Spring 2005

Numerical simulation of nanopulse penetration of biological matter using the z -transform

Shengjun Su

NUMERICAL SIMULATION OF NANOPULSE PENETRATION OF BIOLOGICAL MATTER USING THE Z-TRANSFORM

by

Shengjun Su, B.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2005

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

April 26, 2005
Date

We hereby recommend that the dissertation prepared under our supervision by Shengjun Su

entitled Numerical Simulation of Nanopulse Penetration of Biological Matter Using the Z-Transform

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computational Analysis and Modeling

Supervisor of Dissertation Research

Head of Department

CAM
Department

Recommendation concurred in:

Advisory Committee

Approved:
Director of Graduate Studies

Approved:
Dean of the Graduate School

Dean of the College

GS Form 13
(5/03)

ABSTRACT

Short duration, fast rise time ultra-wide-band (UWB) electromagnetic pulses ("nanopulses") are generated by numerous electronic devices in use today. Moreover, many novel technologies involving nanopulses are under development and expected to become widely used soon. Study of nanopulse bioeffects is needed to probe their useful range in possible biomedical and biotechnological applications, and to ensure human safety.

Based on the well-known dispersive properties of biological matter and their expression as a summation of terms corresponding to the main polarization mechanisms, the Cole-Cole expression is commonly employed to describe the frequency dependence of the dielectric properties of a tissue. Solving the Maxwell's equations coupled with the Cole-Cole expression, however, is difficult because it is not easy to convert the equations from the frequency domain to the time domain.

In this work we develop a computational approach to investigating electromagnetic fields in biological matter exposed to nanopulses, where the relative dielectric constant is given by the Cole-Cole expression for the frequency dependence of the dielectric properties of tissues. The Cole-Cole expression is first transformed to the z-domain using the z-transform method and then approximated by a second-order Taylor series of variable z. After converting the result from the frequency domain to the time

domain, the finite-difference time-domain method (FDTD) is used to solve Maxwell's equations coupled with the Cole-Cole expression, and a perfectly matched layer is applied to eliminate reflections from the boundary.

The method is then applied to investigating the penetration of a short electromagnetic pulse into biological matter, where the relative dielectric constant is given by the Cole-Cole expression. Transmission, reflection, and absorption are calculated as a function of pulse width. It is found that these properties depend substantially on pulse characteristics.

Future work in this direction could be examining the relevance of pulse rise time and pulse shape to tissue penetration. Such study could help to elucidate non-thermal mechanisms of nanopulse bioeffects.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author 

Date 05/10/05

GS Form 14
(5/03)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

E	the electric field intensity , V/m
H	the magnetic field intensity, A/m
D	the electric flux density , C/m^2
B	the magnetic flux density, Wb/m^2
J	the current density , A/m^2
ρ	the volume electric charge density , C/m^3
μ	the permeability
ε	the permittivity
σ	the conductivity
μ_0	the permeability of free space.
ε_0	the permittivity of free space.
μ_r	relative permeability
ε_r	relative permittivity
η	the intrinsic impedance
σ_s	Static Conductivity

ACKNOWLEDGMENTS

I wish to express my gratitude to:

Dr. Weizhong Dai, My Ph.D. advisor, for his invaluable guidance, encouragement and generous support throughout my three-year study atn Louisiana Tech University.

Dr. Donald Haynie, for providing me the opportunity to work on this project and his continuous support and guidance during the course of my research.

Dr. Neven Simicevic, for his generous help and advice with the theoretical research.

Dr. Raja Nassar, for being on my advisory committee and for his valuable suggestions during my research.

Dr. Paul Andrei Paun, for his help with work on bio-computing.

Dr. Nathan Champagne, for his advice on simulation of electro-magnetic fields.

Dr. Richard Greechie, for providing me the opportunity to study for my Ph.D. in the CAM program at Louisiana Tech University.

My girlfriend Wei Jiang, for helping me to get through the busiest time ever in my life.

CHAPTER 1

INTRODUCTIONS

1.1 Overview

Short-duration, fast rise time ultra-wide-band (UWB) electromagnetic pulses ("nanopulses") are generated by a broad range of electronic devices in use today, notably communications instruments. There is also interest in nanopulses in the development of ground-penetrating radar. Such devices can produce electromagnetic pulses with pulse widths of just a few nanoseconds and electric field amplitudes greater than $10^5 V/m$. Many new technologies involving nanopulses are expected to become widely available in the near future, as the Federal Communications Commission issued a Final Rule on UWB in 2002 permitting the marketing and operation in the USA of products involving UWB. Study of nanopulse bioeffects, therefore, is needed to ensure human safety and to explore the useful range of such pulses in biomedical and biotechnological application.

Possible technologies involving nanopulses in medical settings include electroporation, allowing chemotherapeutic drugs to enter and kill cancer cells, and the development of new techniques for imaging tissue structures [1]. Possible undesirable health effects resulting from nanopulse exposure are tissue damage, conformational changes in macromolecules, alteration of biochemical reaction rates, membrane effects other than electroporation, and temperature effects. It will be important to know the field

characteristics inside a tissue exposed to nanopulses to give a rational foundation to decision making on approval of a medical procedure involving. In many realistic situations, however, it is no simple matter to measure the electromagnetic field. Mathematical modeling has thus become indispensable for gaining a good grasp of the situation.

A number of mathematical models have been developed for the investigation of electromagnetic bioeffects [2-48]. Lin, for example, has studied the interaction of electromagnetic pulse with biological structures [2, 3, 4]. Samn and Mathur [5] have developed a mathematical model of a gigahertz-transverse electromagnetic-mode cell using FDTD code for an electromagnetic field in a tissue developed by Kunz and Luebbers [6]. The latter work, based on the Yee algorithm [7, 8, 9], included a more recent treatment of absorbing boundary conditions (perfectly matched layer method) [10, 11]. Schoenbach *et al.* [12] have employed a spherical cell model, introduced by Foster [13], and described the coupling of an electric field to the nuclear membrane. The results suggest an increasing probability of electric field interactions with cell substructures in prokaryotic and eukaryotic cells as pulse width is reduced into the sub-microsecond range. Joshi and Schoenbach [14, 15] have studied the temporal dynamics of electroporation of cells subjected to ultrashort voltage pulses, based on a coupled scheme involving the Laplace, Nernst-Planck, and Smoluchowski equations. The same authors have also proposed a self-consistent model analysis of electroporation in biological cells based on an improved energy model [16].

The Cole-Cole expression is commonly employed to model the frequency dependence of the dielectric properties of a tissue [17, 18, 19, 20, 21, 22]:

$$\varepsilon_r^*(\omega) = \varepsilon_\infty + \sum_{m=1}^{4} \frac{\Delta\varepsilon_m}{1 + (j\omega\tau_m)^{1-\alpha_m}} + \frac{\sigma_s}{j\omega\varepsilon_0} \tag{1.1}$$

where $\varepsilon_r^*(\omega)$ is the relative dielectric constant, ε_0 is the permittivity of free space, ω is the angular frequency, ε_∞ is the permittivity in the terahertz frequency range, σ_s is the ionic conductivity, and $j = \sqrt{-1}$. For each dispersion region m, τ is the relaxation time, α is an adjustable parameter between 0 and 1, and $\Delta\varepsilon$ is the change in permittivity in the corresponding frequency range. This model is based on the well-known dispersive properties of biological matter; the expression as a summation of terms corresponds to several main polarization mechanisms; the dielectric spectrum from Hz to GHz shows 4 major regions of dispersion [20]. The complexity of the structure and composition of biological matter is such that each dispersion region is broadened by multiple contributions. With a choice of parameters appropriate to each tissue, Equation (1.1) can be used to predict dielectric behavior over the desired frequency range [19, 21, 22]. Solving Maxwell's equations when coupled to the Cole-Cole expression is difficult, however, because it is not easy to convert from the frequency domain to the time domain. To overcome this difficulty, the Cole-Cole model is usually approximated by a Debye model or a Lorentz model [23, 24, 25] as follows

$$\varepsilon_r^*(\omega) = \varepsilon_\infty + \sum_{m=1}^{N} \frac{\Delta\tilde{\varepsilon}_m}{1 + j\omega\tilde{\tau}_m} + \frac{\tilde{\sigma}_s}{j\omega\varepsilon_0}, \tag{1.2}$$

$$\varepsilon_r^*(\omega) = \varepsilon_\infty + \sum_{m=1}^{N} \frac{\Delta\tilde{\varepsilon}_m \omega}{\omega_m^2 + 2j\omega\tilde{\tau}_m - \omega^2} + \frac{\tilde{\sigma}_s}{j\omega\varepsilon_0}. \tag{1.3}$$

However, $\Delta\tilde{\varepsilon}_m$, $\tilde{\tau}_m$ and $\tilde{\sigma}_s$ could be different from $\Delta\varepsilon_m$, τ_m and σ_s, which means we need to recalculate these parameters.

1.2 Objective of the Research

The objective of this research is to find a new approximation of the Cole-Cole expression without recalculating its parameters and develop a finite difference schemes for solving Maxwell's equations coupled with it. To achieve this objective, the following missions are carried out in this dissertation:

(1) Approximate the Cole-Cole expression based on a z-transformation of the electric displacement and Taylor expansion.

(2) Develop finite difference schemes in 1D, 2D and 3D using finite difference time domain method.

(3) Utilize a perfectly matched layer to eliminate reflection from the boundary.

(4) Apply the schemes to investigate the influence of variations in pulse shapes and in tissues on the electromagnetic fields.

The outcome of the work will provide an efficient numerical simulation for obtaining electromagnetic fields in biological tissues exposed to nanopulses, and give us a better understanding of the bioeffects of nanopulses.

1.3 Overview of the Dissertation

Chapter 2 contains reviews of underlying theory. First, Maxwell's Equations are carefully discussed. Then, the FDTD method is described in detail as well as the z-transform method, which is a useful tool to solve the Maxwell's equation coupled to the Cole-Cole expression. After that follows the review of dielectric properties of biological tissues.

Chapter 3 is concerned with governing equations. Governing equations are listed and discussed carefully then the difficulty in solving the system is presented. Finally, the source term is presented in details.

Chapter 4 discusses the detailed simulation. First of all, the governing equations are discretized. Then the dielectric constant for biological matter is approximated in different ways. After that, the boundary conditions are discussed in 1D, 2D and 3D. Later on, the nanopulse is simulated, as well the energy of it is calculated. Eventually, the whole algorithm is presented step by step.

Chapter 5 shows the numerical results obtained by the FDTD method. Results of different dielectric models and different dimensions of problem space are carefully plotted. Then the transmission, reflection and absorption of the pulse energy are calculated and discussed.

Chapter 6 gives the conclusion of this dissertation and points out the future work as well.

CHAPTER 2

BACKGROUND

2.1 Maxwell's Equations

Based on the work and experiments of Ampere, Gauss, and Faraday, James Clerk Maxwell (1832-1879) unified the studies of his predecessors in four equations and foresaw the physical phenomenon of propagation of electromagnetic waves. Nine years after Maxwell's death, Heinrich Hertz discovered electromagnetic waves experimentally, proving the global view of Maxwell's theory.

The electromagnetic formalism is extremely simple and based primarily on Maxwell's equations. The general, time dependent, Maxwell's equations in differential form can be expressed as follows [26]:

$$\nabla \times E = -\frac{\partial B}{\partial t} \tag{2.1}$$

$$\nabla \times H = J + \frac{\partial D}{\partial t} \tag{2.2}$$

$$\nabla \cdot D = \rho \tag{2.3}$$

$$\nabla \cdot B = 0 \tag{2.4}$$

where the various quantities involved are defined as

E - the electric field intensity

H - the magnetic field intensity

D - the electric flux density

B - the magnetic flux density

J - the current density

ρ -the volume electric charge density

Equation (2.1) is known as Faraday's law of induction. It shows that the time derivative of the magnetic flux density can generate an electric field intensity. This equation defines the basic laws of induction, and it is most often associated with eddy current application.

Equation (2.2) is Ampere's law. It expresses the manner by which a magnetic field can create a split into conduction current (associated with J) and a time variation of the electric flux density (associated with $\partial D / \partial t$). And the term $\partial D / \partial t$ is also known as the displacement current density as opposed to the conduction current density J.

Equation (2.3) is Gauss's law, the observation that the divergence of D is zero or not demonstrates whether the electric flux is conservative. We usually associate it with electrostatic applications.

Equation (2.4) is not associated with a particular law and simply states the nonexistence of isolated magnetic poles. It signifies that the magnetic flux is conservative. Because of the form of the expression it is sometimes referred to as the magnetic form of Gauss's law.

From these equations, we can define a fifth relation. Applying the divergence on both sides of Equation (2.2) gives

$$\nabla \cdot (\nabla \times H) = \nabla \cdot J + \nabla \cdot \frac{\partial D}{\partial t}.$$

Using the fact that $\nabla \cdot (\nabla \times H) = 0$, we have

$$0 = \nabla \cdot J + \frac{\partial}{\partial t}(\nabla \cdot D).$$

Utilizing Equation (2.3) gives

$$\nabla \cdot J = -\frac{\partial \rho}{\partial t}. \tag{2.5}$$

This equation is called the electrical continuity equation. Obviously, from the above derivation, Equation (2.3) can be obtained from Equation (2.2), if the continuity equation is postulated. Similarly, Equation (2.4) can be obtained from Equation (2.1) by applying the divergence on both sides of Equation (2.1). Therefore, only two of the Maxwell's equations, Equation (2.1) and Equation (2.2), are independent.

The Maxwell's equations in differential form, Equation (2.1) and Equation (2.2), are written as linear partial differential equations. The important property of the field equations is defined by the interaction of fields with materials; therefore, it is necessary to associate linearity or nonlinearity of field relations with material properties. Since the field equations describe vector relations, and since there are four field quantities (E, H, D, and B), the independent equations, Equations (2.1) and (2.2), are equivalent to 6 scalar equations in 12 unknowns. Thus two additional relations

$$B = \mu H \tag{2.6}$$

and

$$D = \varepsilon E \tag{2.7}$$

are needed to complete the system, where μ is the permeability of materials and ε the

permittivity. These relations, equivalent to six scalar equations, are called the material constitutive relations which provide a link between four fields. In general, these relations are nonlinear.

In addition, we define a constitutive relation involving current densities and the electric field intensity

$$J = \sigma E \ , \tag{2.8}$$

where σ is the conductivity.

The constitutive relations, Equations (2.6) - (2.8), define the interaction between fields and materials. These magnetic and electric properties of materials are the most important factors in testing because of their effect on material behavior and fields. In terms of modeling, material properties defined not only what type of computation is needed but also limit the model.

Conductivity of a material can be broadly defined as its ability to conduct electric current. Under the influence of an electric field, free electrons move at various velocities. The electron velocity is proportional to the electric field and, therefore, the current density J in a conductor can be directly related to the applied electric field intensity E as Equation (2.8) which is Ohm's law in point form.

Magnetic properties of materials are due to the interaction of external magnetic fields and moving charges in materials. Atomic-scale magnetic fields are produced inside materials through orbiting electrons. Equivalent current loops are generated due to these electrons. These behave like small magnets (magnetic moments). With assemblage of many such magnetic moments, the material volume contains a certain magnetic moment density. A net magnetic field is generated inside the material. This internal field is either

aligned with external field to increase to total field or opposes it to decrease to the total field. If the internal magnetic moments are randomly oriented, as is often the case, the net internal field is zero and the material behaves like free space from the magnetic point of view. This is also the case for nonmagnetic materials. In general, the permeability μ of a material expresses an intrinsic capacity of the material and indicates how much or how little it is susceptible to passage of the magnetic flux. Equation (2.6) shows the meaning of μ in a simple manner. Technically, μ can be a complex number. The imaginary part of the complex permeability represents losses due to damping forces in the magnetic material. The real part represents materials without magnetic losses. In most cases there will be no need to consider permeability as a complex quantity since the real part for most materials is dominant. For practical use, it is common to define a relative permeability μ_r through the relation

$$\mu = \mu_0 \mu_r \ , \tag{2.9}$$

where μ_0 is the permeability of free space.

The electric properties of dielectric materials are largely defined by polarization of charges within the material due to an applied external electric field. Since charges in these materials are bound, conduction is negligible, but polarization of charges (i.e., alignment of electric dipoles with the external electric field) may be significant. This polarization increases the electric flux density in the material. We can thus write the constitutive relation of Equation (2.7) as

$$D = \varepsilon_0 E + P \quad , \tag{2.10}$$

where P is the polarization vector and ε_0 is the permittivity of free space [26]. The polarization vector is proportional to the electric field. Since the polarization vector P is

proportional to the electric field. Since the polarization vector P is proportional to the external electric field intensity E we can write

$$D = \varepsilon_0 E + \varepsilon_0 \chi_e E = \varepsilon_0 (1 + \chi_e) E \quad , \tag{2.11}$$

where χ_e is the electric susceptibility which is a dimensionless quantity. The quantity

$$\varepsilon = \varepsilon_0 (1 + \chi_e) \tag{2.12}$$

is defined as the complex permittivity. This is often written as

$$\varepsilon = \varepsilon' + j\varepsilon'' \tag{2.13}$$

to signify the fact that permittivity is a complex quantity. The imaginary part of ε represents dielectric losses. However, since the loss mechanism in materials includes two loss components; one is due to dielectric, or polarization losses, the other due to conduction currents, the complex permittivity can be redefined as

$$\varepsilon = \varepsilon' - j(\frac{\omega\varepsilon'' + \sigma}{\omega}) \tag{2.14}$$

which most easily describes the loss mechanism. The real part of the complex permittivity is the term we normally associate with dielectrics as the dielectric constant. In lossless materials, the imaginary part is zero. In lossy materials, the imaginary part is nonzero and has two parts; the first, due to ε'' is the dielectric loss while the second due to σ is the conduction, although, in practice the two types of losses are indistinguishable.

2.1.1 Maxwell's Equations in a Non-Frequency Dependent Medium

Assuming all the materials being simulated are nonmagnetic (that is, $H = B / \mu_0$, and non-frequency dependent, which is specified by the relative dielectric constant ε_r and a constant conductivity σ), we can write down the time-dependent Maxwell's curl equations in a general form [26]:

$$\varepsilon \frac{\partial E}{\partial t} = \nabla \times H - J \,, \tag{2.15}$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\mu_0} \nabla \times E \,. \tag{2.16}$$

Substituting

$$J = \sigma E$$

and

$$\varepsilon = \varepsilon_0 \varepsilon_r$$

into Equation (2.15), we obtain :

$$\frac{\partial E}{\partial t} = \frac{1}{\varepsilon_0 \varepsilon_r} \nabla \times H - \frac{\sigma}{\varepsilon_0 \varepsilon_r} E \,. \tag{2.17}$$

Because ε_0 and μ_0 differ by several orders of magnitude, E_z and H_x will differ by several orders of magnitude. This problem can be circumvented by making the following change of variable [27]

$$\widetilde{E} = \sqrt{\frac{\varepsilon_0}{\mu_0}} E \,, \tag{2.18}$$

which gives the "normalized" Maxwell's equations in a non-frequency dependent medium as follows:

$$\frac{\partial \widetilde{E}}{\partial t} = \frac{1}{\varepsilon_r \sqrt{\varepsilon_0 \mu_0}} \nabla \times H - \frac{\sigma}{\varepsilon_0 \varepsilon_r} \widetilde{E} \,, \tag{2.19}$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\sqrt{\varepsilon_0 \mu_0}} \nabla \times \widetilde{E} \,. \tag{2.20}$$

This system is called Gaussian units, a term frequently used by physicist for the reason of simplicity in the formulations.

