NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZATION OF ELECTROMAGNETIC WAVE PROPAGATION THROUGH A HETEROGENEOUS LIQUID CRYSTAL LAYER

by

Michael A. Winslow

March 2013

Thesis Advisor:	Hong Zhou
Second Reader:	Wei Kang

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704–0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE OPTIMIZATION OF ELECTROMAGNETIC WAVE PROPAGATION THROUGH A HETEROGENEOUS LIQUID CRYSTAL LAYER		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Michael A. Winslow			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Advances in technology have given way to concepts in warfare that were once constrained to the world of science fiction. In an effort to stay ahead of any potential adversary's weapons development, we must look down the path of countermeasures to high-energy electromagnetic weapons.

In the attempt to engineer a material that can reduce transmitted beam intensity by the greatest factor, we look to liquid crystals. They have great potential to provide a starting point to engineer a material in order to show increased protection of DoD assets from high-energy beam weapons.

We first develop one-dimensional finite-difference time-domain codes to solve Maxwell's equations in order to model the electromagnetic wave propagation in a liquid crystal layer. After validating numerical results with analytical results for matched anchoring, we investigate the heterogeneous liquid crystal structures with mismatched anchoring conditions and determine the best anchoring conditions to minimize transmitted beam intensity. The main result of the simulation is that for a known incident wave the maximum reduction of the transmitted intensity is achieved with matched anchoring conditions. However, for mixed anchoring conditions, there is evidence that the mixed structure can reduce the intensity for a wider range of waves.

14. SUBJECT TERMS Electromagnetic Wave Propagation, Liquid Crystal Polymer, Finite-Difference Time-Domain Code			**15. NUMBER OF PAGES** 131
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

OPTIMIZATION OF ELECTROMAGNETIC WAVE PROPAGATION THROUGH A HETEROGENEOUS LIQUID CRYSTAL LAYER

Michael A. Winslow
Lieutenant, United States Navy
B.S. University of Arizona, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
March 2013

Author: Michael A. Winslow

Approved by: Hong Zhou, Associate Professor
Thesis Advisor

Wei Kang, Professor
Second Reader

Carlos Borges, Professsor
Chair, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Advances in technology have given way to concepts in warfare that were once constrained to the world of science fiction. In an effort to stay ahead of any potential adversary's weapons development, we must look down the path of countermeasures to high-energy electromagnetic weapons.

In the attempt to engineer a material that can reduce transmitted beam intensity by the greatest factor, we look to liquid crystals. They have great potential to provide a starting point to engineer a material in order to show increased protection of DoD assets from high-energy beam weapons.

We first develop one-dimensional finite-difference time-domain codes to solve Maxwell's equations in order to model the electromagnetic wave propagation in a liquid crystal layer. After validating numerical results with analytical results for matched anchoring, we investigate the heterogeneous liquid crystal structures with mismatched anchoring conditions and determine the best anchoring conditions to minimize transmitted beam intensity. The main result of the simulation is that for a known incident wave the maximum reduction of the transmitted intensity is achieved with matched anchoring conditions. However, for mixed anchoring conditions, there is evidence that the mixed structure can reduce the intensity for a wider range of waves.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

DoD	Department of Defense
FDTD	Finite-Difference Time-Domain
IF	Incident Field
LC	Liquid Crystal
LCP	Liquid Crystal Polymer
LE	Leslie Ericksen
PEC	Perfect Electric Conductor
PML	Perfectly Matched Layer
TF	Total Field

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The success of this thesis and my master's degree completion at the Naval Postgraduate School are due to the generosity, hospitality, and tremendous help of the entire Applied Mathematics Department. In particular, I owe a great deal of gratitude to my thesis advisor, Dr. Hong Zhou, and her post doc Dr. Eric Choate, who have helped me to accomplish this project. I consider it an honor to have worked with both of them. I also would like to thank Dr. Wei Kang for being my second reader.

To my wife and children, I dedicate this thesis and I must thank you for your constant support and love as we progressed through this phase of our naval career. Thank you for standing by me and dealing with my absences from many family occasions. I am truly blessed to have you all in my life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. ELECTROMAGNETIC WAVE PROPAGATION IN LIQUID CRYSTALS

A. INTRODUCTION

War is a "come as you are" business; on the onset of conflict, we can only bring what we have and only until we realize the capabilities of our enemy, can we then decide to try and develop means by which to counter their offensive. Therefore, we must stay ahead of any potential adversary's weapons development and anticipate what could be brought to the table, should that conflict occur. The world of high-energy electromagnetic weapons is no different.

It is through this need to stay ahead that we must look at the possibilities of utilizing known material or engineering a material with the capabilities to minimize the impact of these types of weapons. For this, we look to liquid crystals (LCs).

Through the continued study of propagation of electromagnetic energy in liquid crystals (Hwang & Rey, 2005; Hwang, Han, & Rey, 2007; Kriezis & Elston, 1999; Kriezis & Elston, 2000; Ziogos & Kriezis, 2008), it becomes increasingly important to branch off of the typical models that utilize a consistency in alignment of anisotropic axes with that of the coordinate axes which has been studied and looked at by (Choate & Zhou, 2012.) In order to push passed the constraints of that study, we must first validate the model used by Choate and Zhou and then use the validated model to investigate the differences between matched and mixed anchoring conditions of the liquid crystal layer. This shifts our focus from the matched anchoring conditions on the top and bottom of the homogeneous layer to mismatched or mixed anchoring conditions of an anisotropic layer.

When considering liquid crystals, we see that there are many advantages to utilizing them in an effort to minimize or maximize the intensity of a transmitted beam. For the purposes of this paper, we will take a generic look at liquid crystals but with the specific focusing on the way that the molecules within the liquid crystal medium align. Additionally, even though there have been studies about maximizing the transmitted intensity, we will only be looking at efforts to minimize that transmitted intensity.

In Figure 1, we show a cartoon of the problem setup. We consider a monochromatic electromagnetic wave passing through a liquid crystal layer. The liquid crystal layer is sandwiched between two supporting glass plates. It drives the question as to how adjusting the internal structure of a liquid crystal layer impacts the difference between the intensity of an incident beam as compared to the intensity of the transmitted beam through the layer. Our ability to adjust the internal structure of the material lies within the anchoring conditions at the top and bottom plates located at each end of the layer. In this thesis, we assume that the orientation of the liquid crystal molecules is not affected by the external electromagnetic field.

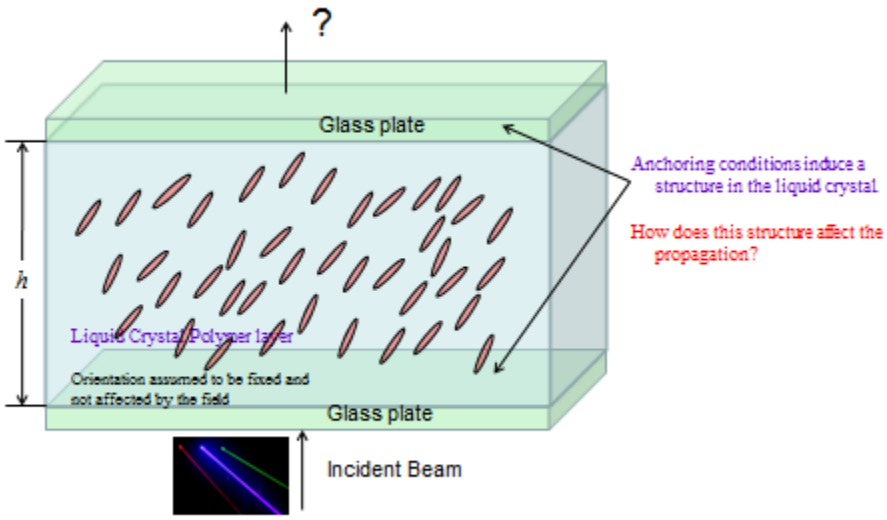

Figure 1. A cartoon description of the problem setup.

B. A BRIEF OVERVIEW OF LIQUID CRYSTALS

The concept of liquid crystals was realized in 1888, by the Austrian chemist Friedrich Reinitzer. While engaged in studies with regard to cholesterol, he noted that the material cholesteryl benzoate had what appeared to be two melting points. The first melting point at 145.5 degrees Celsius resulted in a cloudy liquid that remained cloudy until heated to the next melting point of 178.5 degrees Celsius. At this higher

temperature, the material became a transparent liquid. Further investigation revealed no impurity based cause for this phenomenon.

Through efforts of Reinitzer and crystal optic expert and German physicist, Otto Lehmann, it was determined that the cloudy state was unique and presented a material that was between a liquid and solid with shared properties of both. This opened the door to the world of liquid crystals, a material with anisotropic properties that varied with orientation of molecules (Liquid Crystals, 2013.) Solids have a required order of both the orientation and position of the individual molecules of the substance. Liquids, on the other hand, are not bound by this same principle and can change the orientation and position of the molecules. The discovery by Reinitzer and Lehmann revealed that the material, termed liquid crystals, had an orientation order but no position order. This explains why they chose to term the material as a liquid crystal.

It is through these varying anisotropic properties that we choose to utilize the concept of a liquid crystal as the medium by which to affect transmitted beam intensity. One method used to effect the orientation of the molecules within the liquid crystal is to establish the anchoring conditions on a surface by either a mechanical rubbing or grooving of the surface. This causes the molecules to align with the mechanical aspect of the grooves. Another method is to apply an electric charge to the layer, thereby effecting orientation. Once orientation is established by anchoring conditions, the other molecules in the immediate vicinity of those anchored molecules will align their orientation, with respect to ψ and θ, similar to that of the anchored molecule as illustrated in Figure 2 and Figure 3. With that in mind, we must establish that the orientation is known and predetermined prior to the incident beam interaction and that the electromagnetic field of the beam will have no impact on preexisting molecule orientation.

We use the Leslie-Ericksen (LE) model theory to help us describe the behavior of the liquid crystal which has the characteristics of a rigid-rod system. For a more general description of the mathematical modeling aspects of liquid crystals, see (Zhou, Forest & Wang, 2010.) As seen in Figure 2, the molecules have a high aspect ratio, which in turn allows for anisotropic characteristics. The molecule's centers of mass are random but

there is an orientation order coinciding with the individual molecules. Based on LE theory, we define the unit vector $\mathbf{m}$ as the axis of symmetry of a spheroid molecule where $\mathbf{m}$ is not random. We further define the unit vector $\mathbf{n}$ as the major director that is the preferred direction of $\mathbf{m}$ at the given $(\mathbf{x},t)$. The major director is the local average orientation of molecules.

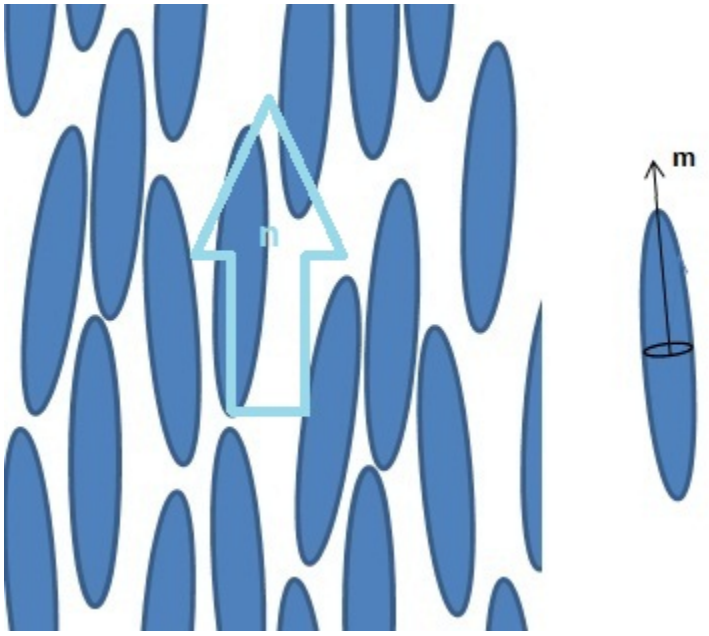

Figure 2. Nematic phase characteristics of LCs.

The nematic phase nature of liquid crystals means that the molecules of the material possess orientation direction for the given property conditions but requires no positional direction and can therefore be considered random.

The orientation direction of the material is described by a major director axis. Through the assumptions that no reorientation of molecules occurs by the electromagnetic field of the incident beam or by the flowing of material within the liquid crystal, we can establish and utilize the major director, $\mathbf{n}$ as determined by the steady state equation

$$(\mathbf{I}-\mathbf{nn})\cdot\nabla^2\mathbf{n}=0 \qquad (1.1)$$

where $\mathbf{nn}$ is the dyadic product, $\nabla = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{bmatrix}^T$ is the gradient operator and $\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplacian operator. That is, $\nabla^2 \mathbf{n}$ is in the null space of $\mathbf{I} - \mathbf{nn}$, which is $\lambda \mathbf{n}$, where λ is an arbitrary scalar. Therefore, we need to solve

$$\nabla^2 \mathbf{n} = \lambda \mathbf{n} . \tag{1.2}$$

We are left with three unknowns and an arbitrary λ that we can choose in order to solve.

Since $\mathbf{n}$ is a unit vector, we can rewrite $\mathbf{n}$ in terms of the angles θ and ψ as

$$\mathbf{n} = \begin{bmatrix} \cos\theta\cos\psi \\ \cos\theta\sin\psi \\ \sin\theta \end{bmatrix}. \tag{1.3}$$

Plugging this form into Equation(1.1), we have a set of equations for both θ and ψ,

$$\begin{aligned} \nabla^2\psi - 2\tan\theta\nabla\theta\cdot\nabla\psi &= 0 \\ \nabla^2\theta + \sin\theta\cos\theta\nabla\psi\cdot\nabla\psi &= 0 \\ \nabla\theta\cdot\nabla\theta + \cos^2\theta\nabla\psi\cdot\nabla\psi &= -\lambda \end{aligned} .$$

We only need to solve the top two equations, with the understanding that λ is arbitrary and its actual value is irrelevant, thereby reducing the system of equations to

$$\begin{aligned} \nabla^2\psi - 2\tan\theta\nabla\theta\cdot\nabla\psi &= 0 \\ \nabla^2\theta + \sin\theta\cos\theta\nabla\psi\cdot\nabla\psi &= 0 \end{aligned} \tag{1.4}$$

The solutions to these equations are determined by the boundary conditions. In our specific case, these boundary conditions are established by the two glass plates which support the liquid crystal layer at $z = 0$ and $z = h$. These plates will be referred to as the bottom and top plates, respectively. Utilizing anchoring conditions that are uniform in the x- and y-directions, the system of Equations (1.4), can be reduced to a system of second order ordinary differential equations in the z-direction. In order to solve, this requires four boundary conditions defined as

$$\theta_{bot} = \theta(z = 0)$$
$$\psi_{bot} = \psi(z = 0)$$
$$\theta_{top} = \theta(z = h)$$
$$\psi_{top} = \psi(z = h)$$

These four boundary conditions are our control parameters for determining the orientation of molecules within the layer and the resultant propagation of light through the layer. The goal of this paper is to determine the effect of varying these parameters on the transmitted intensity of a given incident beam. As seen in Figure 3. , we define θ as the angle between the $xy-$plane and the $z-$axis. The value of θ is limited to values of 0 to $\pi/2$ where $\theta = 0$ is at the $xy-$plane. Additionally, we define ψ as the angle that lies within the $xy-$plane. The value of ψ is limited to values of 0 to $\pi/2$ where $\psi = 0$ is in line with the $x-$axis.

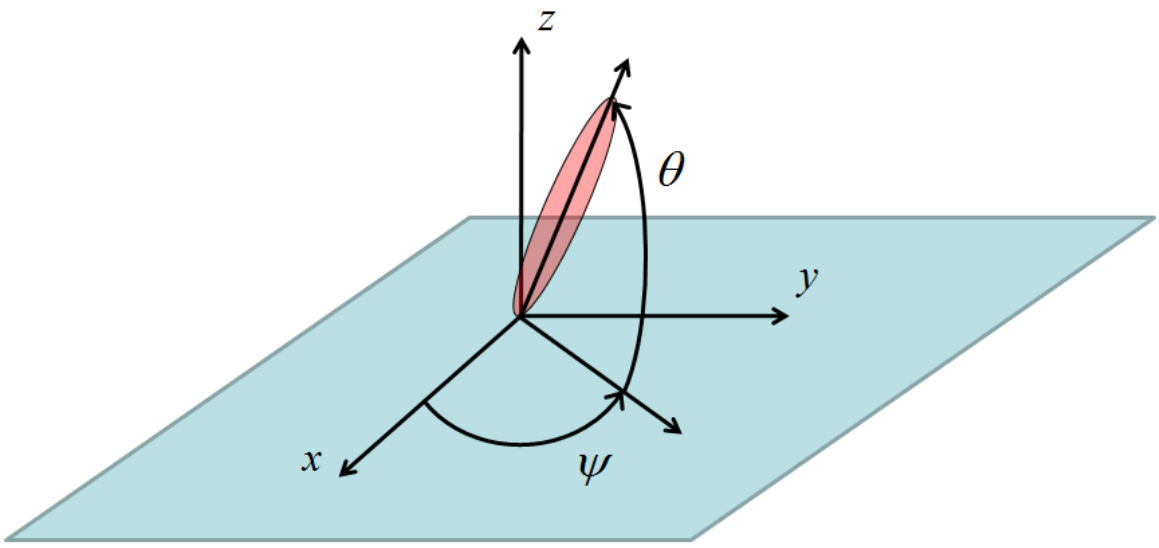

Figure 3. Description of θ and ψ.

It is through the values of θ and ψ at the boundaries that we can effectively tune the layer to achieve maximum or minimum intensity of the transmitted beam. We will show later through model runs of how to best determine θ and ψ for our given assumptions.

C. MAXWELL'S EQUATIONS

This study will begin by first reviewing the fundamental basics of previous studies of electromagnetic plane waves and their propagation through liquid crystal medium of constrained orientation. The orientation structure is determined by anchoring conditions on the top and bottom plates. Our goal is to find the dependence of the transmitted intensity on the structure, controlled by the anchoring conditions. Following that, we will validate the model used in previous work (Choate & Zhou, 2012) through the comparison of incident beam intensity to the transmitted beam intensity across the given medium. Finally, we will run a modified model that will vary the angles of orientation within the given medium to determine overall impact of the transmitted intensity of the given energy beam.

As outlined in work done by Choate and Zhou (Choate & Zhou, 2012), we will maintain consistency in the coordinate system and geometry of the overall system. This choice will allow for a more convenient comparison of data provided by their past studies and it will also allow the validation of the model used in their studies. In order to derive the equations that we will be working with, we must first establish the layer type that we will be working with.

As shown in Figure 4, the regions $z<0$ and $z>h$ are vacuum regions with the given permittivity and permeability, ε_o and μ_o. The region between $z=0$ and $z=h$, is the medium of focus and is modeled by a liquid crystal polymer of thickness h. The orientations in the x- and y-directions are assumed to be homogeneous but are not required to be the same value for x as for y. The orientation is given by the major director, represented by the unit vector **n**, that can vary in space. It is determined by Equation (1.1) coupled with the anchoring conditions at the plates. Since the anchoring conditions are uniform in x- and y-directions, we have a problem. We want to examine the effect of using different anchoring conditions on the top and bottom on the intensity of the transmitted beam. Later, we will solve for the intensity of the transmitted beam.

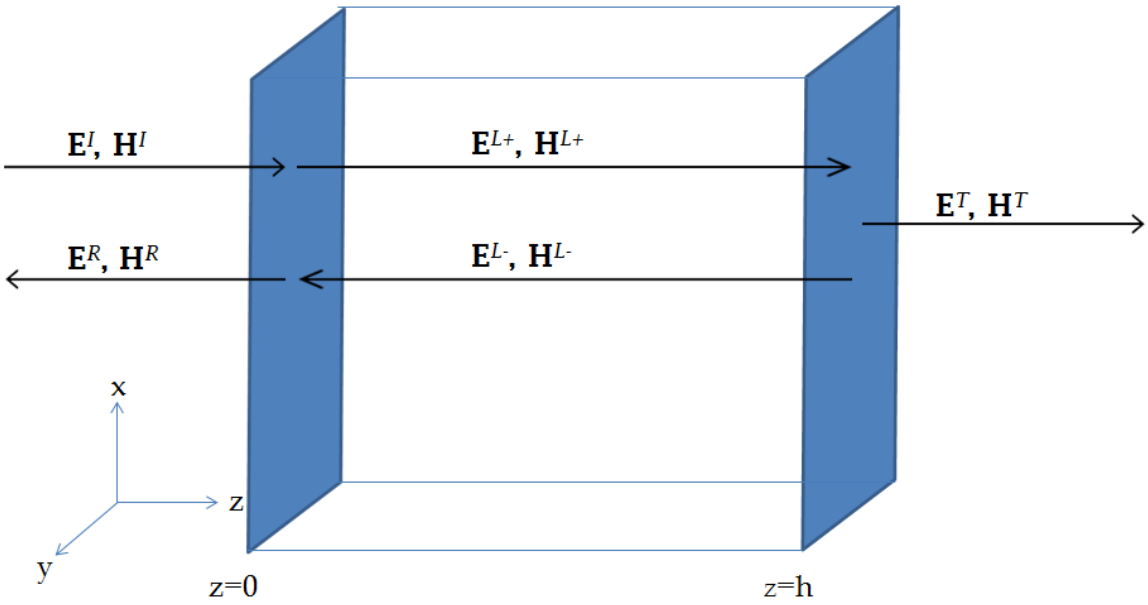

Figure 4. Liquid crystal layer orientation and geometry.

The fundamental starting point for any electrodynamics system is to look at Maxwell's equations

$$\frac{\partial \mathbf{D}}{\partial t} = \nabla \times \mathbf{H} \tag{1.5}$$

$$\nabla \cdot \mathbf{D} = 0 \tag{1.6}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \tag{1.7}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.8}$$

where **E** is the electric field, **D** is the electric displacement field , **H** is the magnetic field, and **B** is the magnetic flux density.

We assume that the liquid crystal is nonmagnetic, such that $\mathbf{B} = \mu_0 \mathbf{H}$, where μ_0 is the permeability in free space. The anisotropy of the liquid crystal appears in the constitutive equation for the electric field,

$$\mathbf{D} = \boldsymbol{\varepsilon} \cdot \mathbf{E} \tag{1.9}$$

for the permittivity tensor,

$$\boldsymbol{\varepsilon} = \varepsilon_{\perp}\mathbf{I} + \triangle\varepsilon \mathbf{nn} \tag{1.10}$$

where $\triangle\varepsilon = \varepsilon_{\parallel} - \varepsilon_{\perp}$. Solving the system of Equations (1.4) for ψ and θ and substituting them into Equation (1.3) gives us the **n** that can be substituted into Equation (1.10) for the permittivity tensor. The extraordinary permittivity, $\varepsilon_{\parallel}$, is the permittivity in the direction of **n**. The ordinary permittivity, $\varepsilon_{\perp}$, is the permittivity in directions orthogonal to **n**. We will assume that $\varepsilon_{\parallel} > \varepsilon_{\perp}$ due to the rod shape of the liquid crystal molecules.

We introduce the rescaling,

$$\tilde{\mathbf{D}} = \frac{1}{\sqrt{\varepsilon_0\mu_0}}\mathbf{D}, \quad \tilde{\mathbf{E}} = \sqrt{\frac{\varepsilon_0}{\mu_0}}\mathbf{E}, \quad \tilde{\boldsymbol{\varepsilon}} = \frac{\boldsymbol{\varepsilon}}{\varepsilon_0},$$

where ε_0 is the vacuum permittivity. While the relative permittivity $\tilde{\boldsymbol{\varepsilon}}$ is dimensionless, both $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{E}}$ are not dimensionless. They have equivalent dimensions to **H**. Also, the free space quantities ε_0 and μ_0 are defined such that $\frac{1}{\sqrt{\varepsilon_0\mu_0}} = c_0$, where c_0 is the speed of light in a vacuum. This rescaling transforms the system (1.5)(1.7)(1.9) into

$$\frac{\partial\tilde{\mathbf{D}}}{\partial t} = \frac{1}{\sqrt{\varepsilon_0\mu_0}}\nabla\times\mathbf{H} \tag{1.11}$$

$$\tilde{\mathbf{D}} = \tilde{\boldsymbol{\varepsilon}}\cdot\tilde{\mathbf{E}} \tag{1.12}$$

$$\frac{\partial\mathbf{H}}{\partial t} = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}\nabla\times\tilde{\mathbf{E}} \tag{1.13}$$

Since the anchoring conditions in the x- and y-directions are homogeneous, the partial derivatives are zero in those directions. Utilizing the condition that all partial derivatives with respect to x and y are zero, the components of Equations (1.11) through (1.13) are:

$$\frac{\partial\tilde{D}_x}{\partial t} = -c_0\frac{\partial H_y}{\partial z}, \tag{1.14}$$

$$\frac{\partial\tilde{D}_y}{\partial t} = c_0\frac{\partial H_x}{\partial z}, \tag{1.15}$$

$$\frac{\partial \tilde{D}_z}{\partial t} = 0, \tag{1.16}$$

$$\tilde{\mathbf{D}} = \tilde{\boldsymbol{\varepsilon}} \cdot \tilde{\mathbf{E}}, \tag{1.17}$$

$$\frac{\partial H_x}{\partial t} = c_0 \frac{\partial \tilde{E}_y}{\partial z}, \tag{1.18}$$

$$\frac{\partial H_y}{\partial t} = -c_0 \frac{\partial \tilde{E}_x}{\partial z}, \tag{1.19}$$

$$\frac{\partial H_z}{\partial t} = 0. \tag{1.20}$$

The form and nature of Equation (1.16) and Equation (1.20) tells us that the components $\tilde{D}_z$ and H_z are constants with respect to time. Knowing that the H_z and $\tilde{D}_z$ components of the incident beam are zero, we set them as $H_z = 0$ and $\tilde{D}_z = 0$ through the remainder of the layer. However, $\tilde{E}_z$ is nonzero inside the layer. As a consequence of $\tilde{D}_z = \tilde{\varepsilon}_{xz}\tilde{E}_x + \tilde{\varepsilon}_{yz}\tilde{E}_y + \tilde{\varepsilon}_{zz}\tilde{E}_z = 0$ combined with Equation (1.17), we have $\tilde{E}_z = -\frac{1}{\tilde{\varepsilon}_{zz}}(\tilde{\varepsilon}_{xz}\tilde{E}_x + \tilde{\varepsilon}_{yz}\tilde{E}_y)$.

Once the electric and magnetic fields are known, we can compute the intensity as follows. First the Poynting vector is defined as the cross product of the electric field and the magnetics field

$$\tilde{\mathbf{S}} = \sqrt{\frac{\mu_0}{\varepsilon_0}}\tilde{\mathbf{E}} \times \mathbf{H}.$$

The intensity is defined as the time average over one period of the magnitude of the Poynting vector or

$$\tilde{I} = \frac{\omega}{2\pi} \int_t^{t+\frac{2\pi}{\omega}} \left|\tilde{\mathbf{S}}\right| dt.$$

For convenience, we drop the tildes for the remainder of the paper.

D. NUMERICAL ALGORITHM

We will solve the system, Equations (1.14) through (1.20) numerically utilizing the one dimensional finite-difference time-domain (FDTD) method (Taflove & Hagness, 2005.) This method uses the Yee cell discretization where the electric field and the magnetic field are stored on staggered grids. This allows for a second order scheme in both space and time while only requiring storage of one field value per grid cell. The problem that we are trying to solve assumes an infinite domain in the z-direction, therefore we have to truncate the computational domain with perfectly matched layers (PMLs) to minimize the artificial reflections created by the truncation of the domain with perfect electric conductor (PEC) boundaries.

The incident beam is a plane wave where the wavelength, polarization, incident angle, electric field $\mathbf{E}_{inc}$, and magnetic field $\mathbf{H}_{inc}$ are known. These fields represent the beam that would pass through the computational domain if there were no liquid crystal layer and are assumed to be known throughout the computational domain. The incident angle is supposed to be perpendicular to the bottom plate and in line with the $z-$axis. In order to introduce the incident beam to the model, we split the computational domain into a total field (TF) region and a scattered field (SF) region, as shown in Figure 5. We define the scattered fields $\mathbf{E}_{scat}$ and $\mathbf{H}_{scat}$ as the differences between the unknown fields and the known incident fields so that the total unknown fields are

$$\mathbf{E}_{total} = \mathbf{E}_{scat} + \mathbf{E}_{inc}, \qquad \mathbf{H}_{total} = \mathbf{H}_{scat} + \mathbf{H}_{inc}.$$

In the TF region, we store and update the total fields, but in the SF regions, we only store and update the scattered fields. This allows us to impose the incident beam at the z_{start} in the SF region without the nonphysical reflections that would be caused by the reflected beam returning to z_{start}. This convention does introduce the difficulty of calculating the electric and magnetic fields as they pass from one region to another, which we will discuss later.

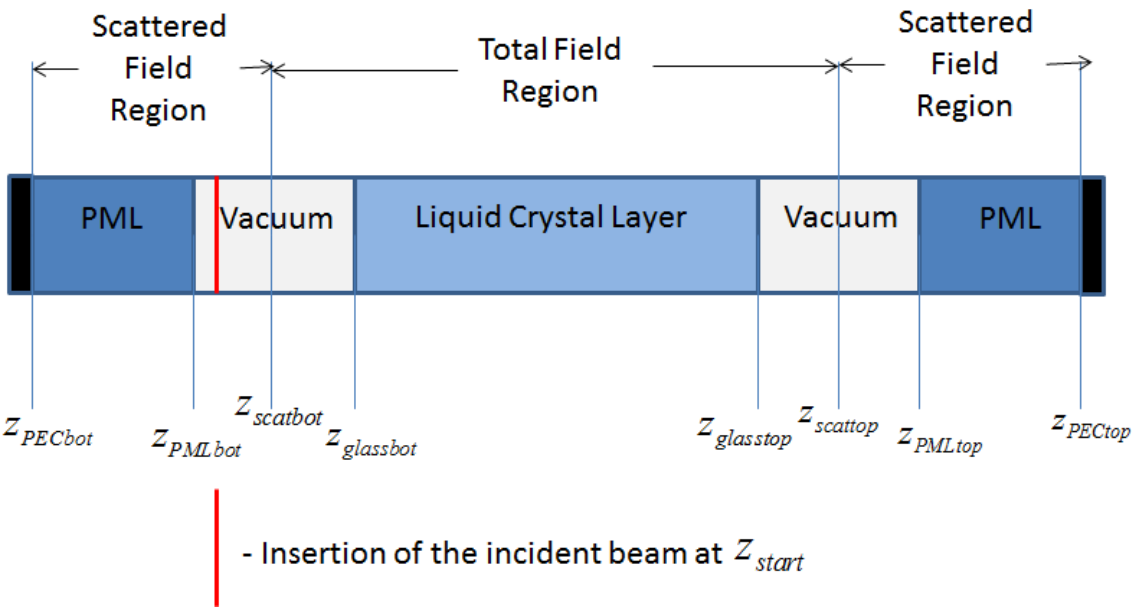

Figure 5. Computational domain.

Our computational domain consists of the liquid crystal layer between the points $z_{glassbot}$ and $z_{glasstop}$ (being $z_{glassbot}+h$) which is symmetrically surrounded on both ends with a region of vacuum between the points $z_{glasstop}$ and z_{PMLtop} and a PML region between the points z_{PMLtop} and z_{PECbot}, which marks the end of the truncated domain. There exist similar layers beyond the bottom end of the liquid crystal layer as shown in Figure 5. Additionally, between $z_{glasstop}$ and z_{PMLtop}, we have a transition between the total field to the scattered field at the point $z_{scattop}$ and similarly at $z_{scatbot}$ for the bottom portion. The incident beam is introduced in the bottom scattered field at z_{start} located between z_{PMLbot} and $z_{scatbot}$.

In order to discretize space and time, we define the grid spacing by the width of the liquid crystal layer, h, between the bottom and top plates, where $\triangle z = \frac{h}{N}$, and N is the number of cells within the layer. We define the grid points $z_k = k\triangle z$ for $k_{PECbot} \le k \le k_{PECtop}$ where $k=0$ corresponds to $z_{glassbot}$. Similarly, for time, $t_n = n\triangle t$ for $n = 0,1,2...$ where $\triangle t$ is determined by the FDTD method. The value of t_{final} is sufficient

such that the transient wave can propagate completely across the liquid crystal layer with two additional wave periods to allow for averaging to compute the intensity of the transmitted beam.

The electric field components are computed to coincide with the grid points.

$$D_{x,k}^{n} = D_x(z_k, t_n) \tag{1.21}$$

$$D_{y,k}^{n} = D_y(z_k, t_n) \tag{1.22}$$

$$E_{x,k}^{n} = E_x(z_k, t_n) \tag{1.23}$$

$$E_{y,k}^{n} = E_y(z_k, t_n). \tag{1.24}$$

In order to maintain the relationship of Equation (1.17), we also store $\boldsymbol{\varepsilon}$ on the grid. Since $\boldsymbol{\varepsilon}$ is independent of time, we can compute $\boldsymbol{\varepsilon}^{-1}$ prior to the loop for n. Additionally, in order to compute $\boldsymbol{\varepsilon}^{-1}$, we solve Equation (1.4) for $\theta(z)$ and $\psi(z)$ for given boundary conditions using MATLAB's bvp4c algorithm.

By contrast, the magnetic field components are computed to coincide with points half way between grid points in both time and space, further referred to as the half-grid as seen in Figure 6. Again, this allows for the method to be second order but only requires storage of one field component per grid cell.

$$H_{x,k+1/2}^{n+1/2} = H_x(z_{k+1/2}, t_{n+1/2}) \tag{1.25}$$

$$H_{y,k+1/2}^{n+1/2} = H_y(z_{k+1/2}, t_{n+1/2}) \tag{1.26}$$

Both the electric field and magnetic field are initialized to zero throughout the entire computational domain.

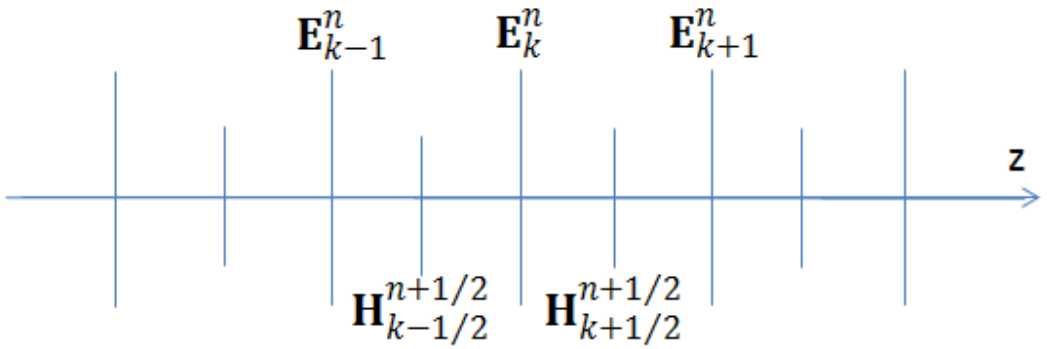

Figure 6. Staggering of grid and half-grid.

We now present the method for maintaining the update scheme current for the value of D_x. We use a second order approximation of Equation (1.14) at $t = t_{n+1/2}$ and $z = z_k$,

$$\left.\frac{\partial D_x}{\partial t}\right|_{z=z_k}^{t=t_{n+1/2}} = -c_0 \left.\frac{\partial H_y}{\partial z}\right|_{z=z_k}^{t=t_{n+1/2}}$$

$$\frac{D_{x,k}^{n+1} - D_{x,k}^{n}}{\Delta t} + O(\Delta t^2) = -c_0 \frac{H_{y,k+1/2}^{n+1/2} - H_{y,k-1/2}^{n+1/2}}{\Delta z} + O(\Delta z^2).$$

The values of $D_{x,k}^{n}$, $H_{y,k+1/2}^{n+1/2}$, and $H_{y,k-1/2}^{n+1/2}$ are known from the previous time step, therefore we can solve for D_x at next time step in order to get an update scheme that is second order in both time and space,

$$D_{x,k}^{n+1} = D_{x,k}^{n} - c_0 \frac{\Delta t}{\Delta z}(H_{y,k+1/2}^{n+1/2} - H_{y,k-1/2}^{n+1/2}) + O(\Delta t^2) + O(\Delta z^2). \qquad (1.27)$$

Similarly for D_y,

$$D_{y,k}^{n+1} = D_{y,k}^{n} - c_0 \frac{\Delta t}{\Delta z}(H_{x,k+1/2}^{n+1/2} - H_{x,k-1/2}^{n+1/2}). \qquad (1.28)$$

From Equation (1.17),

$$E_{x,k}^{n+1} = \varepsilon_{xx,k}^{-1} D_{x,k}^{n+1} + \varepsilon_{xy,k}^{-1} D_{y,k}^{n+1} \qquad (1.29)$$

$$E_{y,k}^{n+1} = \varepsilon_{xy,k}^{-1} D_{x,k}^{n+1} + \varepsilon_{yy,k}^{-1} D_{y,k}^{n+1} \qquad (1.30)$$

where $\boldsymbol{\varepsilon}_k^{-1}$ is the inverse of $\boldsymbol{\varepsilon}$ at z_k. Note, however, that $E_{z,k}^{n+1} = \varepsilon_{zx,k}^{-1} D_{x.k}^{n+1} + \varepsilon_{zy,k}^{-1} D_{y,k}^{n+1}$ is nonzero, although it not required to compute except for when it is necessary to compute $\left|\mathbf{E}_k^{n+1}\right|$.

The update scheme for H is

$$H_{x,k+1/2}^{n+3/2} = H_{x,k+1/2}^{n+1/2} + c_0 \frac{\Delta t}{\Delta z}(E_{y,k+1}^{n+1} - E_{y,k}^{n+1}) \tag{1.31}$$

$$H_{y,k+1/2}^{n+3/2} = H_{y,k+1/2}^{n+1/2} - c_0 \frac{\Delta t}{\Delta z}(E_{x,k+1}^{n+1} - E_{x,k}^{n+1}). \tag{1.32}$$

Slight modifications to this update scheme must be made outside the liquid crystal layer. In the vacuum and PML, Equations (1.29) and (1.30) are simplified since $\boldsymbol{\varepsilon}^{-1}$ is the identity so that $\mathbf{E}_k^n = \mathbf{D}_k^n$. The update schemes for $\mathbf{D}_k^n$ and $\mathbf{H}_{k+1/2}^{n+1/2}$ within the PML are more complicated due to the addition of an absorbing medium.

