DESIGN AND ANALYSIS OF FINITE APERTURE DIFFRACTIVE OPTICAL ELEMENTS

by

STEPHEN DOUGLAS MELLIN

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
The Optical Science and Engineering Program
of
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2001

DISSERTATION APPROVAL FORM

Submitted by Stephen Douglas Mellin in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Optical Science and Engineering.

Accepted on behalf of the Faculty of the School of Graduate Studies by the dissertation committee:

_____ Committee Chair

_____

_____

_____

_____

_____

_____ Program Director

_____ College Dean

_____ Graduate Dean

ABSTRACT
School of Graduate Studies
The University of Alabama in Huntsville

Degree Doctor of Philosophy Program Optical Science & Engineering

Name of Candidate Stephen Douglas Mellin

Title DESIGN AND ANALYSIS OF FINITE APERTURE DIFFRACTIVE OPTICAL ELEMENTS

High-efficiency finite-aperture diffractive optical elements with features on the order of or smaller than the wavelength of the incident illumination have been designed. The focus is on the use of a modified scalar-based iterative design method that incorporates the angular spectrum approach. Upon comparison with rigorous electromagnetic analysis techniques (such as the boundary element method and finite difference time domain analysis), it was found that the scalar-based design method was valid for a surprisingly large parameter space, including sub-wavelength features.

Specifically, two-dimensional 1-2 beamfanner designs explored the limits of scalar diffraction theory for finite aperture diffractive optical elements (DOEs) with sub-wavelength minimum feature sizes that operate in the near field. In using scalar diffraction theory, it is assumed that a DOE profile is described by a transmission function, and the field just past the DOE was propagated to a particular plane of interest using the angular spectrum approach. With appropriate choices of weighting functions used in an iterative angular spectrum algorithm (IASA), devices were accurately designed with sub-wavelength features to operate in the near-field. The scalar-based results were compared with those from rigorous electromagnetic analysis

techniques such as the finite-difference time-domain method (FDTD) and the boundary element method (BEM). The scalar results agreed almost identically with rigorous analyses even for wide angular spreads of the desired field pattern in the plane of interest.

Also presented in this dissertation is the design of a diffuser for use in an autostereoscopic display system. The feature sizes of the device were of the same order of magnitude as the incident wavelength. The device was designed using an Iterative Fresnel Tranform Algorithm (IFTA) and the results were compared with those obtained via rigorous analysis.

Abstract Approval: Committee Chair _____
(Date)

Program Director _____

Graduate Dean _____

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Greg Nordin, for guidance throughout this effort. Secondly, I would like to thank the other graduate students, Arthur Ellis, Jianhua Jiang, and Diana Chambers, and other members of the Diffractive Optics group for their collaboration and many enlightening conversations.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

LIST OF SYMBOLS

Symbol	Definition
A	Angular (plane-wave) spectrum (Volts)
$\vec{A}$	Vector potential (Tesla meters)
$\vec{B}$	Magnetic flux density (Tesla)
c	Free space speed of light (3×10^8 meters/second)
$\vec{D}$	Electric displacement (Coulombs/m^2)
D	Maximum diffractive optic etch depth (μm)
d	Diffractive optic etch depth (μm)
d_v	Uniform diffuser viewing distance (cm)
$\vec{E}$	Electric field (Volts/meter)
F	Fourier transform operator
f_x	Spatial frequency (m^{-1})
$\vec{f}$	Boundary element source term (Volts/m^3)
G	Green's function
$\vec{H}$	Magnetic field (Amperes/meter)
I	Irradiance (Watts/m^2)
$\vec{J}$	Electric current density (Amperes/m^2)
j	$\sqrt{-1}$
$\vec{k}$	Wavevector (m^{-1})
n	Material index of refraction (unitless)
r	Radial distance (μm)

$\bar{S}$	Complex Poynting vector (Watts/m^2)
t	Scalar transmission function (unitless)
U	Scalar field (Volts/meter)
Y	Boundary element admittance matrix ($ohms^{-1}$)
Z	Boundary element impedance matrix (ohms)
x, y, z	Spatial coordinates (m)
γ	Euler's constant (1.781…)
ε_0	Free space electric permittivity (Farads/meter)
η	Diffraction efficiency (unitless)
η_0	Impedance in free space (376.7 ohms)
θ	Angular measure (radians)
Λ	Grating spatial period (m)
λ	Wavelength (m)
μ	Free space magnetic permeability (Henrys/meter)
ξ	Boundary element interpolation parameter
ρ	Electric volume charge density (Coulombs/m^3)
τ	Fresnel transmission coefficient (unitless)
φ	Object plane phase (radians)
Φ	Image plane phase (radians)
Ψ	Boundary element partial derivative
ω	Angular frequency (radians/second)

Chapter 1

INTRODUCTION

1.1 Motivation

Diffractive optics play an important role in a variety of applications in modern optics. Such applications include three-dimensional displays, pixelated infrared cameras, focal plane imagers, optical interconnection designs, spectroscopy, microlens arrays, spectral filtering, beamsplitting and beamshaping [1]. Finite aperture diffractive optical elements (DOEs) are of special interest for applications in which an imaging system is composed of an array of DOEs and the finite apertures in the array impose constraints on DOE designs.

Optical devices consisting of individual or arrays of finite aperture diffractive optical elements (DOEs) are of ever increasing interest to the optics community. This dissertation presents some of the analysis methods, both rigorous and scalar, for finite aperture DOEs. It also describes a novel design algorithm. Emphasis is placed on the study of the validity of the use of scalar methods to analyze DOEs with feature size of the same order of magnitude as the wavelength of incident light. This dissertation also presents two selected applications, specifically, the designs of a 1-2 beamfanners used in focal plane imaging and a diffuser for a three-dimensional autostereoscopic display. These applications are of particular interest in this dissertation because they consider the constraints imposed by the finite apertures on the design of individual DOEs.

The fabrication process of a DOE also imposes additional constraints on its design. One limitation is its minimum feature size. For a finite aperture DOE, the minimum feature size establishes the maximum number of partitions in which the DOE may be divided and constrains its design. Another consideration is the ratio of the wavelength of the light illuminating the DOE to its minimum feature size. This relation between the illuminating wavelength and the DOE minimum feature size determines which theoretical models are valid. There are basically three different wavelength regimes to consider: when the wavelength is much less than, comparable to, or much greater than the DOE minimum feature size.

If the wavelength of light is much less than the minimum feature size, then scalar diffraction models are generally valid in analyzing the DOEs [2]. If the wavelength of illuminating light is of the same order of magnitude of the DOE minimum feature size, then scalar diffraction theory is generally no longer valid. In this case, the analysis of a DOE requires a more rigorous model based on electromagnetic theory, such as the Boundary Element Method (BEM) [3], Method of Moments (MOM) [4], or the Finite-Difference Time-Domain method (FDTD) [5]. In the case where the wavelength is much greater than the DOE minimum feature size, alternate methods such as Effective Medium Theory (EMT) can accurately model DOE behavior and may be used instead of a rigorous electromagnetic analysis [6].

In this dissertation, scalar-based and rigorous techniques, such as FDTD and BEM, used to analyze finite aperture DOEs are presented in detail. It turns out that analysis based on scalar diffraction theory using the angular spectrum approach [2] is surprisingly very accurate in comparison to rigorous methods even for DOEs with sub-wavelength minimum feature sizes. One objective of this dissertation is to assess when scalar diffraction theory is applicable for analyzing DOEs. The concept of zones is introduced in Chapter 5 in which a zone is defined as a region in which the DOE profile is locally continuous and has no sharp edges. As will be discussed in Chapter 5, the zone is an accurate indicator to whether scalar diffraction theory is valid.

After having shown that scalar-based diffraction theory is, in fact, valid for analyzing sub-wavelength DOEs in some cases, a design technique based on scalar theory is also presented. The success of this scalar design technique is determined by how well its results agree with those obtained via rigorous analysis. The design applications considered include beamfanners used in focal plane imagers and diffractive optical diffusers used in autostereoscopic three-dimensional displays. Also presented along with the discussion of a scalar-based design method is a novel re-sampling technique that is used to increase the DOE minimum feature size so that the designed DOEs are easier to fabricate.

1.2 Overview of the dissertation

This dissertation is organized as follows. In Chapter 2, the background of some of the techniques of diffractive optic analysis and design is presented. Specific attention is given to scalar transmission theory and the scalar-based design methods.

In Chapter 3, the theoretical development of BEM is presented in detail along with its implementation. Specifically, the basics starting from Maxwell's equations and Green's second identity are first presented. This is followed by the treatment of two- and three-dimensional boundary value problems. Presented next is the method to determine the total fields everywhere in space to completely solve the electromagnetics problem. The implementation of BEM is discussed next with special attention given to the enforcement of boundary conditions.

In Chapter 4, the details of the theoretical development and implementation of the Finite-Difference Time-Domain method (FDTD) are presented. The FDTD method was also used extensively to assess the scalar-based diffractive optic designs. The presentation of FDTD adopts the same nomenclature used by Taflove [5].

Presented in Chapter 5 are design examples of diffractive optic beamfanners that explore limitations of the use of scalar-based diffraction theory to design DOEs. As will be discussed in

greater detail, scalar-based diffraction theory appears to be applicable in several cases for finite aperture DOE design even though the features are on the order of magnitude or less than the wavelength of illuminating light.

Finite-aperture diffractive optical elements (DOEs) have been designed with features on the order of or smaller than the wavelength of the incident illumination. The use of scalar diffraction theory is generally not considered valid for the design of DOEs with such features. However, several cases have been found in which the use of as scalar-based design is, in fact, quite accurate. A modified scalar-based iterative design method has been developed that incorporates the angular spectrum approach to design diffractive optical elements that operate in the near-field and have sub-wavelength features. This design method is called the iterative angular spectrum approach (IASA). This method is a modified version of an Iterative Fourier Transform Algorithm (IFTA) [7]. Upon comparison with a rigorous electromagnetic analysis technique, specifically, the finite difference time-domain method (FDTD), it was found that the scalar-based design method was surprisingly valid, in some cases, for DOEs having sub-wavelength features. These cases are discussed in greater detail in Chapter 6. Also discussed in Chapter 6 is the development of a re-sampling technique that is used to increase the DOE minimum feature size so that the designed DOEs are easier to fabricate and the effect of DOE etch depth quantization is discussed as well.

In Chapter 7, the design of a diffuser for an autostereoscopic display system is presented. The diffuser is designed using a scalar-based method and the results are compared with those obtained from rigorous analyses. Specifically, a modified version of an Iterative Fresnel Transform Algorithm (IFTA) was used to design the diffusers. Chapter 8 provides concluding remarks.

Appendix A contains the derivation of the power in the diffracted orders of an infinite aperture binary optic beamfanner. The results obtained provide the motivation for the method of heuristically designing the 1-2 beamfanner presented in Chapter 5.

Appendix B presents the derivation of the Fresnel approximation from the angular spectrum approach in two dimensions. Since this material is generally not presented in textbooks or elsewhere in the literature of Fourier optics, it is given in this appendix. The Fresnel approximation in two dimensions was used in Chapter 7 in the design and analysis of the vertical diffuser.

Finally, Appendices C through G contain the computer programs, written in Matlab®, used for the analyses and designs of diffractive optical elements presented in this dissertation. Included are the codes for scalar, BEM, and FDTD analyses in Appendices C, D, and E, respectively. Also included are the codes for the IASA design routines and some novel miscellaneous codes in Appendices F and G, respectively.

1.3 New contributions

Major new work that is presented in this dissertation is in the following list:

1. Expanded study into the limits of scalar diffraction theory.
2. Development of a novel scalar-based design algorithm, the Iterative Angular Spectrum Approach (IASA).
3. Development of a re-sampling technique that is used to increase the DOE minimum feature size so that the designed DOEs are easier to fabricate.
4. Novel design application of a diffractive optic diffuser.

Chapter 2

BACKGROUND

Diffractive optics is a viable technology for a variety of applications in modern optics. Brief reviews of analysis and design methods of diffractive optical elements are presented in this chapter to set the stage for the original work presented in subsequent chapters of this dissertation. The review consists of the discussion of techniques used in diffractive optic analysis and proceeds to a more detailed examination of scalar diffraction theory incorporating the angular spectrum approach and its similarity to two-dimensional electromagnetic field propagation. The review concludes with a brief discussion of scalar-based design techniques.

Specifically, Section 2.1 presents a partial overview of the literature of the analysis of DOEs. Techniques such as the Effective Medium Theory (EMT), Rigorous Coupled-Wave Analysis (RCWA), Boundary Element Method (BEM), and Finite-Difference Time-Domain method (FDTD) are present in this section, since they are valuable tools for analyzing DOEs. The choice of analysis generally depends on the type of geometry involved and the size of the features in relation to the illuminating wavelength.

In Section 2.2, the similarities between rigorous electromagnetic theory in two dimensions and the principles of scalar diffraction theory are presented. The purpose is to illustrate that the angular spectrum approach, commonly used in scalar diffraction theory, is completely analogous to TE and TM field propagation in two dimensions.

Section 2.3 presents some scalar techniques used to design DOEs. The motivation for this presentation is that a novel design routine is presented in Chapter 6, and it is appropriate to discuss other design techniques in the literature for completeness.

2.1 Analysis of diffractive optical elements

Modeling a diffractive optic microstructure for a given application critically depends upon the dimensions involved. If the wavelength of the incident light is much smaller than the minimum feature size of the DOE, then the DOE may be treated as a thin phase modulating plate, and scalar diffraction theory is generally applicable [2]. If the wavelength is much greater than the DOE minimum feature size, alternate methods, such as Effective Medium Theory (EMT), can appropriately model the DOE performance [6].

The basic premise of EMT is that only the zero diffraction orders propagate and higher orders are evanescent [6]. The parameters of greatest interest in the use of EMT are the grating thickness, duty cycle, angle of incidence, and form birefringence of the DOE. When using EMT, the diffractive medium is assumed to behave much like a uniaxial crystal in which the TE and TM modes propagate with different velocities. From this, an effective index of refraction profile is calculated and is used in determining the effective phase profile of the DOE. The EMT can serve as an alternative approach to rigorous vector diffraction theory if the features of the DOE are very small [6,8] and it is used extensively in the design of form birefringent diffractive optic gratings [9].

If the minimum feature size of the DOE is of the order of a wavelength, then it is usually necessary to employ rigorous diffraction theory. This requires finding an exact solution to Maxwell's equations and considering the appropriate boundary conditions in analyzing the DOE [1].

Exact methods such as Rigorous Coupled-Wave Analysis (RCWA) are common in the design and analysis of DOEs [10,11,12]. The RCWA assumes that the DOE has an infinite periodic structure, which allows the fields to be expanded in terms of known eigenfunctions. One major advantage of using this technique is that even though the thickness profile is not sampled, the field in any given image plane is known [13]. However, for the applications considered in

this dissertation, RCWA is inadequate, since it applies only to periodic, infinite-aperture structures.

The Boundary Element Method (BEM) [3,13,14] is particularly suited for the analysis of finite aperture DOEs, since it is not restricted to periodic, infinite-aperture structures. The BEM uses the integral form of the wave equation to describe the fields and their normal derivatives at sampled points over the entire boundary of a diffractive optical element (DOE). In this method, the surface field contributions are used to determine the diffractive fields anywhere in space. There are a few assumptions, however, in using boundary methods to analyze DOEs. The assumptions made in the analysis are that the diffractive optic materials are locally homogeneous and isotropic.

Boundary element equations relate the interaction between the incident field and the scattered field on the surface of the DOE. For a dielectric DOE, the surface contribution is an electric or magnetic field that, in general, consists of TE and TM components. Re-radiation from the surface distribution, in turn, generates a diffracted field. The goal in applying a BEM is to numerically determine the surface distributions given the incident field and the DOE thickness profile. The BEM is discussed in full detail in Chapter 3 of this dissertation, since it is used extensively for analyzing finite aperture DOEs.

The Finite-Difference Time-Domain method (FDTD) is another analysis tool specifically suited for the study of finite aperture DOEs. Unlike BEM, however, FDTD is a differential method. First proposed by K.S. Yee in 1966 [5], FDTD is an elegant and very straightforward way to represent the discretized version of the differential form of Maxwell's equations. An electric field grid, offset both spatially and temporally with respect to a magnetic field grid, is used to obtain updated equations that give field values throughout the entire computational grid in terms of previous grid field values. The equations are used in a leapfrog scheme to march the electric and magnetic fields incrementally forward in time. The details of FDTD are discussed in Chapter 4.

Scalar diffraction theory is another technique used to analyze DOEs. Although there are several assumptions incorporated in its implementation that will be discussed later, it is very efficient when it is applicable. Since it is used extensively in this dissertation, the following section is devoted to its development.

2.2 Scalar transmission theory and angular spectrum propagation

Scalar transmission theory is a very efficient method for the analysis and the design of finite aperture DOEs. The basic premise is that the field just past the DOE interface can be described as the incident field multiplied by a phase function [1]. In this section, only linear, isotropic, homogeneous materials are considered. Also, only two-dimensional cases are considered. In such cases, if considering electric or magnetic fields perpendicular to the two-dimensional plane, vector wave propagation theory simplifies to scalar diffraction theory as will be shown next.

Before discussing plane-wave propagation, first consider the following Maxwell's equations:

$$\nabla \times \vec{E} = -\mu \frac{\partial \vec{H}}{\partial t} \quad \text{and} \tag{2.1}$$

$$\nabla \times \vec{H} = \mu\varepsilon \frac{\partial \vec{E}}{\partial t}. \tag{2.2}$$

Assume a time-harmonic steady-state solution in which the partial time derivative can be replaced by $j\omega$ yielding

$$\nabla \times \vec{E} = -j\omega\mu\vec{H} \quad \text{and} \tag{2.3}$$

$$\nabla \times \vec{H} = j\omega\mu\varepsilon\vec{E}, \tag{2.4}$$

in which the permittivity is given as $\varepsilon = n^2\varepsilon_0$ and μ is the permeability of the medium. Taking the curl of both sides of both Ampere's law and Faraday's law and using the identity

$$\nabla \times \nabla \times \mathrm{E} = \nabla\left(\nabla \cdot \vec{\mathrm{E}}\right) - \nabla^2 \vec{\mathrm{E}} \tag{2.5}$$

and assuming that there are no sources present, i.e., $\left(\nabla \cdot \vec{\mathrm{E}}\right) = 0$, Helmoltz' equation can be derived and is given as follows:

$$\nabla^2 \vec{\mathrm{E}} + \mathrm{n}^2 \mathrm{k}_0 \vec{\mathrm{E}} = 0 \tag{2.6}$$

and

$$\mathrm{k}_0 = \frac{\omega}{\mathrm{c}}, \tag{2.7}$$

in which k_0 is the wave number in free space. A similar equation can be derived for the magnetic field. For a transverse magnetic field, Helmholtz' equation is as follows:

$$\nabla^2 \vec{\mathrm{H}} + \mathrm{n}^2 \mathrm{k}_0 \vec{\mathrm{H}} = 0 . \tag{2.8}$$

Note that Helmholtz' equation for a magnetic field is of exactly the same form as it is for an electric field, i.e., of the form $\mathrm{e}^{-\mathrm{j}\vec{\mathrm{k}}\cdot\vec{\mathrm{r}}}$. The reason for mentioning this is that the solutions of Equations 2.6 and 2.8 are identical in form and also formally equivalent, in two dimensions, to the treatment of scalar waves in scalar diffraction theory. The electric and magnetic fields are vector quantities in nature, but in two-dimensions, the formalism of the calculation of optical path differences is completely analogous to the formalism of the transmission function used in scalar diffraction theory next discussed.

One fundamental assumption of scalar transmission theory is that the optical field just past the DOE can be described by a simple transmission function. For scalar-based analysis of DOE diffraction, a transmission function of the form $t(x)= \tau \exp(j \varphi(x)) \operatorname{rect}(x/L)$ is assumed in which L is the width of the finite aperture, and $\operatorname{rect}(x/L)$ equals unity for $|x| \le L/2$ and is zero otherwise. The profile phase, $\varphi(x)$, can be expressed as $\varphi(x) = 2\pi/\lambda_0 \, \Delta n \, d(x)$ in which $d(x)$ is the etch depth along the DOE, as shown in Figure 2.1. The Fresnel transmission coefficient, τ, is given by $\tau = 2\, n_1 \cos(\theta_1)/(n_1 \cos(\theta_1) + n_2 \cos(\theta_2))$ in which n_1 and n_2 are the refractive indices of

the incident and exiting media, respectively, θ_1 is the angle of the incidence with respect to the optical axis, and θ_2 is the refracted angle as calculated by Snell's law. In every case in this dissertation, only normally incident light is considered, for which the Fresnel transmission coefficient reduces to $\tau = 2\,n_1/(n_1 + n_2)$. The field incident on a DOE is multiplied by the transmission function, t(x), which gives the field just past the DOE interface. Note that the development of the transmission function discussed here could just as well have been employed to describe two-dimensional TE electric or TM magnetic fields just past a dielectric interface in an analogous fashion.

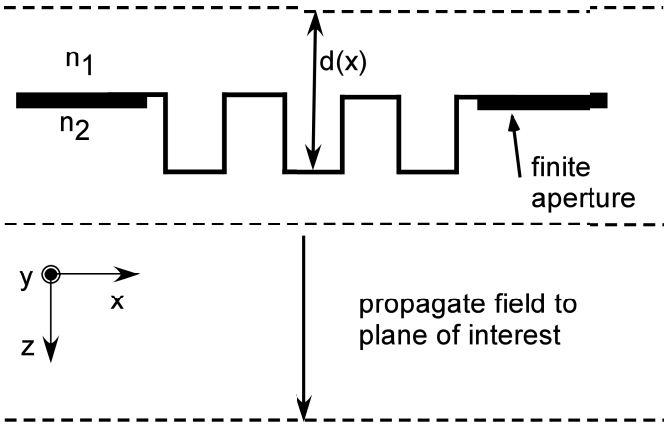

Figure 2.1 Diffractive optic interface.

The field just past the DOE interface can then be propagated to a chosen observation plane using the angular spectrum approach. First, consider Helmholtz' equation and assume that the field takes the form

$$\vec{E} = \vec{E}_0 e^{-j\vec{k}\cdot\vec{r}} = \vec{E}_0 e^{-j(k_x x + k_z z)}. \tag{2.8}$$

Also, assume that k is complex, i.e.,

$$\vec{k} = \vec{k}_r + j\vec{k}_i , \tag{2.9}$$

and $k^2 = \vec{k} \cdot \vec{k} = (\vec{k}_r \cdot \vec{k}_r - \vec{k}_i \cdot \vec{k}_i) + 2j(\vec{k}_r \cdot \vec{k}_i)$,

and if $(\vec{k}_r \cdot \vec{k}_i)=0$, then $k^2 = k_r^2 - k_i^2$.

Consider the wavevector $\vec{k} = k_x \hat{x} + k_z \hat{z}$ and with k_x real, then k_z is

$$k_z = \begin{cases} \sqrt{k^2 - k_x^2} & \text{for } 0 \le k_x^2 \le k^2 \\ -j\sqrt{k_x^2 - k^2} & \text{for } k_x^2 > k^2 \end{cases} . \tag{2.10}$$

The angular spectrum propagation approach uses the Fourier transform of the propagating fields. The Fourier transform and its inverse are given by

$$A_0(f_x) = \int_{-\infty}^{\infty} E(x,z) e^{j(k_x x + k_z z)} dx \tag{2.11}$$

and

$$E(x,z) = \int_{-\infty}^{\infty} A_0(f_x) e^{-j(k_x x + k_z z)} df_x , \tag{2.12}$$

in which f_x is the spatial frequency, and k_x is $k_x = 2\pi f_x$, and the differential is $dk_x = 2\pi df_x$.

Note that determining other field components in two dimensions using Maxwell's equations would require taking partial derivatives with respect to either x or z.

$$\frac{\partial E(x,z)}{\partial x} = \int_{-\infty}^{\infty} (-jk_x) A_0(f_x) e^{-j(k_x x + k_z z)} df_x . \tag{2.13}$$

$$\frac{\partial E(x,z)}{\partial z} = \int_{-\infty}^{\infty} (-jk_z) A_0(f_x) e^{-j(k_x x + k_z z)} df_x . \tag{2.14}$$

One reason for finding other field components is that it is often desirable to determine the irradiance in some plane of interest. This is done by calculating the complex Poynting vector given by

$$\vec{S} = S_x \hat{x} + S_z \hat{z} = \frac{1}{2} \vec{E} \times \vec{H}^* , \tag{2.15}$$

and the irradiance is given by

$$I = \frac{1}{2}\left|\mathrm{Re}\left\{\bar{S}\right\}\cdot \hat{n}\right|, \tag{2.16}$$

and the components of the magnetic field, $\vec{H}$, can be calculated in terms of the spatial partial derivatives of the electric field.

For TE field propagation, in which the electric field is polarized perpendicular to the two-dimensional plane, the irradiance is given by

$$I = \frac{1}{2}\left|\mathrm{Re}\left\{-E_y H_x^*\right\}\right|, \tag{2.17}$$

and E_y is the propagated field given by Equation 2.12. From Equation 2.1, H_x is given by

$$H_x = \frac{1}{j\omega}\frac{\partial E_y}{\partial z}, \tag{2.18}$$

in which $\frac{\partial E_y}{\partial z}$ is determined from Equation 2.14.

For TM field propagation, in which the magnetic field is polarized perpendicular to the two-dimensional plane, the irradiance is given by

$$I = \frac{1}{2}\left|\mathrm{Re}\left\{E_x H_y^*\right\}\right|, \tag{2.19}$$

and H_y is the propagated field analogous in form to Equation 2.12. From Equation 2.2, E_x is given by

$$E_x = \frac{-1}{j\omega}\frac{\partial H_y}{\partial z}, \tag{2.20}$$

in which $\frac{\partial H_y}{\partial z}$ is determined by Equation 2.14 in a similar fashion as the TE case. Note that the irradiance for the TM case is completely identical in form as it is for the TE case.

It is important to note that the concept of irradiance is regarded as an electromagnetic quantity and is generally not mentioned in the literature of scalar diffraction theory. However, since the irradiances for the TE and TM cases are completely identical in two dimensions as

shown above, the concept of irradiance is used in this dissertation, and is incorporated with the development of scalar diffraction theory in two dimensions.

2.3 Design techniques in scalar diffraction theory

This section discusses some of the general procedures available to optimize the design of a DOE for its intended application. A brief overview of the design stages of these procedures and algorithms is presented.

As formulated by Mait [15], many algorithms are presented in the literature, and the following is a list of just a few of them:

1. Quantization
2. Gradient search methods (e.g., steepest descent method)
3. Genetic algorithm
4. Simulated annealing
5. Iterative Fourier Transform Algorithm

The first four methods generally require knowing what the ideal intensity distribution is in advance for a given application. By using a direct sampling technique [16], a continuous DOE thickness profile is assumed and its values are taken at specified intervals along the axes of the object plane. Direct sampling requires that all of the DOE samples are equally spaced along the axes to accommodate later steps of the design algorithm when calculating Fast Fourier Transforms (FFTs) to compute the field in an image plane. The phase profile function is evaluated at discrete intervals and rounded to the nearest allowable quantized phase level allowed by the fabrication process [15]. An example of such a quantized method is a rotationally symmetric iterative discrete on-axis (RSIDO) algorithm. In this algorithm, the phase profile is not directly sampled, but rather the continuous quadratic profile is rounded to the nearest available phase level. The RSIDO method assumes that the DOE has an annular ring structure

and that the algorithm determines the location and phase value of each annular ring. This method has been used for Fresnel zone plate design [16].

Algorithms such as simulated annealing are often used for quantized-phase designs in which all phase levels are rounded to the closest discrete phase level allowed by the fabrication process. Methods such as these often require determining some cost function that characterizes the error (e.g., root-mean-square error) in the intensity produced by the DOE in the image plane. For example, in the case of simulated annealing [17], an ideal intensity profile is determined. Starting with an arbitrary guess of the DOE phase profile, the intensity in the image plane is calculated and is compared to the ideal intensity by calculating a predetermined cost function, which is a quantitative measure of the error between the two. The object phase profile is then altered according to some prescribed mechanism and its intensity profile is calculated. This cost function is then calculated and compared to that of the previous iteration. If the evaluated cost function is less than that of the previous iteration, then the object profile that produced it serves as the new profile. However, if the evaluated cost function is greater than that of the previous iteration, then there is still a probability that the object profile that produced it might serve as the new profile depending upon the size of the difference.

The first four listed methods are examples of quantized algorithms in which the DOE thickness profiles consist of discrete phase levels. These algorithms are also unidirectional, that is, the field distribution in the image plane is not used explicitly to calculate the new DOE etch depth profile after each iteration [15]. In a bidirectional algorithm, the image plane field distribution is used explicitly to calculate the DOE etch depth profile after each iteration of an optimization routine. An example of a bidirectional routine is the Gerchberg-Saxton algorithm [18].

In the Gerchberg-Saxton (GS) algorithm, a phase profile is chosen at random. The phase profile in the object plane is then multiplied by the object reference amplitude profile. For a phase-only diffractive optic, this amplitude profile is unity. Next, the FFT is computed and the

phase information in the image plane is extracted. Then, the product of the phase profile and the reference image plane amplitude is taken. The reference amplitudes are determined by taking the square root of the desired intensity at the sampled positions in the image plane. The product of the reference amplitude and the extracted phase profile is then Inverse Fast Fourier Transformed. Then the phase information is extracted from the Inverse FFT that corresponds to the object phase profile. Constraints such as those imposed by finite apertures on the object are then applied. This process is repeated until an acceptable image intensity is produced, or else until the GS algorithm stagnates.

One major problem that arises with the GS algorithm, in its unmodified form, is that the rendered object phase profiles often do not produce desirable image intensities after many iterations or the algorithm stagnates. Newer methods have been proposed to modify the GS algorithm so that better optimized intensities are produced [7]. Among those newer algorithms, Fractional Fourier Transform techniques (FRT) proposed by Zalevsky, Mendlovic, and Dorsch [19], and the input-output concept proposed by Fienup [20], appear to yield more promising results, since these algorithms are less likely to stagnate than GS in its unmodified form. The major difference between the newer methods and the GS algorithm is that the former only require fractional changes of the image intensity whereas the GS algorithm specifically targets the ideal intensity distribution with each iteration.

The principles of finite aperture DOE design using a modified Iterative Fourier (or Fresnel) Transform Algorithm (IFTA) [20] are the motivation of the design method used in this dissertation. With IFTA, the first step is to choose the form of the object profile of the DOE. The next step is to iteratively calculate the field in the image plane while modifying this field according to a prescribed weighting function. The IFTA relies on recursively calculating the Fourier transforms of object profiles and applying the appropriate constraints on the image and then inverse Fourier transforming the images to produce new object profiles subject to certain

object constraints. In Chapter 7, IFTA is used for a unique application, specifically, the design of a diffuser for use in an autostereoscopic display system.

In Chapter 6, a novel scalar-based method called the Iterative Angular Spectrum Algorithm (IASA) is presented. The method is a modification of IFTA and is used to design 1-2 diffractive optic beamfanners. The novelty of IASA, as its name implies, is that it incorporates the angular spectrum approach into DOE unlike other methods, such as IFTA.

Chapter 3

THE BOUNDARY ELEMENT METHOD

3.1 Introduction

In analyzing finite aperture DOEs, there are a variety of techniques in computational electrodynamics that can be used. One technique used in the analysis of DOEs in this dissertation is the Boundary Element Method (BEM). The purpose of this chapter is to discuss, in detail, the theoretical development of BEM and its implementation. The presentation of the theoretical development BEM in this chapter is, in several respects, a conglomeration of selected previous works presented in the literature by Prather, Mirotznik, and Mait [3,13,14].

The BEM uses the integral form of the wave equation to describe the fields and their normal derivatives at sampled points over the entire boundary of an optical device structure such as a diffractive optical element (DOE). In this method, the surface field contributions are used to determine the diffractive fields anywhere in space. Unlike other methods employing rigorous vector analysis techniques such as rigorous coupled wave analysis (RCWA) [21,22], which are restricted to the analysis of infinitely periodic structures, the BEM is particularly suited for the analysis finite aperture aperiodic DOEs since it is not subject to such restrictions. There are a few assumptions, however, in using boundary methods to analyze DOEs. The assumptions that are made in the analysis are that the DOE is homogeneous and isotropic.

Boundary element equations relate the interaction between the incident field and the scattered field on the surface of the DOE. For a dielectric DOE, the surface contribution is an

electric or magnetic field that, in general, consists of TE and TM components. Re-radiation from the surface distribution, in turn, generates a diffracted field. The goal in applying a BEM is to numerically determine the surface distributions given the incident field and the DOE thickness profile.

This chapter is intended to provide the theoretical development of the rigorous analysis of dielectric diffractive optical elements (DOEs). Starting from Maxwell's equations, the analytical expressions describing the scattered fields due to the presence of a DOE are derived. From Maxwell's equations and the derivation of Green's second identity presented in Section 3.2.1, the mathematical treatment of two- and three-dimensional cases for the boundary value problem in Sections 3.2.2 and 3.2.3, respectively, and boundary conditions are established in Section 3.2.5. An artifact of the BEM construction that deals with integration near singularities is presented in Section 3.2.4. Section 3.2.6 discusses how to determine electromagnetic fields anywhere in space from field values along a dielectric boundary. Emphasis is placed on the examination of the two-dimensional case. Also, the numerical implementation using the point-matching method is presented in Section 3.2.7. Finally, an example of BEM is given in Section 3.2.8.

3.2 BEM formulism

3.2.1 Macroscopic Maxwell's equations and Green's second identity

Like all rigorous electromagnetic methods, the first step is to examine Maxwell's equations. Maxwell's equations in differential form are

$$\nabla \cdot (\varepsilon \vec{E}) = \rho , \tag{3.1}$$

$$\nabla \cdot \vec{H} = 0 , \tag{3.2}$$

$$\nabla \times \vec{E} = -\frac{\partial (\mu \vec{H})}{\partial t} , \text{ and} \tag{3.3}$$

$$\nabla \times \vec{H} = \vec{J} + \frac{\partial(\varepsilon \vec{E})}{\partial t}. \tag{3.4}$$

Assuming time-harmonic dependence and using the field phase convention $\exp[-j(\vec{k} \cdot \vec{r} - \omega t)]$, these equations become

$$\nabla \cdot \vec{E} = \frac{\rho}{\varepsilon}, \tag{3.5}$$

$$\nabla \cdot \vec{H} = 0, \tag{3.6}$$

$$\nabla \times \vec{E} = -j\omega\mu\vec{H}, \text{ and} \tag{3.7}$$

$$\nabla \times \vec{H} = \vec{J} + j\omega\varepsilon\vec{E}. \tag{3.8}$$

Note that the permittivity and permeability in the expressions above are assumed to be both space- and time-invariant.

To obtain the wave equation for the electric field, note that

$$\nabla \times (\nabla \times \vec{E}) = -j\omega\mu(\nabla \times \vec{H}) = -j\omega\mu\vec{J} + \omega^2\mu\varepsilon\vec{E}. \tag{3.9}$$

Using the vector identity,

$$\nabla \times (\nabla \times \vec{E}) = \nabla(\nabla \cdot \vec{E}) - \nabla^2\vec{E} \tag{3.10}$$

yields

$$\nabla^2\vec{E} + k^2\vec{E} = \frac{1}{\varepsilon}\nabla\rho + j\omega\mu\vec{J} = -\vec{f}(\vec{r}), \tag{3.11}$$

in which $k^2 = \omega^2\mu\varepsilon$ in a given medium and $\vec{f}(\vec{r})$ is a spatially dependent source term. The above equation is recognized as Helmholtz' equation and is used in BEM in the analysis of TE wave propagation for two-dimensional problems.

For the case of an incident magnetic field (TM case), Helmholtz' equation is derived in a similar fashion:

$$\begin{aligned} \nabla \times (\nabla \times \vec{H}) &= \nabla(\nabla \cdot \vec{H}) - \nabla^2\vec{H} \\ &= \nabla \times \vec{J} + j\omega\varepsilon\nabla \times \vec{E} \\ &= \nabla \times \vec{J} + \omega^2\mu\varepsilon\vec{H} \end{aligned} \tag{3.12}$$

and

$$\nabla^2\vec{H} + \omega^2\mu\varepsilon\vec{H} = -\nabla\times\vec{J} = -\vec{f}(\vec{r}) \,. \tag{3.13}$$

Again, $\vec{f}(\vec{r})$ is a spatially dependent source term. In the analysis, it will turn out to be more convenient to treat the TE and TM cases separately and assume either an incident $\vec{E}$ or $\vec{H}$ field for a given case.

Since the analysis of a dielectric DOE is treated as a boundary value problem, it is very helpful to not only introduce but also derive Green's second identity, which is done in this section. In the next section it will be used in the formulation of the three-dimensional boundary value problem.

Starting with the Divergence theorem

$$\int_V \nabla\cdot A \, dV' = \oiint_S \vec{A}\cdot\hat{n}\, dS' \tag{3.14}$$

and letting

$$\vec{A} = \phi\nabla\psi \tag{3.15}$$

yields upon substitution

$$\begin{aligned} \nabla\cdot\vec{A} &= \nabla\cdot(\phi\nabla\psi) \\ &= \phi\nabla^2\psi + \nabla\phi\cdot\nabla\psi \end{aligned} \tag{3.16}$$

and

$$\begin{aligned} \vec{A}\cdot\hat{n} &= (\phi\nabla\psi)\cdot\hat{n} \\ &= \phi\frac{\partial\psi}{\partial n}, \end{aligned} \tag{3.17}$$

giving

$$\int_V (\phi\nabla^2\psi + \nabla\phi\cdot\nabla\psi)\, dV' = \oiint_S \phi\frac{\partial\psi}{\partial n} dS' \,, \tag{3.18}$$

in which the normal vector, $\hat{n}$, is directed outward from the closed surface as shown in Figure 3.1. This result is Green's 1st identity. Note that the quantity, ∂n, in Equation 3.18, is a

partial differential along the contour with respect to the primed coordinates. The quantity, n, is left unprimed for notational convenience.

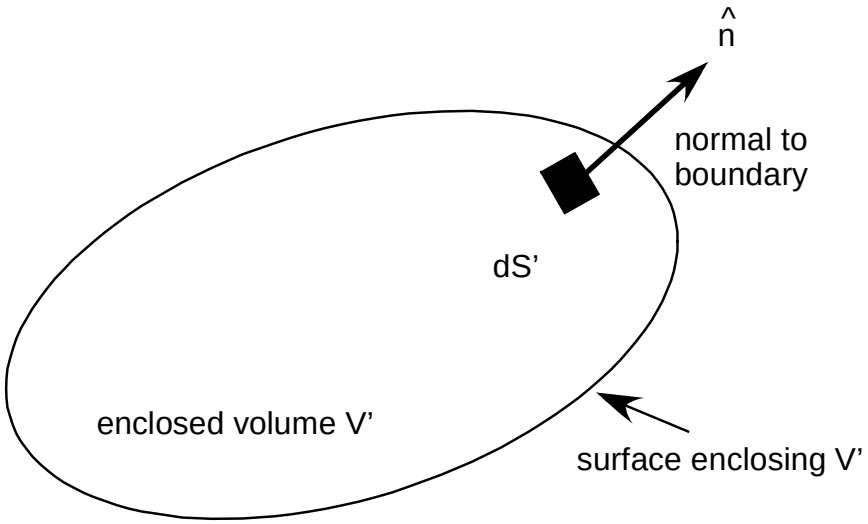

Figure 3.1 Geometry used in the derivation of Green's 2nd theorem.

Now letting

$$\vec{A} = \psi \nabla \phi \tag{3.19}$$

yields a similar equation to the one above,

$$\int_V (\psi \nabla^2 \phi + \nabla \phi \cdot \nabla \psi)\, dV' = \oiint_S \psi \frac{\partial \phi}{\partial n} dS'. \tag{3.20}$$

Subtracting Equation 3.20 from Equation 3.18 gives

$$\int_V (\phi \nabla^2 \psi - \psi \nabla^2 \phi)\, dV' = \oiint_S (\phi \frac{\partial \psi}{\partial n} - \psi \frac{\partial \phi}{\partial n}) dS'. \tag{3.21}$$

This important result is Green's 2nd identity.

3.2.2 Treatment of the three-dimensional boundary value problem

It is best to examine the behavior of the TE and TM waves separately since separate boundary conditions are imposed on each of them. First consider the TE case. Let $\vec{E} = E_y \hat{y}$ and replace φ with E_y and ψ with G in Equation 3.21. This gives

$$\int_V (E_y \nabla^2 G - G \nabla^2 E_y)\, dV' = \oiint_S (E_y \frac{\partial G}{\partial n} - G \frac{\partial E_y}{\partial n}) dS' . \tag{3.22}$$

The unbounded Green's function is given by

$$G(\vec{r}, \vec{r}\,') = \frac{e^{-jk|\vec{r} - \vec{r}\,'|}}{4\pi|\vec{r} - \vec{r}\,'|} \tag{3.23}$$

and is a particular solution to the equation

$$\nabla^2 G(\vec{r}, \vec{r}\,') + k^2 G(\vec{r}, \vec{r}\,') = -\delta(\vec{r} - \vec{r}\,') . \tag{3.24}$$

Combining Equations 3.11, 3.22, and 3.24 gives

$$\begin{aligned} &\int_V [E_y(-\delta(\vec{r} - \vec{r}\,') - k^2 G(\vec{r}, \vec{r}\,')) - G(\vec{r}, \vec{r}\,')(-f(\vec{r}\,') - k^2 E_y)] dV' = \\ &\int_V [-E_y \delta(\vec{r} - \vec{r}\,') + G(\vec{r}, \vec{r}\,') f(\vec{r}\,')] dV' = -E_y(\vec{r}) + \int_V G(\vec{r}, \vec{r}\,') f(\vec{r}\,') dV' \\ &\qquad = \oiint_S [E_y \frac{\partial G(\vec{r}, \vec{r}\,')}{\partial n} - G(\vec{r}, \vec{r}\,') \frac{\partial E_y(\vec{r}\,')}{\partial n}] dS' . \end{aligned} \tag{3.25}$$

As shown in Figure 3.2, the sources and, hence, the incident fields are assumed to originate in Region 1, so

$$\begin{aligned} -E_y(\vec{r}) + E_{inc}(\vec{r}) \delta_{i,1} = \oiint_S \Bigg(& E_y(\vec{r}\,') \frac{\partial G_i(\vec{r}, \vec{r}\,')}{\partial n} \\ & - G_i(\vec{r}, \vec{r}\,') \frac{\partial E_y(\vec{r}\,')}{\partial n} \Bigg) dS' \end{aligned} , \tag{3.26}$$

in which

$$E_{inc}(\vec{r}) \delta_{i,1} = \int_V G_i(\vec{r}, \vec{r}\,') f_i(\vec{r}\,') dV' \tag{3.27}$$

and the sources producing the incident field are assumed to originate in a region designated by Region 1 and the Kronecker delta is $\delta_{i,1} = \begin{cases} 1, & i = 1 \\ 0, & i \neq 1 \end{cases}$, where i denotes either Region 1 or 2, the incident and exiting media, respectively. Equation 3.27 requires proof which is done next. It should be noted that the following proof was performed without the aid of references since none were found in the literature.

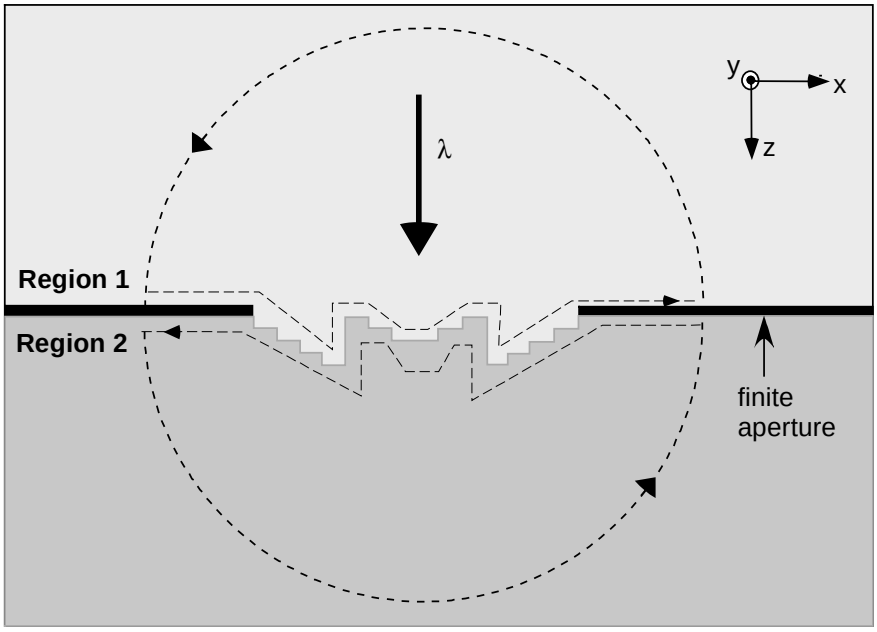

Figure 3.2 BEM geometry for a DOE contour.

To prove that $E_{inc}(\vec{r})\delta_{i,1} = \int_V G_i(\vec{r},\vec{r}\,')f_i(\vec{r}\,')dV'$ start with

$$\vec{f}(\vec{r}) = -\frac{1}{\varepsilon}\nabla\rho - j\omega\mu\vec{J} \tag{3.28}$$

in Region 1 which contains the sources; it follows that

$$\int_v \vec{f}(\vec{r}\,')G_i(\vec{r},\vec{r}\,')dV' = -\frac{1}{\varepsilon}\int_V \nabla'\rho(\vec{r}\,')G_i(\vec{r},\vec{r}\,')dV' - j\omega\mu\int_V \vec{J}(\vec{r}\,')G_i(\vec{r},\vec{r}\,')dV'. \tag{3.29}$$

By analogy with linear circuit theory, the Green's function is the impulse response of this system and any linear operation acting upon an input has the exact same effect on the output; therefore,

$$-\frac{1}{\varepsilon}\int_V \nabla'\rho(\vec{r}\,')G_i(\vec{r},\vec{r}\,')dV' = -\frac{1}{\varepsilon}\nabla\int_V \rho(\vec{r}\,')G_i(\vec{r},\vec{r}\,')dV'. \tag{3.30}$$

From classical electrodynamics, the electric potential Φ in terms of the volume charge density is given by

$$\Phi(\vec{r}) = \frac{1}{\varepsilon}\int_V \rho(\vec{r}\,')G(\vec{r},\vec{r}\,')dV'. \tag{3.31}$$

Also, the vector potential is given by

$$\vec{A}(\vec{r}) = \mu\int_V \vec{J}(\vec{r}\,')G(\vec{r},\vec{r}\,')dV'. \tag{3.32}$$

In general, the electric and magnetic fields are given by

$$\vec{E}(\vec{r},t) = -\nabla\Phi(\vec{r},t) - \frac{\partial\vec{A}(\vec{r},t)}{\partial t} \text{ and} \tag{3.33}$$

$$\vec{H}(\vec{r},t) = \frac{1}{\mu}\nabla\times\vec{A}(\vec{r},t). \tag{3.34}$$

Given the dependence on source terms in Equations 3.33 and 3.34, the electric and magnetic fields may be treated as incident fields.

For the case of time-harmonic dependence, these equations become

$$\vec{E}(\vec{r}) = -\nabla\Phi(\vec{r}) - j\omega\vec{A}(\vec{r}) \text{ and} \tag{3.35}$$

$$\vec{H}(\vec{r}) = \frac{1}{\mu}\nabla\times\vec{A}(\vec{r}), \tag{3.36}$$

and upon substitution

$$\int_v \vec{f}_i(\vec{r}\,')G_i(\vec{r},\vec{r}\,')dV' = -\nabla\Phi(\vec{r}) - j\omega\vec{A}(\vec{r}). \tag{3.37}$$

Therefore,

$$\int_v \vec{f}(\vec{r}\,')G(\vec{r},\vec{r}\,')dV' = \vec{E}_{inc}(\vec{r}) . \tag{3.38}$$

Also note for an incident magnetic field, that

$$\vec{f}(\vec{r}) = \nabla \times \vec{J} \tag{3.39}$$

and that

$$\begin{aligned} \int_v \vec{f}(\vec{r}\,')G(\vec{r},\vec{r}\,')dV' &= \int_v (\nabla' \times \vec{J}(\vec{r}\,'))G(\vec{r},\vec{r}\,')dV \\ &= \nabla \times \int_v \vec{J}(\vec{r}\,')G(\vec{r},\vec{r}\,')dV \\ &= \frac{1}{\mu}\nabla \times \vec{A}(\vec{r}) = \vec{H}_{inc}(\vec{r}) \end{aligned} . \tag{3.40}$$

3.2.3 Treatment of the two-dimensional boundary value problem

Note that for the following, $\hat{n}_2 = -\hat{n}_1 = \hat{n}$, since the normal unit vectors of the closed surfaces enclosing the volumes are different.

In Region 1:

$$E_1(\vec{r}) = E_{inc}(\vec{r}) + \oiint_S (E_1(\vec{r}\,')\frac{\partial G_1(\vec{r},\vec{r}\,')}{\partial n} - G_1(\vec{r},\vec{r}\,')\frac{\partial E_1(\vec{r}\,')}{\partial n})dS' . \tag{3.41}$$

In Region 2:

$$E_2(\vec{r}) = -\oiint_S (E_2(\vec{r}\,')\frac{\partial G_2(\vec{r},\vec{r}\,')}{\partial n} - G_2(\vec{r},\vec{r}\,')\frac{\partial E_2(\vec{r}\,')}{\partial n})dS' . \tag{3.42}$$

For two-dimensional problems in which the fields and their normal derivatives are assumed constant with respect to y', the surface integral dS' is designated by $dS' = dy'd\ell'$ and the integration can be performed over y'. In doing so, the integration only acts on the Green's function.

$$\begin{aligned} \int_{-\infty}^{\infty} G(\vec{r},\vec{r}\,')dy' &= \int_{-\infty}^{\infty} \frac{e^{-jk|\vec{r}-\vec{r}\,'|}}{4\pi|\vec{r}-\vec{r}\,'|}dy' \\ &= \frac{1}{4j}H_0^{(2)}(k|\vec{r}-\vec{r}\,'|) \end{aligned} , \tag{3.43}$$

in which $H_0^{(2)}$ is the zero order Hankel function of the second kind. Since the normal vector is in the x-z plane, the two-dimensional Green's function may be substituted everywhere in Equation 3.41. This gives in Region 1,

$$E_1(\vec{r}) = E_{inc}(\vec{r}) + \int_C (E_1(\vec{r}\,')\frac{\partial G_1(\vec{r},\vec{r}\,')}{\partial n} - G_1(\vec{r},\vec{r}\,')\frac{\partial E_1(\vec{r}\,')}{\partial n})d\ell' . \tag{3.44}$$

In Region 2,

$$E_2(\vec{r}) = -\int_C (E_2(\vec{r}\,')\frac{\partial G_2(\vec{r},\vec{r}\,')}{\partial n} - G_2(\vec{r},\vec{r}\,')\frac{\partial E_2(\vec{r}\,')}{\partial n})d\ell' . \tag{3.45}$$

The contour, C, is the two-dimensional boundary of the diffractive optical element and

$$G(\vec{r},\vec{r}\,') = \frac{1}{4j}H_0^{(2)}\left(k_i\left|\vec{r}-\vec{r}\,'\right|\right) \text{ and} \tag{3.46}$$

$$\frac{\partial G(\vec{r},\vec{r}\,')}{\partial n} = k_i\frac{j}{4}H_1^{(2)}\left(k_i\left|\vec{r}-\vec{r}\,'\right|\right)\hat{n}\cdot\hat{r}\,, \tag{3.47}$$

in which $\hat{r} = \frac{\vec{r}-\vec{r}\,'}{\left|\vec{r}-\vec{r}\,'\right|}$. Note that caution must be exercised in using Equation 3.47 since the normal vectors are different for Regions 1 and 2.

3.2.4 The effect of singularities

Since the Green's functions are singular at positions where $\vec{r} = \vec{r}\,'$, care must be exercised when evaluating Equations 3.46 and 3.47. The singularities are handled by considering a small circular contour of radius ε as shown in Figure 3.3.

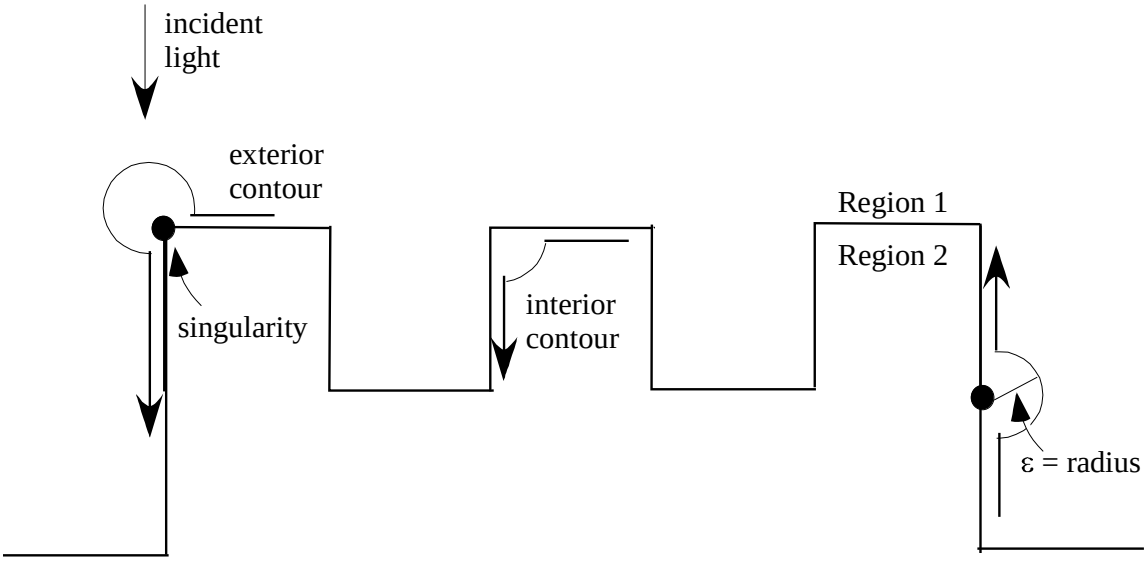

Figure 3.3 Singularities encountered in contour integration along DOE boundary.

Consider the following integral,

$$\oint_C G(\vec{r},\vec{r}\,')\frac{\partial E_2(\vec{r}\,')}{\partial n}d\ell' = \oint_{Cauchy} G(\vec{r},\vec{r}\,')\frac{\partial E_2(\vec{r}\,')}{\partial n}d\ell + \lim_{\varepsilon\to 0}\int_{-\frac{\theta}{2}}^{\frac{\theta}{2}} G(\vec{r},\vec{r}\,')\frac{\partial E_2(\vec{r}\,')}{\partial n}\varepsilon d\theta', \tag{3.48}$$

where $\varepsilon = \lim_{\vec{r}'\to\vec{r}}\left|\vec{r}-\vec{r}\,'\right|$ and θ is the exterior angle at a boundary point. Using the small argument approximation for Hankel's functions,

$$H_0^{(2)}(x) \approx 1 - j\frac{2}{\pi}\ln\left(\frac{\gamma x}{2}\right) \quad \gamma = 1.781, \tag{3.49}$$

where γ=Euler's constant and

$$H_1^{(2)}(x) \approx \frac{x}{2} + j\frac{2}{\pi x} \qquad x \to 0 \text{ and} \tag{3.50}$$

$$\oint_C G(\vec{r},\vec{r}')\frac{\partial E_2(\vec{r}')}{\partial n}d\ell' = \oint_{Cauchy} G(\vec{r},\vec{r}')\frac{\partial E_2(\vec{r}')}{\partial n}d\ell' + \lim_{\varepsilon\to 0}\int_{-\frac{\theta}{2}}^{\frac{\theta}{2}}[1-j\frac{2}{\pi}\ln(\frac{\gamma k_i\varepsilon}{2})]\frac{\partial E_2(\vec{r}')}{\partial n}\varepsilon\, d\theta' \tag{3.51}$$

since $\lim_{\varepsilon\to 0}\varepsilon\ln(\varepsilon) = 0$,

$$\oint_C G(\vec{r},\vec{r}')\frac{\partial E_2(\vec{r}')}{\partial n}d\ell' = \oint_{Cauchy} G(\vec{r},\vec{r}')\frac{\partial E_2(\vec{r}')}{\partial n}d\ell. \tag{3.52}$$

For the other integral,

$$\begin{aligned}\oint_C E(\vec{r}')\frac{\partial G_i(\vec{r},\vec{r}')}{\partial n}d\ell' &= \oint_{Cauchy} E(\vec{r}')\frac{\partial G_i(\vec{r},\vec{r}')}{\partial n}d\ell' \\ &+ \frac{jk_i}{4}(\hat{n}\cdot\hat{r})\lim_{\varepsilon\to 0}\int_{-\frac{\theta}{2}}^{\frac{\theta}{2}} E(\vec{r}')[\frac{k_i\varepsilon}{2}+j\frac{2}{\pi k_i\varepsilon}]\varepsilon\, d\theta' \\ &= \oint_{Cauchy} E(\vec{r}')\frac{\partial G_i(\vec{r},\vec{r}')}{\partial n}d\ell' - \frac{\theta}{2\pi}E(\vec{r}).\end{aligned} \tag{3.53}$$

Note that $\hat{n}\cdot\hat{r} = 1$ over the circular arc.

After performing the contour integration, the fields at a point on the boundary become

$$E_1(\vec{r}_s)\left(\frac{\theta}{2\pi}\right) = E_{inc}(\vec{r}_s) + \oint_C\left(E_1(\vec{r}')\frac{\partial G_1(\vec{r},\vec{r}')}{\partial n} - G_1(\vec{r},\vec{r}')\frac{\partial E_1(\vec{r}')}{\partial n}\right)d\ell' \tag{3.54}$$

and

$$E_2(\vec{r}_s)\left(1-\frac{\theta}{2\pi}\right) + \int_C (E_2(\vec{r}')\frac{\partial G_2(\vec{r},\vec{r}')}{\partial n} - G_2(\vec{r},\vec{r}')\frac{\partial E_2(\vec{r}')}{\partial n})d\ell' = 0 \tag{3.55}$$

for Regions 1 and 2, respectively, in which θ is the exterior angle as determined in region 2.

3.2.5 Determination of the boundary conditions

Figure 3.4 shows the fields at an interface between Regions 1 and 2. In general, boundary conditions take the form

$$\hat{n}\times(\vec{E}_2 - \vec{E}_1) = 0, \tag{3.56}$$

$$\hat{n} \times (\vec{H}_2 - \vec{H}_1) = \vec{J}_s , \tag{3.57}$$

$$\hat{n} \cdot (\vec{D}_2 - \vec{D}_1) = \rho_s , \text{ and} \tag{3.58}$$

$$\hat{n} \cdot (\vec{B}_2 - \vec{B}_1) = 0 , \tag{3.59}$$

where $\vec{J}_s$ and ρ_s are the surface current and charge density on the boundary.