2.1.2 Maxwell's Equations in a Frequency Dependent Medium

The dielectric constant and conductivity of most media vary at different frequencies, which is of most interest in our research. As for getting to these frequency dependent materials, it is necessary to change the formulation slightly and introduce the use of the flux density into the simulation. As such, we write down a more general form of Maxwell's Equations [28]:

$$\frac{\partial D}{\partial t} = \nabla \times H \tag{2.21}$$

$$D(\omega) = \varepsilon_0 \varepsilon_r(\omega) E(\omega) \tag{2.22}$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\mu_0} \nabla \times E \tag{2.23}$$

where D is the electric flux density and the relative dielectric constant, $\varepsilon_r(\omega)$, is described by the Cole-Cole expression we will discuss later in this chapter. Normalizing these equations with

$$\widetilde{E} = \sqrt{\frac{\varepsilon_0}{\mu_0}} E,$$

and

$$\widetilde{D} = \sqrt{\frac{1}{\varepsilon_0 \cdot \mu_0}} D$$

leads to

$$\frac{\partial \tilde{D}}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \nabla \times H \,, \tag{2.24}$$

$$\tilde{D}(\omega) = \varepsilon_r(\omega)\tilde{E}(\omega), \tag{2.25}$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\sqrt{\mu_0 \varepsilon_0}} \nabla \times \tilde{E} \,. \tag{2.26}$$

Equation (2.24)-(2.26) are the "normalized" Maxwell's equations in a frequency dependent medium.

2.2 FDTD Method

A number of mathematical models have been developed for the investigation of electromagnetic bioeffects [2-48]. Especially, the finite difference time domain (FDTD) formulation is developed as a convenient tool for solving scattering problems of EM field. The FDTD methods, first introduced by Yee [9] in 1966 and later developed by Taflove and other [29-35], is a direct solution of Maxwell's time-dependent curl equations.

2.2.1 Notations

Following Yee's notation [9], we denote a grid point in the solution region as

$$(i, j, k) = (i\Delta x, j\Delta y, k\Delta z), \tag{2.27}$$

and any function of space and time as

$$F^n(i, j, k) = F(i\Delta x, j\Delta y, k\Delta z, n\Delta t), \tag{2.28}$$

where $\Delta x = \Delta y = \Delta z$ is the space increment, and Δt is the time increment, while i, j, k and n are integers. Using the second-order accurate central finite difference

approximation for spatial and temporal derivatives, one may obtain the following two equations:

$$\frac{\partial F^n(i,j,k)}{\partial x} = \frac{F^n(i+\frac{1}{2},j,k) - F^n(i-\frac{1}{2},j,k)}{\Delta x} + O(\Delta x^2), \tag{2.29}$$

and

$$\frac{\partial F^n(i,j,k)}{\partial t} = \frac{F^{n+1/2}(i,j,k) - F^{n-1/2}(i,j,k)}{\Delta t} + O(\Delta t^2). \tag{2.30}$$

2.2.2 Finite Difference Scheme in Free Space

Since $D = \varepsilon_0 E$ and $B = \mu_0 H$ in free space, the time-dependent Maxwell's curl equations can be written as

$$\frac{\partial E}{\partial t} = \frac{1}{\varepsilon_0} \nabla \times H, \tag{2.31}$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\mu_0} \nabla \times E, \tag{2.32}$$

which represent a system of six scalar equations. We can express them in the rectangular coordinate system as follows:

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon_0}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}\right), \tag{2.33}$$

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu_0}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right), \tag{2.34}$$

with corresponding expressions for y and z directions.

Taking the central difference approximations, Equations (2.29) and (2.30), for both the temporal and spatial derivatives and using Δx for all the spatial increments we

can write down the explicit finite difference approximation of Equations (2.33) and (2.34) as follows:

$$E_x^{n+1}(i+1/2, j, k) = E_x^{n-1/2}(i+1/2, j, k) + \frac{\Delta t}{\varepsilon_0 \Delta x}[H_z^{n+1/2}(i+1/2, j+1/2, k) - H_z^{n+1/2}(i+1/2, j-1/2, k) + H_y^{n+1/2}(i+1/2, j, k-1/2) - H_y^{n+1/2}(i+1/2, j, k+1/2)], \quad (2.35)$$

and

$$H_x^{n+1/2}(i, j+1/2, k+1/2) = H_x^{n-1/2}(i, j+1/2, k+1/2) + \frac{\Delta t}{\mu_0 \Delta x}[E_y^n(i, j+1/2, k+1) - E_y^n(i, j+1/2, k) + E_z^n(i, j, k+1/2) - E_z^n(i, j+1, k+1/2)], \quad (2.36)$$

with corresponding expressions for y and z directions.

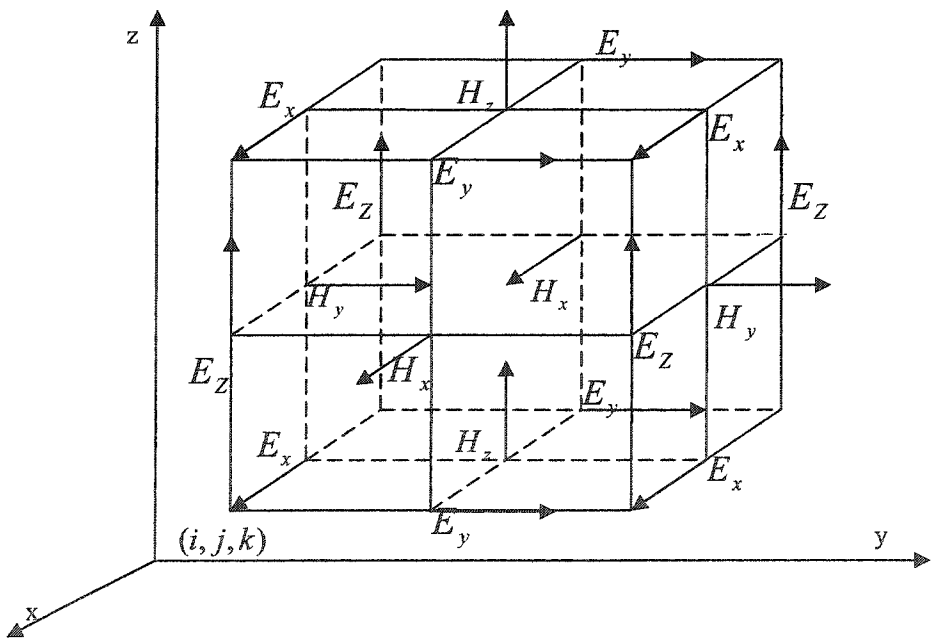


Figure 2.1 Positions of the field components in a Yee's Cell

Since the formulation of Equations (2.35) and (2.36) assumes that the E and H fields are interleaved in both time and space, we have to use Yee cell [9] to discretize the

components of E and H , as shown in figure 2.1. Notice that the calculations are arranged in alternative layers in both space and time. For example, the new value of E is calculated from the previous value of E and the most recent values of H. This is the fundamental paradigm of the FDTD method [9].

2.2.3 Stability

To ensure the accuracy of the computed results, the spatial increment Δx has to be small compared to the wavelength (usually $\leq \lambda/10$) or minimum dimension of the scatterer. To ensure the stability of the FDTD schemes, the temporal increment Δt must satisfy the following "Courant Condition" [6, 35]:

$$\Delta t \leq \frac{\Delta x}{\sqrt{n} \cdot c_0} \tag{2.37}$$

where c_0 is the speed of light in the free space and n is the number of space dimension. In order to simplify the problem, we determine Δt by

$$\Delta t = \frac{\Delta x}{2 \cdot c_0} \tag{2.38}$$

2.3 The Z-Transform

Z-transform method [28, 38, 39] is a tool to convert the equations from the frequency domain to the time domain. In this section we briefly introduced the z-transform method as described in [28].

2.3.1 Definition of the Z-Transform

When dealing with functions in the sampled time domain the z transform is extremely useful; i.e., we can rewrite a function $x(t)$ in the form of

$$x(t)=\sum_{n=0}^{\infty}x(n\cdot\Delta t)\delta(t-n\cdot\Delta t) \quad , \tag{2.39}$$

where Δt is a uniform time interval, and δ is the Dirac delta function:

$$\delta(t)=1 \qquad \text{if } t=0$$

$$\delta(t)=0 \qquad \text{elsewhere.} \tag{2.40}$$

Then the Z transform is defined by

$$Z[x(t)]=X(z)=\sum_{n=0}^{\infty}x(n\cdot\Delta t)z^{-n} \quad . \tag{2.41}$$

Example 1:

Suppose we have a function like

$$x(t)=\delta(t)+0.5\delta(t-\Delta t)+0.25\delta(t-2\cdot\Delta t) \tag{2.42}$$

which can be written in the z-domain as

$$X(z)=1+0.5z^{-1}+0.25z^{-2} \tag{2.43}$$

Notice that the first term is simply 1, because z^0=1.

In this form, the z^{-1} operator associated with each delay interval Δt and can be thought of as delay operator. Namely, if

$$y(t)=x(t-\Delta t), \tag{2.44}$$

then

$$y(t)=\delta(t-\Delta t)+0.5\delta(t-2\cdot\Delta t)+0.25\delta(t-3\cdot\Delta t). \tag{2.45}$$

In the z-domain, it can be written as

$$Y(z)=z^{-1}+0.5z^{-2}+0.25z^{-3} \quad . \tag{2.46}$$

Obviously, it means

$$Y(z)=z^{-1}X(z) \quad . \tag{2.47}$$

And if

$$w(t) = x(t - n \cdot \Delta t) \tag{2.48}$$

for the same reason, one may obtain

$$W(z) = z^{-n} X(z) \tag{2.49}$$

If $W(z)$ is defined as above, then

$$W(z) = z^{-n} + 0.5z^{-(n+1)} + 0.25z^{-(n+2)} \quad , \tag{2.50}$$

which, in turn, going back to time domain, means that

$$w(t) = \delta(t - n \cdot \Delta t) + 0.5\delta[t - (n+1) \cdot \Delta t] + 0.25\delta[t - (n+2) \cdot \Delta t] \quad . \tag{2.51}$$

We can add two z-transforms by lumping the same powers of z^{-1} together. For instance, adding $X(z)$ of Equation (2.43) and $Y(z)$ of Equation (2.46) gives

$$X(z) + Y(z) = 1 + 1.5z^{-1} + 0.75z^{-2} + 0.25z^{-3} \tag{2.52}$$

Going back to the time domain, we obtain

$$x(t) + y(t) = \delta(t) + 1.5\delta(t - \Delta t) + 0.75\delta(t - 2 \cdot \Delta t) + 0.25\delta(t - 3 \cdot \Delta t) \ , \tag{2.53}$$

which could be obtained by adding $x(t)$ of Equation (2.42) and $y(t)$ of Equation (2.45) together, being sure to keep like delta terms together.

2.3.2 Convolution Using the Z-Transform

Starting with the definition of convolution in the discrete time domain,

$$y(t) = \sum_{i=0}^{\infty} h(t - i\Delta t) x(i\Delta t), \tag{2.54}$$

where $h(t)$ is a casual function which is zero for t less than zero.

We take the Z transform on both sides of Equation (2.54)

$$\sum_{n=0}^{\infty} y(n\Delta t) z^{-n} = \sum_{n=0}^{\infty} \sum_{i=0}^{\infty} h(n\Delta t - i\Delta t) x(i\Delta t) z^{-n} \ , \tag{2.55}$$

and then interchange the summation signs:

$$Y(z) = \sum_{i=0}^{\infty} x(i\Delta t) \sum_{n=0}^{\infty} h(n\Delta t - i\Delta t) z^{-n} \quad . \tag{2.56}$$

Multiplying by $z^{-i} \cdot z^{i}$ gives

$$Y(z) = \sum_{i=0}^{\infty} x(i\Delta t) z^{-i} \sum_{n=0}^{\infty} h(n\Delta t - i\Delta t) z^{-(n-i)} \, . \tag{2.57}$$

Using the parameter $m = n - i$ results in

$$Y(z) = \sum_{i=0}^{\infty} x(i\Delta t) z^{-i} \sum_{m=0}^{\infty} h(m\Delta t) z^{-m} \, , \tag{2.58}$$

which means

$$Y(z) = H(z) \cdot X(z) \, . \tag{2.59}$$

(Notice that m starts from zero, because $h(m\Delta t)$ is zero for the values of m less than zero.) This illustrates the convolution theorem for discrete functions: **convolution in the discrete time domain becomes multiplication in the z-domain.**

While dealing with the continuous functions, the convolution in the time domain is

$$y(t) = \int_{0}^{\infty} h(t - \tau) x(\tau) d\tau \tag{2.60}$$

where $h(t)$ is casual. The integral in Equation (2.60) can be approximated by

$$y(t) \cong \Delta t \sum_{i=0}^{\infty} h(t - i\Delta t) x(i\Delta t) \tag{2.61}$$

Taking the Z transform on both sides

$$\sum_{n=0}^{\infty} y(n\Delta t) z^{-n} = \Delta t \sum_{n=0}^{\infty} \sum_{i=0}^{\infty} h(n\Delta t - i\Delta t) x(i\Delta t) z^{-n} \quad . \tag{2.62}$$

Identical to the previous derivation, one may obtain

$$Y(z) = \Delta t H(z) \cdot X(z) \quad . \tag{2.63}$$

Therefore, the convolution theorem for continuous functions is: **convolution in the continuous time domain is multiplication in the z-domain and an extra Δt.**

Example 2:

Suppose an exponentially decaying function:

$$h(t) = \sum_{n=0}^{\infty} e^{-n\Delta t / t_0} \delta(t - n\Delta t) \qquad n = 0,1,2,3... \tag{2.64}$$

and a discretized unit step function

$$u(t) = \delta(t - n\Delta t) \qquad n = 0,1,2,3... \tag{2.65}$$

Since

$$\sum_{n=0}^{\infty} a^n = \frac{1}{1-a} \quad \text{when} \quad |a| < 1, \tag{2.66}$$

$H(z)$ can be calculated as follows:

$$H(z) = \sum_{n=0}^{\infty} e^{-n\Delta t / t_0} z^{-n} = \sum_{n=0}^{\infty} (e^{-\Delta t / t_0} z^{-1})^n = \frac{1}{1 - e^{-\Delta t / t_0} z^{-1}} \tag{2.67}$$

and similarly

$$U(z) = \frac{1}{1 - z^{-1}} \quad . \tag{2.68}$$

Then the desired convolution is

$$Y(Z) = H(z) \cdot U(z) = \frac{1}{1 - (1 + e^{-\Delta t / t_0}) z^{-1} + e^{-\Delta t / t_0} z^{-2}} \quad . \tag{2.69}$$

To get a solution in the time domain, we take the partial fraction expansion of $Y(z)$

$$Y(z) = \frac{A}{1 - e^{-\Delta t / t_0} z^{-1}} + \frac{B}{1 - z^{-1}} \tag{2.70}$$

where $A = -\frac{e^{-\Delta t/t_0}}{1-e^{-\Delta t/t_0}}$ and $B = \frac{1}{1-e^{-\Delta t/t_0}}$.

Since

$$Y(z) = \sum_{n=0}^{\infty} \frac{-e^{-\Delta t/t_0}}{1-e^{-\Delta t/t_0}} e^{-n\Delta t/t_0} z^{-n} + \sum_{n=0}^{\infty} \frac{1}{1-e^{-\Delta t/t_0}} z^{-n} \quad , \tag{2.71}$$

from which, going back to the time domain, one may obtain

$$y(t) = \sum_{n=0}^{\infty} \frac{-e^{-\Delta t/t_0}}{1-e^{-\Delta t/t_0}} e^{-n\Delta t/t_0} \delta(t-n\Delta t) + \sum_{n=0}^{\infty} \frac{1}{1-e^{-\Delta t/t_0}} \delta(t-n\Delta t)$$

$$= \sum_{n=0}^{\infty} \frac{1-e^{-(n+1)\Delta t/t_0}}{1-e^{-\Delta t/t_0}} \delta(t-n\Delta t)$$

$$= \frac{1-e^{-(n+1)\Delta t/t_0}}{1-e^{-\Delta t/t_0}} \qquad n = 0,1,2,... \tag{2.72}$$

Or we can obtain Equation (2.72) using the convolution theorem for discrete functions,

$$y(t) = \sum_{i=0}^{\infty} h(t-i\Delta t)u(i\Delta t) = \sum_{i=0}^{n} h(n\Delta t - i\Delta t) = \sum_{i=0}^{n} e^{-(n-i)\Delta t/t_0} = \frac{1-e^{-(n+1)\Delta t/t_0}}{1-e^{-\Delta t/t_0}} \tag{2.73}$$

On the other hand, we could simply get the solution from Equation (2.69)

$$Y(z) = 1 + (1+e^{-\Delta t/t_0}) z^{-1} Y(z) - e^{-\Delta t/t_0} z^{-2} Y(z) \tag{2.74}$$

which, in the time domain, means

$$y(t) = 1 + (1+e^{-\Delta t/t_0}) y(t-\Delta t) - e^{-\Delta t/t_0} y(t-2\Delta t) \quad . \tag{2.75}$$

2.3.3 Alternative Methods to Formulate the Z-Transform

We can solve the problems stated in the frequency domain in the time domain. Our approach is to take the partial fraction expansion of the frequency domain expression, find the corresponding Z transforms, and solve the problems in the z-domain.

Our success depends on the ability to manipulate the frequency domain expression that can be found in a table like Table 2.1 as follows:

Table 2.1 Transform among the Time, Frequency, and Z-Domain [28]

Time Domain	Frequency Domain	Z Domain
$\delta(t)$	1	1
$u(t)$	$\frac{1}{j\omega}$	$\frac{1}{1-z^{-1}}$
$tu(t)$	$\frac{1}{(j\omega)^2}$	$\frac{z^{-1}}{(1-z^{-1})^2}$
$e^{-\alpha t}u(t)$	$\frac{1}{\alpha + j\omega}$	$\frac{1}{1-z^{-1}e^{-\alpha\Delta t}}$

However, for some expressions like the Cole-Cole expression, Equation (1.1), it is difficult to find out the corresponding Z transform directly, Thus we must find an alternative approximation to solve this problem.