This scheme applies to both the TF region, where it updates the total field, and the SF region, where it updates the SF, but at the boundaries between the two regions, the scheme must be altered to avoid approximating the spatial derivatives with one total field value and one scattered field value. To update $D_{x,k_{scatbot}}^{n+1}$ at the bottom transition, we need the total field value $H_{y,k_{scatbot-1/2}}^{n+1/2}$, but since this is in the SF region, it represents the value of the scattered field. Therefore we must adjust the equation using the known incident field as

$$D_{x,k_{scatbot}}^{n+1} = D_{x,k_{scatbot}}^{n} - c_0 \frac{\Delta t}{\Delta z}(H_{y,k_{scatbot+1/2}}^{n+1/2} - H_{y,k_{scatbot-1/2}}^{n+1/2} - H_{inc,y,k_{scatbot-1/2}}^{n+1/2})$$

$$D_{y,k_{scatbot}}^{n+1} = D_{y,k_{scatbot}}^{n} + c_0 \frac{\Delta t}{\Delta z}(H_{x,k_{scatbot+1/2}}^{n+1/2} - H_{x,k_{scatbot-1/2}}^{n+1/2} - H_{inc,x,k_{scatbot-1/2}}^{n+1/2}).$$

The magnetic field is similar with the exception that the electric displacement field value that is updated is in the TF region whereas the magnetic field value that is updated is in the SF region. This results in a magnetic field value update

$$H_{x,k_{scatbot}-1/2}^{n+3/2} = H_{x,k_{scatbot}-1/2}^{n+1/2} + c_0 \frac{\Delta t}{\Delta z}(E_{x,k_{scatbot}}^{n+1} - E_{x,k_{scatbot}-1}^{n+1} - E_{inc,x,k_{scatbot}}^{n+1})$$

$$H_{y,k_{scatbot}-1/2}^{n+3/2} = H_{y,k_{scatbot}-1/2}^{n+1/2} - c_0 \frac{\triangle t}{\triangle z}(E_{y,k_{scatbot}}^{n+1} - E_{y,k_{scatbot}-1}^{n+1} - E_{inc,y,k_{scatbot}}^{n+1}).$$

This is similarly done at the top transition.

In order to truncate the computational domain without introducing nonphysical reflections, we must introduce the PML in order to absorb the wave as it leaves the computational domain. The absorption properties of the layer are characterized by the absorption coefficient σ. The behavior of σ and its impact on the portion of the beam of interest are such that we absorb the beam slow enough so that the PML does not act as a hard boundary and reflect the beam. Conversely, it must absorb the beam quick enough to minimize the magnitude of reflection within the layer and in turn cause any unwanted reflections.

Choosing the absorption characteristics of σ allows us to imitate endpoints of the computational domain that go on to infinity.

$$\sigma(x) = \sigma_{\max}\left(\frac{x}{d}\right)^m$$

where x is the depth into the PML and d is the width of the PML.

Again, we present the method for maintaining the update scheme current for the value of D_x, this time however, inside the PML. We use a second order approximation of Equation (1.14) at $t = t_{n+1/2}$ and $z = z_k$,

$$\frac{\partial D_x}{\partial t} + \sigma D_x = -c_0 \frac{\partial H_y}{\partial z}$$

$$\frac{D_{x,k}^{n+1} - D_{x,k}^n}{\triangle t} + \sigma_k \frac{D_{x,k}^{n+1} + D_{x,k}^n}{2} = -c_0 \frac{H_{y,k+1/2}^{n+1/2} - H_{y,k-1/2}^{n+1/2}}{\triangle z}$$

$$D_{x,k}^{n+1} = \frac{2 - \triangle t\sigma_k}{2 + \triangle t\sigma_k} D_{x,k}^n - \frac{2c_0}{2 + \triangle t\sigma_k} \frac{\triangle t}{\triangle z}(H_{y,k+1/2}^{n+1/2} - H_{y,k-1/2}^{n+1/2}). \quad (1.33)$$

Similarly for D_y, H_x, and H_y

$$D_{y,k}^{n+1} = \frac{2 - \triangle t\sigma_k}{2 + \triangle t\sigma_k} D_{x,k}^n - \frac{2c_0}{2 + \triangle t\sigma_k} \frac{\triangle t}{\triangle z}(H_{x,k+1/2}^{n+1/2} - H_{x,k-1/2}^{n+1/2}) \quad (1.34)$$

$$H_{x,k+1/2}^{n+3/2} = \frac{2-\Delta t\sigma_{k+1/2}}{2+\Delta t\sigma_{k+1/2}} H_{x,k+1/2}^{n+1/2} - \frac{2c_0}{2+\Delta t\sigma_{k+1/2}} \frac{\Delta t}{\Delta z}(E_{y,k+1}^{n+1} - E_{y,k}^{n+1}) \tag{1.35}$$

$$H_{y,k+1/2}^{n+3/2} = \frac{2-\Delta t\sigma_{k+1/2}}{2+\Delta t\sigma_{k+1/2}} H_{y,k+1/2}^{n+1/2} - \frac{2c_0}{2+\Delta t\sigma_{k+1/2}} \frac{\Delta t}{\Delta z}(E_{x,k+1}^{n+1} - E_{x,k}^{n+1}). \tag{1.36}$$

The incident beam at $z_{start} = k_{start}\Delta z$ is introduced in the form of

$$\begin{aligned} E_{x,k_{start}}^{n} &= E^{I}\sin(-\omega n\Delta t) \\ E_{y,k_{start}}^{n} &= 0 \end{aligned} \tag{1.37}$$

Time, t, depends on the ε of the material. This wait allows for initialization transient to pass and allow beam to travel one complete period such that intensity, $I = \int_{t}^{t+\frac{2\pi}{\omega}} |\mathbf{E}\times\mathbf{H}|dt$ can be calculated.

In order to calculate the intensity of the transmitted beam, we must first calculate the components of the Poynting vector **S**. Knowing that $H_z = 0$

$$\mathbf{S} = \sqrt{\frac{\mu_0}{\varepsilon_0}}\mathbf{E}\times\mathbf{H} = \sqrt{\frac{\mu_0}{\varepsilon_0}}\begin{bmatrix} -E_z H_x \\ E_z H_y \\ E_x H_y - E_y H_x \end{bmatrix}.$$

This requires that the components be known at the same space and same time. By the very nature of our staggered grid set up, this requires us to interpolate between grid points or half-grid points for **E** or **H** as necessary to compute the value of **S**. We choose to compute **S** on the grid so we must compute **H** to correlate with the space and time of the grid from the half grid. For example in the *z*- component,

$$S_{z,k}^{n} = \sqrt{\frac{\mu_0}{\varepsilon_0}}\left(E_{x,k}^{n}\tilde{H}_{y,k}^{n} - E_{y,k}^{n}\tilde{H}_{x,k}^{n}\right)$$

where $\tilde{\mathbf{H}}$ is the recomputed **H** from the half-grid to the grid and

$$\tilde{\mathbf{H}}_{k}^{n} = 1/4\left(\mathbf{H}_{k+1/2}^{n+1/2} + \mathbf{H}_{k+1/2}^{n-1/2} + \mathbf{H}_{k-1/2}^{n+1/2} + \mathbf{H}_{k-1/2}^{n-1/2}\right).$$

Once **S** is known, we then use the trapezoidal method to approximate the intensity

$$I = \frac{\omega \Delta t}{4\pi}\left(\left|\mathbf{S}_k^{n=T_1}\right| + 2\sum_{n=T_1+1}^{T_{fin}-1} \left|\mathbf{S}_k^{n}\right| + \left|\mathbf{S}_k^{n=T_{fin}}\right| \right)$$

where T_1 represents the start time of the last period. This calculation is computed for a value of k that satisfies $k_{glasstop+1} < k < k_{scattop+1}$.

II. ANALYTICAL SOLUTION AND NUMERICAL VALIDATION

When the same anchoring conditions are used on the top and bottom plates, we refer to Equations (1.4) because these equations have constant solutions throughout the liquid crystal layer. This is to say that when the anchoring conditions are matched on the top and bottom plates, this results in molecule orientations that are homogeneous throughout the LC layer. In this scenario, ψ and θ are constant across the LC layer.

In this case, the electromagnetic problem can be solved analytically. In this chapter, we present a derivation of this solution and then we compare the results to the numerical solution in order to determine the optimal grid cell size for the given h. The basis for determining the optimal grid cell size is to ensure that we solve the problem with reasonable accuracy but also within a reasonable computational time frame.

A. ANALYTICAL SOLUTION

In order to solve the problem analytically, we introduce the known incident beam, with components $\mathbf{E}^I$ and $\mathbf{H}^I$, from Equation (1.37), and four unknown resultant beams that we will solve for in terms of the incident beam. The reflected beam, $\mathbf{E}^R$ and $\mathbf{H}^R$, exists on the same region as the incident beam but travels in the opposite direction. The transmitted beam, $\mathbf{E}^T$ and $\mathbf{H}^T$, exists in the opposite region as the incident beam and travels in the same direction. Internal to the LC layer are two beams. The first is comprised of $\mathbf{E}^{L+}$ and $\mathbf{H}^{L+}$ and travels in the positive z-direction whereas the other beam is comprised of $\mathbf{E}^{L-}$ and $\mathbf{H}^{L-}$ and travels in the negative z-direction within the layer. These are depicted in Figure 7.

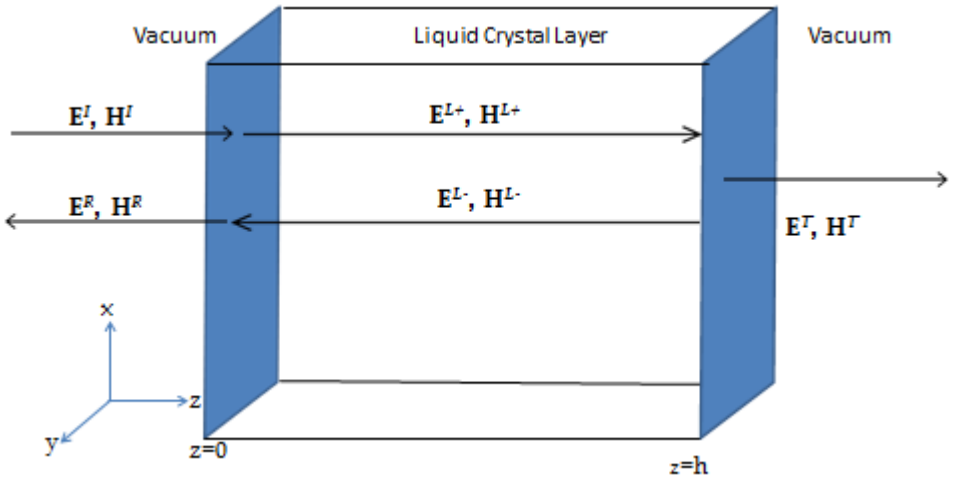

Figure 7. Analytical beam components.

In this section, we will use the complex forms of the electric and magnetic fields and we also introduce a phase shift that allows us to match these results with the results of Equation (1.37). Therefore, our fields take the form

$$\mathbf{E}e^{-i\omega t}e^{-i(k_o z_{start}+\pi/2)}$$

$$\mathbf{D}e^{-i\omega t}e^{-i(k_o z_{start}+\pi/2)}$$

$$\mathbf{H}e^{-i\omega t}e^{-i(k_o z_{start}+\pi/2)}.$$

Plugging these forms into Equations (1.11) through (1.13) gives us the following

$$-i\omega\boldsymbol{\varepsilon}\cdot\mathbf{E}=c_0\nabla\times\mathbf{H} \tag{2.1}$$

$$i\omega\mathbf{H}=c_0\nabla\times\mathbf{E}. \tag{2.2}$$

From Equation (1.37), the incident beam is

$$\mathbf{E}^I=\begin{bmatrix}E_x^I\\0\\0\end{bmatrix}e^{ik_0 z},\quad \mathbf{H}^I=\begin{bmatrix}0\\E_x^I\\0\end{bmatrix}e^{ik_0 z} \tag{2.3}$$

where $k_0=\omega\sqrt{\varepsilon_0\mu_0}=\dfrac{\omega}{c_0}$ is the wave number in vacuum. The transmitted and reflected beams take the forms

$$\mathbf{E}^R = \begin{bmatrix} E_x^R \\ E_y^R \\ 0 \end{bmatrix} e^{-ik_0 z}, \ \mathbf{H}^R = \begin{bmatrix} E_y^R \\ -E_x^R \\ 0 \end{bmatrix} e^{-ik_0 z} \tag{2.4}$$

$$\mathbf{E}^T = \begin{bmatrix} E_x^T \\ E_y^T \\ 0 \end{bmatrix} e^{ik_0 z}, \ \mathbf{H}^T = \begin{bmatrix} -E_y^T \\ E_x^T \\ 0 \end{bmatrix} e^{ik_0 z}. \tag{2.5}$$

To express the unknown beam within the layer, we must first determine the wave numbers and the basis vectors for waves supported by the anisotropic LC medium. We need to define this constant vector $\mathbf{E} = \mathbf{E}_0 e^{ikz}$ and the corresponding $\mathbf{H} = \mathbf{H}_0 e^{ikz}$ for the unknown constant vectors, $\mathbf{E}_0$ and $\mathbf{H}_0$ and the wave vector $\mathbf{k} = (0,0,k)^T$ where k is an unknown wave number that is either positive or negative. Using these forms, Equation (2.1) and Equation (2.2), are now

$$-\omega \boldsymbol{\varepsilon} \cdot \mathbf{E}_0 = c_0 \mathbf{k} \times \mathbf{H}_0 \tag{2.6}$$

$$\omega \mathbf{H}_0 = c_0 \mathbf{k} \times \mathbf{E}_0 \tag{2.7}$$

We now solve Equation (2.7) in terms of $\mathbf{H}_0$ and substitute it into Equation (2.6).

$$-\omega \boldsymbol{\varepsilon} \cdot \mathbf{E}_0 = c_0 \mathbf{k} \times (\frac{c_0}{\omega} \mathbf{k} \times \mathbf{E}_0)$$

$$-\frac{\omega}{c_0^2}^2 \boldsymbol{\varepsilon} \cdot \mathbf{E}_0 = \mathbf{k} \times (\mathbf{k} \times \mathbf{E}_0) = \mathbf{k}(\mathbf{k} \cdot \mathbf{E}_0) - (\mathbf{k} \cdot \mathbf{k})\mathbf{E}_0$$

$$= \mathbf{k}\mathbf{k} \cdot \mathbf{E}_0 - k^2 \mathbf{E}_0$$

$$(k^2 \mathbf{I} - \mathbf{k}\mathbf{k} - \frac{\omega^2}{c_0^2} \boldsymbol{\varepsilon}) \cdot \mathbf{E}_0 = \mathbf{0}. \tag{2.8}$$

The 3x3 matrix, $k^2 \mathbf{I} - \mathbf{k}\mathbf{k} - \frac{\omega^2}{c_0^2} \boldsymbol{\varepsilon}$, must be singular, therefore we can set its determinant to zero and solve for k.

$$\det(k^2 \mathbf{I} - \mathbf{k}\mathbf{k} - \frac{\omega^2}{c_0^2} \boldsymbol{\varepsilon}) = 0$$

$$\varepsilon_{zz}k^4 - \frac{\omega^2}{c_0^2}[(\varepsilon_{xx}+\varepsilon_{yy})\varepsilon_{zz} - \varepsilon_{xy}^2 - \varepsilon_{yz}^2]k^2$$
$$+\frac{\omega^4}{c_0^4}[(\varepsilon_{xx}\varepsilon_{yy} - \varepsilon_{xy}^2)\varepsilon_{zz} - \varepsilon_{xx}\varepsilon_{yz}^2 - \varepsilon_{yy}\varepsilon_{xz}^2 + 2\varepsilon_{xy}\varepsilon_{yz}\varepsilon_{xz}] = 0 \quad .$$

This is a quadratic equation in k^2. Recalling that $\boldsymbol{\varepsilon} = (\varepsilon_\perp \mathbf{I} + \triangle\varepsilon \mathbf{nn})$ and knowing that $\mathbf{n}$ is a unit vector, we arrive at

$$\begin{aligned}
\varepsilon_{xx} &= \varepsilon_\perp + \triangle\varepsilon n_x^2 \\
\varepsilon_{yy} &= \varepsilon_\perp + \triangle\varepsilon n_y^2 \\
\varepsilon_{zz} &= \varepsilon_\perp + \triangle\varepsilon n_z^2 \\
\varepsilon_{xy} &= \triangle\varepsilon n_x n_y \\
\varepsilon_{xz} &= \triangle\varepsilon n_x n_z \\
\varepsilon_{yz} &= \triangle\varepsilon n_y n_z
\end{aligned} \quad .$$

Solving with substitution, we obtain

$$k^2 = \frac{\omega^2}{c_0^2}\frac{2\varepsilon_\perp + \varepsilon_\perp \triangle\varepsilon(1+n_z^2) \pm \varepsilon_\perp \triangle\varepsilon(1-n_z^2)}{2(\varepsilon_\perp + \triangle\varepsilon n_z^2)} .$$

Utilizing $2\varepsilon_\perp^2 + 2\varepsilon_\perp \triangle\varepsilon = 2\varepsilon_\perp(\varepsilon_\perp + \triangle\varepsilon)$ and $\triangle\varepsilon = \varepsilon_\parallel - \varepsilon_\perp$, we get

$$k^2 = \begin{cases} \dfrac{\omega^2\varepsilon_\perp\varepsilon_\parallel}{c_0^2(\varepsilon_\perp + \triangle\varepsilon n_z^2)} = \dfrac{\omega^2\varepsilon_\perp\varepsilon_\parallel}{c_0^2(\varepsilon_\perp + \triangle\varepsilon \sin^2\theta)} = \dfrac{\omega^2\varepsilon_\perp\varepsilon_\parallel}{c_0^2(\varepsilon_\perp \cos^2\theta + \varepsilon_\parallel \sin^2\theta)} \\ \dfrac{\omega^2\varepsilon_\perp}{c_0^2} \end{cases}$$

This results in two distinct wave numbers that the wave can propagate through the LC layer. We start with the ordinary wave number, $k_\perp = \frac{\omega}{c_0}\sqrt{\varepsilon_\perp} = \frac{\omega}{c_0}n_\perp$, where $n_\perp = \sqrt{\varepsilon_\perp}$ is the ordinary index of refraction. The corresponding eigenvector of Equation (2.8) gives us a basis vector for the electric field

$$\mathbf{E}_0 = \mathbf{E}_\perp = \begin{bmatrix} -\sin\psi \\ \cos\psi \\ 0 \end{bmatrix} .$$

The corresponding basis vector for the magnetic field is

$$\mathbf{H}_0 = \mathbf{H}_\perp = \frac{c_0}{\omega}\mathbf{k}\times\mathbf{E}_0 = -n_\perp \begin{bmatrix} \cos\psi \\ \sin\psi \\ 0 \end{bmatrix}.$$

The extraordinary wave number is $k_\theta = \frac{\omega}{c_0}\sqrt{\frac{\varepsilon_\perp \varepsilon_\parallel}{\varepsilon_\perp \cos^2\theta + \varepsilon_\parallel \sin^2\theta}} = \frac{\omega}{c_0} n_\theta$, where $n_\theta = \sqrt{\frac{\varepsilon_\perp \varepsilon_\parallel}{\varepsilon_\perp \cos^2\theta + \varepsilon_\parallel \sin^2\theta}}$ is the extraordinary index of refraction. In the case that $\theta = \frac{\pi}{2}$, $n_\theta = n_\perp$ and so the material is effectively isotropic for this anchoring condition. We have the corresponding basis vector for the electric field

$$\mathbf{E}_0 = \mathbf{E}_\theta = \begin{bmatrix} \cos\psi \\ \sin\psi \\ -\frac{\triangle\varepsilon}{\varepsilon_{zz}}\sin\theta\cos\theta \end{bmatrix}$$

$$\mathbf{H}_0 = \mathbf{H}_\theta = n_\theta \begin{bmatrix} -\sin\psi \\ \cos\psi \\ 0 \end{bmatrix}.$$

These basis vectors allow us to express the electric and magnetic fields within the layer as

$$\mathbf{E}^{L\pm} = E_\perp^{L\pm}\mathbf{E}_\perp e^{\pm ik_\perp z} + E_\theta^{L\pm}\mathbf{E}_\theta e^{\pm ik_\theta z}$$

$$\mathbf{H}^{L\pm} = \pm E_\perp^{L\pm}\mathbf{H}_\perp e^{\pm ik_\perp z} \pm E_\theta^{L\pm}\mathbf{H}_\theta e^{\pm ik_\theta z}.$$

We now have the general solution for the electromagnetic field but we must now apply the boundary conditions at the two interfaces at $z = 0$ and $z = h$. The boundary conditions come from forcing the continuity of the tangential components at the boundaries. That is that $E_x, E_y, H_x,$ and H_y must be continuous at $z = 0$ and $z = h$. This creates a system of eight equations that will be used to determine the eight unknowns of those equations.

We will then look at the equations at $z = 0$.

$$E_x^I + E_x^R = -\sin\psi(E_\perp^{L+} + E_\perp^{L-}) + \cos\psi(E_\theta^{L+} + E_\theta^{L-}) \tag{2.9}$$

$$E_y^R = \cos\psi(E_\perp^{L+} + E_\perp^{L-}) + \sin\psi(E_\theta^{L+} + E_\theta^{L-}) \tag{2.10}$$

$$E_y^R = n_\perp \cos\psi(-E_\perp^{L+} + E_\perp^{L-}) + n_\theta \sin\psi(-E_\theta^{L+} + E_\theta^{L-}) \tag{2.11}$$

$$E_x^I - E_x^R = n_\perp \sin\psi(-E_\perp^{L+} + E_\perp^{L-}) + n_\theta \cos\psi(E_\theta^{L+} - E_\theta^{L-}). \tag{2.12}$$

And similarly, we will look at $z = h$

$$E_x^T e^{ik_0 h} = -\sin\psi(E_\perp^{L+} e^{ik_\perp h} + E_\perp^{L-} e^{-ik_\perp h}) + \cos\psi(E_\theta^{L+} e^{ik_\theta h} + E_\theta^{L-} e^{-ik_\theta h}) \tag{2.13}$$

$$E_y^T e^{ik_0 h} = \cos\psi(E_\perp^{L+} e^{ik_\perp h} + E_\perp^{L-} e^{-ik_\perp h}) + \sin\psi(E_\theta^{L+} e^{ik_\theta h} + E_\theta^{L-} e^{-ik_\theta h}) \tag{2.14}$$

$$\begin{aligned} -E_y^T e^{ik_0 h} = {} & n_\perp \cos\psi(-E_\perp^{L+} e^{ik_\perp h} + E_\perp^{L-} e^{-ik_\perp h}) \\ & + n_\theta \sin\psi(-E_\theta^{L+} e^{ik_\theta h} + E_\theta^{L-} e^{-ik_\theta h}) \end{aligned} \tag{2.15}$$

$$\begin{aligned} E_x^T e^{ik_0 h} = {} & n_\perp \sin\psi(-E_\perp^{L+} e^{ik_\perp h} + E_\perp^{L-} e^{-ik_\perp h}) \\ & + n_\theta \cos\psi(E_\theta^{L+} e^{ik_\theta h} - E_\theta^{L-} e^{-ik_\theta h}) \end{aligned}. \tag{2.16}$$

We will meticulously walk through the derivation of an equation through the summation and subtraction of Equations (2.9) through (2.16). This will allow us to simplify the system of equations to four equations with four unknowns.

Equation (2.13) plus Equation (2.16) results in

$$\begin{aligned} 2E_x^T e^{ik_0 h} = {} & \sin\psi[-E_\perp^{L+} e^{ik_\perp h}(1+n_\perp) - E_\perp^{L-} e^{-ik_\perp h}(1-n_\perp)] \\ & + \cos\psi[E_\theta^{L+} e^{ik_\theta h}(1+n_\theta) + E_\theta^{L-} e^{-ik_\theta h}(1-n_\theta)] \end{aligned}. \tag{2.17}$$

Equation (2.13) minus Equation (2.16) results in

$$\begin{aligned} 0 = {} & -\sin\psi[E_\perp^{L+} e^{ik_\perp h}(1-n_\perp) + E_\perp^{L-} e^{-ik_\perp h}(1+n_\perp)] \\ & + \cos\psi[E_\theta^{L+} e^{ik_\theta h}(1-n_\theta) + E_\theta^{L-} e^{-ik_\theta h}(1+n_\theta)] \end{aligned}. \tag{2.18}$$

Equation (2.14) minus Equation (2.15) results in

$$\begin{aligned} 2E_y^T e^{ik_0 h} = {} & \cos\psi[E_\perp^{L+} e^{ik_\perp h}(1+n_\perp) + E_\perp^{L-} e^{-ik_\perp h}(1-n_\perp)] \\ & + \sin\psi[E_\theta^{L+} e^{ik_\theta h}(1+n_\theta) + E_\theta^{L-} e^{-ik_\theta h}(1-n_\theta)] \end{aligned}. \tag{2.19}$$

Equation (2.14) plus Equation (2.15) results in

$$0 = \cos\psi[E_{\perp}^{L+}e^{ik_{\perp}h}(1-n_{\perp}) + E_{\perp}^{L-}e^{-ik_{\perp}h}(1+n_{\perp})] + \sin\psi[E_{\theta}^{L+}e^{ik_{\theta}h}(1-n_{\theta}) + E_{\theta}^{L-}e^{-ik_{\theta}h}(1+n_{\theta})] \tag{2.20}$$

Equation (2.20) times $\cos\psi$ plus Equation (2.18) times $-\sin\psi$ results in

$$\begin{aligned} 0 &= \sin^2\psi[E_{\perp}^{L+}e^{ik_{\perp}h}(1-n_{\perp}) + E_{\perp}^{L-}e^{-ik_{\perp}h}(1+n_{\perp})] - \cos\psi\sin\psi[E_{\theta}^{L+}e^{ik_{\theta}h}(1-n_{\theta}) + E_{\theta}^{L-}e^{-ik_{\theta}h}(1+n_{\theta})] \\ &+ \cos^2\psi[E_{\perp}^{L+}e^{ik_{\perp}h}(1-n_{\perp}) + E_{\perp}^{L-}e^{-ik_{\perp}h}(1+n_{\perp})] + \cos\psi\sin\psi[E_{\theta}^{L+}e^{ik_{\theta}h}(1-n_{\theta}) + E_{\theta}^{L-}e^{-ik_{\theta}h}(1+n_{\theta})] \\ &= E_{\perp}^{L+}e^{ik_{\perp}h}(1-n_{\perp}) + E_{\perp}^{L-}e^{-ik_{\perp}h}(1+n_{\perp}) \end{aligned}$$

$$E_{\perp}^{L+} = -E_{\perp}^{L-}\frac{e^{-i2k_{\perp}h}(1+n_{\perp})}{(1-n_{\perp})} \tag{2.21}$$

Similarly,

$$E_{\theta}^{L+} = -E_{\theta}^{L-}\frac{e^{-i2k_{\theta}h}(1+n_{\theta})}{(1-n_{\theta})} \tag{2.22}$$

Equation (2.9) plus Equation (2.12) results in

$$2E_x^I = -\sin\psi[E_{\perp}^{L+}(1+n_{\perp}) + E_{\perp}^{L-}(1-n_{\perp})] + \cos\psi[E_{\theta}^{L+}(1+n_{\theta}) + E_{\theta}^{L-}(1-n_{\theta})] \tag{2.23}$$

Equation (2.10) minus Equation (2.11) results in

$$0 = \cos\psi[E_{\perp}^{L+}e^{ik_{\perp}h}(1+n_{\perp}) + E_{\perp}^{L-}e^{-ik_{\perp}h}(1-n_{\perp})] + \sin\psi[E_{\theta}^{L+}e^{ik_{\theta}h}(1+n_{\theta}) + E_{\theta}^{L-}e^{-ik_{\theta}h}(1-n_{\theta})] \tag{2.24}$$

Equation (2.23) times $\cos\psi$ plus Equation (2.24) times $\sin\psi$ results in

$$2E_x^I\cos\psi = E_{\theta}^{L+}(1+n_{\theta}) + E_{\theta}^{L-}(1-n_{\theta}). \tag{2.25}$$

Equation (2.24) times $\cos\psi$ minus Equation (2.23) times $\sin\psi$ results in

$$-2E_x^I\sin\psi = E_{\perp}^{L+}(1+n_{\perp}) + E_{\perp}^{L-}(1-n_{\perp}). \tag{2.26}$$

Substituting Equation (2.21) into Equation (2.26) results in

$$-2E_x^I\sin\psi = E_{\perp}^{L-}[(1-n_{\perp}) - \frac{e^{-i2k_{\perp}h}(1+n_{\perp})^2}{(1-n_{\perp})}].$$

This leads us to a solution for the electric field related to the ordinary index of refraction within the liquid crystal layer in both the positive and negative z-directions

$$E_{\perp}^{L-} = -\frac{2E_x^I \sin\psi(1-n_{\perp})}{(1-n_{\perp})^2 - (1+n_{\perp})^2 e^{-i2k_{\perp}h}}$$

$$E_{\perp}^{L+} = \frac{2E_x^I \sin\psi(1+n_{\perp})}{(1-n_{\perp})^2 e^{i2k_{\perp}h} - (1+n_{\perp})^2}.$$

Similarly for the extraordinary component

$$E_{\theta}^{L-} = \frac{2E_x^I \cos\psi(1-n_{\theta})}{(1-n_{\theta})^2 - (1+n_{\theta})^2 e^{-i2k_{\theta}h}}$$

$$E_{\theta}^{L+} = \frac{-2E_x^I \cos\psi(1+n_{\theta})}{(1-n_{\theta})^2 e^{i2k_{\theta}h} - (1+n_{\theta})^2}.$$

We can now get the transmitted electric field in both the $x-$ and $y-$directions by substituting $E_{\theta}^{L-}, E_{\theta}^{L+}, E_{\perp}^{L-}, E_{\perp}^{L+}$ into Equation (2.13)

$$E_x^T = \frac{-\sin\psi(E_{\perp}^{L+}e^{ik_{\perp}h} + E_{\perp}^{L-}e^{-ik_{\perp}h}) + \cos\psi(E_{\theta}^{L+}e^{ik_{\theta}h} + E_{\theta}^{L-}e^{-ik_{\theta}h})}{e^{ik_0h}}$$

$$= -\frac{\sin\psi}{e^{ik_0h}}\left[\frac{2E_x^I \sin\psi[(1+n_{\perp})-(1-n_{\perp})]}{(1-n_{\perp})^2 e^{ik_{\perp}h} - (1+n_{\perp})^2 e^{-ik_{\perp}h}}\right]$$

$$+\frac{\cos\psi}{e^{ik_0h}}\left[\frac{2E_x^I \cos\psi[(1-n_{\theta})-(1+n_{\theta})]}{(1-n_{\theta})^2 e^{ik_{\theta}h} - (1+n_{\theta})^2 e^{-ik_{\theta}h}}\right]$$

$$= -\frac{4}{e^{ik_0h}}\left[\frac{n_{\perp}E_x^I \sin^2\psi}{(1-n_{\perp})^2 e^{ik_{\perp}h} - (1+n_{\perp})^2 e^{-ik_{\perp}h}} - \frac{n_{\theta}E_x^I \cos^2\psi}{(1-n_{\theta})^2 e^{ik_{\theta}h} - (1+n_{\theta})^2 e^{-ik_{\theta}h}}\right]$$

$$E_x^T = \frac{4E_x^I}{e^{ik_0h}}\left[\frac{n_{\perp}\sin^2\psi}{(1+n_{\perp})^2 e^{-ik_{\perp}h} - (1-n_{\perp})^2 e^{ik_{\perp}h}} + \frac{n_{\theta}\cos^2\psi}{(1+n_{\theta})^2 e^{-ik_{\theta}h} - (1-n_{\theta})^2 e^{ik_{\theta}h}}\right]$$

Similarly,

$$E_y^T = \frac{4E_x^I}{e^{ik_0h}}\left[\frac{n_{\theta}\cos\psi\sin\psi}{(1+n_{\theta})^2 e^{-ik_{\theta}h} - (1-n_{\theta})^2 e^{ik_{\theta}h}} - \frac{n_{\perp}\cos\psi\sin\psi}{(1+n_{\perp})^2 e^{-ik_{\perp}h} - (1-n_{\perp})^2 e^{ik_{\perp}h}}\right]$$

We now have solutions for the electric field of the transmitted beam with respect to the incident beam which is a known quantity. Our next step is to derive the intensity of the transmitted beam with respect to known quantities.

$$I = \frac{1}{2\pi/\omega}\int_{t}^{t+\frac{2\pi}{\omega}} \left|E \times H\right| dt$$

$$I^T = \frac{1}{2}\sqrt{\frac{\varepsilon_o}{\mu_o}}\underline{E}^T \bullet \underline{E}^{T*} = \frac{1}{2}\sqrt{\frac{\varepsilon_o}{\mu_o}}(E_x^T E_x^{T*} + E_y^T E_y^{T*})$$

$$\begin{aligned}
I^T = \frac{16}{2}\sqrt{\frac{\varepsilon_o}{\mu_o}}(E_x^I)^2 \Bigg(& \left[\frac{n_\perp \sin^2\psi}{(1+n_\perp)^2 e^{-ik_\perp h} - (1-n_\perp)^2 e^{ik_\perp h}} + \frac{n_\theta \cos^2\psi}{(1+n_\theta)^2 e^{-ik_\theta h} - (1-n_\theta)^2 e^{ik_\theta h}}\right] \\
& \times \left[\frac{n_\perp \sin^2\psi}{(1+n_\perp)^2 e^{ik_\perp h} - (1-n_\perp)^2 e^{-ik_\perp h}} + \frac{n_\theta \cos^2\psi}{(1+n_\theta)^2 e^{ik_\theta h} - (1-n_\theta)^2 e^{-ik_\theta h}}\right] \\
& + \left[\frac{n_\theta \sin\psi\cos\psi}{(1+n_\theta)^2 e^{-ik_\theta h} - (1-n_\theta)^2 e^{ik_\theta h}} - \frac{n_\perp \cos\psi\sin\psi}{(1+n_\perp)^2 e^{-ik_\perp h} - (1-n_\perp)^2 e^{ik_\perp h}}\right] \\
& \times \left[\frac{n_\theta \sin\psi\cos\psi}{(1+n_\theta)^2 e^{ik_\theta h} - (1-n_\theta)^2 e^{-ik_\theta h}} - \frac{n_\perp \cos\psi\sin\psi}{(1+n_\perp)^2 e^{-ik_\perp h} - (1-n_\perp)^2 e^{ik_\perp h}}\right]\Bigg) \\
= 8\sqrt{\frac{\varepsilon_o}{\mu_o}}(E_x^I)^2 & \left[\frac{{n_\perp}^2 \sin^2\psi}{(1+n_\perp)^4 - (1+n_\perp)^2(1-n_\perp)^2(e^{i2k_\perp h} + e^{-i2k_\perp h}) + (1-n_\perp)^4}\right. \\
& \left. + \frac{{n_\theta}^2 \cos^2\psi}{(1+n_\theta)^2 - (1+n_\theta)^2(1-n_\theta)^2(e^{i2k_\theta h} + e^{-i2k_\theta h}) + (1-n_\theta)^4}\right] \\
= 4\sqrt{\frac{\varepsilon_o}{\mu_o}}(E_x^I)^2 & \left[\frac{{n_\perp}^2 \sin^2\psi}{1+6{n_\perp}^2 + {n_\perp}^4 - (1-n_\perp)^2\cos(2k_\perp h)} + \frac{{n_\theta}^2 \cos^2\psi}{1+6{n_\theta}^2 + {n_\theta}^4 - (1-{n_\theta}^2)^2\cos(2k_\theta h)}\right] \\
= 8I^I & \left[\frac{{n_\perp}^2 \sin^2\psi}{1+6{n_\perp}^2 + {n_\perp}^4 - (1-n_\perp)^2\cos(2k_\perp h)} + \frac{{n_\theta}^2 \cos^2\psi}{1+6{n_\theta}^2 + {n_\theta}^4 - (1-{n_\theta}^2)^2\cos(2k_\theta h)}\right]
\end{aligned}$$

where the intensity of the incident beam is $I^I = \frac{1}{2}\sqrt{\frac{\varepsilon_o}{\mu_o}}(E_x^I)^2$. Solving for the remaining unknowns with respect to known values and summarizing those already solved for, we have

$$\begin{aligned}
I^T = 8I^I & \left[\frac{{n_\perp}^2 \sin^2\psi}{1+6{n_\perp}^2 + {n_\perp}^4 - (1-n_\perp)^2\cos(2k_\perp h)}\right. \\
& \left. + \frac{{n_\theta}^2 \cos^2\psi}{1+6{n_\theta}^2 + {n_\theta}^4 - (1-{n_\theta}^2)^2\cos(2k_\theta h)}\right]
\end{aligned} \tag{2.27}$$

$$E_x^T = \frac{4E_x^I}{e^{ik_0 h}}\left[\frac{n_\perp \sin^2\psi}{(1+n_\perp)^2 e^{-ik_\perp h} - (1-n_\perp)^2 e^{ik_\perp h}} + \frac{n_\theta \cos^2\psi}{(1+n_\theta)^2 e^{-ik_\theta h} - (1-n_\theta)^2 e^{ik_\theta h}}\right] \tag{2.28}$$

$$E_y^T = \frac{4E_x^I}{e^{ik_0 h}}\left[\frac{n_\theta \cos\psi \sin\psi}{(1+n_\theta)^2 e^{-ik_\theta h} - (1-n_\theta)^2 e^{ik_\theta h}} - \frac{n_\perp \cos\psi \sin\psi}{(1+n_\perp)^2 e^{-ik_\perp h} - (1-n_\perp)^2 e^{ik_\perp h}}\right] \tag{2.29}$$

$$E_\perp^{L-} = -\frac{2E_x^I \sin\psi(1-n_\perp)}{(1-n_\perp)^2 - (1+n_\perp)^2 e^{-i2k_\perp h}} \tag{2.30}$$

$$E_\perp^{L+} = \frac{2E_x^I \sin\psi(1+n_\perp)}{(1-n_\perp)^2 e^{i2k_\perp h} - (1+n_\perp)^2} \tag{2.31}$$

$$E_\theta^{L+} = \frac{-2E_x^I \cos\psi(1+n_\theta)}{(1-n_\theta)^2 e^{i2k_\theta h} - (1+n_\theta)^2} \tag{2.32}$$

$$E_\theta^{L-} = \frac{2E_x^I \cos\psi(1-n_\theta)}{(1-n_\theta)^2 - (1+n_\theta)^2 e^{-i2k_\theta h}} \tag{2.33}$$

$$E_x^R = \left[\frac{(1-n_\theta^{\,2})E_x^I \cos^2\psi(1-e^{-2ik_\theta h})}{(1-n_\theta)^2 - (1+n_\theta)^2 e^{-2ik_\theta h}} - \frac{(1-n_\perp^{\,2})E_x^I \sin^2\psi(1-e^{i2k_\perp h})}{(1-n_\perp)^2 e^{i2k_\perp h} - (1+n_\perp)^2}\right] \tag{2.34}$$

$$E_y^R = E_x^I \cos\psi \sin\psi\left[\frac{(1-n_\theta^{\,2})(1-e^{-i2k_\theta h})}{(1-n_\theta)^2 - (1+n_\theta)^2 e^{-2ik_\theta h}} + \frac{(1-n_\perp^{\,2})(1-e^{i2k_\perp h})}{(1-n_\perp)^2 e^{i2k_\perp h} - (1+n_\perp)^2}\right] \tag{2.35}$$

Equation (2.27) gives an explicit formula for the intensity of the transmitted beam. We will use it to validate our FDTD codes in the next section.

B. VALIDATION OF NUMERICAL CODE

Before we can compare the analytical and numerical solutions, we must first determine, through the FDTD method, the optimal mesh size that takes into account the relationship between our time step and our spatial step that will give accurate results. We compute multiple transmitted intensities with a varying mesh grid size and show them in Figure 8. This illustrates that all solutions converge to a mesh grid with a value of grid points approximately N=3100. It is through this that we can say with confidence that the 3200 grid points will provide a solution that is consistently accurate. However, we must also look at acceptable error and the time required computing the numerical solution.