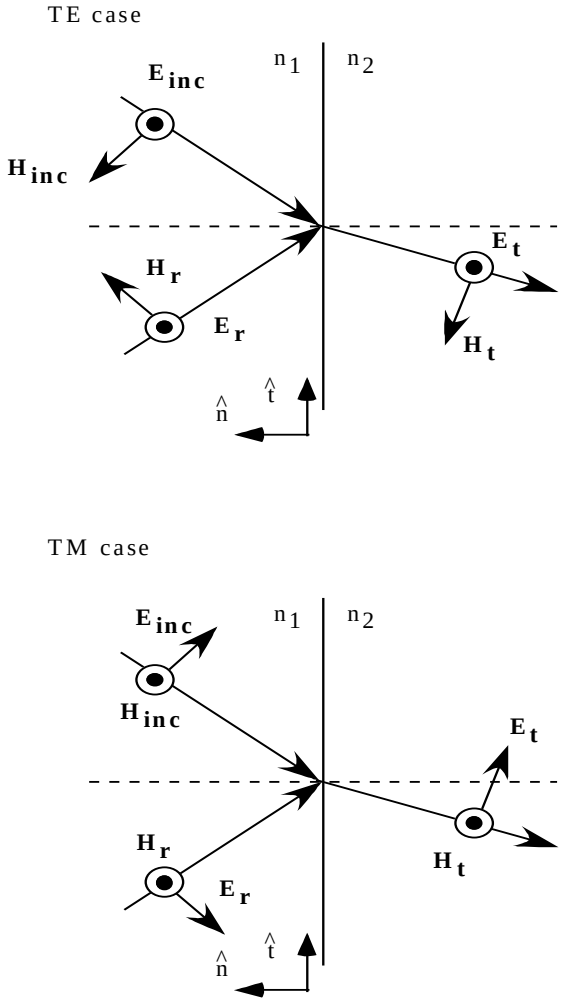

Figure 3.4 TE and TM fields at a dielectric boundary.

For the TE case,

$$\hat{n}\times(\vec{E}_2-\vec{E}_1)=0 \quad \text{yielding } E_y|_1=E_y|_2 . \tag{3.60}$$

For the normal derivative, note that $\vec{E}=E_y\hat{y}$ and the following vector identities,

$$\vec{a}\times(\vec{b}\times\vec{c})=(\vec{a}\cdot\vec{c})\vec{b}-(\vec{a}\cdot\vec{b})\vec{c}\,, \tag{3.61}$$

$$\vec{a}\cdot(\vec{b}\times\vec{c})=\vec{b}\cdot(\vec{c}\times\vec{a})=\vec{c}\cdot(\vec{a}\times\vec{b})\,, \tag{3.62}$$

$$\vec{E}=E_y\hat{y}=\left|\vec{E}\right|e^{-j(\vec{k}\cdot\vec{r})}\hat{y}\,, \text{ and} \tag{3.63}$$

$$\nabla E_y=-j\vec{k}E_y=-j(\vec{k}\cdot\hat{t})E_y\hat{t}+(-j)(\vec{k}\cdot\hat{n})E_y\hat{n}\,, \tag{3.64}$$

in which $\hat{t}$ and $\hat{n}$ are the tangential and normal unit vectors, respectively. Using the time-harmonic Maxwell's equation

$$\nabla\times\vec{E}=-j\omega\mu\vec{H}=-j\vec{k}\times\vec{E} \tag{3.65}$$

and vector identities above,

$$\vec{k}=\frac{\omega\mu}{(\vec{E}\cdot\vec{E})}\vec{E}\times\vec{H}\,, \tag{3.66}$$

$$\nabla E_y=-\frac{jE_y\omega\varepsilon}{(\vec{E}\cdot\vec{E})}[((\vec{E}\times\vec{H})\cdot\hat{t})\hat{t}+((\vec{E}\times\vec{H})\cdot\hat{n})\hat{n}]\,, \text{ and} \tag{3.67}$$

$$\hat{n}\cdot\nabla E_y=-j\omega\mu(\hat{y}\times\vec{H})\cdot\hat{n}=-j\omega\mu\,\hat{y}\cdot(\vec{H}\times\hat{n})=j\omega\mu\,\hat{y}\cdot(\hat{n}\times\vec{H})\,. \tag{3.68}$$

Recall that $\hat{n}\times(\vec{H}_2-\vec{H}_1)=0$ at the boundary. So for Region 1,

$$\hat{n}\cdot\nabla E_y|_1=j\omega\mu\,\hat{y}\cdot(\hat{n}\times\vec{H}_1)\,, \tag{3.69}$$

and in Region 2,

$$\hat{n}\cdot\nabla E_y|_2=j\omega\mu\,\hat{y}\cdot(\hat{n}\times\vec{H}_2) \text{ and since } \hat{n}\times(\vec{H}_2-\vec{H}_1)=0 \tag{3.70}$$

and

$$\hat{n}\cdot\nabla E_y|_1 = \hat{n}\cdot\nabla E_y|_2\,, \tag{3.71}$$

or equivalently

$$\left.\frac{\partial E_y}{\partial n}\right|_1 = \left.\frac{\partial E_y}{\partial n}\right|_2\,. \tag{3.72}$$

For the TM case,

$$\hat{n}\times(\vec{H}_2-\vec{H}_1)=\vec{J}_s=0\,, \tag{3.73}$$

assuming no surface currents yielding

$$H_y|_1 = H_y|_2\,. \tag{3.74}$$

For the normal derivative, note that $\vec{H}=H_y\hat{y}$ and using vector identities yields

$$\nabla H_y = -j\vec{k}H_y = -j(\vec{k}\cdot\hat{t})H_y\hat{t}+(-j)(\vec{k}\cdot\hat{n})H_y\hat{n}\,. \tag{3.75}$$

Again, using the time-harmonic Maxwell's equations and the vector identities yields

$$\vec{k}=\frac{\omega\varepsilon}{(\vec{H}\cdot\vec{H})}\vec{E}\times\vec{H}\,, \tag{3.76}$$

$$\begin{aligned}\nabla H_y &= -\frac{jH_y\omega\varepsilon}{(\vec{H}\cdot\vec{H})}[((\vec{E}\times\vec{H})\cdot\hat{t})\hat{t}+((\vec{E}\times\vec{H})\cdot\hat{n})\hat{n}]\\ &= -\frac{jH_y\omega\varepsilon}{(\vec{H}\cdot\vec{H})}[\vec{H}\cdot(\hat{t}\times\vec{E})\hat{t}+(\vec{H}\cdot(\hat{n}\times\vec{E})\hat{n}],\ \text{and}\end{aligned} \tag{3.77}$$

$$\begin{aligned}\hat{n}\cdot\nabla H_y &= \frac{-jH_y\omega\varepsilon}{(\vec{H}\cdot\vec{H})}\vec{H}\cdot(\hat{n}\times\vec{E})\\ &= -j\omega\varepsilon\,\hat{y}\cdot(\hat{n}\times\vec{E})\end{aligned}\,. \tag{3.78}$$

Recall that $\hat{n}\times(\vec{E}_2-\vec{E}_1)=0$ at boundary and that $\varepsilon=\varepsilon_0 n_1^2$.

So for Region 1,

$$\hat{n}\cdot\nabla H_y|_1 = -j\omega\varepsilon_0 n_1^2\vec{H}_1\cdot(\hat{n}\times\vec{E}_1) = -j\omega\varepsilon\,\hat{y}\cdot(\hat{n}\times\vec{E}_1)\,, \tag{3.79}$$

and in Region 2,

$$\hat{n}\cdot\nabla H_y|_2 = -j\omega\varepsilon_0 n_2^2\vec{H}_2\cdot(\hat{n}\times\vec{E}_2) = -j\omega\varepsilon\,\hat{y}\cdot(\hat{n}\times\vec{E}_2)\,. \tag{3.80}$$

Therefore,

$$\frac{1}{n_1^2}\hat{n}\cdot\nabla H_y\,|_1=\frac{1}{n_2^2}\hat{n}\cdot\nabla H_y\,|_2\,, \tag{3.81}$$

or equivalently

$$\left.\frac{1}{n_1^2}\frac{\partial H_y}{\partial n}\right|_1=\left.\frac{1}{n_2^2}\frac{\partial H_y}{\partial n}\right|_2. \tag{3.82}$$

On the surface, the fields on the surface on the boundary of the dielectric diffractive optical element for either the TE and TM incident illumination may be represented by

$$\Phi_1(\vec{r}_s)\left(\frac{\theta}{2\pi}\right)=\Phi_{inc}(\vec{r}_s)+\oint_C\left(\Phi_1(\vec{r}\,')\frac{\partial G_1(\vec{r},\vec{r}\,')}{\partial n}-p_1G_1(\vec{r},\vec{r}\,')\frac{\partial \Psi_1(\vec{r}\,')}{\partial n}\right)d\ell' \tag{3.83}$$

and

$$\Phi_2(\vec{r}_s)\left(1-\frac{\theta}{2\pi}\right)+\int_C(\Phi_2(\vec{r}\,')\frac{\partial G_2(\vec{r},\vec{r}\,')}{\partial n}-p_2G_2(\vec{r},\vec{r}\,')\frac{\partial \Psi_2(\vec{r}\,')}{\partial n})d\ell'=0 \tag{3.84}$$

for Regions 1 and 2, respectively, in which

$$\Phi=\begin{cases}E_y & \text{for TE}\\ H_y & \text{for TM}\end{cases},\ \Psi_i=\begin{cases}\dfrac{\partial E_y}{\partial n} & \text{for TE}\\ \dfrac{1}{n_i^2}\dfrac{\partial H_y}{\partial n} & \text{for TM}\end{cases},\ \text{and}\ p_i=\begin{cases}1 & \text{for TE}\\ n_i^2 & \text{for TM}\end{cases}, \tag{3.85}$$

for i=1,2.

3.2.6 Calculation of the total fields

Finding the total fields requires determining the scattered fields above. Numerical techniques such as BEM are required, in general, since the scattered fields may only be solved analytically for a handful of problems. But once the fields and their normal derivatives on the surface are determined, the following equations can be used to determine the total fields:

$$E_1(\vec{r})=E_{inc}(\vec{r}_s)+\oint_C\left(E_1(\vec{r}\,')\frac{\partial G_1(\vec{r},\vec{r}\,')}{\partial n}-G_1(\vec{r},\vec{r}\,')\frac{\partial E_1(\vec{r}\,')}{\partial n}\right)d\ell' \tag{3.86}$$

and

$$E_2(\vec{r}) = -\int_C \left(E_2(\vec{r}\,')\frac{\partial G_2(\vec{r},\vec{r}\,')}{\partial n} - G_2(\vec{r},\vec{r}\,')\frac{\partial E_2(\vec{r}\,')}{\partial n}\right)d\ell' \tag{3.87}$$

for Regions 1 and 2, respectively. Similar equations hold for the magnetic fields.

For the analysis of finite aperture DOEs, it is often of interest to determine the field at some plane in the exiting medium. This can be achieved using Equation 3.87. In some applications, however, it is often of interest to determine the fields in more than just one plane. Therefore, to make the computation more efficient, Equation 3.87 need only be used once in calculating the field just past a DOE interface, and a Fast Fourier Transform (FFT) technique such as the angular spectrum approach [2,23] can be used to determine the field at any other plane.

3.2.7 Numerical implementation of the boundary element method for dielectric structures

So that the BEM can be implemented in a computer algorithm, N sample points are taken over an entire modeled dielectric boundary such that the scattered electric field and its normal derivative can be expressed in terms of interpolation functions as follows [3]:

$$E^{sc}(r'(\xi)) = \sum_{n=1}^{N} \hat{E}_n^{sc} = \sum_{n=1}^{N} E_n^{sc}(\xi)\phi_1(\xi) + E_{n+1}^{sc}(\xi)\phi_2(\xi) \tag{3.88}$$

and

$$\Psi^{sc}(r'(\xi)) = \sum_{n=1}^{N} \hat{\Psi}_n^{sc} = \sum_{n=1}^{N} \Psi_n^{sc}(\xi)\phi_1(\xi) + \Psi_{n+1}^{sc}(\xi)\phi_2(\xi)\,, \tag{3.89}$$

in which $\phi_1(\xi)$ and $\phi_2(\xi)$ are interpolation functions and n denotes a sampled point on the boundary. If $\phi_1(\xi)$ and $\phi_2(\xi)$ are linear interpolation functions, for example, they may be expressed as

$$\phi_1(\xi) = \frac{(1-\xi)}{2} \tag{3.90}$$

and

$$\phi_1(\xi) = \frac{(1+\xi)}{2}, \tag{3.91}$$

with ξ =[-1,1]. Higher-order interpolation functions may also be used if desired [24]. Using this notation, the contour of the DOE is represented by a local coordinate transformation,

$$r'(\xi) = r'[\hat{x}_n(\xi), \hat{y}_n(\xi)], \tag{3.92}$$

in which

$$\hat{x}_n(\xi) = x_n(\xi)\phi_1(\xi) + x_{n+1}(\xi)\phi_2(\xi) \tag{3.93}$$

and

$$\hat{y}_n(\xi) = y_n(\xi)\phi_1(\xi) + y_{n+1}(\xi)\phi_2(\xi). \tag{3.94}$$

The nodal coordinates (x_n,y_n) and (x_{n+1},y_{n+1}) represent sampled points on the dielectric boundary and $\hat{x}_n(\xi)$ and $\hat{y}_n(\xi)$ are interpolated coordinate values between nodes. Note that in using the BEM formulation, the nodal coordinate values are ordered in a sequential counterclockwise manner.

Substitution of Equations 3.88 and 3.89 into Equation 3.85 yields a system of two equations and 2N unknowns. To obtain a set of 2N equations and 2N unknowns to determine the fields and their normal derivatives at the sampled points on the boundary of the dielectric, an inner product between both sides of Equation 3.85 is taken with a set of N weighting functions. In this analysis, Dirac-delta functions serve as the weighting function that sample the boundary of the dielectric:

$$\omega_m = \delta(r'(\xi) - r_m') = \begin{cases} 1, & r'(\xi) = r_m' \\ 0, & \text{otherwise} \end{cases}, \tag{3.95}$$

in which m=1,2,…,N and r_m' is a set of N position vectors along the contour of the dielectric. This approach is referred to as point collocation or point matching [4].

Substituting Equations 3.88-3.91 into Equation 3.85 and performing an inner product between ω_m yields a set of linear algebraic equations:

$$\begin{bmatrix} Z1_{n,m} & p_1 Y1_{n,m} \\ Z2_{n,m} & -p_2 Y2_{n,m} \end{bmatrix} \begin{bmatrix} H_m^{sc} \\ \Psi_m^{sc} \end{bmatrix} = \begin{bmatrix} H_m^{inc} \\ 0 \end{bmatrix}, \tag{3.96}$$

in which

$$Z2_{n,m} = \left(1 - \frac{\theta_n}{2\pi}\right)\delta_{nm} + \int_{-1}^{1} \left\{ \frac{\Delta\ell_n}{2}\phi_1(\xi)\frac{\partial G_2\left[r'(x_n, y_n), r_m'\right]}{\partial n} + \frac{\Delta\ell_{n-1}}{2}\phi_2(\xi)\frac{\partial G_2\left[r'(x_{n-1}, y_{n-1}), r_m'\right]}{\partial n} \right\} d\xi, \tag{3.97}$$

$$Y2_{n,m} = \int_{-1}^{1} \left\{ \frac{\Delta\ell_n}{2}\phi_1(\xi)G_2\left[r'(x_n, y_n), r_m'\right] + \frac{\Delta\ell_{n-1}}{2}\phi_2(\xi)G_2\left[r'(x_{n-1}, y_{n-1}), r_m'\right] \right\} d\xi, \tag{3.98}$$

$$Z1_{n,m} = \left(\frac{\theta_n}{2\pi}\right)\delta_{nm} - \int_{-1}^{1} \left\{ \frac{\Delta\ell_n}{2}\phi_1(\xi)\frac{\partial G_1\left[r'(x_n, y_n), r_m'\right]}{\partial n} + \frac{\Delta\ell_{n-1}}{2}\phi_2(\xi)\frac{\partial G_1\left[r'(x_{n-1}, y_{n-1}), r_m'\right]}{\partial n} \right\} d\xi, \text{ and} \tag{3.99}$$

$$Y1_{n,m} = \int_{-1}^{1} \left\{ \frac{\Delta\ell_n}{2}\phi_1(\xi)G_1\left[r'(x_n, y_n), r_m'\right] + \frac{\Delta\ell_{n-1}}{2}\phi_2(\xi)G_1\left[r'(x_{n-1}, y_{n-1}), r_m'\right] \right\} d\xi. \tag{3.100}$$

Also note that $d\xi = 2d\ell'/\Delta\ell_n$, $\Delta\ell_n$ is the length of the corresponding line segment n, and δ_{nm} is the Kronecker delta function defined by

$$\delta_{nm} \equiv \begin{cases} 1, & n = m \\ 0, & n \neq m \end{cases}. \tag{3.101}$$

The integral equations in Equations 3.97-3.100 specify a unique relationship between the electric field and its normal derivative at each sampled point on the boundary and every other point, which comprises the boundary-value problem. Once the solution to Equation 3.96 is

determined, Equations 3.86 and 3.87 (or their TM equivalents) may be used to calculate the scattered field at any observation point. This BEM formulism treats the cases of particular interest in this dissertation, finite aperture dielectric DOEs. In the next section, an example of a closed-contoured dielectric structure is presented.

3.2.8 Numerical example: Dielectric cylinder

An excellent method with which to test the accuracy of the user implementation of a BEM computer code is to solve a problem with a known solution. The analysis of a dielectric cylinder is such a case. It also falls into a very exclusive class of problems in which an analytic solution is known to exist.

So far, the development of BEM was restricted to the application of finite aperture diffractive optical elements. In the application of BEM to dielectrics with closed contours, however, the mathematical formulism is slightly different. However, the implementation of the point matching method can still be tested and assessed.

Consider a TE uniform plane wave traveling in the +x direction in free space that is incident normally on an infinite two-dimensional lossless dielectric cylinder of radius a as shown in Figure 3.5. The incident, scattered, and transmitted electric fields can be written as [4]

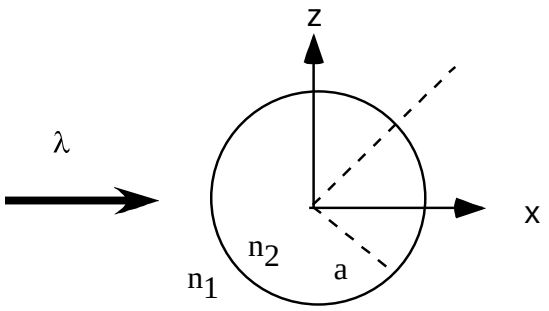

Figure 3.5 Dielectric cylinder geometry.

$$\vec{E}_{inc} = \hat{z} E_0 \sum_{n=-\infty}^{+\infty} j^{-n} J_n(k_1\rho) e^{jn\phi}, \tag{3.102}$$

$$\vec{E}_{scat} = \hat{z} E_o \sum_{n=-\infty}^{+\infty} a_n H_n^{(2)}(k_1\rho) e^{jn\phi}, \text{ and} \tag{3.103}$$

$$\vec{E}_{trans} = \hat{z} E_0 \sum_{n=-\infty}^{+\infty} [b_n J_n(k_2\rho) + c_n Y_n(k_2\rho)] e^{jn\phi} \tag{3.104}$$

in which

$$a_n = j^{-n} \frac{J_n'(k_1a) J_n(k_2a) - \sqrt{\varepsilon_r/\mu_r} J_n(k_1a) J_n'(k_2a)}{\sqrt{\varepsilon_r/\mu_r} J_n'(k_2a) H_n^{(2)}(k_1a) - J_n(k_2a) H_n^{(2)'}(k_1a)}, \tag{3.105}$$

$$b_n = j^{-n} \frac{J_n(k_1a) H_n^{(2)'}(k_1a) - J_n'(k_1a) H_n^{(2)}(k_1a)}{J_n(k_2a) H_n^{(2)'}(k_1a) - \sqrt{\varepsilon_r/\mu_r} J_n'(k_2a) H_n^{(2)}(k_1a)}, \tag{3.106}$$

$$c_n = 0, \tag{3.107}$$

and ρ is the radial distance from the origin and φ is the positive angle with respect to the +x axis. k_1 and k_2 are the wavenumbers in media 1 and 2, respectively. $H_n^{(2)}$ is the Hankel function of the second kind of the nth order. The Bessel functions, J_n and Y_n, are of the first and second kind, respectively, of order n. Also note that the primes (′) denote the total derivative with respect to the total arguments. Note that the ratio of permittivities of the two media are expressed as $\varepsilon_r = \frac{\varepsilon_2}{\varepsilon_1} = \frac{n_2^2}{n_1^2}$ and assuming a non-magnetic material, the relative permeability, μ_r, is set to unity.

For a TM uniform plane wave traveling in the +x direction in free space that is incident normally on the same lossless dielectric cylinder, the incident, scattered, and transmitted electric fields can be written as [4]

$$\vec{H}_{inc} = \hat{z} H_0 \sum_{n=-\infty}^{+\infty} j^{-n} J_n(k_1\rho) e^{jn\phi}, \tag{3.107}$$

$$\vec{H}_{scat} = \hat{z} H_o \sum_{n=-\infty}^{+\infty} a_n H_n^{(2)}(k_1\rho) e^{jn\phi}, \tag{3.108}$$

$$\vec{H}_{trans} = \hat{z}\, H_o \sum_{n=-\infty}^{+\infty} [b_n J_n(k_2\rho) + c_n Y_n(k_2\rho)] e^{jn\phi}\,, \tag{3.109}$$

$$a_n = j^{-n} \frac{J_n'(k_1a)J_n(k_2a) - \sqrt{\mu_r/\varepsilon_r}\, J_n(k_1a)J_n'(k_2a)}{\sqrt{\mu_r/\varepsilon_r}\, J_n'(k_2a)H_n^{(2)}(k_1a) - J_n(k_2a)H_n^{(2)\prime}(k_1a)}, \tag{3.110}$$

$$b_n = j^{-n} \frac{J_n(k_1a)H_n^{(2)\prime}(k_1a) - J_n'(k_1a)H_n^{(2)\prime}(k_1a)}{J_n(k_2a)H_n^{(2)\prime}(k_1a) - \sqrt{\mu_r/\varepsilon_r}\, J_n'(k_2a)H_n^{(2)}(k_1a)}\,, \text{ and} \tag{3.111}$$

$$c_n = 0\,. \tag{3.112}$$

In the numerical implementation of BEM for the example of a dielectric cylinder, the following equations hold for the closed geometry [3]:

$$\begin{bmatrix} Z2_{n,m} & -Y2_{n,m} \\ Z1_{n,m} & Y1_{n,m} \end{bmatrix} \begin{bmatrix} E_m^{sc} \\ \Psi_m^{sc} \end{bmatrix} = \begin{bmatrix} -Z2_{n,m} & Y2_{n,m} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} E_m^{inc} \\ \Psi_m^{inc} \end{bmatrix} \tag{3.113}$$

for TE case, and

$$\begin{bmatrix} Z2_{n,m} & -n_2^2 Y2_{n,m} \\ Z1_{n,m} & n_1^2 Y1_{n,m} \end{bmatrix} \begin{bmatrix} H_m^{inc} \\ \Psi_m^{inc} \end{bmatrix} = \begin{bmatrix} -Z2_{n,m} & n_2^2 Y2_{n,m} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} H_m^{inc} \\ \Psi_m^{inc} \end{bmatrix} \tag{3.114}$$

for TM case. The expressions for Z2, Y2, Z1, and Y1 are completely identical to those in Equations 3.97-3.100, respectively.

To assess the implementation of BEM, the scattered fields for both the TE and TM cases are determined on the surface of the dielectric cylinder, i.e., for $\rho=a$. Figure 3.6 shows the comparison of the scattered field amplitudes and phases calculated via BEM and analytical results for both the TE and TM cases for the following parameters, as an example, λ = 1.0 μm, a = 0.5 μm, n_1=1.0, and n_2=1.5.

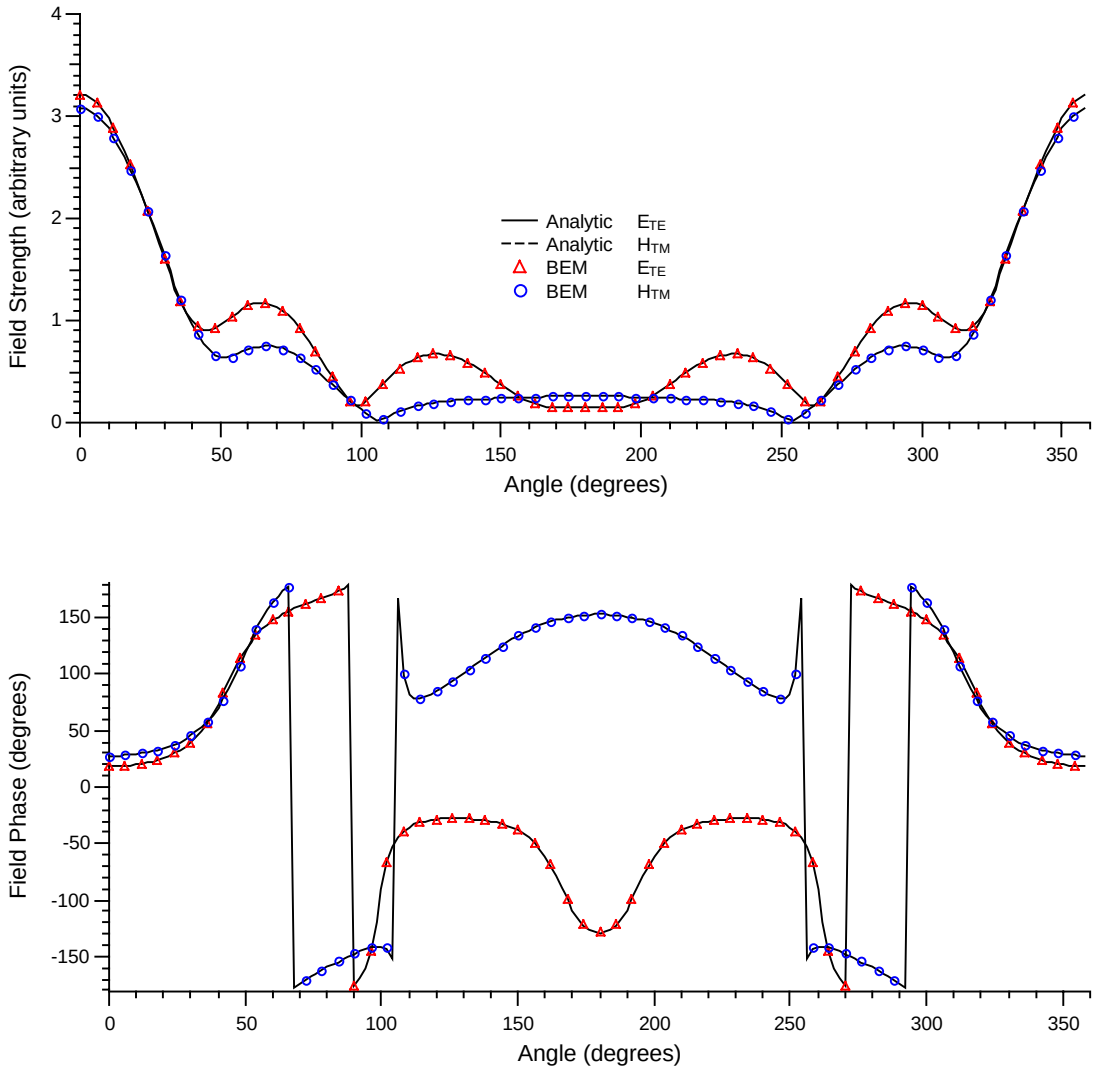

Figure 3.6 BEM analysis of a dielectric cylinder.

3.3 Further remarks on BEM

This chapter discussed the theoretical development of BEM used to rigorously assess designs of the finite aperture DOEs. One limitation of using a BEM is that it requires a large amount of computer memory to obtain results. In using BEM, it is necessary to calculate with matrices of sizes proportional to the square of the number of sample points taken along a dielectric boundary. In some applications, this might create too high a computational burden. Therefore, a more efficient method, the finite-difference time domain method (FDTD) in which the computational memory increases only linearly with the number of sample points taken in the computational lattice is presented in the next chapter. The boundary element is still very useful, however, in that it still gives accurate results and is an excellent tool in which to validate other rigorous methods.

Chapter 4

THE FINITE-DIFFERENCE TIME-DOMAIN METHOD

4.1 Introduction

The Finite-Difference Time-Domain method (FDTD) is perhaps the most popular numerical method for solving problems in electrodynamics. First proposed by K.S. Yee in 1966 [25], FDTD is an elegant and very straightforward way to represent the discretized version of the differential form of Maxwell's equations. An electric field grid offset both spatially and temporally with respect to a magnetic field grid is used to obtain updated equations that give field values throughout the entire computational grid in terms of previous grid field values. The equations are used in a leapfrog scheme to march the $\vec{E}$ and $\vec{H}$ fields incrementally forward in time. Despite the simplicity of Yee's algorithm, it did not originally receive much attention due to high computational cost and inherent limitations of the method at that time (e.g., the inability to treat boundary conditions). However as time progressed, the computational cost decreased dramatically, in terms of both computer speed and memory. Also, the original inherent shortcomings of FDTD were alleviated with developments in the treatment of boundary conditions. In recent years, this led to much greater interest in FDTD.

Section 4.2 presents the implementation of FDTD, specifically the Yee time marching algorithm, in Section 4.2.1 and the TE and TM field implementation in Section 4.2.2. Finite difference expressions for Maxwell's equations in two dimensions are discussed in Section 4.2.3. Then the issues of numerical stability and dispersion are briefly addressed in Section 4.2.4.

Sections 4.2.5 through 4.2.7 give the derivation of the type of absorbing boundary conditions (ABCs) used to truncate the FDTD computational grid, namely Mur 2nd order ABCs. The Total-field/scattered-field formulation is presented in Section 4.2.8. Problem symmetry is addressed in Section 4.2.9. A technique, relative permittivity spatial averaging, used to average the relative permittivity near a discontinuity, is presented in Section 4.2.10. An example problem of a dielectric cylinder, which was used to test the FDTD implementation, is discussed in Section 4.2.11. The choice of field propagation to a plane of interest outside the computational grid is discussed in Section 4.2.12. Finally, in Section 4.2.13, an overview of the software capabilities developed for FDTD analysis is presented.

4.2 FDTD implementation

4.2.1 Yee time-marching algorithm

The implementation of FDTD involves Maxwell's curl equations, i.e., Faraday's Law and Ampere's Law. In this chapter, specific attention is given to the two-dimensional implementation of FDTD since all of the analyses in this and later chapters are two-dimensional in nature. Maxwell's curl equations take the general form,

$$\nabla \times \vec{E} = -\mu \frac{\partial \vec{H}}{\partial t} \qquad \text{(Faraday's law) and} \tag{4.1}$$

$$\nabla \times \vec{H} = \vec{J} + \mu \frac{\partial (\varepsilon \vec{E})}{\partial t} \qquad \text{(Ampere's law),} \tag{4.2}$$

in which $\vec{E}$ and $\vec{H}$ are the electric field having units of Volts/meter and the magnetic field having units of Amperes per meter, respectively. The current density is denoted as $\vec{J}$ (units of Amperes per meter squared), ε is the permittivity of the medium, and μ is the permeability of the medium. If there are no sources of current, i.e., J=0, then Ampere's law becomes

$$\nabla \times \vec{H} = \mu \frac{\partial (\varepsilon \vec{E})}{\partial t}. \tag{4.3}$$

The basic idea of the Yee algorithm is to solve for both the electric and magnetic fields in time and space using the coupled Maxwell's curl equations rather than solving for each individually with a wave equation. By using both the electric and magnetic field information, the solution is more robust in the sense that it is more accurate for a wider class of structures since both electric and magnetic materials can be modeled in a straightforward fashion. The next section presents the specific case of Maxwell's equations in two dimensions.

4.2.2 Maxwell's equations in two dimensions (TE and TM cases)

In solving the Maxwell's curl equation in two dimensions, it is possible to divide the problem into two separate ones by solving for two eigenmodes of the problem, specifically the transverse electric (TE) and transverse magnetic (TM) cases separately. The TE case is defined for an electric field perpendicular to the two-dimensional plane while the magnetic field components lie within the plane as shown in Figure 4.1. In contrast, the TM case is defined for a magnetic field perpendicular to the two-dimensional plane while the electric field components lie within the plane as shown in Figure 4.2.

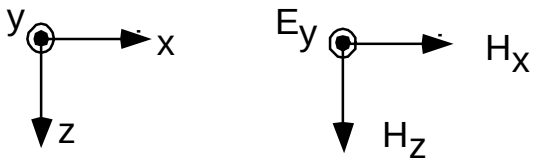

Figure 4.1 FDTD TE geometry.

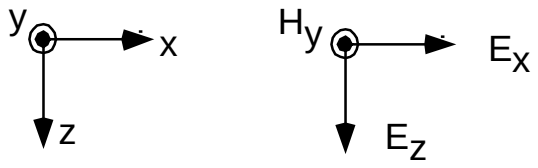

Figure 4.2 FDTD TM geometry.

First consider the TE case in which

$$\begin{aligned}\nabla\times\vec{E} &= \hat{z}\frac{\partial E_y}{\partial x}-\hat{x}\frac{\partial E_y}{\partial z}\\ &= -\mu\left[\hat{x}\frac{\partial H_x}{\partial t}-\hat{z}\frac{\partial H_z}{\partial t}\right]\end{aligned} \tag{4.4}$$

and

$$\begin{aligned}\nabla\times\vec{H} &= \hat{y}\left[\frac{\partial H_x}{\partial z}-\frac{\partial H_z}{\partial x}\right]\\ &= \hat{y}\varepsilon\frac{\partial E_y}{\partial t}.\end{aligned} \tag{4.5}$$

Note that the speed of light and the impedance in free space may written as

$$c=\frac{1}{\sqrt{\mu\varepsilon_o}} \tag{4.6}$$

and

$$\eta_o=\sqrt{\frac{\mu}{\varepsilon_o}}, \tag{4.7}$$

respectively. Note that the permittivity may be written as $\varepsilon=n^2\varepsilon_o$ in which n is the refractive index of the medium and ε_0 is the permittivity of free space and is equal to $8.8514\ldots\times10^{-12}$ Farads/meter and μ is the permeability has the value of $4\pi\times10^{-7}$ Henries/meter. In

all cases considered, all materials are assumed non-magnetic and the therefore the value of μ is assumed constant in all circumstances presented. The values for the speed of light and the impedance of free space are $2.9979\ldots\times10^{8}$ meters/sec and 376.7 Ω, respectively. The permeability and relative permittivity may now alternatively be expressed as

$$\mu=\frac{\eta_o}{c} \quad \text{and} \tag{4.8}$$

$$\varepsilon=\frac{n^2}{c\eta_o}. \tag{4.9}$$

Now the time derivatives of all the TE field components can be expressed as

$$\frac{\partial E_y}{\partial t}=\frac{c\eta_o}{n^2}\left(\frac{\partial H_x}{\partial z}-\frac{\partial H_z}{\partial x}\right), \tag{4.10}$$

$$\frac{\partial H_x}{\partial t}=\frac{c}{\eta_o}\frac{\partial E_y}{\partial z}, \text{ and} \tag{4.11}$$

$$\frac{\partial H_z}{\partial t}=-\frac{c}{\eta_o}\frac{\partial E_y}{\partial x}. \tag{4.12}$$

The TM case is derived in a similar fashion. The time derivatives of the field components are then

$$\frac{\partial H_y}{\partial t}=-\frac{c}{\eta_o}\left(\frac{\partial E_x}{\partial z}-\frac{\partial E_z}{\partial x}\right), \tag{4.13}$$

$$\frac{\partial E_x}{\partial t}=-\frac{c\eta_o}{n^2}\frac{\partial H_y}{\partial z}, \text{ and} \tag{4.14}$$

$$\frac{\partial E_z}{\partial t}=\frac{c\eta_o}{n^2}\frac{\partial H_y}{\partial x}. \tag{4.15}$$

Equations 4.10 through 4.15 represent the field relations in analytic form for both the TE and TM cases. Implementing FDTD requires discretizing Maxwell's curl equations so that

practical problems may be solved numerically. The use of finite differences makes this possible and is presented in the following section.

4.2.3 Finite difference expressions for Maxwell's equations in two dimensions

In a discretized uniform rectangular lattice, a space point is denoted as (i,j,k) = (iΔx, jΔy, kΔz), in which Δx, Δy, and Δz are the space lattice increments in the x, y, and z coordinate directions, respectively, and i, j, and k are integers. A function u of space and time may be evaluated at a discrete point in space and at a discrete point in time as u(iΔx, jΔy, kΔz, nΔt) = $u^{n}_{i,j,k}$ in which Δt is the time increment which is assumed uniform over the observation interval and n is an integer.

Using central-difference or centered finite-difference expressions for the space and time derivatives that are accurate to second-order in both space and time increments, the partial derivative of u in the x-direction (keeping t constant) can be expressed as [26]

$$\frac{\partial u}{\partial x}(i\Delta x, j\Delta y, k\Delta z, n\Delta t) = \frac{u^{n}_{i+1/2,j,k} - u^{n}_{i-1/2,j,k}}{\Delta x} + O[(\Delta x)^2]. \tag{4.16}$$

Note the ±1/2 increment in the x-coordinate of u denoted by the i subscript giving a finite difference over ±1/2 Δx. Similarly, the first time partial derivative of u at a fixed point in space can be expressed as

$$\frac{\partial u}{\partial t}(i\Delta x, j\Delta y, k\Delta z, n\Delta t) = \frac{u^{n+1/2}_{i,j,k} - u^{n-1/2}_{i,j,k}}{\Delta t} + O[(\Delta t)^2]. \tag{4.17}$$

Note the ±1/2 increment in the time coordinate of u denoted by the n superscript yielding a finite difference over ±1/2 Δt. In his original paper, Yee chose this notation because he wished to interweave the E and H components in time at interval of 1/2 Δt to implement a leapfrog algorithm. Note for two-dimensional problems, one direction may be ignored (e.g., the y-direction) and one may drop the k subscript and reassign the i and j subscripts to the x and z directions, respectively.

Consider the expressions for the TE case. Using finite differences, the discretized version of these equations may expressed as

$$\frac{E_y\big|_{i,j}^{n+1}-E_y\big|_{i,j}^{n}}{\Delta t}=\frac{c\eta_o}{n^2}\left(\frac{H_x\big|_{i,j+1/2}^{n+1/2}-H_x\big|_{i,j-1/2}^{n+1/2}}{\Delta z}-\frac{H_z\big|_{i+1/2,j}^{n+1/2}-H_z\big|_{i-1/2,j}^{n+1/2}}{\Delta x}\right), \tag{4.18}$$

$$\frac{H_x\big|_{i,j}^{n+1/2}-H_x\big|_{i,j}^{n-1/2}}{\Delta t}=\frac{c}{\eta_o}\left(\frac{E_y\big|_{i,j+1/2}^{n}-E_y\big|_{i,j-1/2}^{n}}{\Delta z}\right), \text{ and} \tag{4.19}$$

$$\frac{H_z\big|_{i,j}^{n+1/2}-H_z\big|_{i,j}^{n-1/2}}{\Delta t}=-\frac{c}{\eta_o}\left(\frac{E_y\big|_{i+1/2,j}^{n}-E_y\big|_{i-1/2i,j}^{n}}{\Delta x}\right). \tag{4.20}$$

Similarly for the TM case,

$$\frac{H_y\big|_{i,j}^{n+1}-H_y\big|_{i,j}^{n}}{\Delta t}=-\frac{c}{\eta_o}\left(\frac{E_x\big|_{i,j+1/2}^{n+1/2}-E_x\big|_{i,j-1/2}^{n+1/2}}{\Delta z}-\frac{E_z\big|_{i+1/2,j}^{n+1/2}-E_z\big|_{i-1/2,j}^{n+1/2}}{\Delta x}\right), \tag{4.21}$$

$$\frac{E_x\big|_{i,j}^{n+1/2}-E_x\big|_{i,j}^{n-1/2}}{\Delta t}=-\frac{c\eta_o}{n^2}\left(\frac{H_y\big|_{i,j+1/2}^{n}-H_y\big|_{i,j-1/2}^{n}}{\Delta z}\right), \text{ and} \tag{4.22}$$

$$\frac{E_z\big|_{i,j}^{n+1/2}-E_z\big|_{i,j}^{n-1/2}}{\Delta t}=\frac{c\eta_o}{n^2}\left(\frac{H_y\big|_{i+1/2,j}^{n}-H_y\big|_{i-1/2i,j}^{n}}{\Delta x}\right). \tag{4.23}$$

As shown in Figures 4.3 and 4.4, the basic Yee cell is constructed such that the electric and magnetic fields are coupled spatially for either the TE or TM case, respectively. The Yee algorithm also centers the electric and magnetic fields in time in what is called a leapfrog arrangement. With this arrangement, all of the electric field computations at the points of interest are completed and stored in computer memory at a particular point in time using the previously stored magnetic field data stored in computer memory. Then all the magnetic field computations are made based on the electric field data just computed. This process continues until the time-marching process is concluded. Note that the Yee cells shown above are just one cell of a single lattice structure. Figure 4.5 shows how the Yee cell geometry may appear for a three-dimensional application. Figure 4.6 shows an example of what a total lattice structure may look

like in a practical two-dimensional application such as a dielectric interface between two media having refractive indices of n_1 and n_2. In the following section, the relation between the time stepping interval and the spacing of field components in the computational Yee cell grid, which allows the algorithm to converge, is discussed.

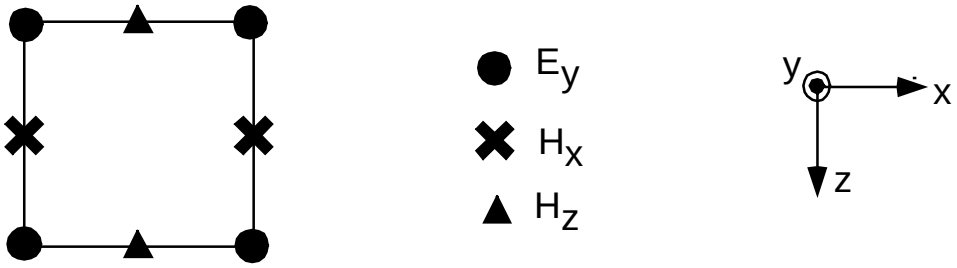

Figure 4.3 Yee cell geometry for TE case.

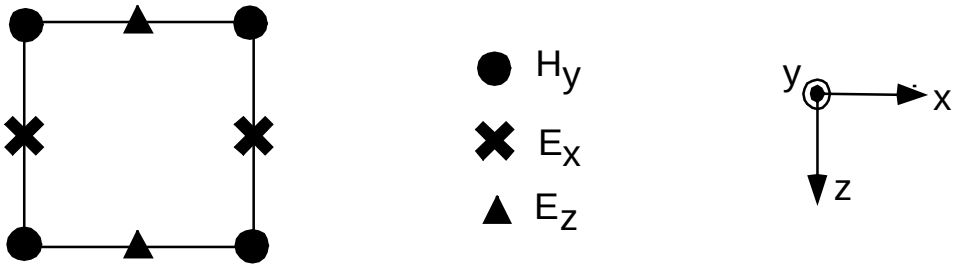

Figure 4.4 Yee cell geometry for TM case.

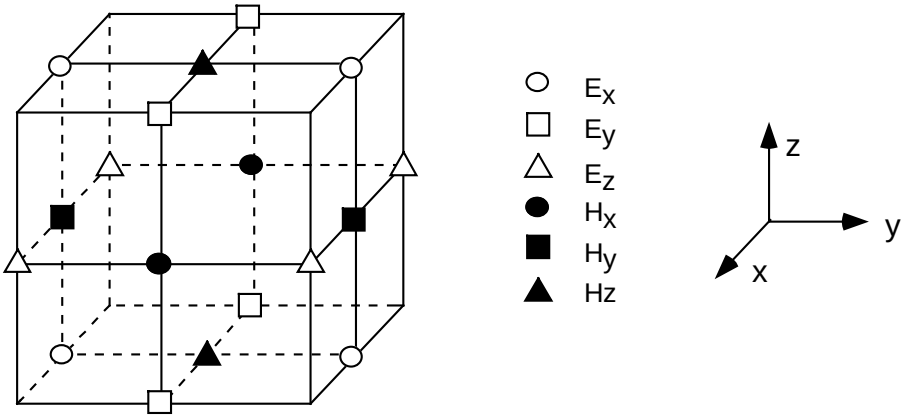

Figure 4.5 Three-dimensional Yee cell geometry.

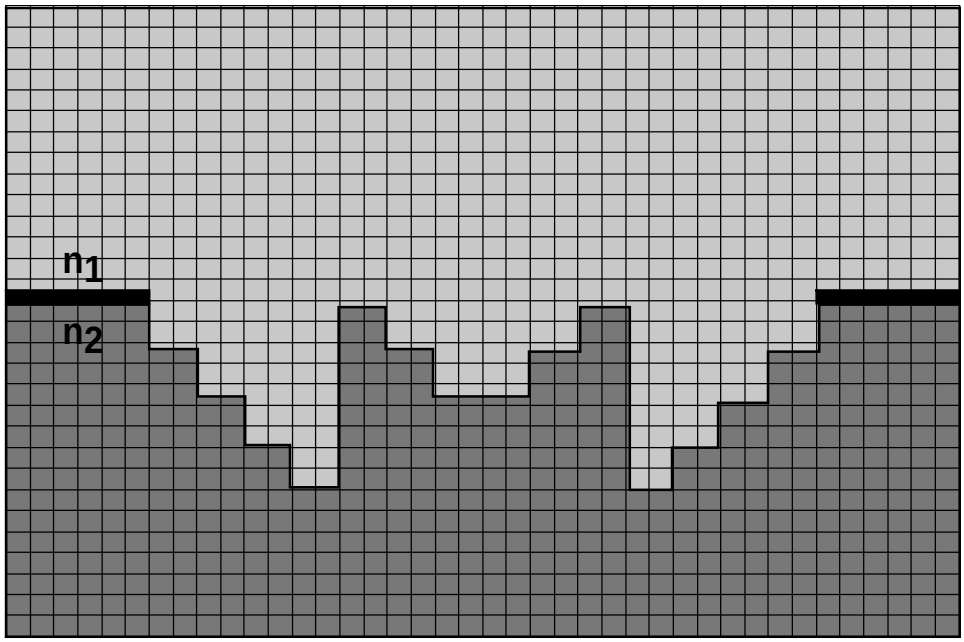

Figure 4.6 Example Yee cell lattice structure.

4.2.4 Numerical stability and dispersion

The numerical implementation of FDTD requires that the time increment, Δt, be bound by the size of the spatial increments, Δx, Δy, and Δz. The constraint is necessary to avoid the effects of numerical instability [27]. In the execution of the FDTD algorithm, this numerical instability often causes the total energy within the FDTD computational grid to diverge toward infinity, which is physically unrealizable. The domain of convergence in which the basic Yee-algorithm is numerically stable is as follows:

$$\frac{c\Delta t}{\Delta x} \leq 1 \qquad \text{(1-D case),} \qquad (4.24)$$

$$c\Delta t \cdot \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta z)^2}} \leq 1 \qquad \text{(2-D case), and} \qquad (4.25)$$

$$c\Delta t \cdot \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}} \leq 1 \qquad \text{(3-D case).} \qquad (4.26)$$

Other factors besides the basic Yee algorithm may, in general, affect the entire FDTD procedure. Such factors include boundary conditions, variable and unstructured meshing, and lossy, dispersive, nonlinear, and gain materials [5].

The numerical implementation of FDTD also causes some degree of dispersion of the simulated wave modes in the computational lattice. This effect is referred to as numerical dispersion [28]. The phase velocity of the wave modes in the FDTD grid differs from the true speed of light in the medium in which it propagates. In FDTD, the speed of light can, in fact, vary with the modal wavelength, the direction of propagation in the grid, and with the size of the Yee grid cells. The FDTD grid, in essence, behaves like a “numerical aether” [5] such that an electromagnetic wave interacting within that structure will generally propagate with a slower velocity. Consequently, the propagating wave will accumulate phase errors that can possibly lead

to waveform broadening, spurious anisotropy, and pseudorefraction. Generally, the effect of numerical dispersion should be accounted for in the implementation of the FDTD procedure, especially for large electrical structures by using dispersion-optimized Yee algorithms [5]. In principle, if the computational lattice is discretized by more Yee cells (of smaller size), the effects of numerical dispersion become less significant. In all example applications investigated in this dissertation, the effects of numerical dispersion were negligible. This was determined by noting that further refinement of the computational lattice having no noticeable change in simulated results.

4.2.5 Absorbing boundary conditions

An important consideration in the implementation of FDTD is the truncation of the Yee computational lattice. Of course, no computer can store an unlimited amount of data. This point necessitates the use of absorbing boundary conditions (ABCs) so that impinging scattered fields on the outermost lattice planes are absorbed as shown in Figure 4.7.

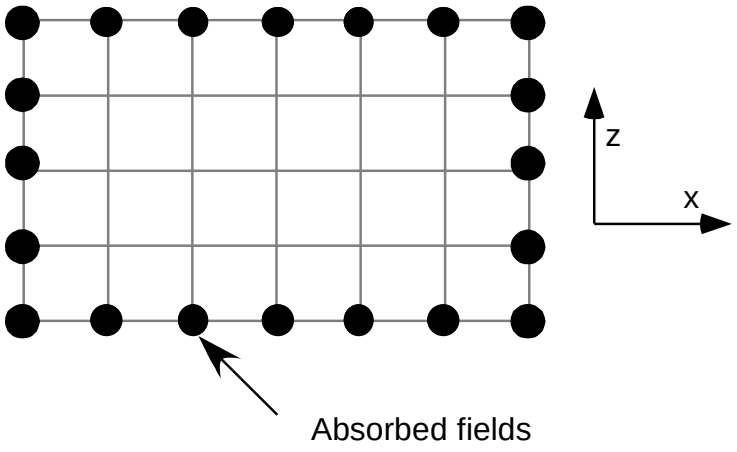

Figure 4.7 Absorbing boundary conditions at outermost points.

The boundary conditions cannot be directly obtained from the numerical implementation of Maxwell's curl equations defined in terms of finite differences. The reason is that FDTD utilizes a central-differencing scheme that requires knowledge of the fields one-half Yee cell from each side of a given point. Therefore, central-differences cannot be used at the outermost lattice planes since the fields are not known outside the lattice and the ABCs must be derived by some other means. One method of implementing ABCs is discussed next.

4.2.6 Engquist-Madja one-way wave equations

Engquist and Madja [29] developed a method for implementing ABCs in a Cartesian FDTD grid. The premise of the theory is that partial differential equations that permit field propagation in only certain directions, called one-way wave equations, can be employed at outermost lattice planes. With this implementation, fields are absorbed at the outermost planes.

For example, consider the two-dimensional wave equation

$$\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial z^2}-\frac{1}{v^2}\frac{\partial^2 U}{\partial t^2}=0\,, \tag{4.27}$$

in which U is a field component and v is the phase velocity given by v=c/n. A partial differential operator, L, is defined as

$$L\equiv\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial z^2}-\frac{1}{v^2}\frac{\partial^2}{\partial t^2}\equiv D_x^2+D_z^2-\frac{1}{v^2}D_t^2\,, \tag{4.28}$$

such that the wave equation is compactly expressed as

$$LU=0\,. \tag{4.29}$$

Engquist and Madja showed that the wave operator can be factored as

$$LU=L^+L^-U=0\,, \tag{4.30}$$

in which $L^{\pm}$ is defined as

$$L^{\pm}\equiv D_x-\frac{D_t}{v}\sqrt{1-S^2}\quad\text{with} \tag{4.31}$$

$$S \equiv \frac{D_z}{(D_t / v)}. \tag{4.32}$$

Engquist and Madja showed that the application of $L^{\pm}$ to the wave function U at the right or left grid boundary, respectively, completely absorbs a wave propagating toward the boundary at any angle. Therefore,

$$L^{\pm} U = 0, \tag{4.33}$$

applied at the right and left boundary planes, functions as an exact analytic ABC for waves originating from the interior of the FDTD grid.

The presence of the square-root operator in Equation 4.31 prohibits direct numerical implementation of Equation 4.33 as an ABC. However, the square-root function can be approximated using a Taylor series expansion. Consider the first two terms of the Taylor series expansion

$$\sqrt{1 - S^2} \cong 1 - \frac{1}{2} S^2. \tag{4.34}$$

(Note: it is these two terms in the expansion that give rise to the name Mur 2nd order boundary conditions discussed in the next section.)

Then substituting Equation 4.34 into Equation 4.33 yields

$$L^{\pm} U = D_x D_t \pm \left(\frac{1}{v} D_t^2 - \frac{v}{2} D_z^2 \right) U = 0, \tag{4.35}$$

or alternatively,

$$\left[\frac{\partial^2}{\partial x \partial t} \pm \left(\frac{1}{v} \frac{\partial^2}{\partial t^2} - \frac{v}{2} \frac{\partial^2}{\partial z^2} \right) \right] U = 0. \tag{4.36}$$

Therefore, at the appropriate outermost lattice planes, the approximate analytical ABCs may be expressed for each boundary:

Left boundary:

$$\frac{\partial^2 U}{\partial x \partial t} - \frac{1}{v} \frac{\partial^2 U}{\partial t^2} + \frac{v}{2} \frac{\partial^2 U}{\partial z^2} = 0 \tag{4.37}$$

Right boundary:

$$\frac{\partial^2 U}{\partial x \partial t} + \frac{1}{v}\frac{\partial^2 U}{\partial t^2} - \frac{v}{2}\frac{\partial^2 U}{\partial z^2} = 0 \tag{4.38}$$

Bottom boundary:

$$\frac{\partial^2 U}{\partial z \partial t} - \frac{1}{v}\frac{\partial^2 U}{\partial t^2} + \frac{v}{2}\frac{\partial^2 U}{\partial x^2} = 0 \tag{4.39}$$

Top boundary:

$$\frac{\partial^2 U}{\partial z \partial t} + \frac{1}{v}\frac{\partial^2 U}{\partial t^2} - \frac{v}{2}\frac{\partial^2 U}{\partial x^2} = 0\,. \tag{4.40}$$

So far, the approximate analytic expressions for the ABCs have been derived. The next step is to implement the ABCs suitably in a numerical scheme. The finite-difference scheme reported by Mur is discussed next.

4.2.7 Mur 2nd order boundary conditions

One simple and successful scheme to implement ABCs was introduced by Mur [30]. For simplicity, the scheme is illustrated at the leftmost boundary of the FDTD grid. Let $W_{0,j}^{n}$ represent a Cartesian component of either the electric or magnetic field located in the leftmost plane of the FDTD grid tangential to this boundary. Mur implemented the partials of Equation 4.36 as central differences expanded about some auxiliary grid point (1/2,j). The mixed partial derivative with respect to x and t is written as

$$\begin{aligned}\left.\frac{\partial^2 W}{\partial x \partial t}\right|_{1/2,j}^{n} &= \frac{1}{2\Delta t}\left(\left.\frac{\partial W}{\partial x}\right|_{1/2,j}^{n+1} - \left.\frac{\partial W}{\partial x}\right|_{1/2,j}^{n}\right) \\ &= \frac{1}{2\Delta t}\left(\frac{W|_{1,j}^{n+1} - W_{0,j}^{n+1}}{\Delta x} - \frac{W|_{1,j}^{n-1} - W_{0,j}^{n-1}}{\Delta x}\right).\end{aligned} \tag{4.41}$$

The second time derivative is expressed as the average of the second time derivatives at adjacent points (0,j) and (1,j):

$$\left.\frac{\partial^2 W}{\partial t^t}\right|_{1/2,j}^{n} = \frac{1}{2}\left(\left.\frac{\partial^2 W}{\partial t^2}\right|_{0,j}^{n} - \left.\frac{\partial^2 W}{\partial t^2}\right|_{1,j}^{n} \right) = \frac{1}{2}\left(\frac{W|_{0,j}^{n+1} - 2W_{0,j}^{n} + W_{0,j}^{n-1}}{(\Delta t)^2} - \frac{W|_{1,j}^{n+1} - 2W_{1,j}^{n} + W_{1,j}^{n-1}}{(\Delta t)^2} \right). \tag{4.42}$$

The second spatial derivative is expressed as the average of the second spatial derivatives at adjacent points (0,j) and (1,j):

$$\left.\frac{\partial^2 W}{\partial x^t}\right|_{1/2,j}^{n} = \frac{1}{2}\left(\left.\frac{\partial^2 W}{\partial x^2}\right|_{0,j}^{n} - \left.\frac{\partial^2 W}{\partial x^2}\right|_{1,j}^{n} \right) = \frac{1}{2}\left(\frac{W|_{0,j+1}^{n} - 2W_{0,j}^{n} + W_{0,j-1}^{n}}{(\Delta x)^2} - \frac{W|_{1,j+1}^{n} - 2W_{1,j}^{n} + W_{1,j-1}^{n}}{(\Delta x)^2} \right). \tag{4.43}$$

Substituting Equations 4.41, 4.42, and 4.43 into Equation 4.37 yields

$$\begin{aligned} W|_{0,j}^{n+1} = &- W|_{0,j}^{n} + \left(\frac{v\Delta t - \Delta x}{v\Delta t + \Delta x} \right)\left(W|_{1,j}^{n+1} + W|_{0,j}^{n-1} \right) \\ &+ \left(\frac{2\Delta x}{v\Delta t + \Delta x} \right)\left(W|_{0,j}^{n} + W|_{1,j}^{n} \right) \\ &+ \left(\frac{(c\Delta t)^2 \Delta x}{2(\Delta z)^2 (v\Delta t + \Delta x)} \right)\left(W|_{0,j+1}^{n} - 2W|_{0,j}^{n} + W|_{0,j-1}^{n} \right. \\ &\qquad \left. + W|_{1,j+1}^{n} - 2W|_{1,j}^{n} + W|_{1,j-1}^{n} \right). \end{aligned} \tag{4.44}$$

Similar expressions may be derived for the other three boundary planes and Equation 4.44 is applicable for either the TE or TM cases. More sophisticated methods exist for implementing ABCs; however, Mur 2nd order boundary conditions are perhaps the simplest. Since they are accurate and simple, Mur 2nd order boundary conditions are used in every FDTD application presented in this dissertation.

Perhaps the most accurate ABCs to implement are Perfectly Matched Layer (PML) ABCs introduced by J.P. Berenger in 1994 [31]. The novelty of Berenger's PML is that impinging

waves of arbitrary incidence, polarization, and frequency are matched at the boundary. A hypothetical material that is both electrically and magnetically lossy surrounds the usual FDTD computational grid. The basic idea is that the lossy material absorbs scattered radiation originating in the usual FDTD grid. The use of PML ABCs is reportedly more accurate than Mur 2nd order boundary conditions [32]; however, the implementation of the former is more complex.

4.2.8 Total-field/scattered-field formulation

There is yet another fundamental point in implementing FDTD to solve electromagnetic problems. In this section, a method in which to introduce electromagnetic field excitations into the FDTD lattice is presented.

In Yee's original presentation of FDTD [25], the incident wave was introduced in the lattice as an initial condition. However, this approach had fundamental problems, which severely inhibited its usefulness [5]. A hard source approach is based upon assigning a desired time dependence to the electric or magnetic field components in the FDTD lattice to simulate an incident field [5]. This approach is problematic as well. At those points in which the total electric or magnetic fields are defined, there exists no mechanism to allow for any reflection of waves from material surfaces. This flaw with the hard source implementation introduces spurious retroreflections in the FDTD lattice. Therefore, the hard source approach is not the most effective means to implement an incident field.