Fourier transform theory tells us that a multiplication by $j\omega$ in the frequency domain becomes a derivative in time domain:

Let

$$f(t) = \int F(\omega)e^{j\omega t}d\omega, \tag{2.76}$$

then

$$f'(t) = j\omega\int F(\omega)e^{j\omega t}d\omega = j\omega f(t), \tag{2.77}$$

which can be approximated by:

$$f'(t) \cong \frac{f(t) - f(t - \Delta t)}{\Delta t} \quad . \tag{2.78}$$

Taking the Z transform on Equations. (2.77) and (2.78) gives:

$$Z(f'(t)) = j\omega F(z) \tag{2.79}$$

and

$$Z(f'(t)) \cong \frac{Z(f(t)) - Z(f(t - \Delta t))}{\Delta t} = \frac{1 - z^{-1}}{\Delta t} F(z) \quad . \tag{2.80}$$

Thus, the transition from the frequency domain to the z-domain is made by simply making the replacement

$$j\omega \Rightarrow \frac{1 - z^{-1}}{\Delta t} \quad . \tag{2.81}$$

As an example, for equation $e^{-\alpha t}u(t)$, the transition from the frequency domain to the z-domain becomes

$$\frac{1}{\alpha + j\omega} \Rightarrow \frac{1}{\alpha + \dfrac{1 - z^{-1}}{\Delta t}} = \frac{\Delta t}{\alpha \Delta t + 1 - z^{-1}} \tag{2.82}$$

Utilizing an approximation

$$\frac{1}{1 + \mu} \cong e^{-\mu} \qquad \text{for} \quad \mu << 1, \tag{2.83}$$

Equation (2.82) becomes

$$\frac{\Delta t}{\alpha \Delta t + 1 - z^{-1}} = \frac{\Delta t / (1 + \alpha \Delta t)}{1 - z^{-1} / (1 + \alpha \Delta t)} \cong \frac{\Delta t e^{-\alpha \Delta t}}{1 - z^{-1} e^{-\alpha \Delta t}} \quad . \tag{2.84}$$

The corresponding item in the table 2.1 is

$$\frac{1}{1 - z^{-1} e^{-\alpha \Delta t}} \quad . \tag{2.85}$$

Comparing it with Equation (2.84), we find when $\Delta t \to 0$, Equation (2.84) approximates Equation (2.85) multiplied by Δt because the convolution theorem for continuous functions has been taken into account. As a matter of fact, the factor Δt, has to be added when we make a transition from frequency domain to z-domain for a continuous function.

2.4 Dielectric Properties of Biological Tissues

Propagation of electromagnetic waves in materials such as dielectrics and conductors is determined by their electrical parameters. In the case of dielectrics, chief among these is the complex permittivity, ε, of the dielectric material. It is normal to refer to the relative permittivity, ε_r, of a dielectric as being its permittivity with respect to that of free space, ε_0, such that:

$$\varepsilon = \varepsilon_0 \varepsilon_r \tag{2.86}$$

The relative permittivity of a dielectric is defined as the factor by which the capacitance of a capacitor increases when the volume between and around its plates is filled with dielectric as compared with free space. It is known that the permittivity of a dielectric is determined by its molecular/atomic structure but no theory exists to relate the two. It is also known that permittivity is often frequency and temperature dependent, since certain phenomena which determine its permittivity are functions of frequency and temperature. The orientation of polar molecules changes in sympathy with an applied radio frequency electric field. This phenomenon has a significant effect in determining the permittivity of the material. The ability of the polar molecules to align with the applied electric field at radio frequency is determined by the kinematics of the molecular

structure and is described by relaxation theory.

The main features of the dielectric spectrum of tissues have been reviewed and reported by Foster and Schwan [36]. Their theoretical analysis is characterized by a single relaxation process centered around a single relaxation time constant.

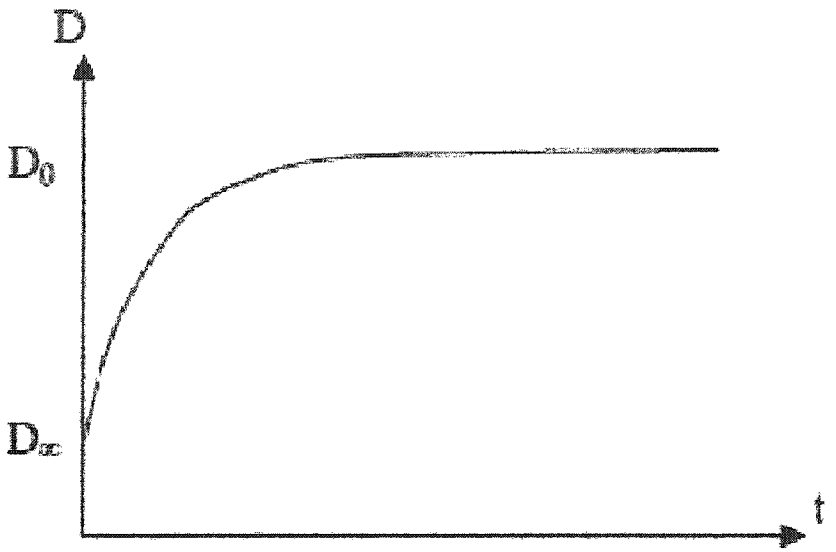


Figure 2.2 Transient response of polar dielectric.

In the simplest case, the polarization of a sample will relax towards the steady state as a first-order process characterized by a single time constant, τ. Thus the transient response looks like Figure 2.2, which has the form:

$$D = D_{\infty} + (D_0 - D_{\infty})(1 - e^{-t/\tau}), \tag{2.87}$$

where D_0 is the final value of D and D_{∞} is the initial value of D. Since $D = \varepsilon_r \varepsilon_0 E$, $D_{\infty} = \varepsilon_{\infty} \varepsilon_0 E$ and $D_0 = \varepsilon_s \varepsilon_0 E$ where ε_{∞} is relative permittivity at infinite frequency ($\omega\tau >> 1$) and ε_s static relative permittivity ($\omega\tau << 1$), we can rewrite Equation (2.87) in terms of permittivity:

$$\varepsilon_r \varepsilon_0 E = \varepsilon_\infty \varepsilon_0 E + (\varepsilon_s \varepsilon_0 E - \varepsilon_\infty \varepsilon_0 E)(1 - e^{-t/\tau}) \ . \tag{2.88}$$

Canceling $\varepsilon_0 E$ both sides leaves

$$\varepsilon_r = \varepsilon_\infty + (\varepsilon_s - \varepsilon_\infty) - (\varepsilon_s - \varepsilon_\infty) e^{-t/\tau} \ . \tag{2.89}$$

Taking Laplace transforms on both sides to convert to the frequency domain and simplify, one may obtain

$$\varepsilon_r = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + s\tau} \ . \tag{2.90}$$

Setting $s = j\omega$ so as to transform back to the frequency domain gives

$$\varepsilon_r = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + j\omega\tau} \ , \tag{2.91}$$

which is a single relaxation Debye equation. This equation exhibits a relaxation frequency centered on $f_c = 1/2\pi\tau$, and strictly, ε_r and ε_∞ refer to the relative permittivity well below and well above f_c respectively. The magnitude of the dispersion is described as $\Delta\varepsilon = \varepsilon_s - \varepsilon_\infty$. Equation (2.91) omits the currents flowing at infinite time such as would arise due to the movement of ions in a constant field. The model is expanded to include a static conductivity term, σ_s, where $\sigma_s = j\omega\varepsilon_0\varepsilon_s$. Including this term results in

$$\varepsilon_r = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + j\omega\tau} + \frac{\sigma_s}{j\omega\varepsilon_0} \ . \tag{2.92}$$

The dielectric spectrum of a tissue is characterized by several relaxation regions each of which is the manifestation of a polarization mechanism characterized by a single time constant, τ , namely, every relaxation region corresponds to a single Debye expression. Thus, we can model the dielectric spectrum of tissues with the summation of

Debye dispersions in addition to a conductivity term [23]:

$$\varepsilon_r(\omega) = \varepsilon_\infty + \sum_{m=1}^{N} \frac{\Delta\varepsilon_m}{1 + j\omega\tau_m} + \frac{\sigma_s}{j\omega\varepsilon_0}, \tag{2.93}$$

where N is the number of relaxation regions.

However, both of the structure and composition of biological material are so complicated that each dispersion region may be broadened by multiple contributions to it. This effect could be empirically accounted for by introducing a distribution parameter, thus giving an alternative to the Debye model known as the Cole-Cole expression [17]:

$$\varepsilon_r(\omega) = \varepsilon_\infty + \sum_{m=1}^{4} \frac{\Delta\varepsilon_m}{1 + (j\omega\tau_m)^{1-\alpha_m}} + \frac{\sigma_s}{j\omega\varepsilon_0}, \tag{2.94}$$

where the adjustable parameter α ,between 0 and 1, is a measure of the broadening of the dispersion. This model is based on the well-known dispersive properties of biological matter and their expression as a summation of terms corresponding to the main polarization mechanisms [7]. The dielectric spectrum extends from Hz to GHz and shows four major regions of dispersion. With a choice of parameters appropriate to each tissue, Equation (2.23) can be used to predict its dielectric behavior over the desired frequency range [19, 20].

2.5 Conclusion

As we discussed in the last section, the Cole-Cole expression could describe the broadening of the dispersion so well that it is commonly employed to model the frequency dependence of dielectric properties of a biological tissue. However, it is difficult to solve the Maxwell's equations when coupled to the Cole-Cole expression because of the difficulty to convert the equations from the frequency domain to the time

domain. Therefore, the z-transform is used to simplify the problem. The Cole-Cole expression is first transformed from the frequency domain to the z-domain using the z-transform method and then approximated by a second-order Taylor series of variable z. After that, the Cole-Cole expression in the z-domain is transformed to the time domain. Finally, FDTD is employed to solve Maxwell's equations.

CHAPTER 3

GOVERNING EQUATIONS

3.1 Governing Equations for the Study of Bioeffects

In our research we assume all the biological tissues being simulated are nonmagnetic: that is, $H = B/\mu_0$, and the dielectric properties of them are isotropic but frequency dependent. Therefore, the governing equations in our research are the "normalized" Maxwell's equations in a frequency dependent medium as follows [28]:

$$\frac{\partial \tilde{D}}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \nabla \times H , \tag{3.1}$$

$$\tilde{D}(\omega) = \varepsilon_r(\omega)\tilde{E}(\omega), \tag{3.2}$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\sqrt{\mu_0 \varepsilon_0}} \nabla \times \tilde{E} . \tag{3.3}$$

where $\tilde{D}(\omega) = \sqrt{1/(\varepsilon_0 \mu_0)} D(\omega)$ is the "normalized" electric flux density, $\tilde{E}(\omega) = \sqrt{\varepsilon_0 / \mu_0} E(\omega)$ is the "normalized" the electric density. In order to simplify the notation, we still use D and E instead of $\tilde{D}$ and $\tilde{E}$ respectively in the later discussion.

Equations (3.1) - (3.3) produce 9 scalar equations as follows:

$$\frac{\partial D_x}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \right), \tag{3.4}$$

$$\frac{\partial H_x}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right), \tag{3.5}$$

$$\frac{\partial D_y}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x}\right), \tag{3.6}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right), \tag{3.7}$$

$$\frac{\partial D_z}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right), \tag{3.8}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right), \tag{3.9}$$

$$D_x(\omega) = \varepsilon_r(\omega) E_x(\omega), \tag{3.10}$$

$$D_y(\omega) = \varepsilon_r(\omega) E_y(\omega), \tag{3.11}$$

$$D_z(\omega) = \varepsilon_r(\omega) E_z(\omega). \tag{3.12}$$

There are 9 components of D, E and H in Equations (3.4)-(3.12). In one-dimensional space, we use D_z, E_z and H_x with propagation in the y direction and let rest of the components be zero. In two-dimensional space, there are two groups of vectors for us to choose: (1) the transverse magnetic (TM) mode composed of D_z, E_z, H_x and H_y, or (2) the transverse electric (TE) mode consisting of D_x, E_x, D_y, E_y, and H_z. Interested in E_z field, we work with TM mode and set the rest to be zero. In three- dimensional space, we use all of the 9 components. The systematic interleaving of fields is illustrated in the Yee cell (Figure 2.1).

3.2 Difficulty in Solving the System

The relative dielectric constant, $\varepsilon_r(\omega)$, appearing in the governing equation (3.2) is usually described by the Cole-Cole expression as follows:

$$\varepsilon_r(\omega) = \varepsilon_\infty + \sum_{m=1}^{4} \frac{\Delta\varepsilon_m}{1 + (j\omega\tau_m)^{1-\alpha_m}} + \frac{\sigma_s}{j\omega\varepsilon_0}. \tag{3.13}$$

Notice that the parameters α_m are non-integer which makes it extremely difficult to convert the Cole-Cole expression from frequency domain to the time domain. The usual way to overcome this difficulty is to approximate the Cole-Cole expression by a Debye model or a Lorentz model [26] as follows:

$$\varepsilon_r^*(\omega) = \varepsilon_\infty + \sum_{m=1}^{N} \frac{\Delta\tilde{\varepsilon}_m}{1 + j\omega\tilde{\tau}_m} + \frac{\tilde{\sigma}_s}{j\omega\varepsilon_0}, \tag{3.14}$$

or

$$\varepsilon_r^*(\omega) = \varepsilon_\infty + \sum_{m=1}^{N} \frac{\Delta\tilde{\varepsilon}_m\omega}{\omega_m^2 + 2j\omega\tilde{\tau}_m - \omega^2} + \frac{\tilde{\sigma}_s}{j\omega\varepsilon_0}. \tag{3.15}$$

However, $\Delta\tilde{\varepsilon}_m$, $\tilde{\tau}_m$ and $\tilde{\sigma}_s$ are different from $\Delta\varepsilon_m$, τ_m and σ_s, namely, we have to recalculate these parameters, which is a large amount of work. After that, the Debye model or the Lorentz model is transformed from the frequency domain to the time domain using the inverse Fourier transform.

In this study, we develop a new approach by transforming the Cole-Cole expression to the z-domain using the Taylor series in z. The advantage of this approach is that the values of $\Delta\varepsilon_m$, τ_m and σ_s can be used directly. z-transform method and then approximating the result using a second-order

3.3 Source Term

The source term in our simulation is a short duration, fast rise time ultra-wide-band (UWB) electromagnetic pulse, nanopulse, characterized by the pulse widths of less than a few nanoseconds and extremely large electric field amplitudes greater than $10^5 V/m$. In our research, we use the following Gaussian pulse to model a nanopulse:

$$E(t) = Ae^{-(c-t)^2/w^2} V/m, \tag{3.16}$$

where A is the amplitude, c is the center of the pulse, and w is the pulse width. The energy stored in an electric field is calculated as follows:

$$Energy = \varepsilon_0 c_0 \int_0^T E^2(t)dt, \tag{3.17}$$

where c_0 is the speed of light in the free space. Equation (3.16) is used as a source term in our simulation later on.

CHAPTER 4

NUMERICAL SIMULATION

4.1 Finite Difference Scheme

In our simulation, all the spatial increment Δx, Δy and Δz are assumed the same size. To simplify the notation, we replace Δy and Δz by Δx, because Δx is so commonly used for a spatial increment. Once Δx is chosen, as we mentioned in Chapter 2, the time step is determined by $\Delta t = \frac{\Delta x}{2 \cdot c_0}$ to ensure the accuracy and stability. Since the speed of light $c_0 = 1/\sqrt{\varepsilon_0 \mu_0}$ in the free space, we obtain

$$\frac{1}{\sqrt{\varepsilon_0 \mu_0}} \frac{\Delta t}{\Delta x} = \frac{1}{2}. \tag{4.1}$$

Taking the central difference approximations for both the temporal and spatial derivatives in the govern equations (3.4) - (3.9) and using equation (4.1) gives

$$\begin{aligned} & D_x^{n+1/2}(i+1/2, j, k) = D_x^{n-1/2}(i+1/2, j, k) + \\ & \frac{1}{2} \cdot [H_z^n(i+1/2, j+1/2, k) - H_z^n(i+1/2, j-1/2, k) \\ & - H_y^n(i+1/2, j, k+1/2) + H_y^n(i+1/2, j, k-1/2)], \end{aligned} \tag{4.2}$$

$$\begin{aligned} & D_y^{n+1/2}(i, j+1/2, k) = D_y^{n-1/2}(i, j+1/2, k) + \\ & \frac{1}{2} \cdot [H_x^n(i, j+1/2, k+1/2) - H_x^n(i, j+1/2, k-1/2) \\ & - H_z^n(i+1/2, j+1/2, k) + H_z^n(i-1/2, j+1/2, k], \end{aligned} \tag{4.3}$$

$$D_z^{n+1/2}(i,j,k+1/2) = D_z^{n-1/2}(i,j,k+1/2) + \frac{1}{2}\cdot[H_y^n(i+1/2,j,k+1/2) - H_y^n(i-1/2,j,k+1/2) - H_x^n(i,j+1/2,k+1/2) + H_x^n(i,j-1/2,k+1/2)], \quad (4.4)$$

$$H_x^{n+1}(i,j+1/2,k+1/2) = H_x^n(i,j+1/2,k+1/2) + \frac{1}{2}\cdot[E_y^{n+1/2}(i,j+1/2,k+1) - E_y^{n+1/2}(i,j+1/2,k) - E_z^{n+1/2}(i,j+1,k+1/2) + E_z^{n+1/2}(i,j,k+1/2)], \quad (4.5)$$

$$H_y^{n+1}(i+1/2,j,k+1/2) = H_y^n(i+1/2,j,k+1/2) + \frac{1}{2}\cdot[E_z^{n+1/2}(i+1,j,k+1/2) - E_z^{n+1/2}(i,j,k+1/2) - E_x^{n+1/2}(i+1/2,j,k+1) + E_x^{n+1/2}(i+1/2,j,k)], \quad (4.6)$$

$$H_z^{n+1}(i+1/2,j+1/2,k) = H_z^n(i+1/2,j+1/2,k) + \frac{1}{2}\cdot[E_x^{n+1/2}(i+1/2,j+1,k) - E_x^{n+1/2}(i+1/2,j,k) - E_y^{n+1/2}(i+1,j+1/2,k) + E_y^{n+1/2}(i,j+1/2,k)]. \quad (4.7)$$

The E and H fields are assumed interleaved around the Yee cell (figure 2.1) whose origin is at the location i,j,k. Every E field is located ½ cell width from the origin in direction of its orientation while every H field is offset ½ cell width in each direction except that of its orientation. The values of E and H are calculated by separate loops. and they employ the interleaving described above. We choose corresponding components of the fields for 1D, 2D and 3D as described in chapter 3.

4.2 Approximation of Dielectric Constant

The relative dielectric constant, $\varepsilon_r(\omega)$, of biological tissues vary at different frequencies and can be commonly modeled by the Cole-Cole expression, Equation (3.13). The usual way to approximate the Cole-Cole expression is using a Debye model or a

Lorentz model, Equations (3.14) or (3.15), which needs a large amount work of reparameterization.

Our alternative approach, in this research, is transforming the Cole-Cole expression to the z-domain using the z-transform method and then using a second-order Taylor approximation of the Cole-Cole expression to convert from the frequency domain to the time domain.

In this section, instead of discussing the reparameterization process described in [40], we show how the Debye model is transformed from frequency domain to the time domain using the Fourier theory at first, then followed by a z-transform method. After the discussion of the Debye model, the transformation of the Cole-Cole model from the frequency domain to the time domain using the z-transform method is presented in detail.

4.2.1 Formulating the Debye Model

Assuming a material that can be adequately represented by the following Debye formulation:

$$\varepsilon_r = \varepsilon_\infty + \frac{\chi}{1 + j\omega\tau} + \frac{\sigma}{j\omega\varepsilon_0}, \tag{4.8}$$

we substitute Equation (4.8) into Equation (3.2), which gives:

$$D(\omega) = \varepsilon_\infty E(\omega) + \frac{\chi}{1 + j\omega\tau} E(\omega) + \frac{\sigma}{j\omega\varepsilon_0} E(\omega). \tag{4.9}$$

To simulate this medium in FDTD, Equation (4.9) must be put into time domain. There are two ways to solve this problem. The first one is using Fourier transform, which is the traditional method. The second one is using a much easier Z transform. We will discuss both of the two methods in the following paragraphs.

