Jacobian at surface points
  ! dx: distance between x surface points
  ! c: spline coefficients
  ! theta: coordinate of end circle
```

```
  ! dtheta:  distance between angle points

  public surf_clean, surf_setup, x, y, dy, sterm, theta, phi
  public xp, yp, ct, cb, dx, pl, tl, dtheta, r
  real(kind=k2), allocatable:: x(:), y(:), xp(:), yp(:)
  real(kind=k2), allocatable:: sterm(:), theta(:), dy(:),phi(:)
  real(kind=k2), allocatable:: ct(:,:),cb(:,:)
  real(kind=k2)::pl, dx, r, tl, dtheta
  character(len=*),parameter::titles(1)=(/"surface"/)
  ! logical,parameter::debug=.TRUE.
contains

  subroutine surf_clean
    implicit none
    if(allocated(x)) deallocate(x)
    if(allocated(y)) deallocate(y)
    if(allocated(xp)) deallocate(xp)
    if(allocated(yp)) deallocate(yp)
    if(allocated(ct)) deallocate(ct)
    if(allocated(cb)) deallocate(cb)
    if(allocated(sterm)) deallocate(sterm)
    if(allocated(theta)) deallocate(theta)
    if(allocated(phi)) deallocate(phi)
    if(allocated(dy)) deallocate(dy)
  end subroutine surf_clean

  subroutine surf_setup(sigma, spacing)
    use params, only: len, patches, spacing, k2, surfplot, thick, tpatches, zero
    use pconst, only:dpi
    use displot
    implicit none
    real(kind=k2),intent(in)::sigma, spacing
    integer::i, j, np, ns, na, nt, bb
    real(kind=k2),allocatable::dybot(:), dytop(:)
    real(kind=k2),allocatable:: theta1(:), thetaa(:), thetab(:), phia(:),phib(:)
    real(kind=k2),allocatable:: xl(:), yl(:), xr(:), yr(:)
    real(kind=k2), allocatable:: sterml(:), stermr(:)
    real(kind=k2), allocatable:: x1(:), ytop(:), ybot(:), stermtop(:), stermbot(:)
    real(kind=k2)::pi=dpi
```

```
pl = len/patches ! patch length = tot length div number of patches
r=thick/2; ! radius is half of scatterer thickness
tl=pi/tpatches; ! theta length = pi / theta patches
dtheta=tl/4; ! delta theta between each quadrature point on sides

if (len/=0) then
  dx = pl/4 ! distance between quadrature points

  ns = 4*patches ! number of spline points
  np = len/spacing+1 ! number of knots
  na = 4*tpatches ! number of angle points
  nt = 2*ns+2*na ! size of arrays

  allocate(x1(ns),ybot(ns),ytop(ns))
  allocate(dybot(ns),dytop(ns))
  allocate(stermbot(ns),stermtop(ns))
  allocate(xp(np),yp(np),ct(4,np),cb(4,np))

  x1=(/ ((0.5+i)*dx, i=0,ns-1) /)

  ! Make the bottom surface
 ! call make_surface(sigma, xp, yp, cb, spacing, -thick, np-2)
  ybot =-thick+0*x1!(/ ( ppvalu(xp, cb, np-1, 4, x1(i), 0), i=1,ns) /)
  dybot=(/ ( ppvalu(xp, cb, np-1, 4, x1(i), 1), i=1,ns) /)
  stermbot = sqrt(1+dybot**2) ! path integral Jacobian

  ! Make the top surface
  call make_surface(sigma, xp, yp, ct, spacing, zero, np)
  ytop =(/ ( ppvalu(xp, ct, np-1, 4, x1(i), 0), i=1,ns) /)
  dytop=(/ ( ppvalu(xp, ct, np-1, 4, x1(i), 1), i=1,ns) /)
  stermtop = sqrt(1+dytop**2) ! path integral Jacobian
else
  ns = 0
  na = 4*tpatches
  nt = 2*ns+2*na
 ! write(6,*)'This is a circle'
end if
```