The total-field/scattered-field formulation [30,33] is perhaps the most effective method of introducing an incident field into a computational lattice. This approach is based on the linear nature of Maxwell's equations and the decomposition of the fields as

$$\vec{E}_{total} = \vec{E}_{inc} + \vec{E}_{scat} \qquad \text{and} \tag{4.45}$$

$$\vec{H}_{total} = \vec{H}_{inc} + \vec{H}_{scat}\,, \tag{4.46}$$

in which the total fields are the superposition of the incident and scattered fields.

Consider a one-dimensional lattice, as shown in Figure 4.8. To effectively introduce an incident field and allow for any scattering, the FDTD grid is partitioned into different zones: the total-field region and the scattered field region.

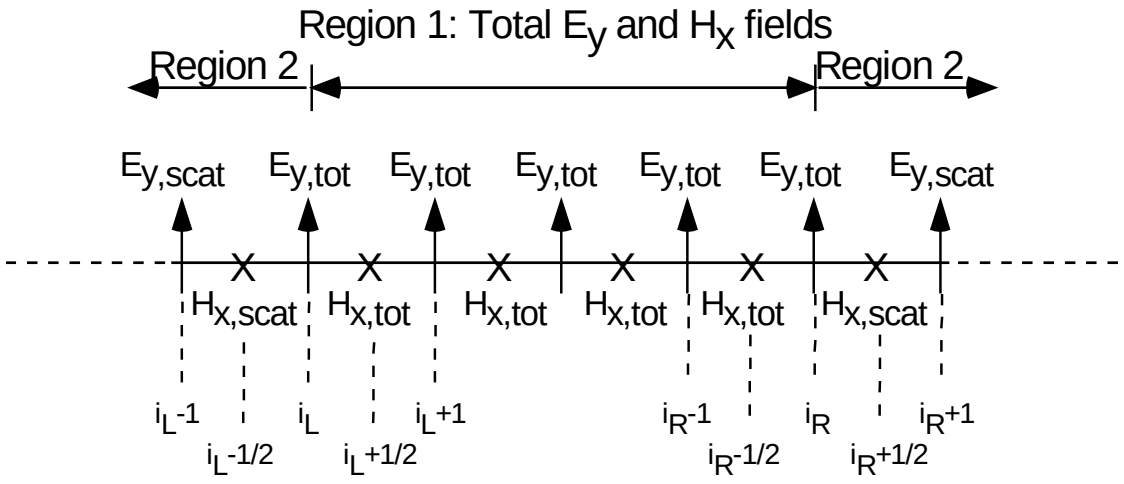

Figure 4.8 Total-field/scattered-field formulation in one dimension.

For the TE-one dimensional case, the connecting conditions for the left side of the grid are

$$E_{y,tot}\Big|_{i_L}^{n+1} = E_{y,tot}\Big|_{i_L}^{n} + \frac{c\Delta t}{n^2\Delta z}\eta_0\left(H_{x,tot}\Big|_{i_L+1/2}^{n+1/2} - H_{x,scat}\Big|_{i_L-1/2}^{n+1/2} - H_{x,inc}\Big|_{i_L-1/2}^{n+1/2}\right) \tag{4.47}$$

since

$$H_{x,tot}\Big|_{i_L-1/2}^{n+1/2} = H_{x,inc}\Big|_{i_L-1/2}^{n+1/2} + H_{x,scat}\Big|_{i_L-1/2}^{n+1/2}. \tag{4.48}$$

Also,

$$H_{x,scat}\Big|_{i_L-1/2}^{n+1/2} = H_{x,scat}\Big|_{i_L-1/2}^{n-1/2} + \frac{c\Delta t}{\eta_0 \Delta z}\left(E_{y,tot}\Big|_{i_L}^{n} - E_{y,scat}\Big|_{i_L-1}^{n} - E_{y,inc}\Big|_{i_L}^{n}\right) \tag{4.49}$$

since $E_{y,tot}\Big|_{i_L}^{n} = E_{y,inc}\Big|_{i_L}^{n} + E_{y,scat}\Big|_{i_L}^{n}$.

Similarly, for the right hand side of the grid the connecting conditions are

$$E_{y,tot}\Big|_{i_R}^{n+1} = E_{y,tot}\Big|_{i_R}^{n} + \frac{c\Delta t}{n^2 \Delta z}\eta_0\left(H_{x,scat}\Big|_{i_R+1/2}^{n+1/2} - H_{x,tot}\Big|_{i_R-1/2}^{n+1/2} + H_{x,inc}\Big|_{i_R+1/2}^{n+1/2}\right) \tag{4.50}$$

and

$$H_{x,scat}\Big|_{i_R+1/2}^{n+1/2} = H_{x,scat}\Big|_{i_R+1/2}^{n-1/2} + \frac{c\Delta t}{\eta_0 \Delta z}\left(E_{y,scat}\Big|_{i_R+1}^{n} - E_{y,tot}\Big|_{i_R}^{n} - E_{y,inc}\Big|_{i_R}^{n}\right). \tag{4.51}$$

For the TM case on the left hand side of the grid

$$H_{y,tot}\Big|_{i_L}^{n+1} = H_{y,tot}\Big|_{i_L}^{n} - \frac{c\Delta t}{\eta_0 \Delta z}\left(E_{x,tot}\Big|_{i_L+1/2}^{n+1/2} - E_{x,scat}\Big|_{i_L-1/2}^{n+1/2} - E_{x,inc}\Big|_{i_L-1/2}^{n+1/2}\right) \quad \text{and} \tag{4.52}$$

$$E_{x,scat}\Big|_{i_L-1/2}^{n+1/2} = E_{x,scat}\Big|_{i_L-1/2}^{n-1/2} + \frac{c\Delta t}{n^2 \Delta z}\eta_0\left(H_{y,tot}\Big|_{i_L}^{n} - H_{y,scat}\Big|_{i_L-1}^{n} - H_{y,inc}\Big|_{i_L}^{n}\right) \tag{4.53}$$

and for the right hand side of the grid

$$H_{y,tot}\Big|_{i_R}^{n+1} = H_{y,tot}\Big|_{i_R}^{n} - \frac{c\Delta t}{\eta_0 \Delta z}\left(E_{x,scat}\Big|_{i_R+1/2}^{n+1/2} - E_{x,tor}\Big|_{i_R-1/2}^{n+1/2} + E_{x,inc}\Big|_{i_R+1/2}^{n+1/2}\right) \quad \text{and} \tag{4.54}$$

$$E_{x,scat}\Big|_{i_R+1/2}^{n+1/2} = E_{x,scat}\Big|_{i_R+1/2}^{n-1/2} + \frac{c\Delta t}{n^2 \Delta z}\eta_0\left(H_{y,scat}\Big|_{i_R+1}^{n} - H_{y,tot}\Big|_{i_R}^{n} + H_{y,inc}\Big|_{i_R}^{n}\right). \tag{4.55}$$

In two dimensions, the connecting conditions follow in a similar fashion. The equations for E_y and H_x along the x-direction (connecting the left and right sides) are the same as in the one-dimensional case. However, it is necessary to consider how the E_y and H_z are connected at the top and bottom sides along the z-direction. Figure 4.9 shows the implementation for the TE case. The TM case would follow analogously (by replacing E components with H components and vice versa).

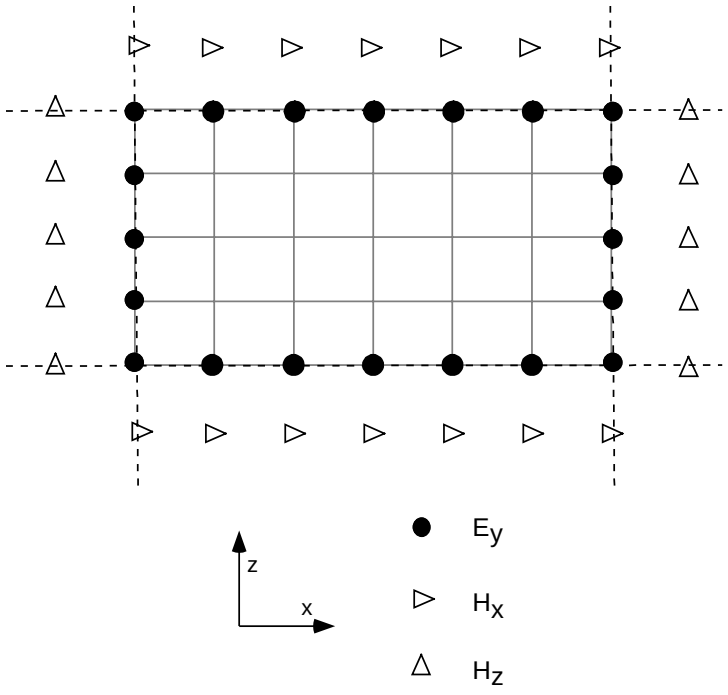

Figure 4.9 Total-field/scattered-field implementation in two dimensions.

For the TE case, the connecting conditions for the bottom side of the FDTD grid are

$$E_{y,tot}\Big|_{j_B}^{n+1} = E_{y,tot}\Big|_{j_B}^{n} + \frac{c\Delta t}{n^2\Delta x}\eta_0\left(H_{z,tot}\Big|_{j_B+1/2}^{n+1/2} - H_{z,scat}\Big|_{j_B-1/2}^{n+1/2} - H_{z,inc}\Big|_{j_B-1/2}^{n+1/2}\right) \text{ and} \qquad (4.56)$$

$$H_{z,scat}\Big|_{j_B-1/2}^{n+1/2} = H_{z,scat}\Big|_{j_B-1/2}^{n-1/2} + \frac{c\Delta t}{\eta_0\Delta x}\left(E_{y,tot}\Big|_{j_B}^{n} - E_{y,scat}\Big|_{j_B-1}^{n} - E_{y,inc}\Big|_{j_L}^{n}\right). \qquad (4.57)$$

For the top side of the grid

$$E_{y,tot}\Big|_{j_T}^{n+1} = E_{y,tot}\Big|_{j_T}^{n} + \frac{c\Delta t}{n^2\Delta x}\eta_0\left(H_{z,scat}\Big|_{j_T+1/2}^{n+1/2} - H_{z,tot}\Big|_{j_T-1/2}^{n+1/2} + H_{z,inc}\Big|_{j_T+1/2}^{n+1/2}\right) \text{ and} \qquad (4.58)$$

$$H_{z,scat}\Big|_{j_T+1/2}^{n+1/2} = H_{z,scat}\Big|_{j_T+1/2}^{n-1/2} + \frac{c\Delta t}{\eta_0\Delta x}\left(E_{y,scat}\Big|_{j_t+1}^{n} - E_{y,tot}\Big|_{j_T}^{n} - E_{y,inc}\Big|_{j_T}^{n}\right). \qquad (4.59)$$

For the TM case, the connecting conditions for the bottom side of the FDTD grid are

$$H_{y,tot}\Big|_{j_B}^{n+1} = H_{y,tot}\Big|_{j_B}^{n} - \frac{c\Delta t}{\eta_0 \Delta x}\left(E_{z,tot}\Big|_{j_B+1/2}^{n+1/2} - E_{z,scat}\Big|_{j_B-1/2}^{n+1/2} - E_{z,inc}\Big|_{j_B-1/2}^{n+1/2}\right) \text{ and} \tag{4.60}$$

$$E_{z,scat}\Big|_{j_B-1/2}^{n+1/2} = E_{z,scat}\Big|_{j_B-1/2}^{n-1/2} + \frac{c\Delta t}{n^2 \Delta x}\eta_0 \left(H_{y,tot}\Big|_{j_B}^{n} - H_{y,scat}\Big|_{j_B-1}^{n} - H_{y,inc}\Big|_{j_B}^{n}\right), \tag{4.61}$$

and for the top side of the grid

$$H_{y,tot}\Big|_{j_T}^{n+1} = H_{y,tot}\Big|_{j_T}^{n} - \frac{c\Delta t}{\eta_0 \Delta x}\left(E_{z,scat}\Big|_{j_T+1/2}^{n+1/2} - E_{z,tor}\Big|_{j_T-1/2}^{n+1/2} + E_{z,inc}\Big|_{j_T+1/2}^{n+1/2}\right) \quad \text{and} \tag{4.62}$$

$$E_{z,scat}\Big|_{j_t+1/2}^{n+1/2} = E_{z,scat}\Big|_{j_t+1/2}^{n-1/2} + \frac{c\Delta t}{n^2 \Delta x}\eta_0 \left(H_{y,scat}\Big|_{j_T+1}^{n} - H_{y,tot}\Big|_{j_T}^{n} + H_{y,inc}\Big|_{j_T}^{n}\right). \tag{4.63}$$

For a three-dimensional lattice, implementation of the total-field/scattered-field approach follows in an analogous fashion. For all cases considered, the incident illumination is a normally incident plane wave. However, other types of illumination are allowable in principle.

4.2.9 FDTD grid symmetry

In certain applications in which symmetry exists, it is more efficient and less computationally burdensome to implement FDTD with those considerations in mind. For example, if an optical element is symmetric about its optical axis and is illuminated normally with a TE electric field or a TM magnetic field, the analysis can be simplified by taking advantage of the symmetry. Letting z be the optical axis, the symmetry relations are

for the TE case,

$$E_y(-x) = E_y(x)\ , \tag{4.64}$$

$$H_x(-x) = H_x(x)\ , \text{ and} \tag{4.65}$$

$$H_z(-x) = -H_z(x)\ , \tag{4.66}$$

and for the TM case,

$$H_y(-x)=H_y(x)\ , \tag{4.67}$$

$$E_x(-x)=E_x(x)\ \text{, and} \tag{4.68}$$

$$E_z(-x)=-E_z(x)\ . \tag{4.69}$$

Taking advantage of the symmetry of the problem generally cuts the computational time in half and requires about one half as much memory. Besides symmetry, other geometries exist such that the computational FDTD lattice may be truncated. For example, structures that are periodic in two or three dimensions need only be analyzed over one spatial period. Therefore, the computational lattice only needs to be large enough to encompass a given period of the structure [34].

4.2.10 Relative permittivity spatial averaging

Given that in many DOE applications, there are sharp spatial transitions in the relative permittivity across a DOE interface, the accuracy of the FDTD time-marching algorithm is improved by relative permittivity spatial averaging [5]. At a given point in the FDTD grid, an average relative permittivitty may be calculated using the integral form of Maxwell's equations and this would yield more accurate results.

Using a two-dimensional TE FDTD grid and applying Ampere's law in integral form around a rectangular loop in which E_y is at the center gives

$$\oint_C \vec{H}\cdot d\ell = \iint_S \vec{J}\cdot d\vec{S} + \iint_S \frac{\partial \vec{D}}{\partial t}\cdot d\vec{S}\ . \tag{4.70}$$

If no sources are present, then J=0, yielding

$$\oint_C \vec{H}\cdot d\ell = \iint_S \frac{\partial \vec{D}}{\partial t}\cdot d\vec{S} = \iint_S \frac{\partial (\varepsilon \vec{E})}{\partial t}\cdot d\vec{S} = \frac{1}{c\eta_0}\iint_S \frac{\partial (n^2 \vec{E})}{\partial t}\cdot d\vec{S}\ . \tag{4.71}$$

Introducing finite-differences yields

$$\oint_C \vec{H} \cdot d\vec{\ell} = \frac{1}{c\eta_0} \frac{\Delta E_y}{\Delta t} \iint_S n^2 \cdot d\vec{S} . \tag{4.72}$$

Representing Equation 4.71 in discretized form gives

$$\Delta H_x \Delta x + \Delta H_z \Delta z = \frac{1}{c\eta_0} \frac{\Delta E_y}{\Delta t} \iint_S n^2 \cdot d\vec{S} , \tag{4.73}$$

$$\frac{\Delta H_x}{\Delta z} + \frac{\Delta H_z}{\Delta x} = \frac{1}{c\eta_0} \frac{\Delta E_y}{\Delta t} \frac{\iint_S n^2 \cdot d\vec{S}}{\Delta x \Delta z} , \text{ and} \tag{4.74}$$

$$\frac{\Delta H_x}{\Delta z} + \frac{\Delta H_z}{\Delta x} = \frac{1}{c\eta_0} \frac{\Delta E_y}{\Delta t} n_{eff}^2 , \tag{4.75}$$

in which the effective relative permittivity may be represented as

$$n_{eff}^2 = \frac{\iint_S n^2 \cdot d\vec{S}}{\Delta x \Delta z} . \tag{4.76}$$

Note that the above effective relative permittivity is an average over a Yee cell. The derivation could have also been performed using a TM grid giving the same result. In a simple situation in which there are only two materials present, it is practical in many problems to apply relative permittivity spatial averaging to only those cells closest to a dielectric boundary. For DOE applications, for example, the effective relative permittivity at points near a boundary as shown in Figure 4.10 may be expressed as

$$n_{eff}^2 = \frac{n_1^2 \cdot \text{Area}_1 + n_2^2 \cdot \text{Area}_2}{\text{Area}_1 + \text{Area}_2} . \tag{4.77}$$

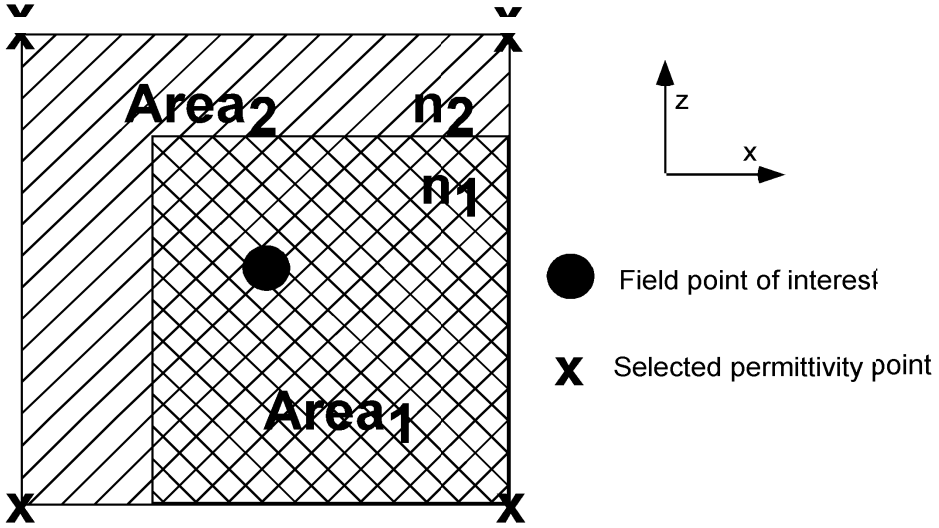

Figure 4.10 Relative permittivity spatial averaging.

4.2.11 Example: Dielectric cylinder

One method in which to test the accuracy of the user implementation of an FDTD computer code is to solve a problem with a known solution. A case of a dielectric cylinder is such a case. It also falls into a very exclusive class of problems in which an analytic solution is known to exist.

Consider a TE uniform plane wave traveling in the +x direction in free space that is incident normally on a lossless dielectric cylinder of radius a as shown in Figure 4.11. The incident, scattered, and transmitted electric fields can be written as [4]

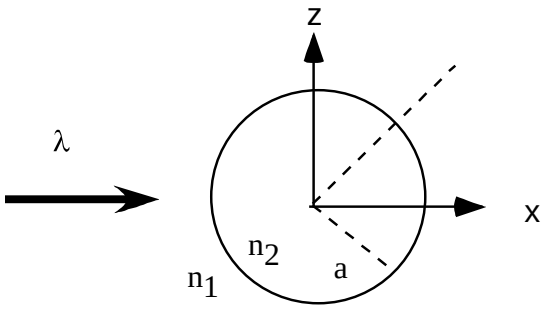

Figure 4.11 Dielectric cylinder geometry.

$$\vec{E}_{inc} = \hat{z}\, E_0 \sum_{n=-\infty}^{+\infty} j^{-n} J_n(k_1\rho) e^{jn\phi}\,, \tag{4.78}$$

$$\vec{E}_{scat} = \hat{z}\, E_o \sum_{n=-\infty}^{+\infty} a_n H_n^{(2)}(k_1\rho) e^{jn\phi}\,, \text{ and} \tag{4.79}$$

$$\vec{E}_{trans} = \hat{z}\, E_0 \sum_{n=-\infty}^{+\infty} [b_n J_n(k_2\rho) + c_n Y_n(k_2\rho)] e^{jn\phi}\,, \tag{4.80}$$

in which

$$a_n = j^{-n} \frac{J_n'(k_1a)J_n(k_2a) - \sqrt{\varepsilon_r/\mu_r}\, J_n(k_1a)J_n'(k_2a)}{\sqrt{\varepsilon_r/\mu_r}\, J_n'(k_2a)H_n^{(2)}(k_1a) - J_n(k_2a)H_n^{(2)\prime}(k_1a)}, \tag{4.81}$$

$$b_n = j^{-n} \frac{J_n(k_1a)H_n^{(2)\prime}(k_1a) - J_n'(k_1a)H_n^{(2)}(k_1a)}{J_n(k_2a)H_n^{(2)\prime}(k_1a) - \sqrt{\varepsilon_r/\mu_r}\, J_n'(k_2a)H_n^{(2)}(k_1a)}, \tag{4.82}$$

$$c_n = 0\,, \tag{4.83}$$

and ρ is the radial distance from the origin and φ is the positive angle with respect to the +x axis. k_1 and k_2 are the wave numbers in media 1 and 2, respectively. The Hankel function of the second kind of the nth order is given by $H_n^{(2)}$. The Bessel functions J_n and Y_n are of the first and

second kind, respectively, of order n. Also note that the primes (') denote the total derivative with respect to the total arguments.

Note that the ratio of permittivities of the two media may be expressed as $\varepsilon_r = \frac{\varepsilon_2}{\varepsilon_1} = \frac{n_2^2}{n_1^2}$, and assuming a non-magnetic material, the relative permeability, μ_r, is set to unity.

Consider a TM uniform plane wave traveling in the +x direction in free space that is incident normally on the same lossless dielectric cylinder of radius a as shown in Figure 4.11. The incident, scattered, and transmitted electric fields can be written as

$$\vec{H}_{inc} = \hat{z} H_0 \sum_{n=-\infty}^{+\infty} j^{-n} J_n(k_1\rho) e^{jn\phi}, \tag{4.84}$$

$$\vec{H}_{scat} = \hat{z} H_o \sum_{n=-\infty}^{+\infty} a_n H_n^{(2)}(k_1\rho) e^{jn\phi}, \text{ and} \tag{4.85}$$

$$\vec{H}_{trans} = \hat{z} H_o \sum_{n=-\infty}^{+\infty} [b_n J_n(k_2\rho) + c_n Y_n(k_2\rho)] e^{jn\phi}, \tag{4.86}$$

in which

$$a_n = j^{-n} \frac{J_n'(k_1a) J_n(k_2a) - \sqrt{\mu_r/\varepsilon_r} J_n(k_1a) J_n'(k_2a)}{\sqrt{\mu_r/\varepsilon_r} J_n'(k_2a) H_n^{(2)}(k_1a) - J_n(k_2a) H_n^{(2)'}(k_1a)}, \tag{4.87}$$

$$b_n = j^{-n} \frac{J_n(k_1a) H_n^{(2)'}(k_1a) - J_n'(k_1a) H_n^{(2)}(k_1a)}{J_n(k_2a) H_n^{(2)'}(k_1a) - \sqrt{\mu_r/\varepsilon_r} J_n'(k_2a) H_n^{(2)}(k_1a)}, \text{ and} \tag{4.88}$$

$$c_n = 0. \tag{4.89}$$

The parameters for the dielectric cylinder, which are the same as in Chapter 3, are as follows: λ = 1.0 μm, a = 0.5 μm, n_1=1.0, and n_2=1.5.

Figure 4.12 shows the comparison of the scattered field amplitudes and phases calculated via FDTD and analytical results for both the TE and TM cases for the specified parameters.

Noting the excellent agreement between FDTD and analytical results, one may conclude that the implementation of the FDTD code is in fact correct.

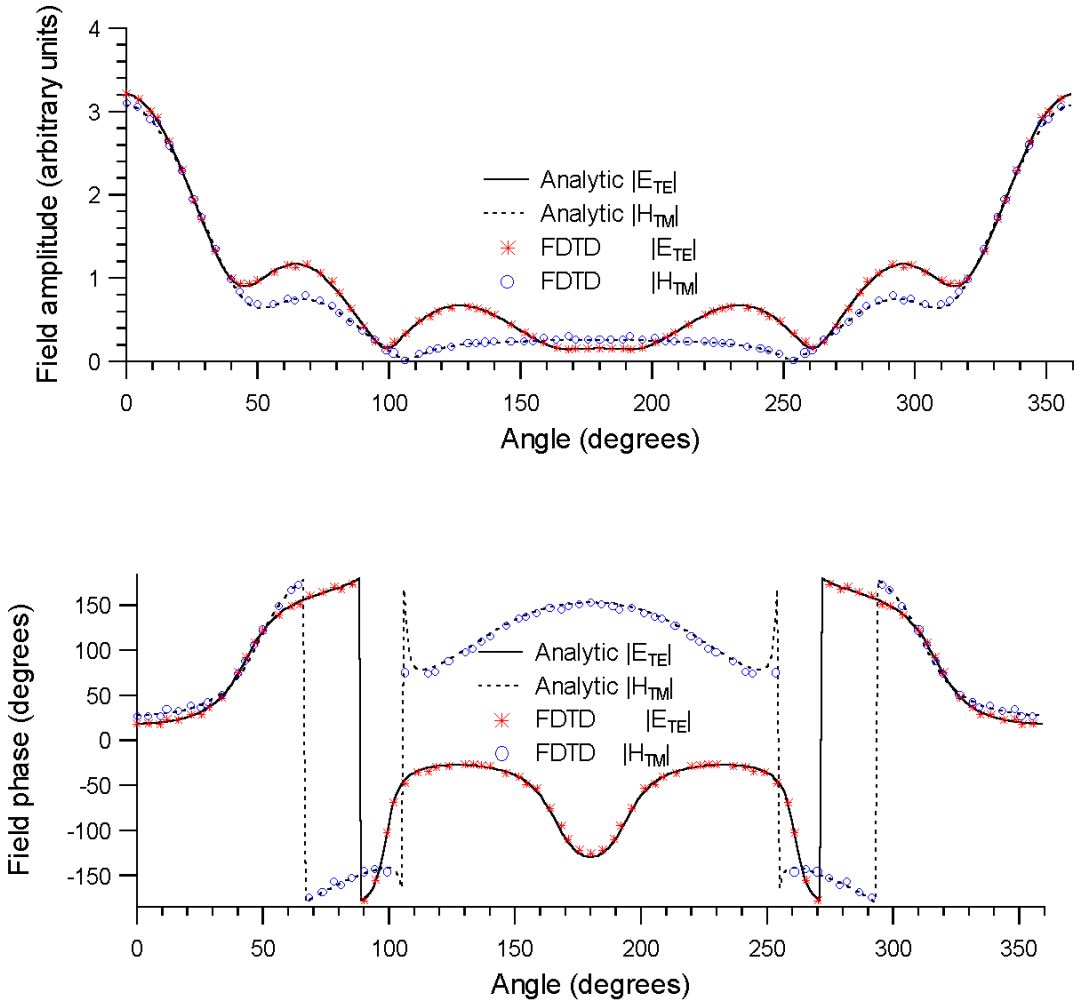

Figure 4.12 FDTD analysis of a dielectric cylinder.

4.2.12 Field propagation to plane of interest

Once the FDTD time-marching algorithm has converged to a time-harmonic steady-state solution, fields may be determined everywhere in space. In finite aperture DOE applications, one may propagate the fields in an exiting medium to a given plane of interest using a variety of methods. Two such methods are Green's theorem method [35] or the angular spectrum approach [2,23]. In all applications considered, the angular spectrum approach was used since it is more computationally efficient. It is also worth mentioning that while running the FDTD algorithm, monitoring the total energy within the computational grid is useful in determining whether a time-harmonic steady-state solution has been reached. If the total energy within the grid remains constant after a certain number of iterations, it is reasonable to assume that a time-harmonic steady-state solution has been obtained.

In every application considered, there is a homogeneous material in the exiting medium of a DOE. Consider a cross-section of the FDTD grid in an exiting medium as shown in Figure 4.13. Assume the field is essentially zero to the left and to the right of that line segment in the cross-section. From this, the angular spectrum of the TE electric field or the TM magnetic field may be calculated at this plane using a Fast Fourier Transform (FFT) [23]. The angular spectrum of the field may be propagated to the plane of interest. Then the field may be calculated in the plane of interest by taking an inverse FFT of the spatial frequency spectrum in that plane. Other propagation methods include the Fresnel or Fraunhofer approximations.

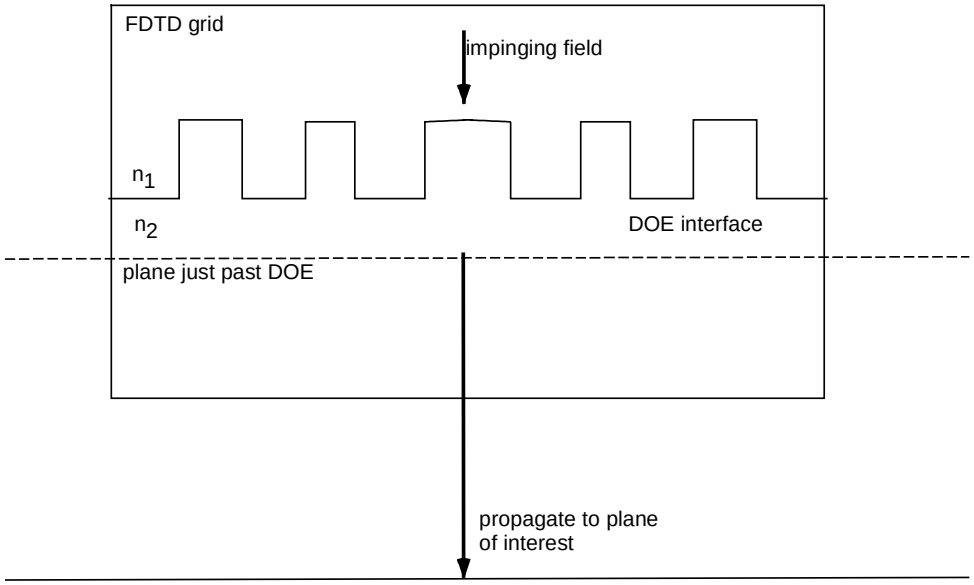

Figure 4.13 Propagation to plane outside FDTD computational grid.

4.2.13 FDTD capabilities

As discussed in Chapter 2 on scalar diffraction theory, if a TE electric field or TM magnetic fields is known at a plane, other field components may be calculated at that plane [23]. For the TE case, for example, if E_y is known then H_x and H_z can be calculated. Similarly, for the TM case, if H_y is known, then E_x and E_z can be calculated at the cross-sectional plane of the FDTD grid. Using plane wave propagation techniques, all the field components can be determined at a plane of interest well past the FDTD computational grid. The Poynting vector and, hence, the irradiance can then be determined in a given plane of interest as given by

$$I = \frac{1}{2}\left|\mathrm{Re}\left\{\vec{S}\right\}\cdot\hat{n}\right| , \tag{4.90}$$

in which $\hat{n}$ is the outward unit normal to the plane of interest and $\bar{S}$ is the Poynting vector and given by

$$\bar{S}=S_x\hat{x}+S_z\hat{z}=\frac{1}{2}\bar{E}\times\bar{H}^*. \tag{4.91}$$

For the TE case, the components can be expressed as

$$S_x=\ \frac{1}{2}E_yH_z^*\ \text{ and }\ S_z=-\frac{1}{2}E_yH_x^*, \tag{4.92}$$

and for the TM case

$$S_x=-\frac{1}{2}E_zH_y^*\ \text{ and }\ S_z=\ \frac{1}{2}E_xH_y^*. \tag{4.93}$$

Also, using Maxwell's equations, electromagnetic quantities such as the electric and magnetic energy within the FDTD computational grid may be calculated.

Extensive work for this dissertation has been done on implementing the FDTD code for a variety of applications. Such applications would include but are not limited to polarimetry and field component animation. If both TE and TM cases are run, polarimetric quantities such as diattenuation and retardance, respectively, may be calculated using the following equations:

$$D=\frac{E_{x,TM}^2-E_{y,TE}^2}{E_{x,TM}^2+E_{y,TE}^2}\ \text{ and} \tag{4.94}$$

$$R=\text{phase}(E_{x,TM})-\text{phase}(E_{y,TE}). \tag{4.95}$$

Using animation tools such as Apple QuickTime™, it is also possible to visualize the electric or magnetic fields in an FDTD grid. In such cases, it is necessary to produce only image plots of the real parts of these fields. The animation feature allows the user to examine, for example, aperture effects or DOE edge scattering effects in a particular application.

4.2 Summary of FDTD

In this chapter, the FDTD implementation is introduced and derived for practical diffractive optic applications. Future software development will likely entail the development of FDTD for three-dimensional problem applications and development of animation routines to allow for visualization of electric and magnetic fields near diffractive optic interfaces to study the effects of scattering along DOE boundaries.

Chapter 5

LIMITS OF SCALAR DIFFRACTION THEORY

5.1 Motivation

Scalar diffraction theory is often used in the design and analysis of diffractive optical elements. It is straightforward to use and the computational burden is minimal. Generally, the use of scalar diffraction theory is only considered valid if the DOE minimum feature size is much greater than the wavelength of incident light [8]. However, several applications have been found in which a scalar-based diffraction theory is surprisingly accurate, even though the minimum features are smaller than the wavelength of the incident light.

In this chapter, design examples of beamfanners are presented that illustrate the use of scalar-based design methods and explore limitations of the use of scalar-based diffraction theory to design DOEs. There are two fundamental assumptions associated with the application of scalar diffraction theory to design finite-aperture DOEs. The first is that the optical field just past the DOE can be described by a simple transmission function [2]. The second involves the choice of method to propagate the fields to the plane of interest. In most applications presented in the literature, the plane of interest is often located in the far-field. In the application presented here, however, the plane of interest is in the near field, so the usual selection of Fraunhofer diffraction for the propagation method is not accurate. Fresnel propagation is commonly used in the near-field. This method, however, is still only approximate and is inaccurate for propagating fields with high spatial frequency content. Therefore the angular spectrum approach [2,23] is used,

which is rigorous within scalar diffraction theory and is also formally equivalent to electromagnetic TE electric field and TM magnetic field propagation. To make the distinction clear, the term angular spectrum (AS) scalar diffraction theory is used to emphasize that the angular spectrum approach is used as the method of propagation.

The limits of scalar diffraction theory as the DOE minimum feature size approaches the wavelength of illuminating light have not been fully quantified in the literature, and only a few authors have attempted to assess scalar theory in this regime. Gremaux and Gallagher investigated the limits of scalar diffraction theory for perfectly conducting gratings [36]. In this work, however, the investigation is limited to phase-only finite aperture dielectric DOEs.

Pommet, et al. [37] examined the limits of scalar diffraction theory for diffractive phase elements. They compared the diffraction characteristics of single-level and multilevel dielectric gratings predicted by scalar transmission theory with those obtained by rigorous coupled-wave theory (RCWT) [21] and found that, in general, the error of scalar theory is significant when the grating period is less than 14 wavelengths. They also report that the error is minimized when the fill factor approaches 50% for DOEs with smaller features of 2 wavelengths. They also assert that for DOEs having an overall fill factor of 50%, the larger period of the DOE replaces the smaller feature size as the condition of validity for scalar diffraction theory. In this chapter, the findings in regards to the relation between the size of DOE features and the wavelength of illuminating light are discussed, specifically, that AS scalar theory is accurate provided that the spatial period of the DOE is not much less than twice the free-space wavelength. In Section 5.2, the designs of 1-2 beamfanners are presented and the accuracy of a scalar diffraction approach used to heuristically design and analyze the DOEs is assessed by comparing with rigorous results.

5.2 Heuristic design and analysis of 1-2 beamfanners

Consider a DOE that functions as a 1-2 beamfanner shown in Figure 5.1. The DOEs function is to maximize the amount of light directed into the two apertures in what will be referred to as the observation plane. Note that the observation plane is in the near-field. A plane-wave with a free-space wavelength, λ_0, of 5 μm is normally incident on a silicon-air interface bounded by a 50 μm ($10\lambda_0$) metallic aperture having infinite conductivity. The minimum feature size of the DOE is chosen to be 1 μm ($0.2\lambda_0$) and the observation plane at a distance of 100 μm in air ($20\lambda_0$). Next, a heuristic method is considered to design a DOE to perform the 1-2 beamfanning function.

To begin the design, take an infinite grating with spatial period, Λ, and truncate it to fit the bounds of the finite aperture. The fill factor of the grating is set to 50% and its depth, d, is set such that the power directed to the ±1 diffracted orders is maximized. As proven in Appendix A, this occurs when the relative phase shift between adjacent zones is π radians, which corresponds to $d = \lambda_0/(2\Delta n)$ for normally incident light, where Δn is the difference in refractive index between silicon and air. In this chapter, a zone is defined as a region in which the DOE profile is locally continuous and has no sharp edges, e.g., for a binary grating with a 50% fill factor, a zone constitutes half the grating period. Next, a quadratic phase profile is added to the grating to focus the light in the near field observation plane.

For scalar-based analysis of DOE diffraction, a transmission function of the form $t(x)=\tau\exp(j\varphi(x))\,\mathrm{rect}(x/L)$ is assumed in which L is the width of the finite aperture and $\mathrm{rect}(x/L)$ equals unity for $|x|<L/2$ and is zero otherwise. The profile phase, $\varphi(x)$, can be expressed as $\varphi(x) = (2\pi/\lambda_0\Delta n)\,d(x)$ in which $d(x)$ is the etch depth along the DOE. The Fresnel transmission coefficient, τ, is given by $\tau=2n_1\cos(\theta_1)/(n_1\cos(\theta_1)+n_2\cos(\theta_2))$ in which n_1 and

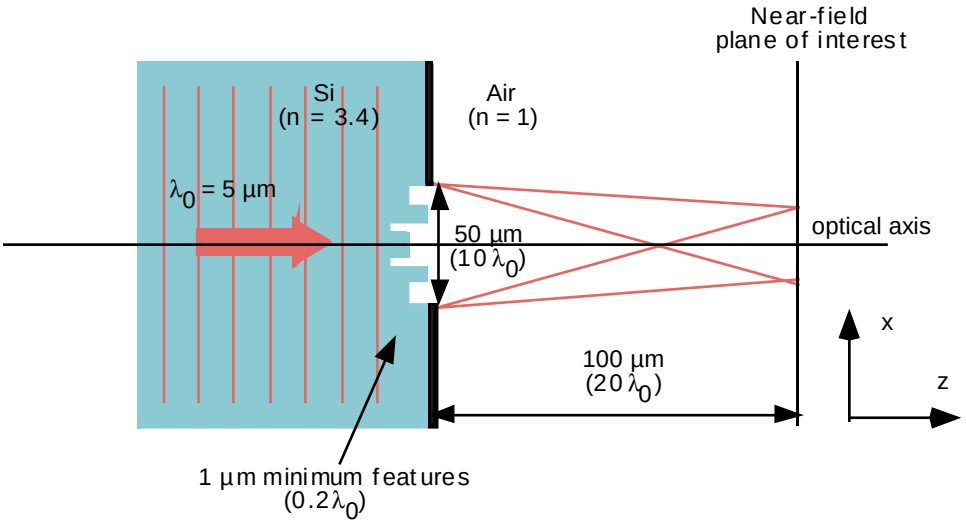

Figure 5.1 Diffractive optic geometry.

n_2 are the refractive indices of silicon and air, respectively, θ_1 is the angle of the incidence with respect to the optical axis and θ_2 is the refracted angle as calculated by Snell's law. In every case in this work, only normally incident light is considered, for which the Fresnel transmission coefficient reduces to $\tau=2n_1/(n_1+n_2)$. The field incident on a DOE is multiplied by the transmission function, t(x), which gives the field just past the DOE interface. This field is then propagated to a chosen observation plane using the angular spectrum approach. Once the field is known in this observation plane, all other field component can be obtained via Maxwell's equations, and the power and the diffraction efficiency can be calculated. By using AS scalar-based theory, the goal is to determine what DOE profile, $d(x)$, best performs its intended function. In the heuristic designs presented, special attention is give to the design of 1-2 beamfanners.

Given that the DOEs have sub-wavelength features, it may appear counterintuitive to use a scalar-based approach in their design. It turns out, however, that several of these designs are

very accurate. All scalar-based results were compared to those obtained using a rigorous finite-difference time-domain method (FDTD) for both TE and TM modes [5]. The FDTD analysis, using Mur 2nd order boundary conditions and the total-field/scattered-field formulation to introduce the incident fields, employed a time-marching scheme to analyze the electric and magnetic field values at sampled points within the FDTD grid, which enclosed the entire DOE structure. At a plane immediately past the DOE in the exiting medium intersecting the FDTD grid, the field was propagated to the observation plane using the angular spectrum approach [2,23].

In the test cases, the effect was examined of varying the spatial period of the grating on the accuracy of applying AS scalar diffraction to analyze DOEs. The values of L chosen were 8 λ_0, 4 λ_0, 2 λ_0, 1.6 λ_0, 1.4 λ_0, and 1.2 λ_0, and the resultant DOE profiles for these cases are shown in Figure 5.2. Figure 5.3 shows the corresponding near-field irradiance in the observation plane. Given that the distance of this observation plane is 100 μm from the DOE interface and the location of the diffracted orders in that plane, note that the periods are consistent with those predicted by the grating equation: $\lambda / n = \Lambda \sin(\theta)$ [8], in which λ is the reduced wavelength in the propagating medium, θ is the angle of the first diffracted order with respect to the optical axis, i.e., $\theta = \tan^{-1}(s/2z)$, s is the separation between the diffracted orders in the observation plane, and z is the distance to that plane. In Figure 5.3, note that for larger grating periods the AS scalar results agree very well with those predicted by FDTD, whereas for smaller grating periods the validity of AS scalar theory appears to break down.

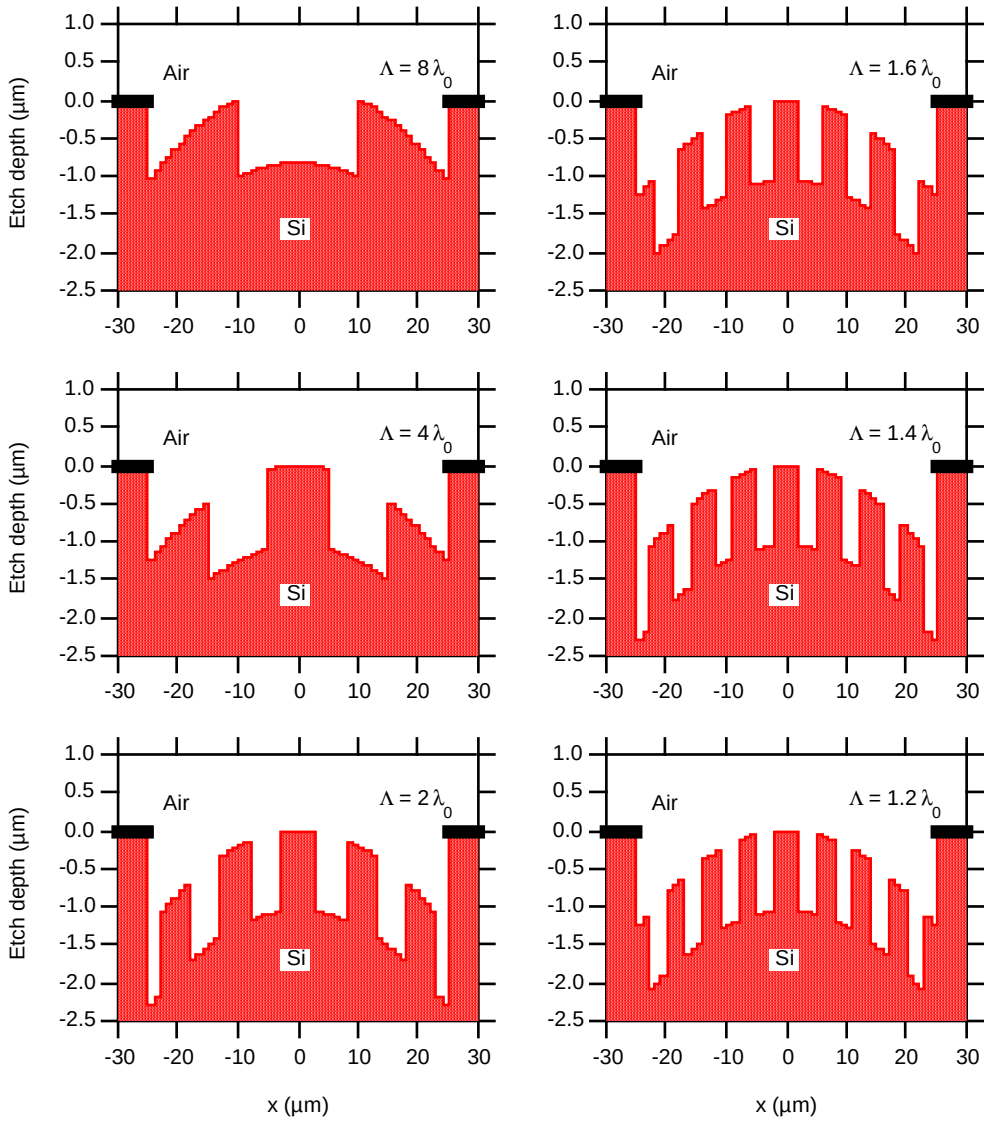

Figure 5.2 Heuristically designed DOE profiles.

To quantify these results, the diffraction efficiency, η, is defined as the fraction of light diffracted within the photodetector apertures in the observation plane, that is,

$$\eta = \frac{\int_{\text{apertures}} \text{Idx}}{\int_{\text{all space}} \text{Idx}}, \tag{5.1}$$

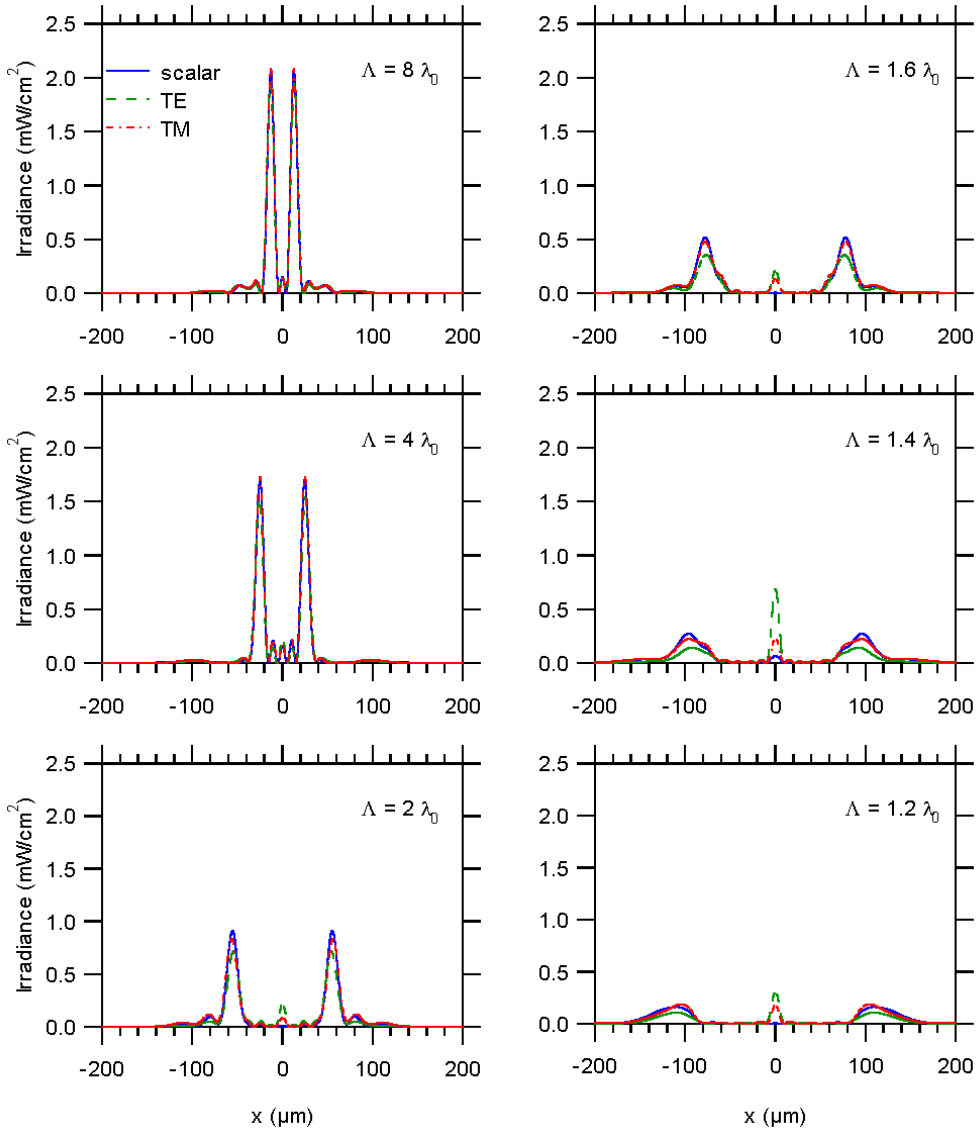

Figure 5.3 Irradiance profiles for heuristically designed DOE profiles.

in which I is the irradiance given by $I = \left|\mathrm{Re}\{\vec{S}\} \cdot \hat{n}\right|$ and $\vec{S}$ is the complex Poynting vector in the observation plane and $\hat{n}$ is the outward unit normal from the observation plane. The width of each of the photodetector apertures is chosen to be 15 μm. The error in terms of diffraction efficiency between rigorous TE or TM results with respect to AS scalar results is defined here as

$$\text{Error}_s = \left| \frac{\eta|_{\text{TE or TM}} - \eta|_{\text{scalar}}}{\eta|_{\text{scalar}}} \right|, \tag{5.2}$$

and the difference between the TM result with respect to the TE result as

$$\text{TE/TM difference} = \left| \frac{\eta|_{\text{TM}} - \eta|_{\text{TE}}}{\eta|_{\text{TE}}} \right|. \tag{5.3}$$

Table 5.1 lists the diffraction efficiencies calculated from AS scalar theory and from FDTD TE and TM cases as well as the defined errors.

Table 5.1 Diffraction efficiencies and errors for heuristically designed 1-2 beamfanners.

Λ/□0	DE-scalar (%)	DE-TE (%)	DE-TM (%)	TE error (%)	TM error (%)	TE-TM difference (%)
8	84.333	84.09	83.39	0.28812	1.1174	0.83164
4	81.616	81.55	79.889	0.080934	2.1161	2.0368
2	65.867	59.370	60.609	9.864	7.982	2.0873
1.6	47.018	40.062	40.913	14.793	12.983	2.1239
1.4	28.594	16.343	22.471	42.846	21.414	37.5
1.2	7.4285	4.4222	5.5904	40.47	24.744	26.417

For smaller grating periods, the deviation of AS scalar results from those predicted by FDTD becomes more significant. This indicates that AS scalar theory works well provided that the spatial period, Λ, is not much less than twice the free-space wavelength, i.e., $\Lambda \geq 2\lambda_0$. This assessment appears to indicate that perhaps the validity and accuracy of AS scalar diffraction theory is slightly broader than other results reported in the literature [37]. In terms of zones, AS scalar diffraction theory is accurate for zone sizes greater than or equal to the free space wavelength of incident light. Also note that the sub-wavelength minimum feature size (1 μm) is the same for each beamfanner in the analysis. Therefore, one may conclude that the validity of AS scalar diffraction theory does not necessarily depend on DOE minimum feature size but rather on the size of the DOE zones.

Like AS scalar diffraction, the implementation of the FDTD method also incorporates the angular spectrum approach to propagate the field just past the DOE to the observation plane. Therefore, one can conclude that it is the assumption of a simple transmission function that causes the breakdown of the validity of AS scalar theory in comparison with FDTD. Therefore the fields immediately past the DOE structure in the exit medium were examined and the field amplitudes and phases predicted by AS scalar theory were compared with those predicted by FDTD. Figures 5.4 and 5.5 show the results of the comparison for the phases and amplitudes, respectively, for all the test cases considered. Note that the phases calculated by FDTD agree very well with AS scalar theory for large grating periods and only deviate slightly as the grating period progressively decreases relative to λ_0. The field amplitudes calculated via FDTD, however, differ more dramatically from the AS scalar result, which is simply a rectangle function scaled by the Fresnel transmission coefficient. This apparent disagreement in field magnitudes can be reconciled somewhat by examining only those field components that actually propagate to the observation plane which are not evanescent as will be shown next.

First consider the angular spectra of the fields at a plane just past the DOE shown in Figure 5.6. Note that for the cases of Λ equaling $8\ \lambda_0$, $4\ \lambda_0$, and $2\ \lambda_0$, the magnitudes of the angular spectra calculated from rigorous analysis agree well with what was calculated from the scalar analysis while the cases $1.6\lambda_0$, $1.4\lambda_0$, and $1.2\lambda_0$, tend to deviate. Also note that the spatial frequency components outside the range of $[-1/\lambda_0, 1/\lambda_0]$ are evanescent.

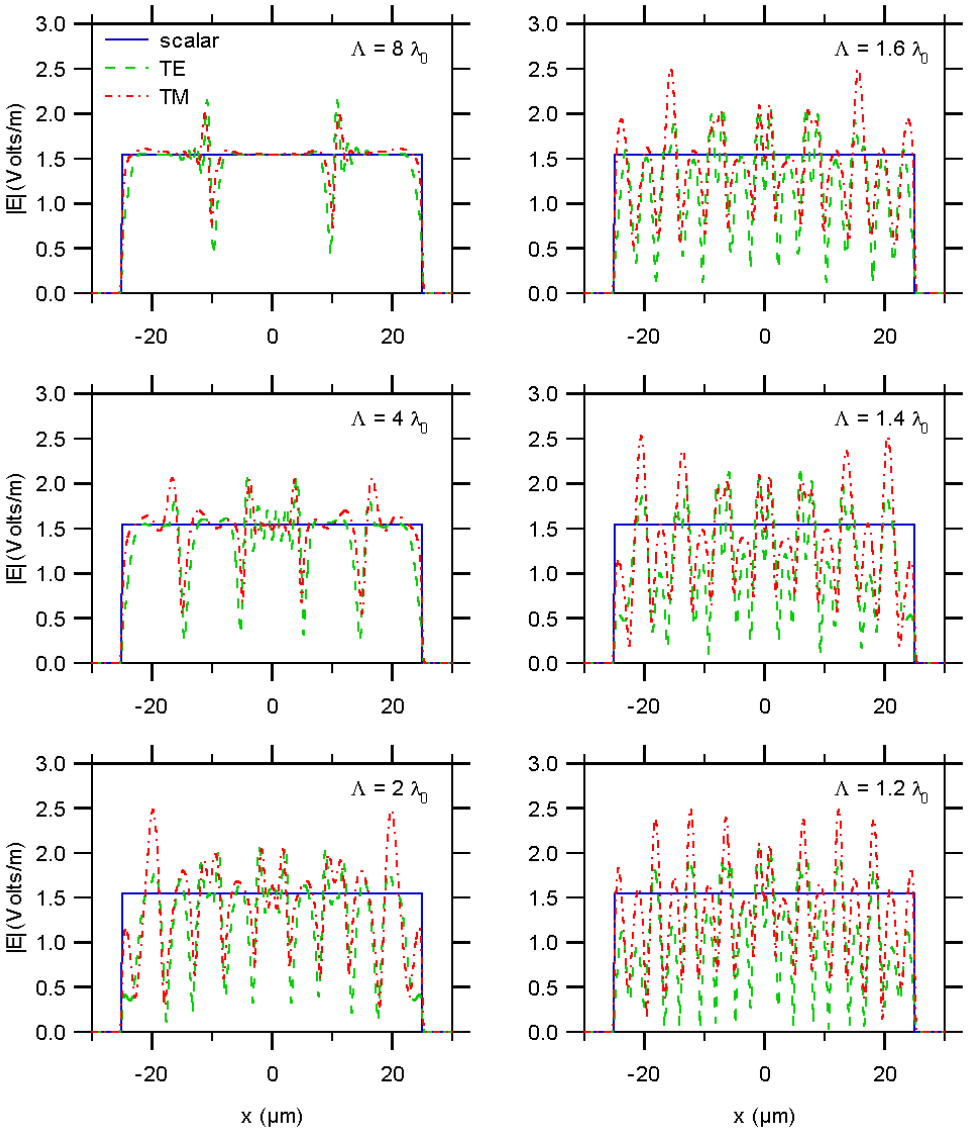

Figure 5.4 Total field amplitudes for heuristic gratings.

From the angular spectrum calculations, one can examine only those field components just past the DOE that actually propagate to the observation plane. The non-evanescent components of the total field that do, in fact, propagate to the observation plane are referred to as the propagating field. As is shown in Figure 5.7, the comparison of the phases for the propagating field just past the DOE is not much different from the total field phase case.

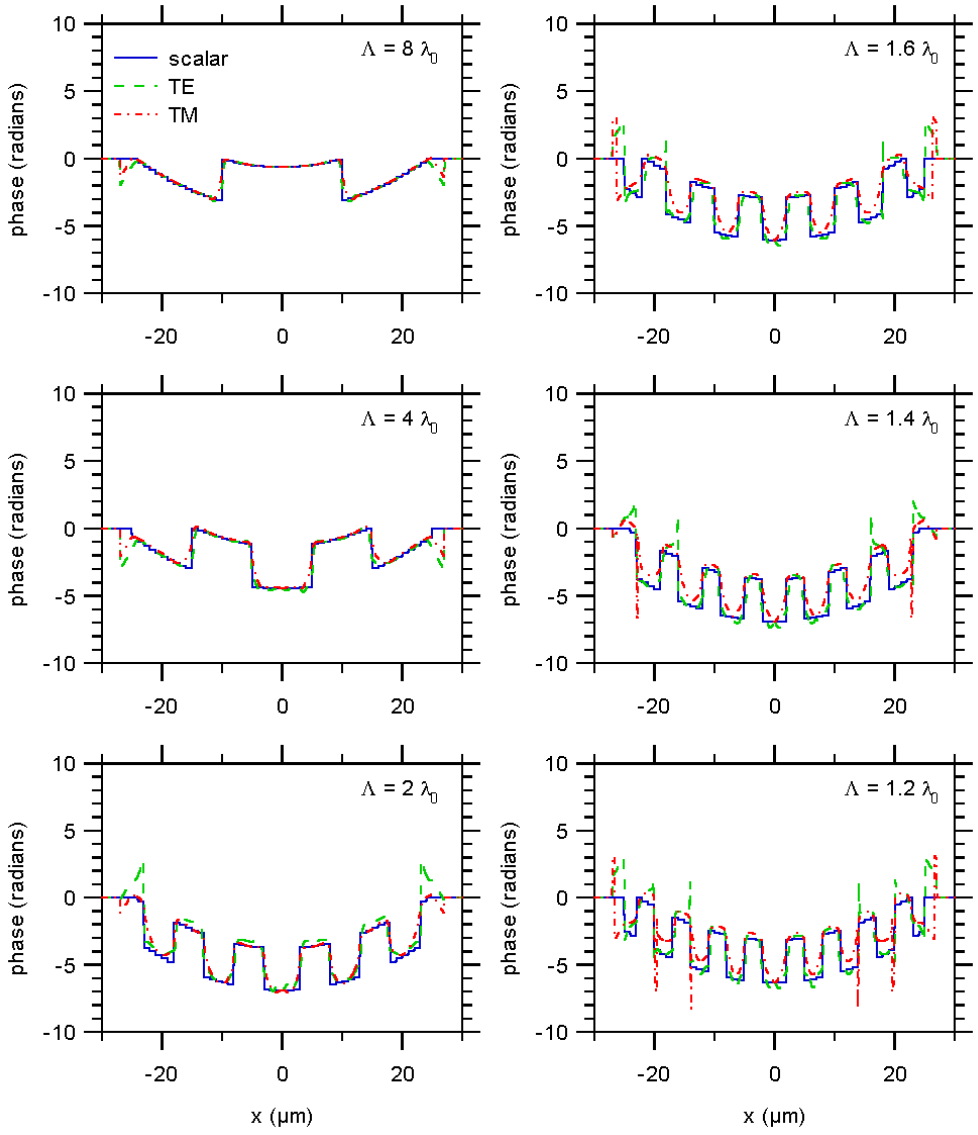

Figure 5.5 Total field phases for heuristic gratings.