First, we start with Fourier Transform. Let us define the last two terms of Equation (4.9) as

$$S(\omega) = \frac{\chi}{1 + j\omega\tau} E(\omega) \,, \tag{4.10}$$

$$I(\omega) = \frac{\sigma}{j\omega\varepsilon_0} E(\omega) \,. \tag{4.11}$$

In the first term, Equation (4.10), the inverse Fourier transform of $\frac{\chi}{1 + j\omega\tau}$ is $(\chi/\tau)e^{-(t/\tau)}u(t)$, where $u(t)$ is the Heavyside function, which is 0 for $t < 0$ and 1 thereafter. Equation (4.10) in the frequency domain becomes the convolution:

$$S(t) = \frac{\chi}{\tau} \int e^{-(t-t')/\tau} E(t') \cdot dt' \tag{4.12}$$

in the time domain. Approximating this gives

$$S^n = \frac{\chi\Delta t}{\tau} \sum_{i=0}^{n} [e^{-\Delta t(n-i)/\tau} \cdot E^i]$$

$$= \frac{\chi\Delta t}{\tau} \{E^n + \sum_{i=0}^{n-1} [e^{-\Delta t(n-i)/\tau} \cdot E^i]\} \,. \tag{4.13}$$

Note that

$$S^{n-1} = \frac{\chi\Delta t}{\tau} \sum_{i=0}^{n-1} [e^{-\Delta t(n-1-i)/\tau} \cdot E^i]$$

$$= \frac{\chi\Delta t}{\tau} e^{\Delta t/\tau} \sum_{i=0}^{n-1} [e^{-\Delta t(n-1)/\tau} \cdot E^i] \,. \tag{4.14}$$

Combining these two equations gives

$$S^n = \frac{\chi\Delta t}{\tau} E^n + e^{-\Delta t/\tau} S^{n-1} \,. \tag{4.15}$$

In the latter term, Equation (4.11), Fourier theory tells us that $1/j\omega$ in the frequency

domain is integration in the time domain, so Equation (4.11) becomes

$$I(t) = \frac{\sigma}{\varepsilon_0} \int_0 E(t')dt' \ . \tag{4.16}$$

Similarly, it can be approximated as a summation over the time steps.

$$I^n = \frac{\sigma \Delta t}{\varepsilon_0} \sum_{i=0}^{n} E^i$$

$$= \frac{\sigma \Delta t}{\varepsilon_0} (E^n + \sum_{i=0}^{n-1} E^i)$$

$$= \frac{\sigma \Delta t}{\varepsilon_0} E^n + I^{n-1} . \tag{4.17}$$

Going back to the time domain, Equation (4.9) can be reformulated as

$$D(t) = \varepsilon_\infty E(t) + S(t) + I(t) . \tag{4.18}$$

We can write it in the sampled time domain as follows:

$$D^n = \varepsilon_\infty E^n + S^n + I^n$$

$$= \varepsilon_\infty E^n + (\frac{\chi \Delta t}{\tau} E^n + e^{-\Delta t/\tau} S^{n-1}) + (\frac{\sigma \Delta t}{\varepsilon_0} E^n + I^{n-1}) . \tag{4.19}$$

Solving for E^n in Equation (4.19), one may obtain

$$E^n = \frac{D^n - I^{n-1} - e^{-\Delta t/\tau} S^{n-1}}{\varepsilon_\infty + \frac{\sigma \Delta t}{\varepsilon_0} + \frac{\chi \Delta t}{\tau}} , \tag{4.20}$$

$$S^n = e^{-\Delta t/\tau} S^{n-1} + \frac{\chi \Delta t}{\tau} E^n , \tag{4.21}$$

$$I^n = I^{n-1} + \frac{\sigma \Delta t}{\varepsilon_0} E^n . \tag{4.22}$$

Then, we will show the advantage of using Z transform for the FDTD formulation of the frequency dependent media. In order to avoid dealing with troublesome

convolution integrals in the time domain, Equation (4.9) can be immediately written in the z-domain. According to Table 2.1 in Chapter 2, taking the Z transform on terms $\frac{\chi}{1+j\omega\tau}$ and $\frac{\sigma}{j\omega\varepsilon_0}$, we obtain:

$$\frac{\chi}{1+j\omega\tau} = \frac{\chi/\tau}{1/\tau + j\omega} \Rightarrow \frac{\chi/\tau}{1-z^{-1}e^{-\Delta t/\tau}}, \tag{4.23}$$

$$\frac{\sigma}{j\omega\varepsilon_0} = \frac{\sigma/\varepsilon_0}{j\omega} \Rightarrow \frac{\sigma/\varepsilon_0}{1-z^{-1}}. \tag{4.24}$$

Notice that $\frac{\chi}{1+j\omega\tau}\cdot E(\omega)$ in the frequency domain is the convolution in the time domain, on the other hand, $\frac{\chi/\tau}{1-z^{-1}e^{-\Delta t/\tau}}\cdot E(z)\cdot\Delta t$ is also the convolution in the time domain because of the convolution theory of Z transform described in section 2.3.2. Thus, we can write $S(\omega)$, Equation (4.10), in the z-domain as

$$S(z) = \frac{\chi/\tau}{1-z^{-1}e^{-\Delta t/\tau}}\cdot E(z)\cdot\Delta t. \tag{4.25}$$

Similarly, $I(\omega)$, Equation (4.11), can be written in the z-domain as

$$I(z) = \frac{\sigma/\varepsilon_0}{1-z^{-1}}\cdot E(z)\cdot\Delta t. \tag{4.26}$$

Rearranging these two equations gives

$$S(z) = \chi/\tau\cdot E(z)\cdot\Delta t + e^{-\Delta t/\tau}z^{-1}S(z), \tag{4.27}$$

$$I(z) = \sigma/\varepsilon_0\cdot E(z)\cdot\Delta t + z^{-1}I(z). \tag{4.28}$$

Equation (4.9) in the z-domain then becomes

$$D(z) = \varepsilon_\infty E(z) + [\frac{\chi\Delta t}{\tau}E(z) + e^{-\Delta t/\tau}z^{-1}S(z)] + [\frac{\sigma\Delta t}{\varepsilon_0}E(z) + z^{-1}I(z)], \tag{4.29}$$

from which we solve $E(z)$ by

$$E(z) = \frac{D(z) - z^{-1}I(z) - e^{-\Delta t/\tau} z^{-1} S(z)}{\varepsilon_\infty + \frac{\sigma \Delta t}{\varepsilon_0} + \frac{\chi \Delta t}{\tau}}. \tag{4.30}$$

Here is the advantage of the Z transform: to get to the sampled time domain, replace $I(z)$ with I^n, $z^{-1}I(z)$ with I^{n-1}, and make a similar replacement with the other parameters. What we get is

$$E^n = \frac{D^n - I^{n-1} - e^{-\Delta t/\tau} S^{n-1}}{\varepsilon_\infty + \frac{\sigma \Delta t}{\varepsilon_0} + \frac{\chi \Delta t}{\tau}}, \tag{4.31}$$

$$S^n = e^{-\Delta t/\tau} S^{n-1} + \frac{\chi \Delta t}{\tau} E^n, \tag{4.32}$$

$$I^n = I^{n-1} + \frac{\sigma \Delta t}{\varepsilon_0} E^n, \tag{4.33}$$

which is exactly the same as what we got using the previous method. The difference is we didn't have to do anything with integrals and approximations, which makes the advantage of Z transform evident when dealing with complicated formulations like Cole-Cole expression.

4.2.2 Formulating the Cole-Cole Model

We said in the second chapter that most of the biological materials are modeled by an empirical Cole-Cole expression, Equation (2.94), as follows:

$$\varepsilon_r(\omega) = \varepsilon_\infty + \sum_{m=1}^{4} \frac{\Delta \varepsilon_m}{1 + (j\omega\tau_m)^{1-\alpha_m}} + \frac{\sigma_s}{j\omega\varepsilon_0}.$$

Substituting it into Equation (3.2), we obtain

$$D(\omega) = \varepsilon_\infty E(\omega) + \sum_{m=1}^{4} \frac{\Delta\varepsilon_m E(\omega)}{1 + (j\omega\tau_m)^{1-\alpha_m}} + \frac{\sigma_s E(\omega)}{j\omega\varepsilon_0}. \qquad (4.34)$$

Obviously, it is difficult to make a transition directly from the frequency domain to the time domain because of the exponent $1-\alpha_m$, but we can make a transition from the frequency domain to the z-domain, and then go back to the time domain from the z-domain.

Similar to the previous section, we let

$$I(\omega) = \frac{\sigma_s E(\omega)}{j\omega\varepsilon_0}. \qquad (4.35)$$

and

$$S_m(\omega) = \frac{\Delta\varepsilon_m E(\omega)}{1+(j\omega\tau_m)^{1-\alpha_m}}, \quad m = 1,2,3,4. \qquad (4.36)$$

In the z- transform theorem discussed in section 2.3.3, we know that $j\omega$ can be replaced by $\frac{1-z^{-1}}{\Delta t}$, furthermore, the factor Δt that we usually add in the convolution theorem is already in this approximation, so that in the z-domain we don't need to put an extra Δt to correspond to the convolution in the time domain. Thus, applying the z-transform to Equation (4.35) and (4.36) we obtain

$$I(z) = \frac{\sigma_s / \varepsilon_0}{1-z^{-1}} \cdot E(z) \cdot \Delta t, \qquad (4.37)$$

and

$$S_m(z) = \frac{\Delta\varepsilon_m E(z)}{1+(\frac{\tau_m}{\Delta t})^{1-\alpha_m}(1-z^{-1})^{1-\alpha_m}}, \quad m = 1,2,3,4. \qquad (4.38)$$

From Equation (4.37) and (4.38), we have

$$I(z) = \frac{\sigma_s \Delta t}{\varepsilon_0} \cdot E(z) + z^{-1} I(z) \ , \tag{4.39}$$

and

$$S_m(z)[1 + (\frac{\tau_m}{\Delta t})^{1-\alpha_m}(1 - z^{-1})^{1-\alpha_m}] = \Delta\varepsilon_m E(z) \ , m = 1,2,3,4. \tag{4.40}$$

It is noteworthy that if α_m is not 0 or 1, then powers of z in Equation (4.40) are not integers. This parameter complicates determination of the time steps when Equation (4.40) is converted back to the time domain.

The situation is simplified by employing a second-order Taylor approximation as follows:

$$(1 - z^{-1})^{1-\alpha_m} \cong 1 - (1 - \alpha_m) z^{-1} - \frac{1}{2}(1 - \alpha_m)\alpha_m z^{-2} . \tag{4.41}$$

Substituting Equation (4.41) into Equation (4.40) and rearranging terms, we obtain

$$S_m(z) = \frac{(\frac{\tau_m}{\Delta t})^{1-\alpha_m}}{1 + (\frac{\tau_m}{\Delta t})^{1-\alpha_m}}[(1 - \alpha_m) z^{-1} S_m(z) + \frac{1}{2}(1 - \alpha_m)\alpha_m z^{-2} S_m(z)]$$

$$+ \frac{\Delta\varepsilon_m}{1 + (\frac{\tau_m}{\Delta t})^{1-\alpha_m}} E(z) \ . \tag{4.42}$$

Then Equation (4.34) in the z-domain becomes

$$D(z) = \varepsilon_\infty E(z) + \sum_{m=1}^{4} S_m(z) + I(z)$$

$$= AE(z) + \sum_{m=1}^{4} B_m[(1 - \alpha_m) z^{-1} S_m(z) + \frac{1}{2}(1 - \alpha_m)\alpha_m z^{-2} S_m(z)] + z^{-1} I(z), \tag{4.43}$$

where

$$A = \varepsilon_\infty + \frac{\sigma_s \Delta t}{\varepsilon_0} + \sum_{m=1}^{4} \frac{\Delta \varepsilon_m}{1 + (\frac{\tau_m}{\Delta t})^{1-\alpha_m}}, \tag{4.44}$$

$$S_m(z) = B_m[(1-\alpha_m)z^{-1}S_m(z) + \frac{1}{2}(1-\alpha_m)\alpha_m z^{-2}S_m(z)] \tag{4.45}$$

$$+ \frac{\Delta \varepsilon_m}{1 + (\frac{\tau_m}{\Delta t})^{1-\alpha_m}} E(z) ,$$

and

$$B_m = \frac{(\frac{\tau_m}{\Delta t})^{1-\alpha_m}}{1 + (\frac{\tau_m}{\Delta t})^{1-\alpha_m}} , \quad m = 1,2,3,4. \tag{4.46}$$

Hence, we can transform Equation (4.43) back to the time domain and obtain E at the time step n as follows:

$$E^n = \frac{1}{A}\{D^n - I^{n-1} - \sum_{m=1}^{4} B_m[(1-\alpha_m)S_m^{n-1} + \frac{1}{2}(1-\alpha_m)\alpha_m S_m^{n-2}]\} , \tag{4.47}$$

where I^n and S_m^n $(m = 1,2,3,4)$ are calculated as follows:

$$I^n = \frac{\sigma_s \Delta t}{\varepsilon_0} \cdot E^n + I^{n-1} , \tag{4.48}$$

$$S_m^n = B_m[(1-\alpha_m)S_m^{n-1} + \frac{1}{2}(1-\alpha_m)\alpha_m S_m^{n-2}]$$

$$+ \frac{\Delta \varepsilon_m}{1 + (\frac{\tau_m}{\Delta t})^{1-\alpha_m}} E^n . \tag{4.49}$$

4.3 Boundary Conditions

4.3.1 Boundary Conditions in One-Dimensional Space

In one-dimensional space, absorbing boundary conditions are necessary to keep outgoing electric and magnetic fields from being reflected back into the problem space. Usually, in computing the electric field, we need to know the surrounding magnetic field, which is a fundamental assumption of the FDTD method. However, we do not have the value to one side at the edge of the problem space. Since there is no source outside the problem space, the fields at the edge must be propagating outward. Thus we can estimate the value at the edge by using the value next to it.

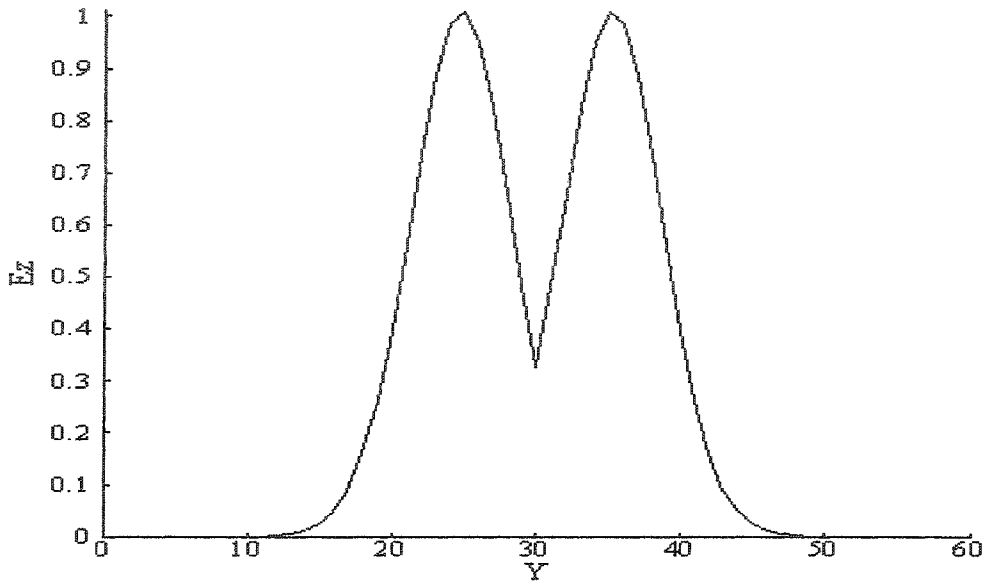
Suppose we are looking for a boundary condition at the end where $j = 0$. If a wave is going toward a boundary in free space, it is traveling at the speed of light. So in one time step, it travels:

$$\text{Distance} = c_0 \cdot \Delta t = c_0 \cdot \frac{\Delta x}{2c_0} = \frac{\Delta x}{2} .$$

This equation explains that it takes two time steps for a wave front to cross one grid. So an reasonable boundary condition in one dimension is

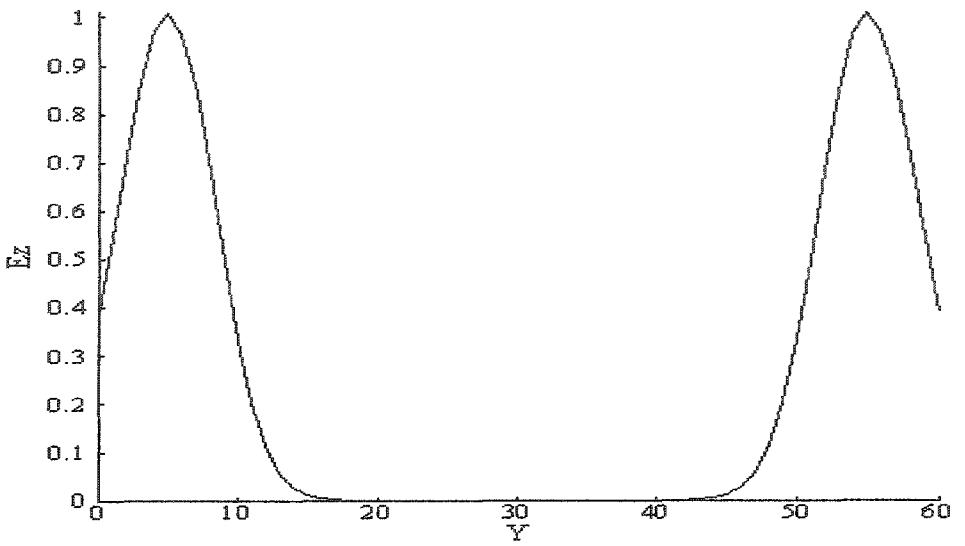
$$E_z^n(0) = E_z^{n-2}(0).$$

To implement this method, we just simply store a value of $E_z(1)$ for two time steps, and then put it in $E_z(0)$. Boundary condition like this has to be implemented at both ends. Figure 4.1 shows the results of a one-dimensional simulation in free space with absorbing boundary condition. A pulse that originates in the center propagates outwards and is absorbed without any reflection.

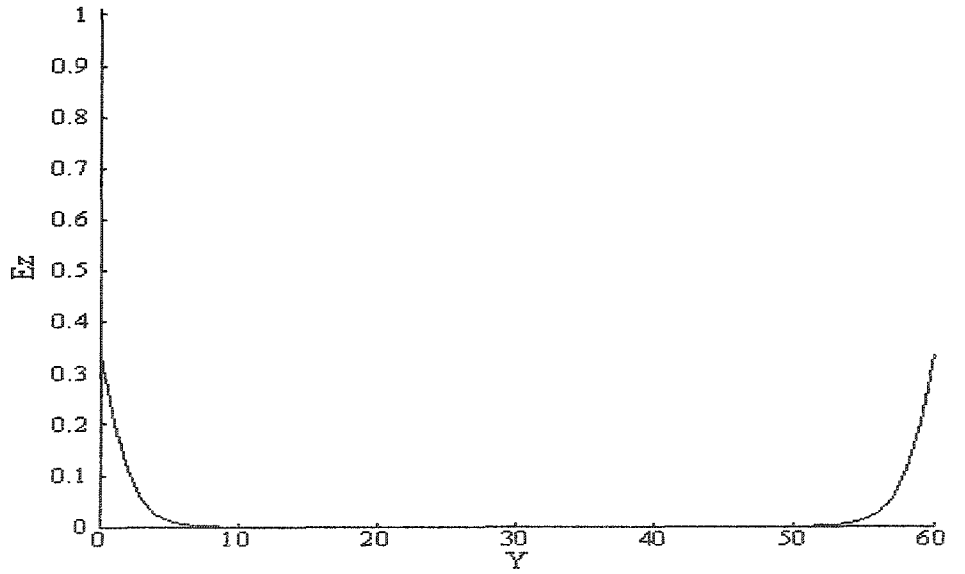


(a)

Figure 4.1. — (a), (b), (c) Simulation of an FDTD program in one-dimensional free space with absorbing boundary conditions at different time steps: (a) 110 time steps; (b) 150 time steps; (c) 170 time steps.