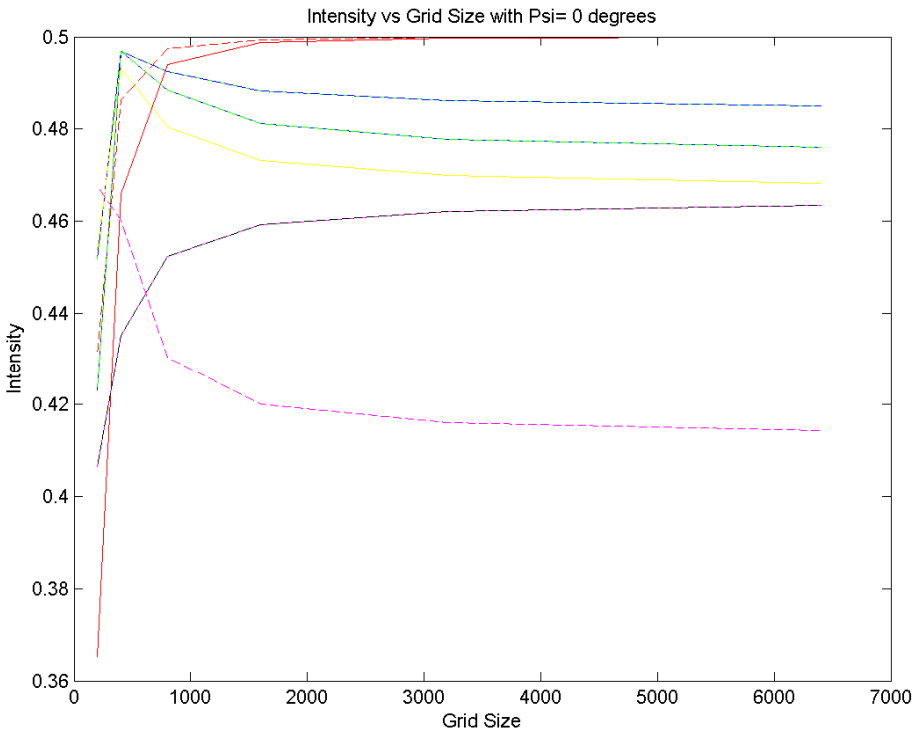

Figure 8. Transmitted intensity as compared to grid spacing with $\psi = 0$.

Additionally, we look at the overall error between the known true solution and the numerical solution for different grid mesh sizes. Table 1 is a collection of various values of intensity errors between the true value and the numerical value for the given N, ψ,

and where θ varies from 0 to $\pi/2$ in increments of $\pi/20$. We have provided the maximum, minimum and average values for each of the values of *N*. Additionally, we have provided the percentage error which is close to the value of the difference in intensity.

		Mesh to True Solution Error					Mesh to True Solution Percent Error				
(ψ,θ)		200	400	800	1600	3200	200	400	800	1600	3200
$(0,\theta)$	MIN	0.0232	0.0037	0.0052	0.0012	0.0004	2.49%	0.47%	0.52%	0.12%	0.04%
	MAX	0.2667	0.0797	0.0294	0.0135	0.0064	26.67%	8.01%	3.19%	1.40%	0.67%
	AVG	0.1408	0.0479	0.0183	0.0077	0.0035	15.18%	5.15%	1.98%	0.83%	0.38%
$(\pi/20,\theta)$	MIN	0.0259	0.0043	0.0052	0.0012	0.0004	2.77%	0.53%	0.52%	0.12%	0.04%
	MAX	0.2635	0.0784	0.0288	0.0131	0.0063	26.35%	7.88%	3.12%	1.36%	0.65%
	AVG	0.1389	0.0470	0.0179	0.0075	0.0034	14.92%	5.03%	1.93%	0.81%	0.37%
$(\pi/10,\theta)$	MIN	0.0338	0.0060	0.0052	0.0012	0.0004	3.38%	0.60%	0.52%	0.12%	0.04%
	MAX	0.2543	0.0747	0.0271	0.0121	0.0058	25.43%	7.47%	2.71%	1.21%	0.58%
	AVG	0.1334	0.0441	0.0167	0.0070	0.0032	13.34%	4.41%	1.67%	0.70%	0.32%
$(3\pi/20,\theta)$	MIN	0.0375	0.0086	0.0052	0.0012	0.0004	3.38%	0.60%	0.52%	0.12%	0.04%
	MAX	0.2399	0.0690	0.0245	0.0105	0.0050	25.43%	7.47%	2.71%	1.21%	0.58%
	AVG	0.1247	0.0397	0.0149	0.0062	0.0028	13.34%	4.41%	1.67%	0.70%	0.32%
$(\pi/5,\theta)$	MIN	0.0075	0.0096	0.0052	0.0012	0.0004	0.75%	0.96%	0.52%	0.12%	0.04%
	MAX	0.2218	0.0618	0.0211	0.0089	0.0041	22.18%	6.18%	2.11%	0.89%	0.41%
	AVG	0.1137	0.0341	0.0125	0.0051	0.0023	11.37%	3.41%	1.25%	0.51%	0.23%
$(\pi/4,\theta)$	MIN	0.0128	0.0010	0.0040	0.0012	0.0004	1.28%	0.10%	0.40%	0.12%	0.04%
	MAX	0.2016	0.0538	0.0174	0.0071	0.0032	20.16%	5.38%	1.74%	0.71%	0.32%
	AVG	0.1085	0.0279	0.0100	0.0040	0.0018	10.85%	2.79%	1.00%	0.40%	0.18%
$(3\pi/10,\theta)$	MIN	0.0190	0.0002	0.0012	0.0008	0.0004	1.90%	0.02%	0.12%	0.08%	0.04%
	MAX	0.1814	0.0457	0.0137	0.0053	0.0024	18.14%	4.57%	1.37%	0.53%	0.24%
	AVG	0.1126	0.0231	0.0074	0.0029	0.0013	11.26%	2.31%	0.74%	0.29%	0.13%
$(7\pi/20,\theta)$	MIN	0.0651	0.0102	0.0007	0.0000	0.0001	6.51%	1.02%	0.07%	0.00%	0.01%
	MAX	0.1631	0.0384	0.0103	0.0036	0.0016	16.31%	3.84%	1.03%	0.36%	0.16%
	AVG	0.1216	0.0246	0.0052	0.0019	0.0008	12.16%	2.46%	0.52%	0.19%	0.08%
$(2\pi/5,\theta)$	MIN	0.1020	0.0192	0.0019	0.0000	0.0001	10.20%	1.92%	0.19%	0.00%	0.01%
	MAX	0.1484	0.0326	0.0076	0.0023	0.0010	14.84%	3.26%	0.76%	0.23%	0.10%
	AVG	0.1288	0.0261	0.0049	0.0012	0.0005	12.88%	2.61%	0.49%	0.12%	0.05%
$(9\pi/20,\theta)$	MIN	0.1262	0.0251	0.0043	0.0008	0.0002	12.62%	2.51%	0.43%	0.08%	0.02%
	MAX	0.1388	0.0288	0.0058	0.0015	0.0005	13.88%	2.88%	0.58%	0.15%	0.05%
	AVG	0.1334	0.0270	0.0051	0.0011	0.0004	13.34%	2.70%	0.51%	0.11%	0.04%
$(\pi/2,\theta)$	MIN	0.1352	0.0274	0.0052	0.0011	0.0004	13.52%	2.74%	0.52%	0.11%	0.04%
	MAX	0.1352	0.0274	0.0052	0.0012	0.0004	13.52%	2.74%	0.52%	0.12%	0.04%
	AVG	0.1352	0.0274	0.0052	0.0012	0.0004	13.52%	2.74%	0.52%	0.12%	0.04%

Table 1. Absolute and percent error between the true intensity and the numerical intensity for the given N, ψ, and θ.

The information in Table 1 can be summarized in Table 2 such that a more reasonable comparison can be done for which value of *N* to choose. In Table 2, we have averaged all of the minimum, maximum and average values for the given *N* as well as list the maximum and minimum values for that value of *N*.

	Mesh to True Solution Error (ψ,θ)					Mesh to True Solution Percent Error				
	200	400	800	1600	3200	200	400	800	1600	3200
AVG MIN	0.0511	0.0101	0.0035	0.0008	0.0003	5.09%	0.99%	0.35%	0.08%	0.03%
AVG MAX	0.1774	0.0465	0.0147	0.0060	0.0028	17.87%	4.71%	1.52%	0.62%	0.29%
AVG AVG	0.1139	0.0293	0.0091	0.0035	0.0015	11.57%	3.00%	0.94%	0.36%	0.16%
MIN	0.0075	0.0002	0.0007	0.0000	0.0001	0.75%	0.02%	0.07%	0.00%	0.01%
MAX	0.2667	0.0797	0.0294	0.0135	0.0064	26.67%	8.01%	3.19%	1.40%	0.67%

Table 2. Summary of intensity error and intensity percent error from Table 1.

The error introduced by selecting a value of *N=800* is an acceptable error for purposes of evaluating the overall impact of varying our anchoring conditions. All figures, unless otherwise stated, are generated using a mesh grid with *N*=800.

We have shown that we are able to compute the known true solution for the electromagnetic wave with matched anchoring conditions. We show through the FDTD method and computer computation time requirements that the most efficient mesh grid spacing for accuracy to the known true solution is *N*=3200 but this is not the most efficient when compared to acceptable error and time computational requirements. Figure 9. depicts the comparison of the normalized intensity of both the numerical and analytical solution using a mesh grid with *N*=3200. This is achieved by setting $\psi = 0$ and varying θ from 0 to $\pi/2$. It should also be noted that the graph depicts multiple values of θ that produce a maximum intensity but only one value that produces a global minimum intensity. We will look at this closer later.

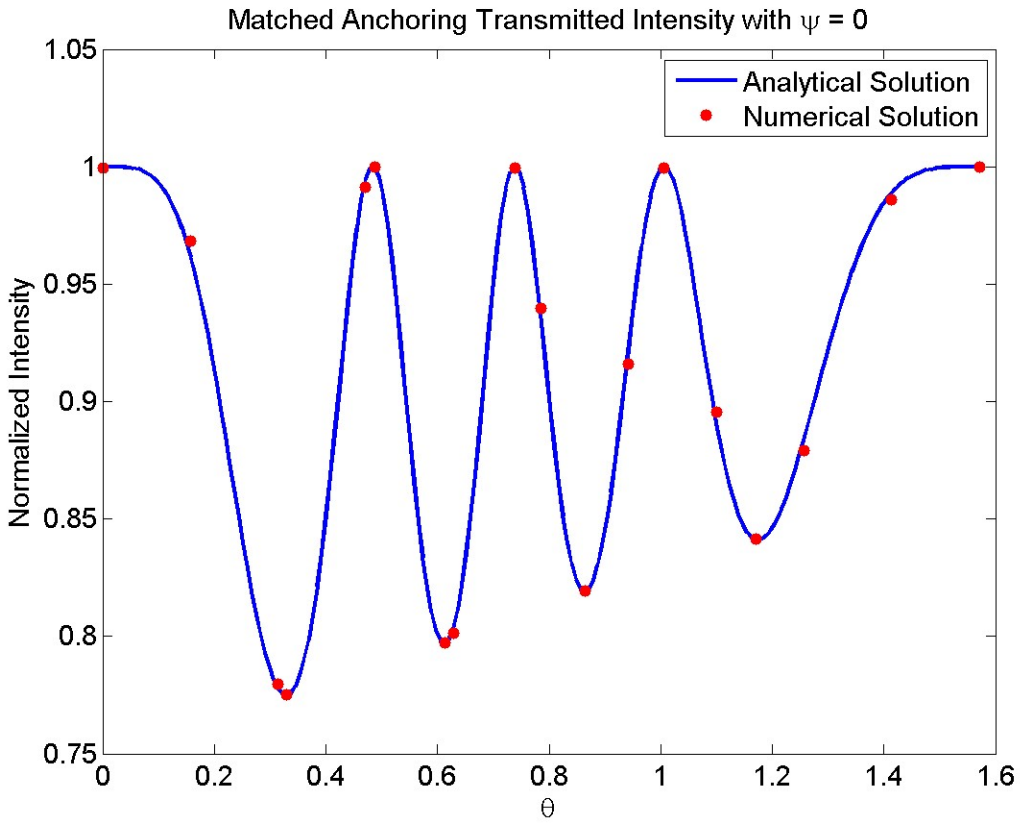

Figure 9. True versus numerical solution.

THIS PAGE INTENTIONALLY LEFT BLANK

III. NUMERICAL SIMULATIONS

A. MATCHED PSI

As a point of convention, we will use the term intensity and transmitted intensity to mean the same intensity of the plane wave. Otherwise we will clarify the intensity of point. We start first by stepping away from the known solution of matched, $\psi_{top} = \psi_{bot}$ and $\theta_{top} = \theta_{bot}$, anchoring conditions and shift to a combination of mixed, $\psi_{top} \neq \psi_{bot}$ and/or $\theta_{top} \neq \theta_{bot}$, and matched anchoring conditions on both the top and bottom of the LC layer. At this point we can no longer utilize the true analytical solution in our process as it is not valid for mixed anchoring conditions. We have set $\psi = 0$ and maintain that value for the duration of each individual data run.

Figures 10 through 19 are similar in that the value of ψ is held constant and each individual curve on the specific graph is represented by a value of θ_{top} ranging from 0 to $\pi/2$ in increments of $\pi/20$. While holding θ_{top} at that value, we vary θ_{bot} from 0 to $\pi/2$ in increments of $\pi/100$. The important point to take away from the curve is not the data itself but more as to what happens to the shape and relative relationship of the curves as ψ is changed while maintaining matched conditions.

It is from Figure 10 through Figure 19 that we see that the relative relationship between the curves on one graph do not change as compared to those relative relationships on another graph. What we see is the effective compression of the curves and the minimum intensity going up as ψ increases. We also note that the maximum value of transmitted intensity does not change. It is seen that as the value of ψ approaches $\psi = \pi/2$, the intensity solution converges to a value of one. This means that the layer is essentially transparent to the beam with that specific polarization.

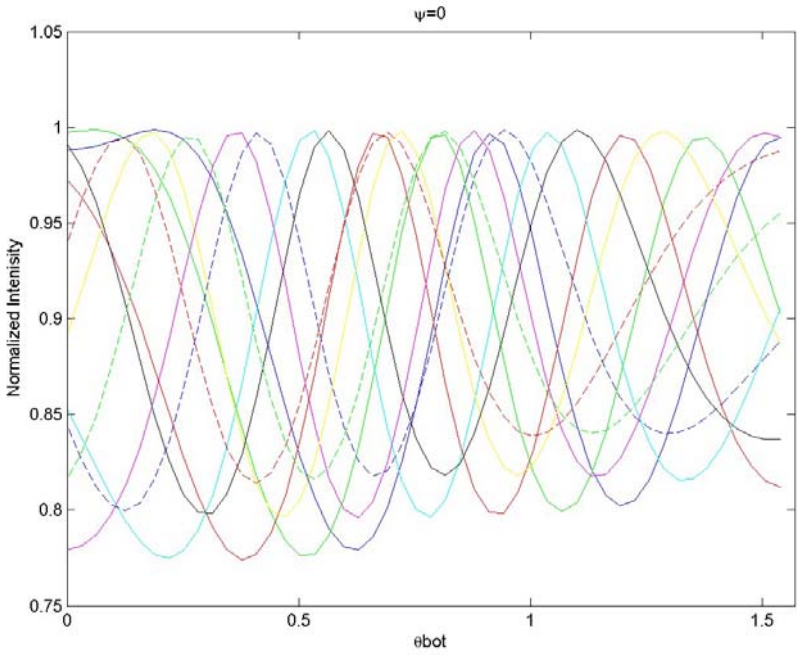

Figure 10. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

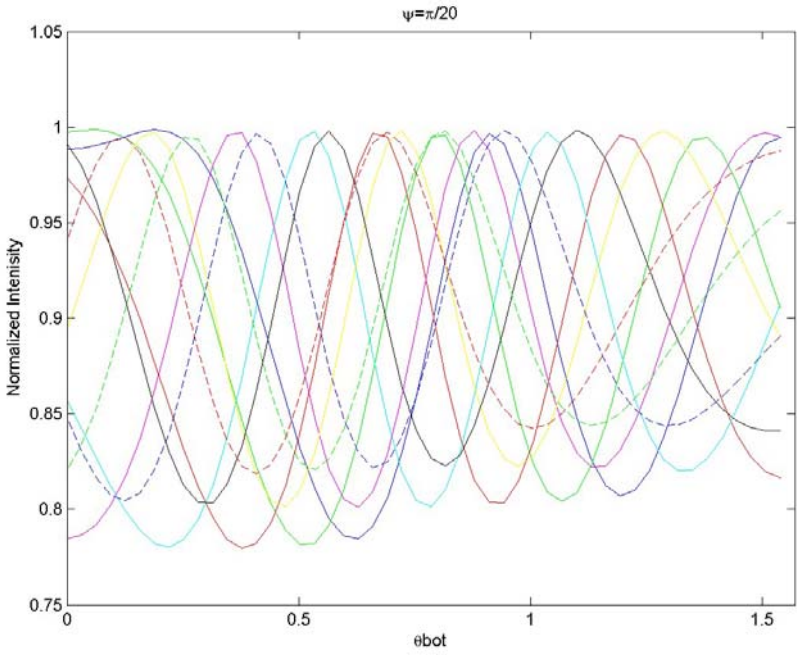

Figure 11. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

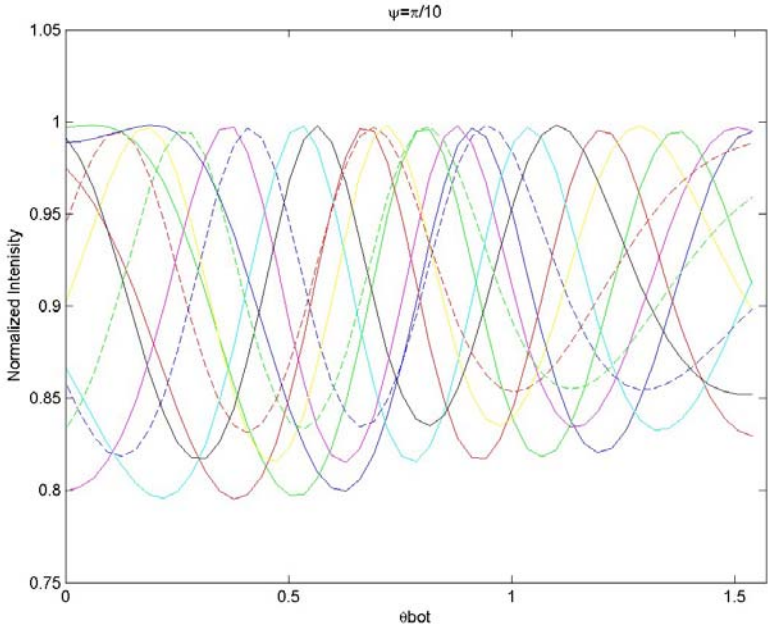

Figure 12. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

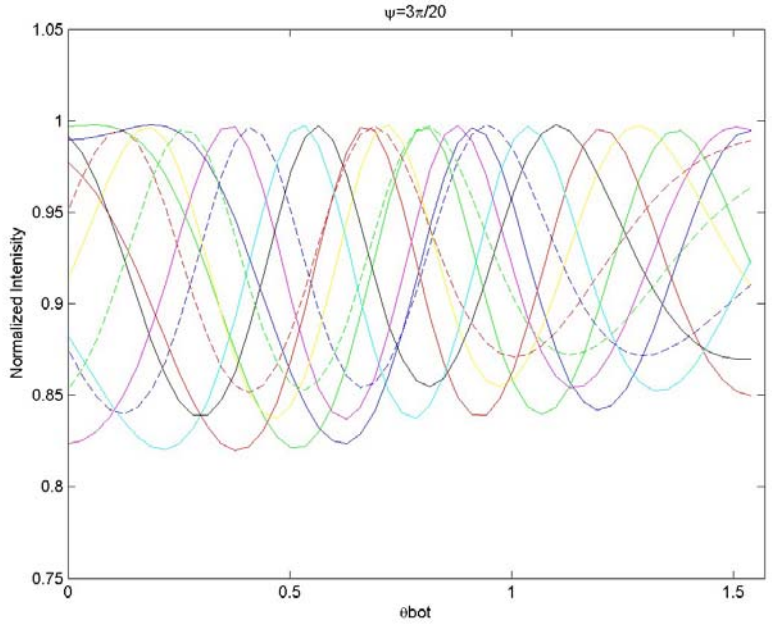

Figure 13. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

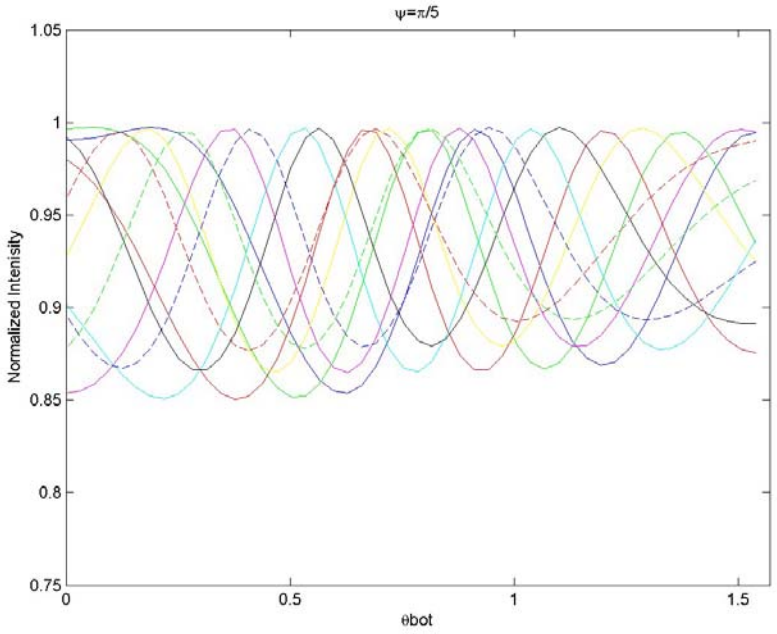

Figure 14. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

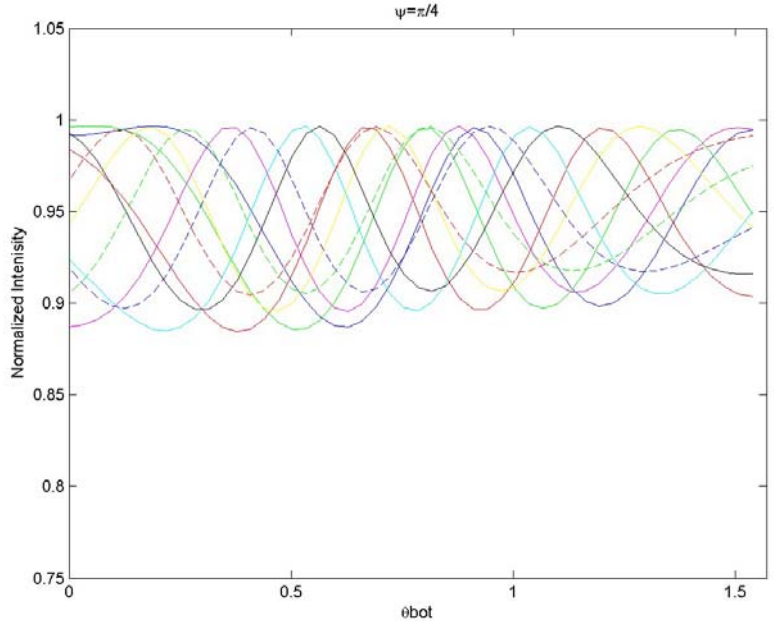

Figure 15. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

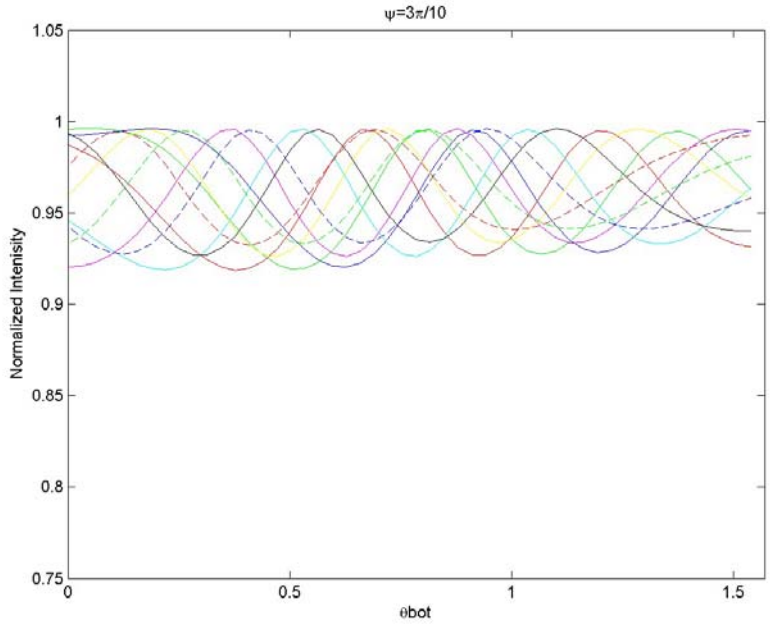

Figure 16. Transmitted intensity - ψ matched with θ_{bot} ranging from to $\pi/2$ in $\pi/20$ increments.

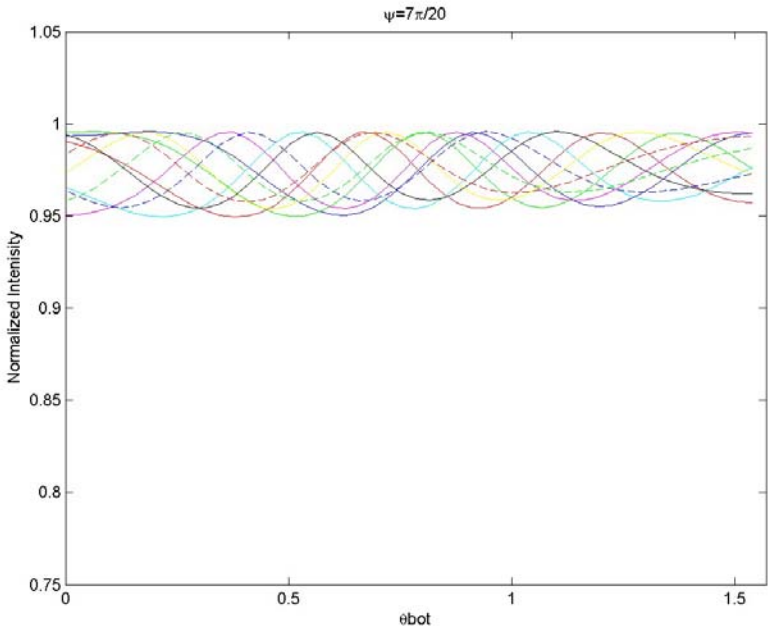

Figure 17. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

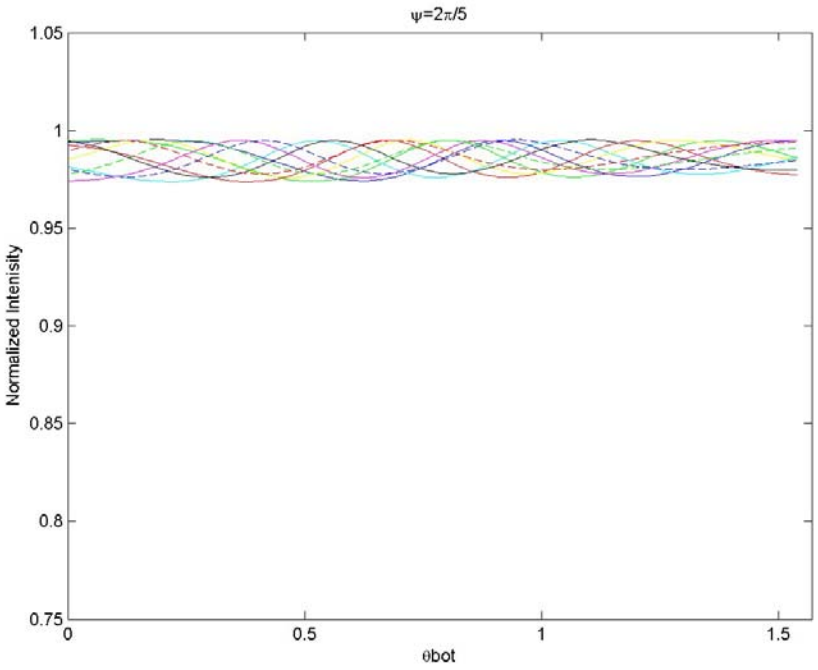

Figure 18. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

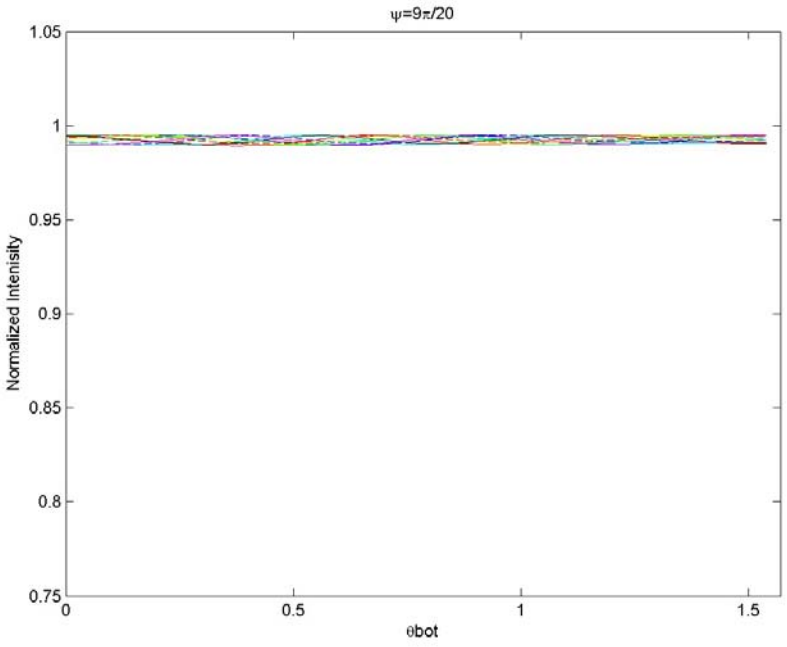

Figure 19. Transmitted intensity - ψ matched with θ_{bot} ranging from to $\pi/2$ in $\pi/20$ increments.

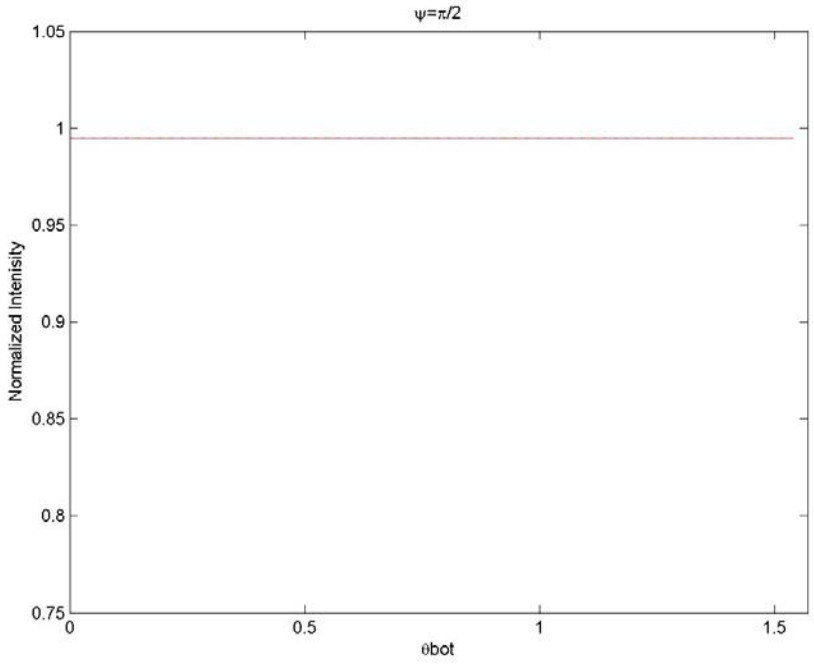

Figure 20. Transmitted intensity - ψ matched with θ_{bot} ranging from 0 to $\pi/2$ in $\pi/20$ increments.

This is a significant finding in that anytime we have $\psi = \pi/2$, as shown in Figure 20, the intensity solution is independent of θ.

We now move forward and investigate the effects of setting the value of θ_{top} to a fixed value, vary θ_{bot} from 0 to $\pi/2$ in fifty increments and vary the matched value of ψ. The results as shown in Figure 21 through Figure 31. show this relationship with each graph representing a different value of θ_{top}. The curve of $\psi = \pi/2$ is consistently at the top represented by the dashed blue line. Additionally the curve of $\psi = 0$ is consistently at the bottom of the collection of curves. These two curves represent the upper and lower bounds of the solution and this is consistent with the solution for intensity as shown in Equation (2.27).