In contrast, examination of the propagating field amplitudes just past the DOE reveals that scalar and rigorous results are in much better agreement than the total field analysis as shown in Figure 5.8. The analysis of the propagating field amplitudes just past the DOE indicate that the scalar and rigorous results are in agreement for the cases of Λ equaling $8\lambda_0$, $4\lambda_0$, and $2\lambda_0$. However, AS scalar results tend to differ significantly for the cases when the grating period is less than $2\ \lambda_0$, which was also the case from the analysis of the near-field diffraction patterns. From this, one can conclude that the amplitude of the propagating field just past the DOE is the most dominant factor in determining whether AS scalar diffraction theory is valid in the test cases. Notice that local minima in the propagating field amplitudes occur roughly at the same transverse locations along the object plane as the zone boundaries of the DOE in all the test cases. This also indicates that the DOE edge effects are a dominating factor in determining whether AS scalar diffraction theory is valid.

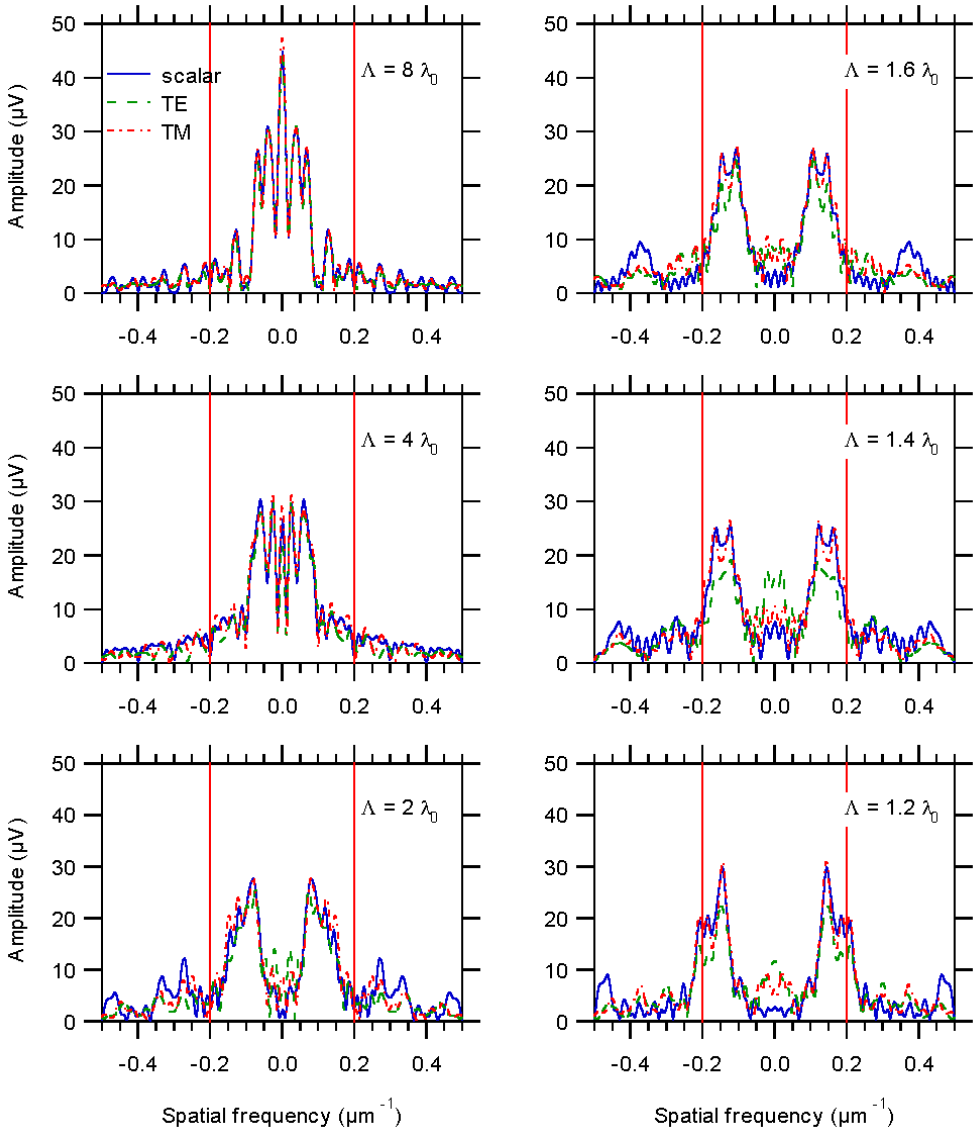

Figure 5.6 Angular spectrum magnitudes for heuristic gratings just past DOE.

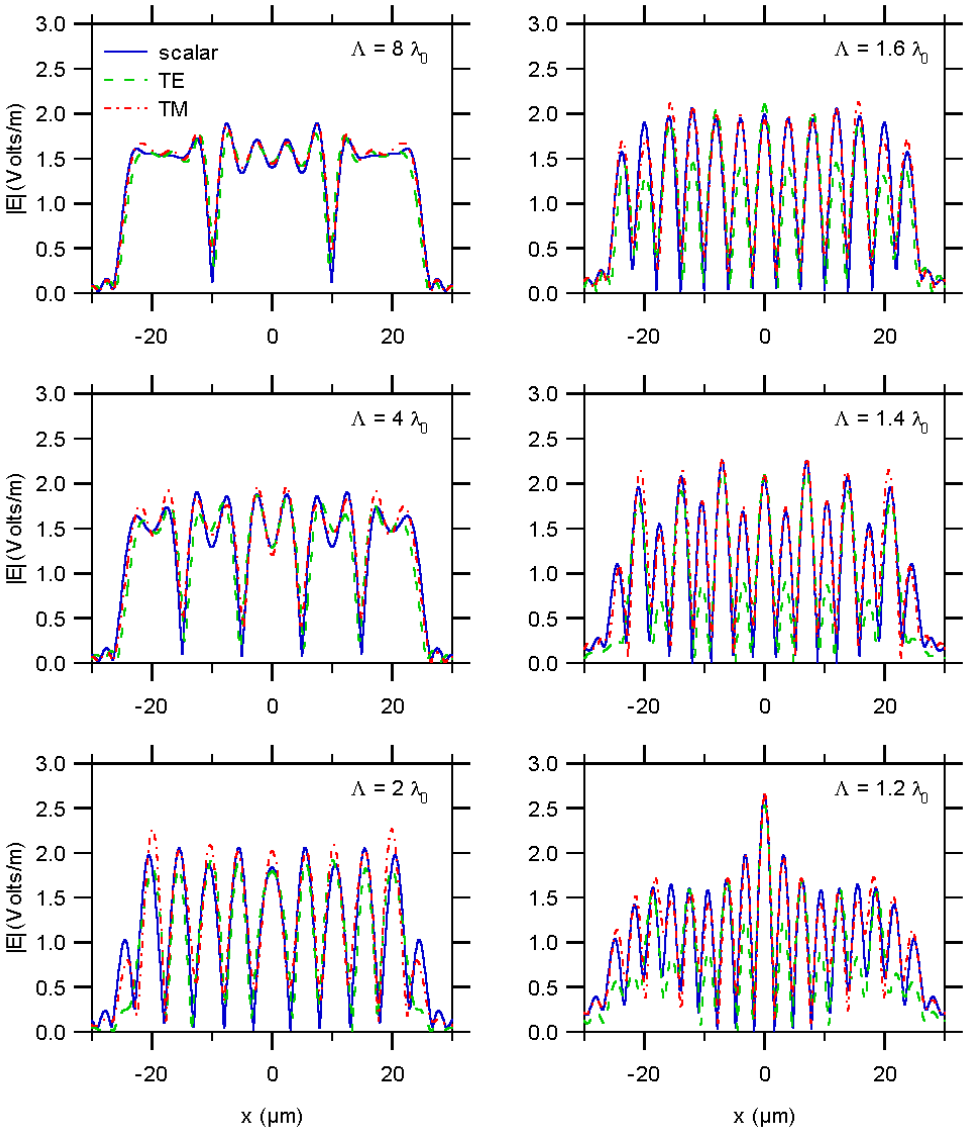

Figure 5.7 Propagating field amplitudes for heuristic gratings just past DOE.

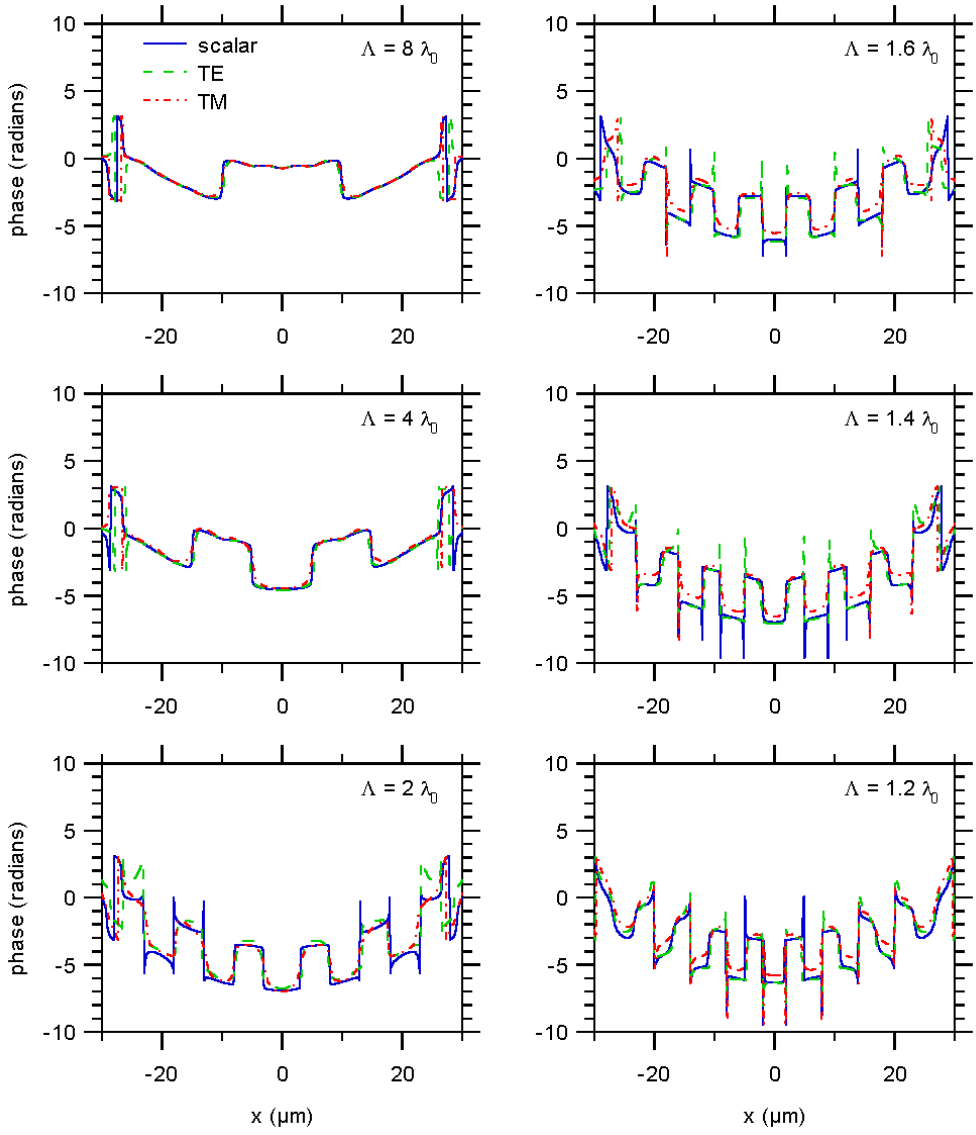

Figure 5.8 Propagating field phases for heuristic gratings just past DOE.

In summary, it appears that AS scalar diffraction theory is applicable in some cases for finite aperture DOE design with features on the order of or less than the wavelength of illuminating light. Results indicate that AS scalar model is a viable method provided that the spatial period, Λ, is not much less than twice the free-space wavelength, i.e., $\Lambda \geq 2\lambda_0$ or, in terms

of zones, AS scalar diffraction theory is accurate for zone sizes greater than or equal to the free space wavelength of incident light. Furthermore, the DOE minimum feature size is not a reliable metric to assess the validity of AS scalar diffraction theory. Even if the minimum feature sizes are less than the wavelength, results indicate that the size of the DOE zones and edge effects are dominating factors in determining whether AS scalar diffraction theory is valid.

Chapter 6

THE ITERATIVE ANGULAR SPECTRUM ALGORITHM

High-efficiency finite-aperture diffractive optical elements (DOEs) have been designed with features on the order of or smaller than the wavelength of the incident illumination. The use of scalar diffraction theory is generally not considered valid for the design of DOEs with such features. However, several cases have been found in which the use of a scalar-based design is, in fact, quite accurate as discussed in the previous chapter. A modified scalar-based iterative design method is presented in this chapter that incorporates the angular spectrum approach to design DOEs having sub-wavelength features that operate in the near-field. This design method is called the iterative angular spectrum approach (IASA). Upon comparison with a rigorous electromagnetic analysis technique, specifically, the finite difference time-domain method (FDTD), it was found that the scalar-based design method was surprisingly valid for DOEs having sub-wavelength features.

Also presented in this chapter is a re-sampling technique that is used to increase the DOE minimum feature size so that the designed DOEs are easier to fabricate. This re-sampling is employed as a post-processing step after IASA is completed. As will be discussed in Section 6.4 for the specific case of a 1-2 beamfanner IASA design, this re-sampling technique when used after IASA yields diffraction efficiencies in an observation plane comparable to a beamfanner with smaller features that has not been re-sampled. Also, the effect of quantizing DOE etch depth levels is also discussed with specific attention given to the 1-2 beamfanner application.

6.1 Introduction

A bidirectional modified iterative design algorithm is presented that is an extension of the Gerchberg-Saxton [18] algorithm, the iterative Fresnel-transform algorithm, and the iterative Fourier-transform algorithm [20,38-41]. Adopting the nomenclature used by Mait [15], bidirectional means that updating the DOE profile requires knowledge of the inversion of the operation used to obtain the field the DOE produces and how variations in the response affect the DOE. In contrast, unidirectional algorithms such as gradient-descent methods and simulated annealing [17,42-45], characterize the DOE by a finite set of quantized parameters such as phase levels, and do not require an inversion operation to update the DOE. As discussed by Mait, when the model of an optical system cannot be inverted, unidirectional algorithms must be used.

The design method presented incorporates the angular spectrum approach in a bidirectional, iterative algorithm. The motivation for including the angular spectrum approach is that it requires no approximations as does Fresnel and Fraunhofer propagation and is therefore more accurate. The major issue to consider is how to deal with evanescent spectral components that arise with the angular spectrum approach, since high spectral frequency information about the DOE structure could be lost in the propagation step of an iterative procedure. Without effective treatment of the evanescent components, any iterative algorithm would have to be unidirectional. It is discussed how to treat evanescent spectral components to make the iterative algorithm truly bidirectional. In Section 6.2, the AS scalar-based iterative design method, IASA, design examples, and a comparison with results from a rigorous electromagnetic analysis is presented.

6.2 The iterative angular spectrum algorithm

Having discussed the validity of AS scalar diffraction theory, a scalar-based design algorithm for DOEs with sub-wavelength feature sizes is presented. The algorithm itself is discussed and a set of test cases is presented.

6.2.1 Design method

An iterative angular spectrum algorithm (IASA) is used in the design of the beamfanners discussed in this chapter. The steps to designing a DOE using IASA are as follows:

1. Assume initial DOE etch depth profile
2. Calculate transmission function and then the field just past DOE
3. Obtain angular spectrum of field via Fast-Fourier Transform (FFT)
4. Forward propagate angular spectrum
5. Determine field via Inverse FFT
6. Compare to desired diffraction pattern
7. If diffraction pattern is satisfactory, IASA is complete, otherwise modify field amplitude according to predetermined weight functions
8. Obtain angular spectrum of this field
9. Back-propagate angular spectrum to DOE object plane
10. Add evanescent components from part 3 to obtain total angular spectrum
11. Fourier Transform spectrum to obtain field just past DOE
12. Modify DOE profile applying appropriate constraints (i.e., phase-only DOE bounded by a finite aperture)
13. Go back to step 2

The most interesting feature of IASA is that to accurately obtain the modified angular spectrum in step 10, the evanescent spectral components from the field that was forward-

propagated are added to the modified field after it is back-propagated. This step is absolutely necessary to accurately recover the modified DOE phase profile. It is this feature that makes IASA novel since, to the best of knowledge, the treatment of evanescent frequencies in iterative algorithms has never been addressed in the literature.

6.2.2 1-2 beamfanner design

In the 1-2 beamfanner design, a unit amplitude plane wave is assumed normally incident on the finite aperture DOE as shown in Figure 6.1. The DOE splits the light between two photodetectors in an observation plane in the near field. The free space wavelength of illumination is 5 μm and the silicon substrate has minimum features of 1 μm bounded by a 50 μm aperture. It is assumed that the etch depth levels are not quantized, although they could be as a post-processing step after IASA has run to completion. The apertures of each photodetector in the observation plane are chosen to have a width of 15 μm. To assess the range in which the use of IASA is valid and accurate, beamfanners were designed that would require grating periods comparable in size to and smaller than the wavelength in free space. The cases presented next are beamfanners intended to split light with peak separations of 25, 50, 100, 150, 200, and 250 μm in an observation plane located 340 μm in the silicon from the silicon-air interface.

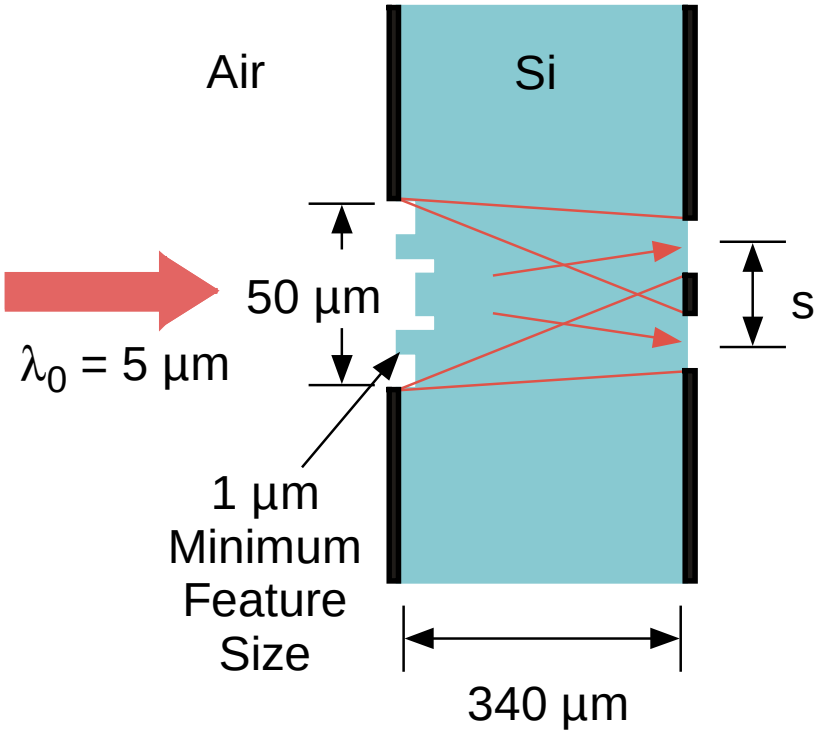

Figure 6.1 Diffractive optic geometry.

IASA was used to design finite aperture DOEs that split light in the plane of interest with the peak separations mentioned above. The assumed initial profile was a flat interface bounded by a 50 µm infinitely conducting finite aperture. One criterion in choosing the weighting functions for a particular case is that the field amplitudes are increased each iteration at positions in the observation plane corresponding to the locations of the photodetector apertures. Another criterion is that the energy of the light impinging on the DOE is homogeneously split between the two peaks in the near-field. This is done to ensure that the DOE profile does not converge to two adjacent microlenses that focus light independently of one another. After every iteration, the amplitude of the field in the observation plane is brought fractionally closer to an optimum profile for the specified weighting functions. For all cases considered, the algorithm converged within 1000 iterations.

The resultant DOE etch depth profiles produced by IASA that yield peak separations of 25, 50, 100, 150, 200, and 250 μm in the observation plane are shown in Figure 6.2, respectively. Note that each etch depth profile looks progressively more like a grating (with possibly several 2π phase resets) with a quadratic envelope, in which the grating performs the beamfanning function while the quadratic phase component focuses light in the observation plane. The etch depths in each case between each zone are roughly a π phase difference, corresponding to an etch depth between adjacent peaks and troughs of 1.03 μm in these cases, which is an expected result for a binary grating having the maximum diffraction efficiency [37]. Given the location of the observation plane and the peaks of the diffracted orders, the periods are consistent with those predicted by the grating equation: $\lambda / n = \Lambda \sin(\theta)$ [8], in which λ is the reduced wavelength of the medium of propagation, Λ is the grating period and θ is the angle of the first diffracted order with respect to the unit normal to the plane of the silicon-air interface. Upon inspection of the DOE profiles, the ratios of the grating periods to the free-space wavelength are approximately 8, 4, 2, 1.4, 1.04, and 0.85, corresponding to peak separations of 25, 50, 100, 150, 200, and 250 μm, respectively.

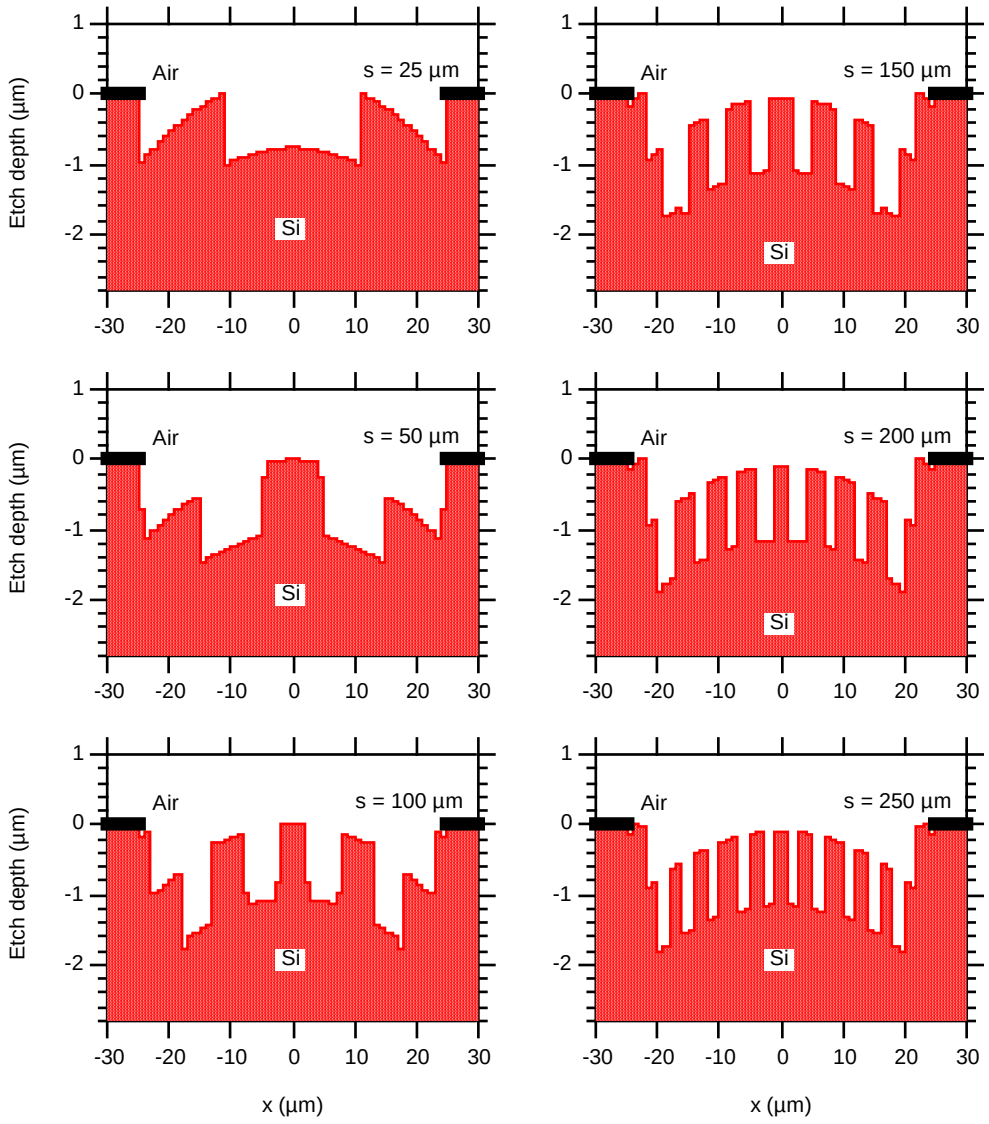

Figure 6.2 IASA DOE profiles.

A rigorous electromagnetic analysis based on the FDTD method as discussed in Chapter 4, was performed for both the TE and TM cases to determine the success of IASA. Figure 6.3 shows the calculated irradiances in the observation plane for the scalar-predicted result and TE and TM results obtained via FDTD for peak separations of 25, 50, 100, 150, 200, and 250 µm, respectively.

Note that for the closer peak separations, 25, 50, and 100 μm, the AS scalar results from IASA agree very well with those predicted by rigorous electromagnetic theory. Given that the observation plane is 340 μm from the silicon-air interface, this would correspond to angular peak separations of 4, 8, and 17 degrees. For the larger peak separations, 150, 200, and 250 μm, corresponding to angular separations of 25, 33, and 40 degrees, respectively, the deviation of IASA results from those predicted by FDTD becomes more significant. This may indicate that IASA works well provided that the angular spread of the near-field pattern in the observation plane is not too large (i.e., not greater than perhaps 20 degrees). In terms of the spatial period, Λ, it would again appear that AS scalar diffraction theory is accurate when $\Lambda \geq 2\lambda_0$.

The diffraction efficiencies and errors, as defined by Equations 5.1-5.3, are shown in Table 6.1.

Table 6.1 Diffraction efficiencies and errors for 1-2 beamfanners designed via IASA.

Peak separation (μm)	DE-scalar (%)	DE-TE (%)	DE-TM (%)	TE error (%)	TM error (%)	TE-TM difference (%)
25	0.7978	0.79604	0.78198	0.22024	1.9829	1.7666
50	0.76081	0.74491	0.72821	2.0893	4.2847	2.2422
100	0.74797	0.7235	0.69976	3.2719	6.4461	3.2816
150	0.78564	0.59971	0.69721	23.666	11.257	16.257
200	0.73737	0.49471	0.67428	32.91	8.556	36.3
250	0.67837	0.38871	0.55754	42.7	17.812	43.435

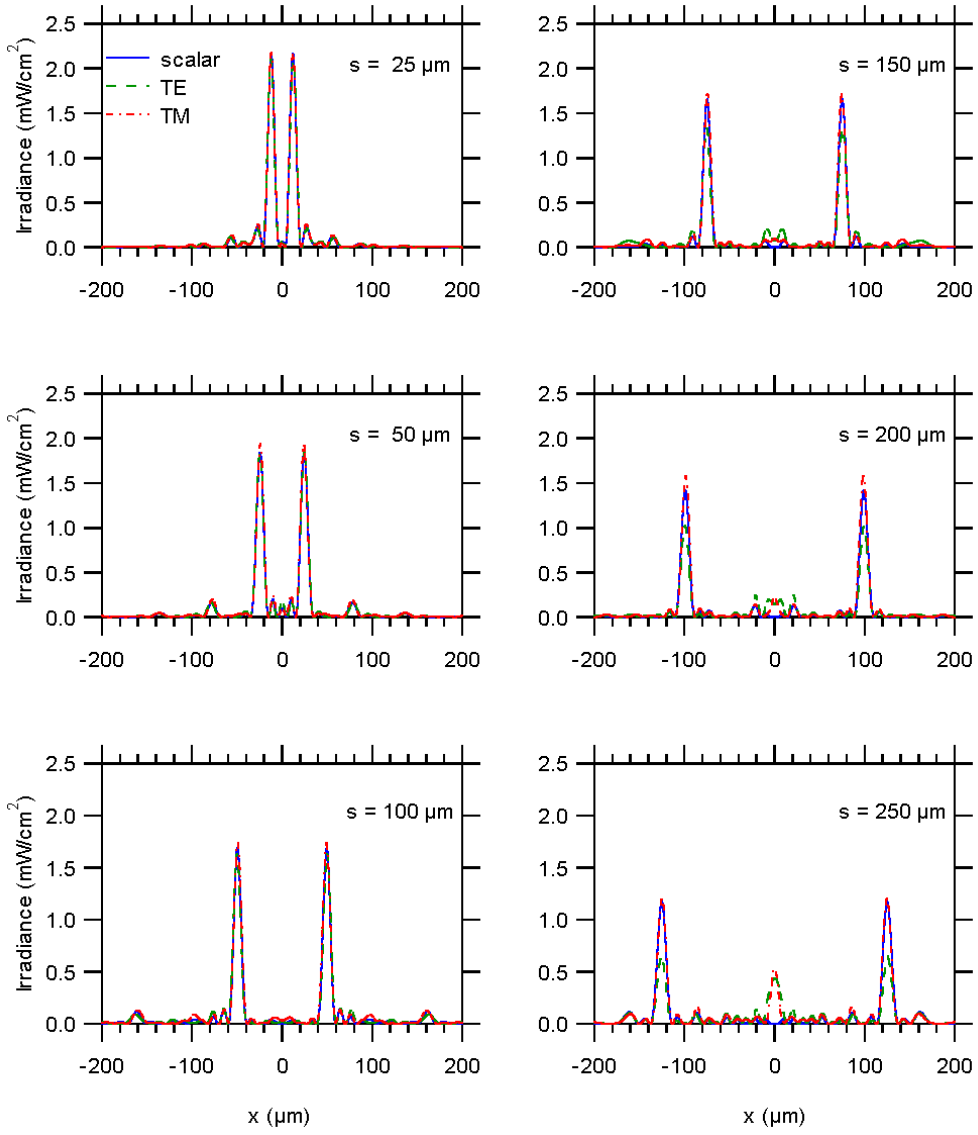

Figure 6.3 Irradiances for IASA profiles.

Note that the errors for TE and TM illumination with respect to scalar results are less than 7 percent for cases in which the peak separations are 25, 50, and 100 μm. In these cases, IASA is effective in designing the DOEs for these functions. For larger peak separations, however, the errors become significant, indicating that the validity of AS scalar theory breaks down in these cases and IASA is less successful. Also, the TE error is less than the TM error for the smaller

peak separations while the TM error is less than the TE error for the larger peak separations. The reasons for this are not yet understood.

6.2.3 1-3 and 1-4 beamfanner designs

The advantage to using IASA over a heuristic method is that the former does not require prior knowledge of the form of the DOE profile. Although, 1-2 beamfanner designs can be done by either heuristic means or numerically, for example, with IASA, this application falls into a very limited class of diffractive optic design problems. For example, designs for a 1-3 and a 1-4

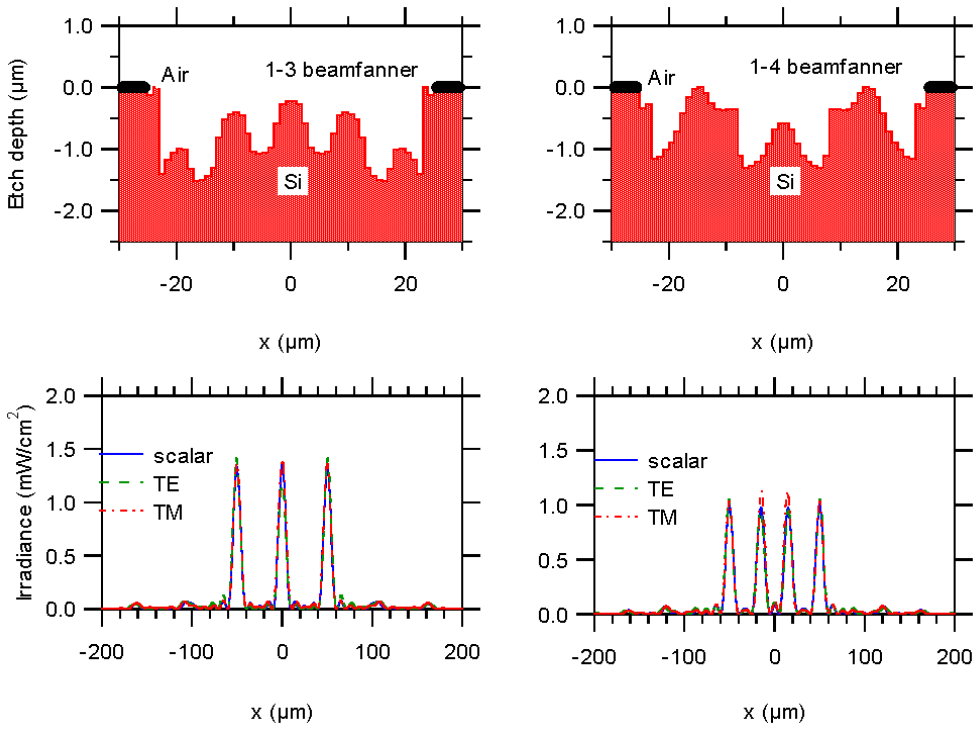

Figure 6.4 IASA: Multiple peak beamfanners.

Table 6.2 Diffraction efficiencies and errors for 1-3 and 1-4 beamfanners.

Number of peaks	DE-scalar (%)	DE-TE (%)	DE-TM (%)	TE error (%)	TM error (%)	TE-TM difference (%)
3	83.031	81.262	81.208	2.1305	2.1955	0.06642
4	82.247	81.444	81.449	0.97571	0.96965	0.006123

beamfanner cannot be done heuristically, and must be done numerically. Note that IASA, in general, could also be used for beamfanner designs for even greater number of diffracted peaks, but in this work the designs are restricted to the following 1-3 and 1-4 beamfanners.

Consider an air-silicon interface as shown in Figure 6.1, in which silicon is the exiting medium and consider an observation plane located 340 μm from the interface in the silicon. Let the finite aperture be 50 μm wide and consider 1 μm minimum feature sizes. For the designs of 1-3 and 1-4 beamfanners, with desired peak separations of 50 μm and 33.3 μm, respectively, the weighting functions are very similar to the 1-2 beamfanner designs with the difference that the field amplitudes are equally maximized at more locations. Figure 6.4 shows the resultant DOE profiles with IASA and the corresponding near-field diffraction patterns in the observation plane. Table 6.2 lists the diffraction efficiencies calculated from AS scalar theory and for FDTD TE and TM cases as well as the previously defined errors. It appears that the use of IASA is, in fact, accurate since there is excellent agreement between the scalar and rigorous results.

6.3 Comparison between FDTD and BEM

In the development of BEM and FDTD, the example of a dielectric cylinder was used to test the implementation of these rigorous methods. For each method, the results agreed identically with the analytic, theoretical predictions. However, no comparison has been made, so far, to compare the performance of FDTD versus BEM for finite aperture applications.

Figure 6.5 and Figure 6.6 show the results obtained using both FDTD and BEM for the beamfanners designed in this chapter for the TE and TM cases, respectively. Note that there is excellent agreement between FDTD and BEM for the TE cases for devices that yield peak separations of 25, 50 and 100 μm in the observation plane. However, there is some deviation of the BEM results from those obtained via FDTD for larger peak separations. For the TM case, the BEM results match quite well in comparison with FDTD for peak separations of 25, 50, 100, 150 and 200 μm while for the 250 μm case, BEM appears to break down.

In the comparison of FDTD and BEM, 600 sample points were taken along the DOE contour for the BEM analysis. Presumably, if fewer points are taken, then the results obtained via BEM should be even worse. Figures 6.7 and 6.8 show the comparison between FDTD and BEM if only 300 points are taken using BEM along the DOE contour for the TE and TM cases, respectively. Note that the FDTD data is the same as in Figures 6.5 and 6.6.

For the TE case with only 300 sampled BEM points, the disagreement is apparent even for a peak separation of 100 μm, as shown in Figure 6.7. For larger peak separations, BEM fails more drastically in comparison with FDTD. As shown in Figure 6.8, the breakdown of BEM occurs for peak separation of 25 μm and progressively worsens for larger peak separations for the TM case.

The explanation for the apparent discrepancy between BEM and FDTD is that the DOE profiles that yield large peak separations have many more abrupt edges as shown in Figure 6.2. In the implementation of BEM, more sampled points must be taken along those edges. Considering that the computational burden increases as the square of the number of sample points, BEM tends to break down if an insufficient number of sample points is taken. On the other hand, the computer memory constraint of FDTD only increases linearly with respect to the number of points within the computational lattice. However, in principle, if a sufficient number of sample points are taken, results from BEM will agree identically with those from FDTD.

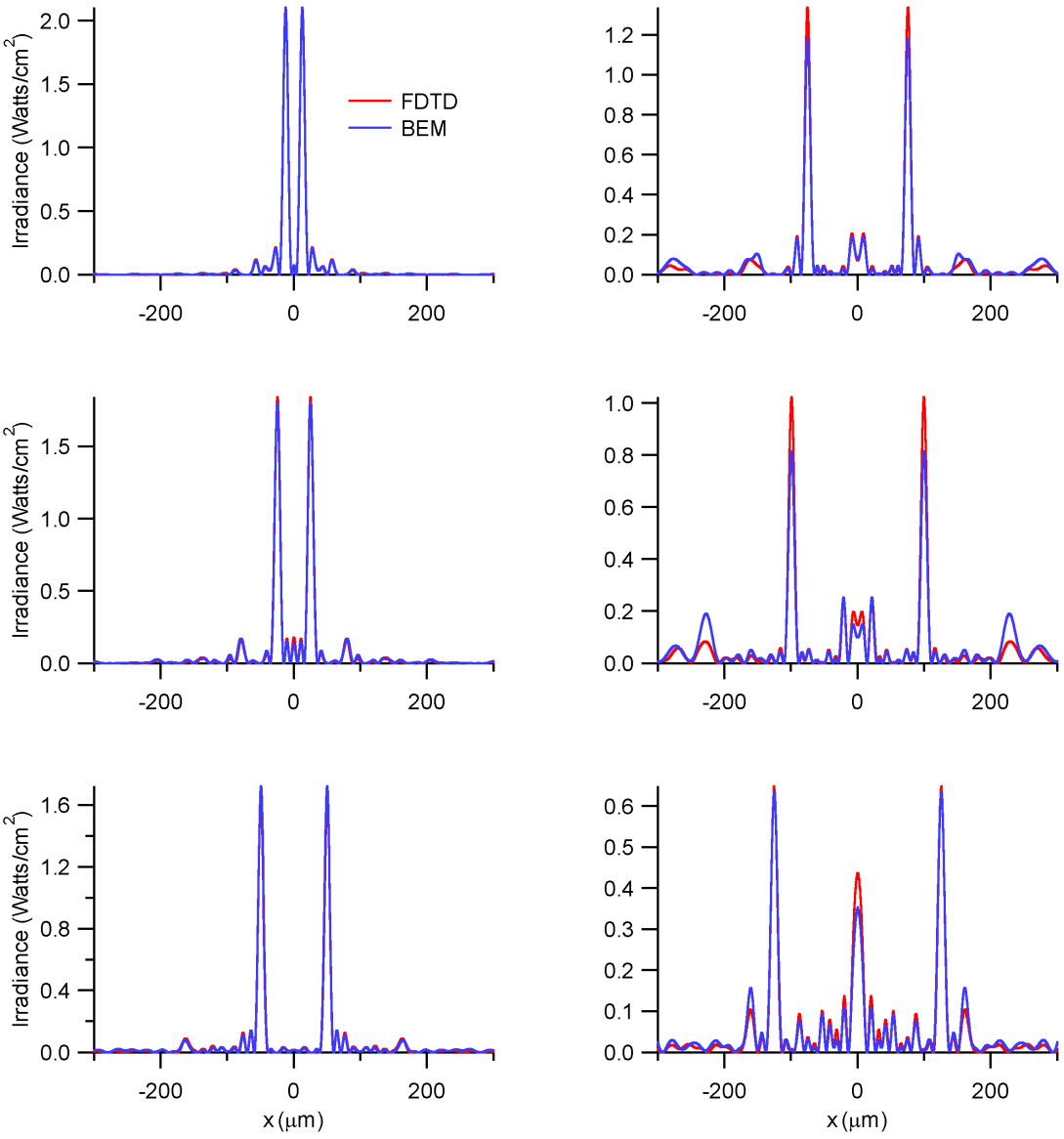

Figure 6.5 FDTD-BEM comparison for TE IASA results.

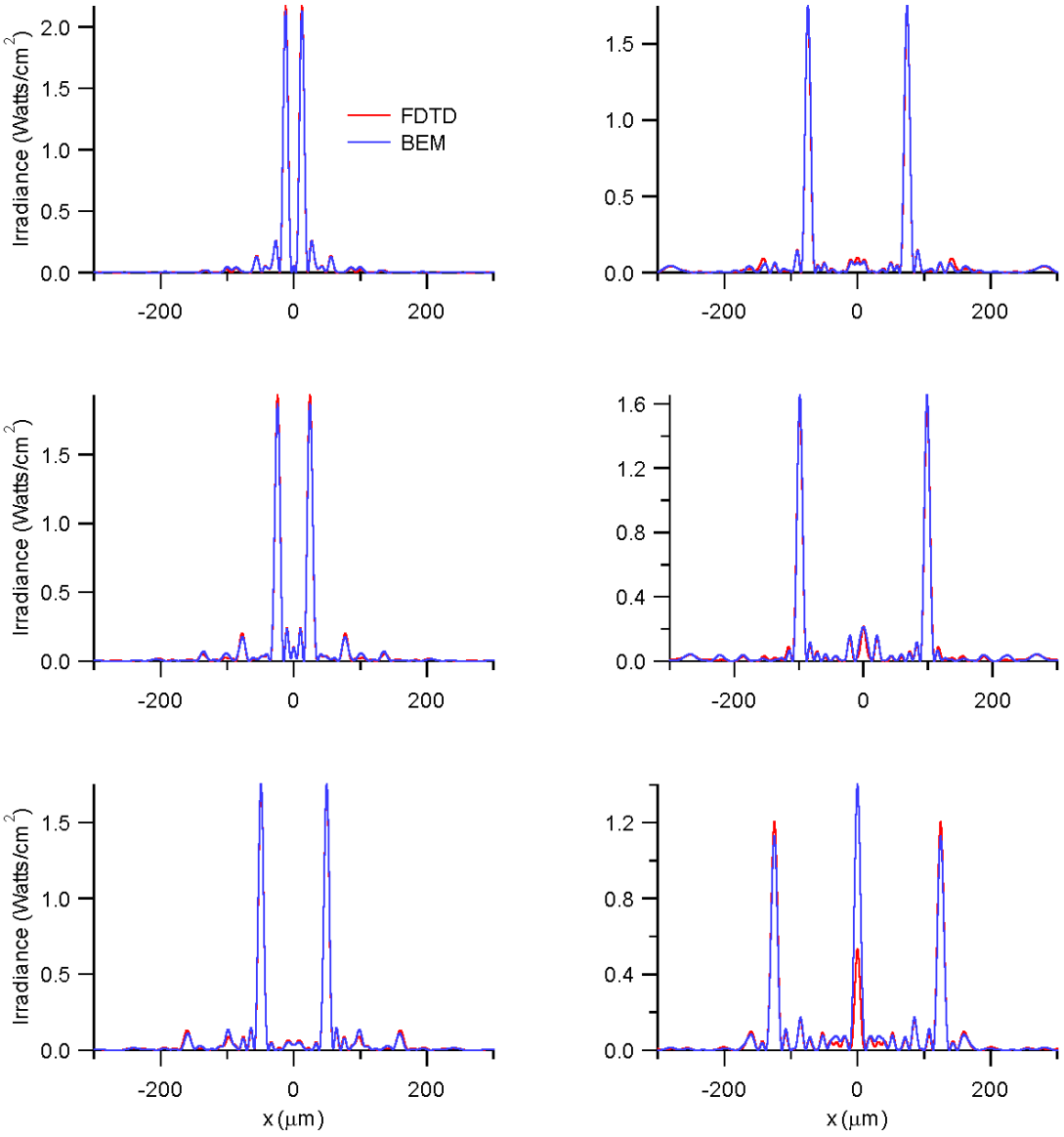

Figure 6.6 FDTD-BEM comparison for TM IASA results.

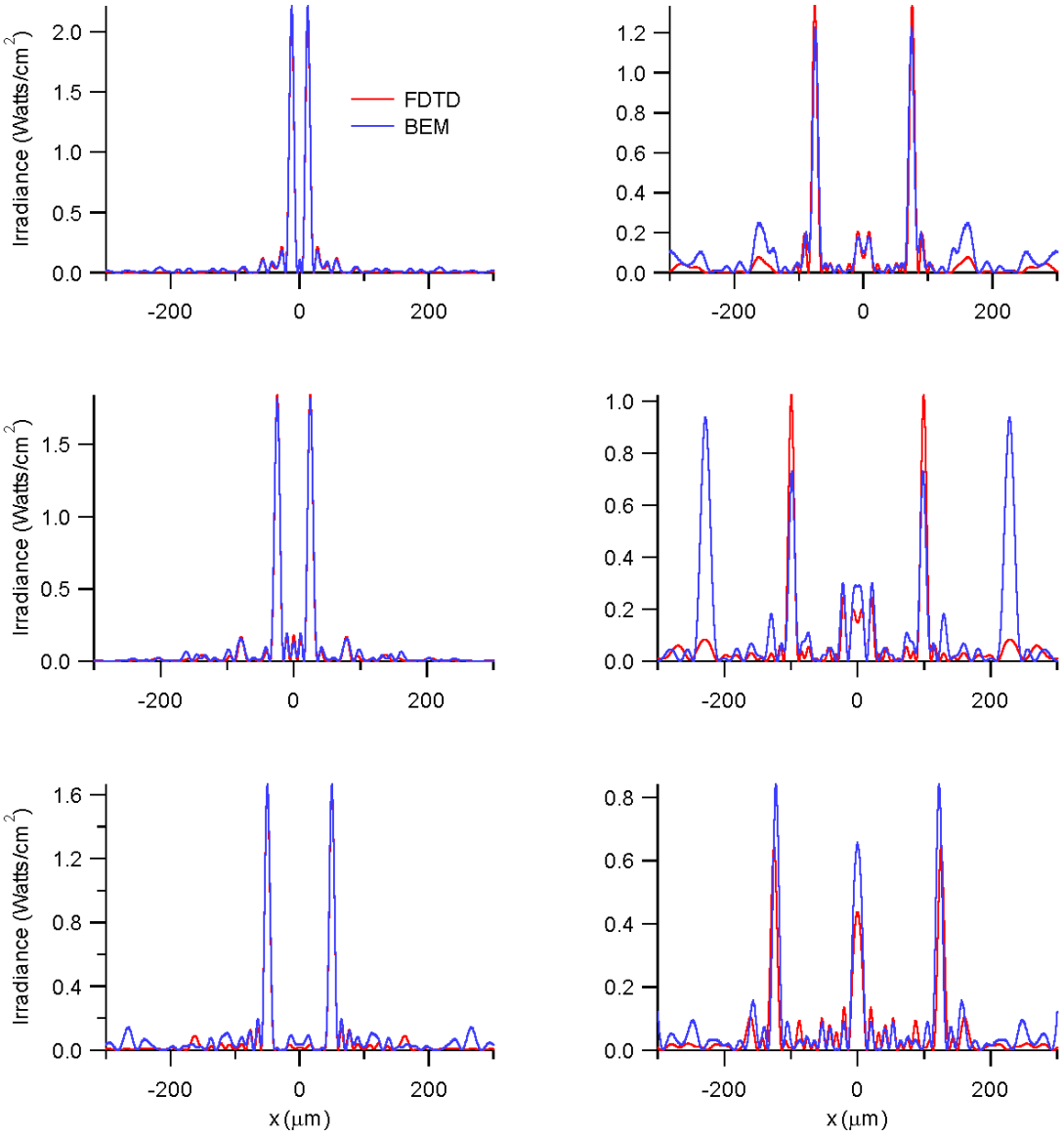

Figure 6.7 TE case with fewer BEM sample points.

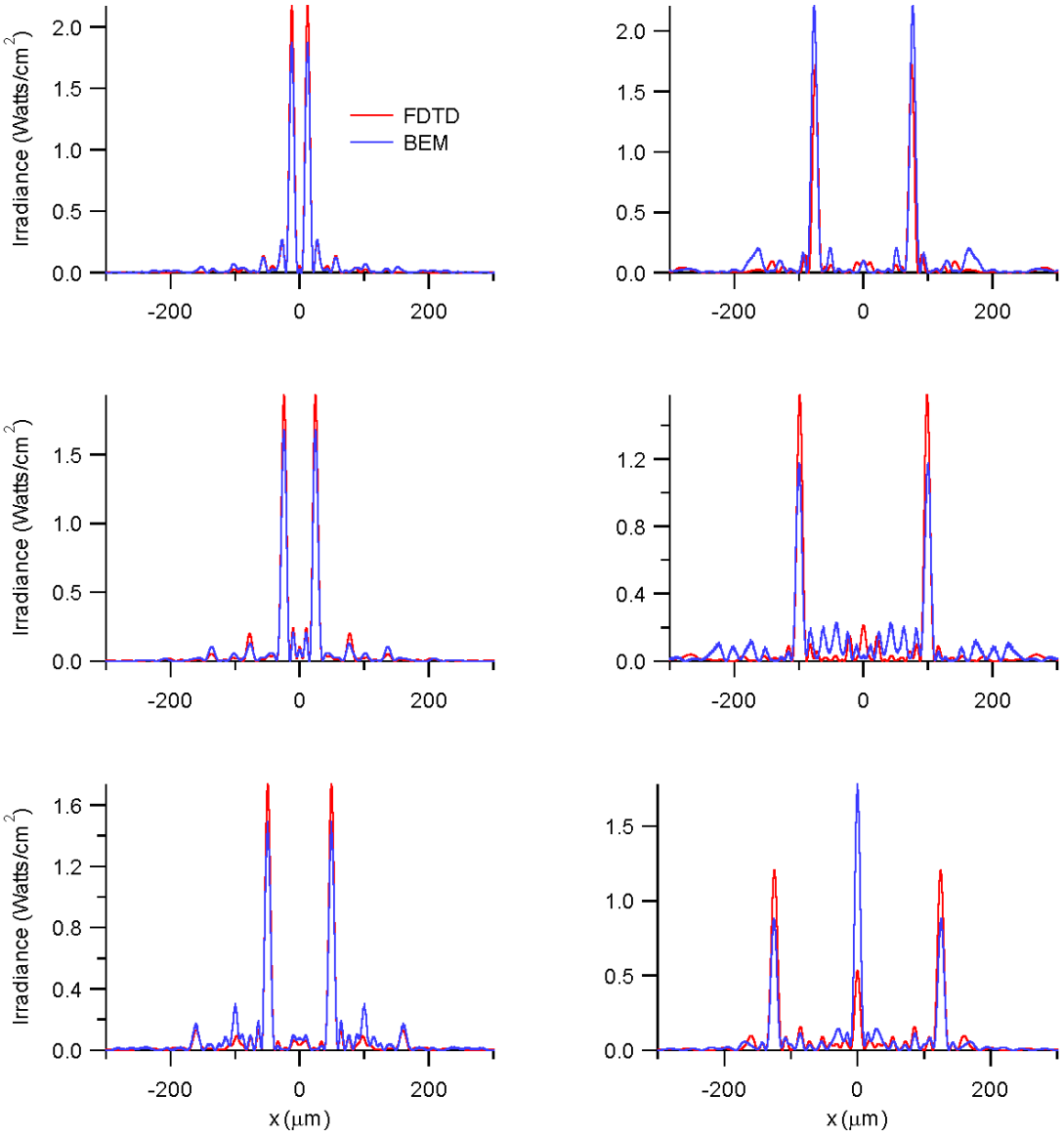

Figure 6.8 TM case with fewer BEM sample points.

6.4 Re-sampling and quantization

A re-sampling technique to increase the DOE minimum feature size used in conjunction with IASA has been developed. A re-sampling technique is introduced as a post-processing step after using IASA to increase the DOE minimum feature size so that the designed DOEs are easier to fabricate. In the case of the 1-2 beamsplitter design that produces two peaks separated by 25μm in the observation plane, for example, results appear to indicate that a DOE design obtained by IASA followed by the proposed re-sampling works rather well, as will be shown in this section.

Re-sampling requires that sampled points along the DOE thickness profile are grouped together and locally averaged thereby increasing the DOEs minimum feature size. Also, the etch depth levels are reset in such a way that the minimum etch depth is zero. This is done for every re-sampled case presented in this chapter. The FDTD method is used to assess the validity of the scalar-based designed DOEs with increased feature sizes using re-sampling. The reason for this is that the feature sizes and zones may still be of the same order as the wavelength of illuminating light. Also examined in this section is the effect of quantizing the DOE etch depth levels. The reason for examining this effect is due to fabrication restraints. In practical applications, DOE etch depth levels are generally quantized.

For the specific case of the 1-2 beamfanner with 1μm minimum feature size bounded by a 50 μm aperture, the effects of quantizing the etch depth levels are examined for 32, 16, 8, 4, and 2 discrete and equally spaced levels. The effect of re-sampling this DOE to 2, 5, and 10 μm minimum feature sizes is also examined.

Figures 6.9 through 6.13 show the quantized versions the original DOE designed via IASA for 32, 16, 8, 4, and 2 etch depth levels respectively along with the corresponding observation plane intensities calculated via AS scalar theory and FDTD. As shown in Table 6.3, the diffraction efficiency as defined in Equation 5.1 does not change appreciably even for as few

as 8 etch depth levels. From this, one may conclude that quantizing the etch depth levels has little effect on DOE performance.

As an aside, note that all scalar-based results were compared to those obtained using FDTD as shown in Figures 6.9 through 6.13 for the irradiance profiles. Also, Tables 6.4 and 6.5 show the diffraction efficiencies calculated via FDTD for the TE and TM cases, respectively. Note that the scalar results agree almost identically with the rigorous results. This would indicate that scalar-based analysis is indeed valid for all the cases presented in this section.

Also, re-sampling the DOE profile to as high as 5 μm minimum feature sizes yields diffraction efficiencies comparable to the case without re-sampling, as shown in Table 6.3. From this, it may be concluded that re-sampling is a valuable tool after using IASA to design DOEs with high diffraction efficiencies and larger minimum feature sizes.

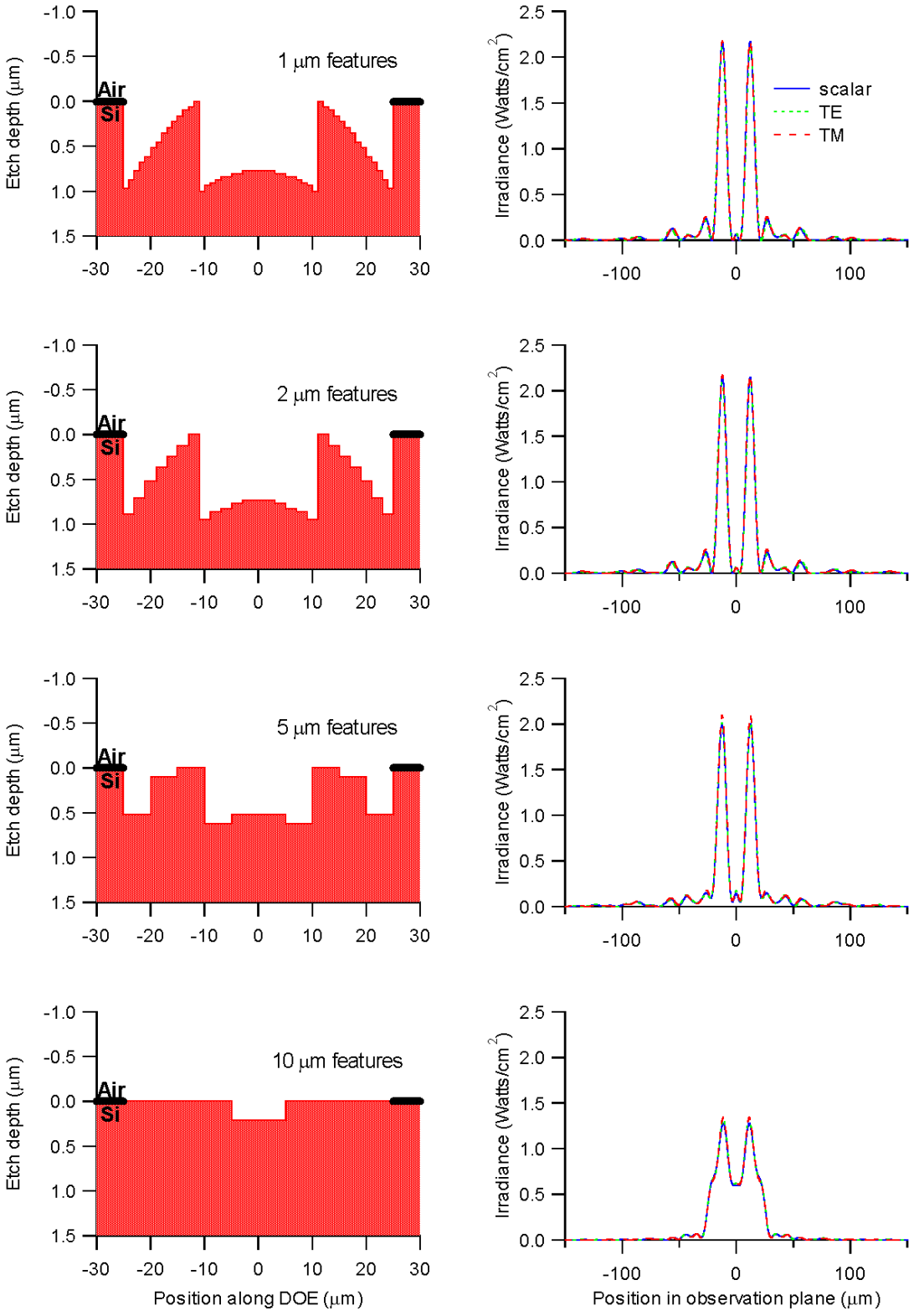

Figure 6.9 DOE with 32 etch depth levels.