(b)



(c)

(Figure 4.1, Continued)

4.3.2 Boundary Conditions in Two-Dimensional Space

Suppose we are simulating a pulse generated from a point source propagating in two-dimensional free space. As the front of the pulse eventually come to the edge of the problem space. If nothing were done to address this situation, unpredictable reflections would happen. There is no way to distinguish between the real pulse and the reflected junk. However, there have been numerous approaches to this problem [6, 35].

One of the most efficient methods to reduce the reflection is the perfectly matched layer (PML) developed by Berenger [10, 11]. The basic idea is this: suppose a plane wave is propagating in medium A and strikes in a medium B, the fraction that is reflected is given by the reflection coefficient Γ, which is dictated by the intrinsic impedances η_A and η_B [41]:

$$\Gamma = \frac{\eta_A - \eta_B}{\eta_A + \eta_B}, \tag{4.50}$$

where η_A and η_B are determined by the dielectric constants ε and the permeabilities μ of the two media

$$\eta = \sqrt{\frac{\mu}{\varepsilon}}\ . \tag{4.51}$$

We have assumed that the medias are nonmagnetic, that is, $\mu = \mu_0$. So when a pulse is traveling in medium A with $\varepsilon_A = 1$ and it impinges upon medium B with $\varepsilon_B = 4$, it "finds" a change in impedance and reflected a portion of itself given by Equation (4.50). However, if μ changes with ε so that η remains a constant, Γ will be zero and no reflection will occur. This is not a final solution yet, because the pulse will continue propagating in the new medium. What we really want is a lossy medium that makes the

pulse die out before it hits the boundary. This is accomplished by making both ε and μ complex numbers, because the imaginary part of dialectic constant represents the decay.

On the other hand, in two-dimensional space, the governing equations can be reduced to:

$$\frac{\partial D_z}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right), \tag{4.51}$$

$$D_z(\omega) = \varepsilon_r(\omega) E_z(\omega), \tag{4.52}$$

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\partial E_z}{\partial y}, \tag{4.53}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\partial E_z}{\partial x}. \tag{4.54}$$

If the electromagnetic field vectors are characterized by sinusoidal variation in time, they can be written in the phasor forms. As an example, the electric filed intensity can be written as $E = E_0 e^{j\omega t}$. The derivative with respect to time, $d/d\omega$, is therefore $j\omega$ in the Fourier domain. By such, Equations (4.51)-(4.54) can be written in the Fourier domain as follows:

$$j\omega D_z = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right), \tag{4.55}$$

$$D_z(\omega) = \varepsilon_r(\omega) E_z(\omega), \tag{4.56}$$

$$j\omega H_x = -\frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\partial E_z}{\partial y}, \tag{4.57}$$

$$j\omega H_y = \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\partial E_z}{\partial x} \,. \tag{4.58}$$

Then, we will add some fictitious dielectric constants and permeabilities ε_{Fz}, μ_{Fx} and μ_{Fy} [42, 43]:

$$j\omega D_z \cdot \varepsilon_{Fz}(x) \cdot \varepsilon_{Fz}(y) = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right), \tag{4.59}$$

$$D_z(\omega) = \varepsilon_r(\omega) E_z(\omega) \,, \tag{4.60}$$

$$j\omega H_x \cdot \mu_{Fx}(x) \cdot \mu_{Fx}(y) = -\frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\partial E_z}{\partial y} \,, \tag{4.61}$$

$$j\omega H_y \cdot \mu_{Fy}(x) \cdot \mu_{Fy}(y) = \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\partial E_z}{\partial x} \,. \tag{4.62}$$

From the above equations, we can see these fictitious values have nothing to do with the *real* values of $\varepsilon_r(\omega)$ that specify the medium.

Sacks, et al. [44] shows two conditions to build a PML:

(1) The impedance going from the background medium (medium A) to the PML (medium B) must be constant,

$$\eta_A = \eta_B = \sqrt{\frac{\mu_{Fm}}{\varepsilon_{Fm}}} = 1 \qquad m = x, y \text{ or } z \,. \tag{4.63}$$

The impedance is 1 because we normalize the units.

(2) In the direction perpendicular to the boundary, the relative dielectric constant and relative permeability must be the inverse of those in the other directions, i.e., in the x direction,

$$\varepsilon_{Fx}(x) = \frac{1}{\varepsilon_{Fn}(x)}, \quad n = y \text{ or } z, \tag{4.64}$$

$$\mu_{Fx}(x) = \frac{1}{\mu_{Fn}(x)}, \quad n = y \text{ or } z. \tag{4.65}$$

We will assume that each of these is a complex quantity of the form

$$\varepsilon_{Fx}(x) = \mu_{Fx}(x) = (1 + \frac{\sigma_D(x)}{j\omega\varepsilon_0})^{-1}, \tag{4.66}$$

$$\mu_{Fn}(x) = \varepsilon_{Fn}(x) = 1 + \frac{\sigma_D(x)}{j\omega\varepsilon_0}, \quad n = y \text{ or } z. \tag{4.67}$$

Obviously, Equation (4.66) and (4.67) fulfills Equation (4.63)-(4.65). If σ_D increases gradually as it goes into the PML, Equations (4.59), (4.61) and (4.62) will cause $D_z(x), H_x(x)$ and $H_y(x)$ to be attenuated.

Starting by implementing a PML only in the x direction, we retain only the x dependent values of ε_F and μ_F in Equations (4.59), (4.61) and (4.62) and obtain:

$$j\omega D_z \cdot \varepsilon_{Fz}(x) = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}), \tag{4.68}$$

$$j\omega H_x \cdot \mu_{Fx}(x) = -\frac{1}{\sqrt{\mu_0\varepsilon_0}}\frac{\partial E_z}{\partial y}, \tag{4.69}$$

$$j\omega H_y \cdot \mu_{Fy}(x) = \frac{1}{\sqrt{\mu_0\varepsilon_0}}\frac{\partial E_z}{\partial x}. \tag{4.70}$$

Using the values of Equation (4.66) and (4.67), one may obtain as follows:

$$j\omega D_z \cdot [1 + \frac{\sigma_D(x)}{j\omega\varepsilon_0}] = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}), \tag{4.71}$$

$$j\omega H_x \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}]^{-1} = -\frac{1}{\sqrt{\mu_0\varepsilon_0}}\frac{\partial E_z}{\partial y}, \tag{4.72}$$

$$j\omega H_y \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}] = \frac{1}{\sqrt{\mu_0\varepsilon_0}}\frac{\partial E_z}{\partial x}. \tag{4.73}$$

First, take a look at the left side of Equation (4.71):

$$j\omega D_z \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}] = j\omega D_z + \frac{\sigma_D(x)}{\varepsilon_0} D_z. \tag{4.74}$$

Going back to the time, Equation (4.71) becomes

$$\frac{\partial D_z}{\partial t} + \frac{\sigma_D(x)}{\varepsilon_0} D_z = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}) . \tag{4.75}$$

Taking the finite difference approximation, we obtain:

$$\frac{D_z^{n+1/2}(i,j) - D_z^{n-1/2}(i,j)}{\Delta t} + \frac{\sigma_D(i)}{\varepsilon_0}\frac{D_z^{n+1/2}(i,j) + D_z^{n+1/2}(i,j)}{2}$$

$$= \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{H_y^n(i+1/2,j) - H_y^n(i-1/2,j)}{\Delta x})$$

$$- \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{H_x^n(i,j+1/2) - H_x^n(i,j-1/2)}{\Delta x}). \tag{4.76}$$

Rearranging Equation (4.76) gives:

$$D_z^{n+1/2}(i,j) = gi3(i)D_z^{n-1/2}(i,j) + gi2(i)\cdot\frac{1}{2}\cdot[H_y^n(i+1/2,j) - H_y^n(i-1/2,j) - H_x^n(i,j+1/2) + H_y^n(i,j-1/2)], \tag{4.77}$$

where

$$gi2(i) = \frac{1}{1+\sigma_D(i)\Delta t/(2\cdot\varepsilon_0)}, \tag{4.78}$$

$$gi3(i) = \frac{1-\sigma_D(i)\Delta t/(2\cdot\varepsilon_0)}{1+\sigma_D(i)\Delta t/(2\cdot\varepsilon_0)}. \tag{4.79}$$

An almost identical treatment of Equation (4.73) gives:

$$H_y^{n+1}(i+1/2,j) = fi3(i+1/2)H_y^n(i+1/2,j) + fi2(i+1/2)\cdot\frac{1}{2}\cdot[E_z^{n+1/2}(i+1,j) - E_z^{n+1/2}(i,j)], \tag{4.80}$$

where

$$fi2(i+1/2) = \frac{1}{1+\sigma_D(i+1/2)\Delta t/(2\cdot\varepsilon_0)}, \tag{4.81}$$

$$fi3(i+1/2) = \frac{1-\sigma_D(i+1/2)\Delta t/(2\cdot\varepsilon_0)}{1+\sigma_D(i+1/2)\Delta t/(2\cdot\varepsilon_0)}. \tag{4.82}$$

Equation (4.72) will require a different treatment than the other two. Start by rewriting it as

$$j\omega H_x = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}[\frac{\partial E_z}{\partial y} + \frac{\sigma_D(x)}{\varepsilon_0}\frac{1}{j\omega}\frac{\partial E_z}{\partial y}]. \tag{4.83}$$

Because $1/j\omega$ can be regarded as an integration operator over time and $j\omega$ as a derivative over time, we can rewrite Equation (4.83) in the time domain as:

$$\frac{H_x^{n+1}(i,j+1/2)-H_x^n(i,j+1/2)}{\Delta t} = \frac{1}{\sqrt{\varepsilon_0\mu_0}}[\frac{E_z^{n+1/2}(i,j)-E_z^{n+1/2}(i,j+1)}{\Delta x} + \frac{\sigma_D(i)}{\varepsilon_0}\Delta t\sum_{t=0}^{n}\frac{E_z^{n+1/2}(i,j)-E_z^{n+1/2}(i,j+1)}{\Delta x}]. \tag{4.84}$$

Finally, we can write the schemes as follows:

$$curl = E_z^{n+1/2}(i,j) - E_z^{n+1/2}(i,j+1), \tag{4.85}$$

$$I_{Hx}^{n+1/2}(i,j+1/2) = I_{Hx}^{n-1/2}(i,j+1/2) + curl, \tag{4.86}$$

$$H_x^{n+1}(i,j+1/2) = H_x^n(i,j+1/2) + 0.5curl + fi1(i)\cdot I_{Hx}^{n+1/2}(i,j+1/2) \tag{4.87}$$

with

$$fi1(i) = \frac{\sigma_D(i)}{2\varepsilon_0}. \tag{4.88}$$

In calculating f and g, it is not necessary to actually vary conductivities. Instead, one may use the auxiliary parameter

$$x_n = \frac{\sigma_D(i)}{2\varepsilon_0}, \tag{4.89}$$

that increases as it penetrates more deeply into the perfectly matched layer. Here, we employ an empirical formula given in [28] and let

$$x_n = \frac{1}{3}\cdot\left(\frac{i}{l_{pml}}\right)^3, \quad i = 1,2,...,l_{pml}, \tag{4.90}$$

where l_{pml} is the length of the perfectly matched layer. Now the quantities f and g are calculated as follows:

$$fi1(i) = x_n(i),\ fi2(i+1/2) = \frac{1}{1+x_n(i+1/2)},\ fi3(i+1/2) = \frac{1-x_n(i+1/2)}{1+x_n(i+1/2)}, \tag{4.91}$$

$$gi2(i) = \frac{1}{1+x_n(i)},\ gi3(i) = \frac{1-x_n(i)}{1+x_n(i)}. \tag{4.92}$$

Throughout the main problem space, $fi1$ is 0, while the other four are 1.

So far, the implementation of the PML in the x direction is completed. Obviously, it should also be done in the y direction. We have to add y dependent terms from Equations (4.59), (4.61) and (4.62). Therefore, we have

$$j\omega D_z \cdot [1 + \frac{\sigma_D(x)}{j\omega\varepsilon_0}][1 + \frac{\sigma_D(y)}{j\omega\varepsilon_0}] = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}), \tag{4.93}$$

$$j\omega H_x \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}]^{-1}[1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}] = -\frac{1}{\sqrt{\mu_0\varepsilon_0}}\frac{\partial E_z}{\partial y} \quad , \tag{4.94}$$

$$j\omega H_y \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}][1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}]^{-1} = \frac{1}{\sqrt{\mu_0\varepsilon_0}}\frac{\partial E_z}{\partial x} \quad . \tag{4.95}$$

Using the same procedure as before, we obtain the schemes for D_z:

$$\begin{aligned} D_z^{n+1/2}(i,j) &= gi3(i)\cdot gj3(i)\cdot D_z^{n-1/2}(i,j) \\ &+ gi2(i)\cdot gj2(i)\cdot\frac{1}{2}\cdot[H_y^n(i+1/2,j) \\ &- H_y^n(i-1/2,j) - H_x^n(i,j+1/2) + H_y^n(i,j-1/2)], \end{aligned} \tag{4.96}$$

for H_x:

$$\begin{aligned} curl &= E_z^{n+1/2}(i,j) - E_z^{n+1/2}(i,j+1) \ , \\ I_{Hx}^{n+1/2}(i,j+1/2) &= I_{Hx}^{n-1/2}(i,j+1/2) + curl \ , \\ H_x^{n+1}(i,j+1/2) &= fj3(j+1/2)H_x^n(i,j+1/2) \\ &+ fj2(j+1/2)\cdot[0.5curl + fi1(i)\cdot I_{Hx}^{n+1/2}(i,j+1/2)] \end{aligned} \tag{4.97}$$

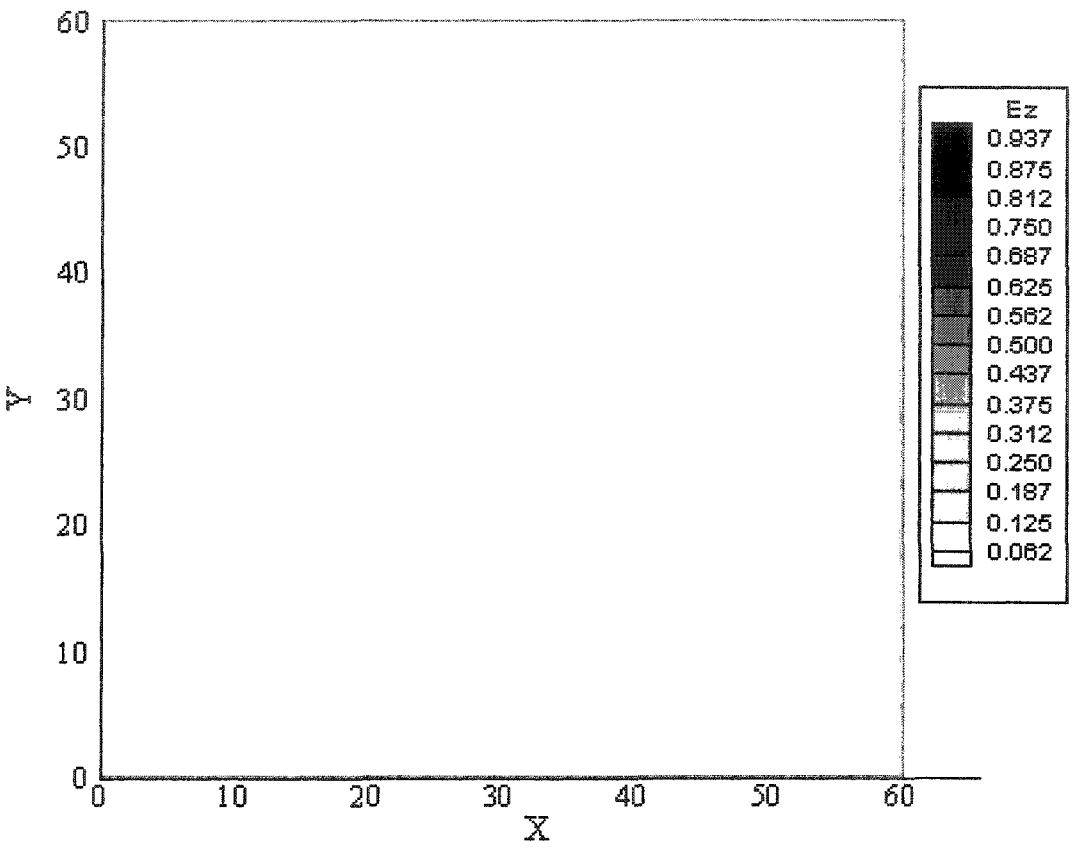
and for H_y:

$$\begin{aligned} curl &= E_z^{n+1/2}(i+1,j) - E_z^{n+1/2}(i,j), \\ I_{Hx}^{n+1/2}(i+1/2,j) &= I_{Hx}^{n-1/2}(i+1/2,j) + curl \ , \\ H_y^{n+1}(i+1/2,j) &= fi3(i+1/2)H_y^n(i+1/2,j) \\ &+ fi2(i+1/2)\cdot[0.5curl + fj1(j)\cdot I_{Hx}^{n+1/2}(i+1/2,j)], \end{aligned} \tag{4.98}$$

where fj and gj have same expression as fi and gi.