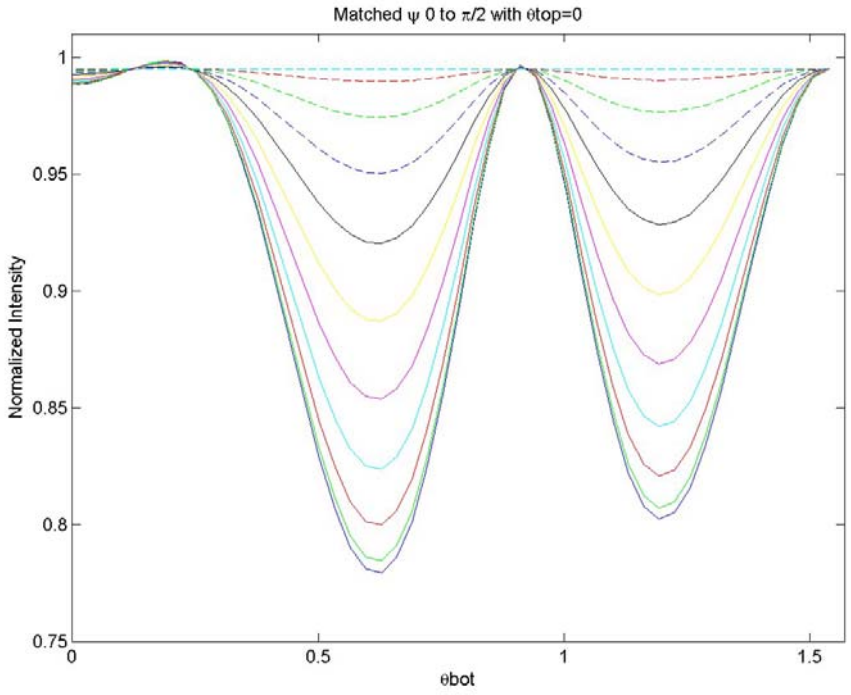

Figure 21. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

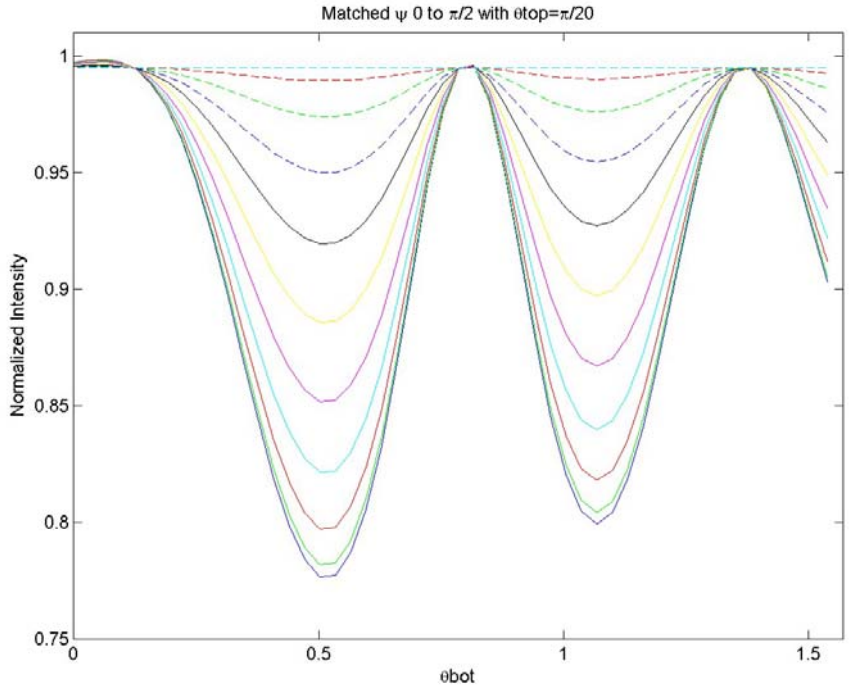

Figure 22. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

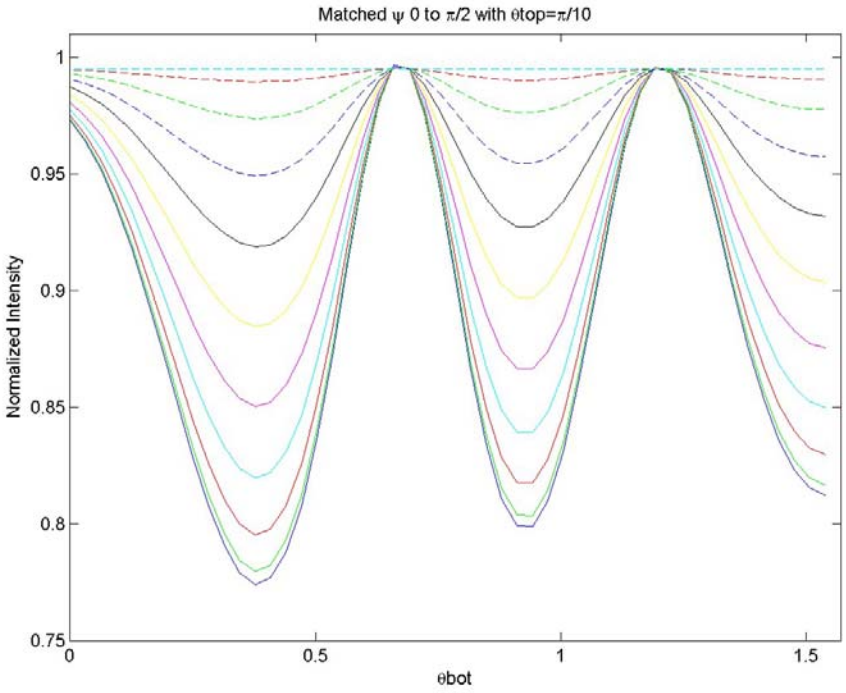

Figure 23. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

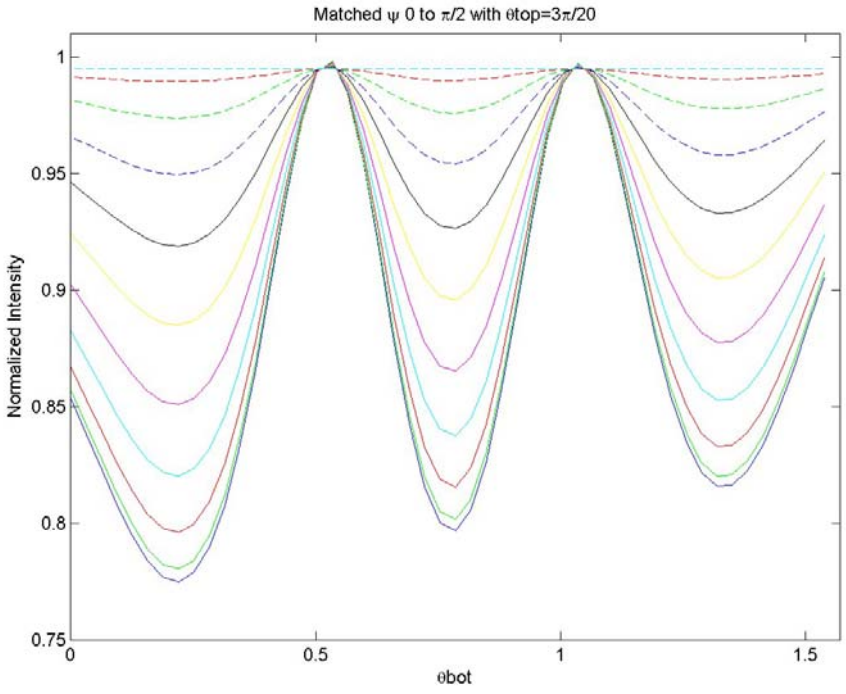

Figure 24. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

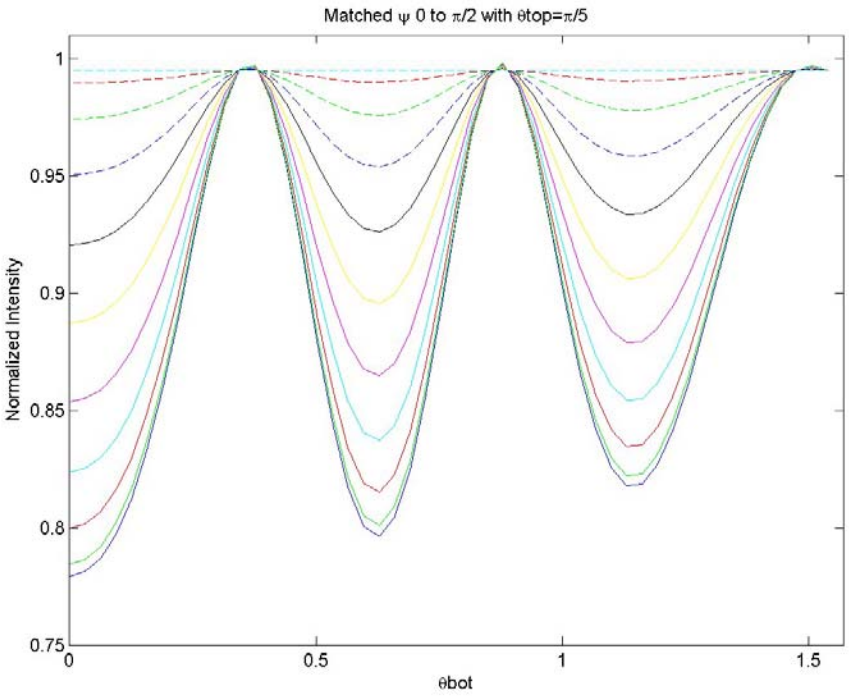

Figure 25. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

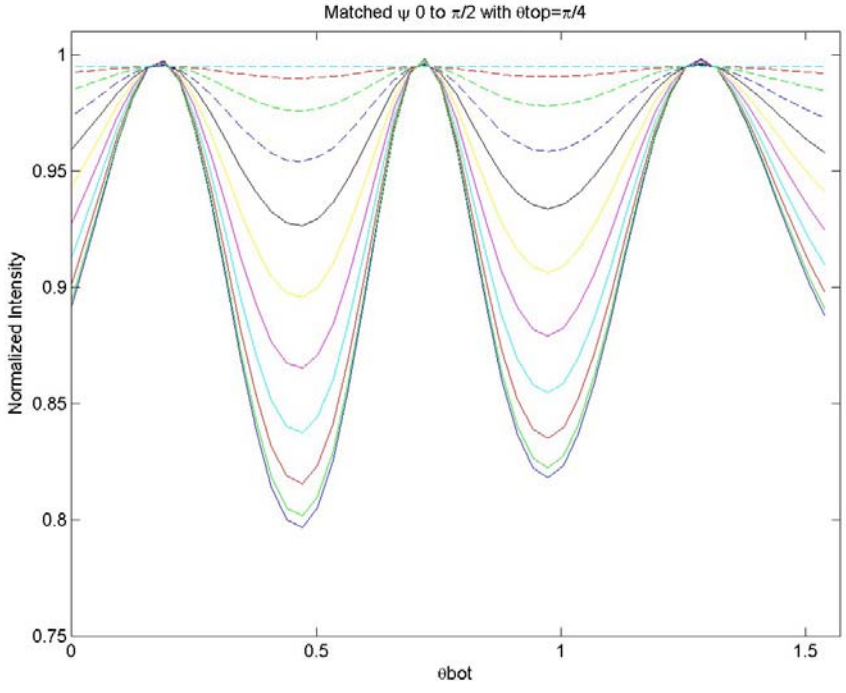

Figure 26. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

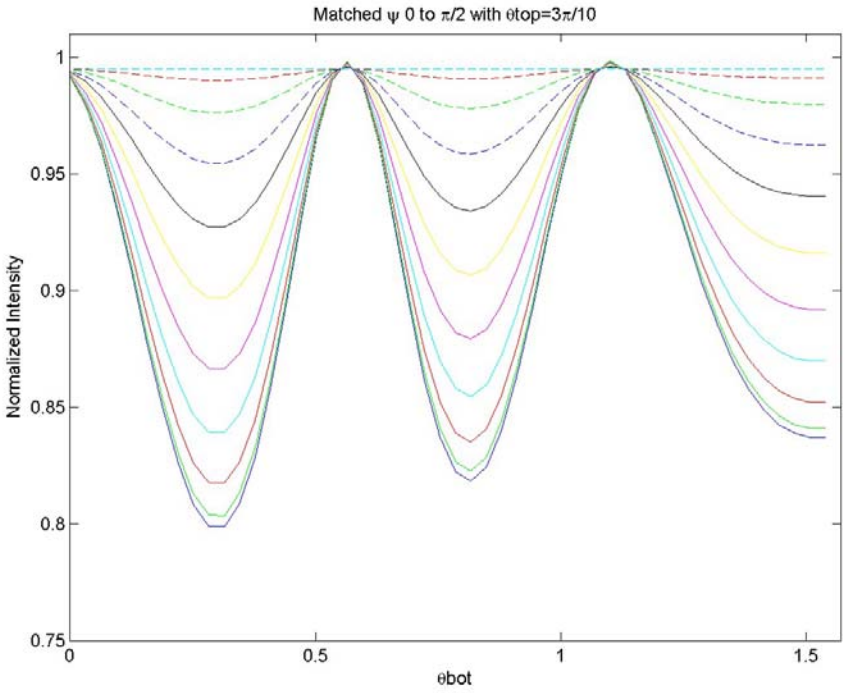

Figure 27. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

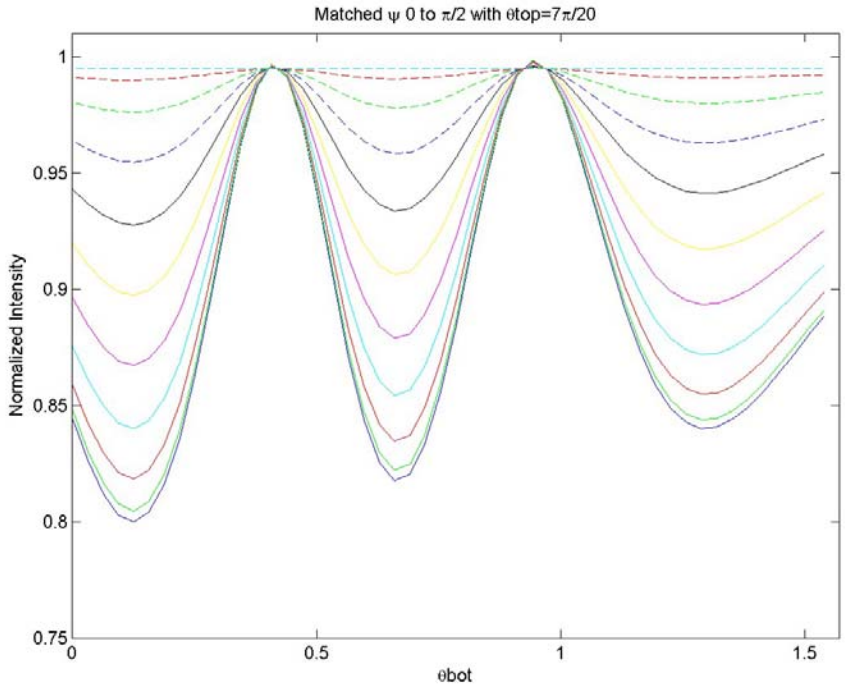

Figure 28. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

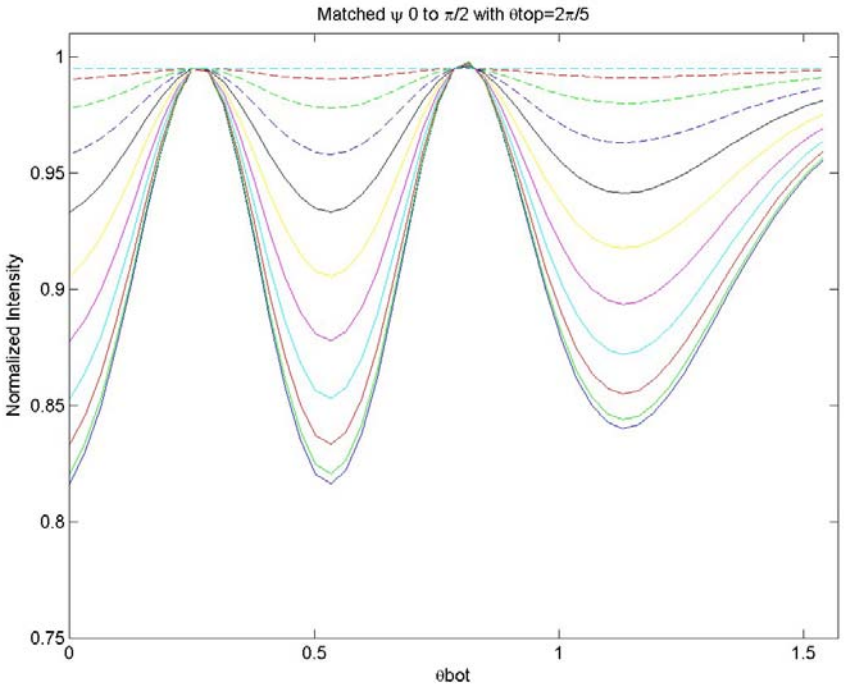

Figure 29. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

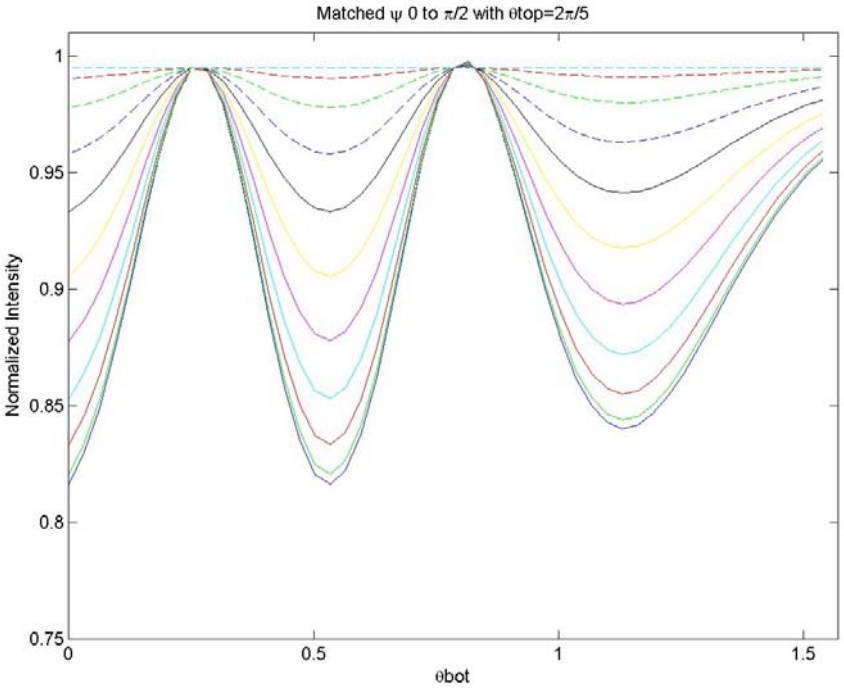

Figure 30. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

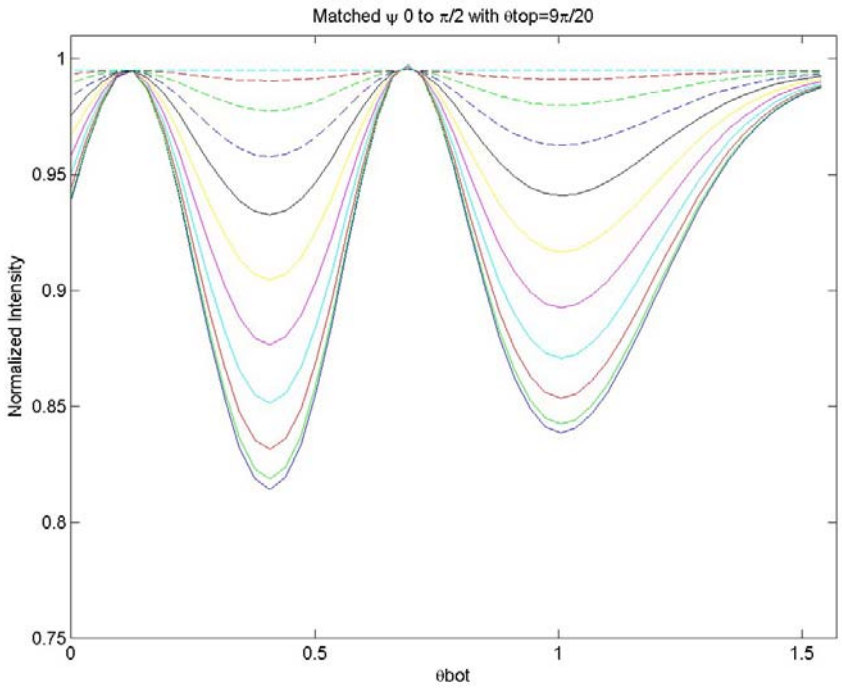

Figure 31. Normalized transmitted intensity for $\psi = 0$ to $\psi = \pi / 2$ and θ_{top} fixed.

In a focus to reduce the intensity of the transmitted beam, we can therefore utilize the lower bound of the scenario and for matched ψ, we see that we get the maximum intensity reduction at $\psi = 0$ for any value or combination of θ as compared to any other matched value for ψ. This allows us to, is the case of matched ψ, to set its value to 0 and remove it as a variable impacting intensity. This then means that the overall intensity reduction is dependent on the value of θ.

By plotting both a surface plot and contour plot as seen in Figure 32 and Figure 33, we see that there is a global minimum at lower values of θ. The graphs are utilizing the same data which consists of the calculated intensity with $\psi = 0$ and 50 data points for both θ_{top} and θ_{bot}. There is also a curvature of the valleys at the individual minimums as well as curvature at the peaks of Figure 32. This curvature can also be seen in Figure 33.

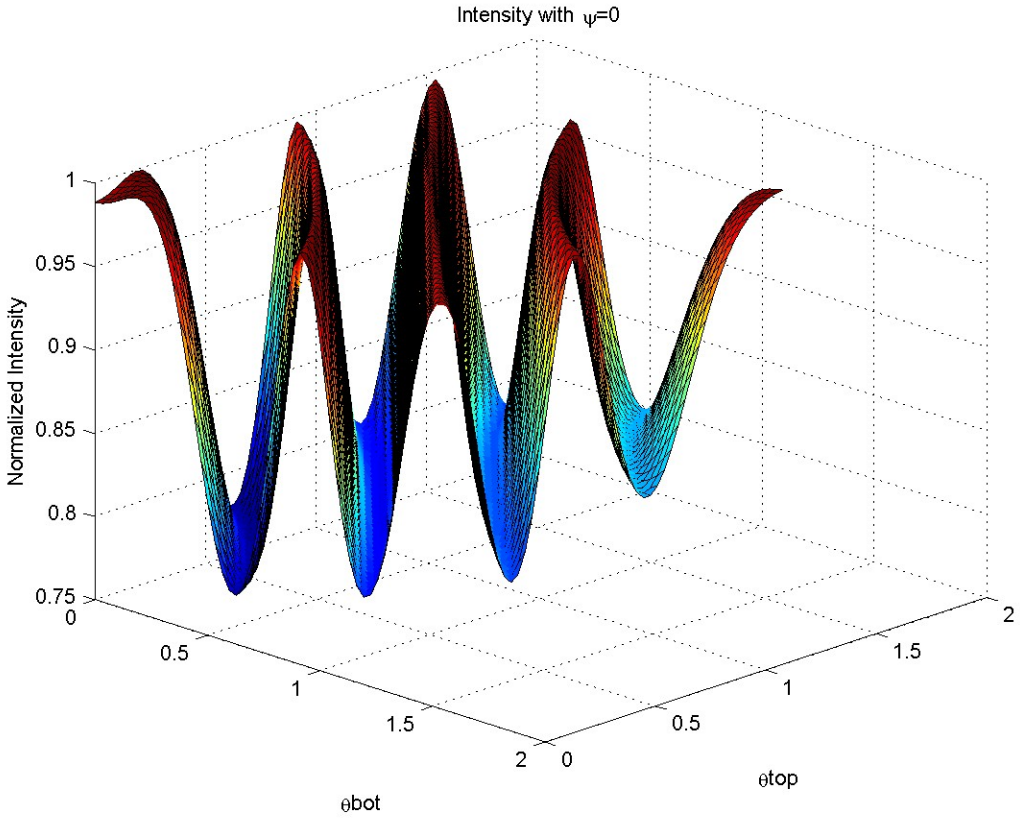

Figure 32. Surface plot of transmitted intensity with $\psi = 0$ for mixed θ.

Line 1 on Figure 33, the diagonal line that connects the lower left corner and the upper right corner, represents matched anchoring for all parameters where $\psi_{top} = \psi_{bot}$ and $\theta_{top} = \theta_{bot}$. The diagonal, Line 2, is perpendicular to Line 1 and contains the points where the values of θ_{top} and θ_{bot} are such that they average to the same value for all points on that line. This is true for all lines parallel to Line 2. The area of interest that corresponds to the global minimum seen in Figure 30 is shown by Line 3 on Figure 33 and correlates to the area where the average is approximately 0.35 for θ_{top} and θ_{bot}. In Figure 34, we take a closer look at the area of interest by taking a slice of Figure 32, where $\theta_{top} = \theta_{bot}$.

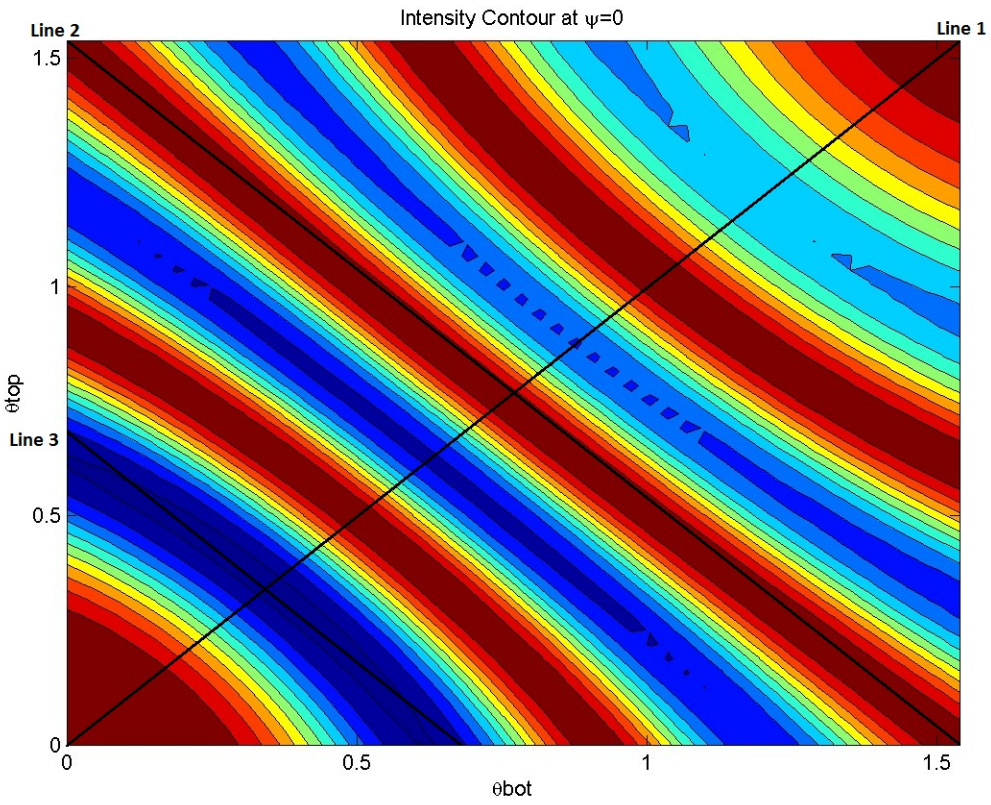

Figure 33. Contour plot of transmitted intensity with $\psi = 0$ for mixed θ.

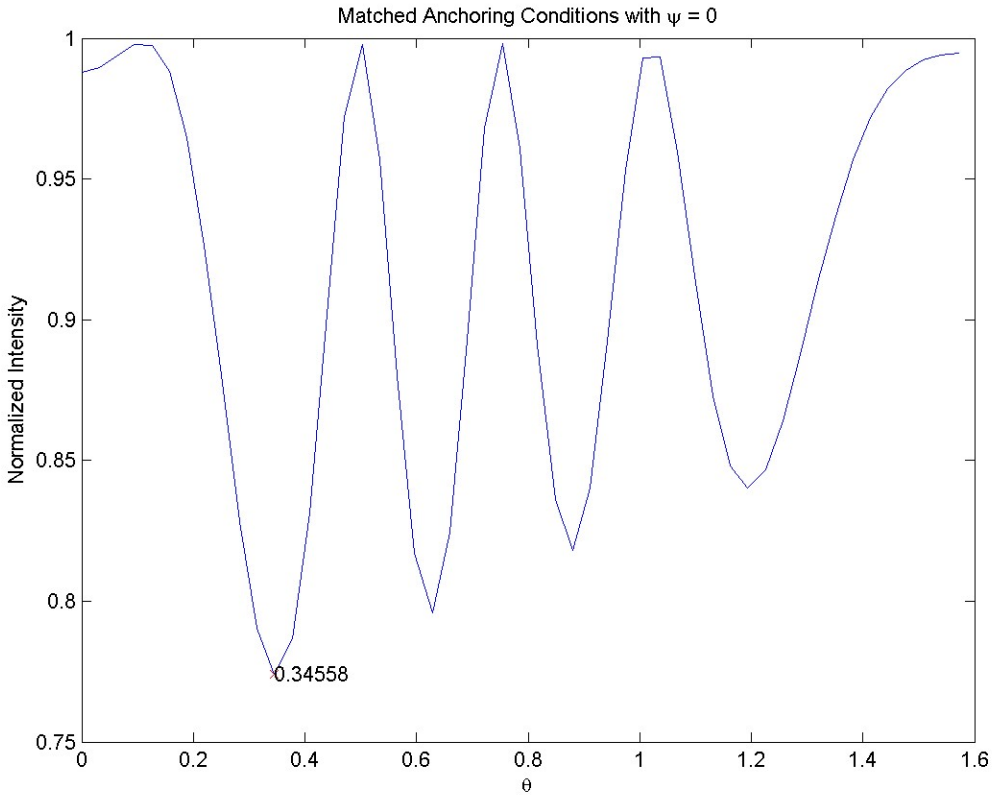

Figure 34. Transmitted intensity for matched θ and matched $\psi = 0$.

As expected, we see that the global minimum is 0.3456 radians. From both the surface and contour plots, we want to find all values of θ that are on that line that averages to 0.3456 radians. Figure 35 shows the transmitted intensity for different values of θ where the average of θ_{top} and θ_{bot} is 0.3456.

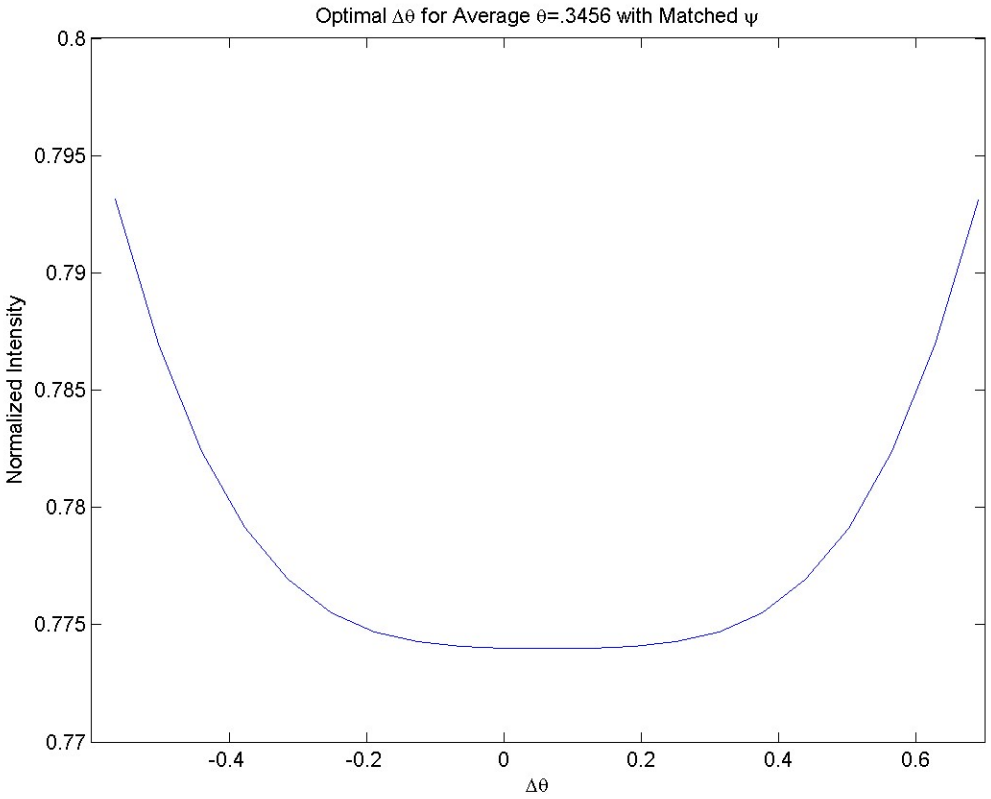

Figure 35. Impact on transmitted intensity for mixed θ around an average $\theta = 0.3456$.

It is clear that the best reduction that we can achieve for a plane wave of known polarization is to use matched ψ and matched θ where the average of $\theta = 0.3456$. Additionally, when utilizing mixed theta for a matched psi, the minimum transmitted intensity is dominated by selecting θ_{top} and θ_{bot} such that the average is still 0.3456 and the difference is near zero.

B. MATCHED THETA

We now take a look at a matched θ with a mixed ψ. We first look at a twisted orientation of the LC molecules by setting ψ_{top} and ψ_{bot} perpendicular to one another and then to swap those anchoring conditions. The results can be seen in Figure 36 through Figure 44 for matched values of $\theta = 0,\ 0.3456,\ \pi/4$. Figure 36 shows that there is little

difference between the curves with regard to reduction in intensity. This is consistent with the associated contour and surface plot. The greatest reduction is a twisted structure established by setting $\psi_{top} = \pi / 2$ and $\psi_{bot} = 0$.

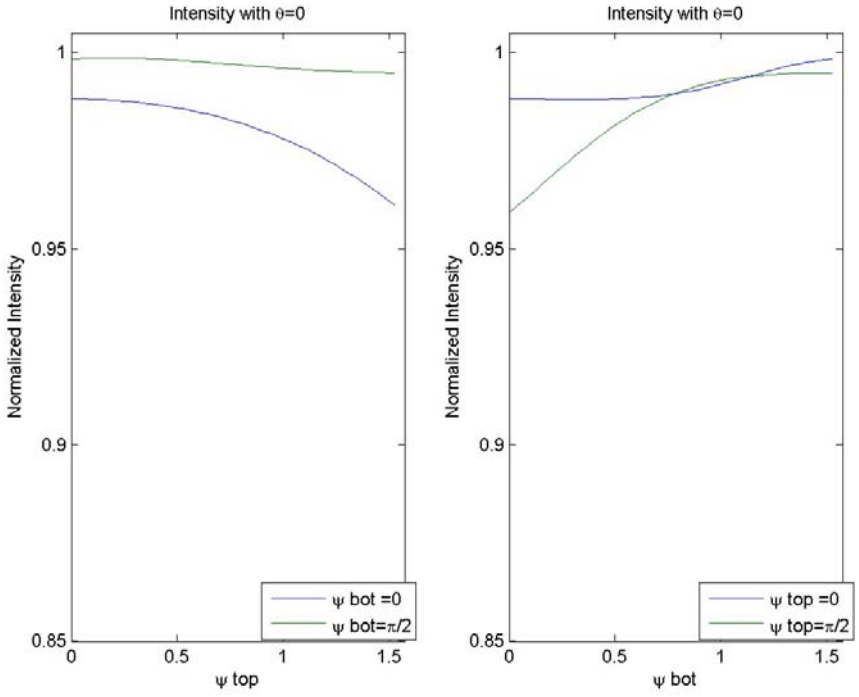

Figure 36. Mixed ψ with matched $\theta = 0$

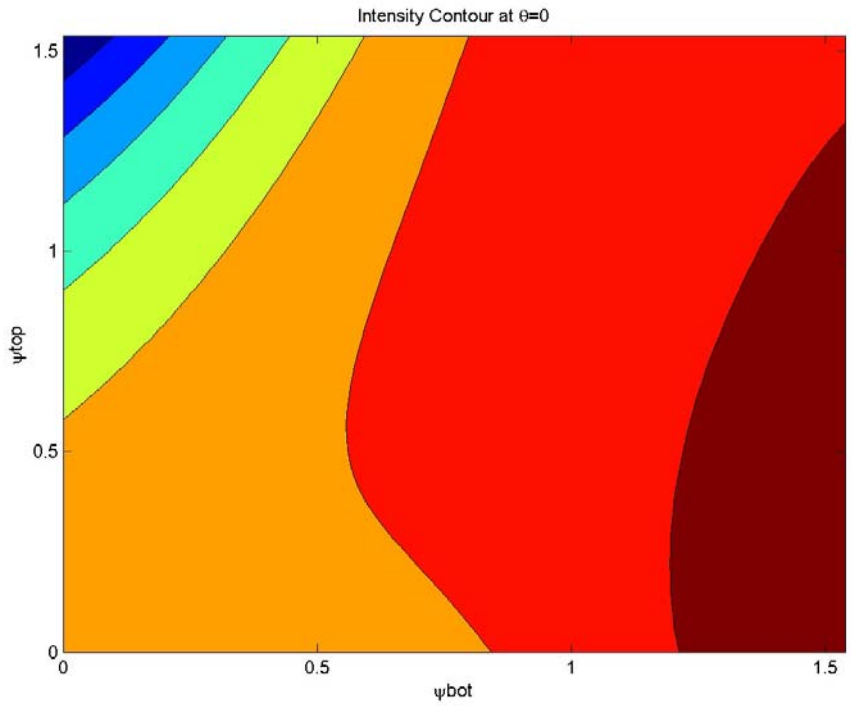

Figure 37. Contour plot of $\theta = 0$ with mixed ψ.

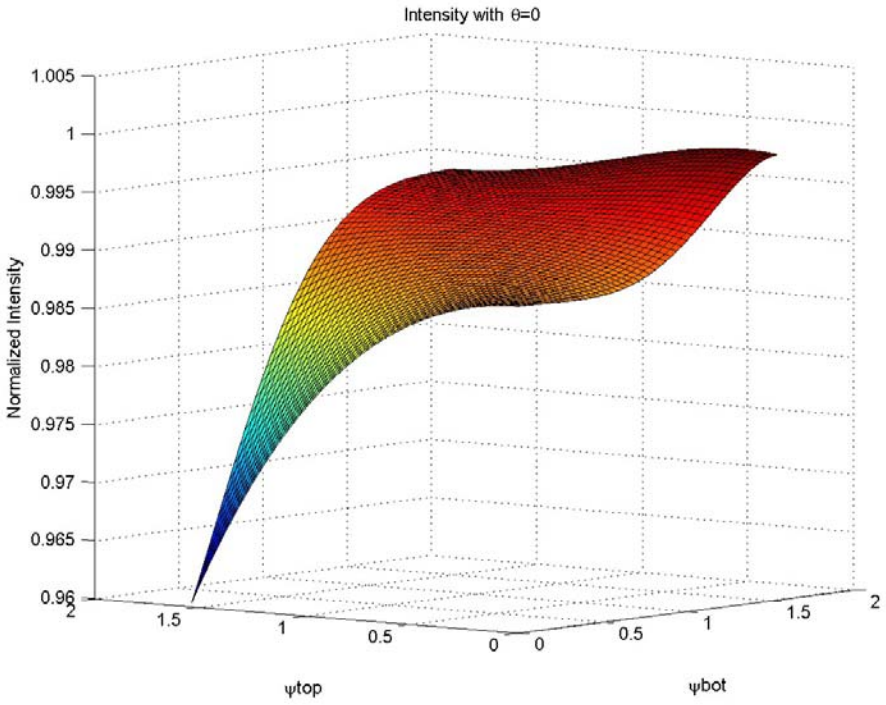

Figure 38. Surface plot of $\theta = 0$ with mixed ψ.

The results of $\theta = \pi / 4$ are similar in that the maximum reduction occurs with $\psi_{top} = \pi / 2$ and $\psi_{bot} = 0$ however is should be noted that the reduction is about ten percent greater than with $\theta = 0$.

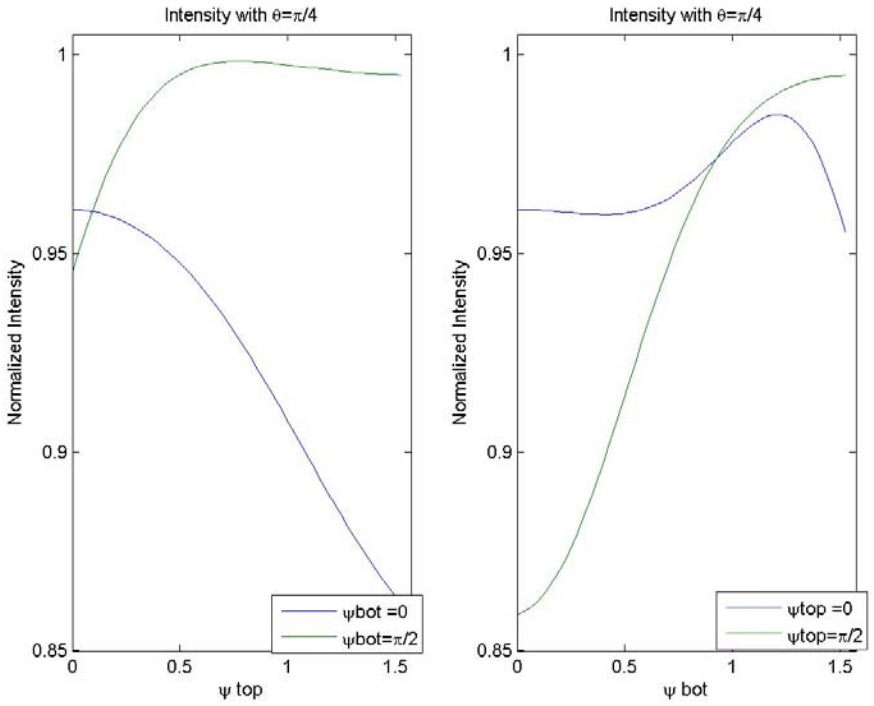

Figure 39. Mixed ψ with matched $\theta = \pi / 4$

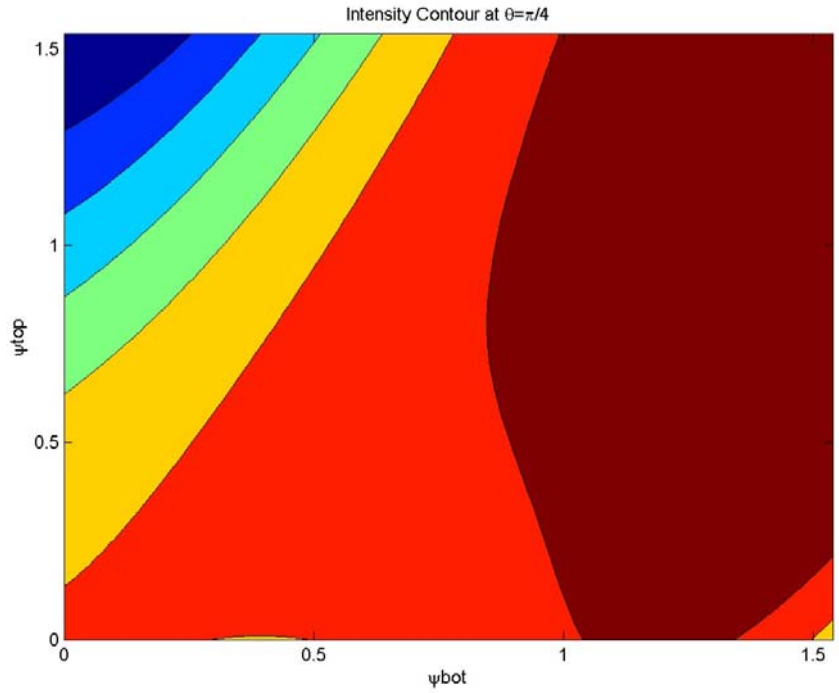

Figure 40. Contour plot of $\theta = \pi / 4$ with mixed ψ.

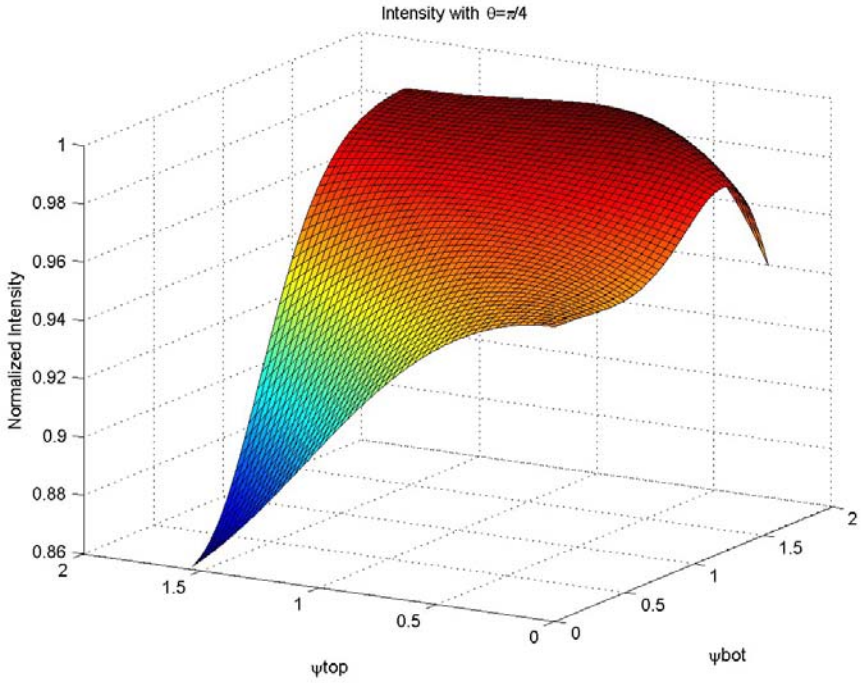

Figure 41. Surface plot of $\theta = \pi / 4$ with mixed ψ.