```
      allocate(theta1(na),thetaa(na),thetab(na))
      allocate(xl(na),yl(na),xr(na),yr(na))
      allocate(sterml(na),stermr(na))
      allocate(phia(na),phib(na))

      ! Make the side loops
      do j=1,4*tpatches
         theta1(j)=(-.5+j)*dtheta; ! setup for side (theta) quad
      end do

      phia = theta1; ! phi is angle of line tangent to surface
      phib = theta1-pi;
      thetaa = theta1-pi/2; ! theta is angle of position in polar
      thetab = theta1+pi/2;

      xr=r*cos(thetaa)+len; ! x values for right side
      yr=r*sin(thetaa)-r; ! y values for right side
      xl=r*cos(thetab);   ! x values for left side
      yl=r*sin(thetab)-r; ! y values for left side
      stermr=1; ! Jacobian for right side
      sterml=1; ! Jacobian for left side

! This section fills arrays with the information on the scatterer
! x,y: scatterer boundary values
! dy: surface derivatives
! sterm: surface Jacobian

      allocate(theta(nt))
      allocate(x(nt))
      allocate(y(nt))
      allocate(dy(nt))
      allocate(sterm(nt))
      allocate(phi(nt))

    if(len /= 0) then

! Bottom of Scatterer
      do bb=1,4*patches
         theta(bb)=0;
```

```
        phi(bb)=0;
        x(bb)=x1(bb);
        y(bb)=ybot(bb);
        dy(bb)=dybot(bb);
        sterm(bb)=stermbot(bb);
    end do

! Right side of Scatterer
    do bb=1,4*tpatches
        theta(4*patches+bb)=thetaa(bb);
        phi(4*patches+bb)=phia(bb);
        x(4*patches+bb)=xr(bb);
        y(4*patches+bb)=yr(bb);
        dy(4*patches+bb)=0;
        sterm(4*patches+bb)=stermr(bb);
    end do

! Top of Scatterer
    do bb=1,4*patches
        theta(4*(patches+tpatches)+bb)=0;
        phi(4*(patches+tpatches)+bb)=0;
        x(4*(patches+tpatches)+bb)=len-x1(bb);
        y(4*(patches+tpatches)+bb)=ytop(bb);
        dy(4*(patches+tpatches)+bb)=dytop(bb);
        sterm(4*(patches+tpatches)+bb)=stermtop(bb);
    end do

! Left side of Scatterer
    do bb=1,4*tpatches
        theta(4*(2*patches+tpatches)+bb)=thetab(bb);
        phi(4*(2*patches+tpatches)+bb)=phib(bb);
        x(4*(2*patches+tpatches)+bb)=xl(bb);
        y(4*(2*patches+tpatches)+bb)=yl(bb);
        dy(4*(2*patches+tpatches)+bb)=0;
        sterm(4*(2*patches+tpatches)+bb)=sterml(bb);
    end do

    ! Deallocate used variables
    deallocate(x1)
```

```
  deallocate(ytop)
  deallocate(ybot)
  deallocate(dytop)
  deallocate(dybot)
  deallocate(stermtop)
  deallocate(stermbot)
  deallocate(sterml)
  deallocate(stermr)
  else
  ! Surface is a circle
     x=(/xr, xl/)
     y=(/yr, yl/)
     phi=(/phia, phib/)
     theta=(/thetaa, thetab/)
     dy=0
     sterm = (/stermr, sterml/)

  end if

  deallocate(theta1, thetaa, thetab)
  deallocate(xr, xl)
  deallocate(yr, yl)
  deallocate(phia,phib)

  if(surfplot) then
     call plot(real(x),real(y*1.001+0.001),'x','y',titles,'line')
  endif
end subroutine surf_setup

subroutine make_surface(sigma, xp, yp, coef, spacing, ht, np)
  use params, only: k2, externSurface
  use random
  implicit none
  integer, intent(in)::np
  real(kind=k2), intent(in)::sigma, spacing, ht
  real(kind=k2), intent(out)::xp(:),yp(:),coef(:,:)
  integer::i,ios
  real(kind=k2)::dummy
  real,external::rand
```