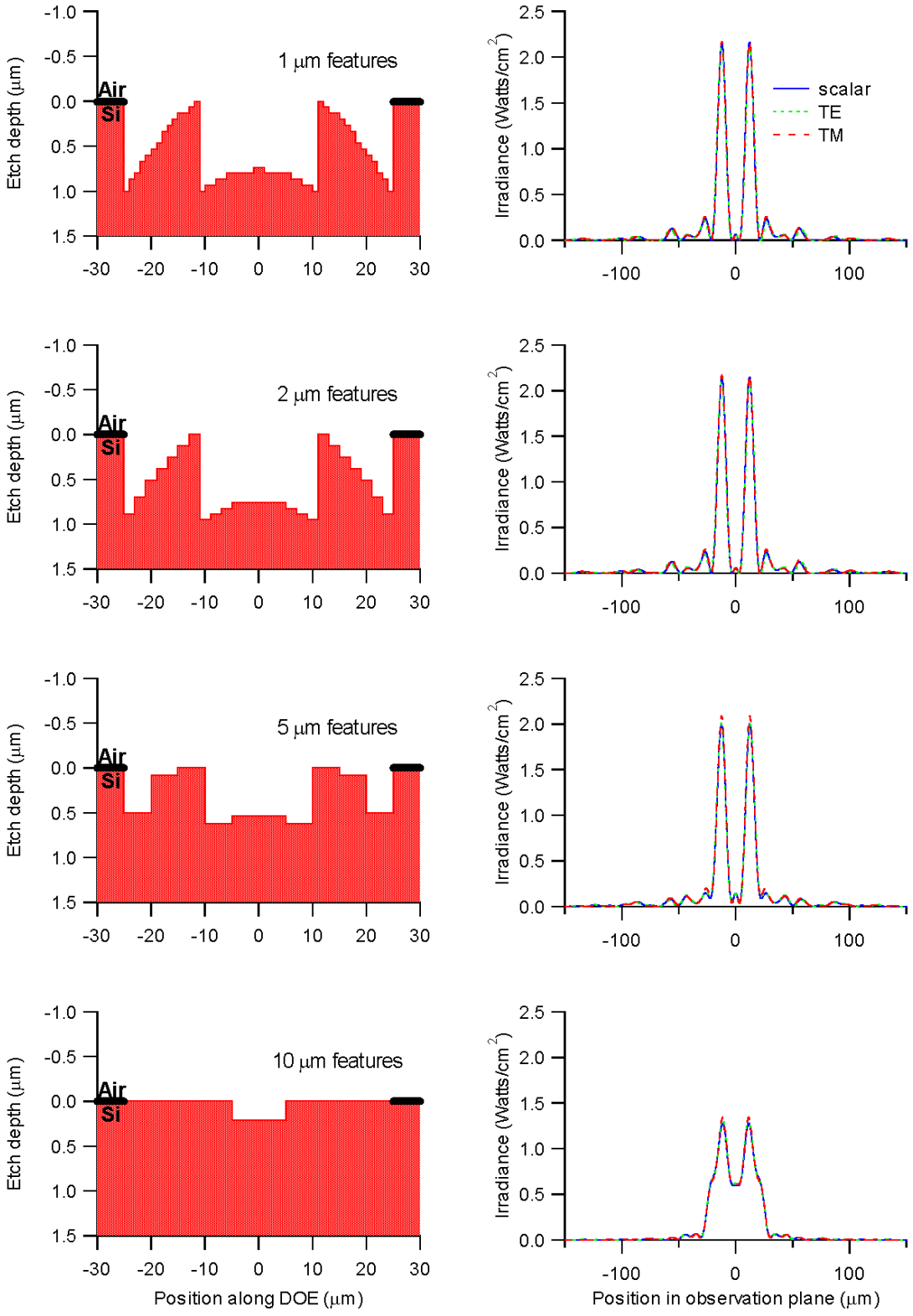

Figure 6.10 DOE with 16 etch depth levels.

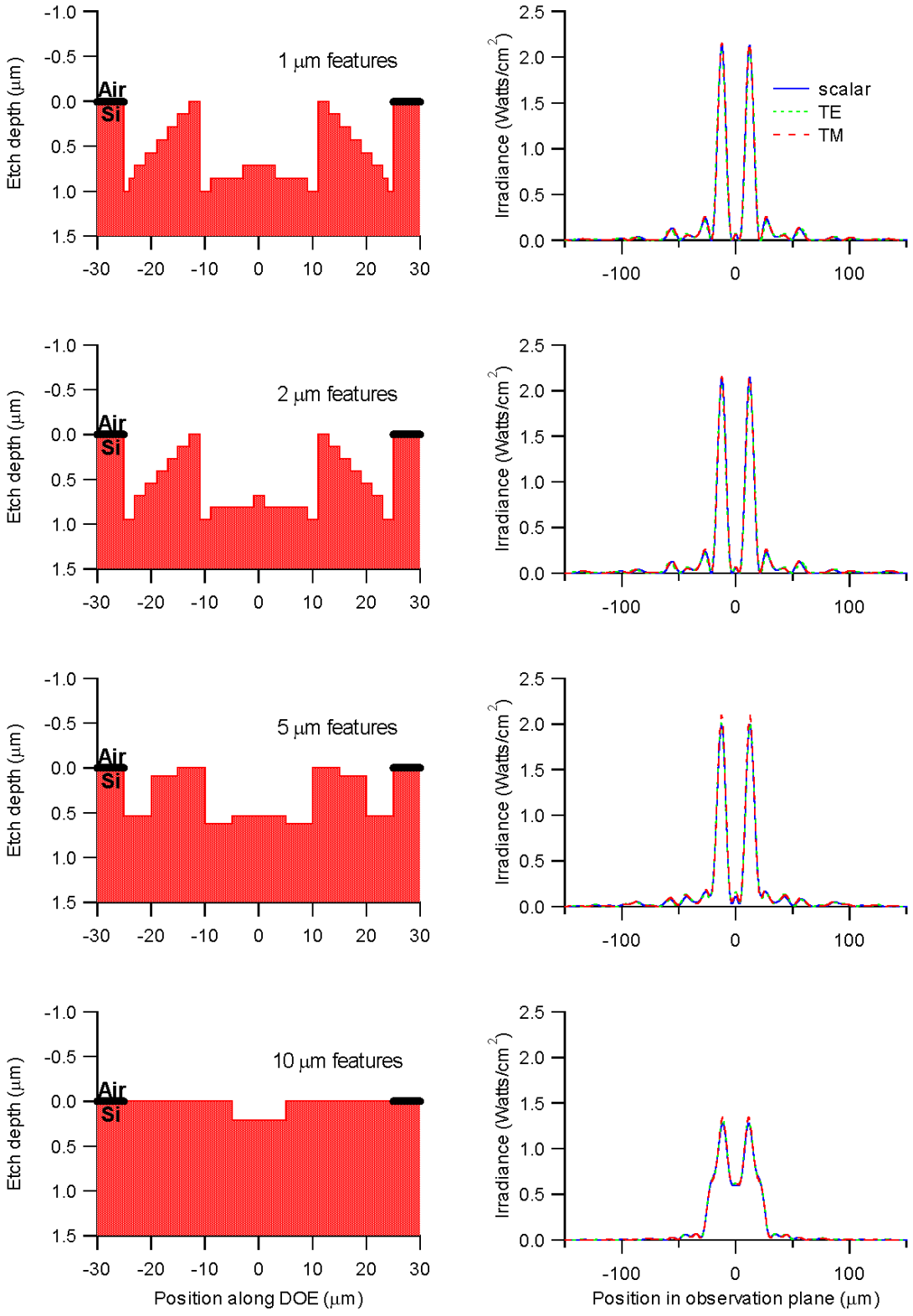

Figure 6.11 DOE with 8 etch depth levels.

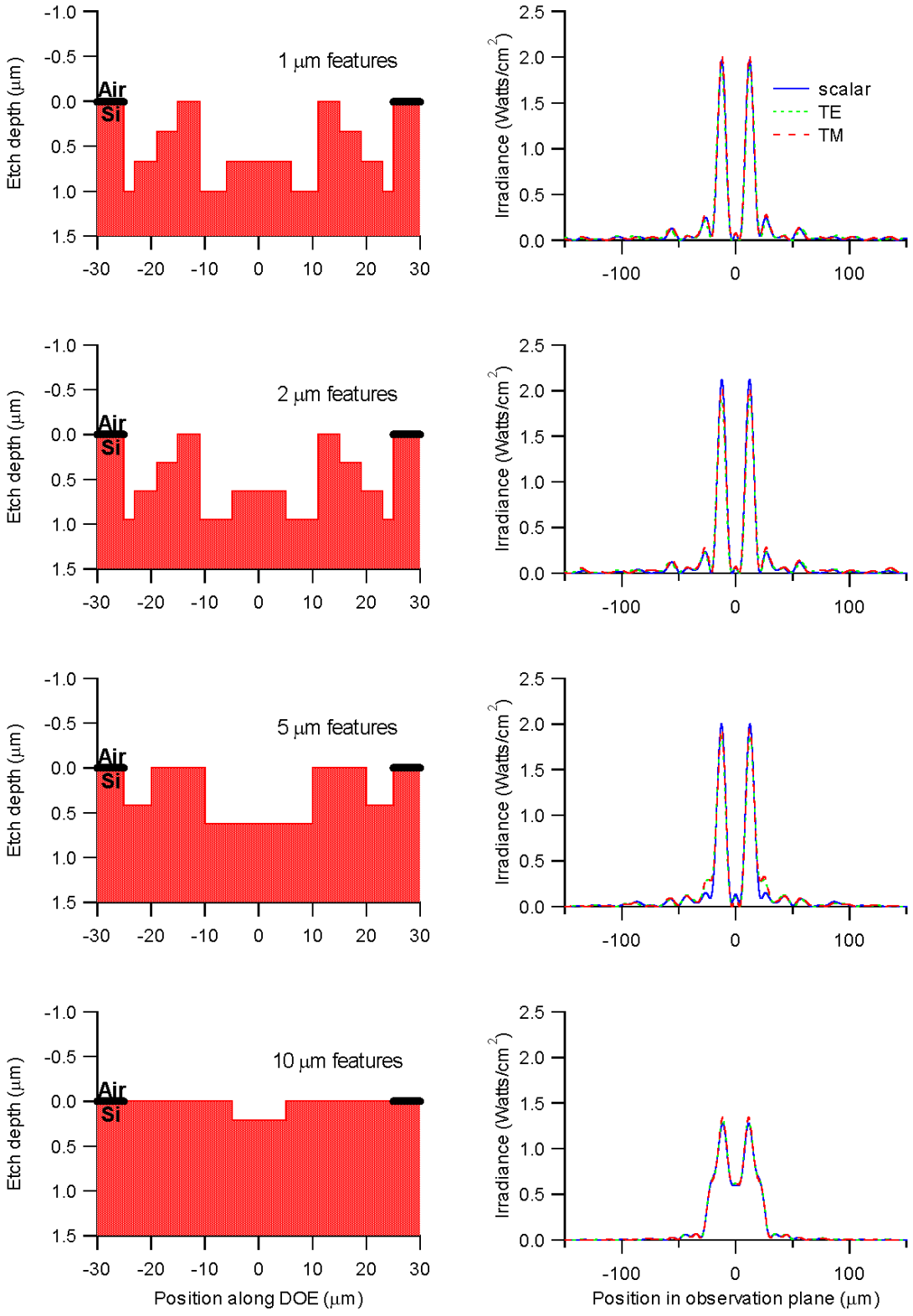

Figure 6.12 DOE with 4 etch depth levels.

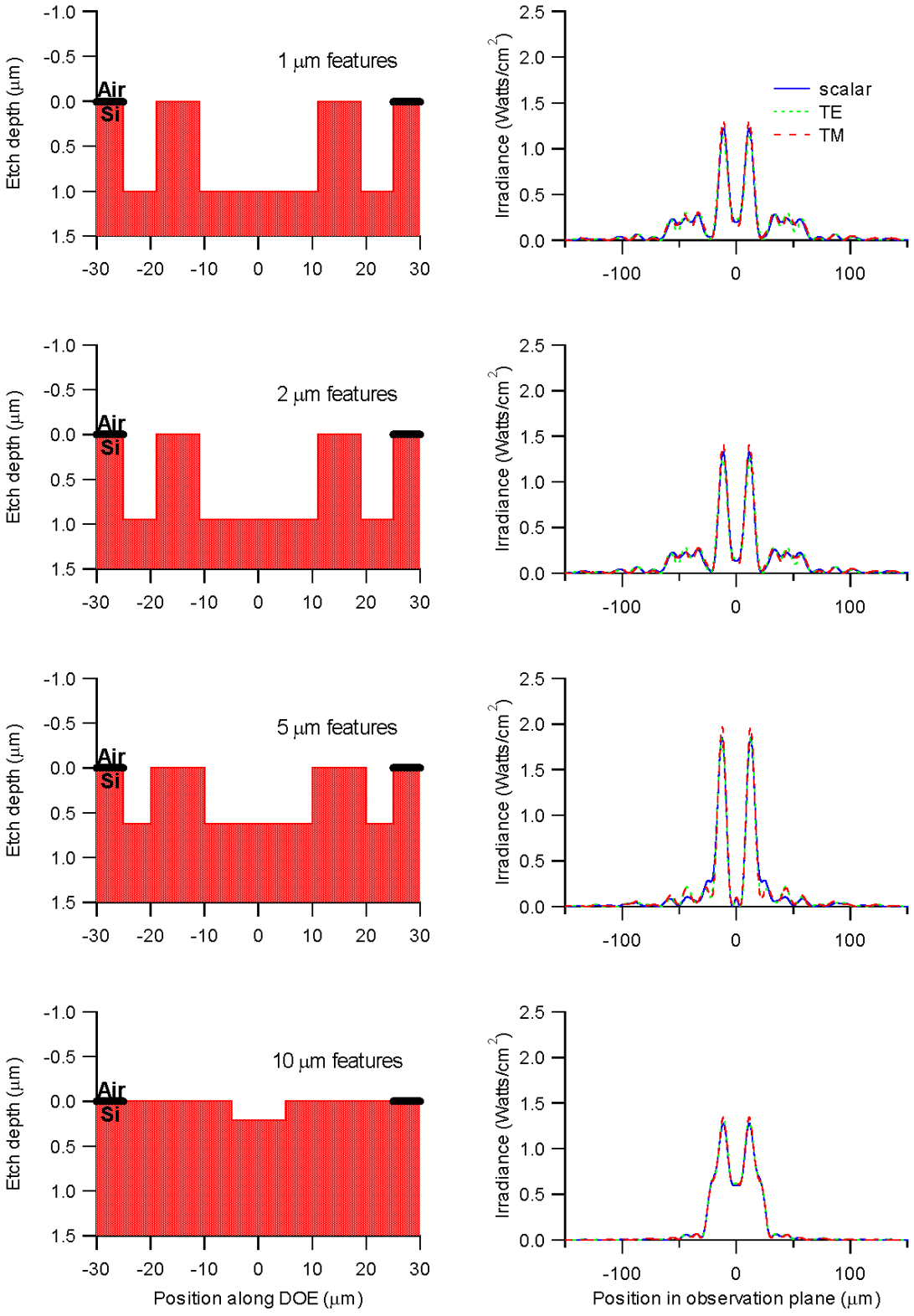

Figure 6.13 DOE with 2 etch depth levels.

Table 6.3 Scalar diffraction efficiencies with re-sampling and quantization.

Minimum feature size (μm)	DE (%) 32 levels	DE (%) 16 levels	DE (%) 8 levels	DE (%) 4 levels	DE (%) 2 levels
1	79.77	79.56	78.69	73.95	50.24
2	79.51	79.43	78.74	73.62	53.45
5	77.06	76.92	76.84	72.37	71.62
10	64.18	64.18	64.18	64.18	64.18

Table 6.4 Diffraction efficiencies with re-sampling and quantization for TE case.

Minimum feature size (μm)	DE (%) 32 levels	DE (%) 16 levels	DE (%) 8 levels	DE (%) 4 levels	DE (%) 2 levels
1	0.7965	0.7927	0.7834	0.7249	0.4952
2	79.30	79.22	78.02	72.39	52.94
5	76.70	76.64	76.46	72.17	70.94
10	64.51	64.51	64.51	64.51	64.51

Table 6.5 Diffraction efficiencies with re-sampling and quantization for TM case.

Minimum feature size (μm)	DE (%) 32 levels	DE (%) 16 levels	DE (%) 8 levels	DE (%) 4 levels	DE (%) 2 levels
1	78.24	78.01	77.06	72.12	51.28
2	77.48	77.46	76.88	71.93	54.36
5	76.25	76.21	76.06	72.07	71.10
10	64.97	64.97	64.97	64.97	64.97

6.5 Applicability of IASA

In summary, IASA is a novel design tool for finite aperture DOE design with features on the order of or less than the wavelength of illuminating light. Regardless of whether the exiting medium is air or silicon, it appears that the AS scalar model is a viable method provided that the

spatial period, Λ, is not much less than twice the free-space wavelength, i.e., $\Lambda \geq 2\lambda_0$. The presented iterative design method, IASA, is surprisingly accurate for designing finite aperture DOEs with sub-wavelength features. Also, IASA can be used to design finite aperture DOEs which cannot be designed heuristically, such as 1-3 and 1-4 or higher-order beamfanners.

Chapter 7

DIFFUSERS FOR THREE-DIMENSIONAL DISPLAYS

7.1 Motivation

The motivation for designing a vertical diffuser stems from the research done at UAH on a three-dimensional autostereoscopic display based on the partial pixel architecture [46,47]. The geometry of the partial pixel architecture is shown in Figure 7.1. It consists of a pixelated display and a well-defined viewing region located at a distance d_v from the display. The viewing region is partitioned into a series of virtual viewing slits, where each slit is approximately one pupil diameter wide. A unique two-dimensional image is visible from each virtual viewing slit. When an appropriate pair of images, that is, a stereoscopic pair, is simultaneously viewed by the left and the right eyes, the scene appears to be three-dimensional. Horizontal motion parallax is provided by displaying multiple stereo-image pairs [46].

The basic geometry of the three-dimensional display is shown in Figure 7.1. The pixilated display is located in the x-y plane, and a viewing region is located a distance d_v from the plane. One may think of the viewing region as a series of adjacent virtual viewing slits that are approximately one pupil diameter wide. As with holographic stereograms [46,47], each eye of the observer sees a different image on the display, as shown in Figure 7.2. When the appropriate set of stereopair images is presented on the display in which a single image visible through each virtual viewing slit, the image appears three-dimensional. Furthermore, the display exhibits one-dimensional parallax, as an observer moves his or her head from side to side within the viewing

region. The display is called autostereoscopic since no special headgear is required for its viewing.

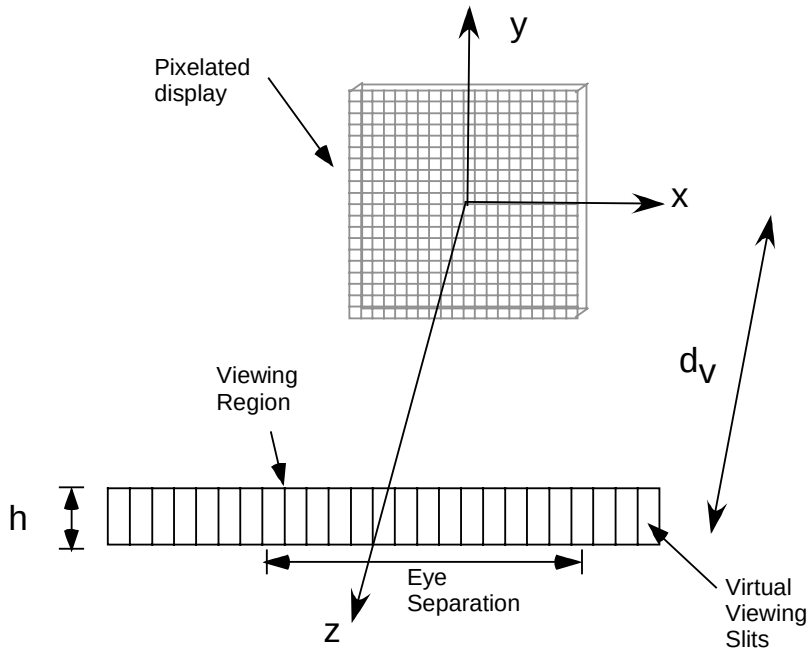

Figure 7.1 Three-dimensional display geometry.

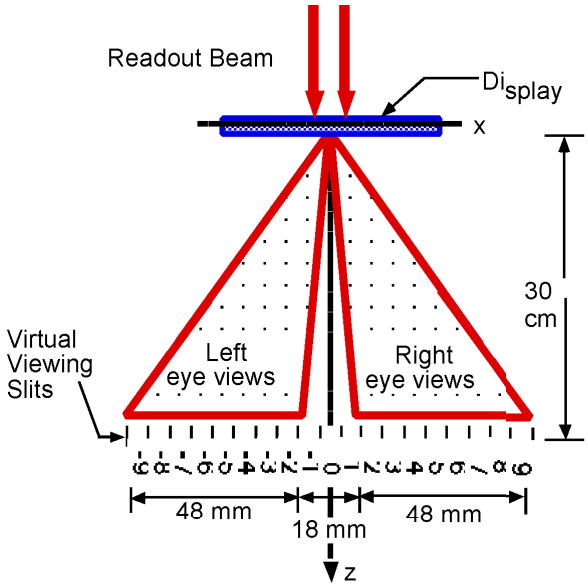

Figure 7.2 Left and right eye views of autostereoscopic display.

The geometry of a single pixel of the display is shown in Figure 7.3. A single pixel consists of an array of partial pixels that are each composed of an LCD pixel and a DOE grating, as shown in Figure 7.4. The function of each partial pixel is to direct light to its corresponding virtual viewing slit in the observation region. This produces an autostereoscopic pair of a single part of an image. In conjunction with all other single pixels, the total image may be observed in the viewing region.

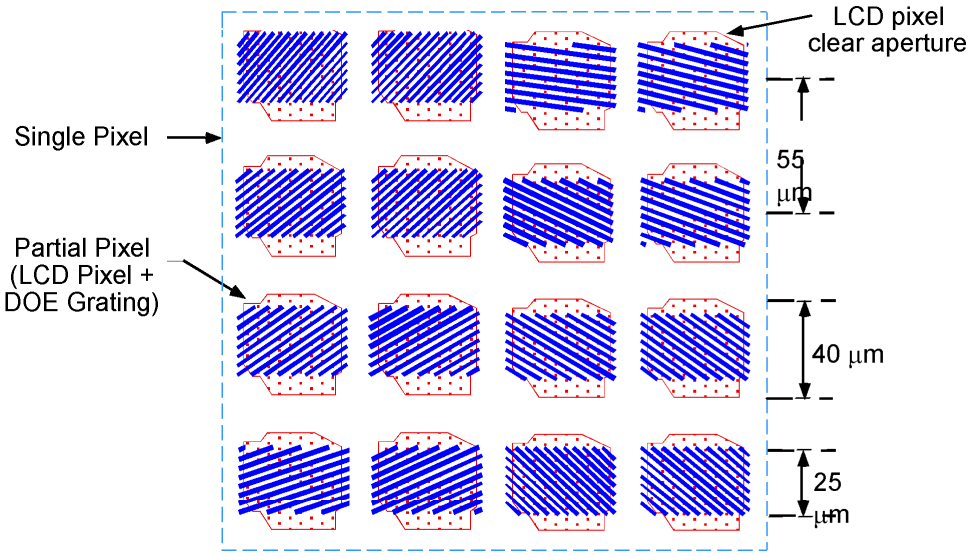

Figure 7.3 Geometry of a single pixel.

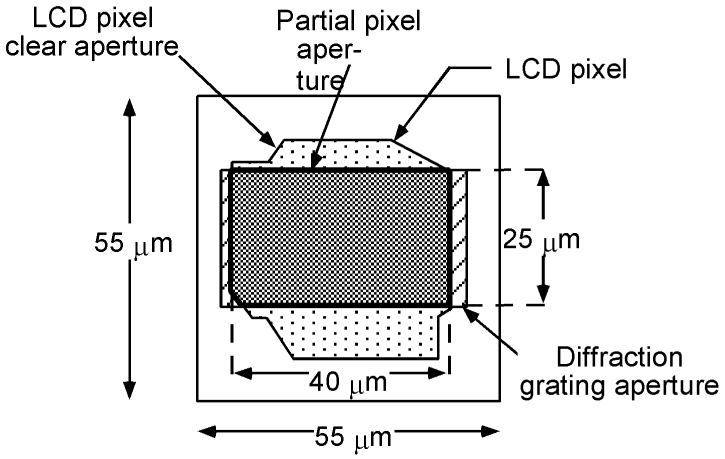

Figure 7.4 Geometry of a partial pixel.

The readout geometry was such that all diffraction orders except the appropriate +1 orders fell outside the region, as shown in Figure 7.5. Therefore, no cross talk exists with the virtual viewing slits due to higher diffraction orders from any of the gratings. The height of the virtual viewing slits is physically defined by the diffraction from the aperture of each partial pixel [46,47]. Note that the particular configuration shown in Figure 7.5 is for a monochrome display developed at UAH.

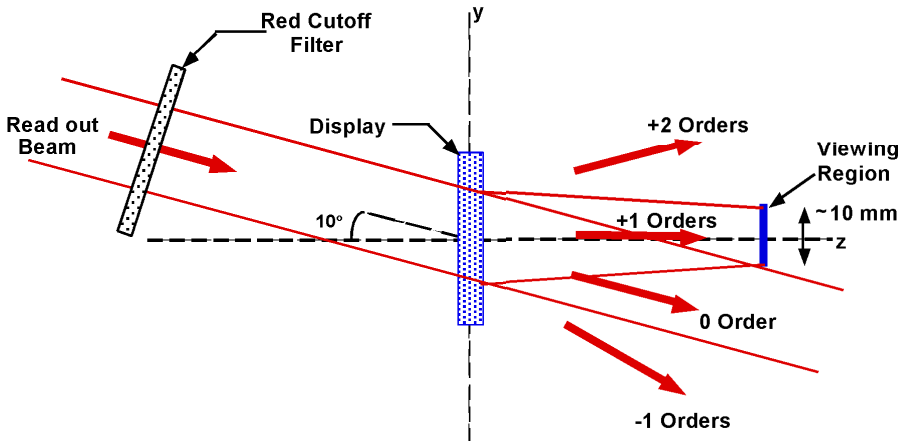

Figure 7.5 Schematic of readout geometry.

Several options exist to increase the height of the virtual viewing slits. The motivation for doing this is that the height of the virtual viewing slits is often too small for an observer to view the scenes comfortably and move his or her head freely in the vertical direction since the height of the virtual viewing slits is only 10 mm for the UAH display. It is important that when extending the height of the virtual viewing slits, the intensity is uniform in the vertical direction. Also, since the partial pixel architecture diffracts several orders, the zeroeth order must not overlap the first order diffracted beam since the latter contains the information of interest. One

potential remedy is to reduce the size of the grating in the partial pixel architecture to expand the height of the virtual viewing slit. The problem with this approach is that the partial pixels incorporate a liquid crystal display in conjunction with a grating. A liquid crystal display with smaller pixels may not readily be commercially available to achieve this. Therefore, one must find an alternate approach. Another possible remedy is to insert another optical element consisting of an array of narrow horizontal slits after the pixelated display. Properly aligned, this element would diffract the light emanating from each row of the pixelated display thereby increasing the height of the virtual slits. The problem with this approach is that by using such a small aperture as a diffuser, a large fraction of the light would be blocked, thus reducing the overall power throughput of the system.

A vertical diffuser could also be constructed using a Holographic Optical Element (HOE) [7,48]. First, a master hologram is constructed by recording the image of a ground glass screen with a locally random thickness profile. This master hologram is then illuminated by a narrow slit of light and the real image of the ground glass is transferred to a second plate. This second plate could then serve as a vertical diffuser.

Diffractive optics technology offers the designer another method to design a vertical diffuser to uniformly expand the virtual viewing slits by tailoring an array of DOEs to perform this task. Given that the height of the virtual viewing slits (10 mm) is too small, the goal of the diffuser design is to expand the viewing region as large as possible without introducing cross talk between the +1 order and all other orders. A diffuser that extends the height of the viewing region to approximately 50 mm will accomplish this. As will be presented, an iterative algorithm is used to design an array of finite aperture DOEs that extends the height of the virtual viewing slits uniformly along the vertical direction.

For the monochromatic display developed at UAH, a flashlight is used as the illuminating source followed by a spectral filter transmitting only red light. For the color display developed at UAH, three different filters are used in the set-up. The filters are red, green, and blue. All

diffuser designs must consider the wavelength of light in question. In this chapter, only red light illumination is considered in the designs assuming λ=630 nm with a 10 nm bandwidth. However, the principles of the algorithm are applicable to green or blue light illumination (440±10 nm and 550±10 nm, respectively) as in the case of the color display developed at UAH [46,47].

7.2 Diffractive optic design using the Fraunhofer or Fresnel approximation

In this section, the basics of finite aperture DOE design using a modified Iterative Fourier (or Fresnel) Transform Algorithm (IFTA) are discussed [7]. The first step is to choose the form of the object profile of the DOE. The next step is to iteratively calculate the field in the image plane while modifying this field according to a prescribed weighting function. An illustrative example of designing diffusers for an autostereoscopic display is presented in the following section. Considerations of designing and analyzing finite aperture DOEs assuming only that the Fresnel approximation is valid are also discussed.

Figure 7.6 illustrates a general one-dimensional IFTA for designing a finite aperture DOE for a given application using scalar diffraction theory and the Fraunhofer approximation. It is assumed that the DOE is illuminated by a plane wave of unit amplitude. The first step of the algorithm is to choose some initial DOE thickness profile. The initial choice for any DOE thickness profile must meet certain criteria. Specifically, the DOE is phase-only and is bounded by a specified finite aperture. The transmission function of such a DOE takes the form

$$t(x_1) = \exp[j\,\phi(x_1)]\,\mathrm{rect}[x_1/L]\,, \qquad \text{where} \quad \mathrm{rect}(x) = \begin{cases} 1 \text{ if } |x| \le \frac{1}{2} \\ 0 \text{ otherwise} \end{cases}, \tag{7.1}$$

in which $\phi(x_1)$ is the phase distribution within the object plane and the $\mathrm{rect}[x_1/L]$ term indicates that the DOE is bound by a finite aperture of width, L. Assuming that the DOE is illuminated by a plane wave of unit amplitude, the field in the object plane, $U(x_1)$, may be written as

$$U(x_1) = t(x_1) = \exp[j\,\phi(x_1)]\ \mathrm{rect}[x_1/L]. \tag{7.2}$$

Also, the phase function, $\phi(x_1)$, depends upon the DOE thickness profile, $d(x_1)$, and takes the form

$$\phi(x_1) = \phi(d(x_1)) = 2\,\pi/\lambda\,(n-n_0)\,d(x_1), \tag{7.3}$$

in which n and n_0 are the indices of refraction for the DOE material and air, respectively, and λ is the wavelength of illuminating light. In this dissertation, only monochromatic light is considered. The object profile then takes the form

$$U(x_1) = \exp[2\,\pi/\lambda\,(n-n_0)\,d(x_1)]\ \mathrm{rect}[x_1/L]. \tag{7.4}$$

The thickness profile, $d(x_1)$, is obtained via direct sampling in which a continuous thickness profile is assumed and its values are taken at specified intervals along the object plane axis, x_1 [16]. Again, direct sampling requires that the DOE samples are equally spaced along the x_1 axis to accommodate later steps of the design algorithm when calculating FFTs. Often, but not always, the values of the continuous thickness profile evaluated at the midpoints of the DOE partitions serve as the sampled thickness profile, $d(x_1)$. The specific choice of the initial profile will generally depend upon the application for which the DOE is intended.

As shown in Figure 7.6, the next step after having chosen some initial phase profile of the object is to calculate its Fourier transform using a FFT [49]. The square of the magnitude of the FFT is proportional the far-field intensity in the Fraunhofer region. Generally, it is the case that this intensity is not optimal according to some predetermined measure and therefore the DOE thickness profile must be modified so that it will produce a more desirable intensity distribution for its required application.

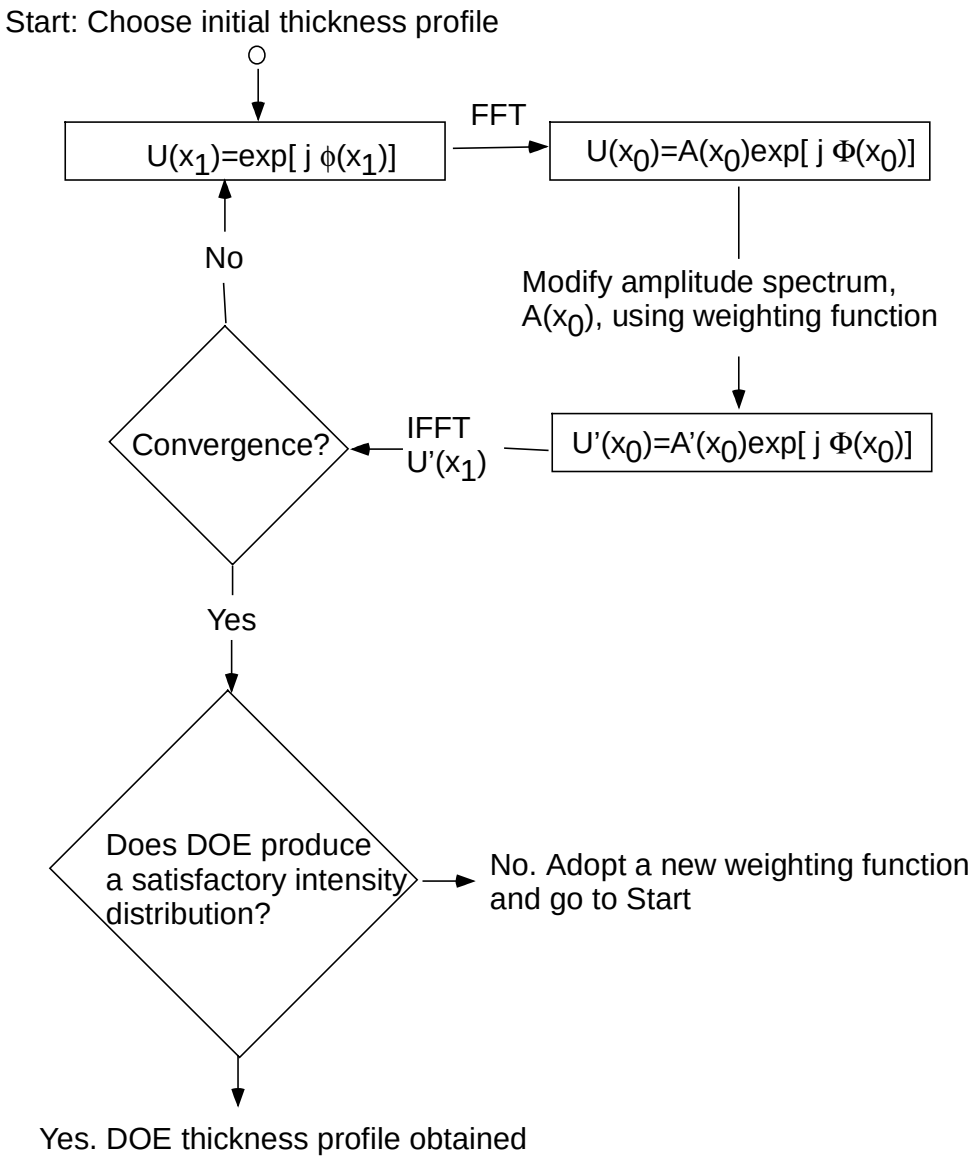

Figure 7.6 DOE design algorithm using the Fraunhofer approximation.

In the IFTA, as shown in Figure 7.6, after an initial object thickness profile has been selected and directly sampled over appropriate intervals with its appropriately applied constraints, e.g., finite aperture, the FFT, $U_i(x_0)$, is then calculated. That is,

$$U_i(x_1) = \exp[j\, \phi_i(d(x_1))] \text{ ------- FFT } \rightarrow U_i(x_0) = A_i(x_0) \exp[j\, \Phi_i(x_0)], \qquad (7.5)$$

in which $A_i(x_0)$ is the far-field amplitude and $\Phi_i(x_0)$ is the phase of the FFT, $U_i(x_0)$. Since it is the intensity that is of interest, we note that $I_i(x_0) \propto A_i(x_0)*A_i(x_0) = |A_i(x_0)|^2$ and that the intensity can be changed by altering the amplitude only while the far-field phase, $\Phi_i(x_0)$, which is an extra degree of freedom that does not affect the intensity, does not need to be changed throughout the iterative procedure. Note that when using an FFT, the spacing in the image plane is $\Delta x_0 = \lambda z / (N \Delta x_1)$, in which Δx_1 is the spacing in the object plane, λ is the wavelength of light, z is the distance to the image plane, and N is the number of sampled points.

The next step is to alter the amplitude according to some predetermined weighting function as follows:

$$U_i(x_0)=A_i(x_0) \exp[j \Phi_i(x_0)] \text{ ---Modify } \rightarrow U_i'(x_0)= A_i'(x_0) \exp[j \Phi_i(x_0)]. \qquad (7.6)$$

The weighting function that determines how much each sampled point of the amplitude spectrum changes depends entirely upon the what the desired intensity is. In general, each sampled point of $A_i(x_0)$ is changed some fractional amount so as to bring the intensity closer to what is desired. In general, the weighting function is a function of both $A_i(x_0)$ and x_0 and we can write

$$A_i'(x_0) = \text{WEIGHT}(x_0;A_i(x_0)), \qquad (7.7)$$

in which $\text{WEIGHT}(x_0;A(x_0))$ is the weighting function. It is assumed that very small fractional changes are best in order to avoid the same sort of problems encountered by using an unmodified Gerchberg-Saxton algorithm, namely stagnation [18]. In the examples discussed in this chapter, the amplitude at each point in the image plane is usully changed by a random fractional amount between 0 and 2% closer to what is desired. The choice of a 0 to 2% perturbation may appear to

be arbitrary, but the point is to change the profile only slightly in order to avoid stagnation in the algorithm whenever possible.

The next step is to calculate the Inverse Fast Fourier Transform (IFFT) of the perturbed transfer function, $U_i'(x_0)$, which, after applying the appropriate constraints such as unit amplitude and finite aperture, yields a modified phase profile. This is given by

$$U_i'(x_0) = A_i'(x_0)\exp[j\,\Phi_i(x_0)]\text{---IFFT—and apply constraints}$$

$$\rightarrow U_{i+1}'(x_1) = \exp[j\,\phi_{i+1}'(x_1)], \qquad (7.8)$$

in which $U_{i+1}'(x1)$ and $\phi_{i+1}'(x_1)$ are the new object and phase profile, respectively. The new DOE thickness profile, $d_{i+1}'(x_1)$, can be determined from the new phase distribution, $\phi_{i+1}'(x_1)$.

This procedure is done iteratively until the profile no longer changes. If the weighting function used in changing the far-field amplitudes was appropriately chosen, then a suitable object profile is rendered. Otherwise, a new choice in determining either the initial profile from the onset or a change of the weighting function used in the iteration process is in order.

So far, the DOE designs presented only included the case when the Fraunhofer approximation is valid. Another modification of the IFTA using only the Fresnel approximation is now considered.

Using the Fresnel approximation [2], the field in the image plane is given by

$$U(x_0) = \int U(x_1)\ \exp[j\,k\,x_1^{\,2}/2z]\ \exp[-j\ k\,x_0\,x_1/z]\,dx_1\,, \qquad (7.9)$$

in which $U(x_1)$ is the object, k is the wavenumber, $2\pi/\lambda$, and z is the distance between the object and image planes. $U(x_0)$ is essentially the Fourier transform of $t_1(x)\exp\left[j\frac{k}{2z}x_1^2\right]$.

Designing a DOE, assuming that the Fresnel approximation is valid, is not a difficult matter once it has been designed using the Fraunhofer approximation. It requires that the object profile obtained in the previous part of the algorithm be multiplied by a corrective phase factor $\exp\left[-j\frac{k}{2z}x_1^2\right]$ to compensate for the exponential term in the Fresnel diffraction integral. Therefore if some phase function, $\phi(x_1)$, was obtained from the iteration procedure discussed in the previous section, then the final profile with correction is represented by

$$\phi(x_1) \rightarrow \phi(x_1) - j\frac{k}{2z}x_1^2 . \tag{7.10}$$

The DOE etch depth is then extracted from knowledge of the phase using Equation 7.3.

An alternative approach is to simply incorporate the quadratic profile directly in the algorithm. This method was the one actually used to design the vertical diffuser discussed in the next section.

7.3 Vertical diffuser design and analysis for the three-dimensional display

One application considered in this dissertation is the design of a diffractive optic vertical diffuser that is to be placed after each partial pixel of the ICVision autostereoscopic display, as shown in Figure 7.4 [46]. With unit amplitude plane-wave illumination of wavelength λ, the diffractive optic is intended to diffuse the incident beam vertically in the far-field at a distance d_v from the object plane. The numerical parameters for the ICVision application are w = 50 mm, L=38 μm, d_v =30 cm, λ=633 nm and the minimum feature size equals 1 μm.

The DOE in the array is confined by a finite aperture of width, L, is phase-only, and expands light uniformly over a vertical range, w, in an image plane with little light outside this region. For optimal viewing of this display, the intensity pattern must vary rather smoothly in the viewing region. Here, “optimal viewing” means that an observer can comfortably see a three-

dimensional image produced by the ICVision display over a vertical height, w, and a distance d_v from the display. As mentioned previously, since several orders are diffracted, the zeroeth and higher orders must not overlap the first order diffracted beam since the latter contains the information of interest.

The initial phase profile used for the iterative algorithm to calculate the Fraunhofer intensity is modeled as a Fresnel lens having a focal length of f, as shown in Figure 7.7. An intensity pattern in the image plane is produced that roughly spreads all of the light over the desired viewing region over a height of 50 mm (as opposed to 10 mm without a diffuser). The value of f that spread light over the desired region was 240 μm. As shown in Figure 7.8, the far-field intensity for this lens does not appear to be uniform within the viewing region as desired. However, it can be used as a starting point for a modified IFTA.

Note that the minimum feature size of the diffractive optic is 1μm while the illuminating wavelength is 0.6328μm. This would suggest that perhaps scalar diffraction theory may be inaccurate for the design of such a device. Therefore, the scalar results are compared with results obtained via the Finite-Difference Time-Domain Method (FDTD) discussed in previous chapters. Note that TE and TM are defined in the same manner as they were in previous chapters and, as shown in Figure 7.8, that there is excellent agreement with the scalar result.

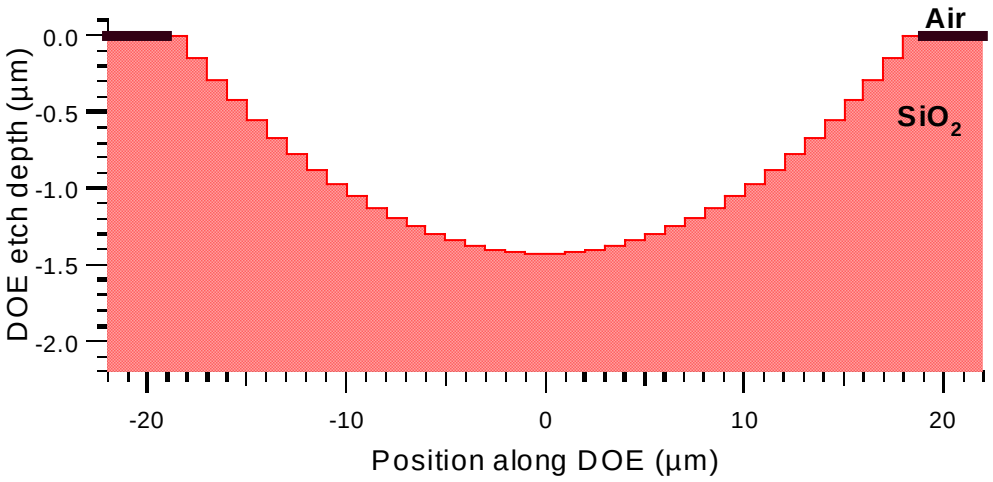

Figure 7.7 Initial lens thickness profile.

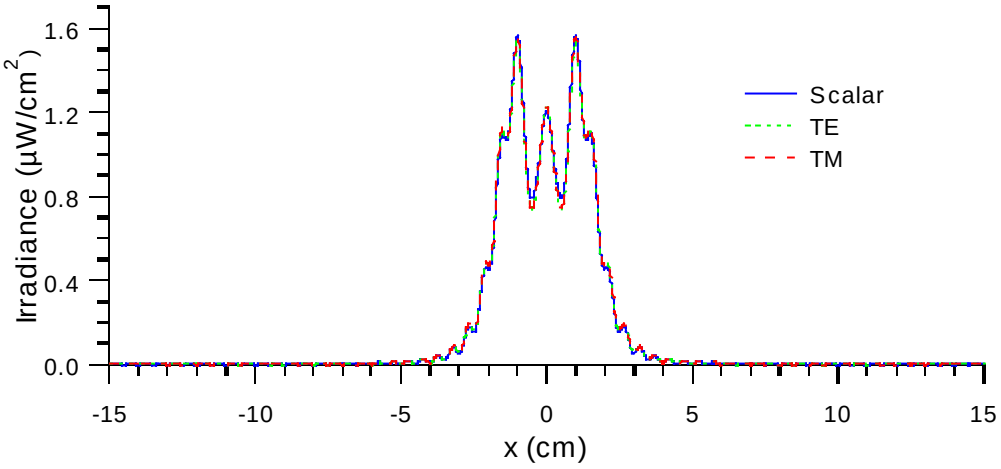

Figure 7.8 Resultant irradiance distribution in viewing region.

The next step is to choose a weighting function for the iterative procedure. Determining this weighting function is a slightly difficult matter since it is not intuitive how to produce a highly uniform intensity profile, but the weighting function that yields the highest intensity uniformity within the viewing region that was found is

$$\text{WEIGHT}(x_0) = W_0(x_0) \left[\frac{\int dx_0 \quad |A(x_0)|^2}{\int dx_0 \quad |W_0(x_0)|^2} \right]^{1/2}, \tag{7.11}$$

with $W_0(x_0) = \varepsilon\,(A_i(x_0) - \text{mean}[A_i(x_0)$ over the interval: $\{- w/2 < x_0 < w/2\}$ and zero outside]). The mean function takes the arithmetic mean of the sampled points of the far-field amplitude, $A(x_0)$, over the specified interval, and ε is a small, random perturbation ranging from 0 to 0.02. So that the amplitudes outside the viewing region, $-w/2$ mm $< x_0 < w/2$ mm, which are of little interest, are not altered implicitly when the amplitudes of the FFT are renormalized, the weighting function, itself, is normalized explicitly throughout the algorithm. The modified amplitude in the IFTA is now expressed as

$$A'(x_0) = [A(x_0) - \text{WEIGHT}(x_0)]\,. \tag{7.12}$$

An explanation of why the initial thickness profile and weighting function for the IFTA were selected can be offered after defining one important quantity, the diffraction efficiency. The diffraction efficiency, η, is defined here as the fraction of light entering the specified viewing region, i.e.,

$$\eta \equiv \frac{\int dx_0 \text{rect}\left(\frac{x_0}{w}\right)\left|A(x_0)\right|^2}{\int dx_0 \left|A(x_0)\right|^2}, \tag{7.13}$$

in which w is the size of the viewing region. It is important, in the first step of the IFTA, to choose a Fresnel lens having a high diffraction efficiency that yields an intensity that roughly

spans the entire viewing region, so that after the algorithm converges, the amplitudes, and hence intensities, within the viewing region tend to converge to an average value while a rather high diffraction efficiency is still maintained. Trying to improve the diffraction efficiencies with other weighting functions generally comes at the expense of the uniformity of intensity within the viewing region.

Figure 7.9 shows the final etch depth profile after 500 iterations. The intensity distribution the DOE produces along with the comparison with rigorous results is shown in Figure 7.10. The scalar intensity profile is quite uniform within the viewing region and the diffraction efficiency is roughly 83%.

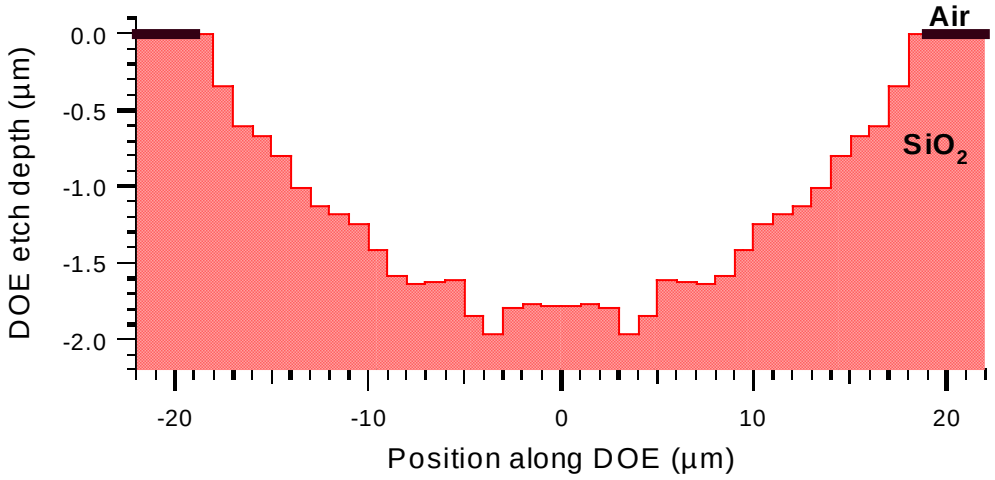

Figure 7.9 Resultant etch depth profile for vertical diffuser design.

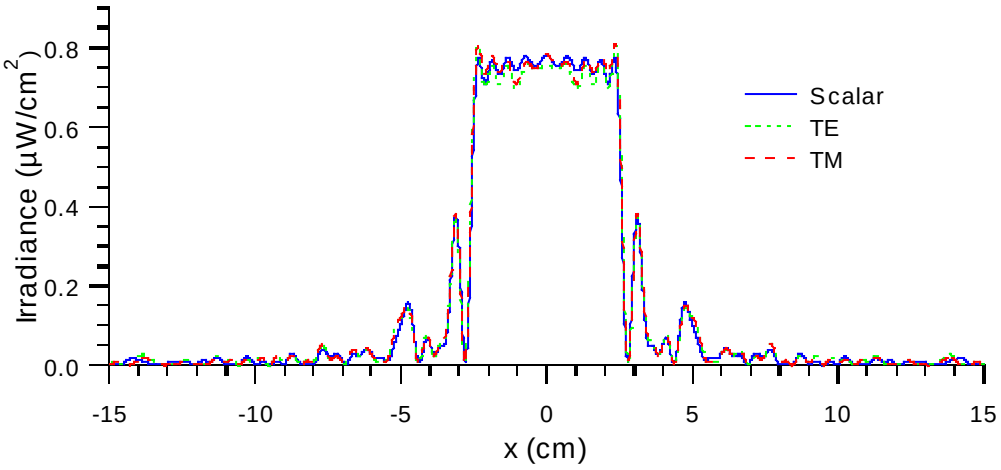

Figure 7.10 Vertical diffuser irradiance distribution.

This chapter presented a unique application that dealt with the design of diffractive optic diffusers using an iterative technique. Future research would entail designing diffractive optic diffusers that operate for green and blue light. Also, the fabrication and testing of these devices would be of interest as well.

Chapter 8

SUMMARY AND CONCLUSIONS

This dissertation presented the design and analysis of finite aperture diffractive optical elements. Specifically, DOEs with features on the same order of magnitude as the wavelength of incident light were designed and analyzed. The tools of rigorous analysis that describe DOE performance were also discussed in detail.

The theoretical development of BEM can be used to rigorously assess designs of the finite aperture DOEs. One limitation of using a BEM is that it requires a large amount of computer memory to obtain results. Therefore, the implementation of a more efficient method, the FDTD, was discussed and derived for practical DOE applications. The boundary element method was still very useful, however, in that it still gives accurate results and is an excellent tool with which to validate other rigorous methods.

The limits of AS scalar diffraction theory were also discussed. It appears that AS scalar diffraction theory is applicable in several cases for finite aperture DOE design with features on the order of magnitude or less than the wavelength of illuminating light. Regardless of whether the exiting medium is air or silicon, it appears that AS scalar model is a viable method provided that the spatial period, Λ, is not much less than twice the free-space wavelength, i.e., $\Lambda \geq 2\lambda_0$. Results also indicated that the size of the DOE zones and edge effects are dominating factors in determining whether AS scalar diffraction theory is valid. The presented iterative design method, IASA, was surprisingly accurate for designing finite aperture DOEs with sub-wavelength features. Also, IASA can be used to design finite aperture DOEs which cannot be designed

heuristically, such as 1-3 and 1-4 or higher-order beamfanners. Also discussed in the design of fine aperture DOEs was the development of a re-sampling technique that is used to increase the DOE minimum feature size so that the designed DOEs are easier to fabricate and the effect of DOE etch depth quantization was discussed as well.

Also, the design of a diffuser for an autostereoscopic display system was presented. The features of the DOE were of the same order of magnitude as the incident illumination. The diffuser was designed using a scalar-based method and the results are compared with those obtained from rigorous analyses. Specifically, a modified version of an Iterative Fresnel Transform Algorithm (IFTA) was used to design the diffusers. The results of the scalar analysis were well approximated to those obtained via FDTD.

Future investigations would include more extensive analysis of the fields near DOE interfaces to more fully assess the limits of scalar-based diffraction theory. This investigation will also include examining the effects of DOE edges on the fields just past a DOE and may further explain the reasons why, as well as the extent to which, AS scalar diffraction theory is valid. This might be measured quantitatively in terms of the depths of sharp edges with respect to the wavelength of incident illumination.

Future software development will likely involve the development of FDTD for three-dimensional problem applications and animation routines to allow for problem visualization. Further investigation of use of FDTD for designing DOEs would also be of interest. Future analysis may also require investigation of problematic numerical dispersion effects inherent with the implementation of FDTD.

Future research might also include an analysis of the effects of broad-band illumination of DOEs with regards to performance. Optimization routines could be further developed in DOE designs for broad-band cases. Considerations such as the mutual coherence effects between sampled points on DOE surfaces could also be examined.

APPENDICES

APPENDIX A

Power in diffracted orders of a 1-2 beamfanner

The purpose of this appendix is to derive an expression for the diffraction efficiency for an infinite-aperture binary optic illuminated by a unit-amplitude plane wave. The results obtained provide the motivation for the method of heuristically designing the 1-2 beamfanner presented in Chapter 5.

Consider the transmission function, $t(x)$, of a diffractive optic of the form

$$t(x) = \exp\left[j\frac{2\pi}{\lambda}\Delta n\, d(x) \right] \tag{A.1}$$

in which λ is the free space wavelength, $\Delta n = n_2 - n_1$, where n_1 and n_2 are the indices of refraction for the entrance and exiting media, respectively, and d(x) is the etch depth profile. The transmission function may be expanded into a Fourier series of the form

$$t(x) = \sum_{m=-\infty}^{\infty} a_m \exp\left(-j\frac{2\pi m}{\Lambda} x \right), \tag{A.2}$$

in which

$$a_m = \frac{1}{\Lambda} \int_{-\Lambda/2}^{\Lambda/2} t(x) \exp\left(j\frac{2\pi m}{\Lambda} x \right) dx, \tag{A.3}$$

in which Λ is the grating period as shown in Figure A.1.

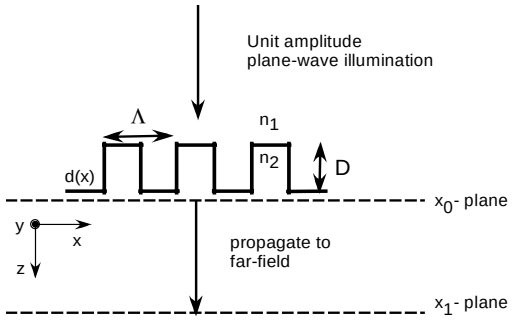

Figure A.1 Geometry of an infinite aperture binary grating.

If the grating has a fill factor of f and a maximum etch, D, then the coefficients in Equation A.3 are given by

$$a_m = \frac{1}{m\pi}\left(e^{jk\Delta nD} - 1\right)\sin\left(m\pi f\right) \quad \text{for } m \neq 0 \tag{A.4}$$

and

$$a_0 = (1-f) + fe^{jk\Delta nD}, \tag{A.5}$$

in which k is the wave number given by $k=2\pi/\lambda$.

If $e^{jk\Delta nD} = -1$, in which case $k\Delta nD=\pi$, and hence $D = \frac{\pi}{k\Delta n} = \frac{\lambda}{2\Delta n}$ and the diffraction efficiency is optimized. The coefficients take the form

$$\begin{aligned} a_m &= \frac{-2}{m\pi}\sin\left(m\pi f\right) \\ a_0 &= 1-2f \end{aligned}. \tag{A.6}$$

Also, if the fill factor equals ½, the odd orders are optimized and all even orders are zero, then

$$a_m = \frac{-2}{m\pi}\sin\left(\frac{m\pi}{2}\right). \tag{A.7}$$

For normal incident unit amplitude plane wave illumination, the power of the diffracted orders in the far-field is given by

$$|a_m|^2 = \left(\frac{2}{m\pi}\right)^2 \sin^2\left(\frac{m\pi}{2}\right), \tag{A.8}$$

or equivalently,

$$|a_m|^2 = \begin{cases} \left(\frac{2}{m\pi}\right)^2 & \text{for m odd} \\ 0 & \text{for m even} \end{cases}. \tag{A.9}$$

The diffraction efficiency, η_m, of the m^{th} diffracted order is defined as

$$\eta_m = \frac{|a_m|^2}{\sum_{m=-\infty}^{\infty} |a_m|^2}. \tag{A.10}$$

Note that the diffraction efficiency is normalized and that the denominator is given by

$$\sum_{\substack{m=-\infty \\ \text{odd}}}^{\infty} |a_m|^2 = 2\sum_{\substack{m=1 \\ \text{odd}}}^{\infty} |a_m|^2 = 2\left(\frac{4}{\pi^2}\right)\left(\frac{1}{1^2}+\frac{1}{3^2}+\frac{1}{5^2}+\frac{1}{7^2}+\cdots\right) = \frac{8}{\pi^2}\left(\frac{\pi^2}{8}\right) = 1. \tag{A.11}$$

Therefore, the diffraction efficiency simplifies to

$$\eta_m = \begin{cases} \left(\frac{2}{m\pi}\right)^2 & \text{for m odd} \\ 0 & \text{for m even} \end{cases}. \tag{A.12}$$

The plot of the diffraction efficiencies is shown in Figure A.2. Note that the power is optimized in the ±1 orders which is desirable for a 1-2 beamfanner. The major difference between the diffractive optic presented in this appendix and that heuristically designed in Chapter 5 is that a finite aperture bounded the latter. Also, the diffractive optic in Chapter 5 included curvature to focus in an observation plane.