Finally, Figures 42 through 44 give a greater band to the value of θ_{top} as long as θ_{bot} is bounded to a value <0.5. It should be noted that the maximum reduction occurs at $\theta_{top} < 1$ with $\theta_{bot} < 0.1$. This is consistent with our findings for $\theta = 0.3456$ and $\psi = 0$ for the matched conditions discussed before. The results shown here also suggest that there is more complicated relationship between the transmitted intensity and psi value.

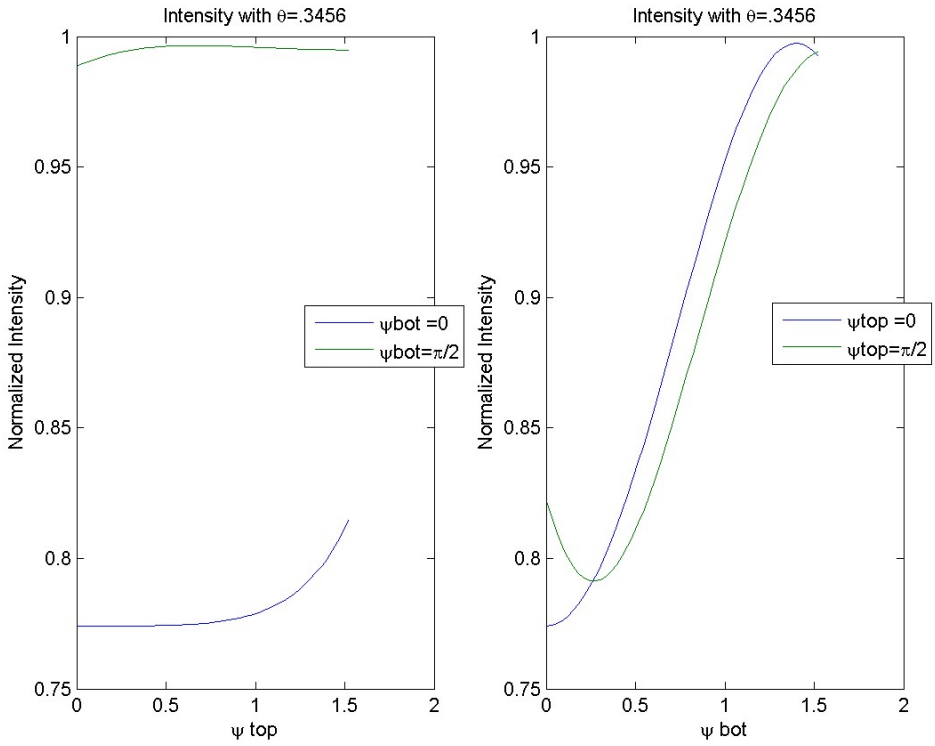

Figure 42. Mixed ψ with matched $\theta = 0.3456$

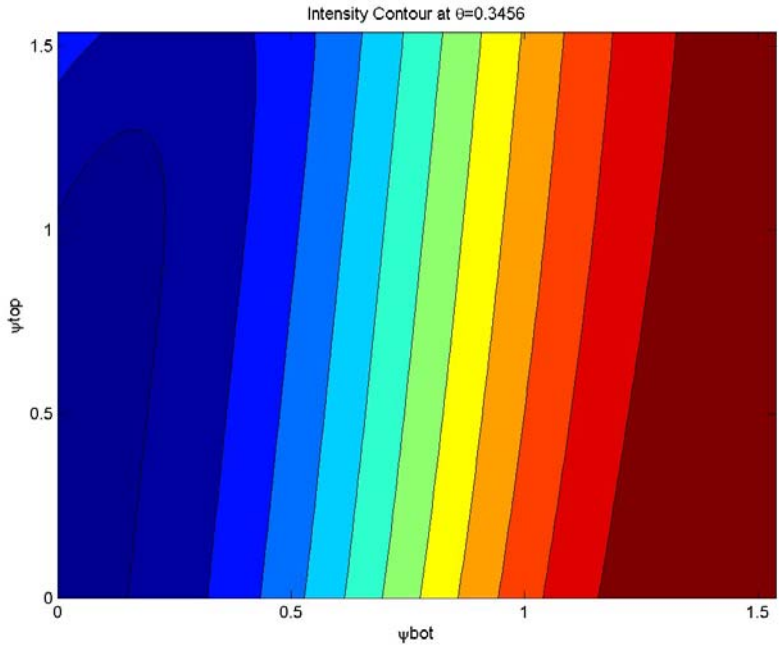

Figure 43. Contour plot of $\theta = 0.3456$ with mixed ψ.

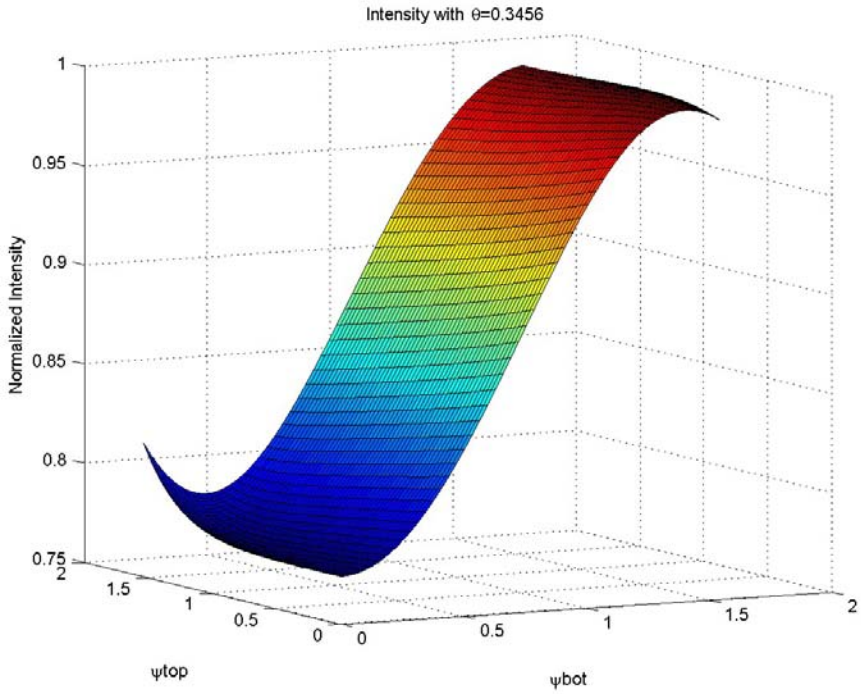

Figure 44. Surface plot of $\theta = 0.3456$ with mixed ψ.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS

A. SUMMARY OF RESULTS

We have looked at the propagation of a plane wave through a LC layer, paying special attention to how the orientation of molecules within the layer affects transmission of an incident beam. Knowing that the anchoring conditions are what determines overall molecular orientation within the LC layer, our goal was to find the optimal anchoring conditions that minimize the transmitted intensity of a known incident beam through the LC layer. These four anchoring conditions, two anchoring conditions at each plate, are independent of one another. This eventually leads to an optimization problem in four-dimensional space.

Given the complexity of this problem, we have constrained the scope of this study by limiting the incident beam to a specific beam of known parameters. We used an incident beam of known wavelength, polarization, perpendicular incident angle, and intensity. There exist other beams that may have a smaller or greater impact from the designed layer. The process of minimizing the intensity of a known beam is different than the process for the vast spectrum of possible beams in wavelength, polarization, and incident angle. This allows us to show preferential focus towards best reducing one specific beam intensity.

We have used a FDTD code to propagate the plane wave through the computational domain. For orientations using matched anchoring conditions, we were able to validate that code by comparing the computed numerical solution to that of the known analytical solution. We validated the solution by comparing meshes consisting of 200, 400, 800, 1600, and 3200 grid points, respectively. We looked at the error between the magnetic field solutions, the electric field solutions, and the intensity solutions through the entire computational domain. Knowing that the mesh grid with N=3200 provided a more accurate solution, we chose to utilize a mesh grid with N=800 to conserve processing time without significant reduction in accuracy.

We first started looking at matched anchoring conditions for both ψ and θ, of which the true solution is known through analytical means. We then looked at selecting fixed values of ψ and varying θ. For $\psi = \pi/2$, the known true solution, for this beam, reduces to a solution that is not dependent on θ. Consistent with the true solution, we observed that as matched ψ approached $\pi/2$, all solutions converged to maximum intensity. Ultimately for any matched value of ψ, the solution is bounded by those solutions with $\psi = 0$ and $\psi = \pi/2$.

With both matched ψ and matched θ, the relationship of intensity was consistent with that of Equation (2.27) and any value of matched ψ could be determined through the linear interpolation of the values of $\psi = 0$ and $\psi = \pi/2$. Also, for all values of matched ψ, the value of $\psi = 0$ was always the greatest intensity reduction regardless of the value of matched θ. This allowed us to set matched ψ to zero for both maximum intensity reduction and to remove it as a variable. We were then able to look at θ through the span of mixed and matched values and determine that the best value was an average θ of 0.3456 with the $\Delta\theta \leq .2$ in order to show a maximum reduction of approximately 25 percent. This solution was verified by a run at a high mesh of 3200.

This segued into the scenario of matched ψ and mixed θ, whereas mentioned before, the average of the thetas dominated the output. For matched θ and mixed ψ, our preliminary results show that there is more complicated relationship between the transmitted intensity and psi value.

B. FUTURE/ONGOING WORK

There continues to be studies in the specifics of this paper. Specifically to this study would be to expand to a 3-dimensional analysis of the problem. This would require a more efficient way to store data than was used in this study as memory was constantly an issue with certain data runs. Also, working toward a multilayer look would be a simple step forward by just treating this study as an individual layer. Multilayer also has the aspect of changing the thickness of the LC medium which was not done in our study.

Investigating into incident beams of different wavelengths and incident angles is very crucial to determine the feasibility of using a LC layer, or similar material, as a protective layer for DoD assets. Other areas of interest are coupled electromagnetic fields where the orientation can be adjusted as needed throughout the medium rather than just anchoring conditions at the boundaries.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

A. MATLAB CODE FOR NUMERICAL SOLUTION

This MATLAB code was used as the foundation in order to calculate and generate curves for the electric field, magnetic field, and intensity. The original code was created by Dr. Eric Choate and modified in order to suit our application. Many alterations were made for the various sections with this code as the baseline for computations.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This implements the Finite Difference Time Domain (FDTD) algorithm for
propagation of the electric and magnetic fields of a plane wave across
a system of liquid crystal polymer (LCP) domain.
%The purpose of this code was to test the convergence of the algorithm.
That is, by using a finer grid, we should get a better approximation of
the true solution.  FDTD is second order, and so we expect that the
error should scale like dx^2, where dx is the grid spacing.  This code
solves the same problem for six successively finer grids.  On each
grid, two different waves are propagated.  The incident beam Exinc and
Hyinc ignores the LCP and propagates through a vacuum, and then the
beam of interest with Ex, Ey, Ez, Hx, and Hy. (Hz is identically zero.)
It is a 1-dimensional problem with the beam propagating in the z-
direction.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;

%% Material parameters and anchoring conditions
global h psibot thetabot psitop thetatop

% Physical dimensions and the medium
%
% The LCP layer has the physical width realh, which is chosen to be 10
free space wavelengths of the incident beam.  But, I nondimensionalize
the system so that the width of the LCP layer is h=1.  The anisotropic
LCP layer goes from z=0 to z=1.  The LCP is sandwiched between two
isotropic glass layers of (nondimensional) thickness hglass. (However,
for this code I wanted to ignore the reflections of the glass to focus
on the effect of the LCP, and so I chose the glass to be optically the
same as the air/vacuum, but to make the code work right, I still
include the glass layers as one numerical cell thick.)  Outside the
glass layers are layers of air (really vacuum) of thickness hair.  This
is the physical system, but there are two more artificial layers used
to help handle the difficulties of artificially truncating the
computational domain without creating nonphysical reflections.  The
scattering layer is width hscat, and the perfectly matched layer (PML)
is of width hPML.

realh=6.330e-6; %physical gap between glass plates
h=1;            %rescaled gap width
%hglass=h/4;     %rescaled width of glass plates
```

```
hair=h/4;       %rescaled width of the air layers
hscat=h/32;     %rescaled width of the scattering layer outside the PML
hPML=h/32;      %rescaled width of the PML

epsilon_perp=1.5^2;     %ordinary relative permittivity of the LCP
epsilon_para=1.7^2;     %extraordinary relative permittivity of the LCP
delta_epsilon=epsilon_para-epsilon_perp;
epsilon_air=1;          %relative permittivity in air and scatter field
epsilon_glass=1; %1.5^2;        %relative permittivity in glass
epsilon_perp

% The incident beam takes the form
% Exinc=E0*sin(wavenumber*(z-zstart)-omega*t)*(1-exp(-(t-
tdelay)^2/sigmadelay))
%
% The Gaussian decay term is there to ease the incident beam into the
system.  If it suddenly turns on at full force, it can introduce
artificial reflections.  I included the possibility of a tdelay, but I
set it to zero.  This decays to a simple sine wave.  Hyinc is the same.
Also, I have used the nondimensional timescale t0=realh/c0.

E0=1;           %strength of the incident field
lambda=6.330e-7; %meters
c0=1;  %299792458;    %speed of light in meters per second
omega=2*pi*realh/lambda;        %2*pi*c0/lambda;
wavenumber=2*pi*realh/lambda;
tdelay=0;
sigmadelay=0.5;

EndTime=150*2*pi/omega % The final time

% Description of the LCP orientation
%
% This uses the Leslie-Ericksen (uniaxial) description of the
orientation with a unit vector called the major director
n=(cos(theta(z))*cos(psi(z)),cos(theta(z))*sin(psi(z)),sin(theta(z)))^T
%
% The angles theta(z) and psi(z) are solutions to a system of ODEs that
are re-solved at each successively finer grid, but the anchoring
(boundary) conditions are our control variables in the system unixial
anchoring conditions at the bottom of the LCP layer (z=0)
psibot=2*pi/3;
thetabot=pi/4;        % theta=0 is tangential
% unixial anchoring conditions at the top of the LCP layer (z=1)
psitop=2*pi/3;
thetatop=pi/4;        % theta=0 is tangential
ntheta=sqrt(epsilon_perp*epsilon_para/(epsilon_perp*(cos(thetabot))^2+e
psilon_para*(sin(thetabot))^2));% extraordinary index of refraction
nperp=sqrt(epsilon_perp); % ordinary index of refraction
kperp=omega*nperp/c0; % ordinary wave number
ktheta=omega*ntheta/c0; % extraordinary wave number
tic
%% The coarsest grid
```

```
% Numerical parameters
NLCP=100;                   % number of numerical cells in the LCP
dz=h/NLCP;                  % width of one cell
% The so-called CFL condition is a relationship between dt and dz that
must
% be satisfied by a grid in order for the solution to converge.  In
this
% case, the CFL condition is c0*dt/dz <= 1.
cdtoverdz=1;
dt=dz*cdtoverdz;            % time step
Tfin=round(EndTime/dt)+1;   % must have one more time step beyond
EndTime to get H at Tfin
timeperiod=round(2*pi/omega/dt);    % the number time steps in one time
period

% The physical parts of the system (LCP, glass, and air layers) are
called the total field.  The other parts are part of the scattered
field.  In the total field, electromagnetic field variables E, D, and H
store the sum of the scattered field and the incident beam, which we
know at all locations and times; however, in the scattered field, these
variables store only the scattered field.  We do this because we do not
want the source of the incident beam to be a part of the "real" system.
By using  a "hard source" (a point at which we specify what the
electric field is), we create a point that makes an artificial
reflection if the wave returns there.
The PML uses two parameters to quantify how it absorbs the
electromagnetic field without reflection
sigmamax=2/dt;               % in the PML, maximum value of sigma at the
PEC boundary
sigmam=3;                   % in the PML, sigmaz=sigmamax((kPMLbot+1-
k)/kPMLbot)^sigmam

Nglass=1; %round(hglass/dz);  %number of numerical cells in the glass
Nair=round(hair/dz);        % number of numerical cells in the air
between glass and scattering layer
Nscat=round(hscat/dz);      % number of numerical cells in the
scattering later
NPML=round(hPML/dz);        % number of numerical cells in the
perfectly matched layer
kPMLbot=NPML;               % sigma defined but equal to zero at
k=kPMLbot
kscatbot=kPMLbot+Nscat;     % k=kscatbot is considered to be in the
total field; kscatbot-1/2 is in the scattered field
kairbot=kscatbot+Nair;      % k=kairbot is the bottom edge of the
bottom plate
kglassbot=kairbot+Nglass;   % k=kglassbot is the top edge of the bottom
plate/where the anchoring condition is applied
kglasstop=kglassbot+NLCP;   % k=kglasstop is the bottom edge of the top
plate/where the anchoring condition is applied
kairtop=kglasstop+Nglass;   % k=kairtop is the top edge of the top
plate
kscattop=kairtop+Nair;      % k=kscattop is in the total field;
kscattop+1/2 is not
kPMLtop=kscattop+Nscat;     % sigma defined but equal to zero at
k=kPMLtop
```

```
Ntot=kPMLtop+NPML;          % Ntot cells in total
kincstart=kscatbot-2;       % location of the incident field
ze=2:Ntot;                  % k values for updating the gradients of H
in the D and E equations
zh=2:Ntot+1;                % k values for updating the gradients of E
in the H equations

%  Define electromagnetic field variables
%
%  This code is set up to store the electromagnetic variables for all
values of z_k=(k-kglassbot-1)*dz but to only store the values of
t_n=n*dt for the last time period (including both the first and
lasttime of the period).  We overwrite the values at each time step
until we get to n=Tbreak
Tbreak=Tfin-2*timeperiod-1;
%at which time we switch to storing all the time steps.  E and D are
known on the space-time grid.  At time zero, they are zero, but this is
not stored.  Ex and Dx are zero at k=1 and k=Ntot+1 but they are stored
so that Ex(k,n) stores Ex at z=(k-kglassbot-1)*dz and t=n*dt, or Ex_k^n
is stored as Ex(k+1,n).

Dx=zeros(Ntot+1,2*timeperiod+1);
Dy=zeros(Ntot+1,2*timeperiod+1);
Dz=zeros(Ntot+1,2*timeperiod+1);
Ex=zeros(Ntot+1,2*timeperiod+1);
Ey=zeros(Ntot+1,2*timeperiod+1);
Ez=zeros(Ntot+1,2*timeperiod+1);

% H is known on the half-grid.  Hx(k,n) is Hx at z=(k-kglassbot+1/2)*dz
% and t=(n-1/2)*dt, or Hx_(k+1/2)^(n+1/2) is stored as Hx(k+1,n+1).
% Hz is zero and not stored.  We have to store one more time step so
that we can average the half-grid times to get the on-grid values.
Similarly, we average the half-grid spaces to get the on-grid values,
but because we E and D store the zero values at the boundary, there is
one more stored value of Ex than Hy.

Hx=zeros(Ntot,2*timeperiod+2);
Hy=zeros(Ntot,2*timeperiod+2);

% The incident beam is stored as Exinc, Eyinc, Hxinc, and Hyinc
although the incident beam we use has Eyinc=Hxinc=0.
Exinc=zeros(Ntot+1,2*timeperiod+1);
Eyinc=zeros(Ntot+1,2*timeperiod+1);
Hxinc=zeros(Ntot,2*timeperiod+2);
Hyinc=zeros(Ntot,2*timeperiod+2);

% The "tilde" H vales interpolate the half-grid values of H onto the
grid
Hxtilde=zeros(Ntot-1,2*timeperiod+1);
Hytilde=zeros(Ntot-1,2*timeperiod+1);
Hxinctilde=zeros(Ntot-1,2*timeperiod+1);
Hyinctilde=zeros(Ntot-1,2*timeperiod+1);
```

```
% The Poynting vector, the cross product of E and H, is known on the
space-time grid.  It is not stored at k=1 or k=Ntot+1
Poyntingx=zeros(Ntot-1,2*timeperiod+1);
Poyntingy=zeros(Ntot-1,2*timeperiod+1);
Poyntingz=zeros(Ntot-1,2*timeperiod+1);
PoyntingMag=zeros(Ntot-1,2*timeperiod+1);  %The magnitude of the
Poynting vector

% Compute the permittivity tensor epsilon
%
% First, we must compute the LCP orientation by solving an ODE.  It
uses different z values used for computing the orientation
z1=linspace(0,h,NLCP+1);
solinit=bvpinit(z1,@LEodeinit);
sol=bvp4c(@LEodefun,@LEbcfun,solinit);
u=deval(sol,z1);
psiLE=u(1,:);
thetaLE=u(3,:);

n1=cos(thetaLE).*cos(psiLE);
n2=cos(thetaLE).*sin(psiLE);
n3=sin(thetaLE);

% The permittivity tensor epsilon is symmetric and stored on the grid
for the whole domain.  In the LCP, we use the above orientation, but we
also need to define it in the isotropic regions

epsxx=zeros(Ntot+1,1);
epsyy=zeros(Ntot+1,1);
epszz=zeros(Ntot+1,1);
epsxy=zeros(Ntot+1,1);
epsxz=zeros(Ntot+1,1);
epsyz=zeros(Ntot+1,1);

epsxx(1:kPMLbot+1)=1; epsyy(1:kPMLbot+1)=1; epszz(1:kPMLbot+1)=1;
epsxx(kPMLbot+2:kairbot)=epsilon_air;
epsyy(kPMLbot+2:kairbot)=epsilon_air;
epszz(kPMLbot+2:kairbot)=epsilon_air;
epsxx(kairbot+1:kglassbot)=epsilon_glass;
epsyy(kairbot+1:kglassbot)=epsilon_glass;
epszz(kairbot+1:kglassbot)=epsilon_glass;
epsxx(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n1.^2;
epsyy(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n2.^2;
epszz(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n3.^2;
epsxy(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n2;
epsxz(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n3;
epsyz(kglassbot+1:kglasstop+1)=delta_epsilon*n2.*n3;
epsxx(kglasstop+2:kairtop+1)=epsilon_glass;
epsyy(kglasstop+2:kairtop+1)=epsilon_glass;
epszz(kglasstop+2:kairtop+1)=epsilon_glass;
epsxx(kairtop+2:kPMLtop)=epsilon_air;
epsyy(kairtop+2:kPMLtop)=epsilon_air;
epszz(kairtop+2:kPMLtop)=epsilon_air;
```

```
epsxx(kPMLtop+1:Ntot)=1; epsyy(kPMLtop+1:Ntot)=1;
epszz(kPMLtop+1:Ntot)=1;

% For computation, we really need the inverse of epsilon, and so we
will pre-compute that.
epsdet=epsxx.*(epsyy.*epszz-epsyz.^2)-epsxy.*(epsxy.*epszz-
epsxz.*epsyz)+epsxz.*(epsxy.*epsyz-epsyy.*epsxz);
epsinvxx=(epsyy.*epszz-epsyz.^2)./epsdet;
epsinvxy=(epsxz.*epsyz-epsxy.*epszz)./epsdet;
epsinvxz=(epsxy.*epsyz-epsyy.*epsxz)./epsdet;
epsinvyy=(epsxx.*epszz-epsxz.^2)./epsdet;
epsinvyz=(epsxy.*epsxz-epsxx.*epsyz)./epsdet;
epsinvzz=(epsxx.*epsyy-epsxy.^2)./epsdet;

% Define coefficients used in our update scheme.
% This is where the PML absorbing factors are used.
C_E=ones(Ntot+1,1);             % factor of the D^n term in the D
update equations
D_E=cdtoverdz*ones(Ntot+1,1);   % factor of the curl H term in the D
update equations
C_H=ones(Ntot,1);               % factor of the H^n term in the H
update equations
D_H=cdtoverdz*ones(Ntot,1);     % factor of the curl H term in the H
update equations
for k=1:kPMLbot                 % fill in the effect of sigma in the
PML
   sigmaz=sigmamax*((kPMLbot-k+1)/kPMLbot)^sigmam;
   C_E(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_E(Ntot+2-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(Ntot+2-k)=2*cdtoverdz/(2+dt*sigmaz);
   sigmaz=sigmamax*((kPMLbot-k+0.5)/kPMLbot)^sigmam;
   C_H(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_H(Ntot+1-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(Ntot+1-k)=2*cdtoverdz/(2+dt*sigmaz);
end

% The starting main loop, not storing every step
Exinc(kincstart+1,1)=-E0*sin(omega*dt)*(1-exp(-(dt-
tdelay)^2/sigmadelay));
Hx(kscatbot,1)=-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,1)=D_H(kscatbot)*Exinc(kscatbot+1,1);
Hxinc(:,1)=D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,1)=-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
for n=2:Tbreak-1
   % Update dD/dt=curl(H) at t=(n+1)*dt
   Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
   Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
   % There is a correction at scattered field-total field boundary

Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
      Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-
D_E(kscatbot+1)*Hxinc(kscatbot,1);
```

```
      Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);
   % Get E from E=epsilon^-1*D

Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);

Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
   % Update the incident electric field
   Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-
1,1));
   Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-
1,1));
   Exinc(kincstart+1,1)=-E0*sin(omega*n*dt)*(1-exp(-(n*dt-
tdelay)^2/sigmadelay));
   % Update dH/dt=-curl(E) at t=(n+3/2)*dt
   Hx(:,1)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
   Hy(:,1)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
   % Correction at the scattered field-total field boundary
      Hx(kscatbot,1)=Hx(kscatbot,1)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
      Hy(kscatbot,1)=Hy(kscatbot,1)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,1)=Hx(kscattop+1,1)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
      Hy(kscattop+1,1)=Hy(kscattop+1,1)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
   % Update the incident magnetic field
   Hxinc(:,1)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
   Hyinc(:,1)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
end
% Now for the last period, we store each time step.  Also, we need to
start computing the Poynting vector, and so we need to compute Htilde,
which is H interpolated onto the grid. The first time through is
different than the following loop and the last time step.
Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
  Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
  Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
  Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);
Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);
Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
Ez(ze,1)=epsinvxz(ze).*Dx(ze,1)+epsinvyz(ze).*Dy(ze,1)+epsinvzz(ze).*Dz
(ze,1);
Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-1,1));
Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-1,1));
Exinc(kincstart+1,1)=-E0*sin(omega*Tbreak*dt)*(1-exp(-(Tbreak*dt-
tdelay)^2/sigmadelay));
Hx(:,2)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
```

```
Hy(:,2)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
  Hx(kscatbot,2)=Hx(kscatbot,2)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
  Hy(kscatbot,2)=Hy(kscatbot,2)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,2)=Hx(kscattop+1,2)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
  Hy(kscattop+1,2)=Hy(kscattop+1,2)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
Hxinc(:,2)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,2)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
Hxinctilde(:,1)=(Hxinc(ze,1)+Hxinc(ze,2)+Hxinc(ze-1,1)+Hxinc(ze-
1,2))/4;
Hyinctilde(:,1)=(Hyinc(ze,1)+Hyinc(ze,2)+Hyinc(ze-1,1)+Hyinc(ze-
1,2))/4;
Hxtilde(:,1)=(Hx(ze,1)+Hx(ze,2)+Hx(ze-1,1)+Hx(ze-1,2))/4;
Hytilde(:,1)=(Hy(ze,1)+Hy(ze,2)+Hy(ze-1,1)+Hy(ze-1,2))/4;
Poyntingx(:,1)=-Ez(ze,1).*Hytilde(:,1);
Poyntingy(:,1)=Ez(ze,1).*Hxtilde(:,1);
Poyntingz(:,1)=Ex(ze,1).*Hytilde(:,1)-Ey(ze,1).*Hxtilde(:,1);
PoyntingMag(:,1)=sqrt(Poyntingx(:,1).^2+Poyntingy(:,1).^2+Poyntingz(:,1
).^2);
for n=1:2*timeperiod
   Dx(ze,n+1)=C_E(ze).*Dx(ze,n)-D_E(ze).*(Hy(ze,n+1)-Hy(ze-1,n+1));
   Dy(ze,n+1)=C_E(ze).*Dy(ze,n)+D_E(ze).*(Hx(ze,n+1)-Hx(ze-1,n+1));

Dx(kscatbot+1,n+1)=Dx(kscatbot+1,n+1)+D_E(kscatbot+1)*Hyinc(kscatbot,n+
1);
     Dy(kscatbot+1,n+1)=Dy(kscatbot+1,n+1)-
D_E(kscatbot+1)*Hxinc(kscatbot,n+1);
     Dx(kscattop+1,n+1)=Dx(kscattop+1,n+1)-
D_E(kscattop+1)*Hyinc(kscattop+1,n+1);

Dy(kscattop+1,n+1)=Dy(kscattop+1,n+1)+D_E(kscattop+1)*Hxinc(kscattop+1,
n+1);

Ex(ze,n+1)=epsinvxx(ze).*Dx(ze,n+1)+epsinvxy(ze).*Dy(ze,n+1)+epsinvxz(z
e).*Dz(ze,n+1);

Ey(ze,n+1)=epsinvxy(ze).*Dx(ze,n+1)+epsinvyy(ze).*Dy(ze,n+1)+epsinvyz(z
e).*Dz(ze,n+1);

Ez(ze,n+1)=epsinvxz(ze).*Dx(ze,n+1)+epsinvyz(ze).*Dy(ze,n+1)+epsinvzz(z
e).*Dz(ze,n+1);
   Exinc(ze,n+1)=C_E(ze).*Exinc(ze,n)-D_E(ze).*(Hyinc(ze,n+1)-Hyinc(ze-
1,n+1));
   Eyinc(ze,n+1)=C_E(ze).*Eyinc(ze,n)+D_E(ze).*(Hxinc(ze,n+1)-Hxinc(ze-
1,n+1));
   Exinc(kincstart+1,n+1)=-E0*sin(omega*(Tbreak+n)*dt)*(1-exp(-
((Tbreak+n)*dt-tdelay)^2/sigmadelay));
   Hx(:,n+2)=C_H.*Hx(:,n+1)+D_H.*(Ey(zh,n+1)-Ey(zh-1,n+1));
   Hy(:,n+2)=C_H.*Hy(:,n+1)-D_H.*(Ex(zh,n+1)-Ex(zh-1,n+1));
     Hx(kscatbot,n+2)=Hx(kscatbot,n+2)-
D_H(kscatbot)*Eyinc(kscatbot+1,n+1);

Hy(kscatbot,n+2)=Hy(kscatbot,n+2)+D_H(kscatbot)*Exinc(kscatbot+1,n+1);
```

```
Hx(kscattop+1,n+2)=Hx(kscattop+1,n+2)+D_H(kscattop+1)*Eyinc(kscattop+1,
n+1);
     Hy(kscattop+1,n+2)=Hy(kscattop+1,n+2)-
D_H(kscattop+1)*Exinc(kscattop+1,n+1);
   Hxinc(:,n+2)=C_H.*Hxinc(:,n+1)+D_H.*(Eyinc(zh,n+1)-Eyinc(zh-1,n+1));
   Hyinc(:,n+2)=C_H.*Hyinc(:,n+1)-D_H.*(Exinc(zh,n+1)-Exinc(zh-1,n+1));
   Hxinctilde(:,n+1)=(Hxinc(ze,n+1)+Hxinc(ze,n+2)+Hxinc(ze-
1,n+1)+Hxinc(ze-1,n+2))/4;
   Hyinctilde(:,n+1)=(Hyinc(ze,n+1)+Hyinc(ze,n+2)+Hyinc(ze-
1,n+1)+Hyinc(ze-1,n+2))/4;
   Hxtilde(:,n+1)=(Hx(ze,n+1)+Hx(ze,n+2)+Hx(ze-1,n+1)+Hx(ze-1,n+2))/4;
   Hytilde(:,n+1)=(Hy(ze,n+1)+Hy(ze,n+2)+Hy(ze-1,n+1)+Hy(ze-1,n+2))/4;
   Poyntingx(:,n+1)=-Ez(ze,n+1).*Hytilde(:,n+1);
   Poyntingy(:,n+1)=Ez(ze,n+1).*Hxtilde(:,n+1);
   Poyntingz(:,n+1)=Ex(ze,n+1).*Hytilde(:,n+1)-
Ey(ze,n+1).*Hxtilde(:,n+1);

PoyntingMag(:,n+1)=sqrt(Poyntingx(:,n+1).^2+Poyntingy(:,n+1).^2+Poyntin
gz(:,n+1).^2);
end
% Now we compute the intensity, which is the time-average of the
magnitude of the Poynting vector over one period.  We use the trapezoid
rule to approximate the integration
Intensity=PoyntingMag(:,1);
for n=2:2*timeperiod
    Intensity=Intensity+2*PoyntingMag(:,n);
end
Intensity=Intensity+PoyntingMag(:,2*timeperiod+1);
Intensity=Intensity*dt/E0^2/(4*pi/omega);

% We are now done with this grid, and so we store the values to compare
later with the finer grids.
kPMLbot1=kPMLbot; kscatbot1=kscatbot; kairbot1=kairbot;
kglassbot1=kglassbot;
kglasstop1=kglasstop; kairtop1=kairtop; kscattop1=kscattop;
kPMLtop1=kPMLtop;
timeperiod1=timeperiod;
Ntot1=Ntot; dz1=dz; dt1=dt;
Ex1=Ex; Ey1=Ey; Ez1=Ez;
Hx1=Hx; Hy1=Hy;
Exinc1=Exinc; Eyinc1=Eyinc;
Hxinc1=Hxinc; Hyinc1=Hyinc;
Hxinctilde1=Hxinctilde; Hyinctilde1=Hyinctilde;
Hxtilde1=Hxtilde; Hytilde1=Hytilde;
Poyntingx1=Poyntingx; Poyntingy1=Poyntingy; Poyntingz1=Poyntingz;
PoynitngMag1=PoyntingMag; Intensity1=Intensity;
Tfin1=Tfin; Tbreak1=Tbreak;
kincstart1=kincstart;
zTotE1=dz*((1:Ntot+1)-kglassbot-1);     % locations of E, D, and
epsilon, z=0 is bottom of the LCP, z=1 is top
zTotH1=dz*((1:Ntot)-1/2-kglassbot);     % locations where H is known;
not contained in previous refinement
toc
tic
```

```
%% Finer grid
%  We now repeat the above loop for a grid spacing that is one-half the
previous grid in both space and time.
refinement=2;
NLCP=NLCP*refinement;           %number of numerical cells in the LCP
dz=h/NLCP;
Nglass=Nglass*refinement;       %number of numerical cells in the glass
Nair=Nair*refinement;           %number of numerical cells in the air
between glass and scattering layer
Nscat=Nscat*refinement;         %number of numerical cells in the
scattering later
NPML=NPML*refinement;           %number of numerical cells in the
perfectly matched layer
cdtoverdz=1;
dt=dz*cdtoverdz;
Tfin=floor(EndTime/dt)+1;
timeperiod=round(2*pi/omega/dt);
sigmamax=2/dt; %10^17;        %in the PML, sigmaz=sigmamax((kPML-
1)/(NPML-1))^sigmam
sigmam=3;
kPMLbot=NPML;               % sigma defined but equal to zero at
k=kPMLbot
kscatbot=kPMLbot+Nscat;     % k=kscatbot is considered to be in the
total field; kscatbot-1/2 is in the scattered field
kairbot=kscatbot+Nair;      % k=kairbot is the bottom edge of the
bottom plate
kglassbot=kairbot+Nglass;   % k=kglassbot is the top edge of the bottom
plate/where the anchoring condition is applied
kglasstop=kglassbot+NLCP;   % k=kglasstop is the bottom edge of the top
plate/where the anchoring condition is applied
kairtop=kglasstop+Nglass;   % k=kairtop is the top edge of the top
plate
kscattop=kairtop+Nair;      % k=kscattop is in the total field;
kscattop+1/2 is not
kPMLtop=kscattop+Nscat;     % sigma defined but equal to zero at
k=kPMLtop
Ntot=kPMLtop+NPML;          % Ntot cells in total% ze=2:Ntot;
ze=2:Ntot;                   % k values for updating the gradients of H
in the D and E equations
zh=2:Ntot+1;
z2=linspace(0,h,NLCP+1);  %z values used for computing the orientation
kincstart=refinement*(kincstart1-kglassbot1)+kglassbot;

%% compute the permittivity
solinit=bvpinit(z2,@LEodeinit);
sol=bvp4c(@LEodefun,@LEbcfun,solinit);
u=deval(sol,z2);
psiLE=u(1,:);
thetaLE=u(3,:);

n1=cos(thetaLE).*cos(psiLE);
n2=cos(thetaLE).*sin(psiLE);
n3=sin(thetaLE);

epsxx=zeros(Ntot+1,1);
```

```
epsyy=zeros(Ntot+1,1);
epszz=zeros(Ntot+1,1);
epsxy=zeros(Ntot+1,1);
epsxz=zeros(Ntot+1,1);
epsyz=zeros(Ntot+1,1);

epsxx(1:kPMLbot+1)=1; epsyy(1:kPMLbot+1)=1; epszz(1:kPMLbot+1)=1;
epsxx(kPMLbot+2:kairbot)=epsilon_air;
epsyy(kPMLbot+2:kairbot)=epsilon_air;
epszz(kPMLbot+2:kairbot)=epsilon_air;
epsxx(kairbot+1:kglassbot)=epsilon_glass;
epsyy(kairbot+1:kglassbot)=epsilon_glass;
epszz(kairbot+1:kglassbot)=epsilon_glass;
epsxx(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n1.^2;
epsyy(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n2.^2;
epszz(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n3.^2;
epsxy(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n2;
epsxz(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n3;
epsyz(kglassbot+1:kglasstop+1)=delta_epsilon*n2.*n3;
epsxx(kglasstop+2:kairtop+1)=epsilon_glass;
epsyy(kglasstop+2:kairtop+1)=epsilon_glass;
epszz(kglasstop+2:kairtop+1)=epsilon_glass;
epsxx(kairtop+2:kPMLtop)=epsilon_air;
epsyy(kairtop+2:kPMLtop)=epsilon_air;
epszz(kairtop+2:kPMLtop)=epsilon_air;
epsxx(kPMLtop+1:Ntot)=1; epsyy(kPMLtop+1:Ntot)=1;
epszz(kPMLtop+1:Ntot)=1;

epsdet=epsxx.*(epsyy.*epszz-epsyz.^2)-epsxy.*(epsxy.*epszz-
epsxz.*epsyz)+epsxz.*(epsxy.*epsyz-epsyy.*epsxz);
epsinvxx=(epsyy.*epszz-epsyz.^2)./epsdet;
epsinvxy=(epsxz.*epsyz-epsxy.*epszz)./epsdet;
epsinvxz=(epsxy.*epsyz-epsyy.*epsxz)./epsdet;
epsinvyy=(epsxx.*epszz-epsxz.^2)./epsdet;
epsinvyz=(epsxy.*epsxz-epsxx.*epsyz)./epsdet;
epsinvzz=(epsxx.*epsyy-epsxy.^2)./epsdet;