```
    if(externSurface)then
      ! Read in the spline points from a file
      ! Check the points for consistency
      write(6,'(a)')'Reading surface data from surface.txt'
      open(unit=7,file="surface.txt",status="old")
      read(unit=7,fmt=*,iostat=ios)(xp(i),i=1,np),(yp(i),i=1,np)
      if(ios /= 0) then
        stop 'Error reading surface file.  Not enough data points.'
      end if
      read(unit=7,fmt='(f)',iostat=ios)dummy
      if(ios ==0) then
        stop 'Error reading surface file.  Too many data points.'
      end if
      close(unit=7)
      write(6,'(a)')'Read data successfully from surface.txt'
    else
      xp = (/ ((i-1)*spacing, i=1,np) /)
      yp(2:np-1) = (/ (random_normal()*sigma,i=2,np-1) /) + ht
    end if

    yp(1)=ht
    yp(np)=ht
    coef(1,:)=yp
    coef(2,1)=0
    coef(2,np)=0
    call cubspl(xp, coef, np, 1, 1)

  end subroutine make_surface

end module fullsurface
```

Appendix C

Matlab Code

This matlab code loads the TE and TM data based on the parameters of the run. It calculates the mean values for each set of parameters and plots the function versus qh for each specific knot separation. The functions for each knot separation are then fit to a cubic polynomial and the coefficient values are plotted for both TE and TM.

```
clear all; close all; clc;
format long e

knots=input('knots= ');                  % input values for marylou run
angles=input('angles= ');
heights=input('heights= ');
samples=input('samples= ');

params=knots*angles*heights;             % total parameters
params2=angles*heights;                  % total qh values

isep=1:knots;                            % index values
iangle=1:angles;
iheight=1:heights;

sep=isep*(20.0/knots);                   % knot separation
angle=15.0+(iangle-1)*(75.0/(angles-1)); % incident angle
height=iheight*(0.1/heights);            % roughness height
```

```
load TMTrial1.txt
Rm=-log(TMTrial1);
load TETrial5.txt
Re=-log(TETrial5);

Sm=zeros(params,samples);
Se=zeros(params,samples);

for i=1:samples
    Sm(1:params,i)=Rm((1:params)+(i-1)*params);
    Se(1:params,i)=Re((1:params)+(i-1)*params);
end

for i=1:params;
    qbare(i)=(sum(Se(i,:))/samples)';
    q2bare(i)=(sum(Se(i,:).^2)/(samples-1))';
    qbarm(i)=(sum(Sm(i,:))/samples)';
    q2barm(i)=(sum(Sm(i,:).^2)/(samples-1))';
end

qsqe(1:params)=sqrt(q2bare(1:params)-qbare(1:params).^2*samples/(samples-1))';
qsqm(1:params)=sqrt(q2barm(1:params)-qbarm(1:params).^2*samples/(samples-1))';

for j=iheight
    qh(iangle+(j-1)*angles)=2*pi*sin(angle(iangle)*pi/180)*height(j);
end

H=ttest(qbare,qbarm)

tbare=zeros(params2,knots);
tsqe=zeros(params2,knots);
tbarm=zeros(params2,knots);
tsqm=zeros(params2,knots);
for j=1:knots
    tbare(1:params2,j)=qbare(j+((1:params2)-1)*knots);
    tsqe(1:params2,j)=qsqe(j+((1:params2)-1)*knots);
    tbarm(1:params2,j)=qbarm(j+((1:params2)-1)*knots);
```

```
    tsqm(1:params2,j)=qsqm(j+((1:params2)-1)*knots);
end

mdat(:,1)=qh;
mdat(:,2:knots+1)=tbarm;
mdat(:,knots+2:2*knots+1)=tsqm;
edat(:,1)=qh;
edat(:,2:knots+1)=tbare;
edat(:,knots+2:2*knots+1)=tsqe;
sdate=sortrows(edat);
sdatm=sortrows(mdat);

options=fitoptions('poly3');
options.lower=[-Inf -Inf -Inf -0];
options.upper=[Inf Inf Inf 0];

qhf=sdate(:,1);