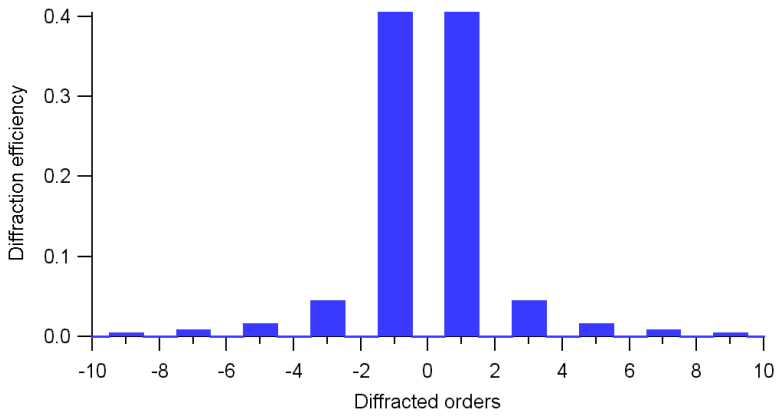

Figure A.2 Diffraction efficiency for binary grating.

APPENDIX B

Derivation of the Fresnel approximation from the angular spectrum approach

This appendix presents the derivation of the Fresnel approximation from the angular spectrum approach in two dimensions. Since this material is generally not presented in textbooks or elsewhere in the literature of Fourier optics, it is given in this appendix. The Fresnel approximation in two dimensions was used in Chapter 7 in the design and analysis of the vertical diffuser.

Adopting the nomenclature used by Goodman [2], the field in an observation plane a distance z from an object plane is expressed as

$$E(x_0) = \int_{-\infty}^{+\infty} A_0(f_x) \exp\left[-j\left(k_z z + k_x x_0\right)\right] df_x \,, \tag{B.1}$$

in which x_0 and x_1 are the image and object plane positions, respectively, the angular spectrum, $A_0(f_x)$, is the Fourier transform of the field in the object plane denoted as of $E(x_1)$ and the wave vector components, k_x and k_z , are given by

$$k_x = 2\pi f_x \qquad \text{and}$$
$$k_z = \begin{cases} \dfrac{2\pi}{\lambda}\sqrt{1-\left(f_x\lambda\right)^2} & \text{for } \left|f_x\right| \le \dfrac{1}{\lambda} \\ -j\dfrac{2\pi}{\lambda}\sqrt{1-\left(f_x\lambda\right)^2} & \text{otherwise} \end{cases}, \tag{B.2}$$

respectively, in which λ is the reduced wavelength in the medium of propagation. The angular spectrum of the field in the object plane is expressed as

$$A_0(f_x) = F\{E(x_1)\} = \int_{-\infty}^{+\infty} E(x_1) \exp\left[j2\pi f_x x_1\right] dx_1 \,, \tag{B.3}$$

in which the function, F, is the Fourier transform operator.

For small spatial frequencies, one may use the approximation

$$\sqrt{1-\varepsilon} \approx 1-\frac{1}{2}\varepsilon \quad \text{and let} \quad \varepsilon = (f_x\lambda)^2 .$$

Upon substitution and further manipulation, the field in an observation plane is given by

$$\begin{aligned} E(x_0) &= \int_{-\infty}^{+\infty} A_0(f_x)\exp\left[-j2\pi\left(\sqrt{1-(f_x\lambda)^2}\,\frac{z}{\lambda}+f_x x_0\right)\right]df_x \\ &\approx \int_{-\infty}^{+\infty} A_0(f_x)\exp\left[-j2\pi\left(\left(1-f_x^2\lambda^2\right)\frac{z}{\lambda}+f_x x_0\right)\right]df_x \\ &= e^{-\frac{j2\pi z}{\lambda}}\int_{-\infty}^{+\infty} A_0(f_x)\exp\left[j\left(\pi\lambda z f_x^2+2\pi f_x x_0\right)\right]df_x \end{aligned}$$

$$E(x_0) = e^{-jkz}\int_{-\infty}^{+\infty} dx_1 E(x_1)\underbrace{\int_{-\infty}^{\infty} df_x \exp\left[j\left(\pi\lambda z f_x^2-2\pi f_x\left(x_0-x_1\right)\right)\right]}_{I}$$

$$\begin{aligned} I &= \int_{-\infty}^{\infty} df_x \exp\left[j\pi\lambda z\left(f_x^2-\frac{2\pi\left(x_0-x_1\right)}{\pi\lambda z}f_x\right)\right] \\ &= e^{-j\frac{\pi}{\lambda z}(x_0-x_1)^2}\int_{-\infty}^{\infty} df_x e^{-\frac{\pi\lambda z}{j}f_x^2} = e^{-j\frac{\pi}{\lambda z}(x_0-x_1)^2}\sqrt{\frac{j}{\lambda z}} . \end{aligned}$$

The field is then

$$E(x_0) = \frac{e^{-jkz}}{\sqrt{-j\lambda z}}\int_{-\infty}^{+\infty} dx_1 E(x_1) e^{-j\frac{\pi}{\lambda z}\left((x_0-x_1)^2\right)}, \tag{B.4}$$

or equivalently,

$$E(x_0) = \frac{e^{-jkz}}{\sqrt{-j\lambda z}} e^{-j\frac{\pi x_0^2}{\lambda z}}\int_{-\infty}^{+\infty} dx_1 E(x_1) e^{-j\frac{\pi x_1^2}{\lambda z}} e^{j2\pi\frac{x_0}{\lambda z}x_1} . \tag{B.5}$$

Equation B.5 represents the electric field in an observation plane using two-dimensional Fresnel propagation. Equivalently, Equation B.5 may be expressed as

$$E(x_0) = \frac{e^{-jkz}}{\sqrt{-j\lambda z}} e^{-j\frac{\pi x_0^2}{\lambda z}} F\left\{e^{-j\frac{\pi x_1^2}{\lambda z}} E(x_1)\right\}, \tag{B.6}$$

in which $E(x_0)$ is proportional to the Fourier transform of the product of the field in the object plane and a quadratic phase term evaluated at a spatial frequency, $f_x = \frac{x_0}{\lambda z}$.

APPENDIX C

Scalar analysis codes

This appendix contains the computer programs used for the scalar analysis of diffractive optical elements presented in this dissertation. Note that the codes are written in Matlab®.

```
%*************************************************************************
% Scalar theory analysis of DOE structures.
%
%
%
%       Last revision:    12/11/98
%*************************************************************************
Start_time=cputime;      % Time program run
clear FILENAMES FILES_OUT           % for later use
% Open file containing the following input parameters :
%-------------------------------------------------------------------------
%       lam             = incident wavelength [microns]
%       n1              = index of refraction of medium 1 (air)
%       n2              = index of refraction of DOE (silicon)
%       P               = Number of partitions of DOE profile (if there is re-sampling)
%       Q               = Number of quantized DOE levels (0 means no quantization)
%       samp    = samples along transverse features (samp=1 -> default value)
%       Uamp    = magnitude of incident field     [Volts/m or Amps/m]
%       Uang    = ccw angle k vector makes with positive x axis [radians]
%       xUmax = largest  position of field incident on DOE [microns]
%       xUmin  = smallest position of field incident on DOE [microns]
%       FFTpower= power of FFT in near to far field transformation (2^N)
%       z_dist   = distance to the image plane [microns]
%       x0_max          = max position in image plane [microns]

%       x0_min          = min position in image plane [microns]
%       filenumber              : number of scalar files to evaluate
%       filename                :       file containing the scalar data.
%-------------------------------------------------------------------------

MasterFile='MasterSCALAR01.txt';                % Master batch file including options
fprintf(['\n-------------------------------------------------------------------------------------\n']);
in_file=input(['Enter SCALAR master file ( default-> ',MasterFile,' ):   ']);
if not(isempty(in_file)),MasterFile=in_file;end;clear in_file
fid = fopen(MasterFile,'r');
for k=1:5,fgets(fid);end;
temp=fscanf(fid,'%s\n',1);
temp=fscanf(fid,'%s',1);
TManalysis =fscanf(fid,'%s',1);
TManalysis_script =fscanf(fid,'%s\n',1); % script file for TM analysis (if nec.)


temp=fscanf(fid,'%s',1);
ASfiltering =fscanf(fid,'%s',1);
ASfiltering_script =fscanf(fid,'%s\n',1);  % script file for Ang. Spect. analysis

temp=fscanf(fid,'%s',1);
fieldimageplot =fscanf(fid,'%s',1);
```

```
fieldimageplot_script =fscanf(fid,'%s\n',1);% script file for field image plotting

temp=fscanf(fid,'%s',1);
file_storage =fscanf(fid,'%s\n',1);                    % file to store data (if nec.)

temp=fscanf(fid,'%s',1);
pre_processing =fscanf(fid,'%s',1);
pre_processing_script =fscanf(fid,'%s\n',1);% Pre-processing of transmission fcn. (if nec.)

temp=fscanf(fid,'%s',1);
post_processing =fscanf(fid,'%s',1);
post_processing_script =fscanf(fid,'%s\n',1);% Pre-processing of analysis (if nec.)

temp=fscanf(fid,'%s',1);
number_of_runs =fscanf(fid,'%g\n',1); % if running script file input multiple # of times


temp=fscanf(fid,'%s',1);
paranumber=fscanf(fid,'%g\n',1);          % # of parameter files
PARAM=[''];
for k=1:paranumber,
temp=fscanf(fid,'%s',1);
parameters =fscanf(fid,'%s\n',1);% partial names assigned to output
PARAM=char(PARAM,parameters);
end
PARAM(1,:)=[];
fclose(fid);

%-------------------------------------------------------------------------------
% For Angular spectrum and filtered field analysis
fxmin=-10^(10);fxmax= 10^(10);fxspace=0;
if upper(ASfiltering(1))=='Y',
        fprintf(['\nRunning Angular spectrum and filtering option:  ',ASfiltering_script,' \n']);

        fxmintemp=input(['Enter FX min (µm^(-1)):   ']);
        if not(isempty(fxmintemp)),fxmin=fxmintemp;end;

        fxmaxtemp=input(['Enter FX max (µm^(-1)):   ']);
        if not(isempty(fxmaxtemp)),fxmax=fxmaxtemp;end;

        fxspacetemp=input(['Enter FX spacing (µm^(-1)):   ']);
        if not(isempty(fxspacetemp)),fxspace=fxspacetemp;end;
end;    clear fxspacetemp fxmaxtemp fxmintemp
%-------------------------------------------------------------------------------
XSIP=0;         % field image plot option
ZSIP=0;
if upper(fieldimageplot(1))=='Y',
        fprintf(['\nRunning field image plot option:  ',fieldimageplot_script,' \n']);
        XSIPtemp=input(['Enter x image plot spacing (µm):   ']);
        if not(isempty(XSIPtemp)),XSIP=XSIPtemp;end;
```

```
        ZSIPtemp=input(['Enter z image plot spacing (µm):   ']);
        if not(isempty(ZSIPtemp)),ZSIP=ZSIPtemp;end;
end;clear XSIPtemp ZSIPtemp
%-------------------------------------------------------------------------------
%*******************************************************************************


for PARAM1=1:paranumber,     % super-batch mode for multiple parameter files
        parameters=PARAM(PARAM1,:);parameters=parameters(not(isspace(parameters)));

fid = fopen(parameters,'r');
for k=1:5,fgets(fid);end;
for k=1:14,temp = fscanf(fid,'%s',1);a(k)=fscanf(fid,'%g\n',1);end;
fgets(fid);
temp=fscanf(fid,'%s',1);
data_save=fscanf(fid,'%s\n',1);          % Save computed data? (Y/N)
temp=fscanf(fid,'%s',1);                 % type of field transformation:
xform=fscanf(fid,'%s\n',1);                      % 'angularspectrum', 'fresnel', or 'fraunhofer'.
temp=fscanf(fid,'%s',1);                 % units of image plane positions:
x0units=fscanf(fid,'%s\n',1);            % 'microns', 'centimeters', or 'millimeters'
temp=fscanf(fid,'%s',1);
filenumber=fscanf(fid,'%g\n',1);% # of files
for casenumber=1:filenumber,
temp=fscanf(fid,'%s',1);
FILENAMES(casenumber,:)=fscanf(fid,'%s',1);          % file containing scalar data
FILES_OUT(casenumber,:) =fscanf(fid,'%s\n',1);       % partial names assigned to output
end

fclose(fid);
lam             = a(1);                          %
n1              = a(2);                          %
n2              = a(3);                          %
P               = a(4);                          %
Q               = a(5);                          %
samp1           = a(6);
E_amp           = a(7);                          %
Uang            = a(8);%* angle of incident wave (for now, assume normal incidence)
xUmax           = a(9);                          %
xUmin           = a(10);                         %
FFTpower       = a(11);                          %
z_dist          = a(12);                         %
x0_max          = a(13);                         %
x0_min          = a(14);                         %

clear a fid temp

for casenumber=1:filenumber,             % Loop for batch mode
        for runnumber= 1:number_of_runs,       %       Loop for # of runs (if necessary)
        filename=FILENAMES(casenumber,:);filename=filename(not(isspace(filename)));
        file_out=FILES_OUT(casenumber,:);file_out=file_out(not(isspace(file_out)));
```

```
        file_out_lett=file_out(find(isletter(file_out)));

%-------------------------------------------------------------------------
% Xdat = Position along DOE [microns]
% Z      =DOE etch depth into silicon [microns]
if prod(filename(length(filename)-3:length(filename))=='.txt'),    % scalar file ?
        [Xdat,Z]=reading_files(filename);% Reads data from scalar file
% Xdat = Position along DOE [microns]
% Z      =DOE etch depth into silicon [microns]
% For geometries other than those described by scalar files
else
        eval(filename); % If not a scalar file input
end
%-------------------------------------------------------------------------
% Evaluate any pre-processing operations
if upper(pre_processing(1))=='Y',eval(pre_processing_script);end;
%-------------------------------------------------------------------------
% Adjust profile if there is re-sampling
[Z,x1]=Repartition1(Z,Xdat,P,samp1,[],[],n2-n1);
mfs=abs(x1(2)-x1(1));   % minimum scalar spacing
x1_size=length(x1);
L=max(x1)-min(x1)+mfs;          % width of DOE
%-------------------------------------------------------------------------
% Quantize etch depth levels (note: if Q=0 -> no quantization)
Z=quantize(Z,Q);                        % quantization is done after re-sampling
Zmax=max(Z);                            % maximum etch depth
Zmin=min(Z);                            % minimum etch depth
%-------------------------------------------------------------------------
% Display SCALAR run statistics
fprintf(['-------------------------------------------------------------------------------------\n']);
fprintf(['Running SCALAR analysis for file:\t',filename,'\t\t']);
fprintf([datestr(now,8),'\t',datestr(now,1),'\t',datestr(now,16),'\n\n\n']);
fprintf(['maximum DOE etch depth   :\t\t\t\t\t\t\t\tz= ',num2str(max(abs(Z))),' microns \n']);
%-------------------------------------------------------------------------
% Plane wave spectrum preliminaries
DFTsize=max([power(2,FFTpower);0*4*power(2,ceil(log2(x1_size)))]);% size of FFT
x1_AS=[(0:DFTsize/2-1),(-DFTsize/2:-1)].'*mfs+min(abs(x1));  % object plane positions
[microns]
%-------------------------------------------------------------------------
% Use scalar techniques to obtain Ey at image plane

%d=5;R=sqrt(d^2+x1.^2);E_amp=exp(-j*2*pi*n1/lam*R)./R; % spherical illumination

E_amp1=E_amp/sqrt(n1);                  % Power consideration of incident field
Ey1=E_amp1.*exp(j*2*pi/lam*((n2-n1).*Z-n2*Zmax)); % transmission function
Ey_m1=[Ey1;zeros(DFTsize-x1_size,1)];
Ey_m1=shiftud(Ey_m1,-floor(x1_size/2),1).*(x1_AS>=xUmin&x1_AS<=xUmax);
%-------------------------------------------------------------------------
% Compensate for transmission coefficient
theta1=Uang-pi/2;
```

```
theta2=asin(n1*sin(theta1)/n2);             % Snell's Law
Tmiss  =2*n1*cos(theta1)/(n1*cos(theta1)+n2*cos(theta2)); % trans. coeff.for E
TmissTM=2*n2*cos(theta1)/(n1*cos(theta2)+n2*cos(theta1)); % trans. coeff.for H
%---------------------------------------------------------------------------
% Physical constants
dn=n2-n1;
mu=4*pi*10^(-7);                            % permeability of free space [Henrys/m]
epsilon0=8.8541*10^(-12);       % permittivity of free space [Farads/m]
c=10^(-9)/sqrt(mu*epsilon0);% speed of light in units of [microns/femtosecond]
omega=2*pi*c/lam;                           % angular frequency of light [radians/femtosecond]
N_0=sqrt(mu/epsilon0);          % impedendence in free space [ohms]
%---------------------------------------------------------------------------
% Evaluate any pre-processing operations
%if upper(pre_processing(1))=='Y',eval(pre_processing_script);end;
%---------------------------------------------------------------------------
if lower(xform(1:5))=='angul',
        [ey,hx]=MaxwellPropagation01('E',Tmiss*Ey_m1,z_dist-Zmax,x1_AS,lam,n2);
        x0=x1_AS;
else
        [ey,x0]=scalar_analysis1(Tmiss*Ey_m1,x1_AS,z_dist-Zmax,lam/n2,xform);
        [ey,hx]=MaxwellPropagation01('E',ey,0,x0,lam,n2);
end;

if upper(TManalysis(1))=='Y', % TM analysis option
        if lower(xform(1:5))=='angul',
                [hy,ex,ez]=MaxwellPropagation01('H',TmissTM*Ey_m1*n1/N_0,z_dist-
Zmax,x1_AS,lam,n2);
                x0=x1_AS;
        else
                [hy,x0]=scalar_analysis1(TmissTM*Ey_m1*n1/N_0,x1_AS,z_dist-
Zmax,lam/n2,xform);
                [hy,ex,ez]=MaxwellPropagation01('H',hy,0,x0,lam,n2);
        end;
end;
%---------------------------------------------------------------------------
%---------------------------------------------------------------------------
% Units of x0 positions
if lower(x0units(1:5))=='milli',x0=x0/1000;end;
if lower(x0units(1:5))=='centi',x0=x0/10000;end;
%---------------------------------------------------------------------------
assignin('base',['XDOE',file_out],flatten([x1-mfs/2,x1+mfs/2].'));
assignin('base',['TDOE',file_out],flatten([Z,Z].'));% plot DOE profile
assignin('base',['Tpha',file_out],(2*abs(dn)/lam)*flatten([Z,Z].'));% Phase in units of ¹
%---------------------------------------------------------------------------
x0=fftshift(x0); % In batch mode, positions are often redundant
xmin=min(find(x0>=x0_min));xmax=max(find(x0<=x0_max));

xpos=mod((xmin+DFTsize/2:xmax+DFTsize/2),DFTsize);
xpos=xpos+DFTsize*(xpos==0);            % Shifted FFT positions
```

```
xsmallest=x0(xmin);xlargest=x0(xmax);
x0=x0(xmin:xmax);
assignin('base',['XSCALAR' ,file_out_lett],x0);                    % x0 positions
%-------------------------------------------------------------------------
% assign names to field components
        ey=ey(xpos);%fftshift(ey);ey=ey(xmin:xmax);
        hx=hx(xpos);%fftshift(hx);hx=hx(xmin:xmax);

        assignin('base',['eySCALAR',file_out],ey);
        assignin('base',['hxSCALAR',file_out],hx);
        assignin('base',['SESCALAR',file_out],0.5*real(-ey.*conj(hx)));
if upper(TManalysis(1))=='Y', % TM analysis option

        ex=ex(xpos);%fftshift(ex);ex=ex(xmin:xmax);
        hy=hy(xpos);%fftshift(hy);hy=hy(xmin:xmax);
        ez=ez(xpos);%fftshift(ez);ez=ez(xmin:xmax);

        assignin('base',['hySCALAR',file_out],hy);
        assignin('base',['exSCALAR',file_out],ex);
        assignin('base',['ezSCALAR',file_out],ez);
        assignin('base',['SHSCALAR',file_out],0.5*real( ex.*conj(hy)));
end;
%-------------------------------------------------------------------------
if upper(data_save(1))=='Y',      % save data?
                saving_files([x0,abs(ey)],['E_temp',file_out,'.txt'],...
                        'BigSpace240:Users:Mellin:Temporary_datafiles:');
end;
%-------------------------------------------------------------------------
% Angular spectrum and filtered field analysis of aperture
if upper(ASfiltering(1))=='Y',eval(ASfiltering_script);end;
%-------------------------------------------------------------------------
% Construct image plot of field components as they propagate to detector plane
if upper(fieldimageplot(1))=='Y',

        [EyPLOTtemp]=feval(fieldimageplot_script,'E',Ey1,x1,...

        DFTsize,[xsmallest;xlargest;XSIP],[Zmax;z_dist+ZSIP;ZSIP],lam,n2);
                assignin('base',['Eyscalarimageplot',file_out],EyPLOTtemp); clear EyPLOTtemp
                assignin('base',['ximageplotscalar',file_out],x_imageplotting); clear
x_imageplotting
                assignin('base',['zimageplotscalar',file_out],z_imageplotting); clear
z_imageplotting
end;
%-------------------------------------------------------------------------
% Evaluate any post-processing operations
if upper(post_processing(1))=='Y',eval(post_processing_script);end;
%-------------------------------------------------------------------------
        end     % Close loop for # of runs (if necessary)
end     % Closing batch mode
clear FILENAMES FILES_OUT
```

```
%-----------------------------------------------------------------------
end     % Closing super-batch mode
%-----------------------------------------------------------------------
End_time=cputime-Start_time;
fprintf(['\nrun time = ',num2str(floor(End_time/60)),' min. ',num2str(rem(End_time,60)),'
seconds\n' ])
fprintf(['-------------------------------------------------------------------------------\n']);
%-----------------------------------------------------------------------
```

```
function [E,x,A0,fx]=scalar_analysis1(E,x,z,lam,xform);
%************************************************************************
%
% function [E,x,A0,fx]=scalar_analysis1(E,x,z,lam,xform)
%
% Scalar analysis of DOE structures with several options for field
% transformations ('angularspectrum', 'fresnel', or 'fraunhoffer').
%
%       Inputs:
%
%               E       =       incident disturbance
%               x       =       object plane positions
%               z       =       distance to image plane
%               lam     =       reduced wavelength in the propagating medium
%               xform= type of field transformation must be either
%                               'angularspectrum', 'fresnel', or 'fraunhoffer'.
%
%       Outputs:
%
%               E       =       field in image plane
%               x       =       image plane positions
%               A0      =       Angular spectrum of incident field before propagation
%               fx      =       spatial frequencies corresponding to angular spectrum
%
%       NOTE: All lengths are in units of MICRONS
%       NOTE:           exp(-jkz) notation      ->      replace fft by ifft and vice versa.
%
%       Created:        12/11/98
%************************************************************************
% Use plane wave spectrum to obtain E at image plane
s1=size(E);
s2=size(x);
E=flatten(E);
x=flatten(x);
dftsize=length(E);      % size of FFT
dx=abs(x(2)-x(1));
```

```
if lower(xform(1:5))=='angul',
        fx=[(0:dftsize/2-1),(-dftsize/2:-1)].'/(dftsize*dx);% fx:spatial frequencies
        kernel=conj(exp(j*2*pi*z./lam.*sqrt(1-(lam.*fx).^2)));   % kernel->propagation kernel
        A0=ifft(E)*dftsize*dx;  % NOTE: (dx*dftsize) term is for normalization
        E=fft(A0.*kernel)/(dftsize*dx);
        clear kernel       % note: x stays the same
end
if lower(xform(1:5))=='fresn',    % Use Fresnel approx to obtain E at image plane
        E=E.*exp(-j*pi/(lam*z)*x.^2);
        x=[(0:dftsize/2-1),(-dftsize/2:-1)].'*lam*z/(dftsize*dx);
        C=exp(-j*2*pi*z/lam)/sqrt(lam*z).*exp(-j*pi/(lam*z)*x.^2)*exp(j*pi/4);
        fx=[(0:dftsize/2-1),(-dftsize/2:-1)].'/(dftsize*dx);% fx:spatial frequencies
        A0=ifft(E)*dftsize*dx;  % NOTE: (dx*dftsize) term is for normalization
        E=C.*A0;         clear C;
end
if lower(xform(1:5))=='fraun',    % Use Fraunhoffer approx to obtain E at image plane
        x=[(0:dftsize/2-1),(-dftsize/2:-1)].'*lam*z/(dftsize*dx);
        C=exp(-j*2*pi*z/lam)/sqrt(lam*z).*exp(-j*pi/(lam*z)*x.^2)*exp(j*pi/4);
        fx=[(0:dftsize/2-1),(-dftsize/2:-1)].'/(dftsize*dx);% fx:spatial frequencies
        A0=ifft(E)*dx*dftsize;  % NOTE: (dx*dftsize) term is for normalization
        E=C.*A0;         clear C;
end;
E=reshape(E,s1);
x=reshape(x,s2);
A0=reshape(A0,s1);
fx=reshape(fx,s2);
%-------------------------------------------------------------------------
```

APPENDIX D

BEM analysis codes

This appendix contains the computer programs used for the BEM analysis of diffractive optical elements presented in this dissertation. Note that the codes are written in Matlab®.

```
%*************************************************************************
% BOUNDARY ELEMENT METHOD (BEM)                    Latest revision: 11/18/98
% This program is used to calculate the diffracted intensity pattern
% generated by a DOE using a Boundary Element Method (BEM).  This modified
% version restructures the DOE to suit the geometry of a 1-2 beamsplitter.
%
%       Reference:  Prather, Mirotznik, Mait. Boundary integral methods applied
%                                to the analysis of diffractive optical elements.
%                                J. Opt. Soc. Am. A/Vol.14, No.1 (January 1997).
%*************************************************************************
*
clear FILENAMES FILES_OUT          % used later
% Select a file for a device and analysis type

parameters='SPLITTER_BEMfile02.txt';                % file containing the inputs
% Opens file containing the following input parameters :
%-------------------------------------------------------------------------
%       lam             = incident wavelength
%       n1              = index of refraction of medium 1 (air)
%       n2              = index of refraction of DOE (silicon)
%       P               = Number of partitions of DOE profile (if there is re-sampling)
%       Q               = Number of quantized DOE levels (0 means no quantization)
%       aper_max= max value of "sampled" aperture [microns]
%       aper_min= min value of "sampled" aperture [microns]
%       samp    = number of samples along DOE contour per transverse feature
%       IntPts  = # of points to take in the interpolation in finding Y & Z matrices
%       Uamp    = magnitude of incident field
%       Uang    = ccw angle k vector makes with positive x axis [radians]
%       xUmax = largest  position of field incident on DOE [microns]
%       xUmin  = smallest position of field incident on DOE [microns]
%       bess    = number of interpolation points in Hankel function approximation
%       yzpart  = number of partitions in Y & Z matrix calc. (memory considerations)
%       Zint    = distance to an intermediate plane [microns]
%       Zfar    = distance to the image plane [microns]
%       x0_max          = max position in image plane [microns]

%       x0_min          = min position in image plane [microns]

%       dx0             = spacing in image plane [microns]
%       filename= file containing the scalar data.
%-------------------------------------------------------------------------
fid = fopen(parameters,'r');
for k=1:5,fgets(fid);end;
for k=1:22,temp = fscanf(fid,'%s',1);a(k)=fscanf(fid,'%g\n',1);end;
fgets(fid);temp=fscanf(fid,'%s',1);
geomtype=fscanf(fid,'%s\n',1);          % 'opened' or 'closed' geometric boundary contour
temp=fscanf(fid,'%s',1);
data_save=fscanf(fid,'%s\n',1);         % Save computed data? (Y/N)
```

```
temp=fscanf(fid,'%s',1);                % type of field transformation:
xform=fscanf(fid,'%s\n',1);             % 'angularspectrum', 'fresnel', or 'fraunhoffer'.
temp=fscanf(fid,'%s',1);                % units of image plane positions:
x0units=fscanf(fid,'%s\n',1);           % 'microns', 'centimeters', or 'millimeters'
temp=fscanf(fid,'%s',1);
filenumber=fscanf(fid,'%g\n',1);% # of files
for casenumber=1:filenumber,
temp=fscanf(fid,'%s',1);
FILENAMES(casenumber,:)=fscanf(fid,'%s',1);          % file containing scalar data
FILES_OUT(casenumber,:) =fscanf(fid,'%s\n',1);       % partial names assigned to output
end

fclose(fid);
lam                 = a(1);
n1                  = a(2);
n2                  = a(3);
P                   = a(4);
Q                   = a(5);
aper_max            = a(6);
aper_min            = a(7);
samp                = a(8);
IntPts              = a(9);
Uamp                = a(10);
Uang                = a(11);
xUmax               = a(12);
xUmin               = a(13);
gauss_order         = a(14);% order of super_gaussian modelling the incident field
bess                = a(15);
yzpart              = a(16);
Zint                = a(17);% distance to intermediate plane
Zfar                = a(18);% distance to image plane
x0_max                      = a(19);
x0_min                      = a(20);
dx0                         = a(21);
FFTpower            = a(22);                clear a fid temp

time1=cputime;          % For timing the program runs
for casenumber=1:filenumber,            % Loop for batch mode
        filename=FILENAMES(casenumber,:);
        file_out=FILES_OUT(casenumber,:);
if prod(filename(length(filename)-3:length(filename))=='.txt'),    % scalar file ?
%-------------------------------------------------------------------------------
% Open a file containing scalar data.
fid = fopen(filename);
                data_file = fscanf(fid,'%g %g',[2 inf]); % Data has two rows.
        data_file = data_file';     % Must be transposed-> 2 columns.
fclose(fid); % closes file
%-------------------------------------------------------------------------------
% Extract DOE thicknesses and corresponding locations in object plane.
Xdat=data_file(:,1);       % object positions (in ascending order & equally spaced)
```

```
Tdat=data_file(:,2);       % thickness profile
Ldat=length(Xdat);                 % length of input file

if (lower(geomtype(1))=='o')&(n2==1),Tdat=Tdat-max(Tdat);end; % etch INTO material
% Adjust profile if there is re-sampling (also flips data for ccw BEM format)
if P==0,P=length(Xdat);end                                          % default case (no re-sampling)
width=max(Xdat)-min(Xdat)+abs(Xdat(2)-Xdat(1));                % lateral width of DOE
[microns]
X = mean(Xdat)+(linspace(1,-1,P)*(P-1)/P).'*width/2;% positions now in descending order
d = rect((RowPad(Xdat.',P)-ColPad(X,length(Xdat)))*P/width); % intermediate variable
T = sum(RowPad(Tdat.',P).*d,2)./sum(d,2);       clear d

Treset=find(abs(T)==min(abs(T)));Treset=T(Treset(1));
T = T-Treset;    % This is the adjusted thickness profile with re-sampling and reset
samp1=round(samp*Ldat/P);
%-----------------------------------------------------------------------------------
% Quantize etch depth levels (note: if Q=0 -> no quantization)
T=quantize(T,Q);                          % quantization is done after re-sampling
%-----------------------------------------------------------------------------------
% Put scalar data in appropriate form for BEM
% NOTE: DOE contour follows a ccw direction (i.e. positions in descending order)
mfs=abs(X(2)-X(1));     % This is the minimum feature size of the DOE
dx1=mfs/samp1;                  % size of transverse sample
% Consider the boundary outside aperture.
xright=fliplr((max(X)+mfs:mfs: aper_max));              tright=zeros(size(xright));
xleft =(min(X)-mfs:-mfs:aper_min);                              tleft =zeros(size(xleft));

X1=[xright,X.',xleft];
T1=[tright,T.',tleft];
L1=length(X1);                                                      % Length of appended data file

X2=RowPad(X1,samp1+1)+ColPad(linspace(1,-1,samp1+1).',L1)*mfs/2;
T2=RowPad(T1,samp1+1);                                              % Matrix containing all samples per
feature

m=abs((T1(2:L1)-T1(1:L1-1))/dx1);       % Decide how well to sample edges
Ns=(m<1)+(m>=1).*(round(m)+1);                  % Ns is the number of samples on edge

xdoe=X2(1,1);                                                       % initialize with first values
tdoe=T2(1,1);                                                       % from right-most DOE position
for ii=1:L1-1,
        if Ns(ii)==1,     % considering edge effects
                tadd=0.5*(T2(samp1+1,ii)        +T2(1,ii+1));% just take midpoints for given
edge
        else                            % otherwise sample edge
                tadd=linspace(T2(samp1+1,ii),T2(1,ii+1),Ns(ii)).';
        end
        xdoe=[xdoe;X2((2:samp1),ii);(X1(ii)-mfs/2)*ones(Ns(ii),1)];
        tdoe=[tdoe;T2((2:samp1),ii);tadd];% append values for top surfaces & edges
end
```

```
xdoe=[xdoe;X2((2:samp1+1),L1)];        % append final values after last edge
tdoe=[tdoe;T2((2:samp1+1),L1)];                              % FINAL DOE PROFILE

clear X1 T1 X2 T2 xright tright xleft tleft % save computer memory
Nsize=length(xdoe);                          % number of samples on DOE surface
else
        eval(filename); % If not a scalar file input
end
%-------------------------------------------------------------------------------
% Find the shifted values for the DOE positions and thickness profile.
% NOTE: it is assumed at edges that thickness does not change (unlike for "closed" contours).
x_plus=[xdoe(2:Nsize);xdoe(Nsize)-dx1];
x_minus=[xdoe(1)+dx1;xdoe(1:Nsize-1)];
t_plus=[tdoe(2:Nsize);tdoe(Nsize)];
t_minus=[tdoe(1);tdoe(1:Nsize-1)];
% Find the differential line segments for contour integration
diffLP=sqrt((x_plus-xdoe).^2+(t_plus-tdoe).^2);
diffLM=sqrt((x_minus-xdoe).^2+(t_minus-tdoe).^2);

% Find the exterior angles of a DOE profile for BEM.
% NOTE: normal vectors point outward from region 2.

a1=atan2(t_minus-tdoe,x_minus-xdoe);  % ccw angle w.r.t positive x axis
a2=atan2(t_plus-tdoe,x_plus-xdoe);

theta=mod(a1-a2,2*pi);                                  % exterior angles
theta=theta.*(theta<1.99*pi);                  % checks for round-off error
normal_ang=mod(a2+theta/2,2*pi);               % This is the angle of the normal vector
ang_plus=mod(a2+pi/2,2*pi);                    % gives the Nth normal angle for THE EDGES
ang_minus=mod(a1-pi/2,2*pi);                   % gives the (N-1)th normal angle for THE
EDGES
%-------------------------------------------------------------------------------
% Display run statistics to Command window.
fprintf(['-------------------------------------------------------------------------------\n']);
fprintf(['\n\nRunning BEM for file:\t',filename,'\t\t']);
fprintf([datestr(now,8),'\t',datestr(now,1),'\t',datestr(now,16),'\n\n\n']);
fprintf(['Number of boundary sample points:\t\t',num2str(Nsize),'\n']);
fprintf(['DOE minimum feature size (microns):\t\t',num2str(mfs),'\n']);
fprintf(['Transverse spacing increment(microns):\t',num2str(dx1),'\n']);
fprintf(['Quantization of thickness levels: \t\t',num2str(Q),' \n']);
fprintf(['\nTiming program blocks (units in seconds):\n']);
%*******************************************************************************
% FIND Y AND Z MATRICES THAT DESCRIBE BOUNDARY POINT COUPLING

                        tz2=cputime;
Z2 =diag(1-theta/(2*pi))+YZcoupling(xdoe,tdoe,lam/n2,'z',IntPts,bess,yzpart);
                        fprintf(['\n\nz2 time:\t\t\t',num2str(cputime-tz2),'\n']);

                        tz1=cputime;
Z1 =diag(theta/(2*pi))-YZcoupling(xdoe,tdoe,lam/n1,'z',IntPts,bess,yzpart);
```

```
                        fprintf(['z1 time:\t\t\t',num2str(cputime-tz1),'\n'])

                        ty2=cputime;
Y2 = YZcoupling(xdoe,tdoe,lam/n2,'y',IntPts,bess,yzpart);
                        fprintf(['y2 time:\t\t\t',num2str(cputime-ty2),'\n'])
                        ty1=cputime;
Y1 = YZcoupling(xdoe,tdoe,lam/n1,'y',IntPts,bess,yzpart);
                        fprintf(['y1 time:\t\t\t',num2str(cputime-ty1),'\n']);

%*************************************************************************
% Introduce incident fields -> uses exp(-jkz) notation
k1=2*pi*n1/lam;
Uinc=Uamp*super_gaussian(xdoe,xUmax-xUmin,gauss_order).*exp(-
j*k1*(sin(Uang)*tdoe+cos(Uang)*xdoe));
Un=-j*k1.*Uinc.*(sin(Uang)*sin(normal_ang)+cos(Uang)*cos(normal_ang));
                                                        % Normal derivative of
incident field
%-------------------------------------------------------------------------
% Find scattered fields and normal derivatives using LU decomposition

                                ts=cputime;      % TE case
[Esc_te,Qsc_te]=Scattered_field2(Z1,Y1,Z2,Y2,geomtype,Uinc,Un);
                                fprintf(['TE inversion:\t\t',num2str(cputime-ts),'\n']);

                                ts=cputime; % TM case
[Esc_tm,Qsc_tm]=Scattered_field2(Z1,Y1*n1^2,Z2,Y2*n2^2,geomtype,Uinc,Un);
                                fprintf(['TM inversion:\t\t',num2str(cputime-ts),'\n'])


clear Y1 Y2 Z1 Z2               % save computer memory
%-------------------------------------------------------------------------
% Find total field in the image plane
%end   % FINISH TESTING INTERMEDIATE PLANE
x0_bem = (x0_min:dx0:x0_max).';             % positions in image plane [microns]

                                tfin=cputime;    % TE case
E=Total_Field11(xdoe,tdoe,lam/n2,Zint,x0_bem,Esc_te,Qsc_te,bess,yzpart);
                                fprintf(['TE field calc:\t\t',num2str(cputime-
tfin),'\n']);

                                tfin=cputime;    % TM case
H=Total_Field11(xdoe,tdoe,lam/n2,Zint,x0_bem,Esc_tm,n2^2*Qsc_tm,bess,yzpart);
                                fprintf(['TM field calc:\t\t',num2str(cputime-
tfin),'\n']);
%-------------------------------------------------------------------------
% Propagate to image plane via scalar field transformation
if (Zint<Zfar)&(Zint>max(tdoe)),                        tfin=cputime;


        lx0=length(x0_bem);
```

```
        DFTsize=power(2,FFTpower);
        x0_bem=[(0:DFTsize/2-1),(-DFTsize/2:-1)].'*dx0+mean(x0_bem);        % object plane
positions [microns]

        E=[E;zeros(DFTsize-lx0,1)];
        E=shiftud(E,-floor(lx0/2),1);

        H=[H;zeros(DFTsize-lx0,1)];
        H=shiftud(H,-floor(lx0/2),1);
%-------------------------------------------------------------------------

% Physical constants
dn=n2-n1;
mu=4*pi*10^(-7);                % permeability of free space [Henrys/m]
epsilon0=8.8541*10^(-12);       % permittivity of free space [Farads/m]
c=10^(-9)/sqrt(mu*epsilon0);    % speed of light in units of [microns/femtosecond]
omega=2*pi*c/lam;               % angular frequency of light [radians/femtosecond]
N_0=sqrt(mu/epsilon0);          % impedendence in free space [ohms]
%-------------------------------------------------------------------------
if lower(xform(1:5))=='angul',
        [E,hx]=MaxwellPropagation01('E',E,Zfar-Zint,x0_bem,lam,n2);
        [H,ex]=MaxwellPropagation01('H',n1/N_0*H,Zfar-Zint,x0_bem,lam,n2);
        x0=x0_bem;
else
        [E,x0]=scalar_analysis1(E,x0_bem,Zfar-Zint,lam/n2,xform);
        [E,hx]=MaxwellPropagation01('E',E,0,x0,lam,n2);
        [H,x0]=scalar_analysis1(n1/N_0*H,x0_bem,Zfar-Zint,lam/n2,xform);
        [H,ex]=MaxwellPropagation01('H',H,0,x0,lam,n2);
end;
%-------------------------------------------------------------------------
        x0_bem=fftshift(x0);
        E=fftshift(E);
        hx=fftshift(hx);
        H=fftshift(H);
        ex=fftshift(ex);
                fprintf([xform,' xform:\t\t',num2str(cputime-tfin),'\n']);
end
%-------------------------------------------------------------------------------
% Units of x0 positions
if lower(x0units(1:5))=='milli',x0_bem=x0_bem/1000;end;
if lower(x0units(1:5))=='centi',x0_bem=x0_bem/10000;end;
%-------------------------------------------------------------------------------

assignin('base',['SEBEM',file_out],0.5*real(-E.*conj(hx)));
assignin('base',['SHBEM',file_out],0.5*real( ex.*conj(H)));
%-------------------------------------------------------------------------------

X_temp=['XBEM',file_out];                       % assign variable names to output
E_temp=['EBEM',file_out];
H_temp=['HBEM',file_out];
```

```
assignin('base',X_temp,x0_bem);                         % x0 positions
assignin('base',E_temp,E);              % E field values
assignin('base',H_temp,H);              % H field values

assignin('base',[E_temp,'PHASE'],angle(E));             % E field phase values
assignin('base',[H_temp,'PHASE'],angle(H));             % H field phase values

assignin('base',['EBEMapertureAmp',file_out],abs(Esc_te));      % checking fields along contour
assignin('base',['EBEMaperturePha',file_out],mod(angle(Esc_te)-
angle(Esc_te(ceil(Nsize/2)))+pi,2*pi));
assignin('base',['HBEMapertureAmp',file_out],abs(Esc_tm));
assignin('base',['HBEMaperturePha',file_out],mod(angle(Esc_tm)-
angle(Esc_tm(ceil(Nsize/2)))+pi,2*pi));

                if upper(data_save(1))=='Y',    % save data?
                                fid = fopen([file_out,'.txt'],'w');
                                fprintf(fid,'%12.8f
%12.8f\n',[eval(X_temp).';eval(E_temp).';eval(H_temp).']);
                        fclose(fid);
                end
%-------------------------------------------------------------------------------
end     % Closing batch mode
clear FILENAMES FILES_OUT
time2=cputime-time1;   % Time it takes to run program
fprintf(['\nrun time = ',num2str(floor(time2/60)),' min.  ',num2str(rem(time2,60)),' seconds\n']);
fprintf(['-------------------------------------------------------------------------------\n']);
```

```
function yz=YZcoupling(XDOE,TDOE,LAM,YZTYPE,INTPTS,BESS,YZPART);
%-------------------------------------------------------------------------
%  yz=YZcoupling(XDOE,TDOE,LAM,YZTYPE,INTPTS,BESS,YZPART) Latest revision:
        11/24/98
%
%               This function is used for finding the Y and Z matrices in BEM
%               for a particular medium which relates the coupling among
%               boundary points along the DOE contour.  Note that for Z matrices,
%               the singularities are not considered.
%
%       XDOE are the lateral positions of DOE           [microns]       (column vector)
%               TDOE are the corresponding etch depths          [microns]       (column vector)
%               LAM is the REDUCED wavelength                                   [microns]
%               YZTYPE is eithier the Y or Z matrices (must be 'y' or 'z')
%               INTPTS are the # of interpolation points
%               BESS are the # of interpolation points for Hankel function approx.
```

```
%                   YZPART is # of partitions of matrix in Hankel fucntion approx.(save computer
memory)
%-------------------------------------------------------------------------
if nargin<7,YZPART=1;  end    % set default value for partitions
if nargin<6,BESS=10000;end    % set default value for Hankel interpolation
u=(linspace(-1,1,INTPTS)*(INTPTS-1)/INTPTS).';% sample interpolation pts
K=2*pi/LAM;                         % wavenumber in medium
C0=-0.25*j*0.5;
C1= K*0.25*j*0.5;                % coefficients of Green's functions

% Find the shifted values for the DOE positions and thickness profile.
Nsize=length(XDOE);
dx1=abs(XDOE(1)-XDOE(2));  %                         for an open DOE contour
dt1=abs(TDOE(1)-TDOE(2));
dL1=sqrt(dx1.^2+dt1.^2);
Dmax=sqrt((XDOE(1)-XDOE(Nsize)).^2+(TDOE(1)-TDOE(Nsize)).^2);  % Distance between
endpoints
if Dmax > 1.5*dL1,GEOMTYPE='opened';else GEOMTYPE='closed';end;
%                   GEOMTYPE opened or closed boundary surfaces          (must be 'opened' or
'closed')
if GEOMTYPE=='opened',        % NOTE: it is assumed at edges that thickness does not change.
        XPLUS=[XDOE(2:Nsize);XDOE(Nsize)-dx1];
        XMINUS=[XDOE(1)+dx1;XDOE(1:Nsize-1)];            % plus & minus positions
        TPLUS=[TDOE(2:Nsize);TDOE(Nsize)];
        TMINUS=[TDOE(1);TDOE(1:Nsize-1)];
        % Find the exterior angles of a DOE profile for BEM.
        % NOTE: normal vectors point outward from region 2.
        a1=atan2(TMINUS-TDOE,XMINUS-XDOE);   % ccw angle w.r.t positive x axis
        a2=atan2(TPLUS-TDOE,XPLUS-XDOE);
        ANGPLUS =mod(a2+pi/2,2*pi);          % gives the Nth normal angle for THE EDGES
        ANGMINUS=mod(a1-pi/2,2*pi);          % gives the (N-1)th normal angle for THE
EDGES
end

if GEOMTYPE=='closed',                % Assuming a closed geometric boundary contour
        XPLUS=shiftud(XDOE,-1,1);    % find n+1 nodes
        TPLUS=shiftud(TDOE,-1,1);
        XMINUS=shiftud(XDOE,1,1);  % find n-1 nodes
        TMINUS=shiftud(TDOE,1,1);   % NOTE: XDOE & TDOE must be column vectors
        % Find the exterior angles of a DOE profile for BEM.
        % NOTE: normal vectors point outward from region 2.
        a1=atan2(TMINUS-TDOE,XMINUS-XDOE);   % ccw angle w.r.t positive x axis
        a2=atan2(TPLUS-TDOE,XPLUS-XDOE);
        ANGPLUS =mod(a2-pi/2,2*pi);          % gives the Nth normal angle for THE EDGES
        ANGMINUS=mod(a1+pi/2,2*pi);          % gives the (N-1)th normal angle for THE
EDGES
end

% Find the differential line segment for contour integration
Lp=sqrt((XPLUS-XDOE).^2+(TPLUS-TDOE).^2);                % Lp are lengths for n row
```

```
Lm=sqrt((XDOE-XMINUS).^2+(TDOE-TMINUS).^2);          % Lm are lengths for n-1 row
%-------------------------------------------------------------------------
% Introduce interpolation functions, w1 and w2
w1=interpolate(u,1);
w2=interpolate(u,2);
%-------------------------------------------------------------------------
X=RowPad(XDOE.',Nsize);
T=RowPad(TDOE.',Nsize);
Xt=X.';
Tt=T.';
%-------------------------------------------------------------------------
if lower(YZTYPE)=='z',          % to find Z matrix
        %-----------------------------------------------------------------
        Xc=RowPad(XPLUS.',Nsize);
        Tc=RowPad(TPLUS.',Nsize);                   % Take care of the "Nth" elements
        Lc=RowPad(Lp.',Nsize);
        Ac=RowPad(ANGPLUS.',Nsize);
        %-----------------------------------------------------------------
        yz=0;
        for n=1:INTPTS,
                X1=(X.*w1(n)+Xc.*w2(n))-Xt;  %check this!!!
                T1=(T.*w1(n)+Tc.*w2(n))-Tt;
                A1=cos(Ac).*X1+sin(Ac).*T1;          % for the nth edge
                T1=sqrt(X1.^2 +  T1.^2);
                X1=w1(n).*Lc.*Hankel_fcn3(K.*T1,1,BESS,YZPART);
                yz=yz+X1.*A1./T1;
        end;                    clear Xc Tc Lc Ac X1 T1 A1
        %-----------------------------------------------------------------
        Xc=RowPad(XMINUS.',Nsize);
        Tc=RowPad(TMINUS.',Nsize);                  % Take care of the "(N-1)th" elements
        Lc=RowPad(Lm.',Nsize);
        Ac=RowPad(ANGMINUS.',Nsize);

        %-----------------------------------------------------------------
        X2=0;
        for n=1:INTPTS,
                X1=(Xc.*w1(n)+X.*w2(n))-Xt;
                T1=(Tc.*w1(n)+T.*w2(n))-Tt;
                A1=cos(Ac).*X1+sin(Ac).*T1;          % for the (n-1)th edge
                T1=sqrt(X1.^2+T1.^2);
                X1=w2(n).*Lc.*Hankel_fcn3(K.*T1,1,BESS,YZPART);
                X2=X2+X1.*A1./T1;
        end;                    clear X T Xt Tt Xc Tc Lc Ac X1 T1 A1
        %-----------------------------------------------------------------
        yz=C1*(yz+X2)*2/INTPTS;                     % Z MATRIX (not including
singularities)
end
%-------------------------------------------------------------------------
if lower(YZTYPE)=='y',          % to find Y matrix
        %-----------------------------------------------------------------
```

```
        Xc=RowPad(XPLUS.',Nsize);
        Tc=RowPad(TPLUS.',Nsize);                   % Take care of the "Nth" elements
        Lc=RowPad(Lp.',Nsize);
        %---------------------------------------------------------------------
        yz=0;
        for n=1:INTPTS,
                X1=(X.*w1(n)+Xc.*w2(n))-Xt;         % for the nth edge
                T1=(T.*w1(n)+Tc.*w2(n))-Tt;
                T1=sqrt(X1.^2 + T1.^2);
                yz=yz+ w1(n).*Lc.*Hankel_fcn3(K.*T1,0,BESS,YZPART);
        end;                    clear Xc Tc Lc X1 T1
        %---------------------------------------------------------------------
        Xc=RowPad(XMINUS.',Nsize);
        Tc=RowPad(TMINUS.',Nsize);                  % Take care of the "(N-1)th" elements
        Lc=RowPad(Lm.',Nsize);
        %---------------------------------------------------------------------
        T2=0;
        for n=1:INTPTS,
                X1=(Xc.*w1(n)+X.*w2(n))-Xt;         % for the (n-1)th edge
                T1=(Tc.*w1(n)+T.*w2(n))-Tt;
                T1=sqrt(X1.^2 + T1.^2);
                T2=T2+w2(n).*Lc.*Hankel_fcn3(K.*T1,0,BESS,YZPART);
        end;                    clear Xc Tc Lc X T Xt Tt X1 T1
        %---------------------------------------------------------------------
        yz=C0*(yz+T2)*2/INTPTS;                                         % Y MATRIX
end
%-------------------------------------------------------------------------------
```

```
function [Esc,Qsc]=Scattered_field2(P,Q,R,S,geomtype,Uinc,Qinc);
%-------------------------------------------------------------------------
% Find scattered fields and normal derivatives using LU decomposition and
% matrix inversion via partitioning.  This function is used in BEM and is
% effective in saving computer memory for huge matrix calculations.
%       Reference: Numerical Recipies in C. (1998).
%
%       [Esc,Qsc]=Scattered_field2(P,Q,R,S,geomtype,Uinc,Qinc)
%
%       P,R,Q,S=        boundary point coupling matrices (NxN matricies)
%       geomtype= opened or closed boundary surfaces  (must be 'opened' or 'closed')
%       Uinc            =       Incident field                  (Nx1 vector)
%       Qinc            =       Normal derivative of inc. field (Nx1 vector)
%
%       Esc             =       Scattered field on DOE contour  (Nx1 vector)
%       Qsc             =       Normal deriv. of field on DOE contour(Nx1 vector)
%
%       Created 11/27/98        -       Latest revision: 12/11/98
```

```
%-------------------------------------------------------------------------
if lower(geomtype(1))=='o',% Assuming an opened geometric boundary contour
        Qinc=zeros(size(Qinc));% Normal derivative of incident field is immaterial

        [L,S]=lu(-S);L=L\R;S=S\L;       clear L R       % LU decomposition
        P=P-Q*S;                        clear Q
        [P,U]=lu(P);y=P\Uinc;Esc=U\y;   clear y U P     % scattered field
        Qsc=-S*Esc;                                     % Normal derivative of scattered field

end
%-------------------------------------------------------------------------
if lower(geomtype(1))=='c',% Assuming a closed geometric boundary contour

        Qinc=-R*Uinc+S*Qinc;
        Uinc=zeros(size(Uinc));

        [L,Q]=lu(-Q);L=L\P;Q=Q\L;       clear L P       % LU decomposition
        R=R-S*Q;                        clear S
        [R,U]=lu(R);y=R\Uinc;Esc=U\y;   clear y U R     % scattered field
        Qsc=-Q*Esc;                                     % Normal derivative of scattered field
end
%-------------------------------------------------------------------------



function E=Total_Field11(xdoe,zdoe,lam,zdist,x,Esc,Qsc,bs,part);
%-------------------------------------------------------------------------
%       E=Total_Field11(xdoe,zdoe,lam,zdist,x,Esc,Qsc,bs,part)
%
%       This function computes the values of the electric (or magnetic) field
%       values in a specified image plane (x,zdist) given the values of the
%       field and its normal derivative (Esc,Qsc) to the DOE surface.  The
%       calculation is done using Green's 2nd identity.
%
%               E       = electric (magnetic) field in image plane      (column vector)
%                               units of [Volts/m] or [Amps/m]
%               xdoe  = x coordinate of sampled points                  (column vector)*
%               zdoe  = z coordinate of sampled points (etch depth)     (column vector)*
%               lam   = wavelength in medium                                    (scalar)*
%               zdist = distance to image plane                 (scalar or column vector)*
%               x     = x coordinates in IMAGE plane                    (column vector)*
%               Esc   = Scattered field values at
%                               sampled points [V/m]                    (column vector)
%               Qsc   = Normal derivatives of scattered
%                               field  at sampled points [V/m/micron]   (column vector)
%               bs      = # of interpolation points for Hankel function approx. (scalar)
%               part  = # of partions of matrix in Hankel fucntion approx.(save computer
        memory)
%
%
```

```
%       *NOTE: lengths above are in units of microns
%
%       Latest revision: 11/25/98
%
%       Reference:  Prather, Mirotznik, Mait. Boundary integral methods applied
%                               to the analysis of diffractive optical elements.
%                               J. Opt. Soc. Am. A/Vol.14, No.1 (January 1997).
%-------------------------------------------------------------------------
if nargin<9,part=1;  end % set default value for partitions
if nargin<8,bs=10000;end        % set default value for Hankel interpolation
k    = 2*pi/lam;                                % wavenumber in the medium
Nx   =length(x);
Nsize=length(xdoe);
%-----------------------------------------------------------------------------------
% Find the shifted values for the DOE positions and thickness profile.

dx1=abs(xdoe(1)-xdoe(2));       %for an open DOE contour
dz1=abs(zdoe(1)-zdoe(2));
dL1=sqrt(dx1.^2+dz1.^2);
Dmax=sqrt((xdoe(1)-xdoe(Nsize)).^2+(zdoe(1)-zdoe(Nsize)).^2);  % Distance between endpoints
if Dmax > 1.5*dL1,geomtype='opened';else geomtype='closed';end;
%               GEOMTYPE opened or closed boundary surfaces          (must be 'opened' or
'closed')

if geomtype=='opened', % NOTE: it is assumed at edges that thickness does not change.
        x_plus =[xdoe(2:Nsize);xdoe(Nsize)-dx1];
        x_minus=[xdoe(1)+dx1;xdoe(1:Nsize-1)];                  % plus & minus positions
        z_plus =[zdoe(2:Nsize);zdoe(Nsize)];
        z_minus=[zdoe(1);zdoe(1:Nsize-1)];
        % Find the exterior angles of a DOE profile for BEM.
        % NOTE: normal vectors point outward from region 2.
        a1=atan2(z_minus-zdoe,x_minus-xdoe); % ccw angle w.r.t positive x axis
        a2=atan2(z_plus-zdoe,x_plus-xdoe);
        theta=mod(a1-a2,2*pi);                                  % exterior angles
        theta=theta.*(theta<1.9999*pi);                 % checks for round-off error
        ang=mod(a2+theta/2,2*pi);               % This is the ccw angle of the normal vector wrt
                                                        x-axis
end

if geomtype=='closed',          % Assuming a closed geometric boundary contour
        x_plus =shiftud(xdoe,-1,1);     % find n+1 nodes
        x_minus=shiftud(xdoe,1,1);      % find n-1 nodes
        z_plus =shiftud(zdoe,-1,1);
        z_minus=shiftud(zdoe,1,1);      % NOTE: XDOE & TDOE must be column vectors
        % Find the exterior angles of a DOE profile for BEM.
        % NOTE: normal vectors point outward from region 2.
        a1=atan2(z_minus-zdoe,x_minus-xdoe); % ccw angle w.r.t positive x axis
        a2=atan2(z_plus-zdoe,x_plus-xdoe);
        theta=mod(a2-a1,2*pi);                  % exterior angles
        theta=theta.*(theta<1.9999*pi);         % checks for round-off error
```

```
        ang=mod(a1+theta/2,2*pi);               % ccw angle of the normal vector for CLOSED
contour
end

% Find the differential line segments for contour integration
diffL =sqrt((x_plus-xdoe).^2+(z_plus-zdoe).^2);          % nth sampled point
diffLM=sqrt((x_minus-xdoe).^2+(z_minus-zdoe).^2);    % (n-1)th sampled point

clear x_plus x_minus z_plus z_minus a1 a2 theta
%-------------------------------------------------------------------------------
% Put positions in matrix form
x2 =ColPad(x,Nsize);
xt=xdoe.';                  x2t=RowPad(xt,Nx);
zt=zdoe.';                  z2t=RowPad(zt,Nx);
if length(zdist)>1, zdist=ColPad(zdist,Nsize);end;
%-------------------------------------------------------------------------
% Find all the relative distances
deltaX=x2t-x2;
deltaZ=z2t-zdist;
clear x2t x2 z2t xt zt zdist                % save computer memory
deltaR=sqrt(deltaX.^2+deltaZ.^2);       % Relative distance matrix
%-------------------------------------------------------------------------
% Calculate Green's function and its normal derivative (g0 and gn)
% Using large arguememt approximations for Hankel functions
g1= 0.25*j*k*Hankel_fcn3(k*deltaR,1,bs,part)./deltaR;  % first order, H1.
gx= g1.*deltaX;clear deltaX
gz= g1.*deltaZ; clear deltaZ g1
SinAng=sin(ang.');              SinAng=RowPad(SinAng,Nx);
CosAng=cos(ang.');              CosAng=RowPad(CosAng,Nx);
gn =CosAng.*gx+SinAng.*gz;                  % normal derivative
clear SinAng CosAng gx gz
g0=-0.25*j*Hankel_fcn3(k*deltaR,0,bs,part);    % Hankel function 2nd kind, zeroth order
clear deltaR                                        % save computer memory
%-------------------------------------------------------------------------
% Use Equation (A-12) from Prather paper to find the Scattered field values
% at image plane by contour integration over DOE boundary:
diffD=0.5*(diffL+diffLM);       % average differential length segments
E= g0*(Qsc.*diffD)-gn*(Esc.*diffD);    % TOTAL SCATTERED FIELD
%-------------------------------------------------------------------------



function [ey,hx,hz]=MaxwellPropagation01(analysis_type,Ey_aper,Z_prop,x0_fdtd,lam,n2);
%-------------------------------------------------------------------
% function [ey,hx,hz]=MaxwellPropagation01(analysis_type,Ey_aper,Z_prop,x0_fdtd,lam,n2);
% Maxwell field component propagation of time-harmonic steady-state
% fields to detector plane.         Created: 10/4/99
%-------------------------------------------------------------------
```

```
% Physical constants
mu=4*pi*10^(-7);                        % permeability of free space [Henrys/m]
epsilon0=8.8541*10^(-12);       % permittivity of free space [Farads/m]
c=10^(-9)/sqrt(mu*epsilon0);% speed of light in units of [microns/femtosecond]
omega=2*pi*c/lam;                       % angular frequency of light [radians/femtosecond]
period=lam/c;                           % period of incident wave [femtoseconds]
N_0=sqrt(mu/epsilon0);


if upper(analysis_type)=='E',
        if nargout<3,
        [ey,hx]=AS_derivatives(Ey_aper,x0_fdtd,Z_prop,lam/n2,'0z');
        hx=-j*c/(omega*N_0)*hx;
        hz= [];
        else
        [ey,hx,hz]=AS_derivatives(Ey_aper,x0_fdtd,Z_prop,lam/n2,'0zx');
        hx=-j*c/(omega*N_0)*hx;
        hz= j*c/(omega*N_0)*hz;
        end
        %stez=0.5*real(-ey.*conj(hx));
        %stex=0.5*real( ey.*conj(hz));
end
if upper(analysis_type)=='H',
        if nargout<3,
                [ey,hx]=AS_derivatives(Ey_aper,x0_fdtd,Z_prop,lam/n2,'0z');
                hx= j*c*N_0/(omega*n2^2)*hx;
                hz=[];
        else
                [ey,hx,hz]=AS_derivatives(Ey_aper,x0_fdtd,Z_prop,lam/n2,'0zx');
                hx= j*c*N_0/(omega*n2^2)*hx;
                hz=-j*c*N_0/(omega*n2^2)*hz;
                %stmz=0.5*real( ex.*conj(hy));
                %stmx=0.5*real(-ez.*conj(hy));
        end
end
```