% Define coefficients
C_E=ones(Ntot+1,1);             % factor of the D^n term in the D
update equations
D_E=cdtoverdz*ones(Ntot+1,1);   % factor of the curl H term in the D
update equations
C_H=ones(Ntot,1);               % factor of the H^n term in the H
update equations
D_H=cdtoverdz*ones(Ntot,1);     % factor of the curl H term in the H
update equations
for k=1:kPMLbot                 % fill in the effect of sigma in the
PML
   sigmaz=sigmamax*((kPMLbot-k+1)/kPMLbot)^sigmam;
   C_E(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_E(Ntot+2-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(Ntot+2-k)=2*cdtoverdz/(2+dt*sigmaz);
   sigmaz=sigmamax*((kPMLbot-k+0.5)/kPMLbot)^sigmam;
   C_H(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
```

```
   D_H(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_H(Ntot+1-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(Ntot+1-k)=2*cdtoverdz/(2+dt*sigmaz);
end

%  Define electromagnetic field variables
Dx=zeros(Ntot+1,2*timeperiod+1);      % D and E are known on the space-
time grid
Dy=zeros(Ntot+1,2*timeperiod+1);      % D and E fields are zero
initially (n=0), but this is not stored.
Dz=zeros(Ntot+1,2*timeperiod+1);      % Dx and Ex are zero at k=1 and
k=Ntot+1 but they are stored so that
Ex=zeros(Ntot+1,2*timeperiod+1);      % Dx(k,n) is Dx at t=n*dt and
z=(k-kglassbot-1)*dz,
Ey=zeros(Ntot+1,2*timeperiod+1);      % or Ex_k^n is stored as
Ex(k+1,n)
Ez=zeros(Ntot+1,2*timeperiod+1);      % H is known on the half-grid.
Hx=zeros(Ntot,2*timeperiod+2);        % Hx(k,n) is Hx at t=(n-1/2)*dt,
z=(k-kglassbot+1/2)*dz
Hy=zeros(Ntot,2*timeperiod+2);        % Hx_(k+1/2)^(n+1/2) is stored as
Hx(k+1,n+1)
Exinc=zeros(Ntot+1,2*timeperiod+1);   % Hz is zero and not stored.  Ez
can be nonzero, dependending on epsilon
Eyinc=zeros(Ntot+1,2*timeperiod+1);   %
Hxinc=zeros(Ntot,2*timeperiod+2);     % Exinc, Eyinc, Hxinc, and Hyinc
are the incident field
Hyinc=zeros(Ntot,2*timeperiod+2);
Hxinctilde=zeros(Ntot-1,2*timeperiod+1);   % Hxinctilde interpolates
Hxinc onto the grid
Hyinctilde=zeros(Ntot-1,2*timeperiod+1);
Hxtilde=zeros(Ntot-1,2*timeperiod+1);      % Hxtilde interpolates Hx
onto the grid
Hytilde=zeros(Ntot-1,2*timeperiod+1);
Poyntingx=zeros(Ntot-1,2*timeperiod+1);  % The Poynting vector is known
on the space-time grid, and so H must be
Poyntingy=zeros(Ntot-1,2*timeperiod+1);  % interpolated in time and in
space.  Thus it is not defined at k=1 or k=Ntot+1
Poyntingz=zeros(Ntot-1,2*timeperiod+1);  % or at t=Tfin
PoyntingMag=zeros(Ntot-1,2*timeperiod+1);
Tbreak=Tfin-2*timeperiod-1;

% The starting main loop, not storing every step
Exinc(kincstart+1,1)=-E0*sin(omega*dt)*(1-exp(-(dt-
tdelay)^2/sigmadelay));
Hx(kscatbot,1)=-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,1)=D_H(kscatbot)*Exinc(kscatbot+1,1);
Hxinc(:,1)=D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,1)=-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
for n=2:Tbreak-1
   Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
   Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));

Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
     Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-
D_E(kscatbot+1)*Hxinc(kscatbot,1);
```

```
     Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);

Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);

Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
   Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-
1,1));
   Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-
1,1));
   Exinc(kincstart+1,1)=-E0*sin(omega*n*dt)*(1-exp(-(n*dt-
tdelay)^2/sigmadelay));
   Hx(:,1)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
   Hy(:,1)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
     Hx(kscatbot,1)=Hx(kscatbot,1)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
     Hy(kscatbot,1)=Hy(kscatbot,1)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,1)=Hx(kscattop+1,1)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
     Hy(kscattop+1,1)=Hy(kscattop+1,1)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
   Hxinc(:,1)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
   Hyinc(:,1)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
end
Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
  Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
  Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
  Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);
Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);
Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
Ez(ze,1)=epsinvxz(ze).*Dx(ze,1)+epsinvyz(ze).*Dy(ze,1)+epsinvzz(ze).*Dz
(ze,1);
Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-1,1));
Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-1,1));
Exinc(kincstart+1,1)=-E0*sin(omega*Tbreak*dt)*(1-exp(-(Tbreak*dt-
tdelay)^2/sigmadelay));
Hx(:,2)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
Hy(:,2)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
  Hx(kscatbot,2)=Hx(kscatbot,2)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
  Hy(kscatbot,2)=Hy(kscatbot,2)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,2)=Hx(kscattop+1,2)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
  Hy(kscattop+1,2)=Hy(kscattop+1,2)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
Hxinc(:,2)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,2)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
```

```
Hxinctilde(:,1)=(Hxinc(ze,1)+Hxinc(ze,2)+Hxinc(ze-1,1)+Hxinc(ze-
1,2))/4;
Hyinctilde(:,1)=(Hyinc(ze,1)+Hyinc(ze,2)+Hyinc(ze-1,1)+Hyinc(ze-
1,2))/4;
Hxtilde(:,1)=(Hx(ze,1)+Hx(ze,2)+Hx(ze-1,1)+Hx(ze-1,2))/4;
Hytilde(:,1)=(Hy(ze,1)+Hy(ze,2)+Hy(ze-1,1)+Hy(ze-1,2))/4;
Poyntingx(:,1)=-Ez(ze,1).*Hytilde(:,1);
Poyntingy(:,1)=Ez(ze,1).*Hxtilde(:,1);
Poyntingz(:,1)=Ex(ze,1).*Hytilde(:,1)-Ey(ze,1).*Hxtilde(:,1);
PoyntingMag(:,1)=sqrt(Poyntingx(:,1).^2+Poyntingy(:,1).^2+Poyntingz(:,1
).^2);

for n=1:2*timeperiod
   Dx(ze,n+1)=C_E(ze).*Dx(ze,n)-D_E(ze).*(Hy(ze,n+1)-Hy(ze-1,n+1));
   Dy(ze,n+1)=C_E(ze).*Dy(ze,n)+D_E(ze).*(Hx(ze,n+1)-Hx(ze-1,n+1));

Dx(kscatbot+1,n+1)=Dx(kscatbot+1,n+1)+D_E(kscatbot+1)*Hyinc(kscatbot,n+
1);
   Dy(kscatbot+1,n+1)=Dy(kscatbot+1,n+1)-
D_E(kscatbot+1)*Hxinc(kscatbot,n+1);
   Dx(kscattop+1,n+1)=Dx(kscattop+1,n+1)-
D_E(kscattop+1)*Hyinc(kscattop+1,n+1);

Dy(kscattop+1,n+1)=Dy(kscattop+1,n+1)+D_E(kscattop+1)*Hxinc(kscattop+1,
n+1);

Ex(ze,n+1)=epsinvxx(ze).*Dx(ze,n+1)+epsinvxy(ze).*Dy(ze,n+1)+epsinvxz(z
e).*Dz(ze,n+1);

Ey(ze,n+1)=epsinvxy(ze).*Dx(ze,n+1)+epsinvyy(ze).*Dy(ze,n+1)+epsinvyz(z
e).*Dz(ze,n+1);

Ez(ze,n+1)=epsinvxz(ze).*Dx(ze,n+1)+epsinvyz(ze).*Dy(ze,n+1)+epsinvzz(z
e).*Dz(ze,n+1);
   Exinc(ze,n+1)=C_E(ze).*Exinc(ze,n)-D_E(ze).*(Hyinc(ze,n+1)-Hyinc(ze-
1,n+1));
   Eyinc(ze,n+1)=C_E(ze).*Eyinc(ze,n)+D_E(ze).*(Hxinc(ze,n+1)-Hxinc(ze-
1,n+1));
   Exinc(kincstart+1,n+1)=-E0*sin(omega*(Tbreak+n)*dt)*(1-exp(-
((Tbreak+n)*dt-tdelay)^2/sigmadelay));
   Hx(:,n+2)=C_H.*Hx(:,n+1)+D_H.*(Ey(zh,n+1)-Ey(zh-1,n+1));
   Hy(:,n+2)=C_H.*Hy(:,n+1)-D_H.*(Ex(zh,n+1)-Ex(zh-1,n+1));
   Hx(kscatbot,n+2)=Hx(kscatbot,n+2)-
D_H(kscatbot)*Eyinc(kscatbot+1,n+1);

Hy(kscatbot,n+2)=Hy(kscatbot,n+2)+D_H(kscatbot)*Exinc(kscatbot+1,n+1);

Hx(kscattop+1,n+2)=Hx(kscattop+1,n+2)+D_H(kscattop+1)*Eyinc(kscattop+1,
n+1);
   Hy(kscattop+1,n+2)=Hy(kscattop+1,n+2)-
D_H(kscattop+1)*Exinc(kscattop+1,n+1);
   Hxinc(:,n+2)=C_H.*Hxinc(:,n+1)+D_H.*(Eyinc(zh,n+1)-Eyinc(zh-1,n+1));
   Hyinc(:,n+2)=C_H.*Hyinc(:,n+1)-D_H.*(Exinc(zh,n+1)-Exinc(zh-1,n+1));
   Hxinctilde(:,n+1)=(Hxinc(ze,n+1)+Hxinc(ze,n+2)+Hxinc(ze-
1,n+1)+Hxinc(ze-1,n+2))/4;
```

```
   Hyinctilde(:,n+1)=(Hyinc(ze,n+1)+Hyinc(ze,n+2)+Hyinc(ze-
1,n+1)+Hyinc(ze-1,n+2))/4;
   Hxtilde(:,n+1)=(Hx(ze,n+1)+Hx(ze,n+2)+Hx(ze-1,n+1)+Hx(ze-1,n+2))/4;
   Hytilde(:,n+1)=(Hy(ze,n+1)+Hy(ze,n+2)+Hy(ze-1,n+1)+Hy(ze-1,n+2))/4;
   Poyntingx(:,n+1)=-Ez(ze,n+1).*Hytilde(:,n+1);
   Poyntingy(:,n+1)=Ez(ze,n+1).*Hxtilde(:,n+1);
   Poyntingz(:,n+1)=Ex(ze,n+1).*Hytilde(:,n+1)-
Ey(ze,n+1).*Hxtilde(:,n+1);

PoyntingMag(:,n+1)=sqrt(Poyntingx(:,n+1).^2+Poyntingy(:,n+1).^2+Poyntin
gz(:,n+1).^2);
end
Intensity=PoyntingMag(:,1);
for n=2:2*timeperiod
    Intensity=Intensity+2*PoyntingMag(:,n);
end
Intensity=Intensity+PoyntingMag(:,2*timeperiod+1);
Intensity=Intensity*dt/E0^2/(4*pi/omega);

kPMLbot2=kPMLbot;  kscatbot2=kscatbot; kairbot2=kairbot;
kglassbot2=kglassbot;
kglasstop2=kglasstop; kairtop2=kairtop; kscattop2=kscattop;
kPMLtop2=kPMLtop;
Ntot2=Ntot; dz2=dz; dt2=dt;
Ex2=Ex; Ey2=Ey; Ez2=Ez;
Hx2=Hx; Hy2=Hy;
Exinc2=Exinc; Eyinc2=Eyinc;
Hxinc2=Hxinc; Hyinc2=Hyinc;
Hxinctilde2=Hxinctilde; Hyinctilde2=Hyinctilde;
Hxtilde2=Hxtilde; Hytilde2=Hytilde;
Poyntingx2=Poyntingx; Poyntingy2=Poyntingy; Poyntingz2=Poyntingz;
PoynitngMag2=PoyntingMag; Intensity2=Intensity;
timeperiod2=timeperiod; Tfin2=Tfin; Tbreak2=Tbreak;
kincstart2=kincstart;
zTotE2=dz*((1:Ntot+1)-kglassbot-1);
zTotH2=dz*((1:Ntot)-1/2-kglassbot);
toc
tic
%% Finer grid
refinement=2;
NLCP=NLCP*refinement           %number of numerical cells in the LCP
dz=h/NLCP;
Nglass=Nglass*refinement;       %number of numerical cells in the glass
Nair=Nair*refinement;           %number of numerical cells in the air
between glass and scattering layer
Nscat=Nscat*refinement;         %number of numerical cells in the
scattering later
NPML=NPML*refinement;           %number of numerical cells in the
perfectly matched layer
cdtoverdz=1;
dt=dz*cdtoverdz;
Tfin=floor(EndTime/dt)+1;
timeperiod=round(2*pi/omega/dt);
sigmamax=2/dt;         %in the PML, sigmaz=sigmamax((kPML-1)/(NPML-
1))^sigmam
```

```
sigmam=3;
kPMLbot=NPML;               % sigma defined but equal to zero at
k=kPMLbot
kscatbot=kPMLbot+Nscat;     % k=kscatbot is considered to be in the
total field; kscatbot-1/2 is in the scattered field
kairbot=kscatbot+Nair;      % k=kairbot is the bottom edge of the
bottom plate
kglassbot=kairbot+Nglass;   % k=kglassbot is the top edge of the bottom
plate/where the anchoring condition is applied
kglasstop=kglassbot+NLCP;   % k=kglasstop is the bottom edge of the top
plate/where the anchoring condition is applied
kairtop=kglasstop+Nglass;   % k=kairtop is the top edge of the top
plate
kscattop=kairtop+Nair;      % k=kscattop is in the total field;
kscattop+1/2 is not
kPMLtop=kscattop+Nscat;     % sigma defined but equal to zero at
k=kPMLtop
Ntot=kPMLtop+NPML;          % Ntot cells in total% ze=2:Ntot;
ze=2:Ntot;                   % k values for updating the gradients of H
in the D and E equations
zh=2:Ntot+1;
z3=linspace(0,h,NLCP+1);  %z values used for computing the orientation
kincstart=refinement*(kincstart2-kglassbot2)+kglassbot;
%% compute the permittivity
solinit=bvpinit(z3,@LEodeinit);
sol=bvp4c(@LEodefun,@LEbcfun,solinit);
u=deval(sol,z3);
psiLE=u(1,:);
thetaLE=u(3,:);

n1=cos(thetaLE).*cos(psiLE);
n2=cos(thetaLE).*sin(psiLE);
n3=sin(thetaLE);

epsxx=zeros(Ntot+1,1);
epsyy=zeros(Ntot+1,1);
epszz=zeros(Ntot+1,1);
epsxy=zeros(Ntot+1,1);
epsxz=zeros(Ntot+1,1);
epsyz=zeros(Ntot+1,1);

epsxx(1:kPMLbot+1)=1; epsyy(1:kPMLbot+1)=1; epszz(1:kPMLbot+1)=1;
epsxx(kPMLbot+2:kairbot)=epsilon_air;
epsyy(kPMLbot+2:kairbot)=epsilon_air;
epszz(kPMLbot+2:kairbot)=epsilon_air;
epsxx(kairbot+1:kglassbot)=epsilon_glass;
epsyy(kairbot+1:kglassbot)=epsilon_glass;
epszz(kairbot+1:kglassbot)=epsilon_glass;
epsxx(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n1.^2;
epsyy(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n2.^2;
epszz(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n3.^2;
epsxy(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n2;
epsxz(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n3;
epsyz(kglassbot+1:kglasstop+1)=delta_epsilon*n2.*n3;
```

```
epsxx(kglasstop+2:kairtop+1)=epsilon_glass;
epsyy(kglasstop+2:kairtop+1)=epsilon_glass;
epszz(kglasstop+2:kairtop+1)=epsilon_glass;
epsxx(kairtop+2:kPMLtop)=epsilon_air;
epsyy(kairtop+2:kPMLtop)=epsilon_air;
epszz(kairtop+2:kPMLtop)=epsilon_air;
epsxx(kPMLtop+1:Ntot)=1; epsyy(kPMLtop+1:Ntot)=1;
epszz(kPMLtop+1:Ntot)=1;
epsdet=epsxx.*(epsyy.*epszz-epsyz.^2)-epsxy.*(epsxy.*epszz-
epsxz.*epsyz)+epsxz.*(epsxy.*epsyz-epsyy.*epsxz);
epsinvxx=(epsyy.*epszz-epsyz.^2)./epsdet;
epsinvxy=(epsxz.*epsyz-epsxy.*epszz)./epsdet;
epsinvxz=(epsxy.*epsyz-epsyy.*epsxz)./epsdet;
epsinvyy=(epsxx.*epszz-epsxz.^2)./epsdet;
epsinvyz=(epsxy.*epsxz-epsxx.*epsyz)./epsdet;
epsinvzz=(epsxx.*epsyy-epsxy.^2)./epsdet;
%% Define coefficients
C_E=ones(Ntot+1,1);             % factor of the D^n term in the D
update equations
D_E=cdtoverdz*ones(Ntot+1,1);   % factor of the curl H term in the D
update equations
C_H=ones(Ntot,1);               % factor of the H^n term in the H
update equations
D_H=cdtoverdz*ones(Ntot,1);     % factor of the curl H term in the H
update equations
for k=1:kPMLbot                 % fill in the effect of sigma in the
PML
   sigmaz=sigmamax*((kPMLbot-k+1)/kPMLbot)^sigmam;
   C_E(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_E(Ntot+2-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(Ntot+2-k)=2*cdtoverdz/(2+dt*sigmaz);
   sigmaz=sigmamax*((kPMLbot-k+0.5)/kPMLbot)^sigmam;
   C_H(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_H(Ntot+1-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(Ntot+1-k)=2*cdtoverdz/(2+dt*sigmaz);
end
%%  Define electromagnetic field variables
Dx=zeros(Ntot+1,2*timeperiod+1);      % D and E are known on the space-
time grid
Dy=zeros(Ntot+1,2*timeperiod+1);      % D and E fields are zero
initially (n=0), but this is not stored.
Dz=zeros(Ntot+1,2*timeperiod+1);      % Dx and Ex are zero at k=1 and
k=Ntot+1 but they are stored so that
Ex=zeros(Ntot+1,2*timeperiod+1);      % Dx(k,n) is Dx at t=n*dt and
z=(k-kglassbot-1)*dz,
Ey=zeros(Ntot+1,2*timeperiod+1);      % or Ex_k^n is stored as
Ex(k+1,n)
Ez=zeros(Ntot+1,2*timeperiod+1);      % H is known on the half-grid.
Hx=zeros(Ntot,2*timeperiod+2);        % Hx(k,n) is Hx at t=(n-1/2)*dt,
z=(k-kglassbot+1/2)*dz
Hy=zeros(Ntot,2*timeperiod+2);        % Hx_(k+1/2)^(n+1/2) is stored as
Hx(k+1,n+1)
```

```
Exinc=zeros(Ntot+1,2*timeperiod+1);   % Hz is zero and not stored.  Ez
can be nonzero, dependending on epsilon
Eyinc=zeros(Ntot+1,2*timeperiod+1);   %
Hxinc=zeros(Ntot,2*timeperiod+2);     % Exinc, Eyinc, Hxinc, and Hyinc
are the incident field
Hyinc=zeros(Ntot,2*timeperiod+2);
Hxinctilde=zeros(Ntot-1,2*timeperiod+1);   % Hxinctilde interpolates
Hxinc onto the grid
Hyinctilde=zeros(Ntot-1,2*timeperiod+1);
Hxtilde=zeros(Ntot-1,2*timeperiod+1);      % Hxtilde interpolates Hx
onto the grid
Hytilde=zeros(Ntot-1,2*timeperiod+1);
Poyntingx=zeros(Ntot-1,2*timeperiod+1);  % The Poynting vector is known
on the space-time grid, and so H must be
Poyntingy=zeros(Ntot-1,2*timeperiod+1);  % interpolated in time and in
space.  Thus it is not defined at k=1 or k=Ntot+1
Poyntingz=zeros(Ntot-1,2*timeperiod+1);  % or at t=Tfin
PoyntingMag=zeros(Ntot-1,2*timeperiod+1);
Tbreak=Tfin-2*timeperiod-1;

% The starting main loop, not storing every step
Exinc(kincstart+1,1)=-E0*sin(omega*dt)*(1-exp(-(dt-
tdelay)^2/sigmadelay));
Hx(kscatbot,1)=-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,1)=D_H(kscatbot)*Exinc(kscatbot+1,1);
Hxinc(:,1)=D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,1)=-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
for n=2:Tbreak-1
   Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
   Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
   Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
   Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
   Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);

Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);

Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
   Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-
1,1));
   Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-
1,1));
   Exinc(kincstart+1,1)=-E0*sin(omega*n*dt)*(1-exp(-(n*dt-
tdelay)^2/sigmadelay));
   Hx(:,1)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
   Hy(:,1)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
   Hx(kscatbot,1)=Hx(kscatbot,1)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
   Hy(kscatbot,1)=Hy(kscatbot,1)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,1)=Hx(kscattop+1,1)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
```

```
   Hy(kscattop+1,1)=Hy(kscattop+1,1)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
   Hxinc(:,1)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
   Hyinc(:,1)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
end
Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
Dx(kscattop+1,1)=Dx(kscattop+1,1)-D_E(kscattop+1)*Hyinc(kscattop+1,1);
Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);
Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);
Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
Ez(ze,1)=epsinvxz(ze).*Dx(ze,1)+epsinvyz(ze).*Dy(ze,1)+epsinvzz(ze).*Dz
(ze,1);
Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-1,1));
Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-1,1));
Exinc(kincstart+1,1)=-E0*sin(omega*Tbreak*dt)*(1-exp(-(Tbreak*dt-
tdelay)^2/sigmadelay));
Hx(:,2)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
Hy(:,2)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
Hx(kscatbot,2)=Hx(kscatbot,2)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,2)=Hy(kscatbot,2)+D_H(kscatbot)*Exinc(kscatbot+1,1);
Hx(kscattop+1,2)=Hx(kscattop+1,2)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
Hy(kscattop+1,2)=Hy(kscattop+1,2)-D_H(kscattop+1)*Exinc(kscattop+1,1);
Hxinc(:,2)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,2)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
Hxinctilde(:,1)=(Hxinc(ze,1)+Hxinc(ze,2)+Hxinc(ze-1,1)+Hxinc(ze-
1,2))/4;
Hyinctilde(:,1)=(Hyinc(ze,1)+Hyinc(ze,2)+Hyinc(ze-1,1)+Hyinc(ze-
1,2))/4;
Hxtilde(:,1)=(Hx(ze,1)+Hx(ze,2)+Hx(ze-1,1)+Hx(ze-1,2))/4;
Hytilde(:,1)=(Hy(ze,1)+Hy(ze,2)+Hy(ze-1,1)+Hy(ze-1,2))/4;
Poyntingx(:,1)=-Ez(ze,1).*Hytilde(:,1);
Poyntingy(:,1)=Ez(ze,1).*Hxtilde(:,1);
Poyntingz(:,1)=Ex(ze,1).*Hytilde(:,1)-Ey(ze,1).*Hxtilde(:,1);
PoyntingMag(:,1)=sqrt(Poyntingx(:,1).^2+Poyntingy(:,1).^2+Poyntingz(:,1
).^2);
for n=1:2*timeperiod
   Dx(ze,n+1)=C_E(ze).*Dx(ze,n)-D_E(ze).*(Hy(ze,n+1)-Hy(ze-1,n+1));
   Dy(ze,n+1)=C_E(ze).*Dy(ze,n)+D_E(ze).*(Hx(ze,n+1)-Hx(ze-1,n+1));

Dx(kscatbot+1,n+1)=Dx(kscatbot+1,n+1)+D_E(kscatbot+1)*Hyinc(kscatbot,n+
1);
   Dy(kscatbot+1,n+1)=Dy(kscatbot+1,n+1)-
D_E(kscatbot+1)*Hxinc(kscatbot,n+1);
   Dx(kscattop+1,n+1)=Dx(kscattop+1,n+1)-
D_E(kscattop+1)*Hyinc(kscattop+1,n+1);

Dy(kscattop+1,n+1)=Dy(kscattop+1,n+1)+D_E(kscattop+1)*Hxinc(kscattop+1,
n+1);
```

```
Ex(ze,n+1)=epsinvxx(ze).*Dx(ze,n+1)+epsinvxy(ze).*Dy(ze,n+1)+epsinvxz(z
e).*Dz(ze,n+1);

Ey(ze,n+1)=epsinvxy(ze).*Dx(ze,n+1)+epsinvyy(ze).*Dy(ze,n+1)+epsinvyz(z
e).*Dz(ze,n+1);

Ez(ze,n+1)=epsinvxz(ze).*Dx(ze,n+1)+epsinvyz(ze).*Dy(ze,n+1)+epsinvzz(z
e).*Dz(ze,n+1);
   Exinc(ze,n+1)=C_E(ze).*Exinc(ze,n)-D_E(ze).*(Hyinc(ze,n+1)-Hyinc(ze-
1,n+1));
   Eyinc(ze,n+1)=C_E(ze).*Eyinc(ze,n)+D_E(ze).*(Hxinc(ze,n+1)-Hxinc(ze-
1,n+1));
   Exinc(kincstart+1,n+1)=-E0*sin(omega*(Tbreak+n)*dt)*(1-exp(-
((Tbreak+n)*dt-tdelay)^2/sigmadelay));
   Hx(:,n+2)=C_H.*Hx(:,n+1)+D_H.*(Ey(zh,n+1)-Ey(zh-1,n+1));
   Hy(:,n+2)=C_H.*Hy(:,n+1)-D_H.*(Ex(zh,n+1)-Ex(zh-1,n+1));
   Hx(kscatbot,n+2)=Hx(kscatbot,n+2)-
D_H(kscatbot)*Eyinc(kscatbot+1,n+1);

Hy(kscatbot,n+2)=Hy(kscatbot,n+2)+D_H(kscatbot)*Exinc(kscatbot+1,n+1);

Hx(kscattop+1,n+2)=Hx(kscattop+1,n+2)+D_H(kscattop+1)*Eyinc(kscattop+1,
n+1);
   Hy(kscattop+1,n+2)=Hy(kscattop+1,n+2)-
D_H(kscattop+1)*Exinc(kscattop+1,n+1);
   Hxinc(:,n+2)=C_H.*Hxinc(:,n+1)+D_H.*(Eyinc(zh,n+1)-Eyinc(zh-1,n+1));
   Hyinc(:,n+2)=C_H.*Hyinc(:,n+1)-D_H.*(Exinc(zh,n+1)-Exinc(zh-1,n+1));
   Hxinctilde(:,n+1)=(Hxinc(ze,n+1)+Hxinc(ze,n+2)+Hxinc(ze-
1,n+1)+Hxinc(ze-1,n+2))/4;
   Hyinctilde(:,n+1)=(Hyinc(ze,n+1)+Hyinc(ze,n+2)+Hyinc(ze-
1,n+1)+Hyinc(ze-1,n+2))/4;
   Hxtilde(:,n+1)=(Hx(ze,n+1)+Hx(ze,n+2)+Hx(ze-1,n+1)+Hx(ze-1,n+2))/4;
   Hytilde(:,n+1)=(Hy(ze,n+1)+Hy(ze,n+2)+Hy(ze-1,n+1)+Hy(ze-1,n+2))/4;
   Poyntingx(:,n+1)=-Ez(ze,n+1).*Hytilde(:,n+1);
   Poyntingy(:,n+1)=Ez(ze,n+1).*Hxtilde(:,n+1);
   Poyntingz(:,n+1)=Ex(ze,n+1).*Hytilde(:,n+1)-
Ey(ze,n+1).*Hxtilde(:,n+1);

PoyntingMag(:,n+1)=sqrt(Poyntingx(:,n+1).^2+Poyntingy(:,n+1).^2+Poyntin
gz(:,n+1).^2);
end
Intensity=PoyntingMag(:,1);
for n=2:2*timeperiod
    Intensity=Intensity+2*PoyntingMag(:,n);
end
Intensity=Intensity+PoyntingMag(:,2*timeperiod+1);
Intensity=Intensity*dt/E0^2/(4*pi/omega);

kPMLbot3=kPMLbot; kscatbot3=kscatbot; kairbot3=kairbot;
kglassbot3=kglassbot;
kglasstop3=kglasstop; kairtop3=kairtop; kscattop3=kscattop;
kPMLtop3=kPMLtop;
Ntot3=Ntot; dz3=dz; dt3=dt;
Ex3=Ex; Ey3=Ey; Ez3=Ez;
```

```
Hx3=Hx; Hy3=Hy;
Exinc3=Exinc; Eyinc3=Eyinc;
Hxinc3=Hxinc; Hyinc3=Hyinc;
Hxinctilde3=Hxinctilde; Hyinctilde3=Hyinctilde;
Hxtilde3=Hxtilde; Hytilde3=Hytilde;
Poyntingx3=Poyntingx; Poyntingy3=Poyntingy; Poyntingz3=Poyntingz;
PoynitngMag3=PoyntingMag; Intensity3=Intensity;
timeperiod3=timeperiod; Tfin3=Tfin; Tbreak3=Tbreak;
kincstart3=kincstart;
zTotE3=dz*((1:Ntot+1)-kglassbot-1);
zTotH3=dz*((1:Ntot)-1/2-kglassbot);
toc

tic
%% Finer grid
refinement=2;
NLCP=NLCP*refinement           %number of numerical cells in the LCP
dz=h/NLCP;
Nglass=Nglass*refinement;       %number of numerical cells in the glass
Nair=Nair*refinement;           %number of numerical cells in the air
between glass and scattering layer
Nscat=Nscat*refinement;         %number of numerical cells in the
scattering later
NPML=NPML*refinement;           %number of numerical cells in the
perfectly matched layer
cdtoverdz=1;
dt=dz*cdtoverdz;
Tfin=floor(EndTime/dt)+1;
timeperiod=round(2*pi/omega/dt);
sigmamax=2/dt; %10^17;        %in the PML, sigmaz=sigmamax((kPML-
1)/(NPML-1))^sigmam
sigmam=3;
kPMLbot=NPML;               % sigma defined but equal to zero at
k=kPMLbot
kscatbot=kPMLbot+Nscat;     % k=kscatbot is considered to be in the
total field; kscatbot-1/2 is in the scattered field
kairbot=kscatbot+Nair;      % k=kairbot is the bottom edge of the
bottom plate
kglassbot=kairbot+Nglass;   % k=kglassbot is the top edge of the bottom
plate/where the anchoring condition is applied
kglasstop=kglassbot+NLCP;   % k=kglasstop is the bottom edge of the top
plate/where the anchoring condition is applied
kairtop=kglasstop+Nglass;   % k=kairtop is the top edge of the top
plate
kscattop=kairtop+Nair;      % k=kscattop is in the total field;
kscattop+1/2 is not
kPMLtop=kscattop+Nscat;     % sigma defined but equal to zero at
k=kPMLtop
Ntot=kPMLtop+NPML;          % Ntot cells in total% ze=2:Ntot;
ze=2:Ntot;                   % k values for updating the gradients of H
in the D and E equations
zh=2:Ntot+1;
z4=linspace(0,h,NLCP+1);  %z values used for computing the orientation
kincstart=refinement*(kincstart3-kglassbot3)+kglassbot;
```

```
%% compute the permittivity
solinit=bvpinit(z4,@LEodeinit);
sol=bvp4c(@LEodefun,@LEbcfun,solinit);
u=deval(sol,z4);
psiLE=u(1,:);
thetaLE=u(3,:);

n1=cos(thetaLE).*cos(psiLE);
n2=cos(thetaLE).*sin(psiLE);
n3=sin(thetaLE);

epsxx=zeros(Ntot+1,1);
epsyy=zeros(Ntot+1,1);
epszz=zeros(Ntot+1,1);
epsxy=zeros(Ntot+1,1);
epsxz=zeros(Ntot+1,1);
epsyz=zeros(Ntot+1,1);

epsxx(1:kPMLbot+1)=1; epsyy(1:kPMLbot+1)=1; epszz(1:kPMLbot+1)=1;
epsxx(kPMLbot+2:kairbot)=epsilon_air;
epsyy(kPMLbot+2:kairbot)=epsilon_air;
epszz(kPMLbot+2:kairbot)=epsilon_air;
epsxx(kairbot+1:kglassbot)=epsilon_glass;
epsyy(kairbot+1:kglassbot)=epsilon_glass;
epszz(kairbot+1:kglassbot)=epsilon_glass;
epsxx(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n1.^2;
epsyy(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n2.^2;
epszz(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n3.^2;
epsxy(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n2;
epsxz(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n3;
epsyz(kglassbot+1:kglasstop+1)=delta_epsilon*n2.*n3;
epsxx(kglasstop+2:kairtop+1)=epsilon_glass;
epsyy(kglasstop+2:kairtop+1)=epsilon_glass;
epszz(kglasstop+2:kairtop+1)=epsilon_glass;
epsxx(kairtop+2:kPMLtop)=epsilon_air;
epsyy(kairtop+2:kPMLtop)=epsilon_air;
epszz(kairtop+2:kPMLtop)=epsilon_air;
epsxx(kPMLtop+1:Ntot)=1; epsyy(kPMLtop+1:Ntot)=1;
epszz(kPMLtop+1:Ntot)=1;

epsdet=epsxx.*(epsyy.*epszz-epsyz.^2)-epsxy.*(epsxy.*epszz-
epsxz.*epsyz)+epsxz.*(epsxy.*epsyz-epsyy.*epsxz);
epsinvxx=(epsyy.*epszz-epsyz.^2)./epsdet;
epsinvxy=(epsxz.*epsyz-epsxy.*epszz)./epsdet;
epsinvxz=(epsxy.*epsyz-epsyy.*epsxz)./epsdet;
epsinvyy=(epsxx.*epszz-epsxz.^2)./epsdet;
epsinvyz=(epsxy.*epsxz-epsxx.*epsyz)./epsdet;
epsinvzz=(epsxx.*epsyy-epsxy.^2)./epsdet;

%% Define coefficients
C_E=ones(Ntot+1,1);             % factor of the D^n term in the D
update equations
D_E=cdtoverdz*ones(Ntot+1,1);   % factor of the curl H term in the D
update equations
```

```
C_H=ones(Ntot,1);               % factor of the H^n term in the H
update equations
D_H=cdtoverdz*ones(Ntot,1);     % factor of the curl H term in the H
update equations
for k=1:kPMLbot                 % fill in the effect of sigma in the
PML
   sigmaz=sigmamax*((kPMLbot-k+1)/kPMLbot)^sigmam;
   C_E(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_E(Ntot+2-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(Ntot+2-k)=2*cdtoverdz/(2+dt*sigmaz);
   sigmaz=sigmamax*((kPMLbot-k+0.5)/kPMLbot)^sigmam;
   C_H(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_H(Ntot+1-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(Ntot+1-k)=2*cdtoverdz/(2+dt*sigmaz);
end

%%  Define electromagnetic field variables
Dx=zeros(Ntot+1,2*timeperiod+1);      % D and E are known on the space-
time grid
Dy=zeros(Ntot+1,2*timeperiod+1);      % D and E fields are zero
initially (n=0), but this is not stored.
Dz=zeros(Ntot+1,2*timeperiod+1);      % Dx and Ex are zero at k=1 and
k=Ntot+1 but they are stored so that
Ex=zeros(Ntot+1,2*timeperiod+1);      % Dx(k,n) is Dx at t=n*dt and
z=(k-kglassbot-1)*dz,
Ey=zeros(Ntot+1,2*timeperiod+1);      % or Ex_k^n is stored as
Ex(k+1,n)
Ez=zeros(Ntot+1,2*timeperiod+1);      % H is known on the half-grid.
Hx=zeros(Ntot,2*timeperiod+2);        % Hx(k,n) is Hx at t=(n-1/2)*dt,
z=(k-kglassbot+1/2)*dz
Hy=zeros(Ntot,2*timeperiod+2);        % Hx_(k+1/2)^(n+1/2) is stored as
Hx(k+1,n+1)
Exinc=zeros(Ntot+1,2*timeperiod+1);   % Hz is zero and not stored.  Ez
can be nonzero, depending on epsilon
Eyinc=zeros(Ntot+1,2*timeperiod+1);   %
Hxinc=zeros(Ntot,2*timeperiod+2);     % Exinc, Eyinc, Hxinc, and Hyinc
are the incident field
Hyinc=zeros(Ntot,2*timeperiod+2);
Hxinctilde=zeros(Ntot-1,2*timeperiod+1);   % Hxinctilde interpolates
Hxinc onto the grid
Hyinctilde=zeros(Ntot-1,2*timeperiod+1);
Hxtilde=zeros(Ntot-1,2*timeperiod+1);      % Hxtilde interpolates Hx
onto the grid
Hytilde=zeros(Ntot-1,2*timeperiod+1);
Poyntingx=zeros(Ntot-1,2*timeperiod+1);  % The Poynting vector is known
on the space-time grid, and so H must be
Poyntingy=zeros(Ntot-1,2*timeperiod+1);  % interpolated in time and in
space.  Thus it is not defined at k=1 or k=Ntot+1
Poyntingz=zeros(Ntot-1,2*timeperiod+1);  % or at t=Tfin
PoyntingMag=zeros(Ntot-1,2*timeperiod+1);
Tbreak=Tfin-2*timeperiod-1;

% The starting main loop, not storing every step
```

```
Exinc(kincstart+1,1)=-E0*sin(omega*dt)*(1-exp(-(dt-
tdelay)^2/sigmadelay));
Hx(kscatbot,1)=-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,1)=D_H(kscatbot)*Exinc(kscatbot+1,1);
Hxinc(:,1)=D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,1)=-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
for n=2:Tbreak-1
   Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
   Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
   Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
   Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
   Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);

Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);

Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
   Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-
1,1));
   Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-
1,1));
   Exinc(kincstart+1,1)=-E0*sin(omega*n*dt)*(1-exp(-(n*dt-
tdelay)^2/sigmadelay));
   Hx(:,1)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
   Hy(:,1)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
   Hx(kscatbot,1)=Hx(kscatbot,1)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
   Hy(kscatbot,1)=Hy(kscatbot,1)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,1)=Hx(kscattop+1,1)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
   Hy(kscattop+1,1)=Hy(kscattop+1,1)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
   Hxinc(:,1)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
   Hyinc(:,1)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
end
%n=Tbreak;
Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
Dx(kscattop+1,1)=Dx(kscattop+1,1)-D_E(kscattop+1)*Hyinc(kscattop+1,1);
Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);
Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);
Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
Ez(ze,1)=epsinvxz(ze).*Dx(ze,1)+epsinvyz(ze).*Dy(ze,1)+epsinvzz(ze).*Dz
(ze,1);
Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-1,1));
Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-1,1));
Exinc(kincstart+1,1)=-E0*sin(omega*Tbreak*dt)*(1-exp(-(Tbreak*dt-
tdelay)^2/sigmadelay));
```