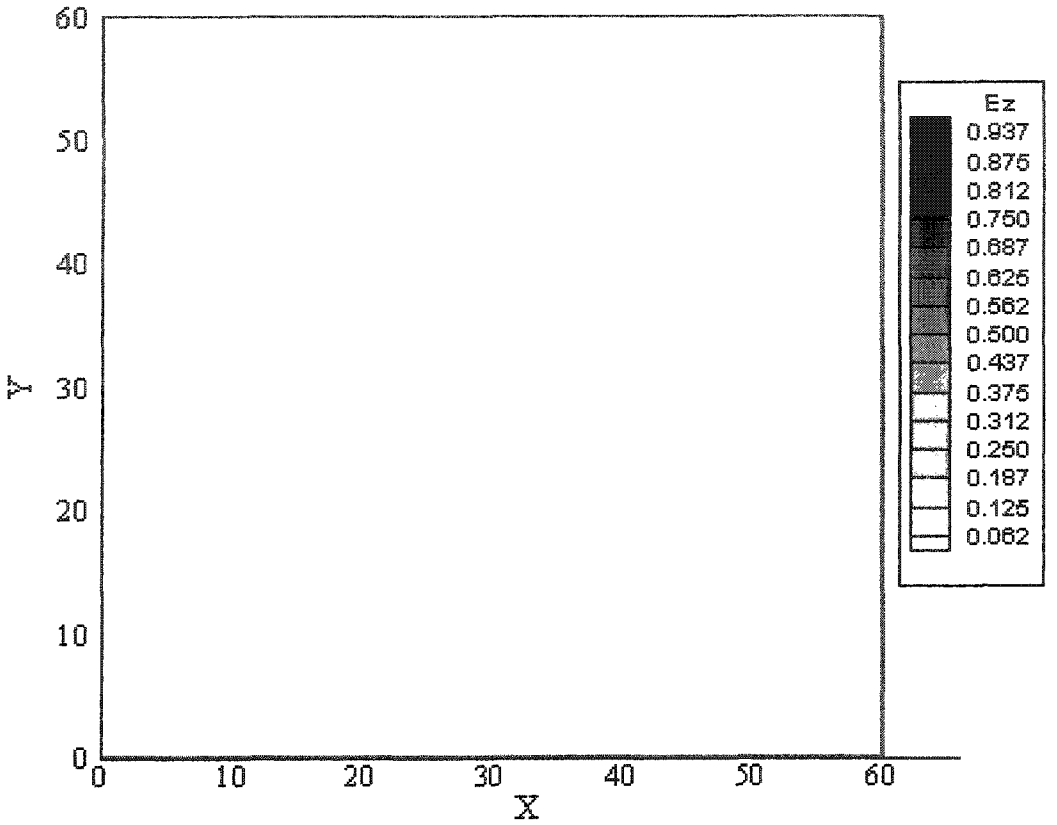
Figure 4.2 shows the results of a two-dimensional simulation in free space with perfectly matched layer. A pulse originates in the center of the problem space. Notice that the outgoing contours remain concentric. Only when the front of the pulse gets within 5

points of the edge, which is the PML, does distortion start to occur. The effectiveness of the PML is apparent in the last figure because no reflection occurs.



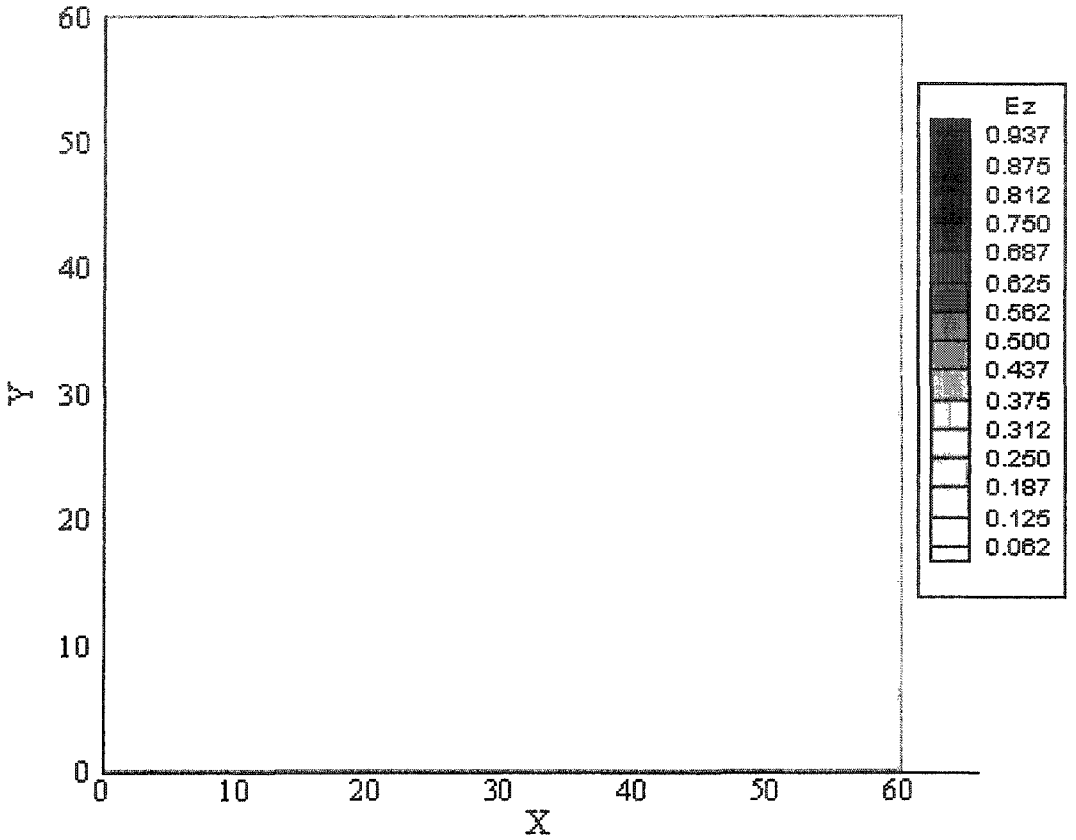
(a)

Figure 4.2. — (a), (b), (c) Simulation of an FDTD program in two-dimensional free space with a perfectly matched layer at different time steps: (a) $160\Delta t$; (b) $170\Delta t$; (c) $180\Delta t$.



(b)

(Figure 4.2, Continued)



(c)

(Figure 4.2, Continued)

4.3.3 Boundary Conditions in Three-Dimensional Space

The development of perfectly matched layer for three dimensions exactly follows the two-dimension case. The difference is there are 6 scalar equations each of which has 3 directions to deal with. So the Maxwell's Equations in the PML can be written as:

$$j\omega D_x \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}]^{-1}[1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}][1+\frac{\sigma_D(z)}{j\omega\varepsilon_0}] = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial H_z}{\partial y}-\frac{\partial H_y}{\partial z}), \tag{4.99}$$

$$j\omega D_y \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}][1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}]^{-1}[1+\frac{\sigma_D(z)}{j\omega\varepsilon_0}] = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial H_x}{\partial z}-\frac{\partial H_z}{\partial x}), \tag{4.100}$$

$$j\omega D_z \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}]\ [1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}][1+\frac{\sigma_D(z)}{j\omega\varepsilon_0}]^{-1} = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial H_y}{\partial x}-\frac{\partial H_x}{\partial y}), \tag{4.101}$$

$$j\omega H_x \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}]^{-1}[1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}][1+\frac{\sigma_D(z)}{j\omega\varepsilon_0}] = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial E_y}{\partial z}-\frac{\partial E_z}{\partial y}), \tag{4.102}$$

$$j\omega H_y \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}][1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}]^{-1}[1+\frac{\sigma_D(z)}{j\omega\varepsilon_0}] = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial E_z}{\partial x}-\frac{\partial E_x}{\partial z}), \tag{4.103}$$

$$j\omega H_z \cdot [1+\frac{\sigma_D(x)}{j\omega\varepsilon_0}]\ [1+\frac{\sigma_D(y)}{j\omega\varepsilon_0}][1+\frac{\sigma_D(z)}{j\omega\varepsilon_0}]^{-1} = \frac{1}{\sqrt{\varepsilon_0\mu_0}}(\frac{\partial E_x}{\partial y}-\frac{\partial E_y}{\partial x}). \tag{4.104}$$

Following the same math we use in the two-dimensional case, we obtain the following finite difference schemes:

$$curl = [H_z^n(i+1/2, j+1/2, k) - H_z^n(i+1/2, j-1/2, k) - H_y^n(i+1/2, j, k+1/2) + H_y^n(i+1/2, j, k-1/2)],$$

$$I_{Dx}^n(i+1/2, j, k) = I_{Dx}^n(i+1/2, j, k) + curl,$$

$$D_x^{n+1/2}(i+1/2, j, k) = gj3(j)gk3(k)D_x^{n-1/2}(i+1/2, j, k) + \frac{1}{2}gj2(j)gk2(k)[curl + gi1(i+1/2)I_{dx}^n(i+1/2, j, k)], \tag{4.105}$$

with corresponding expressions for $D_y^{n+1/2}(i, j+1/2, k)$ and $D_z^{n+1/2}(i, j, k+1/2)$, and

$$curl = [E_y^{n+1/2}(i, j+1/2, k+1) - E_y^{n+1/2}(i, j+1/2, k)$$
$$- E_z^{n+1/2}(i, j+1, k+1/2) + E_z^{n+1/2}(i, j, k+1/2)]$$

$$I_{Hx}^{n+1/2}(i, j+1/2, k+1/2) = I_{Hx}^{n-1/2}(i, j+1/2, k+1/2) + curl$$

$$H_x^{n+1}(i, j+1/2, k+1/2) = fj3(j+1/2)\,fk3(k+1/2)H_x^n(i, j+1/2, k+1/2)$$
$$+\frac{1}{2} fj2(j+1/2)\,fk2(k+1/2)[curl + fi(i)I_{Hx}^{n+1/2}(i, j+1/2, k+1/2) \quad , \quad (4.106)$$

with corresponding expressions for $H_y^{n+1}(i+1/2, j, k+1/2)$ and

$H_z^{n+1}(i+1/2, j+1/2, k)$, where

$$fm1(x) = gm1(x) = \frac{\sigma_D(x)\Delta t}{2\varepsilon_0}, \quad m = i, j, k \ , \quad \text{in PML,}$$

$$fm1(x) = gm1(x) = 0, \quad m = i, j, k \ , \quad \text{in the main problem space,} \quad (4.107)$$

$$fm2(x) = gm2(x) = \frac{1-\sigma_D(x)\Delta t/2\varepsilon_0}{1+\sigma_D(x)\Delta t/2\varepsilon_0} \ , \ m = i, j, k \ , \quad \text{in PML,}$$

$$fm2(x) = gm2(x) = 1 \ , \ m = i, j, k \ , \quad \text{in the main problem space,} \quad (4.108)$$

$$fm3(x) = gm3(x) = \frac{1}{1+\sigma_D(x)\Delta t/2\varepsilon_0} \ , \ m = i, j, k \ , \quad \text{in PML,}$$

$$fm3(x) = gm3(x) = 1 \ , \ m = i, j, k \ , \quad \text{in the main problem space.} \quad (4.109)$$

Notice that $\sigma_D(i)/2\varepsilon_0$ is approximated by the empirical formula Equation (4.90).

4.4 Simulation of a Nanopulse

4.4.1 Simulating a Nanopulse in One-Dimensional Space

In one-dimensional space, the source term, the nanopulse, is calculated by Equation (3.16), after the E_z values are computed within each time step. This is done by simply specifying a value of E_z at the source point, and overriding what was previously calculated. Otherwise, without adding the value of the previous time step to the source term, there would be an "invisible wall" on the source point, which reflects the income wave.

4.4.2 Simulating a Nanopulse in Two-Dimensional Space

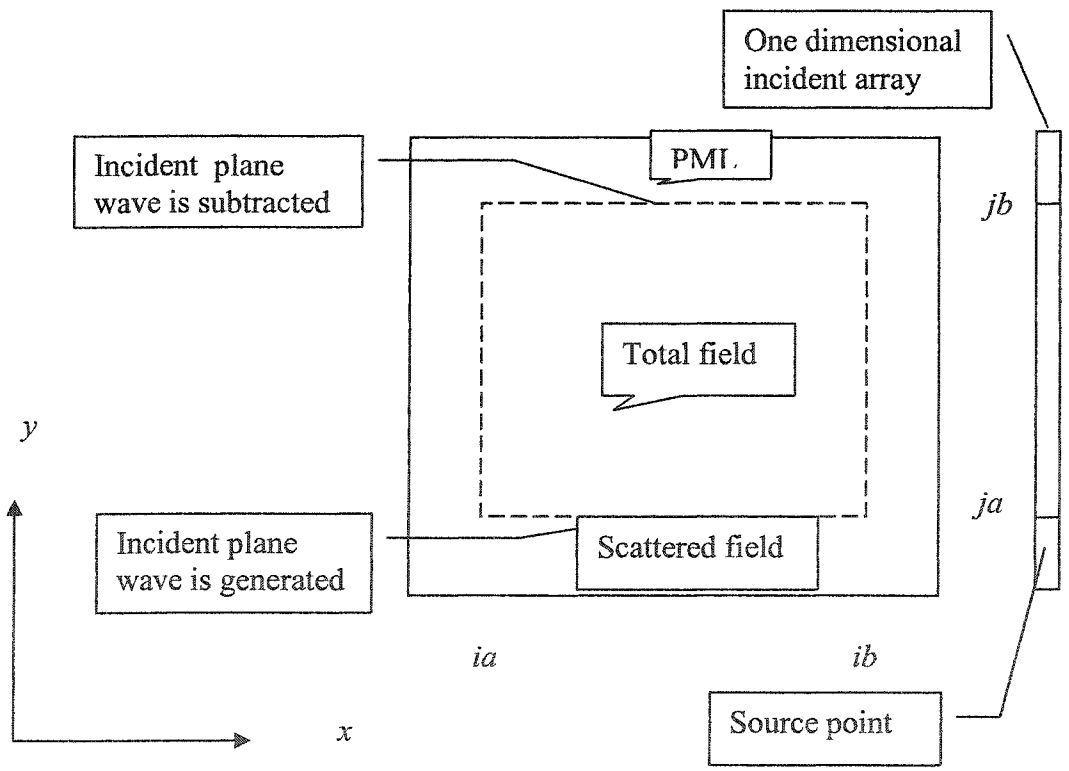


Figure 4.3 Total/Scattered field of the two-dimensional problem space

To simulate a nanopulse in a 2D FDTD program, the problem space must be divided into two regions: the total field and the scattered field (Figure 4.3). The reason to do this is to prevent the plane wave from interacting with the absorbing boundary conditions and minimize the load on the absorbing boundary conditions. Figure 4.3 illustrates how this is accomplished. We use an auxiliary one-dimensional array, which is easy to generate a plane wave. Choosing a source point we add an incident E_z field at that point. Then a plane wave propagates away in both directions. Since it is a one-dimensional array, the boundary conditions are perfect.

In the two-dimensional field every point in the problem space is either in the total field or in the scattered field. Therefore, if a point is in the total field but it uses points in the scattered field to calculate the spatial derivatives when updating its value, it must be modified. The same is true for a point in the scattered field but use the points inside the total field. Finally, in the two dimensional space, there are three place must be modified:

(1) The D_z value at $j = ja$ or $j = jb$

$$D_z(i, ja) = D_z(i, ja) + 0.5H_{x\_inc}(ja - 1/2) \tag{4.110}$$

$$D_z(i, jb) = D_z(i, jb) - 0.5H_{x\_inc}(jb + 1/2) \tag{4.111}$$

(2) The H_x value just outside $j = ja$ or $j = jb$

$$H_x(i, ja - 1/2) = H_x(i, ja - 1/2) + 0.5E_{x\_inc}(ja) \tag{4.112}$$

$$H_x(i, jb + 1/2) = H_x(i, jb + 1/2) - 0.5E_{x\_inc}(jb) \tag{4.113}$$

(3) The H_y value just outside $j = ja$ or $j = jb$

$$H_y(ia - 1/2, j) = H_y(ia - 1/2, j) + 0.5E_{x\_inc}(j) \tag{4.114}$$

$$H_y(ib + 1/2, j) = H_y(ib + 1/2, j) - 0.5E_{x\_inc}(j) \tag{4.115}$$

4.4.3 Simulating a Nanopulse in Three-Dimensional Space

It is similar to generate a nanopulse in three-dimensional space. A plane wave is generated in one plane of the problem space, in this case, we suppose it is XZ plane. So the plane is generated at $j = ja$ and subtracted out at $j = jb$.

Finally, four places must be modified:

(1) The D_y value at $k = ka$ or $k = kb + 1$

$$D_y(i, j+1/2, ka) = D_z(i, j+1/2, ka) - 0.5H_{x\_inc}(j+1/2) \tag{4.116}$$

$$D_y(i, j+1/2, kb+1) = D_z(i, j+1/2, kb+1) + 0.5H_{x\_inc}(j+1/2) \tag{4.117}$$

(2) The D_z value at $j = ja$ or $j = jb$

$$D_z(i, ja, k+1/2) = D_z(i, ja, k+1/2) + 0.5H_{x\_inc}(ja-1/2) \tag{4.118}$$

$$D_z(i, jb, k+1/2) = D_z(i, jb, k+1/2) - 0.5H_{x\_inc}(jb+1/2) \tag{4.119}$$

(3) The H_x value just outside $j = ja$ or $j = jb$

$$H_x(i, ja-1/2, k+1/2) = H_x(i, ja-1/2, k+1/2) + 0.5E_{x\_inc}(ja) \tag{4.120}$$

$$H_x(i, jb+1/2, k+1/2) = H_x(i, jb+1/2, k+1/2) - 0.5E_{x\_inc}(jb) \tag{4.121}$$

(4) The H_y value just outside $j = ja$ or $j = jb$

$$H_y(ia-1/2, j, k+1/2) = H_y(ia-1/2, j, k+1/2) + 0.5E_{x\_inc}(j) \tag{4.122}$$

$$H_y(ib+1/2, j, k+1/2) = H_y(ib+1/2, j, k+1/2) - 0.5E_{x\_inc}(j) \tag{4.123}$$

4.4.4 Pulse Energy

The energy stored in an electric field is calculated as Equation (3.17):

$$Energy = \varepsilon_0 c_0 \int_0^T E^2(t)dt,$$

which can be easily approximated by:

$$Energy = \varepsilon_0 c_0 \Delta t \sum_{n=0}^{T} (E^n)^2 . \tag{4.124}$$

This expression can be used to calculate the transmission, reflection, and absorption of pulse energy by a barrier as a function of pulse width. Transmission is defined as the ratio of the pulse energy entering the barrier, here, a biological tissue to the initial pulse energy; reflection as the ratio of pulse energy reflected by the barrier to the initial pulse energy, and absorption as the ratio of pulse energy absorbed by barrier to the initial pulse energy.

However, one thing must be pointed out that a tricky in calculating the reflection because we need to separate the reflected pulse from the original one. This can be solved as follows: Suppose a pulse is propagating in the y direction. The electric filed amplitude at a fixed point M in front of the barrier is stored versus time. This fact results in the values of total electric field versus time at that point E_{tot}^n. We then remove the barrier and do the calculation again in free space. This method gives us the incident electric field at the point M, E_{inc}^n. Therefore, for each time step, the reflected field is given by

$$E_{ref}^n = E_{tot}^n - E_{inc}^n . \tag{4.125}$$

4.5 Algorithm

The algorithm for simulating the electromagnetic fields in biological tissues exposed to a nanopulse in three-dimensional space can be described as follows:

Step 1. Set up f and g as defined in Equations (4.107)-(4.109).

Step 2. Calculate D from Equation (4.105) and corresponding expressions for the y and z directions.

Step 3. Modify D_y, D_z as described in Equations (4.116)-(4.119).

Step 4. Calculate E from Equation (4.47) and then I^n and S_m^n ($m = 1,2,3,4$) from Equations (4.48) and (4.49).

Step 5. Calculate H from Equation (4.106) and corresponding expressions for the y and z directions.

Step 6. Modify H_x, H_y as described in Equations (4.120)-(4.123).

Step 7. Calculate energy rate from Equation (4.124).

Repeat steps 2-7 until the required steps are carried out.

CHAPTER 5

COMPUTATIONAL RESULTS AND DISCUSSION

5.1 Comparison of Different Models

In general, there are two ways to solve Maxwell's equations when coupled to the Cole-Cole expression. Usually, we can approximate the Cole-Cole model by other models such as a Debye model. However, we need to rewrite all the parameters for the new model, which is a painstaking job. The other way is transforming the Cole-Cole expression to the z-domain using the z-transform method and then approximating the results using a Taylor series in z.