APPENDIX E

FDTD analysis codes

This appendix contains the computer programs used for the FDTD analysis of diffractive optical elements presented in this dissertation. Note that the codes are written in Matlab®.

```
%*************************************************************************

% Two-dimensional formulism of the Finite Difference Time-Domain
% method (FDTD).  References: A.Taflove, Computational Electrodynamics: The
% Finite-Difference Time Domain Method (1995),
% G.S. Smith, Classical Electromatic Radiation, pp. 71-77 and K.S. Yee (1966).
%                                 Last update:  9/10/99
%-------------------------------------------------------------------------
%       This code simulates the two dimensional TE or TM cases
%       for the 1-2 beamsplitter design.  Note that with this
%       code, 2nd order Mur boundary conditions are used.
%*************************************************************************
Start_time=cputime;     % Time program run
clear FILENAMES FILES_OUT          % for later use
% Open file containing the following input parameters :
%-------------------------------------------------------------------------
%       lam             = incident wavelength [microns]
%       n1              = index of refraction of medium 1 (air)
%       n2              = index of refraction of DOE (silicon)
%       P               = Number of partitions of DOE profile (if there is re-sampling)
%       Q               = Number of quantized DOE levels (0 means no quantization)
%       x_right  = max value of "sampled" aperture [microns]
%       x_left   = min value of "sampled" aperture [microns]
%       Yee             = Number of Yee cells along the x direction
%       Aspect  = Aspectratio of change in x over z of grid cells
%       samp1   = number of samples along DOE contour per transverse feature
%       Uamp    = magnitude of incident field     [Volts/m or Amps/m]
%       Uang    = ccw angle k vector makes with positive x axis [radians]
%       xUmax = largest  position of field incident on DOE [microns]
%       xUmin  = smallest position of field incident on DOE [microns]
%       DFTsize= power of FFT in near to far field transformation (2^N)
%       z_dist   = distance to the image plane [microns]
%       x0_max = max position in image plane [microns]
%       x0_min = min position in image plane [microns]
%       Nmax    = # of iterations
%       MagInt  = incident inensity [Watts/cm^2]
%       stateTOG= for permittivity spatial averaging      (Y/N)
%       filenumber              : number of scalar files to evaluate
%       filename                : file containing the scalar data.
%       analysistype    :       'E', 'H', or 'both'
%       BCtype                  :type of boundary conditions (2nd order Mur)
%       xform                   :type of field transformation
%-------------------------------------------------------------------------
MasterFile='MasterFDTD01.txt';                  % Master batch file including options
DispVals=20;                                            % which energy valuses to display
DispXpos='N';                                           % set x-range for detector plane
quantities
fprintf(['\n-------------------------------------------------------------------------------\n']);
in_file=input(['Enter FDTD master file ( default-> ',MasterFile,' ):   ']);
```

```
if not(isempty(in_file)),MasterFile=in_file;end;clear in_file

out_temp=input(['Enter give all detector plane values ( default-> ',DispXpos,' ):   ']);
if not(isempty(out_temp)),DispXpos=out_temp;end;clear out_temp

disptemp=input(['Enter Number of energy display intervals ( default-> ',num2str(DispVals),' ):
']);
if not(isempty(disptemp)),DispVals=disptemp;end;clear in_file disptemp
fid = fopen(MasterFile,'r');
for k=1:5,fgets(fid);end;


temp=fscanf(fid,'%s\n',1);
temp=fscanf(fid,'%s',1);
symmetric =fscanf(fid,'%s',1);
symmetric_script =fscanf(fid,'%s\n',1);              % Symmetric DOE analysis?

temp=fscanf(fid,'%s',1);
MovieFrames =fscanf(fid,'%s',1);
MovieScript =fscanf(fid,'%s\n',1);                          % Movie making option

temp=fscanf(fid,'%s',1);
ASfiltering =fscanf(fid,'%s',1);
ASfiltering_script =fscanf(fid,'%s\n',1);  % script file for Ang. Spect. analysis

temp=fscanf(fid,'%s',1);
poyntingcalc =fscanf(fid,'%s',1);
poyntingcalc_script =fscanf(fid,'%s\n',1);          % script file for Poynting vector analysis

temp=fscanf(fid,'%s',1);
fieldimageplot =fscanf(fid,'%s',1);
fieldimageplot_script =fscanf(fid,'%s\n',1);% script file for field image plotting

temp=fscanf(fid,'%s',1);
polarstudy =fscanf(fid,'%s',1);
polarstudy_script =fscanf(fid,'%s\n',1);   % script file for polarization analysis

temp=fscanf(fid,'%s',1);
fieldgridsave =fscanf(fid,'%s\n',1);                % saving field grid values?

temp=fscanf(fid,'%s',1);
file_storage =fscanf(fid,'%s\n',1);                         % file to store data (if nec.)


temp=fscanf(fid,'%s',1);
pre_processing =fscanf(fid,'%s',1);
pre_processing_script =fscanf(fid,'%s\n',1);% Pre-processing of transmission fcn. (if nec.)

temp=fscanf(fid,'%s',1);
post_processing =fscanf(fid,'%s',1);
post_processing_script =fscanf(fid,'%s\n',1);% Post-processing of analysis (if nec.)
```

```
temp=fscanf(fid,'%s',1);
number_of_runs =fscanf(fid,'%g\n',1); % if running script file input multiple # of times


temp=fscanf(fid,'%s',1);
paranumber=fscanf(fid,'%g\n',1);          % # of parameter files
PARAM=[''];
for k=1:paranumber,
temp=fscanf(fid,'%s',1);
parameters =fscanf(fid,'%s\n',1);% partial names assigned to output
PARAM=char(PARAM,parameters);
end
PARAM(1,:)=[];
fclose(fid);
%*************************************************************************************

%-------------------------------------------------------------------------------
xspacing=0;      % Movie making option
zspacing=0;
Tincrement=20;% default value
MovieFields=['Ey'];
if upper(MovieFrames(1))=='Y',
        fprintf(['\nRunning Movie making option:  ',MovieScript,' \n']);
        MovieFieldsTemp=input(['Enter field type [xyz] components:   ']);
        if not(isempty(MovieFieldsTemp)),MovieFields=MovieFieldsTemp;end;
        xspacingtemp=input(['Enter x spacing (µm):   ']);
        if not(isempty(xspacingtemp)),xspacing=xspacingtemp;end;
        zspacingtemp=input(['Enter z spacing (µm):   ']);
        if not(isempty(zspacingtemp)),zspacing=zspacingtemp;end;
        Tincrementstemp=input(['Enter # frames/period (default=',num2str(Tincrement),'):   ']);
        if not(isempty(Tincrementstemp)),Tincrement=Tincrementstemp;end;
end;clear xspacingtemp zspacingtemp Tincrementstemp MovieFieldsTemp
%-------------------------------------------------------------------------------
% For Angular spectrum and filtered field analysis
fxmin=-10^(10);fxmax= 10^(10);fxspace=0;
if upper(ASfiltering(1))=='Y',
        fprintf(['\nRunning Angular spectrum and filtering option:  ',ASfiltering_script,' \n']);

        fxmintemp=input(['Enter FX min (µm^(-1)):   ']);
        if not(isempty(fxmintemp)),fxmin=fxmintemp;end;

        fxmaxtemp=input(['Enter FX max (µm^(-1)):   ']);
        if not(isempty(fxmaxtemp)),fxmax=fxmaxtemp;end;

        fxspacetemp=input(['Enter FX spacing (µm^(-1)):   ']);
        if not(isempty(fxspacetemp)),fxspace=fxspacetemp;end;
end;    clear fxspacetemp fxmaxtemp fxmintemp
%-------------------------------------------------------------------------------
XSIP=0;          % field image plot option
ZSIP=0;
```

```
if upper(fieldimageplot(1))=='Y',
        fprintf(['\nRunning field image plot option:  ',fieldimageplot_script,' \n']);
        XSIPtemp=input(['Enter x image plot spacing (µm):   ']);
        if not(isempty(XSIPtemp)),XSIP=XSIPtemp;end;
        ZSIPtemp=input(['Enter z image plot spacing (µm):   ']);
        if not(isempty(ZSIPtemp)),ZSIP=ZSIPtemp;end;
        ZDPT=input(['Enter z distance of propagation (µm):   ']);
end;clear XSIPtemp ZSIPtemp ZDPTtemp
%-------------------------------------------------------------------------------
polthresh=0.001;          % polarimetry option
if upper(polarstudy(1))=='Y',
        fprintf(['\nRunning polarimetry option:  ',polarstudy_script,' \n']);
        poltemp=input(['Enter intensity threshold value (default=',num2str(polthresh),'):   ']);
        if not(isempty(poltemp)),polthresh=poltemp;end;
end;clear poltemp
%-------------------------------------------------------------------------------
%*******************************************************************************

for PARAM1=1:paranumber,     % super-batch mode for multiple parameter files
        parameters=PARAM(PARAM1,:);parameters=parameters(not(isspace(parameters)));
%parameters='IASA1to2_FDTDfile03.txt';                  % file containing the inputs
fid = fopen(parameters,'r');
for k=1:5,fgets(fid);end;
for k=1:22,temp = fscanf(fid,'%s',1);a(k)=fscanf(fid,'%g\n',1);end;
fgets(fid);temp=fscanf(fid,'%s',1);
ANALYSIS_TYPES=fscanf(fid,'%s\n',1);
        %*********************************************
temp=fscanf(fid,'%s',1);
BCtype=fscanf(fid,'%s\n',1);
temp=fscanf(fid,'%s',1);                    % type of field transformation:
xform=fscanf(fid,'%s\n',1);                         % 'angularspectrum', 'fresnel', or 'fraunhoffer'.
temp=fscanf(fid,'%s',1);                    % units of image plane positions:
x0units=fscanf(fid,'%s\n',1);               % 'microns', 'centimeters', or 'millimeters'
temp=fscanf(fid,'%s',1);
perm_avg=fscanf(fid,'%s\n',1);              % permittivity spatial averaging ("ON" or "OFF")
temp=fscanf(fid,'%s',1);
data_save=fscanf(fid,'%s\n',1);             % Save computed data? (Y/N)
temp=fscanf(fid,'%s',1);
filenumber=fscanf(fid,'%g\n',1);% # of files
FILENAMES=[''];FILES_OUT=[''];
for casenumber=1:filenumber,
temp=fscanf(fid,'%s',1);
%FILENAMES(casenumber,:)=fscanf(fid,'%s',1);                % file containing scalar data
%FILES_OUT(casenumber,:) =fscanf(fid,'%s\n',1);       % partial names assigned to output
temp1=fscanf(fid,'%s',1);
temp2=fscanf(fid,'%s\n',1);
FILENAMES=char(FILENAMES,temp1);                    % file containing scalar data
FILES_OUT=char(FILES_OUT,temp2);                    % partial names assigned to output
end;FILENAMES(1,:)=[]; FILES_OUT(1,:)=[];clear temp1 temp2

fclose(fid);
```

```
lam                         = a(1);                         %
n1                          = a(2);                         %
n2                          = a(3);                         %
P                           = a(4);                         %
Q                           = a(5);                         %
x_window_right              = a(6);                         % max position of viewing
                                                            window in object plane
x_window_left  = a(7);              % min position of viewing window in object plane
Yee            = a(8);              %
Aspectratio     = a(9);             %
zcush          = a(10);             % # of "extra" Yee cells in z direction
zfront         = a(11);             % if specified, front z position in grid
zback          = a(12);             % if specified, back  z position in grid
Uang           = a(13);             %* angle of incident wave (for now, assume normal
                                                        incidence)
xUmax          = a(14);             %
xUmin          = a(15);             %
FFTpower       = a(16);             %
z_dist         = a(17);             %
x0_max         = a(18);             %
x0_min         = a(19);             %
Nmax           = a(20);             % number of time grid points (program iterations)
MagInt         = a(21);             % intensity of incident field (Watts/cm^2)
TautoPer       = a(22);             % Rise time for incident field

clear a fid temp

for casenumber=1:filenumber,                    % Loop for batch mode
        for runnumber= 1:number_of_runs,        %       Loop for # of runs (if necessary)
        filename=FILENAMES(casenumber,:);filename=filename(not(isspace(filename)));
        file_out=FILES_OUT(casenumber,:);file_out=file_out(not(isspace(file_out)));
        file_out_lett=file_out(find(isletter(file_out)));

%-------------------------------------------------------------------------------------
% Handle any pre-processing
if upper(pre_processing(1))=='Y',eval(pre_processing_script);end;
%-------------------------------------------------------------------------------------
if prod(filename(length(filename)-3:length(filename))=='.txt'),    % scalar file ?
        [Xdat,Zdat]=reading_files(filename);% Reads data from scalar file
        % Xdat = Position along DOE [microns]
        % Z      =DOE etch depth into silicon [microns]
% Then set up FDTD geometry for this case
%-------------------------------------------------------------------------
% Adjust profile if there is re-sampling
[Z,x1]=Repartition1(Zdat,Xdat,P,1,[],[],n2-n1);
mfs=abs(x1(2)-x1(1));   % minimum scalar spacing
L=max(x1)-min(x1)+abs(x1(2)-x1(1));                     % lateral width of DOE [microns]
%-------------------------------------------------------------------------
% Quantize etch depth levels (note: if Q=0 -> no quantization)
Z=quantize(Z,Q);                        % quantization is done after re-sampling
Zmax=max(Z);                            % maximum etch depth
```

```
Zmin=min(Z);                                % minimum etch depth
%-------------------------------------------------------------------------
% For plotting purposes (etch depth profile)
Xdoe=[x_window_left;min(x1)-mfs/2;flatten([x1-
mfs/2,x1+mfs/2].');max(x1)+mfs/2;x_window_right];
Tdoe=[0;0;flatten([Z,Z].');0;0];
assignin('base',['XDOE',file_out],Xdoe);
assignin('base',['TDOE',file_out],Tdoe);
%-------------------------------------------------------------------------
% Physical constants
dn=n2-n1;                                   % change in index
k0=2*pi/lam;                                % wavenumber in free space
mu=4*pi*10^(-7);                            % permeability of free space [Henrys/m]
epsilon0=8.8541*10^(-12);                   % permittivity of free space [Farads/m]
c=10^(-9)/sqrt(mu*epsilon0);% speed of light in units of [microns/femtosecond]
omega=2*pi*c/lam;                           % angular frequency of light [radians/femtosecond]
period=lam/c;                               % period of incident wave [femtoseconds]
N_0=sqrt(mu/epsilon0);                      % impedendence in free space [ohms]
%-------------------------------------------------------------------------
% Yee grid cell spacings
dx=mfs/(2*Yee);                             % spacing in x direction
dz=dx/Aspectratio;                          % spacing in z direction
dX=2*dx;
dZ=2*dz;% Doubled spacings that correspond to Yee cell dimensions.
%-------------------------------------------------------------------------
% Set up geometry along X direction
gridX=dx*(2*(floor(x_window_left/dX)):2*(ceil(x_window_right/dX)));
sX=length(gridX);                   % Number of sampled points along x direction
xEy=gridX(1:2:sX);                  ley=length(xEy);
xHx=gridX(1:2:sX);                  lhx=length(xHx);        % field position values
xHz=gridX(2:2:sX);                  lhz=length(xHz);
%-------------------------------------------------------------------------
% Set up geometry along Z direction
if zcush~=0, % if there is a "cushion"
        gridZ=dz*(2*(floor(Zmin/dZ)-zcush):2*(ceil(Zmax/dZ)+zcush)).';
else                % set grid according to input file if no "cushion"
        gridZ=dz*(2*(floor(zfront/dZ)):2*(ceil(zback/dZ))).';
end;

sZ=length(gridZ);                   % Number of sampled points along z direction
zEy=gridZ(1:2:sZ);                  wey=length(zEy);
zHx=gridZ(2:2:sZ);                  whx=length(zHx);        % field position values
zHz=gridZ(1:2:sZ);                  whz=length(zHz);
%-------------------------------------------------------------------------
% Find corresponding DOE etch depth at each X position
Zc=flatten(RowPad(Z.',2*Yee));
Zc=[0*find(gridX<min(x1-.5*mfs-dx/4)).';([0;Zc]+[Zc;0])/2;...
                                    0*find(gridX>max(x1+.5*mfs+dx/4)).'];
ZcE=Zc(1:2:sX).';       % etch depth at Ey (& Hx) positions
ZcH=Zc(2:2:sX).';       % etch depth at Hz positions
%-------------------------------------------------------------------------
```

```
% Select planes of interest (Front and Back)
front_pos=3;%ceil((d_pos+1)/2);           % this could be at the aperture if desired
mid_pt =ceil((max(Z)-zEy(1))/dZ)+2;%min(find(zEy>=(max(zEy)+max(Z))/2));% midpoint
position of observation point
assignin('base',['Zmidpt',file_out],zEy(mid_pt));
assignin('base',['zEy',file_out],zEy);
assignin('base',['xEy',file_out_lett],xEy);
Zaper=find(zEy==0);                % grid location of aperture (z=0)
% planes of interest (Left and Right)
lf         =round((xUmin-x_window_left+dX)/dX);
rf         =round((xUmax-x_window_left+dX)/dX);
%--------------------------------------------------------------------------
% Introduce time grid
samp_t=ceil(c*period*sqrt(1/dX^2+1/dZ^2));    % time samples per period

if upper(MovieFrames(1))=='Y',samp_t=Tincrement*ceil(samp_t/Tincrement);end;% If
animating via IGOR

dt=period/samp_t;         % time increment [femtoseconds]
time_E=dt*(1:Nmax).';                          % Time vector for E field
Cinc=exp(j*2*pi*n1/lam*(2*dZ));                % const. term for front face
tau=period*TautoPer;                           % Rise time for incident field
phasor_E=Cinc*exp(j*omega*time_E).*(1-exp(-(time_E-dt)/tau));         % E field phasor
%--------------------------------------------------------------------------
% Display FDTD run statistics
fprintf(['\n-------------------------------------------------------------------------------------\n']);
Domain_condition=c*dt*sqrt(1/dX^2+1/dZ^2); % "Domain of dependence condition" Must be<=
unity.
fprintf(['Running FDTD for file:\t',filename,'\t\t']);
fprintf([datestr(now,8),'\t',datestr(now,1),'\t',datestr(now,16),'\n\n\n']);
fprintf(['Stability condition:  ',num2str(Domain_condition),' \n']);
fprintf(['Grid range:  ',num2str(zEy(1)),' < z < ',num2str(zEy(wey)), '\tmicrons\n']);
fprintf(['            ',num2str(xEy(1)),' < x < ',num2str(xEy(ley)), '\tmicrons\n']);
fprintf(['Field grid size:  ',num2str(wey),' by ',num2str(ley),' \n']);
fprintf(['Field point location of finite aperture along z: position # ',num2str(find(zEy==0)),' \n']);
fprintf(['Location of front face of incident grid: position # ',num2str(front_pos),'\t\tz=
',num2str(zEy(front_pos)),' microns\n']);
fprintf(['Location of back  face of incident grid: position # ',num2str(mid_pt),'\t\tz=
',num2str(zEy(mid_pt)),' microns\n']);
fprintf(['maximum DOE etch depth   :\t\t\t\t\t\t\t\tz= ',num2str(max(abs(Z))),' microns \n']);
fprintf(['Yee cell size:  ',num2str(dZ),' by ',num2str(dX),' microns^2\n']);
fprintf(['Aspect ratio:  ',num2str(dX/dZ),' to 1\n']);
fprintf(['Permittivity spatial averaging:  ',perm_avg,' \n']);
fprintf(['Time increment    :  ',num2str(dt),' femtoseconds\t('num2str(samp_t),' samples per
period)','\n']);
fprintf(['Time duration     :  ',num2str(Nmax*dt),' femtoseconds\t(',num2str(Nmax*dt/period),'
time periods)\n']);
fprintf(['# of iterations   :  ',num2str(Nmax),' \n']);
fprintf(['-------------------------------------------------------------------------------\n']);
fprintf(['\titeration','\t\t\ttotal energy\n\t\t\t\t\t\tper area\t\t\t\n\n']);
%--------------------------------------------------------------------------
```

```
% Find relative permittivity at all spatial positions
if (upper(perm_avg(1))=='Y')|(prod(upper(perm_avg)=='ON')),          % Permittivity spatial
averaging routine
        %********
        % nHx2 calculation
        Z2          = RowPad(ZcH,whz);              A=(2:whz);B=(1:whz-1);
        gridZ2      = ColPad(zHz,lhz);              C=(2:lhz);D=(1:lhz-1);
        nHx2        = (gridZ2>=Z2);

        Z2=abs(gridZ2-Z2);      clear gridZ2
        nHx2=(nHx2(A,:).*Z2(A,:)+nHx2(B,:).*Z2(B,:))./(Z2(A,:)+Z2(B,:));      clear Z2
        nHx2=0.5*(nHx2(:,C)+nHx2(:,D));
        nHx2=[nHx2(:,1),nHx2,nHx2(:,lhz-1)];
        nHx2=n1^2+(n2^2-n1^2)*nHx2;          % permittivity at Hx positions
        %********
        % nHz2 calculation
        nEy2    = RowPad(ZcH,whx);              A=(2:whx);B=(1:whx-1);
        gridZ2  = ColPad(zHx,lhz);              C=(2:lhz);D=(1:lhz-1);
        nHz2    = (gridZ2>=nEy2);

        nEy2=abs(gridZ2-nEy2);              clear gridZ2
        nEy2=(nHz2(A,:).*nEy2(A,:)+nHz2(B,:).*nEy2(B,:))./(nEy2(A,:)+nEy2(B,:));
        nHz2=[nHz2(1,:);nEy2;nHz2(whx,:)];
        nHz2=n1^2+(n2^2-n1^2)*nHz2;          % permittivity at Hz positions
        %********
        % nEy2 calculation
        nEy2=0.5*(nEy2(:,C)+nEy2(:,D));              clear A B C D
        nEy2=[nEy2(1,:);nEy2;nEy2(whx-1,:)];
        nEy2=[nEy2(:,1),nEy2,nEy2(:,lhz-1)];
        nEy2=n1^2+(n2^2-n1^2)*nEy2;          % permittivity at Ey positions
        %********
else    % no permittivity averaging
        nHx2=n1^2+(n2^2-n1^2)*(ColPad(zHx,lhx)>=RowPad(ZcE,whx));
        nHz2=n1^2+(n2^2-n1^2)*(ColPad(zHz,lhz)>=RowPad(ZcH,whz));
        nEy2=n1^2+(n2^2-n1^2)*(ColPad(zEy,ley)>=RowPad(ZcE,wey));
end
%-------------------------------------------------------------------------
% For geometries other than those described by scalar files
else
        eval(filename); % If not a scalar file input
end
%-------------------------------------------------------------------------
% Analysis for TE or TM cases

if upper(ANALYSIS_TYPES(1))=='B',          % check for running Both TE and TM cases
        Fcase=2;analysis_case=['E';'H'];
        else;Fcase=1;analysis_case=ANALYSIS_TYPES;end

for TEMcase=1:Fcase,                       % Begin loop for different TE and TM cases
        analysis_type=analysis_case(TEMcase,:);
        if upper(analysis_type)=='E',not_analysis_type='H';else;not_analysis_type='E';end;
```

```
        % Initialize grid points and establish boundary conditions.
        H_inc=zeros(Nmax+2,1);                  % Magnetic field
        E_inc=zeros(Nmax+3,1);                  % Electric field
        fprintf(['\t\t\t\t\t\t\t\t\t\t\t\t\t\t',analysis_type,' case\n']);
%--------------------------------------------------------------------------
% Boundary Condition placed on fields
% Incident field magnitude

if MagInt<=0,MagInt=0.5/N_0;end;       % Set to unit amplitude in air
        E_amp=sqrt(2*N_0*MagInt/n1);Uamp=E_amp;
if upper(analysis_type)=='H',E_amp= n1/N_0*E_amp;end;

%  field amplitude envelope
Cf=exp(-j*2*pi*n1/lam*zEy(front_pos));
E_inc_front=ones(1,rf-lf+1);
H_inc_front=ones(1,rf-lf+1);
H_inc_sides=zeros(Zaper,1);
%--------------------------------------------------------------------------
% Initialize fields
Ey=zeros(size(nEy2));           % Electric field
Hx=zeros(size(nHx2));           % Magnetic field
Hz=zeros(size(nHz2));           % Magnetic field

e1=zeros(Zaper,1);              % fields that handle BCs for incident wave in 1-D
h1=zeros(Zaper-1,1);
%**************************************************************************
iterprint=ceil(Nmax/DispVals);          % which energy values to display during loop
% Total-field/Scattered-field formulation  (See Taflove, Chapter 6, p.117)

if upper(symmetric(1))=='N', % consider symmetry

for n=1:Nmax,  % begin time-marching loop

                if upper(analysis_type)=='E',

                        H_inc=H_inc+(E_inc(2:Nmax+3)-E_inc(1:Nmax+2)).*c*dt/(N_0*dZ);

                        Hx=Hx+(         Ey((2:wey),:)-Ey((1:wey-1),:))   .*
        (c*dt)./(N_0*dZ);
                        Hz=Hz-(         Ey(:,(2:ley))-Ey(:,(1:ley-1)))      .*
        (c*dt)./(N_0*dX);
                        Hx(front_pos-1,(lf:rf))=Hx(front_pos-1,(lf:rf))-E_inc_front.*E_inc(3).*
        (c*dt)./(N_0*dZ);

                        Hz((1:Zaper),lf-1)=Hz((1:Zaper),lf-1)+e1.*      (c*dt)./(N_0*dX);
                        Hz((1:Zaper),rf)  =Hz((1:Zaper),rf)  -e1.*      (c*dt)./(N_0*dX);

                        h1=h1+(e1((2:Zaper),:)-e1((1:Zaper-1),:)).*(c*dt)./(N_0*dZ);% 1-D BC
                        h1(front_pos-1)=h1(front_pos-1)-E_inc(3).*(c*dt)./(N_0*dZ);
                        e1((2:Zaper-1))=e1((2:Zaper-1))+(1/dZ.*(h1((2:Zaper-1))-h1((1:Zaper-
                        2)))).*(N_0*c*dt)./n1^2;
```

```
                e1(front_pos)=e1(front_pos)-H_inc(2).*(N_0*c*dt)./(n1^2*dZ);% 1-D
                BC
                MUR_1d;e1(Zaper)=0; % 1-D BCs

                Ey((2:wey-1),(2:ley-1))=Ey((2:wey-1),(2:ley-1))+...
                                        (       1/dZ.*(Hx((2:whx),(2:lhx-1))-
                                                Hx((1:whx-1),(2:lhx-1)))-...
                                                1/dX.*(Hz((2:whz-1),(2:lhz))-
                                                Hz((2:whz-1),(1:lhz-1)))        ).*...
                                        (N_0*c*dt)./nEy2((2:wey-1),(2:ley-1));


                Ey(front_pos,(lf:rf))=Ey(front_pos,(lf:rf))-H_inc_front.*H_inc(2).*
                                        (N_0*c*dt)./(nEy2(front_pos,(lf:rf))*dZ);

Ey((1:Zaper),lf)=Ey((1:Zaper),lf)+H_inc_sides.*(N_0*c*dt)./(nEy2((1:Zaper),lf)*dX);
                Ey((1:Zaper),rf)=Ey((1:Zaper),rf)-
                H_inc_sides.*(N_0*c*dt)./(nEy2((1:Zaper),rf)*dX);
                Ey(Zaper,:)=Ey(Zaper,:).*rect(xEy/L);   % Impose finite aperture

                E_inc(1)=phasor_E(n)*E_amp*Cf;

                E_inc(2:Nmax)=E_inc(2:Nmax)+N_0*c*dt/(dZ*n1^2).*(H_inc(2:Nmax)
                        -H_inc(1:Nmax-1));

                eval(BCtype);   % Mur 2nd order BCs (See Taflove, Ch. 7, p. 158)
        else
                H_inc=H_inc-(E_inc(2:Nmax+3)-
                E_inc(1:Nmax+2)).*c*dt*N_0/(dZ*n1^2);
                Hx=Hx-(         Ey((2:wey),:)-Ey((1:wey-1),:))  .*
                        (c*dt*N_0)./(nHx2*dZ);
                Hz=Hz+(         Ey(:,(2:ley))-Ey(:,(1:ley-1)))      .*
                        (c*dt*N_0)./(nHz2*dX);

                Hx(front_pos-1,(lf:rf))=Hx(front_pos-
                1,(lf:rf))+E_inc_front.*E_inc(3).*(c*dt*N_0)./(nHx2(front_pos-
                                                        1,(lf:rf))*dZ);

                Hz((1:Zaper),lf-1)=Hz((1:Zaper),lf-1)-
                        e1.*(c*dt*N_0)./(nHz2((1:Zaper),lf-1)*dX);
                Hz((1:Zaper),rf)  =Hz((1:Zaper),rf)
                        +e1.*(c*dt*N_0)./(nHz2((1:Zaper),rf)  *dX);

                h1=h1-(e1(2:Zaper)-e1(1:Zaper-1)).*(c*dt*N_0)./(n1^2*dZ);% 1-D BC
                h1(front_pos-1)=h1(front_pos-1)+E_inc(3).*(c*dt*N_0)./(n1^2*dZ);
                                e1(2:Zaper-1)=e1(2:Zaper-1)-1/dZ.*(h1(2:Zaper-1)-
                                h1(1:Zaper-2)).*(c*dt/N_0);

e1(front_pos)=e1(front_pos)+H_inc(2).*(c*dt./(N_0*dZ));% 1-D BC
                                MUR_1d;e1(Zaper)=0;% 1-D BCs
```

```
                    Ey((2:wey-1),(2:ley-1))=Ey((2:wey-1),(2:ley-1))-...
                                        (       1/dZ.*(Hx((2:whx),(2:lhx-1))-
                                        Hx((1:whx-1),(2:lhx-1)))-...
                                        1/dX.*(Hz((2:whz-1),(2:lhz))-Hz((2:whz-
                                        1),(1:lhz-1)))  ).*...
                                                (c*dt/N_0);


                    Ey(front_pos,(lf:rf))=Ey(front_pos,(lf:rf))+H_inc_front.*H_inc(2).*
                                                (c*dt./(N_0*dZ));
                    Ey((1:Zaper),lf)=Ey((1:Zaper),lf)-H_inc_sides.*
                                                (c*dt)./(nEy2((1:Zaper),lf)*N_0*dX);
                    Ey((1:Zaper),rf)=Ey((1:Zaper),rf)+H_inc_sides.*
                                            (c*dt)./(nEy2((1:Zaper),rf)*N_0*dX);
                    Ey(Zaper,:)=Ey(Zaper,:).*rect(xEy/L);   % Impose finite aperture

                    E_inc(1)=phasor_E(n)*E_amp*Cf;
                    E_inc(2:Nmax)=E_inc(2:Nmax)-c*dt/(N_0*dZ).*(H_inc(2:Nmax)-
                            H_inc(1:Nmax-1));

                    eval(BCtype);   % Mur 2nd order BCs (See Taflove, Ch. 7, p. 158)
                end
%-------------------------------------------------------------------------
if upper(MovieFrames(1))=='Y',eval(MovieScript);end;  % save real(Ey) to Igor for image
ploting, etc.
%-------------------------------------------------------------------------
conditions03;    % script file that displays total energy at selected times.
%-------------------------------------------------------------------------

end                 % ending time-marching loop

else;eval(symmetric_script);end; % If geometry is symmetric
%*************************************************************************
%-------------------------------------------------------------------------
% Plane wave spectrum preliminaries
Z_prop=z_dist-zEy(mid_pt);
DFTsize=max([power(2,FFTpower);4*power(2,ceil(log2(2*ley-1)))]);% size of FFT
x1_fdtd=[(0:DFTsize/2-1),(-DFTsize/2:-1)].'*dX+min(abs(xEy));           % object plane positions
[microns]
fx=[(0:DFTsize/2-1),(-DFTsize/2:-1)].'/(DFTsize*dX);    % spatial frequencies
%-------------------------------------------------------------------------
% Obtain Ey at image plane
Ey_aper=shiftud([Ey(mid_pt,:).';zeros(DFTsize-ley,1)],-floor(ley/2),1); % in aperture
%-------------------------------------------------------------------------
% x positions in image plane
if lower(xform(1:5))=='angul',
        x0temp=fftshift(x1_fdtd); % In batch mode, positions are often redundant
else
        x0temp=fftshift(x1_fdtd/dX*lam/n2*Z_prop/(DFTsize*dX));
end;
if lower(x0units(1:5))=='milli',x0temp=x0temp/1000;end;
```

```
if lower(x0units(1:5))=='centi',x0temp=x0temp/10000;end;
xmin=min(find(x0temp>=x0_min));xmax=max(find(x0temp<=x0_max));

xpos=mod((xmin+DFTsize/2:xmax+DFTsize/2),DFTsize);
xpos=xpos+DFTsize*(xpos==0);          % Shifted FFT positions

xsmallest=x0temp(xmin);xlargest=x0temp(xmax);clear x0temp
%-------------------------------------------------------------------------
if upper(analysis_type)=='E',              % for TE case

        if lower(xform(1:5))=='angul',
                x0_fdtd=x1_fdtd;
                [ey,hx,hz]=MaxwellPropagation01('E',Ey_aper,Z_prop,x1_fdtd,lam,n2);
        else;
                [ey,x0_fdtd]=scalar_analysis1(Ey_aper,x1_fdtd,Z_prop,lam/n2,xform);
                [ey,hx,hz]=MaxwellPropagation01('E',ey,0,x0_fdtd,lam,n2);
        end;

if upper(DispXpos(1))=='N',
                ey=ey(xpos);
                hx=hx(xpos);
                hz=hz(xpos);
else;
                ey=fftshift(ey);
                hx=fftshift(hx);
                hz=fftshift(hz);
end;

        assignin('base',['eyFDTD',file_out],ey);
        assignin('base',['hxFDTD',file_out],hx);
        assignin('base',['hzFDTD',file_out],hz);
        assignin('base',['SEFDTD',file_out],0.5*(-ey.*conj(hx)));

        if (upper(ANALYSIS_TYPES(1))=='B')&(upper(polarstudy(1))=='Y'),
                assignin('base',['EyGRID',file_out],Ey);
        end;
end;

if upper(analysis_type)=='H',              % for TM case

        if lower(xform(1:5))=='angul',
                x0_fdtd=x1_fdtd;
                [hy,ex,ez]=MaxwellPropagation01('H',Ey_aper,Z_prop,x1_fdtd,lam,n2);
        else;
                [hy,x0_fdtd]=scalar_analysis1(Ey_aper,x1_fdtd,Z_prop,lam/n2,xform);
                [hy,ex,ez]=MaxwellPropagation01('H',hy,0,x0_fdtd,lam,n2);
        end;

if upper(DispXpos(1))=='N',
        hy=hy(xpos);
        ex=ex(xpos);
```

```
        ez=ez(xpos);
else;
        hy=fftshift(hy);
        ex=fftshift(ex);
        ez=fftshift(ez);
end;


        assignin('base',['hyFDTD',file_out],hy);
        assignin('base',['exFDTD',file_out],ex);
        assignin('base',['ezFDTD',file_out],ez);
        assignin('base',['SHFDTD',file_out],0.5*( ex.*conj(hy)));

        if (upper(ANALYSIS_TYPES(1))=='B')&(upper(polarstudy(1))=='Y'),
                assignin('base',['ExGRID',file_out],Hx);
        end;
end;
%-------------------------------------------------------------------------
% Angular spectrum and filtered field analysis of aperture
if upper(ASfiltering(1))=='Y',eval(ASfiltering_script);end;
%-------------------------------------------------------------------------
% Evaluation of the Poynting vector components in fdtd grid
if upper(poyntingcalc(1))=='Y',eval(poyntingcalc_script);end;
%-------------------------------------------------------------------------
% Construct image plot of field components as they propagate to detector plane
if upper(fieldimageplot(1))=='Y',

        if isempty(ZDPT),ZDPT=z_dist;end; % default distance to propagate

        [EyPLOTtemp,HxPLOTtemp,HzPLOTtemp]=feval(fieldimageplot_script,analysis_type,
Ey(mid_pt,:),xEy,...

        DFTsize,[xsmallest;xlargest;XSIP],[zEy(mid_pt);ZDPT+ZSIP;ZSIP],lam,n2);
                assignin('base',[analysis_type,'yimageplot',file_out],EyPLOTtemp); clear
EyPLOTtemp
                assignin('base',[not_analysis_type,'ximageplot',file_out],HxPLOTtemp); clear
HxPLOTtemp
                assignin('base',[not_analysis_type,'zimageplot',file_out],HzPLOTtemp); clear
HzPLOTtemp
                assignin('base',['ximageplot',file_out],x_imageplotting); clear x_imageplotting
                assignin('base',['zimageplot',file_out],z_imageplotting); clear z_imageplotting
end;

%-------------------------------------------------------------------------
if upper(fieldgridsave(1))=='Y',
        assignin('base',[analysis_type,'yGRID',file_out],Ey);
        assignin('base',[not_analysis_type,'xGRID',file_out],Hx);
        assignin('base',[not_analysis_type,'zGRID',file_out],Hz);
end;
%-------------------------------------------------------------------------
end     % Ending loop for different TE and TM cases
```

```
%-----------------------------------------------------------------------
% Polarization analysis
if (upper(ANALYSIS_TYPES(1))=='B')&(upper(polarstudy(1))=='Y'),
        eygridtemp=eval(['EyGRID',file_out]);
        exgridtemp=eval(['ExGRID',file_out]);

        [Diattenuation,Retardance]=feval(polarstudy_script,ex,ey,polthresh);
        assignin('base',['Diattenuation',file_out],Diattenuation);
        assignin('base',['Retardance',file_out],Retardance);
        [Dtemp,Rtemp]=feval(polarstudy_script,exgridtemp*exp(j*omega*dt/2),...
                                                0.5*(eygridtemp((1:wey-
1),:)+eygridtemp((2:wey),:)),polthresh);
        assignin('base',['Dgrid',file_out],Dtemp); clear Dtemp eygridtemp exgridtemp
        assignin('base',['Rgrid',file_out],Rtemp); clear Rtemp
end;
%-----------------------------------------------------------------------
% Units of x0 positions
if lower(x0units(1:5))=='milli',x0_fdtd=x0_fdtd/1000;end;
if lower(x0units(1:5))=='centi',x0_fdtd=x0_fdtd/10000;end;
%-----------------------------------------------------------------------
if upper(DispXpos(1))=='N',x0_fdtd=x0_fdtd(xpos);else;x0_fdtd=fftshift(x0_fdtd);end;
assignin('base',['XFDTD',file_out_lett],x0_fdtd);% x0 positions
%-----------------------------------------------------------------------
if upper(data_save(1))=='Y',      % save data?
                saving_files([x0_fdtd,abs(eval([lower(analysis_type),'y'']))],...
                        [upper(analysis_type),'y_tempFDTD',file_out,'.txt'],file_storage);
end
%-----------------------------------------------------------------------
        end     % Close loop for # of runs (if necessary)
end     % Closing batch mode
clear FILENAMES FILES_OUT
%-----------------------------------------------------------------------
end     % Closing super-batch mode
%-----------------------------------------------------------------------
End_time=cputime-Start_time;
fprintf(['\nrun time = ',num2str(floor(End_time/60)),' min. ',num2str(rem(End_time,60)),'
seconds\n' ])
fprintf(['-------------------------------------------------------------------------\n']);
%-----------------------------------------------------------------------
```

```
%*************************************************************************
******
% Mur boundary conditions to 2nd order  8/24/98
```

```
% This script file compute the BCs for the FDTD method given the appropriate field
% grid point values adjacted to the 4 walls of the surface.  Note that this program
% was originally intended to be used in conjuction with the file fdtd_2dim.m for
% the field with the variable name "Ey".
% Reference:  Mur 2nd order boundary conditions (See Taflove, Chapter 7, p. 158)
%*************************************************************************


if n==1,% Initialize BC's
e_f0_Na0=zeros(1,ley);e_f0_Nm1=zeros(1,ley);e_f1_Na0=zeros(1,ley);e_f1_Nm1=zeros(1,ley);
e_b0_Na0=zeros(1,ley);e_b0_Nm1=zeros(1,ley);e_b1_Na0=zeros(1,ley);e_b1_Nm1=zeros(1,ley)
;
e_l0_Na0=zeros(wey,1);e_l0_Nm1=zeros(wey,1);e_l1_Na0=zeros(wey,1);e_l1_Nm1=zeros(wey,
1);
e_r0_Na0=zeros(wey,1);e_r0_Nm1=zeros(wey,1);e_r1_Na0=zeros(wey,1);e_r1_Nm1=zeros(wey
,1);
end;

if upper(analysis_type)=='E',
        vel=c./sqrt(nEy2(2,:));             % velocity in the medium
else
        vel=c./sqrt(nHx2(1,:));
end
A1=(vel*dt-dZ)./(vel*dt+dZ);             % Constants for front and back faces
A2=(2*dZ)./(vel*dt+dZ);
A3=dZ*(vel*dt).^2./(2*dX.^2.*(vel*dt+dZ));
%---------------------------------------
e_f1_Np1=Ey(2,:);                % Front face
e_f0_Np1=-e_f1_Nm1  +        A1.*(e_f1_Np1+e_f0_Nm1)     +
        A2.*(e_f0_Na0+e_f1_Na0);
Ey(1,:) = e_f0_Np1;

Cor_2nd=A3(1:ley-2).*e_f0_Na0(1:ley-2)-2*A3(2:ley-1).*e_f0_Na0(2:ley-1)
+A3(3:ley).*e_f0_Na0(3:ley)+...
                A3(1:ley-2).*e_f1_Na0(1:ley-2)-2*A3(2:ley-1).*e_f1_Na0(2:ley-1)
+A3(3:ley).*e_f1_Na0(3:ley);
Ey(1,(2:ley-1))=Ey(1,(2:ley-1))+Cor_2nd;        % 2nd order correction

e_f0_Nm1 = e_f0_Na0;            % update data
e_f0_Na0 = e_f0_Np1;
e_f1_Nm1 = e_f1_Na0;
e_f1_Na0 = e_f1_Np1;
%---------------------------------------
if upper(analysis_type)=='E',
        vel=c./sqrt(nEy2(wey-1,:));                % velocity in the medium
else
        vel=c./sqrt(nHx2(whx,:));
end
A1=(vel*dt-dZ)./(vel*dt+dZ);             % Constants for front and back faces
A2=(2*dZ)./(vel*dt+dZ);
A3=dZ*(vel*dt).^2./(2*dX.^2.*(vel*dt+dZ));
```

```
%---------------------------------------
e_b1_Np1=Ey(wey-1,:);% Back face
e_b0_Np1=-e_b1_Nm1 +        A1.*(e_b1_Np1+e_b0_Nm1)    +
        A2.*(e_b0_Na0+e_b1_Na0);
Ey(wey,:) = e_b0_Np1;

Cor_2nd=A3(1:ley-2).*e_b0_Na0(1:ley-2)-2*A3(2:ley-1).*e_b0_Na0(2:ley-1)
+A3(3:ley).*e_b0_Na0(3:ley)+...
                A3(1:ley-2).*e_b1_Na0(1:ley-2)-2*A3(2:ley-1).*e_b1_Na0(2:ley-1)
+A3(3:ley).*e_b1_Na0(3:ley);
Ey(wey,(2:ley-1))=Ey(wey,(2:ley-1))+Cor_2nd;          % 2nd order correction

e_b0_Nm1 = e_b0_Na0;                % update data
e_b0_Na0 = e_b0_Np1;
e_b1_Nm1 = e_b1_Na0;
e_b1_Na0 = e_b1_Np1;
%---------------------------------------
if upper(analysis_type)=='E',
        vel=c./sqrt(nEy2(:,2));          % velocity in the medium
else
        vel=c./sqrt(nHz2(:,1));
end
B1=(vel*dt-dX)./(vel*dt+dX);          % Constants for left and right faces
B2=(2*dX)./(vel*dt+dX);
B3=dX*(vel*dt).^2./(2*dZ.^2.*(vel*dt+dX));
%---------------------------------------
e_l1_Np1=Ey(:,2);                % Left face
e_l0_Np1=-e_l1_Nm1   +        B1.*(e_l1_Np1+e_l0_Nm1)      +
        B2.*(e_l0_Na0+e_l1_Na0);
Ey(:,1) = e_l0_Np1;

Cor_2nd=B3(1:wey-2).*e_l0_Na0(1:wey-2)-2*B3(2:wey-1).*e_l0_Na0(2:wey-1)
+B3(3:wey).*e_l0_Na0(3:wey)+...
                B3(1:wey-2).*e_l1_Na0(1:wey-2)-2*B3(2:wey-1).*e_l1_Na0(2:wey-1)
+B3(3:wey).*e_l1_Na0(3:wey);
Ey((2:wey-1),1)=Ey((2:wey-1),1)+Cor_2nd;                    % 2nd order correction

e_l0_Nm1 = e_l0_Na0;          % update data
e_l0_Na0 = e_l0_Np1;
e_l1_Nm1 = e_l1_Na0;
e_l1_Na0 = e_l1_Np1;
%---------------------------------------
if upper(analysis_type)=='E',
        vel=c./sqrt(nEy2(:,ley-1));              % velocity in the medium
else
        vel=c./sqrt(nHz2(:,lhz));
end
B1=(vel*dt-dX)./(vel*dt+dX);          % Constants for left and right faces
B2=(2*dX)./(vel*dt+dX);
B3=dX*(vel*dt).^2./(2*dZ.^2.*(vel*dt+dX));
%---------------------------------------
```

```
e_r1_Np1=Ey(:,ley-1);  % Right face
e_r0_Np1=-e_r1_Nm1  +        B1.*(e_r1_Np1+e_r0_Nm1)      +
        B2.*(e_r0_Na0+e_r1_Na0);
Ey(:,ley) = e_r0_Np1;

Cor_2nd=B3(1:wey-2).*e_r0_Na0(1:wey-2)-2*B3(2:wey-1).*e_r0_Na0(2:wey-1)
+B3(3:wey).*e_r0_Na0(3:wey)+...
                B3(1:wey-2).*e_r1_Na0(1:wey-2)-2*B3(2:wey-1).*e_r1_Na0(2:wey-1)
+B3(3:wey).*e_r1_Na0(3:wey);
Ey((2:wey-1),ley)=Ey((2:wey-1),ley)+Cor_2nd;          % 2nd order correction

e_r0_Nm1 = e_r0_Na0;          % update data
e_r0_Na0 = e_r0_Np1;
e_r1_Nm1 = e_r1_Na0;
e_r1_Na0 = e_r1_Np1;
%-------------------------------------------------------------------------
clear A1 A2 A3 B1 B2 B3                % clear extra constants
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Relative permittivity spatial averging routing (RPSA) for FDTD code
% to handle a single DOE interface.
%
%
% created 10/7/98
%-------------------------------------------------------------------------
% Find relative permittivity at all spatial positions
if upper(perm_avg(2))=='N',      % Permittivity spatial averaging routine
        %********
        % nHx2 calculation
        Z2              = RowPad(ZcH,whz);    A=(2:whz);B=(1:whz-1);
        gridZ2          = ColPad(zHz,lhz);                    C=(2:lhz);D=(1:lhz-1);
        nHx2            = (gridZ2>=Z2);

        Z2=abs(gridZ2-Z2);      clear gridZ2
        nHx2=(nHx2(A,:).*Z2(A,:)+nHx2(B,:).*Z2(B,:))./(Z2(A,:)+Z2(B,:));      clear Z2
        nHx2=0.5*(nHx2(:,C)+nHx2(:,D));
        nHx2=[nHx2(:,1),nHx2,nHx2(:,lhz-1)];
        nHx2=n1^2+(n2^2-n1^2)*nHx2;                % permittivity at Hx positions
        %********
        % nHz2 calculation
        nEy2            = RowPad(ZcH,whx);                    A=(2:whx);B=(1:whx-1);
        gridZ2          = ColPad(zHx,lhz);                    C=(2:lhz);D=(1:lhz-1);
        nHz2            = (gridZ2>=nEy2);

        nEy2=abs(gridZ2-nEy2);                clear gridZ2
        nEy2=(nHz2(A,:).*nEy2(A,:)+nHz2(B,:).*nEy2(B,:))./(nEy2(A,:)+nEy2(B,:));
        nHz2=[nHz2(1,:);nEy2;nHz2(whx,:)];
        nHz2=n1^2+(n2^2-n1^2)*nHz2;                % permittivity at Hz positions
        %********
        % nEy2 calculation
        nEy2=0.5*(nEy2(:,C)+nEy2(:,D));                    clear A B C D
```

```
        nEy2=[nEy2(1,:);nEy2;nEy2(whx-1,:)];
        nEy2=[nEy2(:,1),nEy2,nEy2(:,lhz-1)];
        nEy2=n1^2+(n2^2-n1^2)*nEy2;                    % permittivity at Ey positions
        %********
else    % no permittivity averaging
        nHx2=n1^2+(n2^2-n1^2)*(ColPad(zHx,lhx)>=RowPad(ZcE,whx));
        nHz2=n1^2+(n2^2-n1^2)*(ColPad(zHz,lhz)>=RowPad(ZcH,whz));
        nEy2=n1^2+(n2^2-n1^2)*(ColPad(zEy,ley)>=RowPad(ZcE,wey));
end
%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% Symmetric FDTD time-marching routine for 2-dimensional DOEs that are
% symmmetric about the optical axis (z) and are illuminated by a normally
% incident plane-wave.
%
%       Note that       Ey(-x) =  Ey(x)
%                       Hx(-x) =  Hx(x)
%                       Hz(-x) = -Hz(x)
%
%       created 10/15/99
%-------------------------------------------------------------------------

% Save parameters calculated before symmetry was imposed
ley_old=ley;lhx_old=lhx;lhz_old=lhz;
rf_old=rf;
lf_old=lf;

Ey=Ey(:,(floor(ley/2):ley));
Hx=Hx(:,(floor(lhx/2):lhx));
Hz=Hz(:,(round(lhz/2):lhz));

nEy2=nEy2(:,(floor(ley/2):ley));
nHx2=nHx2(:,(floor(lhx/2):lhx));
nHz2=nHz2(:,(round(lhz/2):lhz));

xEytemp=xEy((floor(ley/2):ley));

ley=(ley+3)/2;   % resize FDTD grid
lhx=(lhx+3)/2;
lhz=lhz/2+1;

rf=(rf_old-(ley_old-ley));
```

```
lf=1;

E_inc_front_old=E_inc_front;
H_inc_front_old=H_inc_front;

E_inc_front=ones(1,rf);
H_inc_front=ones(1,rf);

%---------------------------------------------------------------------------
for n=1:Nmax,  % begin time-marching loop

                if upper(analysis_type)=='E',

                        H_inc=H_inc+(E_inc(2:Nmax+3)-E_inc(1:Nmax+2)).*c*dt/(N_0*dZ);

                        Hx=Hx+(         Ey((2:wey),:)-Ey((1:wey-1),:))   .*
                                (c*dt)./(N_0*dZ);
                        Hz=Hz-(         Ey(:,(2:ley))-Ey(:,(1:ley-1)))      .*
                                (c*dt)./(N_0*dX);
                        Hx(front_pos-1,(lf:rf))=Hx(front_pos-1,(lf:rf))-E_inc_front.*E_inc(3).*
                                (c*dt)./(N_0*dZ);
                        Hz((1:Zaper),rf)  =Hz((1:Zaper),rf)  -e1.*          (c*dt)./(N_0*dX);

                        Hx(:,1)= Hx(:,3);          % Impose symmetry
                        Hz(:,1)=-Hz(:,2);          % Impose anti-symmetry

                        h1=h1+(e1((2:Zaper),:)-e1((1:Zaper-1),:)).*(c*dt)./(N_0*dZ);% 1-D BC
                                        h1(front_pos-1)=h1(front_pos-1)-
                                                E_inc(3).*(c*dt)./(N_0*dZ);
                                        e1((2:Zaper-1))=e1((2:Zaper-
                        1))+(1/dZ.*(h1((2:Zaper-1))-h1((1:Zaper-2)))).*(N_0*c*dt)./n1^2;
                                        e1(front_pos)=e1(front_pos)-
                                        H_inc(2).*(N_0*c*dt)./(n1^2*dZ);% 1-D BC
                                        MUR_1d;e1(Zaper)=0; % 1-D BCs


                                Ey((2:wey-1),(2:ley-1))=Ey((2:wey-1),(2:ley-1))+...
                                        (       1/dZ.*(Hx((2:whx),(2:lhx-1))-
                                                Hx((1:whx-1),(2:lhx-1)))-...
                                                1/dX.*(Hz((2:whz-1),(2:lhz))-
                                                Hz((2:whz-1),(1:lhz-1)))        ).*...
                                                (N_0*c*dt)./nEy2((2:wey-1),(2:ley-1));


                        Ey(front_pos,(lf:rf))=Ey(front_pos,(lf:rf))-H_inc_front.*H_inc(2).*
                                (N_0*c*dt)./(nEy2(front_pos,(lf:rf))*dZ);

                        Ey((1:Zaper),rf)=Ey((1:Zaper),rf)-
                                H_inc_sides.*(N_0*c*dt)./(nEy2((1:Zaper),rf)*dX);
                        Ey(Zaper,:)=Ey(Zaper,:).*rect(xEytemp/L);% Impose finite aperture
```

```
                %Ey(find(nEy2~=real(nEy2)))=0;  % metallic structures

                E_inc(1)=phasor_E(n)*E_amp*Cf;

E_inc(2:Nmax)=E_inc(2:Nmax)+N_0*c*dt/(dZ*n1^2).*(H_inc(2:Nmax)-
        H_inc(1:Nmax-1));

                eval(BCtype);   % Mur 2nd order BCs (See Taflove, Ch. 7, p. 158)

                Ey(:,1)=Ey(:,3);% Impose symmetry
        else
                H_inc=H_inc-(E_inc(2:Nmax+3)-
                        E_inc(1:Nmax+2)).*c*dt*N_0/(dZ*n1^2);
                Hx=Hx-(         Ey((2:wey),:)-Ey((1:wey-1),:))   .*
                        (c*dt*N_0)./(nHx2*dZ);
                Hz=Hz+(         Ey(:,(2:ley))-Ey(:,(1:ley-1)))      .*
                        (c*dt*N_0)./(nHz2*dX);
                Hx(front_pos-1,(lf:rf))=Hx(front_pos-
        1,(lf:rf))+E_inc_front.*E_inc(3).*(c*dt*N_0)./(nHx2(front_pos-1,(lf:rf))*dZ);

                Hz((1:Zaper),rf)  =Hz((1:Zaper),rf)
                        +e1.*(c*dt*N_0)./(nHz2((1:Zaper),rf)  *dX);

                %Hx(find(nHx2~=real(nHx2)))=0;          % metallic structures
                %Hz(find(nHz2~=real(nHz2)))=0;          % metallic structures

                Hx(:,1)= Hx(:,3);          % Impose symmetry
                Hz(:,1)=-Hz(:,2);          % Impose anti-symmetry

                h1=h1-(e1(2:Zaper)-e1(1:Zaper-1)).*(c*dt*N_0)./(n1^2*dZ);% 1-D BC
                h1(front_pos-1)=h1(front_pos-1)+E_inc(3).*(c*dt*N_0)./(n1^2*dZ);
                                        e1(2:Zaper-1)=e1(2:Zaper-1)-
                                1/dZ.*(h1(2:Zaper-1)-h1(1:Zaper-2)).*(c*dt/N_0);

e1(front_pos)=e1(front_pos)+H_inc(2).*(c*dt./(N_0*dZ));% 1-D BC
                                        MUR_1d;e1(Zaper)=0;% 1-D BCs

                Ey((2:wey-1),(2:ley-1))=Ey((2:wey-1),(2:ley-1))-...
                        (1/dZ.*(Hx((2:whx),(2:lhx-1))-Hx((1:whx-1),(2:lhx-1)))-...
                        1/dX.*(Hz((2:whz-1),(2:lhz))-Hz((2:whz-1),(1:lhz-1)))    ).*...
                                                (c*dt/N_0);

                Ey(front_pos,(lf:rf))=Ey(front_pos,(lf:rf))+H_inc_front.*H_inc(2).*
                        (c*dt./(N_0*dZ));

                Ey((1:Zaper),rf)=Ey((1:Zaper),rf)+H_inc_sides.*
                        (c*dt)./(nEy2((1:Zaper),rf)*N_0*dX);
                Ey(Zaper,:)=Ey(Zaper,:).*rect(xEytemp/L);% Impose finite aperture

                E_inc(1)=phasor_E(n)*E_amp*Cf;
```

```
                E_inc(2:Nmax)=E_inc(2:Nmax)-c*dt/(N_0*dZ).*(H_inc(2:Nmax)-
                    H_inc(1:Nmax-1));

                eval(BCtype);   % Mur 2nd order BCs (See Taflove, Ch. 7, p. 158)

                Ey(:,1)=Ey(:,3);% Impose symmetry
            end
            %--------------------------------------------------------------------
            conditions03;    % script file that displays total energy at selected times.


if upper(MovieFrames(1))=='Y',eval(MovieScript);end;  % save real(Ey) to Igor for image
ploting, etc.


end             % ending time-marching loop
%*************************************************************************


% Restore to original sizes and values

Ey=[ Ey(:,(ley:-1:4)),Ey];
Hx=[ Hx(:,(lhx:-1:4)),Hx];
Hz=[-Hz(:,(lhz:-1:3)),Hz];

nEy2=[ nEy2(:,(ley:-1:4)),nEy2];
nHx2=[ nHx2(:,(lhx:-1:4)),nHx2];
nHz2=[ nHz2(:,(lhz:-1:3)),nHz2];

ley=ley_old;lhx=lhx_old;lhz=lhz_old;
rf=rf_old;
lf=lf_old;

E_inc_front=E_inc_front_old;
H_inc_front=H_inc_front_old;

clear E_inc_front_old H_inc_front_old xEy_old ley_old lhx_old lhz_old xEytemp
%--------------------------------------------------------------------




function [e1,e2,e3]=AS_derivatives(E,x,z,lam,dtype);
%*************************************************************************
%
% function U=AS_derivatives(E,x,z,lam,dtype);
%
% Scalar analysis of fields & derivatives after AS propagation
%
%       Inputs:
%
```

```
%               E       =       incident disturbance
%               x       =       object plane positions (same as image plane)
%               z       =       distance of propagation
%               lam     =       reduced wavelength in the propagating medium
%               dtype=  differentiation w.r.t. variable (note: 'y' or '0' just gives AS propagation)
%
%       Outputs:
%
%               e       =       derivatives or field after propagation
%
%       NOTE: All lengths are in units of MICRONS
%       NOTE:           exp(-jkz) notation      ->      replace fft by ifft and vice versa.
%
%       Created:        12/11/98
%*************************************************************************

% Use plane wave spectrum to obtain E or derivatives at image plane
E=flatten(E);
x=flatten(x);
dftsize=length(E);      % size of FFT
dx=abs(x(2)-x(1));
k0=2*pi/lam;

        fx=[(0:dftsize/2-1),(-dftsize/2:-1)]'/(dftsize*dx);% fx:spatial frequencies
        cutoff1=abs(fx)<=(1/lam);
        kernel=conj(exp(j*k0*z.*sqrt(1-(lam.*fx).^2))); % kernel->propagation kernel
        A0=ifft(E);

        xpos=find(lower(dtype)=='x');
        ypos=find(lower(dtype)=='y'|dtype=='0');
        zpos=find(lower(dtype)=='z');
        P=sort([xpos;ypos;zpos]);

        U=zeros(dftsize,length(dtype));

if not(isempty(xpos)),
        U(:,xpos)=fft(A0.*(-j*2*pi*fx).*kernel);                        % x-derivative
end

if not(isempty(zpos)),                                                  % z-derivative
        U(:,zpos)=fft(A0.*k0.*sqrt(1-(lam.*fx).^2).*(-j*cutoff1+j*not(cutoff1)).*kernel);
end

if not(isempty(ypos)),                                                  % field prop.
        if z~=0,U(:,ypos)=fft(A0.*kernel);else;U(:,ypos)=E;end;
end

e1=U(:,P(1));
if length(P)<2,e2=[];else;e2=U(:,P(2));end;
if length(P)<3,e3=[];else;e3=U(:,P(3));end;
%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% Angular spectrum data for FDTD apertures
%
%
%        Created: 10/8/99                    Last Revision 2/7/00
%-------------------------------------------------------------------------

% Angular spectrum of field immediately past DOE
k0=2*pi/lam;
fx=[(0:DFTsize/2-1),(-DFTsize/2:-1)].'/(DFTsize*dX);    % spatial frequencies
cutoff=abs(fx)<(n2/lam);

dfx=abs(fx(2)-fx(1));     % x position spacing
if fxspace<dfx,fxspace=dfx;end;fxspacepos=round(fxspace/dfx);
fxpos=fftshift(fx);
fxminpos=min(find(fxpos>=fxmin));fxmaxpos=max(find(fxpos<=fxmax));
fxpos=mod((fxminpos+DFTsize/2:fxspacepos:fxmaxpos+DFTsize/2),DFTsize).';
fxpos=fxpos+DFTsize*(fxpos==0);       % Shifted FFT positions

file_out_lett=file_out(find(isletter(file_out)));
assignin('base',['FXfdtd',file_out_lett],fx(fxpos));%fx positions

Ey_aper=shiftud([Ey(mid_pt,:).';zeros(DFTsize-ley,1)],-floor(ley/2),1);

A0temp=ifft(Ey_aper)*DFTsize*dX;
assignin('base',[analysis_type,'yASfdtdAMP',file_out],abs(A0temp(fxpos)));

A0ang=unwrap(angle(A0temp(fxpos)));
A0ang=A0ang-A0ang(ceil(length(fxpos)/2));
```

```
if ((upper(ANALYSIS_TYPES(1))=='B')&(upper(analysis_type)=='H')),
        A0ang=A0ang+(2*pi)*round((eval(['EyASfdtdPHA',file_out])-A0ang)/(2*pi));
        assignin('base',['HyASfdtdPHA',file_out],A0ang);
else
        assignin('base',[analysis_type,'yASfdtdPHA',file_out],A0ang);
end;

if upper(analysis_type)=='H',                % Ex angular spectrum only
        extemp=j*c*N_0/(omega*n2^2)*A0temp.*k0.*n2.*sqrt(1-(lam./n2.*fx).^2).*(-
j*cutoff+j*not(cutoff));
        assignin('base',['ExASfdtdAMP',file_out],abs(extemp(fxpos)));
        ExA0ang=unwrap(angle(extemp));
        ExA0ang=ExA0ang-ExA0ang(1);        ExA0ang=ExA0ang(fxpos);
        ExA0ang=ExA0ang+(2*pi)*round((A0ang-ExA0ang)/(2*pi));
        assignin('base',['ExASfdtdPHA',file_out],ExA0ang);
end
%-------------------------------------------------------------------------
% Field immediately past DOE
assignin('base',[analysis_type,'yaperAMP',file_out],abs(Ey(mid_pt,:).'));
Eangtemp=angle(Ey(mid_pt,:).');
%PhaseDOE=(2*pi*dn/lam)*flatten(ZcE);
%Eangtemp=Eangtemp+(PhaseDOE(ceil(ley/2))-Eangtemp(ceil(ley/2)));
%Eangtemp=Eangtemp+(2*pi)*round((PhaseDOE-Eangtemp)/(2*pi));
assignin('base',[analysis_type,'yaperPHA',file_out],Eangtemp);

if upper(analysis_type)=='H',                % find Ex field at aperture
        extemp=fft(extemp)/(DFTsize*dX);
        extemp=shiftud(extemp,floor(ley/2),1);
        extemp=extemp(1:ley);
        assignin('base',['ExaperAMP',file_out],abs(extemp(:)));
        extemp=angle(extemp);
        %extemp=extemp+(PhaseDOE(ceil(ley/2))-extemp(ceil(ley/2)));
        %extemp=extemp+(2*pi)*round((PhaseDOE-extemp)/(2*pi));
        assignin('base',['ExaperPHA',file_out],extemp);
end
%-------------------------------------------------------------------------
% Filtered field immediately past DOE
Etemp=fft(A0temp.*cutoff)/(DFTsize*dX);
Etemp=shiftud(Etemp,floor(ley/2),1);
Etemp=Etemp(1:ley);
assignin('base',[analysis_type,'yfilteraperAMP',file_out],abs(Etemp));

Eangtemp=angle(Etemp);
%Eangtemp=Eangtemp+(PhaseDOE(ceil(ley/2))-Eangtemp(ceil(ley/2)));
%Eangtemp=Eangtemp+(2*pi)*round((PhaseDOE-Eangtemp)/(2*pi));
assignin('base',[analysis_type,'yfilteraperPHA',file_out],Eangtemp);

if upper(analysis_type)=='H',                % find filtered Ex field at aperture
        extemp=j*c*N_0/(omega*n2^2)*A0temp.*k0.*n2.*sqrt(1-(lam./n2.*fx).^2).*(-j*cutoff);
        extemp=fft(extemp)/(DFTsize*dX);
        extemp=shiftud(extemp,floor(ley/2),1);
```

```
        extemp=extemp(1:ley);
        assignin('base',['ExfilteraperAMP',file_out],abs(extemp(:)));
        extemp=angle(extemp);
        %extemp=extemp+(PhaseDOE(ceil(ley/2))-extemp(ceil(ley/2)));
        %extemp=extemp+(2*pi)*round((PhaseDOE-extemp)/(2*pi));
        assignin('base',['ExfilteraperPHA',file_out],extemp);
end
%-------------------------------------------------------------------------
% Find detector plane field magnitude for "filtered" field
% Also averaging the field amplitude past the DOE over the aperture.
Ey_filt=mean(abs(Etemp(find(rect(xEy.'/L)))))*rect(xEy.'/L).*exp(j*angle(Etemp));
Ey_filt=shiftud([Ey_filt;zeros(DFTsize-ley,1)],-floor(ley/2),1);
if upper(analysis_type)=='E',
        Ey_filt=scalar_analysis1(Ey_filt,x1_fdtd,Z_prop,lam/n2,xform);
end

if upper(analysis_type)=='H',
        [Ey_filt,Ex_filt]=MaxwellPropagation01('H',Ey_filt,Z_prop,x1_fdtd,lam,n2);
        assignin('base',['Exfdtdfiltdetectorplane',file_out],Ex_filt(xpos));
end;
assignin('base',[analysis_type,'yfdtdfiltdetectorplane',file_out],Ey_filt(xpos));

clear Etemp Ey_filt Eangtemp A0temp A0ang ExA0ang extemp Ex_filt fxpos
%-------------------------------------------------------------------------

function
[EYPLOT,HXPLOT,HZPLOT]=FDTD_imageplotting01(analysis_type,E,x,DFTsize,xparam,zpa
ram,lam,n2);
%-------------------------------------------------------------------
% FDTD image plotting for field components propagated to a detector plane
%
% function
[EYPLOT,HXPLOT,HZPLOT]=FDTD_imageplotting01(analysis_type,E,x,DFTsize,...
%
        xmin,xmax,xspace,zmin,zmax,zspace,lam,n2);
%
%       Inputs:
%
%               analysis_type= 'E' or 'H'  (for TE or TM respectively)
%               E       =       incident disturbance
%               x       =       object plane positions (same as image plane)
%               zmax = distance of propagation
%               lam     =       reduced wavelength in the propagating medium
%               dtype=  differentiation w.r.t. variable (note: 'y' just gives AS propagation)
%
%       Outputs:
%
%               e       =       derivatives or field after propagation
%
%       NOTE: All lengths are in units of MICRONS
%       NOTE:           exp(-jkz) notation      ->      replace fft by ifft and vice versa.
```

```
%       created:  10/14/99
%--------------------------------------------------------------------

% Physical constants
mu=4*pi*10^(-7);                    % permeability of free space [Henrys/m]
epsilon0=8.8541*10^(-12);           % permittivity of free space [Farads/m]
c=10^(-9)/sqrt(mu*epsilon0);% speed of light in units of [microns/femtosecond]
omega=2*pi*c/lam;                   % angular frequency of light [radians/femtosecond]
period=lam/c;                       % period of incident wave [femtoseconds]
N_0=sqrt(mu/epsilon0);
%--------------------------------------------------------------------

E=flatten(E); % E field at incident plane
x=flatten(x); % x positions

xmin=xparam(1); % min x value
xmax=xparam(2); % max x value
if length(xparam)<3,xspace=0;else;xspace=xparam(3);end;%spacing

zmin=zparam(1); % min z value (corresponds to E incident z position)
zmax=zparam(2); % max z value ( to which plne to propagate)
if length(zparam)<3,zspace=zmax-zmin;else;zspace=zparam(3);end;%spacing
if zspace==0,zspace=zmax-zmin;end;

if upper(analysis_type)=='E',

        if nargout==1,
        EYPLOT=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspace,zmin,zmax,zspace,la
m/n2,'y');
        HXPLOT=[];
        HZPLOT= [];
        end

        if nargout==2,
        [EYPLOT,HXPLOT]=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspace,zmin,zm
ax,zspace,lam/n2,'yz');
        HXPLOT=-j*c/(omega*N_0)*HXPLOT;
        HZPLOT= [];
        end
        if nargout==3,
        [EYPLOT,HXPLOT,HZPLOT]=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspac
e,zmin,zmax,zspace,lam/n2,'yzx');
        HXPLOT=-j*c/(omega*N_0)*HXPLOT;
        HZPLOT= j*c/(omega*N_0)*HZPLOT;
        end
end
if upper(analysis_type)=='H',

        if nargout==1,
        EYPLOT=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspace,zmin,zmax,zspace,la
m/n2,'y');
```

```
        HXPLOT=[];
        HZPLOT= [];
    end

    if nargout==2,
    [EYPLOT,HXPLOT]=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspace,zmin,zm
ax,zspace,lam/n2,'yz');
        HXPLOT= j*c*N_0/(omega*n2^2)*HXPLOT;
        HZPLOT=[];
    end

    if nargout==3,

    [EYPLOT,HXPLOT,HZPLOT]=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspac
e,zmin,zmax,zspace,lam/n2,'yzx');
        HXPLOT= j*c*N_0/(omega*n2^2)*HXPLOT;
        HZPLOT=-j*c*N_0/(omega*n2^2)*HZPLOT;
    end
end




function[E1,E2,E3]=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspace,zmin,zmax,zspace,
lam,dtype);
%---------------------------------------------------------------------
% FDTD image plotting of field component and its spatial derivates as
% they propagate to a detector plane.
%
% function [E1,E2,E3]=FDTD_imageplotting02(E,x,DFTsize,xmin,xmax,xspace,...
%
    zmin,zmax,zspace,lam,dtype);
%
%   Inputs:
%
%       E   =   incident disturbance
%       x   =   object plane positions (same as image plane)
%       zmax = distance of propagation
%       lam =   reduced wavelength in the propagating medium
%       dtype=  differentiation w.r.t. variable (note: 'y' just gives AS propagation)
%
%   Outputs:
%
%       e   =   derivatives or field after propagation
%
%   NOTE: All lengths are in units of MICRONS
%   NOTE:       exp(-jkz) notation      ->      replace fft by ifft and vice versa.
%
%   created:  10/11/99
```

```
%-----------------------------------------------------------------------

dx=abs(x(2)-x(1));        % x position spacing
if xspace<dx,xspace=dx;end;xspace=round(xspace/dx);

x=[(0:DFTsize/2-1),(-DFTsize/2:-1)].'*dx+min(abs(x));% object plane positions [microns]
xtemp=fftshift(x);
xmin=min(find(xtemp>=xmin));xmax=max(find(xtemp<=xmax));
xtemp=xtemp(xmin:xspace:xmax);

FindPos=mod((xmin+DFTsize/2:xspace:xmax+DFTsize/2),DFTsize);
FindPos=FindPos+DFTsize*(FindPos==0);        % Shifted FFT positions

K=2*pi/lam;
        % reduced wavenumber
fx=[(0:DFTsize/2-1),(-DFTsize/2:-1)].'/(DFTsize*dx);% spatial frequencies
cutoff=abs(fx)<=(1/lam);                                                    % cutoff
frequency

kernelY=ones(DFTsize,1);        kernel0=kernelY;
kernelZ=K.*sqrt(1-(lam.*fx).^2).*(-j*cutoff+j*not(cutoff));
kernelX=(-j*2*pi*fx);

        xpos=find(lower(dtype)=='x');    % which components are propagated
        ypos=find(lower(dtype)=='y'|dtype=='0');
        zpos=find(lower(dtype)=='z');
        P=sort([xpos;ypos;zpos]);

kernel=ColPad(conj(exp(j*K*zspace.*sqrt(1-(lam.*fx).^2))),length(P));% propagate in n2

K1=[RowPad(['kernel'],length(P)),upper(dtype(P)).'];
K1=['[',flatten(char(K1.',ColPad(',',length(P)-1))).',']'];
K1=eval(K1); clear kernelX kernelY kernelZ kernel0

Etemp=flatten(E);
ley=length(Etemp);
Etemp=shiftud([Etemp;zeros(DFTsize-ley,1)],-floor(ley/2),1); % in aperture
A0temp=ColPad(ifft(Etemp),length(P)); % 1st AS calculation

[e1,e2,e3]=AS_derivatives(Etemp,x,0,lam,dtype);
Etemp=[e1,e2,e3];clear e1 e2 e3
Earray=Etemp;Earray=ShiftFFTcolumns(Earray);Earray=Earray((xmin:xspace:xmax),:);

ztemp=(zmin:zspace:zmax).';
ntemp=length(ztemp);

for htemp=1:ntemp-1,
                A0temp=A0temp.*kernel;                % propagate angular spectrum by zspace

                Etemp=fft(A0temp.*K1);
                Etemp=Etemp(FindPos,:);
```

```
                Earray=[Earray,Etemp];                  % final array
end;

E1=Earray(:,(1:length(P):size(Earray,2)));
if nargout<2,E2=[];else;E2=Earray(:,(2:length(P):size(Earray,2)));end;
if nargout<3,E3=[];else;E3=Earray(:,(3:length(P):size(Earray,2)));end;

        assignin('base','x_imageplotting',xtemp);% x positions
        assignin('base','z_imageplotting',ztemp); % z positions


% Polarimerty of time-harmonic steady-state fields in detector plane

if upper(analysis_type)=='E',
        [ey,hx,hz]=AS_derivatives(Ey_aper,x1_fdtd,Z_prop,lam/n2,'0zx');
        hx=-j*c/(omega*N_0)*hx;
        hz= j*c/(omega*N_0)*hz;

        stez=0.5*real(-ey.*conj(hx));
        %stex=0.5*real( ey.*conj(hz));
end



if upper(analysis_type)=='H',
        [hy,ex,ez]=AS_derivatives(Ey_aper*n1/N_0,x1_fdtd,Z_prop,lam/n2,'0zx');
        ex= j*c*N_0/(omega*n2^2)*ex;
        ez=-j*c*N_0/(omega*n2^2)*ez;

        stmz=0.5*real( ex.*conj(hy));
        %stmx=0.5*real(-ez.*conj(hy));
end



function [Diattenuation,Retardance]=Polarimetry02(ex,ey,threshold);
%-----------------------------------------------------------------
%
% function [Diattenuation,Retardance]=Polarimetry02(ex,ey,threshold);
%
% Polarimerty of fields in detector plane. Created 9/29/99
%-----------------------------------------------------------------
if nargin<3,threshold=0;end;      % set default value to zero
den=abs(ex).^2+abs(ey).^2;den=den+eps*(den==0);
threshold=threshold*max(den(:));

Diattenuation=((abs(ex).^2-abs(ey).^2)./den).*(den>threshold);

Retardance=mod(angle(ex)-angle(ey),2*pi);
Retardance=(Retardance-2*pi*(Retardance>pi)).*(den>threshold);
%-----------------------------------------------------------------
```

```
% Calculation of the Poynting vector components in the FDTD grid
%
%
% created 10/18/99
%-------------------------------------------------------------------------

xspace=0*dX;   % spacing could be changed later
if xspace<dX,xspace=dX;end;xspace=round(xspace/dX);

zspace=0*dZ;
if zspace<dZ,zspace=dZ;end;zspace=round(zspace/dZ);

if upper(analysis_type)=='E',      % for TE case

        hxtemp=0.5*(Hx((2:whx),(2:lhx-1))+Hx((1:whx-1),(2:lhx-1)))*exp(j*omega*dt/2);
        Poyntingz=-0.5*( Ey((2:wey-1),(2:ley-1)).*conj(hxtemp));  clear hxtemp  % x comp

        hztemp=0.5*(Hz((2:whz-1),(2:lhz))+Hz((2:whz-1),(1:lhz-1)))*exp(j*omega*dt/2);
        Poyntingx= 0.5*( Ey((2:wey-1),(2:ley-1)).*conj(hztemp));  clear hztemp  % z comp

        %stex=0.5*real( ey.*conj(hz));
        %stez=0.5*real(-ey.*conj(hx));
end;

if upper(analysis_type)=='H',      % for TM case

        hxtemp=0.5*(Hx((2:whx),(2:lhx-1))+Hx((1:whx-1),(2:lhx-1)))*exp(j*omega*dt/2);
        Poyntingz= 0.5*( conj(Ey((2:wey-1),(2:ley-1))).*hxtemp);  clear hxtemp  % x comp

        hztemp=0.5*(Hz((2:whz-1),(2:lhz))+Hz((2:whz-1),(1:lhz-1)))*exp(j*omega*dt/2);
        Poyntingx=-0.5*( conj(Ey((2:wey-1),(2:ley-1))).*hztemp);  clear hztemp  % z comp

        %stmx=0.5*real(-ez.*conj(hy));
        %stmz=0.5*real( ex.*conj(hy));
end;

assignin('base',['Poyntingx',analysis_type,file_out],Poyntingx((1:zspace:wey-2),(1:xspace:ley-
2))); clear Poyntingx
assignin('base',['Poyntingz',analysis_type,file_out],Poyntingz((1:zspace:wey-2),(1:xspace:ley-
2))); clear Poyntingz

assignin('base',['xPoynting',analysis_type,file_out],xEy(2:xspace:ley-1).');
assignin('base',['zPoynting',analysis_type,file_out],zEy(2:zspace:wey-1));
```

```
% This program is used to save E (or H) fields over the FDTD lattice at certain moments of time
% to demonstrate the steady-state time-harmonic nature of the fields.
                if n==1,
                        MIN1=[];MAX1=[];frame_num=0;
                end
                snapshot=ceil(samp_t/20);
                time0=samp_t*(floor(Nmax/samp_t)-1);time1=samp_t*floor(Nmax/samp_t);
                if n>=time0&(floor((n-time0)/snapshot)==(n-
                        time0)/snapshot|(n==time0|n==time1)),

                        disp(num2str(n));

                        if frame_num==0,
                                N_steps=4;              % Normally this is 2.
                                N_wind=90;
                                Pr=find(rect(xEy(:)/N_wind));Pl=min(Pr);Pr=max(Pr);
                                Ntemp=(Pl:Pr).';
                                Ltemp1=Ntemp(ceil((Pr+Pl)/2):N_steps:Pr);
                                Ltemp2=Ntemp(ceil((Pr+Pl)/2):-N_steps:Pl);Ltemp2(1)=[];
                                Ntemp=[flipud(Ltemp2);Ltemp1];      clear Ltemp1 Ltemp2
                        xdat=flatten(xEy(Ntemp));       dxdat=abs(xdat(2)-xdat(1));
                        xdat=[xdat(1)-dxdat;xdat(:)]+dxdat/2;
                        zdat=zEy(front_pos:wey-1);zdat=[zdat(1)-dZ;zdat]+dZ/2;

                        time_steps=((time0:snapshot:time1).'-time0)/samp_t;

        saving_files(time_steps,'time_steps.txt','BigSpace240:Users:Mellin:Temporary_datafiles:'
                );
        saving_files(xdat,'xdat.txt','BigSpace240:Users:Mellin:Temporary_datafiles:');

        saving_files(zdat,'zdat.txt','BigSpace240:Users:Mellin:Temporary_datafiles:');
                        end
                        if upper(symmetric)=='Y',
                                Eplot=[ Ey(:,(size(Ey,2):-1:4)),Ey];
                        else;
                                Eplot=Ey;
                        end;

                        Eplot=real(Eplot((front_pos:wey-1),Ntemp));
                                                MAX1=[MAX1;max(Eplot(:))];
                                                MIN1=[MIN1;min(Eplot(:))];
                        saving_files(Eplot.',[analysis_type,'Frame',num2str(frame_num),'.txt'],...

        'BigSpace240:Users:Mellin:Temporary_datafiles:');
                        frame_num=frame_num+1;
                clear Eplot
                end
%-------------------------------------------------------------------------------
```

```
% Script file to make movies during FDTD runs
% This is used to save the real part of E (or H) field components over the FDTD
% lattice at certain moments of time to demonstrate the steady-state time-harmonic nature
% of the fields.
%
%       Created 10/18/99
%-------------------------------------------------------------------------------

%xparam=[-10;10;xspacing];

xparam=[xUmin;xUmax;xspacing];                  % parameters for x positions
zparam=[zEy(front_pos);max(zEy);zspacing];      % parameters for z positions

MovieFieldsTemp=MovieFields;

        temp1=find(MovieFieldsTemp=='y');
        temp2=find((MovieFieldsTemp=='x')|(MovieFieldsTemp=='z'));

        MovieFieldsTemp(temp1)=RowPad('E',length(temp1));
        MovieFieldsTemp(temp2)=RowPad('H',length(temp2));
        MovieFieldsInput=[MovieFieldsTemp(:),MovieFields(:)]; % which compents to save

if upper(analysis_type(1))=='H',
        MovieFieldsTemp(temp1)=RowPad('H',length(temp1));
        MovieFieldsTemp(temp2)=RowPad('E',length(temp2));
end; clear temp1 temp2

        MovieFieldsHeader=[MovieFieldsTemp(:),MovieFields(:)]; % assign prefix

for ntemp=1:length(MovieFields),

        xdat=flatten(eval(['x',MovieFieldsInput(ntemp,:)]));
        zdat=flatten(eval(['z',MovieFieldsInput(ntemp,:)]));
        Edat=eval(MovieFieldsInput(ntemp,:));

        Ey_save_to_Igor02(Edat,xdat,zdat,xparam,zparam,n,Nmax,samp_t,Tincrement,...

        file_storage,MovieFieldsHeader(ntemp,:),file_out);

end; clear ntemp MovieFieldsTemp
```

```
function
Ey_save_to_Igor02(Ey,xdat,zdat,xparam,zparam,n,Nmax,samp_t,Tincrements,file_storage,prefix
,suffix);
%-------------------------------------------------------------------------------
% This function is used to save the real part of E (or H) field components over the FDTD
% lattice at certain moments of time to demonstrate the steady-state time-harmonic nature
% of the fields. This function is usually called by Ey_save_to_Igor01.m script file.
%
%
Ey_save_to_Igor02(Ey,xdat,zdat,xparam,zparam,n,Nmax,samp_t,Tincrements,file_storage,prefix
,suffix);
%
% Last revision: 10/19/99
%-------------------------------------------------------------------------------
                snapshot=ceil(samp_t/Tincrements);      % which frame to capture over a period
                time0=Nmax-samp_t;                                      % min time
                time1=Nmax;                                                     % max time
                if n>=time0&(floor((n-time0)/snapshot)==(n-
time0)/snapshot|(n==time0|n==time1)), % capture frame?

                if length(xdat)==size(Ey,2),symmetric='N';else;symmetric='Y';end; % for
symmetric profiles
                        offset1=(min(abs(xdat))==0);    % If so, compensate for offset


                                xmin=xparam(1);
                                xmax=xparam(2);
                                if length(xparam)<3,xspace=0;else;xspace=xparam(3);end;

                                zmin=zparam(1);
                                zmax=zparam(2);
                                if length(zparam)<3,zspace=0;else;zspace=zparam(3);end;


                                dzdat=abs(zdat(2)-zdat(1));
                                if zspace<dzdat,zspace=dzdat;end;zspace=round(zspace/dzdat);

        zmin=min(find(zdat>=zmin));zmax=max(find(zdat<=zmax));
                                        ztemp=(zmin:zspace:zmax);
                                        zdat=zdat(ztemp);

                                dxdat=abs(xdat(2)-xdat(1));
                                if xspace<dxdat,xspace=dxdat;end;xspace=round(xspace/dxdat);

        xmin=min(find(xdat>=xmin));xmax=max(find(xdat<=xmax));
                                        xtemp=(xmin:xspace:xmax);
                                        xdat=xdat(xtemp);

                                xdat=[xdat(1)-dxdat;xdat]+dxdat/2;      % For Igor imageplot
format
                                zdat=[zdat(1)-dzdat;zdat]+dzdat/2;
```

```
                time_steps=((time0:snapshot:time1).'-time0)/samp_t; % which
time increment

            if n==time0,      % save position data and time increment for later Igor
use
                saving_files(time_steps,['time_steps',suffix,'.txt'],file_storage);
                saving_files(xdat,[prefix,'_xdat',suffix,'.txt'],file_storage);
                saving_files(zdat,[prefix,'_zdat',suffix,'.txt'],file_storage);
            end

            if upper(symmetric(1))=='Y', % adjust to form entire grid if true
                Eplot=[ sign(offset1-.5)*Ey(:,(size(Ey,2):-1:3+offset1)),Ey];
            else;
                Eplot=Ey;
            end;

            Eplot=real(Eplot(ztemp,xtemp));          % Saving real part of field
component

            frame_num=round(abs(((n-time0)/snapshot))); % assign a frame # to
saved data

        saving_files(Eplot.',[prefix,'Frame',num2str(frame_num),suffix,'.txt'],file_storage);
            clear Eplot
        end
```