```
Hx(:,2)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
Hy(:,2)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
Hx(kscatbot,2)=Hx(kscatbot,2)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,2)=Hy(kscatbot,2)+D_H(kscatbot)*Exinc(kscatbot+1,1);
Hx(kscattop+1,2)=Hx(kscattop+1,2)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
Hy(kscattop+1,2)=Hy(kscattop+1,2)-D_H(kscattop+1)*Exinc(kscattop+1,1);
Hxinc(:,2)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,2)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
Hxinctilde(:,1)=(Hxinc(ze,1)+Hxinc(ze,2)+Hxinc(ze-1,1)+Hxinc(ze-
1,2))/4;
Hyinctilde(:,1)=(Hyinc(ze,1)+Hyinc(ze,2)+Hyinc(ze-1,1)+Hyinc(ze-
1,2))/4;
Hxtilde(:,1)=(Hx(ze,1)+Hx(ze,2)+Hx(ze-1,1)+Hx(ze-1,2))/4;
Hytilde(:,1)=(Hy(ze,1)+Hy(ze,2)+Hy(ze-1,1)+Hy(ze-1,2))/4;
Poyntingx(:,1)=-Ez(ze,1).*Hytilde(:,1);
Poyntingy(:,1)=Ez(ze,1).*Hxtilde(:,1);
Poyntingz(:,1)=Ex(ze,1).*Hytilde(:,1)-Ey(ze,1).*Hxtilde(:,1);
PoyntingMag(:,1)=sqrt(Poyntingx(:,1).^2+Poyntingy(:,1).^2+Poyntingz(:,1
).^2);
%for n=Tbreak:Tfin
%makes Dx(:,1) actually at n=Tfin-1-2*timeperiod
for n=1:2*timeperiod
   Dx(ze,n+1)=C_E(ze).*Dx(ze,n)-D_E(ze).*(Hy(ze,n+1)-Hy(ze-1,n+1));
   Dy(ze,n+1)=C_E(ze).*Dy(ze,n)+D_E(ze).*(Hx(ze,n+1)-Hx(ze-1,n+1));

Dx(kscatbot+1,n+1)=Dx(kscatbot+1,n+1)+D_E(kscatbot+1)*Hyinc(kscatbot,n+
1);
   Dy(kscatbot+1,n+1)=Dy(kscatbot+1,n+1)-
D_E(kscatbot+1)*Hxinc(kscatbot,n+1);
   Dx(kscattop+1,n+1)=Dx(kscattop+1,n+1)-
D_E(kscattop+1)*Hyinc(kscattop+1,n+1);

Dy(kscattop+1,n+1)=Dy(kscattop+1,n+1)+D_E(kscattop+1)*Hxinc(kscattop+1,
n+1);

Ex(ze,n+1)=epsinvxx(ze).*Dx(ze,n+1)+epsinvxy(ze).*Dy(ze,n+1)+epsinvxz(z
e).*Dz(ze,n+1);

Ey(ze,n+1)=epsinvxy(ze).*Dx(ze,n+1)+epsinvyy(ze).*Dy(ze,n+1)+epsinvyz(z
e).*Dz(ze,n+1);

Ez(ze,n+1)=epsinvxz(ze).*Dx(ze,n+1)+epsinvyz(ze).*Dy(ze,n+1)+epsinvzz(z
e).*Dz(ze,n+1);
   Exinc(ze,n+1)=C_E(ze).*Exinc(ze,n)-D_E(ze).*(Hyinc(ze,n+1)-Hyinc(ze-
1,n+1));
   Eyinc(ze,n+1)=C_E(ze).*Eyinc(ze,n)+D_E(ze).*(Hxinc(ze,n+1)-Hxinc(ze-
1,n+1));
   Exinc(kincstart+1,n+1)=-E0*sin(omega*(Tbreak+n)*dt)*(1-exp(-
((Tbreak+n)*dt-tdelay)^2/sigmadelay));
   Hx(:,n+2)=C_H.*Hx(:,n+1)+D_H.*(Ey(zh,n+1)-Ey(zh-1,n+1));
   Hy(:,n+2)=C_H.*Hy(:,n+1)-D_H.*(Ex(zh,n+1)-Ex(zh-1,n+1));
   Hx(kscatbot,n+2)=Hx(kscatbot,n+2)-
D_H(kscatbot)*Eyinc(kscatbot+1,n+1);

Hy(kscatbot,n+2)=Hy(kscatbot,n+2)+D_H(kscatbot)*Exinc(kscatbot+1,n+1);
```

```
Hx(kscattop+1,n+2)=Hx(kscattop+1,n+2)+D_H(kscattop+1)*Eyinc(kscattop+1,
n+1);
   Hy(kscattop+1,n+2)=Hy(kscattop+1,n+2)-
D_H(kscattop+1)*Exinc(kscattop+1,n+1);
   Hxinc(:,n+2)=C_H.*Hxinc(:,n+1)+D_H.*(Eyinc(zh,n+1)-Eyinc(zh-1,n+1));
   Hyinc(:,n+2)=C_H.*Hyinc(:,n+1)-D_H.*(Exinc(zh,n+1)-Exinc(zh-1,n+1));
   Hxinctilde(:,n+1)=(Hxinc(ze,n+1)+Hxinc(ze,n+2)+Hxinc(ze-
1,n+1)+Hxinc(ze-1,n+2))/4;
   Hyinctilde(:,n+1)=(Hyinc(ze,n+1)+Hyinc(ze,n+2)+Hyinc(ze-
1,n+1)+Hyinc(ze-1,n+2))/4;
   Hxtilde(:,n+1)=(Hx(ze,n+1)+Hx(ze,n+2)+Hx(ze-1,n+1)+Hx(ze-1,n+2))/4;
   Hytilde(:,n+1)=(Hy(ze,n+1)+Hy(ze,n+2)+Hy(ze-1,n+1)+Hy(ze-1,n+2))/4;
   Poyntingx(:,n+1)=-Ez(ze,n+1).*Hytilde(:,n+1);
   Poyntingy(:,n+1)=Ez(ze,n+1).*Hxtilde(:,n+1);
   Poyntingz(:,n+1)=Ex(ze,n+1).*Hytilde(:,n+1)-
Ey(ze,n+1).*Hxtilde(:,n+1);

PoyntingMag(:,n+1)=sqrt(Poyntingx(:,n+1).^2+Poyntingy(:,n+1).^2+Poyntin
gz(:,n+1).^2);
end
Intensity=PoyntingMag(:,1);
for n=2:2*timeperiod
    Intensity=Intensity+2*PoyntingMag(:,n);
end
Intensity=Intensity+PoyntingMag(:,2*timeperiod+1);
Intensity=Intensity*dt/E0^2/(4*pi/omega);

kPMLbot4=kPMLbot; kscatbot4=kscatbot; kairbot4=kairbot;
kglassbot4=kglassbot;
kglasstop4=kglasstop; kairtop4=kairtop; kscattop4=kscattop;
kPMLtop4=kPMLtop;
Ntot4=Ntot; dz4=dz; dt4=dt;
Ex4=Ex; Ey4=Ey; Ez4=Ez;
Hx4=Hx; Hy4=Hy;
Exinc4=Exinc; Eyinc4=Eyinc;
Hxinc4=Hxinc; Hyinc4=Hyinc;
Hxinctilde4=Hxinctilde; Hyinctilde4=Hyinctilde;
Hxtilde4=Hxtilde; Hytilde4=Hytilde;
Poyntingx4=Poyntingx; Poyntingy4=Poyntingy; Poyntingz4=Poyntingz;
PoynitngMag4=PoyntingMag; Intensity4=Intensity;
timeperiod4=timeperiod; Tfin4=Tfin; Tbreak4=Tbreak;
kincstart4=kincstart;
zTotE4=dz*((1:Ntot+1)-kglassbot-1);
zTotH4=dz*((1:Ntot)-1/2-kglassbot);
toc

tic
%% Finer grid
refinement=2;
NLCP=NLCP*refinement           %number of numerical cells in the LCP
dz=h/NLCP;
Nglass=Nglass*refinement;       %number of numerical cells in the glass
Nair=Nair*refinement;           %number of numerical cells in the air
between glass and scattering layer
```

```
Nscat=Nscat*refinement;         %number of numerical cells in the
scattering later
NPML=NPML*refinement;           %number of numerical cells in the
perfectly matched layer
cdtoverdz=1;
dt=dz*cdtoverdz;
Tfin=floor(EndTime/dt)+1;
timeperiod=round(2*pi/omega/dt);
sigmamax=2/dt; %10^17;        %in the PML, sigmaz=sigmamax((kPML-
1)/(NPML-1))^sigmam
sigmam=3;
kPMLbot=NPML;               % sigma defined but equal to zero at
k=kPMLbot
kscatbot=kPMLbot+Nscat;     % k=kscatbot is considered to be in the
total field; kscatbot-1/2 is in the scattered field
kairbot=kscatbot+Nair;      % k=kairbot is the bottom edge of the
bottom plate
kglassbot=kairbot+Nglass;   % k=kglassbot is the top edge of the bottom
plate/where the anchoring condition is applied
kglasstop=kglassbot+NLCP;   % k=kglasstop is the bottom edge of the top
plate/where the anchoring condition is applied
kairtop=kglasstop+Nglass;   % k=kairtop is the top edge of the top
plate
kscattop=kairtop+Nair;      % k=kscattop is in the total field;
kscattop+1/2 is not
kPMLtop=kscattop+Nscat;     % sigma defined but equal to zero at
k=kPMLtop
Ntot=kPMLtop+NPML;          % Ntot cells in total% ze=2:Ntot;
ze=2:Ntot;                   % k values for updating the gradients of H
in the D and E equations
zh=2:Ntot+1;
z5=linspace(0,h,NLCP+1);  %z values used for computing the orientation
kincstart=refinement*(kincstart4-kglassbot4)+kglassbot;

%% compute the permittivity
solinit=bvpinit(z5,@LEodeinit);
sol=bvp4c(@LEodefun,@LEbcfun,solinit);
u=deval(sol,z5);
psiLE=u(1,:);
thetaLE=u(3,:);

n1=cos(thetaLE).*cos(psiLE);
n2=cos(thetaLE).*sin(psiLE);
n3=sin(thetaLE);

epsxx=zeros(Ntot+1,1);
epsyy=zeros(Ntot+1,1);
epszz=zeros(Ntot+1,1);
epsxy=zeros(Ntot+1,1);
epsxz=zeros(Ntot+1,1);
epsyz=zeros(Ntot+1,1);

epsxx(1:kPMLbot+1)=1; epsyy(1:kPMLbot+1)=1; epszz(1:kPMLbot+1)=1;
```

```
epsxx(kPMLbot+2:kairbot)=epsilon_air;
epsyy(kPMLbot+2:kairbot)=epsilon_air;
epszz(kPMLbot+2:kairbot)=epsilon_air;
epsxx(kairbot+1:kglassbot)=epsilon_glass;
epsyy(kairbot+1:kglassbot)=epsilon_glass;
epszz(kairbot+1:kglassbot)=epsilon_glass;
epsxx(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n1.^2;
epsyy(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n2.^2;
epszz(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n3.^2;
epsxy(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n2;
epsxz(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n3;
epsyz(kglassbot+1:kglasstop+1)=delta_epsilon*n2.*n3;
epsxx(kglasstop+2:kairtop+1)=epsilon_glass;
epsyy(kglasstop+2:kairtop+1)=epsilon_glass;
epszz(kglasstop+2:kairtop+1)=epsilon_glass;
epsxx(kairtop+2:kPMLtop)=epsilon_air;
epsyy(kairtop+2:kPMLtop)=epsilon_air;
epszz(kairtop+2:kPMLtop)=epsilon_air;
epsxx(kPMLtop+1:Ntot)=1; epsyy(kPMLtop+1:Ntot)=1;
epszz(kPMLtop+1:Ntot)=1;

epsdet=epsxx.*(epsyy.*epszz-epsyz.^2)-epsxy.*(epsxy.*epszz-
epsxz.*epsyz)+epsxz.*(epsxy.*epsyz-epsyy.*epsxz);
epsinvxx=(epsyy.*epszz-epsyz.^2)./epsdet;
epsinvxy=(epsxz.*epsyz-epsxy.*epszz)./epsdet;
epsinvxz=(epsxy.*epsyz-epsyy.*epsxz)./epsdet;
epsinvyy=(epsxx.*epszz-epsxz.^2)./epsdet;
epsinvyz=(epsxy.*epsxz-epsxx.*epsyz)./epsdet;
epsinvzz=(epsxx.*epsyy-epsxy.^2)./epsdet;

%% Define coefficients
C_E=ones(Ntot+1,1);             % factor of the D^n term in the D
update equations
D_E=cdtoverdz*ones(Ntot+1,1);   % factor of the curl H term in the D
update equations
C_H=ones(Ntot,1);               % factor of the H^n term in the H
update equations
D_H=cdtoverdz*ones(Ntot,1);     % factor of the curl H term in the H
update equations
for k=1:kPMLbot                 % fill in the effect of sigma in the
PML
   sigmaz=sigmamax*((kPMLbot-k+1)/kPMLbot)^sigmam;
   C_E(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_E(Ntot+2-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(Ntot+2-k)=2*cdtoverdz/(2+dt*sigmaz);
   sigmaz=sigmamax*((kPMLbot-k+0.5)/kPMLbot)^sigmam;
   C_H(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_H(Ntot+1-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(Ntot+1-k)=2*cdtoverdz/(2+dt*sigmaz);
end

%%  Define electromagnetic field variables
```

```
Dx=zeros(Ntot+1,2*timeperiod+1);      % D and E are known on the space-
time grid
Dy=zeros(Ntot+1,2*timeperiod+1);      % D and E fields are zero
initially (n=0), but this is not stored.
Dz=zeros(Ntot+1,2*timeperiod+1);      % Dx and Ex are zero at k=1 and
k=Ntot+1 but they are stored so that
Ex=zeros(Ntot+1,2*timeperiod+1);      % Dx(k,n) is Dx at t=n*dt and
z=(k-kglassbot-1)*dz,
Ey=zeros(Ntot+1,2*timeperiod+1);      % or Ex_k^n is stored as
Ex(k+1,n)
Ez=zeros(Ntot+1,2*timeperiod+1);      % H is known on the half-grid.
Hx=zeros(Ntot,2*timeperiod+2);        % Hx(k,n) is Hx at t=(n-1/2)*dt,
z=(k-kglassbot+1/2)*dz
Hy=zeros(Ntot,2*timeperiod+2);        % Hx_(k+1/2)^(n+1/2) is stored as
Hx(k+1,n+1)
Exinc=zeros(Ntot+1,2*timeperiod+1);   % Hz is zero and not stored.  Ez
can be nonzero, dependending on epsilon
Eyinc=zeros(Ntot+1,2*timeperiod+1);   %
Hxinc=zeros(Ntot,2*timeperiod+2);     % Exinc, Eyinc, Hxinc, and Hyinc
are the incident field
Hyinc=zeros(Ntot,2*timeperiod+2);
Hxinctilde=zeros(Ntot-1,2*timeperiod+1);   % Hxinctilde interpolates
Hxinc onto the grid
Hyinctilde=zeros(Ntot-1,2*timeperiod+1);
Hxtilde=zeros(Ntot-1,2*timeperiod+1);      % Hxtilde interpolates Hx
onto the grid
Hytilde=zeros(Ntot-1,2*timeperiod+1);
Poyntingx=zeros(Ntot-1,2*timeperiod+1);  % The Poynting vector is known
on the space-time grid, and so H must be
Poyntingy=zeros(Ntot-1,2*timeperiod+1);  % interpolated in time and in
space.  Thus it is not defined at k=1 or k=Ntot+1
Poyntingz=zeros(Ntot-1,2*timeperiod+1);  % or at t=Tfin
PoyntingMag=zeros(Ntot-1,2*timeperiod+1);
Tbreak=Tfin-2*timeperiod-1;

% The starting main loop, not storing every step
Exinc(kincstart+1,1)=-E0*sin(omega*dt)*(1-exp(-(dt-
tdelay)^2/sigmadelay));
Hx(kscatbot,1)=-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,1)=D_H(kscatbot)*Exinc(kscatbot+1,1);
Hxinc(:,1)=D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,1)=-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
for n=2:Tbreak-1
   Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
   Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
   Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
   Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
   Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);

Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);
```

```
Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
   Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-
1,1));
   Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-
1,1));
   Exinc(kincstart+1,1)=-E0*sin(omega*n*dt)*(1-exp(-(n*dt-
tdelay)^2/sigmadelay));
   Hx(:,1)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
   Hy(:,1)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
   Hx(kscatbot,1)=Hx(kscatbot,1)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
   Hy(kscatbot,1)=Hy(kscatbot,1)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,1)=Hx(kscattop+1,1)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
   Hy(kscattop+1,1)=Hy(kscattop+1,1)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
   Hxinc(:,1)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
   Hyinc(:,1)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
end
%n=Tbreak;
Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
Dx(kscattop+1,1)=Dx(kscattop+1,1)-D_E(kscattop+1)*Hyinc(kscattop+1,1);
Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);
Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);
Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
Ez(ze,1)=epsinvxz(ze).*Dx(ze,1)+epsinvyz(ze).*Dy(ze,1)+epsinvzz(ze).*Dz
(ze,1);
Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-1,1));
Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-1,1));
Exinc(kincstart+1,1)=-E0*sin(omega*Tbreak*dt)*(1-exp(-(Tbreak*dt-
tdelay)^2/sigmadelay));
Hx(:,2)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
Hy(:,2)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
Hx(kscatbot,2)=Hx(kscatbot,2)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,2)=Hy(kscatbot,2)+D_H(kscatbot)*Exinc(kscatbot+1,1);
Hx(kscattop+1,2)=Hx(kscattop+1,2)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
Hy(kscattop+1,2)=Hy(kscattop+1,2)-D_H(kscattop+1)*Exinc(kscattop+1,1);
Hxinc(:,2)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,2)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
Hxinctilde(:,1)=(Hxinc(ze,1)+Hxinc(ze,2)+Hxinc(ze-1,1)+Hxinc(ze-
1,2))/4;
Hyinctilde(:,1)=(Hyinc(ze,1)+Hyinc(ze,2)+Hyinc(ze-1,1)+Hyinc(ze-
1,2))/4;
Hxtilde(:,1)=(Hx(ze,1)+Hx(ze,2)+Hx(ze-1,1)+Hx(ze-1,2))/4;
Hytilde(:,1)=(Hy(ze,1)+Hy(ze,2)+Hy(ze-1,1)+Hy(ze-1,2))/4;
Poyntingx(:,1)=-Ez(ze,1).*Hytilde(:,1);
Poyntingy(:,1)=Ez(ze,1).*Hxtilde(:,1);
Poyntingz(:,1)=Ex(ze,1).*Hytilde(:,1)-Ey(ze,1).*Hxtilde(:,1);
```

```
PoyntingMag(:,1)=sqrt(Poyntingx(:,1).^2+Poyntingy(:,1).^2+Poyntingz(:,1
).^2);
%for n=Tbreak:Tfin
%makes Dx(:,1) actually at n=Tfin-1-2*timeperiod
for n=1:2*timeperiod
   Dx(ze,n+1)=C_E(ze).*Dx(ze,n)-D_E(ze).*(Hy(ze,n+1)-Hy(ze-1,n+1));
   Dy(ze,n+1)=C_E(ze).*Dy(ze,n)+D_E(ze).*(Hx(ze,n+1)-Hx(ze-1,n+1));

Dx(kscatbot+1,n+1)=Dx(kscatbot+1,n+1)+D_E(kscatbot+1)*Hyinc(kscatbot,n+
1);
   Dy(kscatbot+1,n+1)=Dy(kscatbot+1,n+1)-
D_E(kscatbot+1)*Hxinc(kscatbot,n+1);
   Dx(kscattop+1,n+1)=Dx(kscattop+1,n+1)-
D_E(kscattop+1)*Hyinc(kscattop+1,n+1);

Dy(kscattop+1,n+1)=Dy(kscattop+1,n+1)+D_E(kscattop+1)*Hxinc(kscattop+1,
n+1);

Ex(ze,n+1)=epsinvxx(ze).*Dx(ze,n+1)+epsinvxy(ze).*Dy(ze,n+1)+epsinvxz(z
e).*Dz(ze,n+1);

Ey(ze,n+1)=epsinvxy(ze).*Dx(ze,n+1)+epsinvyy(ze).*Dy(ze,n+1)+epsinvyz(z
e).*Dz(ze,n+1);

Ez(ze,n+1)=epsinvxz(ze).*Dx(ze,n+1)+epsinvyz(ze).*Dy(ze,n+1)+epsinvzz(z
e).*Dz(ze,n+1);
   Exinc(ze,n+1)=C_E(ze).*Exinc(ze,n)-D_E(ze).*(Hyinc(ze,n+1)-Hyinc(ze-
1,n+1));
   Eyinc(ze,n+1)=C_E(ze).*Eyinc(ze,n)+D_E(ze).*(Hxinc(ze,n+1)-Hxinc(ze-
1,n+1));
   Exinc(kincstart+1,n+1)=-E0*sin(omega*(Tbreak+n)*dt)*(1-exp(-
((Tbreak+n)*dt-tdelay)^2/sigmadelay));
   Hx(:,n+2)=C_H.*Hx(:,n+1)+D_H.*(Ey(zh,n+1)-Ey(zh-1,n+1));
   Hy(:,n+2)=C_H.*Hy(:,n+1)-D_H.*(Ex(zh,n+1)-Ex(zh-1,n+1));
   Hx(kscatbot,n+2)=Hx(kscatbot,n+2)-
D_H(kscatbot)*Eyinc(kscatbot+1,n+1);

Hy(kscatbot,n+2)=Hy(kscatbot,n+2)+D_H(kscatbot)*Exinc(kscatbot+1,n+1);

Hx(kscattop+1,n+2)=Hx(kscattop+1,n+2)+D_H(kscattop+1)*Eyinc(kscattop+1,
n+1);
   Hy(kscattop+1,n+2)=Hy(kscattop+1,n+2)-
D_H(kscattop+1)*Exinc(kscattop+1,n+1);
   Hxinc(:,n+2)=C_H.*Hxinc(:,n+1)+D_H.*(Eyinc(zh,n+1)-Eyinc(zh-1,n+1));
   Hyinc(:,n+2)=C_H.*Hyinc(:,n+1)-D_H.*(Exinc(zh,n+1)-Exinc(zh-1,n+1));
   Hxinctilde(:,n+1)=(Hxinc(ze,n+1)+Hxinc(ze,n+2)+Hxinc(ze-
1,n+1)+Hxinc(ze-1,n+2))/4;
   Hyinctilde(:,n+1)=(Hyinc(ze,n+1)+Hyinc(ze,n+2)+Hyinc(ze-
1,n+1)+Hyinc(ze-1,n+2))/4;
   Hxtilde(:,n+1)=(Hx(ze,n+1)+Hx(ze,n+2)+Hx(ze-1,n+1)+Hx(ze-1,n+2))/4;
   Hytilde(:,n+1)=(Hy(ze,n+1)+Hy(ze,n+2)+Hy(ze-1,n+1)+Hy(ze-1,n+2))/4;
   Poyntingx(:,n+1)=-Ez(ze,n+1).*Hytilde(:,n+1);
   Poyntingy(:,n+1)=Ez(ze,n+1).*Hxtilde(:,n+1);
   Poyntingz(:,n+1)=Ex(ze,n+1).*Hytilde(:,n+1)-
Ey(ze,n+1).*Hxtilde(:,n+1);
```

```
PoyntingMag(:,n+1)=sqrt(Poyntingx(:,n+1).^2+Poyntingy(:,n+1).^2+Poyntin
gz(:,n+1).^2);
end
Intensity=PoyntingMag(:,1);
for n=2:2*timeperiod
    Intensity=Intensity+2*PoyntingMag(:,n);
end
Intensity=Intensity+PoyntingMag(:,2*timeperiod+1);
Intensity=Intensity*dt/E0^2/(4*pi/omega);

kPMLbot5=kPMLbot; kscatbot5=kscatbot; kairbot5=kairbot;
kglassbot5=kglassbot;
kglasstop5=kglasstop; kairtop5=kairtop; kscattop5=kscattop;
kPMLtop5=kPMLtop;
Ntot5=Ntot; dz5=dz; dt5=dt;
Ex5=Ex; Ey5=Ey; Ez5=Ez;
Hx5=Hx; Hy5=Hy;
Exinc5=Exinc; Eyinc5=Eyinc;
Hxinc5=Hxinc; Hyinc5=Hyinc;
Hxinctilde5=Hxinctilde; Hyinctilde5=Hyinctilde;
Hxtilde5=Hxtilde; Hytilde5=Hytilde;
Poyntingx5=Poyntingx; Poyntingy5=Poyntingy; Poyntingz5=Poyntingz;
PoynitngMag5=PoyntingMag; Intensity5=Intensity;
timeperiod5=timeperiod; Tfin5=Tfin; Tbreak5=Tbreak;
kincstart5=kincstart;
zTotE5=dz*((1:Ntot+1)-kglassbot-1);
zTotH5=dz*((1:Ntot)-1/2-kglassbot);
tic

toc
%% Finer grid
refinement=2;
NLCP=NLCP*refinement           %number of numerical cells in the LCP
dz=h/NLCP;
Nglass=Nglass*refinement;       %number of numerical cells in the glass
Nair=Nair*refinement;           %number of numerical cells in the air
between glass and scattering layer
Nscat=Nscat*refinement;         %number of numerical cells in the
scattering later
NPML=NPML*refinement;           %number of numerical cells in the
perfectly matched layer
cdtoverdz=1;
dt=dz*cdtoverdz;
Tfin=floor(EndTime/dt)+1;
timeperiod=round(2*pi/omega/dt);
sigmamax=2/dt; %10^17;        %in the PML, sigmaz=sigmamax((kPML-
1)/(NPML-1))^sigmam
sigmam=3;
kPMLbot=NPML;               % sigma defined but equal to zero at
k=kPMLbot
kscatbot=kPMLbot+Nscat;     % k=kscatbot is considered to be in the
total field; kscatbot-1/2 is in the scattered field
kairbot=kscatbot+Nair;      % k=kairbot is the bottom edge of the
bottom plate
```

```
kglassbot=kairbot+Nglass;   % k=kglassbot is the top edge of the bottom
plate/where the anchoring condition is applied
kglasstop=kglassbot+NLCP;   % k=kglasstop is the bottom edge of the top
plate/where the anchoring condition is applied
kairtop=kglasstop+Nglass;   % k=kairtop is the top edge of the top
plate
kscattop=kairtop+Nair;      % k=kscattop is in the total field;
kscattop+1/2 is not
kPMLtop=kscattop+Nscat;     % sigma defined but equal to zero at
k=kPMLtop
Ntot=kPMLtop+NPML;          % Ntot cells in total% ze=2:Ntot;
ze=2:Ntot;                   % k values for updating the gradients of H
in the D and E equations
zh=2:Ntot+1;
z6=linspace(0,h,NLCP+1);  %z values used for computing the orientation
kincstart=refinement*(kincstart5-kglassbot5)+kglassbot;

%% compute the permittivity
solinit=bvpinit(z6,@LEodeinit);
sol=bvp4c(@LEodefun,@LEbcfun,solinit);
u=deval(sol,z6);
psiLE=u(1,:);
thetaLE=u(3,:);

n1=cos(thetaLE).*cos(psiLE);
n2=cos(thetaLE).*sin(psiLE);
n3=sin(thetaLE);

epsxx=zeros(Ntot+1,1);
epsyy=zeros(Ntot+1,1);
epszz=zeros(Ntot+1,1);
epsxy=zeros(Ntot+1,1);
epsxz=zeros(Ntot+1,1);
epsyz=zeros(Ntot+1,1);

epsxx(1:kPMLbot+1)=1; epsyy(1:kPMLbot+1)=1; epszz(1:kPMLbot+1)=1;
epsxx(kPMLbot+2:kairbot)=epsilon_air;
epsyy(kPMLbot+2:kairbot)=epsilon_air;
epszz(kPMLbot+2:kairbot)=epsilon_air;
epsxx(kairbot+1:kglassbot)=epsilon_glass;
epsyy(kairbot+1:kglassbot)=epsilon_glass;
epszz(kairbot+1:kglassbot)=epsilon_glass;
epsxx(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n1.^2;
epsyy(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n2.^2;
epszz(kglassbot+1:kglasstop+1)=epsilon_perp+delta_epsilon*n3.^2;
epsxy(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n2;
epsxz(kglassbot+1:kglasstop+1)=delta_epsilon*n1.*n3;
epsyz(kglassbot+1:kglasstop+1)=delta_epsilon*n2.*n3;
epsxx(kglasstop+2:kairtop+1)=epsilon_glass;
epsyy(kglasstop+2:kairtop+1)=epsilon_glass;
epszz(kglasstop+2:kairtop+1)=epsilon_glass;
epsxx(kairtop+2:kPMLtop)=epsilon_air;
epsyy(kairtop+2:kPMLtop)=epsilon_air;
epszz(kairtop+2:kPMLtop)=epsilon_air;
```

```
epsxx(kPMLtop+1:Ntot)=1; epsyy(kPMLtop+1:Ntot)=1;
epszz(kPMLtop+1:Ntot)=1;

epsdet=epsxx.*(epsyy.*epszz-epsyz.^2)-epsxy.*(epsxy.*epszz-
epsxz.*epsyz)+epsxz.*(epsxy.*epsyz-epsyy.*epsxz);
epsinvxx=(epsyy.*epszz-epsyz.^2)./epsdet;
epsinvxy=(epsxz.*epsyz-epsxy.*epszz)./epsdet;
epsinvxz=(epsxy.*epsyz-epsyy.*epsxz)./epsdet;
epsinvyy=(epsxx.*epszz-epsxz.^2)./epsdet;
epsinvyz=(epsxy.*epsxz-epsxx.*epsyz)./epsdet;
epsinvzz=(epsxx.*epsyy-epsxy.^2)./epsdet;

%% Define coefficients
C_E=ones(Ntot+1,1);             % factor of the D^n term in the D
update equations
D_E=cdtoverdz*ones(Ntot+1,1);   % factor of the curl H term in the D
update equations
C_H=ones(Ntot,1);               % factor of the H^n term in the H
update equations
D_H=cdtoverdz*ones(Ntot,1);     % factor of the curl H term in the H
update equations
for k=1:kPMLbot                 % fill in the effect of sigma in the
PML
   sigmaz=sigmamax*((kPMLbot-k+1)/kPMLbot)^sigmam;
   C_E(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_E(Ntot+2-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_E(Ntot+2-k)=2*cdtoverdz/(2+dt*sigmaz);
   sigmaz=sigmamax*((kPMLbot-k+0.5)/kPMLbot)^sigmam;
   C_H(k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(k)=2*cdtoverdz/(2+dt*sigmaz);
   C_H(Ntot+1-k)=(2-dt*sigmaz)/(2+dt*sigmaz);
   D_H(Ntot+1-k)=2*cdtoverdz/(2+dt*sigmaz);
end

%%  Define electromagnetic field variables
Dx=zeros(Ntot+1,2*timeperiod+1);      % D and E are known on the space-
time grid
Dy=zeros(Ntot+1,2*timeperiod+1);      % D and E fields are zero
initially (n=0), but this is not stored.
Dz=zeros(Ntot+1,2*timeperiod+1);      % Dx and Ex are zero at k=1 and
k=Ntot+1 but they are stored so that
Ex=zeros(Ntot+1,2*timeperiod+1);      % Dx(k,n) is Dx at t=n*dt and
z=(k-kglassbot-1)*dz,
Ey=zeros(Ntot+1,2*timeperiod+1);      % or Ex_k^n is stored as
Ex(k+1,n)
Ez=zeros(Ntot+1,2*timeperiod+1);      % H is known on the half-grid.
Hx=zeros(Ntot,2*timeperiod+2);        % Hx(k,n) is Hx at t=(n-1/2)*dt,
z=(k-kglassbot+1/2)*dz
Hy=zeros(Ntot,2*timeperiod+2);        % Hx_(k+1/2)^(n+1/2) is stored as
Hx(k+1,n+1)
Exinc=zeros(Ntot+1,2*timeperiod+1);   % Hz is zero and not stored.  Ez
can be nonzero, dependending on epsilon
Eyinc=zeros(Ntot+1,2*timeperiod+1);   %
```

```
Hxinc=zeros(Ntot,2*timeperiod+2);     % Exinc, Eyinc, Hxinc, and Hyinc
are the incident field
Hyinc=zeros(Ntot,2*timeperiod+2);
Hxinctilde=zeros(Ntot-1,2*timeperiod+1);   % Hxinctilde interpolates
Hxinc onto the grid
Hyinctilde=zeros(Ntot-1,2*timeperiod+1);
Hxtilde=zeros(Ntot-1,2*timeperiod+1);      % Hxtilde interpolates Hx
onto the grid
Hytilde=zeros(Ntot-1,2*timeperiod+1);
Poyntingx=zeros(Ntot-1,2*timeperiod+1);  % The Poynting vector is known
on the space-time grid, and so H must be
Poyntingy=zeros(Ntot-1,2*timeperiod+1);  % interpolated in time and in
space.  Thus it is not defined at k=1 or k=Ntot+1
Poyntingz=zeros(Ntot-1,2*timeperiod+1);  % or at t=Tfin
PoyntingMag=zeros(Ntot-1,2*timeperiod+1);
Tbreak=Tfin-2*timeperiod-1;

% The starting main loop, not storing every step
Exinc(kincstart+1,1)=-E0*sin(omega*dt)*(1-exp(-(dt-
tdelay)^2/sigmadelay));
Hx(kscatbot,1)=-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,1)=D_H(kscatbot)*Exinc(kscatbot+1,1);
Hxinc(:,1)=D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,1)=-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
for n=2:Tbreak-1
   Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
   Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
   Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
   Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
   Dx(kscattop+1,1)=Dx(kscattop+1,1)-
D_E(kscattop+1)*Hyinc(kscattop+1,1);

Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);

Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);

Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
   Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-
1,1));
   Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-
1,1));
   Exinc(kincstart+1,1)=-E0*sin(omega*n*dt)*(1-exp(-(n*dt-
tdelay)^2/sigmadelay));
   Hx(:,1)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
   Hy(:,1)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
   Hx(kscatbot,1)=Hx(kscatbot,1)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
   Hy(kscatbot,1)=Hy(kscatbot,1)+D_H(kscatbot)*Exinc(kscatbot+1,1);

Hx(kscattop+1,1)=Hx(kscattop+1,1)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
   Hy(kscattop+1,1)=Hy(kscattop+1,1)-
D_H(kscattop+1)*Exinc(kscattop+1,1);
   Hxinc(:,1)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
   Hyinc(:,1)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
```

```
end
%n=Tbreak;
Dx(ze,1)=C_E(ze).*Dx(ze,1)-D_E(ze).*(Hy(ze,1)-Hy(ze-1,1));
Dy(ze,1)=C_E(ze).*Dy(ze,1)+D_E(ze).*(Hx(ze,1)-Hx(ze-1,1));
Dx(kscatbot+1,1)=Dx(kscatbot+1,1)+D_E(kscatbot+1)*Hyinc(kscatbot,1);
Dy(kscatbot+1,1)=Dy(kscatbot+1,1)-D_E(kscatbot+1)*Hxinc(kscatbot,1);
Dx(kscattop+1,1)=Dx(kscattop+1,1)-D_E(kscattop+1)*Hyinc(kscattop+1,1);
Dy(kscattop+1,1)=Dy(kscattop+1,1)+D_E(kscattop+1)*Hxinc(kscattop+1,1);
Ex(ze,1)=epsinvxx(ze).*Dx(ze,1)+epsinvxy(ze).*Dy(ze,1)+epsinvxz(ze).*Dz
(ze,1);
Ey(ze,1)=epsinvxy(ze).*Dx(ze,1)+epsinvyy(ze).*Dy(ze,1)+epsinvyz(ze).*Dz
(ze,1);
Ez(ze,1)=epsinvxz(ze).*Dx(ze,1)+epsinvyz(ze).*Dy(ze,1)+epsinvzz(ze).*Dz
(ze,1);
Exinc(ze,1)=C_E(ze).*Exinc(ze,1)-D_E(ze).*(Hyinc(ze,1)-Hyinc(ze-1,1));
Eyinc(ze,1)=C_E(ze).*Eyinc(ze,1)+D_E(ze).*(Hxinc(ze,1)-Hxinc(ze-1,1));
Exinc(kincstart+1,1)=-E0*sin(omega*Tbreak*dt)*(1-exp(-(Tbreak*dt-
tdelay)^2/sigmadelay));
Hx(:,2)=C_H.*Hx(:,1)+D_H.*(Ey(zh,1)-Ey(zh-1,1));
Hy(:,2)=C_H.*Hy(:,1)-D_H.*(Ex(zh,1)-Ex(zh-1,1));
Hx(kscatbot,2)=Hx(kscatbot,2)-D_H(kscatbot)*Eyinc(kscatbot+1,1);
Hy(kscatbot,2)=Hy(kscatbot,2)+D_H(kscatbot)*Exinc(kscatbot+1,1);
Hx(kscattop+1,2)=Hx(kscattop+1,2)+D_H(kscattop+1)*Eyinc(kscattop+1,1);
Hy(kscattop+1,2)=Hy(kscattop+1,2)-D_H(kscattop+1)*Exinc(kscattop+1,1);
Hxinc(:,2)=C_H.*Hxinc(:,1)+D_H.*(Eyinc(zh,1)-Eyinc(zh-1,1));
Hyinc(:,2)=C_H.*Hyinc(:,1)-D_H.*(Exinc(zh,1)-Exinc(zh-1,1));
Hxinctilde(:,1)=(Hxinc(ze,1)+Hxinc(ze,2)+Hxinc(ze-1,1)+Hxinc(ze-
1,2))/4;
Hyinctilde(:,1)=(Hyinc(ze,1)+Hyinc(ze,2)+Hyinc(ze-1,1)+Hyinc(ze-
1,2))/4;
Hxtilde(:,1)=(Hx(ze,1)+Hx(ze,2)+Hx(ze-1,1)+Hx(ze-1,2))/4;
Hytilde(:,1)=(Hy(ze,1)+Hy(ze,2)+Hy(ze-1,1)+Hy(ze-1,2))/4;
Poyntingx(:,1)=-Ez(ze,1).*Hytilde(:,1);
Poyntingy(:,1)=Ez(ze,1).*Hxtilde(:,1);
Poyntingz(:,1)=Ex(ze,1).*Hytilde(:,1)-Ey(ze,1).*Hxtilde(:,1);
PoyntingMag(:,1)=sqrt(Poyntingx(:,1).^2+Poyntingy(:,1).^2+Poyntingz(:,1
).^2);
%for n=Tbreak:Tfin
%makes Dx(:,1) actually at n=Tfin-1-2*timeperiod
for n=1:2*timeperiod
   Dx(ze,n+1)=C_E(ze).*Dx(ze,n)-D_E(ze).*(Hy(ze,n+1)-Hy(ze-1,n+1));
   Dy(ze,n+1)=C_E(ze).*Dy(ze,n)+D_E(ze).*(Hx(ze,n+1)-Hx(ze-1,n+1));

Dx(kscatbot+1,n+1)=Dx(kscatbot+1,n+1)+D_E(kscatbot+1)*Hyinc(kscatbot,n+
1);
   Dy(kscatbot+1,n+1)=Dy(kscatbot+1,n+1)-
D_E(kscatbot+1)*Hxinc(kscatbot,n+1);
   Dx(kscattop+1,n+1)=Dx(kscattop+1,n+1)-
D_E(kscattop+1)*Hyinc(kscattop+1,n+1);

Dy(kscattop+1,n+1)=Dy(kscattop+1,n+1)+D_E(kscattop+1)*Hxinc(kscattop+1,
n+1);

Ex(ze,n+1)=epsinvxx(ze).*Dx(ze,n+1)+epsinvxy(ze).*Dy(ze,n+1)+epsinvxz(z
e).*Dz(ze,n+1);
```

```
Ey(ze,n+1)=epsinvxy(ze).*Dx(ze,n+1)+epsinvyy(ze).*Dy(ze,n+1)+epsinvyz(z
e).*Dz(ze,n+1);

Ez(ze,n+1)=epsinvxz(ze).*Dx(ze,n+1)+epsinvyz(ze).*Dy(ze,n+1)+epsinvzz(z
e).*Dz(ze,n+1);
   Exinc(ze,n+1)=C_E(ze).*Exinc(ze,n)-D_E(ze).*(Hyinc(ze,n+1)-Hyinc(ze-
1,n+1));
   Eyinc(ze,n+1)=C_E(ze).*Eyinc(ze,n)+D_E(ze).*(Hxinc(ze,n+1)-Hxinc(ze-
1,n+1));
   Exinc(kincstart+1,n+1)=-E0*sin(omega*(Tbreak+n)*dt)*(1-exp(-
((Tbreak+n)*dt-tdelay)^2/sigmadelay));
   Hx(:,n+2)=C_H.*Hx(:,n+1)+D_H.*(Ey(zh,n+1)-Ey(zh-1,n+1));
   Hy(:,n+2)=C_H.*Hy(:,n+1)-D_H.*(Ex(zh,n+1)-Ex(zh-1,n+1));
   Hx(kscatbot,n+2)=Hx(kscatbot,n+2)-
D_H(kscatbot)*Eyinc(kscatbot+1,n+1);

Hy(kscatbot,n+2)=Hy(kscatbot,n+2)+D_H(kscatbot)*Exinc(kscatbot+1,n+1);

Hx(kscattop+1,n+2)=Hx(kscattop+1,n+2)+D_H(kscattop+1)*Eyinc(kscattop+1,
n+1);
   Hy(kscattop+1,n+2)=Hy(kscattop+1,n+2)-
D_H(kscattop+1)*Exinc(kscattop+1,n+1);
   Hxinc(:,n+2)=C_H.*Hxinc(:,n+1)+D_H.*(Eyinc(zh,n+1)-Eyinc(zh-1,n+1));
   Hyinc(:,n+2)=C_H.*Hyinc(:,n+1)-D_H.*(Exinc(zh,n+1)-Exinc(zh-1,n+1));
   Hxinctilde(:,n+1)=(Hxinc(ze,n+1)+Hxinc(ze,n+2)+Hxinc(ze-
1,n+1)+Hxinc(ze-1,n+2))/4;
   Hyinctilde(:,n+1)=(Hyinc(ze,n+1)+Hyinc(ze,n+2)+Hyinc(ze-
1,n+1)+Hyinc(ze-1,n+2))/4;
   Hxtilde(:,n+1)=(Hx(ze,n+1)+Hx(ze,n+2)+Hx(ze-1,n+1)+Hx(ze-1,n+2))/4;
   Hytilde(:,n+1)=(Hy(ze,n+1)+Hy(ze,n+2)+Hy(ze-1,n+1)+Hy(ze-1,n+2))/4;
   Poyntingx(:,n+1)=-Ez(ze,n+1).*Hytilde(:,n+1);
   Poyntingy(:,n+1)=Ez(ze,n+1).*Hxtilde(:,n+1);
   Poyntingz(:,n+1)=Ex(ze,n+1).*Hytilde(:,n+1)-
Ey(ze,n+1).*Hxtilde(:,n+1);

PoyntingMag(:,n+1)=sqrt(Poyntingx(:,n+1).^2+Poyntingy(:,n+1).^2+Poyntin
gz(:,n+1).^2);
end
Intensity=PoyntingMag(:,1);
for n=2:2*timeperiod
    Intensity=Intensity+2*PoyntingMag(:,n);
end
Intensity=Intensity+PoyntingMag(:,2*timeperiod+1);
Intensity=Intensity*dt/E0^2/(4*pi/omega);

kPMLbot6=kPMLbot; kscatbot6=kscatbot; kairbot6=kairbot;
kglassbot6=kglassbot;
kglasstop6=kglasstop; kairtop6=kairtop; kscattop6=kscattop;
kPMLtop6=kPMLtop;
Ntot6=Ntot; dz6=dz; dt6=dt;
Ex6=Ex; Ey6=Ey; Ez6=Ez;
Hx6=Hx; Hy6=Hy;
Exinc6=Exinc; Eyinc6=Eyinc;
Hxinc6=Hxinc; Hyinc6=Hyinc;
```

```
Hxinctilde6=Hxinctilde; Hyinctilde6=Hyinctilde;
Hxtilde6=Hxtilde; Hytilde6=Hytilde;
Poyntingx6=Poyntingx; Poyntingy6=Poyntingy; Poyntingz6=Poyntingz;
PoynitngMag6=PoyntingMag; Intensity6=Intensity;
timeperiod6=timeperiod; Tfin6=Tfin; Tbreak6=Tbreak;
kincstart6=kincstart;
zTotE6=dz*((1:Ntot+1)-kglassbot-1);
zTotH6=dz*((1:Ntot)-1/2-kglassbot);
toc

%% We now will compute the true analytical solution.