We have computed the EM field in a tissue on exposure to a Gaussian pulse with both methods. The pulse was chosen to be $E_z = e^{-(300-n)^2/10000} V/m$, where n is the time step. The pulse shape is shown in Figure 5.1. We placed a slab having the electrical properties of human skin between the 900^{th} and 1100^{th} grid points in the propagating direction. The grid size was $\Delta x = 0.5 \mu m$. The parameter for the Cole-Cole model and Debye model are shown in Table 5.1. Figure 5.2 shows a sequence of snapshots of the electric field intensity versus position during pulse propagation. It is clear that there is no significant difference between the models for the chosen parameter values.

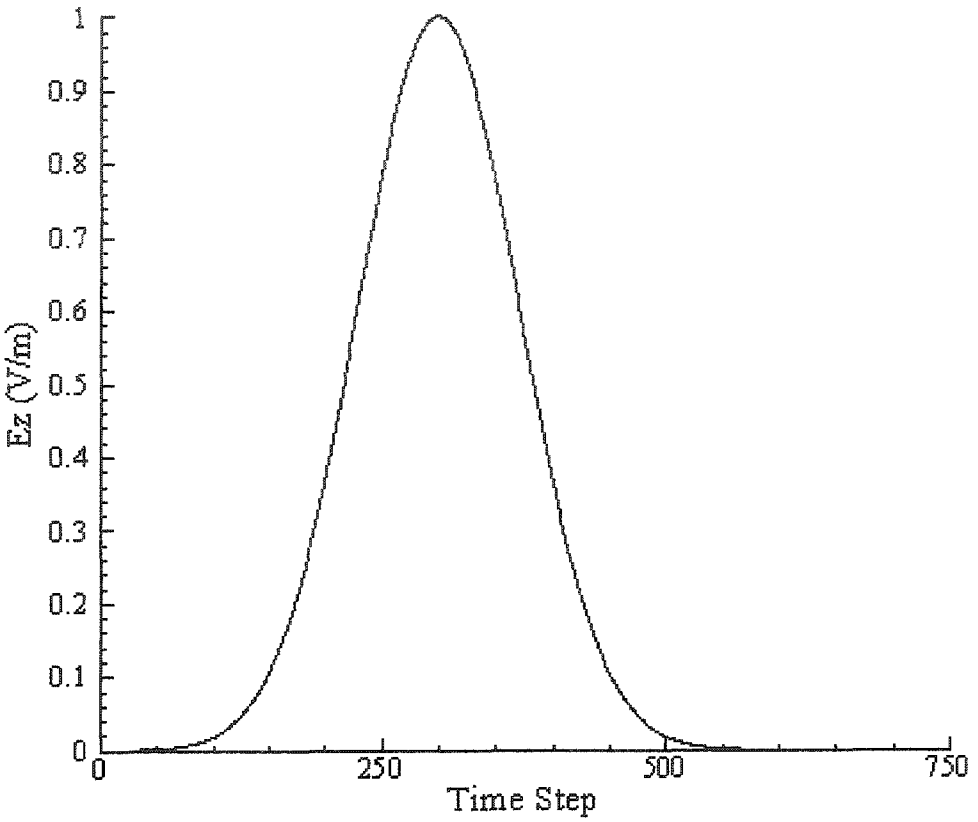
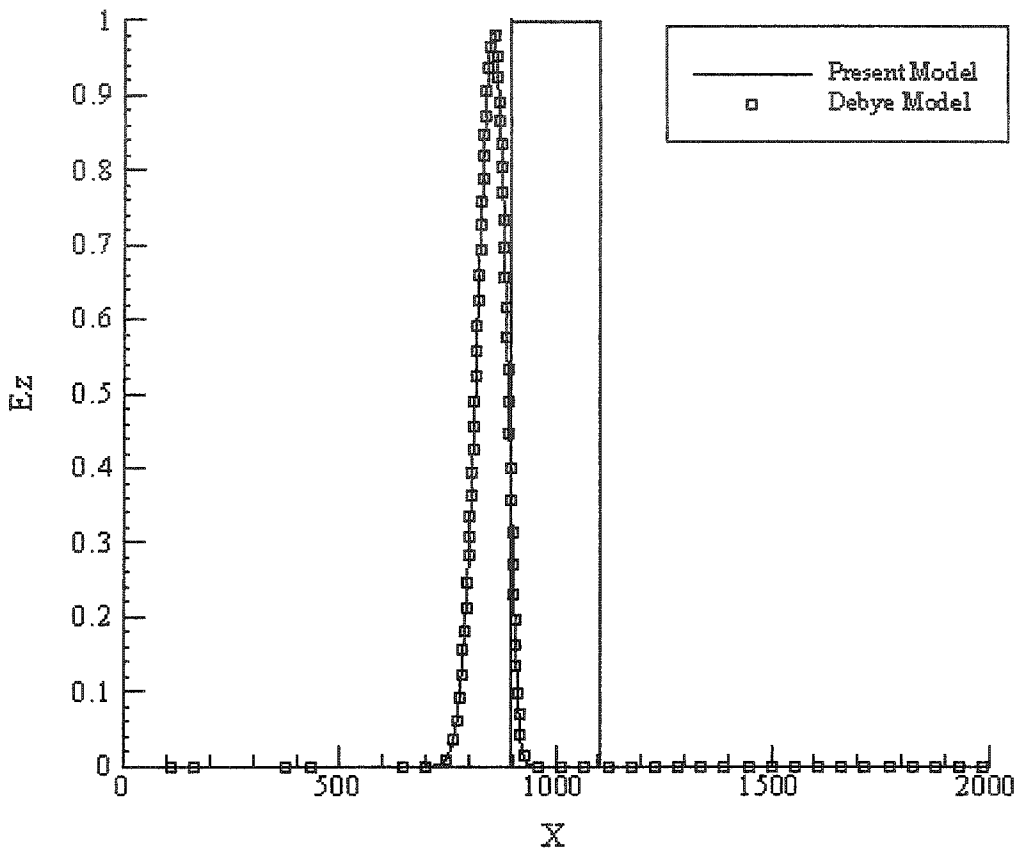


Figure 5.1. Pulse shape of $E_z = e^{-\frac{(300-n)^2}{10000}} V/m$

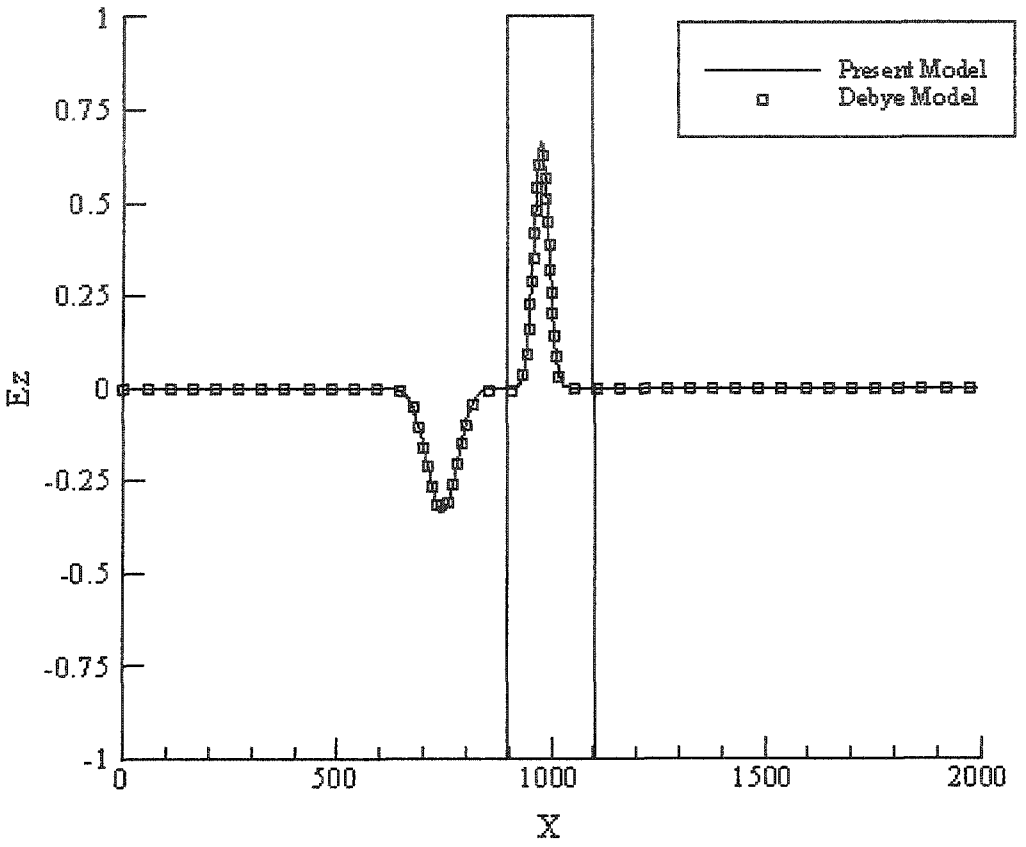
Table 5.1 Dielectric Properties of Human Skin [20, 45]

	Debye	Cole-Cole
ε_∞	4.0	4.0
σ_s	0.014	0.0002
$\Delta\varepsilon_1$	38	32
$\Delta\varepsilon_2$	0	1100
$\Delta\varepsilon_3$	0	0
$\Delta\varepsilon_4$	0	0
$\tau_1(ps)$	6.9	7.23
$\tau_2(ns)$	0	32.48
$\tau_3(\mu s)$	0	0
$\tau_4(ms)$	0	0
α_1	0	0.1
α_2	0	0.2
α_3	0	0
α_4	0	0



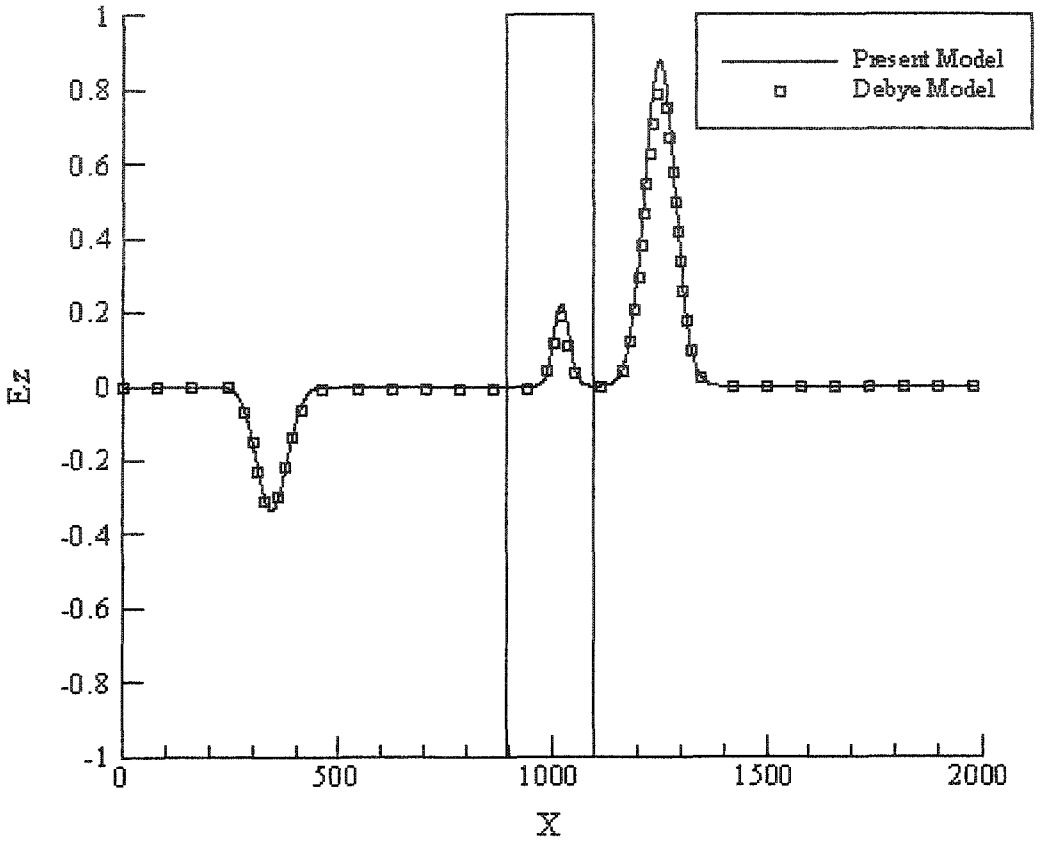
(a)

Figure 5.2.—(a)-(d) A sequence of snapshots of the electric field intensity versus position during pulse propagation at different time steps: (a) 2000 time steps; (b) 2400 time steps; (c) 3200 time steps; (d) 3600 time steps.



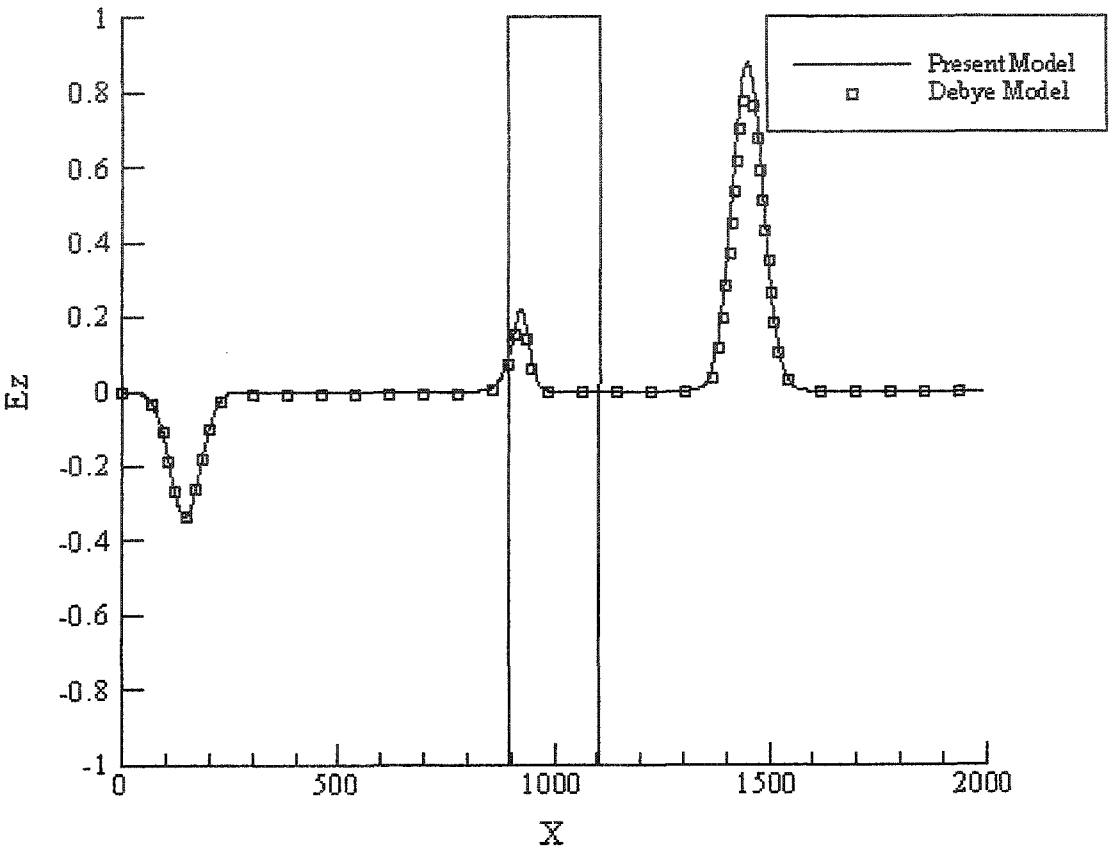
(b)

(Figure 5.2, Continued)



(c)

(Figure 5.2, Continued)



(d)

(Figure 5.2, Continued)

5.2 Numerical Results for Different Dimensions

We have tested the algorithm for obtaining EM fields induced by a Gaussian pulse shown in Figure 5.3. We modeled a slab having the dielectric properties of blood between the 25^{th} and 35^{th} grid points in the y direction. The problem spaces for 1D, 2D and 3D are listed as follows:

1D: $y \in [0,60]\mu m$,

2D: $x \times y \in [0,60]\mu m \times [0,60]\mu m$

3D: $x \times y \times z \in [0,60]\mu m \times [0,60]\mu m \times [0,60]\mu m$.

The parameters in Cole-Cole expression are shown in Table 5.2. The values $\Delta x = \Delta y = \Delta z = 1.0\mu m$ and $\Delta t = \frac{\Delta x}{2c_0} = 1.6667 \times 10^{-15} s$. Figures 5.4(a)-5.9(a), 5.4(b)-5.9(b) and 5.4(c)-5.9(c) show a sequence of snapshots of the electric field intensity versus position for the pulse at different time steps in 1D, 2D and 3D. Propagation and reflection of the pulse are clearly evident, as is the phase change on passing from a region of low dielectric (air) to one of high dielectric (blood). Furthermore, it is clear that the values of E_z are uniform at a fixed position of y in 2D and 3D; thus we can use the E_z values on the symmetric axis to represent the values in the whole space. That is, in the two-dimensional space, we pick up the E_z values on the line $x = 30$, and in the three-dimensional space, we choose the values on the line $x = 30, z = 30$. Figure 5.10 shows a sequence of snapshots of the values of E_z versus y position on the symmetric axis in 1D, 2D and 3D at different time step. As we can see, there is no major difference among these results for different dimensions in the total field, while in the perfect matched layer,

the pulse in 1D had a different behavior from those in 2D and 3D, that is because there is no artificial PML in 1D and we only apply an absorbing boundary condition to it.

Next, a model of a blood cell was placed at the center of the 3D cube. The radius of the cell was 10 μm and the other parameters were the same as the previous case. Figures 5.11(a)-5.15(a) and 5.11(b)-5.15(b) show the contours of E_z when the time are 140, 160, 170, 180 and 200, where the pulse is impinging on the cell, passing through the cell, passing out from the cell, passing through the boundary, and has passed through the boundary. Contours are plotted in the xy cross-section at $z = 30$ and in the half cube where $z = 0 \sim 30$. It is clear that the nanopulse penetrates a spherical cell having the indicated frequency-dependent properties.