APPENDIX F

IASA design codes

This appendix contains the computer programs used for the IASA design routines. Note that the codes are written in Matlab®.

```
% This script file is used for scalar designs of
% 1-N beamsplitters The objective is to optimize DOE profile using
% an iterative Angular Spectrum approach. (Created  8/20/99)
%
%
% Latest revision: 5/3/00
%-------------------------------------------------------------------------------
Start_time=cputime;      % Time program run
% Open file containing the following input parameters :
%-------------------------------------------------------------------------
%       lam             = incident wavelength [µm]
%       n1              = index of refraction of medium 1 (air)
%       n2              = index of refraction of DOE (silicon)
%       P               = Number of partitions of DOE profile (if there is re-sampling)
%       Q               = Number of quantized DOE levels (0 means no quantization)
%       xUmax = largest  position of field incident on DOE [µm]
%       xUmin  = smallest position of field incident on DOE [µm]
%       DFTsize= power of FFT in near to far field transformation (2^N)
%       z_dist   = distance to the image plane [µm]
%       filenumber              : number of scalar files to evaluate
%       filename                : file containing the scalar data.
%-------------------------------------------------------------------------
parameters='IASAscript01.txt';          % file containing the inputs
fid = fopen(parameters,'r');
for k=1:5,fgets(fid);end;
for k=1:10,temp = fscanf(fid,'%s',1);a(k)=fscanf(fid,'%g\n',1);end;
fgets(fid);

temp=fscanf(fid,'%s',1);
PreProcess =fscanf(fid,'%s',1);
PreProcess_script =fscanf(fid,'%s\n',1);  % script file for Pre-processing

temp=fscanf(fid,'%s',1);
PostProcess =fscanf(fid,'%s',1);
PostProcess_script =fscanf(fid,'%s\n',1);% script file for Post-processing

temp=fscanf(fid,'%s',1);
Optimize =fscanf(fid,'%s',1);     % script file for Optimization
out_file = fscanf(fid,'%s\n',1);

temp=fscanf(fid,'%s',1);
data_save=fscanf(fid,'%s',1);           % Save computed data? (Y/N)
saved_file=fscanf(fid,'%s',1);    % If saved, name of file and directory
directory=fscanf(fid,'%s\n',1);

fclose(fid);
lam                             = a(1);                 %
n1                              = a(2);                 %
n2                              = a(3);                 %
P                               = a(4);                 %       Re-sampling
Q                               = a(5);                 %       Quantization
```

```
num                         = a(6);
L                           = a(7);             % size of finite aperture
FFTpower                    = a(8);             %
zdist                       = a(9);             %
iterations                  = a(10);

clear a fid temp
%-------------------------------------------------------------------------
k0=2*pi/lam;                        % wavenumber in free space.
dn=n2-n1;                           % refractive index difference
minsize=L/num;                      % min. feature size OR # of samples taken
DFTsize=power(2,FFTpower);  % size of FFT
dx=L/num;                           % spacing increment in object plane
x1=(0.5*not(mod(num,2))+ [(0:DFTsize/2-1) (-DFTsize/2:-1)]')*dx;% object positions
if not(exist('T0')),T0=zeros(DFTsize,1);end; % Starting etch profile
Einc=ones(size(x1));                % Incident electric field
%-------------------------------------------------------------------------------
% Pre-processing of DOE
if upper(PreProcess(1))=='Y',eval(PreProcess_script);end;
%-------------------------------------------------------------------------
% Begin iterative optimization routine
eval(Optimize); % perform DOE optimization
%-------------------------------------------------------------------------
% Post-processing of DOE
if upper(PostProcess(1))=='Y',eval(PostProcess_script);end;
%-------------------------------------------------------------------------------
% Obtain DOE profile
        % etch = Thickness profile in aperture
        % xdoe = Positions along DOE
[etch,xdoe]=pick(T0,x1,-L/2,L/2);
etch=etch-min(etch);                % This is what gets saved in data files
if n2<n1,etch=etch-max(etch);end;   % if etch is INTO material
mfs=abs(xdoe(2)-xdoe(1));
%-------------------------------------------------------------------------
% For plotting purposes of DOE profile
assignin('base',['xdoe_',out_file],flatten([xdoe-mfs/2,xdoe+mfs/2].'));
assignin('base',['tdoe_',out_file],flatten([etch,etch].'));
%-------------------------------------------------------------------------------
% if saving data
if upper(data_save(1))=='Y',saving_files([xdoe,etch],saved_file,directory);end;
%-------------------------------------------------------------------------------
End_time=cputime-Start_time;
fprintf(['\nrun time = ',num2str(floor(End_time/60)),' min. ',num2str(rem(End_time,60)),'
seconds\n' ])
fprintf(['-------------------------------------------------------------------------------\n']);
%-------------------------------------------------------------------------
```

```
% Optimization of DOE for 1-N beamsplitter

% Design parameters
Npeaks=1;        % Number of desired peaks
Peak_sep=25;    % Separation between peaks
pertmin=0.1;
pertmax=0.1;
peak=8.0*dx;    % EFFECTIVE width of the photodetectors in the focal plane array
xform='angularspzectrum';
dfx=1/(DFTsize*dx);% spatial frequency spacing

% Monitoring preliminaries
fprintf(['     iteration' '   diff.eff.' '      S/N'  '                        left lobe' '    right lobe\n'])
photowid=15;                        % width of the photodetectors in the focal plane array
iterprint=ceil(iterations/10);format short g;
          wt1 =rect((abs(x1)-Peak_sep/2)/photowid);        % SELECT CENTRAL DETECTORS
          wt2 =rect((rem(abs(x1),Peak_sep)-Peak_sep/2)/photowid)-wt1;% OUTSIDE
                                                                                    DETECTORS
%-------------------------------------------------------------------------

for n0=1:iterations,
                              pert=pertmin+rand*(pertmax-pertmin); % perturbative change
%-------------------------------------------------------------------------
% Increase E field amplitudes within given windows
e2=rect(x1/L).*exp(j*2*pi.*T0.*(dn./lam));
[e2,x0,A0,fx]=scalar_analysis1(e2,x1,zdist-max(T0),lam/n2,xform);

InvKernel =exp(j*2*pi*(zdist-max(T0)).*n2./lam.*sqrt(1-(lam./n2.*fx).^2));
cutoff=(abs(fx)<=n2/lam);
pos1=(x1(1)==0)+1;
pos2=find((x1<L/2)&(x1>L/2-dx));
%----------
% contains weighting parameters
Mag_e2=abs(e2);
Ang_e2=angle(e2);

          DE=sum(wt1.*(Mag_e2.^2))/sum(Mag_e2.^2); % diffraction efficiency
          SN=sum(wt1.*(Mag_e2.^2))/sum(wt2.*(Mag_e2.^2));% signal to noise

Nsym=ceil(Npeaks/2);
loc=(1:Npeaks).'*Peak_sep;loc=loc-mean(loc);
loc2=loc(floor(Npeaks/2)+1:Npeaks);
for p=1:Nsym,
          w(:,p)=rect((abs(x1)-loc2(p))/peak);
end
w=sum(w,2);
%-------------------------------------------------------------------------
```

```
if Npeaks<=2,
                NormE=sqrt(sum(Mag_e2.^2));
        Mag_e2=(1-pert)*not(w).*Mag_e2+w.*Mag_e2*(1+pert);
        NormE=NormE/sqrt(sum(Mag_e2.^2));
        Mag_e2=NormE*Mag_e2;        % Normalize accordingly
        %Mag_e2=not(w).*Mag_e2+w.*Mag_e2*(1+pert);
else;
                NormE=sqrt(sum(Mag_e2.^2));
        Mag_e2=not(w).*Mag_e2+w.*Mag_e2+(pert)*w.*(max(Mag_e2)-Mag_e2);
                NormE=NormE/sqrt(sum(Mag_e2.^2));
        Mag_e2=NormE*Mag_e2;        % Normalize accordingly
end;

e2=Mag_e2.*exp(j*Ang_e2);
A0prime=ifft(e2)*DFTsize*dx;
        a0temp1=(A0prime.*InvKernel).*cutoff;a0temp2=A0.*not(cutoff);
A0prime=A0prime.*InvKernel;

normA=sqrt(sum(cutoff.*abs(A0).^2))/sqrt(sum(cutoff.*abs(A0prime).^2));

A0prime=normA*A0prime.*cutoff+A0.*not(cutoff);
e2=fft(A0prime)/(DFTsize*dx);
%---------
T2=unwrap(angle(e2));
T2_1=arguement(T2(1));
T2=T2-T2(1)+T2_1;
T2(DFTsize:-1:DFTsize-(pos2-pos1))=T2(pos1:pos2);% this is done to get both halves of DOE
T2=rect(x1/L).*T2*lam/(2*pi*dn);
%-------------------------------------------------------------------------
% Energy homogenization

if Npeaks~=1,
        e1=(x1>=0).*rect(x1/L).*exp(j*2*pi.*T0.*(dn./lam));
        [e1,x0,A0,fx]=scalar_analysis1(e1,x1,zdist-max(T0),lam/n2,xform);

        % contains weighting parameters
        Mag_e1=abs(e1);
        Ang_e1=angle(e1);
        %-------------------------------------------------------------------------
        w_left = (x1<0);
        w_right= not(w_left);
        %-------------------------------------------------------------------------
        a_left=sum(w_left.*Mag_e1.^2)./sum(Mag_e1.^2);
        a_right=1-a_left;
        %-------------------------------------------------------------------------
                NormE=sqrt(sum(Mag_e1.^2));
        total_weight=1+pert.*((0.5-a_left).*w_left+(0.5-a_right).*w_right);
        Mag_e1=Mag_e1.*total_weight;
                NormE=NormE/sqrt(sum(Mag_e1.^2));
        Mag_e1=NormE*Mag_e1;                % Normalize accordingly
        e1=Mag_e1.*exp(j*Ang_e1);
```

```
        A0prime=ifft(e1)*DFTsize*dx;
        A0prime=A0prime.*InvKernel;

        normA=sqrt(sum(cutoff.*abs(A0).^2))/sqrt(sum(cutoff.*abs(A0prime).^2));
        A0prime=normA*A0prime.*cutoff+A0.*not(cutoff);
        e1=fft(A0prime)/(DFTsize*dx);
        %----------
        T1=unwrap(angle(e1));
        T1_1=arguement(T1(1));
        T1=T1-T1(1)+T1_1;
        T1(DFTsize:-1:DFTsize-(pos2-pos1))=T1(pos1:pos2);% this is done to get both halves of
DOE
        T1=rect(x1/L).*T1*lam/(2*pi*dn);
else;
        T1=T2;Mag_e1=1;  % %for a lens design
end;
%-------------------------------------------------------------------------
% Get updated DOE profile
T0=(T1+T2)/2; % average of the two cases
T0=Repartition1(T0,x1,P,1,-L/2,L/2,n2-n1);      % Repartitioning of DOE
T0=quantize(T0,Q);                              % Quantization
%-------------------------------------------------------------------------
% Conditions monitoring forIASA runs
if floor(n0/iterprint)==n0/iterprint|(n0==1|n0==iterations),
        % Monitor parameters during program run
        lobe_R=(x1>=0).'*(Mag_e1.^2);
        lobe_L=(x1<0).'*(Mag_e1.^2);
        loR=lobe_R/(lobe_L+lobe_R);             % Display parameters at selected times
        loL=1-loR;
        disp([n0,DE,SN,loL,loR]);
end
%-------------------------------------------------------------------------
end     % Closing iterative loop
%-------------------------------------------------------------------------
```

```
%-----------------------------------------------------
% Post-proceesing file for 1-N beamfanner
%
% Created 5/8/00
%-----------------------------------------------------

% Final partitioning and quantization of DOE
Pfinal=0; % Final number of DOE partitions
Qfinal=0;  % Final number of quantized phase levels
samp_out=1;
```

```
x0_min =-300;
x0_max = 300;

if (Pfinal~=0)&(Qfinal~=0),T_before=T0;end;                    % Profile before R&Q
[T0,x1]=Repartition1(T0,x1,Pfinal,samp_out,-L/2,L/2,n2-n1);    % Repartitioning of DOE
T0=quantize(T0,Qfinal);% Quantization
theta1=0;theta2=0;
Tmiss  =2*n1*cos(theta1)/(n1*cos(theta1)+n2*cos(theta2)); % trans. coeff.for E
[ey,hx]=MaxwellPropagation01('E',Tmiss*rect(x1/L).*exp(j*k0*(dn*T0-n2*max(T0))),zdist-
max(T0),x1,lam,n2);
[ey,xtemp]=pick(ey,x0,x0_min,x0_max);
hx=pick(hx,x0,x0_min,x0_max);

% Final observation plane data
assignin('base',['x0_scalar_',out_file],xtemp);clear xtemp          % x0 positions
assignin('base',['e2_scalar_',out_file],abs(ey).^2);                % Field magnitude^2
assignin('base',['SESCALAR',out_file],0.5*real(-ey.*conj(hx)));  % Sz component (Intensity)

%E_field=scalar_analysis1(rect(x1/L).*exp(j*2*pi/lam*dn*T0),x1,zdist,lam/n2,'angularspectrum
');
%inth=scalar_analysis1((x1>=0).*rect(x1/L).*exp(j*2*pi/lam*dn*T0),x1,zdist,lam/n2,'angularsp
ectrum').^2;




%-------------------------------------------------------------------------------
% ConditionsIASA.m is used for monitoring IASA runs
%
%
%       Created: 5/3/00
%-------------------------------------------------------------------------------
if floor(n0/iterprint)==n0/iterprint|(n0==1|n0==iterations),
        % Monitor parameters during program run
        wt1 =rect((abs(x1)-Peak_pos/2)/photowid);
        % SELECT CENTRAL DETECTORS
        wt2 =rect((rem(abs(x1),Peak_pos)-Peak_pos/2)/photowid)-wt1;  % OUTSIDE
DETECTORS
        lobe_R=(x1>=0).'*int1;
        lobe_L=(x1<0).'*int1;
        loR=lobe_R/(lobe_L+lobe_R);             % Display parameters at selected times
        loL=1-loR;
        disp([n0,sum(wt1.*int)*minsize,sum(wt1.*int)/sum(wt2.*int),loL,loR]);
end
```

APPENDIX G

Miscellaneous codes

This appendix contains the miscellaneous computer programs used for the analysis and design of diffractive optical elements presented in this dissertation. Note that the codes are written in Matlab®.

```
function y=ColPad(x,m);
%-------------------------------------------------------------------------------------------------
% y=ColPad(x,m) repeats a column vector "x" m times to construct an
% n by m matrix in which all the columns are identical.
% Does the same as y=x(:,ones(1,m));
%
%         Example: If X = [4  then ColPad(X,3) is [4 4 4
%                          7                       7 7 7
%                          2]                      2 2 2]
%
% Last revision: 5/16/98
%-------------------------------------------------------------------------------------------------
y=x(:,ones(1,m));



function y=element_value(A,m,n);
%-----------------------------------------------------------------
% y=element_value(A,m,n) gives the matrix element values for the
% mth row and nth column of A.  m and n must be vectors of the
% same length and the output y is the same size of m.
%-----------------------------------------------------------------
%       Example:
%
%       If A= [2    8    4    7          and m=[2  ,      n=[3
%              6    5    6    2                 3]          4]
%              5    0    8    4]
%       then
%
%              y=element_value(A,m,n) is [6
%                                         4].
%-----------------------------------------------------------------
N=length(m);
for k=1:N,
        y(k)=A(m(k),n(k));
end
y=reshape(y,size(m));




function y=flatten(x);
%-----------------------------------------------------------
%       y=flatten(x) takes a matrix "x" row by row and constructs
%       a single column vector "y".
```

```
%-----------------------------------------------------------
y=x(:);



function [y,x,p]=pick(y,x,xmin,xmax);
%-----------------------------------------------------
% [y,x,p]=pick(y,x,xmin,xmax) is used to "pick off" the
% values of y that correspond to x within the range of
% xmin <= x <= xmax. p is the list of corresponding
% positions.  All variables are sorted in ascending
% order of x.
%
% Created: 2/11/00
%-----------------------------------------------------
if nargin==3,xmax=xmin;xmin=x;x=y;end;
if nargin==2,xmax=max(x);xmin=min(x);end;

p=find(x>=xmin&x<=xmax);

y=y(p);
x=x(p);

Mtemp=sortrows([x,y,p]);
x=Mtemp(:,1);y=Mtemp(:,2);p=Mtemp(:,3); clear Mtemp
%-----------------------------------------------------


function y=quantize(T,num,maxvalue,minvalue);
%-------------------------------------------------------------------
% y=quantize(T,n,maxvalue,minvalue).
% This function quantizes a dataset, T, to the nearest nth level where
% the limits of thequantization are set by the minimum and
% maximum values in the set by default.
%
% Note:  for n=0 or 1, data set is left unquantized.
%
%       Optional variables:
%
%               maxvalue:       quantization w.r.t this value as upper limit
%               minvalue:       quantization w.r.t this value as lower limit
%
%       Last revision:    11/18/98
%-------------------------------------------------------------------
if nargin<4,minvalue=[];end             % no set min value
if nargin<3,maxvalue=[];end             % no set max value

if (num==0|num==1),                             % otherwise set to default case
        y=T; % no change
else
```

```
        tmax=max([maxvalue;max(T(:))]);
        tmin=min([minvalue;min(T(:))]);
        T=T-tmin;
        dt=(tmax-tmin)/(num-1);
                dt=dt+(dt==0);
        y=tmin+dt*round(T/dt);          % quantized set of values
end


function [y1,varargout]=reading_files(filename,format_type);
%-------------------------------------------------------------------------
% function [y1,varargout]=reading_files(filename,format_type) is used for
% reading data from a file called filename. Included are options to read the
% data according to a type of format. y1,y2, and y3 are column vectors.
%
%       created 10/6/1999
%-------------------------------------------------------------------------

nout=nargout;
                if nargin>1,
                        s1=format_type;                         % format for data
                else
                        s1='%g ';                               % default format
                        %s1='%12.8f ';
                end
                s2=[];for m=1:nout,s2=[s2,s1];end

fid = fopen(filename);
         b = fscanf(fid,s2,[nout inf]);
         b = b.';
fclose(fid);

y1=b(:,1);
for k=1:nout-1, varargout(k) = {eval(['b(:,',num2str(k+1),')'])}; end


function y=rect(x)
% RECT(x) gives 1 for Abs(x)<0.5 and zero otherwise
y = 0.5*(sign((x+eps)+0.5)+sign(0.5-(x-eps)));
function [y,x]=Repartition1(y,x,P,samp1,xmin,xmax,offset);
%-------------------------------------------------------------
% Repartitioning of DOE etch depth data. Created: 5/4/00
%
%       function [y,x]=Repartition1(y,x,P,samp1,xmin,xmax,offset);
%
%       Inputs:
%       y       =       DOE etch depth [µm]                             (column vector)
%       x       =       corresponding DOE positions [µm]        (column vector)
%       P       =       Number of partions
%       samp1=  samples per partition (note-> 0 means no extra sampling)
%       xmin=   minimum range to resample [µm]
```

```
%       xmax=  maximum range to resample [µm]
%       offset= option to account for reseting DOE to minumum etch
%        if offset = 0 -> no reset
%         > 0 -> for n2>n1 (e.g. Air to Si)
%         < 0 -> for n2<n1 (e.g. Si to Air)
%       Outputs:
%       y       =       resampled etch depths [µm]      (column vector)
%       x       =       corresponding positions [µm]    (column vector)
%-------------------------------------------------------------------
if nargin<7,offset=0;elseif isempty(offset),offset=0;end;
if nargin<6,xmax=max(x);elseif isempty(xmax),xmax=max(x);end;
if nargin<5,xmin=min(x);elseif isempty(xmin),xmin=min(x);end;
if nargin<4,samp1=1;end;        % If "true", keep same sampling.
%-------------------------------------------------------------------------
% Pick off the appropriate positions in list
[Z,Xdat,pos]=pick(y,x,xmin,xmax);% values ordered in ascending values of Xdat
Ldat=length(Xdat);                                      % length of input file values in range
%-------------------------------------------------------------------------
% Adjust profile if there is resampling
if (P==0|isempty(P)),P=Ldat;x1=Xdat;                            % default case (no re-sampling)
else;
        L =max(Xdat)-min(Xdat)+abs(Xdat(2)-Xdat(1));    % lateral width of DOE [µm
        x1= mean(Xdat)+(linspace(-1,1,P)*(P-1)/P).'*L/2;        % positions in ascending order
        Z=mean(reshape(flatten(RowPad(Z.',P)),Ldat,P).',2); % resampled etch depth
end;
%-------------------------------------------------------------------------
% Resetting option - adjust thickness profile with re-sampling and reset
Z=Z-min(Z).*(offset>0)-max(Z).*(offset<0);
%-------------------------------------------------------------------------
% Restructure as necessary. That is, sample each resampled feature.
samp=round(samp1*Ldat/P);
if samp1~=0,% Additional sampling DOE contour (samp=1 default value)
samp=samp+(samp==0);
d=abs(x1(2)-x1(1))*linspace(-0.5,0.5,samp)*(samp-1)/samp;% intermediate variable
x1=flatten((ColPad(x1,samp)+RowPad(d,length(x1))).');clear d
Z =flatten(RowPad(Z.',samp));
if Ldat==length(x1),x(pos)=x1;y(pos)=Z;else;x=x1;y=Z;end;
else;
        x=x1;
        y=Z;
end;
%-------------------------------------------------------------------------


function y=RowPad(x,n);
%-----------------------------------------------------------------------------------------------------
% y=RowPad(x,m) repeats a row vector "x" n times to construct an
% n by m matrix in which all the rows are identical.
% Does the same as y=x(ones(1,n),:);
%
```

```
%         Example: If X = [0 1 2] then RowPad(X,2) is [0 1 2
          %                                             0 1 2]
%
% Last revision: 5/16/98
%-------------------------------------------------------------------------------------------------
y=x(ones(1,n),:);


function y=saving_files(data,filename,directory,format_type);
%-------------------------------------------------------------------------
% function y=saving_files(data,filename,directory,format_type) is used for
% saving data in a file called filename. Included are options to save in a
% specified directory with a type of format.
%
%        created 9/9/1999
%-------------------------------------------------------------------------
old_directory=cd;                                   % original operating directory
if nargin>2,
          if not(isempty(directory)),
                    cd(directory);                  % option to save to different directory
          end;
end;
fid = fopen(filename,'w');                          % open file and write data
          dim=size(data,2);
                    if nargin>3,
                              s1=format_type;       % format for data
                    else
                              s1='%12.8f ';         % default format
                    end
          s2=[];for m=1:dim,s2=[s2,s1];end
          header=[s2,'\n'];
   fprintf(fid,header,data.');
    fclose(fid);                                    % close file
            y=fid;
cd(old_directory);                                                  % return to orignal
directory

%-------------------------------------------------------------------------




function y=ShiftFFTcolumns(x);

% Shift DC components to center of spectrum.
% For vectors, ShiftFFTcolumns(X) swaps the top and bottom halves of
% X.  For matrices, ShiftFFTcolumns(X) swaps the top and bottom
% halves of each column. Unlike FFTSHIFT, ShiftFFTcolumns is suitable
% for multiple columns for one-dimensional FFTs.
```

```
n=round(size(x,1)/2);
y=shiftud(x,n,1);
```

```
function y=super_gaussian(x,d,M);
%-----------------------------------------------------------------
%       y=super_gaussian(x,d,M) takes the form:
%
%       y=exp(-2*(x/(d/2)).^(2*M))  and is a useful smoothed
%                               out version of a rect function.
%
%                               x=input coordinate values
%                               d=width of window
%                               M=order of super-gaussian (M=1,2,3,...)
%
%                       Created:          11/6/98
%-----------------------------------------------------------------
y=exp(-2*(x/(d/2)).^(2*M));
```

REFERENCES

[1] J. Tuunen and F. Wyrowski, "Diffractive optics - State of the art," SPIE **2778**, 2-4 (1996).

[2] J. W. Goodman, *Introduction to Fourier Optics* (McGraw-Hill Book Company, New York, 1968).

[3] D. Prather, M. Mirotznik, and J. Mait, "Boundary integral methods applied to the analysis of diffractive optical elements," J. Opt. Soc. Am. A **14**, 34-43 (1997).

[4] C.A. Balanis, *Advanced Engineering Electromagnetics* (John Wiley & Sons, New York, 1989).

[5] A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method* (Artech House, Massachusetts, 1995).

[6] I. Richter, P. Sun, F. Xu, and Y. Fainman, "Design considerations of form birefringent microstructures," Appl. Opt. **34**, 2421-2429 (1995).

[7] J.R. Fienup, "Phase retrieval algorithms: a comparison," Appl. Opt. **21,** 2758-2769 (1982).

[8] M. Born and E. Wolf, *Principles of Optics*, 2nd ed. (Pergamon Press, London, 1964).

[9] C. Gu and P. Yeh, "Form birefringence dispersion in periodic layered media," Opt. Lett. **21**, 504-506 (1996).

[10] K. Hirayama, E.N. Glytsis, and T.K. Gaylord, "Rigorous electromagnetic analysis of diffraction by finite-number-of-periods gratings," J. Opt. Soc. Am. A **14**, 907-917 (1997).

[11] M. Schmitz and O. Bryngdahl, "Rigorous concept for the design of diffractive microlenses with high numerical apertures," J. Opt. Soc. Am. A **14,** 901-906 (1997).

[12] E. Noponen and J. Saarinen, "Rigorous synthesis of diffractive optical elements," SPIE **2689**, 54-65 (1996).

[13] D. Prather, M. Mirotznik, and J. Mait, "Boundary element method for vector modelling diffractive optical elements," SPIE **2404**, 28-39 (1996).

[14] M. Mirotznik, D. Prather, and J. Mait, "A hybrid finite element-boundary element method for the analysis of diffractive elements," J. Mod. Opt., **43**, 1309-1321 (1996).

[15] J. N. Mait, "Understanding diffractive optic design in the scalar domain," J. Opt. Soc. Am. A **12**, 2145-2158 (1995).

[16] M. Kuittinen and P. Herzig, "Encoding of efficient diffractive microlenses," Opt. Lett. **20**, 2156-2158 (1995).

[17] Y. Lin, T. J. Kessler, and G. N. Lawrence, "Design of continuous surface-relief phase plates by surface-based simulated annealing to achieve control of focal plane irradiance," Opt. Lett. **21**, 1703-1705 (1996).

[18] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of phase from image and diffraction plane picutures," Optik. **35**, 237-246 (1971).

[19] Z. Zalevsky, D. Mendlovic, and R. G. Dorsch, "Gerchberg-Saxton algorithm applied in the fractional Fourier or the Fresnel domain," Opt. Lett. **21**, 842-844 (1996).

[20] J. R. Fienup, "Iterative method applied to image reconstruction and to computer-generated holograms," Opt. Eng. **19**, 297-305 (1980).

[21] M. G. Moharam and T. K. Gaylord, "Diffraction analysis of dielectric surface-relief gratings," J. Opt. Soc. Am. A **72**, 1385-1392 (1982).

[22] M. G. Moharam, E. B. Grann, and D. A. Pommet, "Formulation for stable and efficient implementation of rigorous coupled-wave analysis of binary gratings," J. Opt. Soc. Am. A **12**, 1068-1076 (1995).

[23] G. S. Smith, *An Introduction to Classical Electromagnetic Radiation* (Cambridge Univ. Press, New York, 1997).

[24] K. Yashiro and S. Ohkawa, "Boundary element method for electromagnetic scattering from cylinders," IEEE Trans. Antennas Propagat. **AP-33**, 383-389 (1985).

[25] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," IEEE Trans. Antennas Propagat. **AP-14**, 302-307 (1966).

[26] W. F. Ames, *Numerical Methods for Partial Differential Equations*, 2nd ed. (Acedemic Press, New York, 1977).

[27] A. Taflove and M. E. Brodwin, "Numerical solution of steady-state electromagnetic scattering problems use the time-dependent Maxwell's equations," IEEE Trans. Microwave Theory and Techniques **23**, 623-630 (1975).

[28] S. L. Ray, "Numerical dispersion and stability characteristics of time-domain methods on nonorthogonal meshes," IEEE Trans. Antennas Propagat. **AP-41**, 233-235 (1993).

[29] B. Engquist and A. Madja, "Absorbing boundary conditions for the numerical simulation of waves," Mathematics of Computation **31**, 629-651 (1977).

[30] G. Mur, "Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic field equations," IEEE Trans. Electromagnetic Compatibility **23**, 377-382 (1981).

[31] J. P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," J. Computational Physics **114**, 185-200 (1994).

[32] D. S. Katz, E. T. Thiele, and A. Taflove, "Validation and extension to three-dimensions of the Berenger PML absorbing boundary condition for FDTD meshes," IEEE Microwave and Guided Wave Letters **4**, 268-270 (1994).

[33] K. R. Umashankar and A. Taflove, "A novel method to analyze electromagnetic scattering of complex objects," IEEE Trans. Electromagnetic Compatibility **24**, 397-405 (1982).

[34] W. J. Tsay and D. M. Pozar, "Application of the FDTD technique to periodic problems in scattering and radiation," IEEE Microwave and Guided Wave Letters **3**, 250-252 (1993).

[35] J. D. Jackson, *Classical Electrodynamics*, 2nd ed. (John Wiley & Sons, New York, 1975).

[36] D. A. Gremaux and N. C. Gallagher, "Limits of scalar diffraction theory for conducting gratings," Appl. Opt. **32**, 1948-1953 (1993).

[37] D. A. Pommet, M. G. Moharam, and E. B. Grann, "Limits of scalar diffraction theory for diffractive phase elements," Opt. Lett. **11**, 1827-1834 (1995).

[38] P. St. Hilaire, "Phase profiles for holographic stereograms," Opt. Eng. **34**, 83-89 (1995).

[39] N. C. Gallagher and B. Liu, "Method for computing kinoforms that reduces image reconstruction error," Appl. Opt. **12**, 2328-2335 (1973).

[40] F. Wyrowski, "Iterative Fourier-transform algorithm applied to computer holography," J. Opt. Soc. Am. A **5**, 1058-1065 (1988).

[41] F. Wyrowski, "Digital holography as part of diffractive optics," Rep. Prog. Phys. **54** 1481-1571 (1991).

[42] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, "Generalized simulated annealing for function optimization," Technometrics **28**, 209-217 (1986).

[43] B. K. Jennison, J. P. Allebach, and D. W. Sweeney, "Iterative approaches to computer-generated holography," Opt. Eng. **28**, 629-637 (1989).

[44] J. Turunen, A. Vasara, and J. Westerholm, "Kinoform phase relief synthesis: A stochastic method," Opt. Eng. **28**, 1162-1167 (1989).

[45] M. R. Feldman and C. C. Guest, "High-efficiency hologram encoding for generation of spot arrays," Opt. Lett. **14**, 479-481 (1989).

[46] M. Jones, *Partial Pixels: A Real-Time 3-D Display Architecture*, PhD. Dissertation, (University of Alabama in Huntsville, 1996).

[47] G. P. Nordin, J. H. Kulick, M. Jones, P. Nasiatka, R. G. Lindquist, and S. T. Kowel, "Demonstration of a novel three-dimensional autostereoscopic display," Opt. Lett. **19**, 901-903 (1994).

[48] P. St. Hilaire, S. A. Benton, M. Lucente, J. Underkoffler, and H. Yoshikawa, "Real-time holographic display: Improvements using a multichannel acouso-optic modulator and holographic optical elements," SPIE **1461**, 254-261 (1991).

[49] W. H. Press, B. P. Flannery, S. A. Teukolsky, and T. Vetterling, *Numerical Recipes: The Art of Scientific Computing* (Cambridge Univ. Press, New York, 1986).