% System of Equations
ExI=E0;
Tplot6=2*timeperiod6+1;
Exinctrue=E0*sin(wavenumber*(zTotE6-zTotE6(kincstart6+1))-
omega*(Tbreak6+Tplot6-1)*dt6)*(1-exp(-((Tbreak6+Tplot6-1)*dt6-
tdelay)^2/sigmadelay));
zstart=dz*(kincstart6-kglassbot);
ExT=4*ExI/exp(1i*wavenumber*h)*((nperp*sin(psibot)^2)/((1+nperp)^2*exp(
-1i*kperp*h)-(1-
nperp)^2*exp(1i*kperp*h))+(ntheta*(cos(psibot))^2)/((1+ntheta)^2*exp(-
1i*ktheta*h)-(1-ntheta)^2*exp(1i*ktheta*h)));
EyT=4*ExI/exp(1i*wavenumber*h)*((-
nperp*sin(psibot)*cos(psibot))/((1+nperp)^2*exp(-1i*kperp*h)-(1-
nperp)^2*exp(1i*kperp*h))+(ntheta*sin(psibot)*cos(psibot)/((1+ntheta)^2
*exp(-1i*ktheta*h)-(1-ntheta)^2*exp(1i*ktheta*h))));
ExR=ExI*((1-ntheta^2)*(cos(psibot))^2*(1-exp(-i*2*ktheta*h))/((1-
ntheta)^2-(1+ntheta)^2*exp(-2*i*ktheta*h))-(1-
nperp^2)*(sin(psibot))^2*(1-exp(2*i*kperp*h))/((1-
nperp)^2*exp(i*2*kperp*h-(1+nperp)^2)));
EyR=ExI*cos(psibot)*sin(psibot)*((1-ntheta^2)*(1-exp(-
i*2*ktheta*h))/((1-ntheta)^2-(1+ntheta)^2*exp(-2*i*ktheta*h))+(1-
nperp^2)*(1-exp(2*i*kperp*h))/((1-nperp)^2*exp(i*2*kperp*h)-
(1+nperp)^2));
ELplusperp=2*ExI*sin(psibot)*(1+nperp)/((1-nperp)^2*exp(1i*2*kperp*h)-
(1+nperp)^2);
ELminusperp=-2*ExI*sin(psibot)*(1-nperp)/((1-nperp)^2-(1+nperp)^2*exp(-
1i*2*kperp*h));
ELplustheta=-2*ExI*cos(psibot)*(1+ntheta)/((1-
ntheta)^2*exp(1i*2*ktheta*h)-(1+ntheta)^2);
ELminustheta=2*ExI*cos(psibot)*(1-ntheta)/((1-ntheta)^2-
(1+ntheta)^2*exp(-1i*2*ktheta*h));
Exyexp=exp(-1i*(omega*(Tbreak6+Tplot6-1)*dt6+wavenumber*zstart+pi/2));
Hxyexp=exp(-1i*(omega*(Tbreak6+Tplot6-1)*dt6+wavenumber*zstart+pi/2));

% Calculate E Components
ExTtrue=real(Exyexp*(ExT)*exp(1i*wavenumber*zTotE6(kglasstop+1:kscattop
+1)));
EyTtrue=real(Exyexp*(EyT)*exp(1i*wavenumber*zTotE6(kglasstop+1:kscattop
+1)));
ExRtrue=real(Exyexp*ExR*exp(1i*(-
wavenumber*(zTotE6(kscatbot+1:kglassbot)))));
EyRtrue=real(Exyexp*EyR*exp(1i*(-
wavenumber*(zTotE6(kscatbot+1:kglassbot)))));
```

```
ExLtrue=real((-
sin(psibot)*(ELplusperp*exp(1i*kperp*zTotE6(kglassbot+1:kglasstop))+ELm
inusperp*exp(-1i*kperp*zTotE6(kglassbot+1:kglasstop)))...
+cos(psibot)*(ELplustheta*exp(1i*ktheta*zTotE6(kglassbot+1:kglasstop))+
ELminustheta*exp(-1i*ktheta*zTotE6(kglassbot+1:kglasstop))))*Exyexp);
EyLtrue=real((cos(psibot)*(ELplusperp*exp(1i*kperp*zTotE6(kglassbot+1:k
glasstop))+ELminusperp*exp(-1i*kperp*zTotE6(kglassbot+1:kglasstop)))...
+sin(psibot)*(ELplustheta*exp(i*ktheta*zTotE6(kglassbot+1:kglasstop))+E
Lminustheta*exp(-1i*ktheta*zTotE6(kglassbot+1:kglasstop))))*Exyexp);

% Calculate H Components
HxTtrue=real(Hxyexp*(-
EyT)*exp(1i*wavenumber*zTotE6(kglasstop+1:kscattop+1)));
HyTtrue=real(Hxyexp*(ExT)*exp(i*wavenumber*zTotE6(kglasstop+1:kscattop+
1)));
HxRtrue=real(Hxyexp*EyR*exp(-
1i*wavenumber*(zTotE6(kscatbot+1:kglassbot))));
HyRtrue=real(Hxyexp*(-ExR)*exp(-
1i*wavenumber*(zTotE6(kscatbot+1:kglassbot))));
HxLtrue=real((-
nperp*cos(psibot))*Hxyexp*(ELplusperp*exp(i*kperp*zTotE6(kglassbot+1:kg
lasstop))-ELminusperp*exp(-1i*kperp*zTotE6(kglassbot+1:kglasstop)))...
+(-
ntheta*sin(psibot))*Hxyexp*(ELplustheta*exp(1i*ktheta*zTotE6(kglassbot+
1:kglasstop))-ELminustheta*exp(-
1i*ktheta*zTotE6(kglassbot+1:kglasstop))));
HyLtrue=real((-
nperp*sin(psibot))*Hxyexp*(ELplusperp*exp(1i*kperp*zTotE6(kglassbot+1:k
glasstop))-ELminusperp*exp(-1i*kperp*zTotE6(kglassbot+1:kglasstop)))...
+(ntheta*cos(psibot))*Hxyexp*(ELplustheta*exp(i*ktheta*zTotE6(kglassbot
+1:kglasstop))-ELminustheta*exp(-
i*ktheta*zTotE6(kglassbot+1:kglasstop))));

% Calculate E TRUE using E components
ExTRUE=[Exinctrue(kscatbot+1:kglassbot)+ExRtrue ExLtrue ExTtrue];
Eyinctrue=zeros(1,length(Exinctrue));
EyTRUE=[Eyinctrue(kscatbot+1:kglassbot)+EyRtrue EyLtrue EyTtrue];

% Calculate H TRUE using H components
Hyinctildetrue=E0*sin(wavenumber*((zTotH6(1:Ntot6-
1)+zTotH6(2:Ntot6))/2-zTotE6(kincstart6+1))-omega*(Tbreak6+Tplot6-
1)*dt6)*(1-exp(-((Tbreak6+Tplot6-1)*dt6-tdelay)^2/sigmadelay));
HyTRUE=[Hyinctildetrue(kscatbot+1:kglassbot)+HyRtrue HyLtrue HyTtrue];
Hxinctildetrue=zeros(1,length(Hyinctildetrue));
HxTRUE=[Hxinctildetrue(kscatbot+1:kglassbot)+HxRtrue HxLtrue HxTtrue];

% Calculate Intensity TRUE
IntensityTTRUE=zeros(size(ExTRUE));
IntensityTTRUE(kglasstop-kscatbot+1:kscattop-
kscatbot)=4*nperp^2*sin(psibot)^2/(4*nperp^2+(1-
nperp^2)^2*sin(kperp*h)^2)+4*ntheta^2*cos(psibot)^2/(4*ntheta^2+(1-
ntheta^2)^2*sin(ktheta*h)^2);

%% Plots
```

```
% We want to plot the last time step as a function of z for each grid.
Tplot1=2*timeperiod1+1;
Tplot2=2*timeperiod2+1;
Tplot3=2*timeperiod3+1;
Tplot4=2*timeperiod4+1;
Tplot5=2*timeperiod5+1;
Tplot6=2*timeperiod6+1;

% Figure 1 compares the Ex and Hy values of the incident field.  The
green
% vertical lines mark the top and bottom of the LCP layer.  The black
% vertical lines mark the boundaries with the scattered field.
figure(1)
subplot(2,1,1)
plot(zTotE1,Exinc1(:,Tplot1),'b')
hold on
  plot(zTotE2,Exinc2(:,Tplot2),'g')
  plot(zTotE3,Exinc3(:,Tplot3),'r')
  plot(zTotE4,Exinc4(:,Tplot4),'c')
  plot(zTotE5,Exinc5(:,Tplot5),'k')
  plot(zTotE6,Exinc6(:,Tplot6),'b')
  plot(zTotE6,Exinctrue,'m:')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kscatbot-kglassbot kscatbot-kglassbot],[-1.5 1.5],'k')
  plot(dz*[kscattop-kglassbot kscattop-kglassbot],[-1.5 1.5],'k')
hold off
xlim([zTotE1(1) zTotE1(Ntot1+1)])
title('Ex inc')

subplot(2,1,2)
plot((zTotH1(1:Ntot1-1)+zTotH1(2:Ntot1))/2,Hyinctilde1(:,Tplot1),'b')
hold on
  plot((zTotH2(1:Ntot2-1)+zTotH2(2:Ntot2))/2,Hyinctilde2(:,Tplot2),'g')
  plot((zTotH3(1:Ntot3-1)+zTotH3(2:Ntot3))/2,Hyinctilde3(:,Tplot3),'r')
  plot((zTotH4(1:Ntot4-1)+zTotH4(2:Ntot4))/2,Hyinctilde4(:,Tplot4),'c')
  plot((zTotH5(1:Ntot5-1)+zTotH5(2:Ntot5))/2,Hyinctilde5(:,Tplot5),'k')
  plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,Hyinctilde6(:,Tplot6),'b')
  plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,Hyinctildetrue,'m')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kscatbot-kglassbot kscatbot-kglassbot],[-1.5 1.5],'k')
  plot(dz*[kscattop-kglassbot kscattop-kglassbot],[-1.5 1.5],'k')
hold off
xlabel('z')
xlim([zTotH1(1) zTotH1(Ntot1-1)])
title('Hy inc (averaged)')

% Figure 2 shows the point-wise error compared to the true solution for
% each grid
figure(2)
subplot(2,1,1)
semilogy(zTotE1,abs(Exinctrue(1:32:Ntot6+1)'-Exinc1(:,Tplot1)),'b')
hold on
    semilogy(zTotE2,abs(Exinctrue(1:16:Ntot6+1)'-Exinc2(:,Tplot2)),'g')
```

```
    semilogy(zTotE3,abs(Exinctrue(1:8:Ntot6+1)'-Exinc3(:,Tplot3)),'r')
    semilogy(zTotE4,abs(Exinctrue(1:4:Ntot6+1)'-Exinc4(:,Tplot4)),'c')
    semilogy(zTotE5,abs(Exinctrue(1:2:Ntot6+1)'-Exinc5(:,Tplot5)),'k')
    semilogy(zTotE6,abs(Exinctrue'-Exinc6(:,Tplot6)),'b')
hold off
xlim([zTotE1(kscatbot1+1) zTotE1(kscattop1+1)])
title('Ex inc error to true solution')
ylim([10^-10 10^0])

subplot(2,1,2)
semilogy(zTotE1(2:Ntot1),abs(E0*sin(wavenumber*(zTotE1(2:Ntot1)-
zTotE1(kincstart1+1))'-omega*(Tbreak1+Tplot1-1)*dt1)*(1-exp(-
((Tbreak1+Tplot1-1)*dt1-tdelay)^2/sigmadelay))-
Hyinctilde1(:,Tplot1)),'b')
hold on
  semilogy(zTotE2(2:Ntot2),abs(E0*sin(wavenumber*(zTotE2(2:Ntot2)-
zTotE2(kincstart2+1))'-omega*(Tbreak2+Tplot2-1)*dt2)*(1-exp(-
((Tbreak2+Tplot2-1)*dt2-tdelay)^2/sigmadelay))-
Hyinctilde2(:,Tplot2)),'g')
  semilogy(zTotE3(2:Ntot3),abs(E0*sin(wavenumber*(zTotE3(2:Ntot3)-
zTotE3(kincstart3+1))'-omega*(Tbreak3+Tplot3-1)*dt3)*(1-exp(-
((Tbreak3+Tplot3-1)*dt3-tdelay)^2/sigmadelay))-
Hyinctilde3(:,Tplot3)),'r')
  semilogy(zTotE4(2:Ntot4),abs(E0*sin(wavenumber*(zTotE4(2:Ntot4)-
zTotE4(kincstart4+1))'-omega*(Tbreak4+Tplot4-1)*dt4)*(1-exp(-
((Tbreak4+Tplot4-1)*dt4-tdelay)^2/sigmadelay))-
Hyinctilde4(:,Tplot4)),'c')
  semilogy(zTotE5(2:Ntot5),abs(E0*sin(wavenumber*(zTotE5(2:Ntot5)-
zTotE5(kincstart5+1))'-omega*(Tbreak5+Tplot5-1)*dt5)*(1-exp(-
((Tbreak5+Tplot5-1)*dt5-tdelay)^2/sigmadelay))-
Hyinctilde5(:,Tplot5)),'k')
  semilogy(zTotE6(2:Ntot6),abs(E0*sin(wavenumber*(zTotE6(2:Ntot6)-
zTotE6(kincstart6+1))'-omega*(Tbreak6+Tplot6-1)*dt6)*(1-exp(-
((Tbreak6+Tplot6-1)*dt6-tdelay)^2/sigmadelay))-
Hyinctilde6(:,Tplot6)),'b')
hold off
xlim([zTotE1(kscatbot1+1) zTotE1(kscattop1+1)])
title('Hy inc error to true solution')
ylim([10^-10 10^0])
xlabel('z')

% Figure 3(not used) shows the the grids compared to the finest grid
solution

% Figure 4 shows the solutions for the full system on the different
grids
figure(4)
subplot(4,1,1)
  plot(zTotE1,Ex1(:,Tplot1),'b')
hold on
  plot(zTotE2,Ex2(:,Tplot2),'g')
  plot(zTotE3,Ex3(:,Tplot3),'r')
  plot(zTotE4,Ex4(:,Tplot4),'c')
  plot(zTotE5,Ex5(:,Tplot5),'k')
  plot(zTotE6,Ex6(:,Tplot6),'b')
```

```
  plot(zTotE6(kscatbot+1:kscattop+1),ExTRUE,'m:')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kscatbot-kglassbot kscatbot-kglassbot],[-1.5 1.5],'k')
  plot(dz*[kscattop-kglassbot kscattop-kglassbot],[-1.5 1.5],'k')
hold off
xlim([zTotE1(1) zTotE1(Ntot1+1)])
ylim([-1.5 1.5])
title('Ex')

subplot(4,1,2)
  plot(zTotE1,Ey1(:,Tplot1),'b')
hold on
  plot(zTotE2,Ey2(:,Tplot2),'g')
  plot(zTotE3,Ey3(:,Tplot3),'r')
  plot(zTotE4,Ey4(:,Tplot4),'c')
  plot(zTotE5,Ey5(:,Tplot5),'k')
  plot(zTotE6,Ey6(:,Tplot6),'b')
  plot(zTotE6(kscatbot+1:kscattop+1),EyTRUE,'m:')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kscatbot-kglassbot kscatbot-kglassbot],[-1.5 1.5],'k')
  plot(dz*[kscattop-kglassbot kscattop-kglassbot],[-1.5 1.5],'k')
hold off
xlim([zTotE1(1) zTotE1(Ntot1+1)])
ylim([-1.5 1.5])
title('Ey')

subplot(4,1,3) % H components plotted on the x - grid
  plot((zTotH1(1:Ntot1-1)+zTotH1(2:Ntot1))/2,Hxtilde1(1:Ntot1-
1,Tplot1),'b')
hold on
  plot((zTotH2(1:Ntot2-1)+zTotH2(2:Ntot2))/2,Hxtilde2(1:Ntot2-
1,Tplot2),'g')
  plot((zTotH3(1:Ntot3-1)+zTotH3(2:Ntot3))/2,Hxtilde3(1:Ntot3-
1,Tplot3),'r')
  plot((zTotH4(1:Ntot4-1)+zTotH4(2:Ntot4))/2,Hxtilde4(1:Ntot4-
1,Tplot4),'c')
  plot((zTotH5(1:Ntot5-1)+zTotH5(2:Ntot5))/2,Hxtilde5(1:Ntot5-
1,Tplot5),'k')
  plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,Hxtilde6(1:Ntot6-
1,Tplot6),'b')
  plot(zTotE6(kscatbot+1:kscattop+1),HxTRUE,'m:')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kscatbot-kglassbot kscatbot-kglassbot],[-1.5 1.5],'k')
  plot(dz*[kscattop-kglassbot kscattop-kglassbot],[-1.5 1.5],'k')
hold off
ylim([-1.5 1.5])
xlim([zTotH1(1) zTotH1(Ntot1-1)])
title('Hx')

subplot(4,1,4)
  plot((zTotH1(1:Ntot1-1)+zTotH1(2:Ntot1))/2,Hytilde1(1:Ntot1-
1,Tplot1),'b')
```

```
hold on
  plot((zTotH2(1:Ntot2-1)+zTotH2(2:Ntot2))/2,Hytilde2(1:Ntot2-
1,Tplot2),'g')
  plot((zTotH3(1:Ntot3-1)+zTotH3(2:Ntot3))/2,Hytilde3(1:Ntot3-
1,Tplot3),'r')
  plot((zTotH4(1:Ntot4-1)+zTotH4(2:Ntot4))/2,Hytilde4(1:Ntot4-
1,Tplot4),'c')
  plot((zTotH5(1:Ntot5-1)+zTotH5(2:Ntot5))/2,Hytilde5(1:Ntot5-
1,Tplot5),'k')
  plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,Hytilde6(1:Ntot6-
1,Tplot6),'b')
  plot(zTotE6(kscatbot+1:kscattop+1),HyTRUE,'m:')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kscatbot-kglassbot kscatbot-kglassbot],[-1.5 1.5],'k')
  plot(dz*[kscattop-kglassbot kscattop-kglassbot],[-1.5 1.5],'k')
hold off
xlabel('z')
title('Hy')
ylim([-1.5 1.5])
xlim([zTotH1(1) zTotH1(Ntot1-1)])

% Figure 5 shows the the grids compared to the finest grid solution
figure(5)
subplot(2,1,1)
semilogy(zTotE1(kscatbot1+1:kscattop1+1),abs(ExTRUE(1:32:(kscattop-
kscatbot+1))'-Ex1(kscatbot1+1:kscattop1+1,Tplot1)),'b')
hold on
    semilogy(zTotE2(kscatbot2+1:kscattop2+1),abs(ExTRUE(1:16:(kscattop-
kscatbot+1))'-Ex2(kscatbot2+1:kscattop2+1,Tplot2)),'g')
    semilogy(zTotE3(kscatbot3+1:kscattop3+1),abs(ExTRUE(1:8:(kscattop-
kscatbot+1))'-Ex3(kscatbot3+1:kscattop3+1,Tplot3)),'r')
    semilogy(zTotE4(kscatbot4+1:kscattop4+1),abs(ExTRUE(1:4:(kscattop-
kscatbot+1))'-Ex4(kscatbot4+1:kscattop4+1,Tplot4)),'c')
    semilogy(zTotE5(kscatbot5+1:kscattop5+1),abs(ExTRUE(1:2:(kscattop-
kscatbot+1))'-Ex5(kscatbot5+1:kscattop5+1,Tplot5)),'k')
    semilogy(zTotE6(kscatbot+1:kscattop+1),abs(ExTRUE'-
Ex6(kscatbot6+1:kscattop6+1,Tplot6)),'m:')
hold off
xlim([zTotE1(kscatbot1+1) zTotE1(kscattop1+1)])
title('Ex tilde error to Ex True')
ylim([10^-6 10^0])

subplot(2,1,2)
semilogy(zTotE1(kscatbot1+1:kscattop1+1),abs(HyTRUE(1:32:(kscattop-
kscatbot+1))'-Hytilde1(kscatbot1+1:kscattop1+1,Tplot1)),'b')
hold on
    semilogy(zTotE2(kscatbot2+1:kscattop2+1),abs(HyTRUE(1:16:(kscattop-
kscatbot+1))'-Hytilde2(kscatbot2+1:kscattop2+1,Tplot2)),'g')
    semilogy(zTotE3(kscatbot3+1:kscattop3+1),abs(HyTRUE(1:8:(kscattop-
kscatbot+1))'-Hytilde3(kscatbot3+1:kscattop3+1,Tplot3)),'r')
    semilogy(zTotE4(kscatbot4+1:kscattop4+1),abs(HyTRUE(1:4:(kscattop-
kscatbot+1))'-Hytilde4(kscatbot4+1:kscattop4+1,Tplot4)),'c')
    semilogy(zTotE5(kscatbot5+1:kscattop5+1),abs(HyTRUE(1:2:(kscattop-
kscatbot+1))'-Hytilde5(kscatbot5+1:kscattop5+1,Tplot5)),'k')
```

```
    semilogy(zTotE6(kscatbot+1:kscattop+1),abs(HyTRUE'-
Hytilde6(kscatbot+1:kscattop+1,Tplot6)),'m:')
hold off
xlim([zTotE1(kscatbot1+1) zTotE1(kscattop1+1)])
title('Hy tilde error to Hy True ')
ylim([10^-6 10^0])
xlabel('z')

% Figure 6 shows the components of Poynting vector and the intentsity
figure(6)
subplot(2,2,1)
  plot((zTotH1(1:Ntot1-1)+zTotH1(2:Ntot1))/2,Poyntingx1(:,Tplot1),'b')
hold on
  plot((zTotH2(1:Ntot2-1)+zTotH2(2:Ntot2))/2,Poyntingx2(:,Tplot2),'g')
  plot((zTotH3(1:Ntot3-1)+zTotH3(2:Ntot3))/2,Poyntingx3(:,Tplot3),'r')
  plot((zTotH4(1:Ntot4-1)+zTotH4(2:Ntot4))/2,Poyntingx4(:,Tplot4),'c')
  plot((zTotH5(1:Ntot5-1)+zTotH5(2:Ntot5))/2,Poyntingx5(:,Tplot5),'k')
  plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,Poyntingx6(:,Tplot6),'b')
  plot((zTotH6(1:Ntot6-
1)+zTotH6(2:Ntot6))/2,PoyntingxTRUE(:,Tplot6),'m:')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.25 1.25],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.25 1.25],'g')
hold off
title('Poyntingx ')
ylim([-1.25 1.25])
xlim([zTotE1(1) zTotE1(Ntot1-1)])

subplot(2,2,2)
plot((zTotH1(1:Ntot1-1)+zTotH1(2:Ntot1))/2,Poyntingy1(:,Tplot1),'b')
hold on
  plot((zTotH2(1:Ntot2-1)+zTotH2(2:Ntot2))/2,Poyntingy2(:,Tplot2),'g')
  plot((zTotH3(1:Ntot3-1)+zTotH3(2:Ntot3))/2,Poyntingy3(:,Tplot3),'r')
  plot((zTotH4(1:Ntot4-1)+zTotH4(2:Ntot4))/2,Poyntingy4(:,Tplot4),'c')
  plot((zTotH5(1:Ntot5-1)+zTotH5(2:Ntot5))/2,Poyntingy5(:,Tplot5),'k')
  plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,Poyntingy6(:,Tplot6),'b')
  plot((zTotH6(1:Ntot6-
1)+zTotH6(2:Ntot6))/2,PoyntingyTRUE(:,Tplot6),'m:')
  title('Poyntingy ')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.25 1.25],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.25 1.25],'g')
hold off
ylim([-1.25 1.25])
xlim([zTotE1(1) zTotE1(Ntot1-1)])

subplot(2,2,3)
plot((zTotH1(1:Ntot1-1)+zTotH1(2:Ntot1))/2,Poyntingz1(:,Tplot1),'b')
hold on
    plot((zTotH2(1:Ntot2-
1)+zTotH2(2:Ntot2))/2,Poyntingz2(:,Tplot2),'g')
    plot((zTotH3(1:Ntot3-
1)+zTotH3(2:Ntot3))/2,Poyntingz3(:,Tplot3),'r')
    plot((zTotH4(1:Ntot4-
1)+zTotH4(2:Ntot4))/2,Poyntingz4(:,Tplot4),'c')
    plot((zTotH5(1:Ntot5-
1)+zTotH5(2:Ntot5))/2,Poyntingz5(:,Tplot5),'k')
```

```
    plot((zTotH6(1:Ntot6-
1)+zTotH6(2:Ntot6))/2,Poyntingz6(:,Tplot6),'b')
    plot((zTotH6(1:Ntot6-
1)+zTotH6(2:Ntot6))/2,PoyntingzTRUE(:,Tplot6),'m:')
    title('Poyntingz ')
    plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
    plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
hold off
ylim([-0.3 1.25])
xlim([zTotH1(1) zTotH1(Ntot1)])

subplot(2,2,4)
plot((zTotH1(1:Ntot1-1)+zTotH1(2:Ntot1))/2,Intensity1,'b')
hold on
    plot((zTotH2(1:Ntot2-1)+zTotH2(2:Ntot2))/2,Intensity2,'g')
    plot((zTotH3(1:Ntot3-1)+zTotH3(2:Ntot3))/2,Intensity3,'r')
    plot((zTotH4(1:Ntot4-1)+zTotH4(2:Ntot4))/2,Intensity4,'c')
    plot((zTotH5(1:Ntot5-1)+zTotH5(2:Ntot5))/2,Intensity5,'k')
    plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,Intensity6,'b')
    plot((zTotH6(1:Ntot6-1)+zTotH6(2:Ntot6))/2,IntensityTRUE,'m:')
    title('Intensity')
  plot(dz*[kglassbot-kglassbot kglassbot-kglassbot],[-1.5 1.5],'g')
  plot(dz*[kglasstop-kglassbot kglasstop-kglassbot],[-1.5 1.5],'g')
hold off
ylim([0.38 0.55])
xlim([zTotE1(kscatbot1+1) zTotE1(kscattop1+1)])

% Figure 7 shows the error of the Poynting vector components compared
to
% the finest solution.  The x- and y- components are identically zero
for
% all grids.
figure(7)
subplot(2,2,1)
semilogy((zTotH1(1:Ntot1-
1)+zTotH1(2:Ntot1))/2,abs(Poyntingx6(32:32:Ntot6-2,Tplot6)-
Poyntingx1(:,Tplot1)),'b')
hold on
    semilogy((zTotH2(1:Ntot2-
1)+zTotH2(2:Ntot2))/2,abs(Poyntingx6(16:16:Ntot6-2,Tplot6)-
Poyntingx2(:,Tplot2)),'g')
    semilogy((zTotH3(1:Ntot3-
1)+zTotH3(2:Ntot3))/2,abs(Poyntingx6(8:8:Ntot6-2,Tplot6)-
Poyntingx3(:,Tplot3)),'r')
    semilogy((zTotH4(1:Ntot4-
1)+zTotH4(2:Ntot4))/2,abs(Poyntingx6(4:4:Ntot6-2,Tplot6)-
Poyntingx4(:,Tplot4)),'c')
    semilogy((zTotH5(1:Ntot5-
1)+zTotH5(2:Ntot5))/2,abs(Poyntingx6(2:2:Ntot6-2,Tplot6)-
Poyntingx5(:,Tplot5)),'k')
hold off
xlim([zTotE1(kairbot1+1) zTotE1(kairtop1+1)])
ylim([10^-5 1])
title('|Poyntingx (fine) - Poyntingx (course)|')
subplot(2,2,2)
```

```
semilogy((zTotH1(1:Ntot1-
1)+zTotH1(2:Ntot1))/2,abs(Poyntingy6(32:32:Ntot6-2,Tplot6)-
Poyntingx1(:,Tplot1)),'b')
hold on
    semilogy((zTotH2(1:Ntot2-
1)+zTotH2(2:Ntot2))/2,abs(Poyntingy6(16:16:Ntot6-2,Tplot6)-
Poyntingy2(:,Tplot2)),'g')
    semilogy((zTotH3(1:Ntot3-
1)+zTotH3(2:Ntot3))/2,abs(Poyntingy6(8:8:Ntot6-2,Tplot6)-
Poyntingy3(:,Tplot3)),'r')
    semilogy((zTotH4(1:Ntot4-
1)+zTotH4(2:Ntot4))/2,abs(Poyntingy6(4:4:Ntot6-2,Tplot6)-
Poyntingy4(:,Tplot4)),'c')
    semilogy((zTotH5(1:Ntot5-
1)+zTotH5(2:Ntot5))/2,abs(Poyntingy6(2:2:Ntot6-2,Tplot6)-
Poyntingy5(:,Tplot5)),'k')
hold off
xlim([zTotE1(kairbot1+1) zTotE1(kairtop1+1)])
title('|Poyntingy (fine) - Poyntingy (course)|')
ylim([10^-5 1])
subplot(2,2,3)
semilogy((zTotH1(1:Ntot1-
1)+zTotH1(2:Ntot1))/2,abs(Poyntingz6(32:32:Ntot6-2,Tplot6)-
Poyntingz1(:,Tplot1)),'b')
hold on
    semilogy((zTotH2(1:Ntot2-
1)+zTotH2(2:Ntot2))/2,abs(Poyntingz6(16:16:Ntot6-2,Tplot6)-
Poyntingz2(:,Tplot2)),'g')
    semilogy((zTotH3(1:Ntot3-
1)+zTotH3(2:Ntot3))/2,abs(Poyntingz6(8:8:Ntot6-2,Tplot6)-
Poyntingz3(:,Tplot3)),'r')
    semilogy((zTotH4(1:Ntot4-
1)+zTotH4(2:Ntot4))/2,abs(Poyntingz6(4:4:Ntot6-2,Tplot6)-
Poyntingz4(:,Tplot4)),'c')
    semilogy((zTotH5(1:Ntot5-
1)+zTotH5(2:Ntot5))/2,abs(Poyntingz6(2:2:Ntot6-2,Tplot6)-
Poyntingz5(:,Tplot5)),'k')
hold off
ylim([10^-5 1])
title('|Poyntingz (fine) - Poyntingz (course)|')
subplot(2,2,4)
semilogy((zTotH1(1:Ntot1-
1)+zTotH1(2:Ntot1))/2,abs(Intensity6(32:32:Ntot6-2)-Intensity1),'b')
hold on
    semilogy((zTotH2(1:Ntot2-
1)+zTotH2(2:Ntot2))/2,abs(Intensity6(16:16:Ntot6-2)-Intensity2),'g')
    semilogy((zTotH3(1:Ntot3-
1)+zTotH3(2:Ntot3))/2,abs(Intensity6(8:8:Ntot6-2)-Intensity3),'r')
    semilogy((zTotH4(1:Ntot4-
1)+zTotH4(2:Ntot4))/2,abs(Intensity6(4:4:Ntot6-2)-Intensity4),'c')
    semilogy((zTotH5(1:Ntot5-
1)+zTotH5(2:Ntot5))/2,abs(Intensity6(2:2:Ntot6-2)-Intensity5),'k')
hold off
xlim([zTotE1(kairbot1+1) zTotE1(kairtop1+1)])
title('|Intensity (fine) - Intensity (course)|')
```

```
ylim([10^-5 1])

% Figure 8 is the plot of the LCP director angles psi and theta.
figure(8)
plot(z6,psiLE,'b')
hold on
  plot(z6,thetaLE,'r')
hold off
title('Director angles \psi (blue) and \theta (red)')
xlabel('z')
```

B. MATLAB CODE FOR CALLED FUNCTIONS

The following functions are called by the main MATLAB program. The three functions, LEbcfun.m, LEodefun.m, and LEodeinit.m, were created by Dr. Eric Choate and not modified for puposes of this thesis.

```
function res = LEbcfun(ya,yb)
global psibot thetabot psitop thetatop;
res = [ ya(1)-psibot
        ya(3)-thetabot
        yb(1)-psitop
        yb(3)-thetatop ];

function dydx=LEodefun(x,y)
dydx=[ y(2)
       2*tan(y(3))*y(2)*y(4)
       y(4)
       -sin(2*y(3))*y(2)^2/2];

function yinit = LEodeinit(x)
global h psibot thetabot psitop thetatop;
psi=psibot+x*(psitop-psibot)/h;
theta=thetabot+x*(thetatop-thetabot)/h;
yinit = [  psi
           (psitop-psibot)/h
           theta
           (thetatop-thetabot)/h];
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Taflove, A. & Hagness, S. C. (2005). *Computational Electrodynamics: The Finite-Difference Time-Domain Method (3rd ed)* Norwood, MA: Artech House, Inc.

Choate, E. P. & Zhou, H. *Laser Propagation in Biaxial Liquid Crystal Polymers.* Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA.

Choate, E. P.& Zhou, H. *Optimization of Electromagnetic Wave Propagation through a Liquid Crystal Layer.* (2012) Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA.

Liquid Crystals (2013) Retrieved from Nobel Prize website: http://www.nobelprize.org/educational/physics/liquid_crystals/history/index.html

Hwang, D. K. & Rey, A. D. (2005) Computational modeling of light propagation in textured liquid crystals based on the finite-difference time-domain (FDTD) method. *Liquid Crystals,* 32(4), 483-497. doi: 10.1080/02678290500033950.

Hwang, D. K. & Rey, A. D. (2005) Computational modeling of the propagation of light through liquid crystals containing twist disclinations based on the FDTD method. *Applied Optics,* 44(21), 4513-4522. doi: 10.1364/AO.44.004513.

Hwang, D. K., Han, W. H. & Rey, A. D. (2007) Computational rheooptics of liquid crystal plymers. *Journal of Non-Newtonian Fluid Mechanics,* 143(1),10-21. doi:10.1016/j.jnnfm.2006.11.006.

Kriezis, E. E. & Elston, S. J. (1999) Finite-difference time domain method for light wave propagation within liquid crystal devices. *Optics Communications,* 165, 99-105. doi: 10.1016/S0030-4018(99)00219-9.

Kriezis, E. E. & Elston S. J. (2000) Light wave propagation in liquid crystal displays by the 2-D finite-difference time-domain method. *Optics Communications,* 177, 69-77. doi: 10.1016/S0030-4018(00)00595-2.

Ziogod, G. D. & Kriezis, E. E. (2008) Modeling light propagation in liquid crystal devices with a 3-D full-vectore finite-element beam propagation method. *Opt Quant Electron,* 40, 733-748. doi: 10.1007/s11082-008-9261-2.

Zhou, H., Forest, M. G. & Wang, H. (2010) Mathematical studies and simulations of nematic liquid crystal polymers and nanocomposties. *Computational and Theoretical Nanoscience,* 7, 1-16.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Michael Winslow
   Naval Postgraduate School
   Monterey, California

4. Hong Zhou
   Naval Postgraduate School
   Monterey, California

5. Eric Choate
   Naval Postgraduate School
   Monterey, California