DW=2*qhf.^2;

for n=isep
    ks=n*(20.0/knots);
    s=sprintf('Attenuation at %g wavelength knot separation',ks);

    % For S polarization
    Fm=fit(qhf,sdatm(:,n+1),'poly3',options);
    Pm(n,1)=Fm.p1;
    Pm(n,2)=Fm.p2;
    Pm(n,3)=Fm.p3;
    Pm(n,4)=Fm.p4;

    Cm=confint(Fm);
    C1m(n,1)=(Fm.p1-Cm(1,1))/2.06;
    C1m(n,2)=(Cm(2,1)-Fm.p1)/2.06;
    C2m(n,1)=(Fm.p2-Cm(1,2))/2.06;
    C2m(n,2)=(Cm(2,2)-Fm.p2)/2.06;
    C3m(n,1)=(Fm.p3-Cm(1,3))/2.06;
    C3m(n,2)=(Cm(2,3)-Fm.p3)/2.06;
```

```
    % For P polarization
    Fe=fit(qhf,sdate(:,n+1),'poly3',options);
    Pe(n,1)=Fe.p1;
    Pe(n,2)=Fe.p2;
    Pe(n,3)=Fe.p3;
    Pe(n,4)=Fe.p4;

    Ce=confint(Fe);
    C1e(n,1)=(Fe.p1-Ce(1,1))/2.06;
    C1e(n,2)=(Ce(2,1)-Fe.p1)/2.06;
    C2e(n,1)=(Fe.p2-Ce(1,2))/2.06;
    C2e(n,2)=(Ce(2,2)-Fe.p2)/2.06;
    C3e(n,1)=(Fe.p3-Ce(1,3))/2.06;
    C3e(n,2)=(Ce(2,3)-Fe.p3)/2.06;



    M=polyval(Pm(n,:),qhf);
    E=polyval(Pe(n,:),qhf);
    De=sdate(:,n+1);
    Dm=sdatm(:,n+1);

     plot(qhf,DW,'r-',qhf,Dm,'b.',qhf,De,'g.')
     xlabel('qh')
     ylabel('ln(R)')
     title(s)
     axis([0 .7 0 .7])
     legend('Debye Waller Calculation','Reflectance(TM)','Reflectance(TE)')
     pause
end

figure
E1=errorbar(isep,Pe(:,1),C1e(:,1),C1e(:,2),'b.');
    xlabel('Knot Seperation')
    ylabel('Coefficient')
    title('Cubic Term (TE)')
    axis([0 40 -2 2])
figure
E2=errorbar(isep,Pe(:,2),C2e(:,1),C2e(:,2),'b.');
    xlabel('Knot Seperation')
```

```
    ylabel('Coefficient')
    title('Quadratic Term (TE)')
    axis([0 40 -1 3])
figure
E3=errorbar(isep,Pe(:,3),C3e(:,1),C3e(:,2),'b.');
    xlabel('Knot Seperation')
    ylabel('Coefficient')
    title('Linear Term (TE)')
    %axis([0 40 -.4 1.5])
figure
E4=errorbar(isep,Pm(:,1),C1m(:,1),C1m(:,2),'b.');
    xlabel('Knot Seperation')
    ylabel('Coefficient')
    title('Cubic Term (TM)')
    axis([0 40 -2 2])
figure
E5=errorbar(isep,Pm(:,2),C2m(:,1),C2m(:,2),'b.');
    xlabel('Knot Seperation')
    ylabel('Coefficient')
    title('Quadratic Term (TM)')
    axis([0 40 -1 3])
figure
E6=errorbar(isep,Pm(:,3),C3m(:,1),C3m(:,2),'b.');
    xlabel('Knot Seperation')
    ylabel('Coefficient')
    title('Linear Term (TM)')
    axis([0 40 -.3 .1])
```

Index