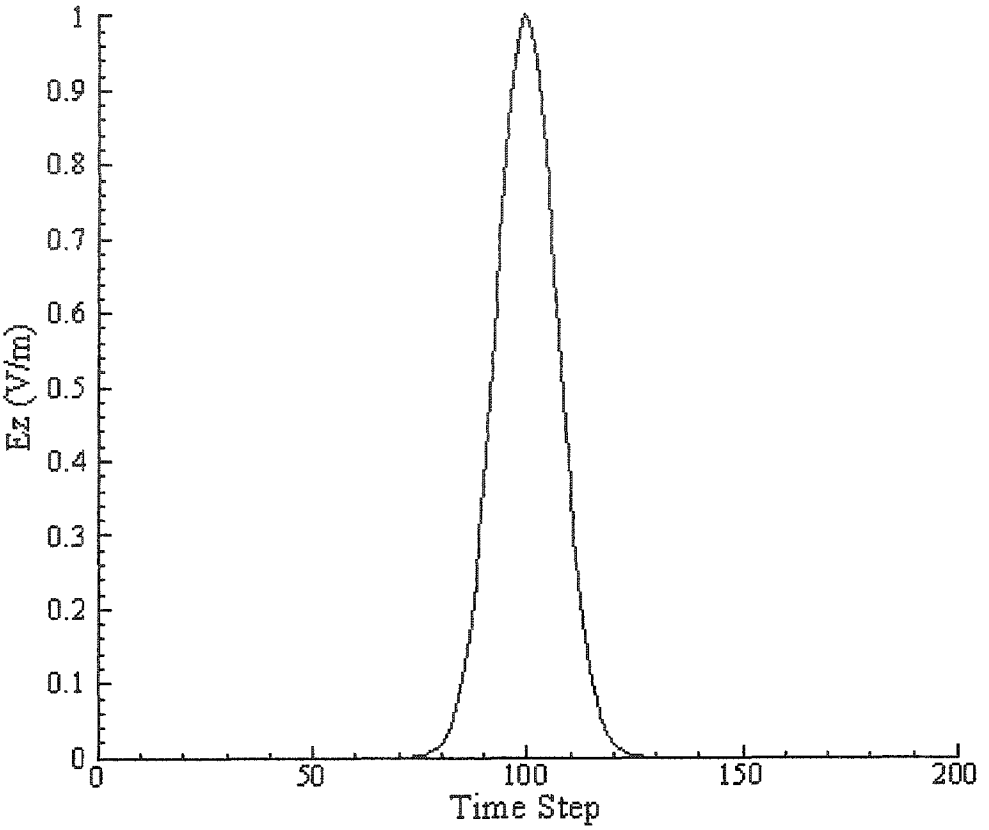
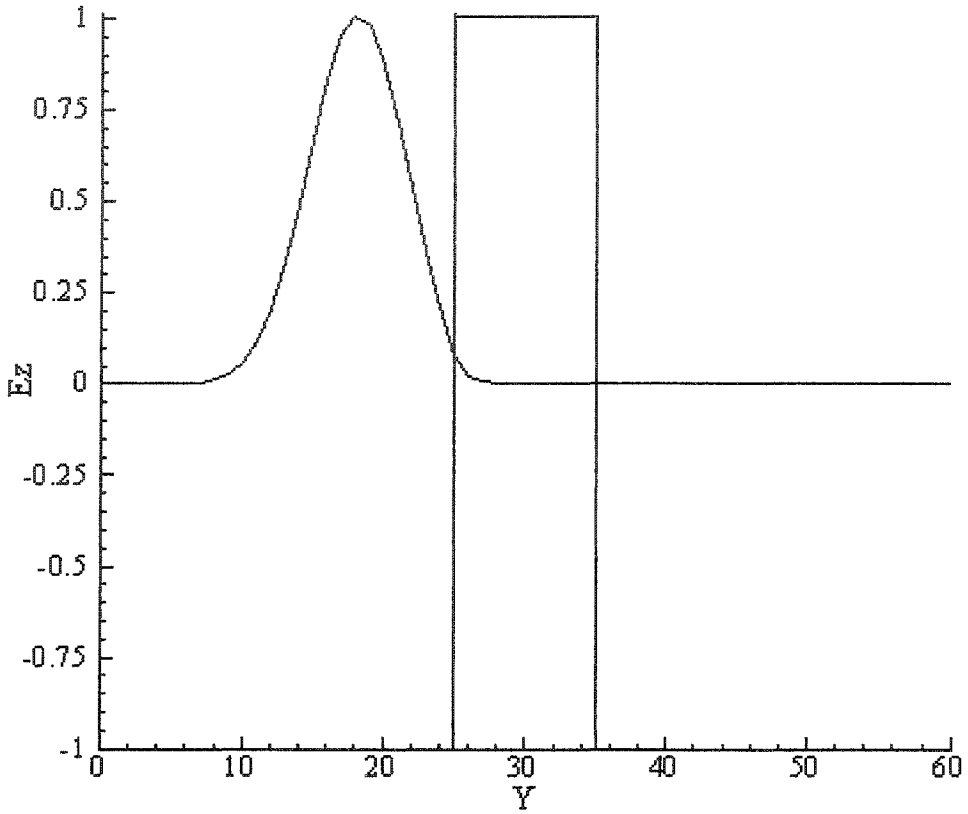
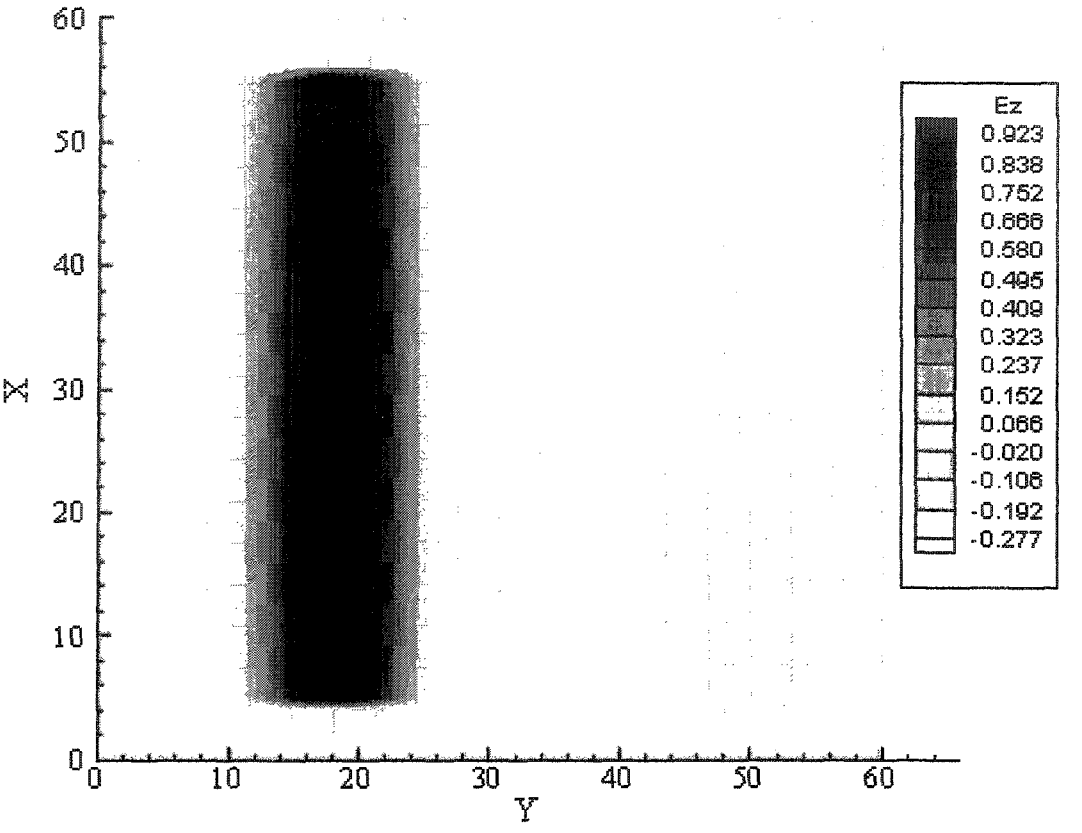


Figure 5.3 Pulse shape of $E_z = e^{\frac{(100-n)^2}{100}} V/m$



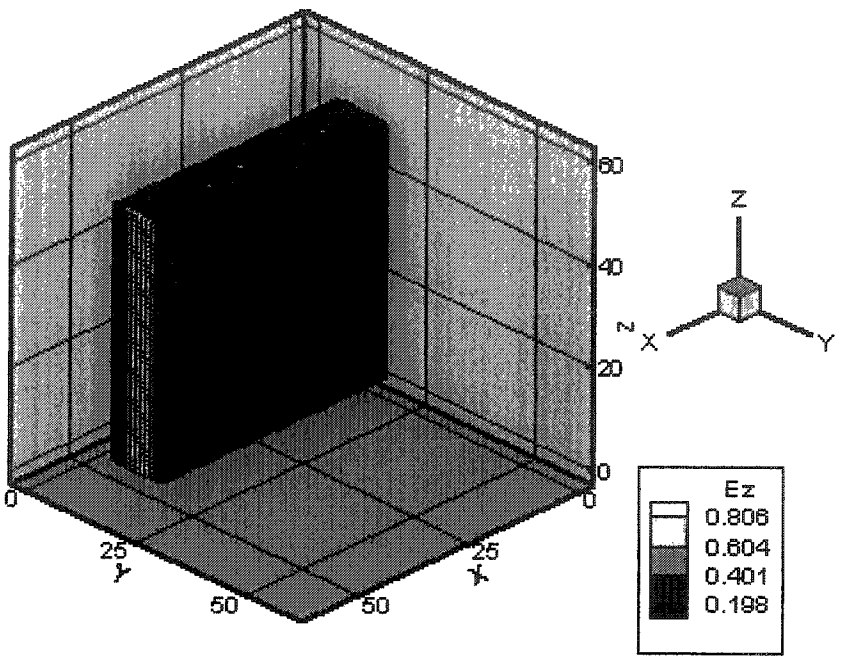
(a)

Figure 5.4.—(a), (b), (c) Snapshots of E_z versus position for the pulse at 130 time steps in: (a) one-dimensional space; (b) two-dimensional space; (c) three-dimensional space.



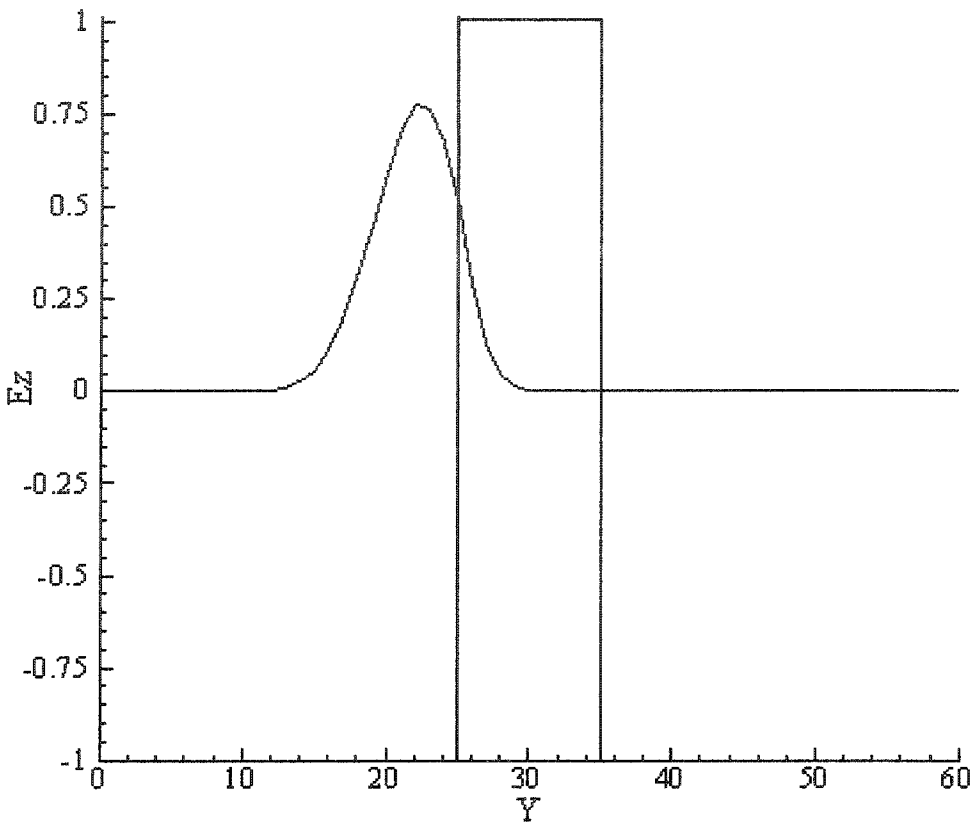
(b)

(Figure 5.4, Continued)



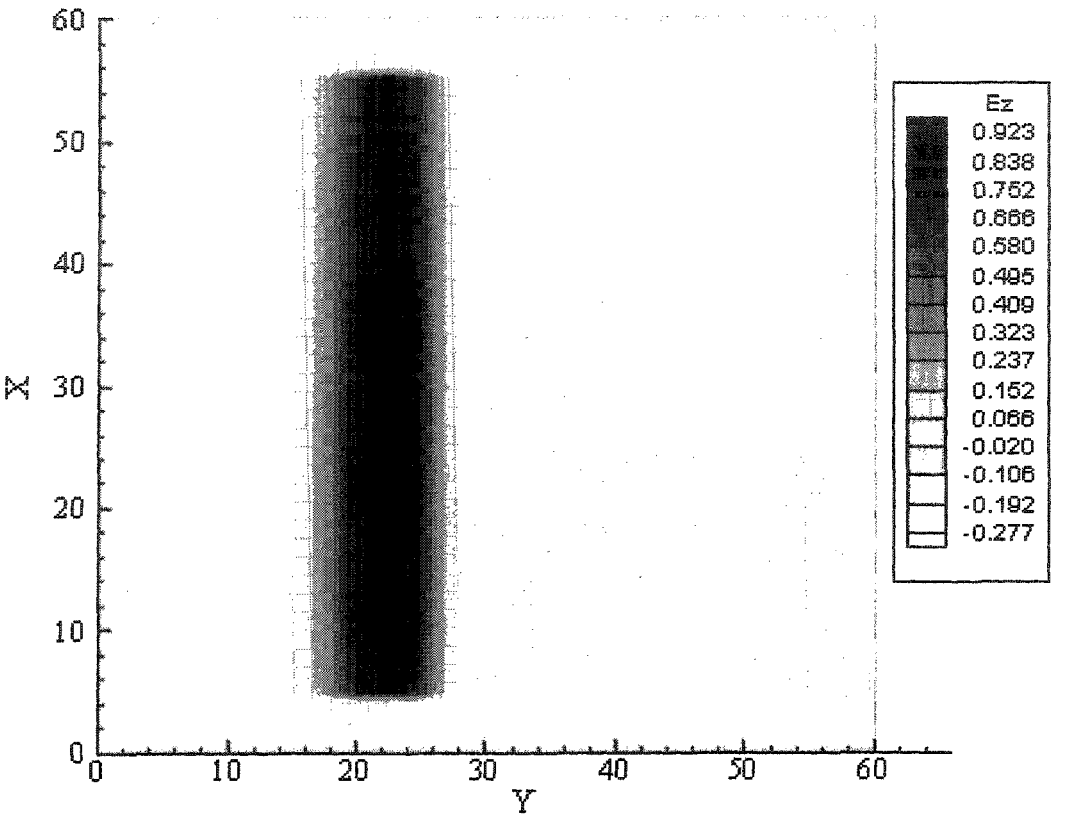
(c)

(Figure 5.4, Continued)



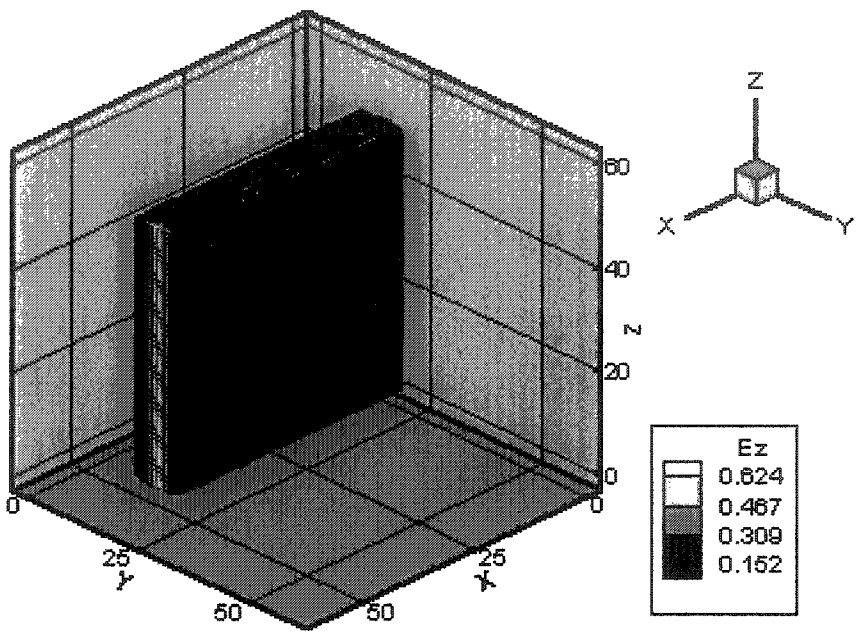
(a)

Figure 5.5.—(a), (b), (c) Snapshots of E_z versus position for the pulse at 140 time steps in: (a) one-dimensional space; (b) two-dimensional space; (c) three-dimensional space.



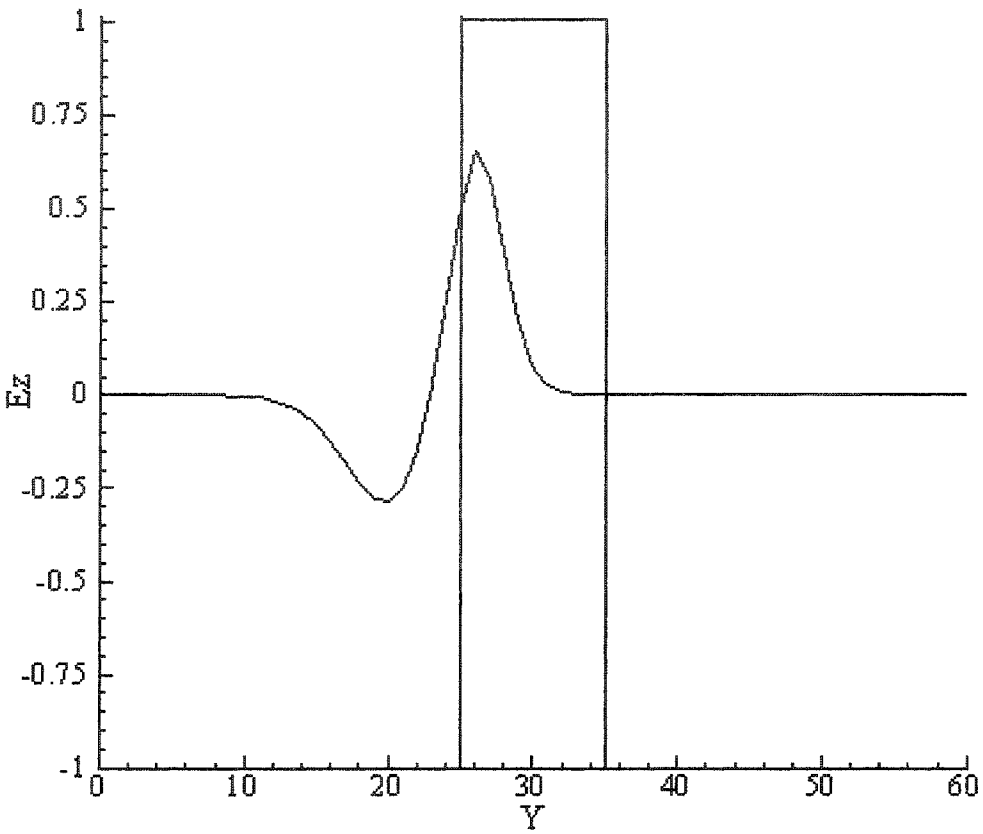
(b)

(Figure 5.5, Continued)



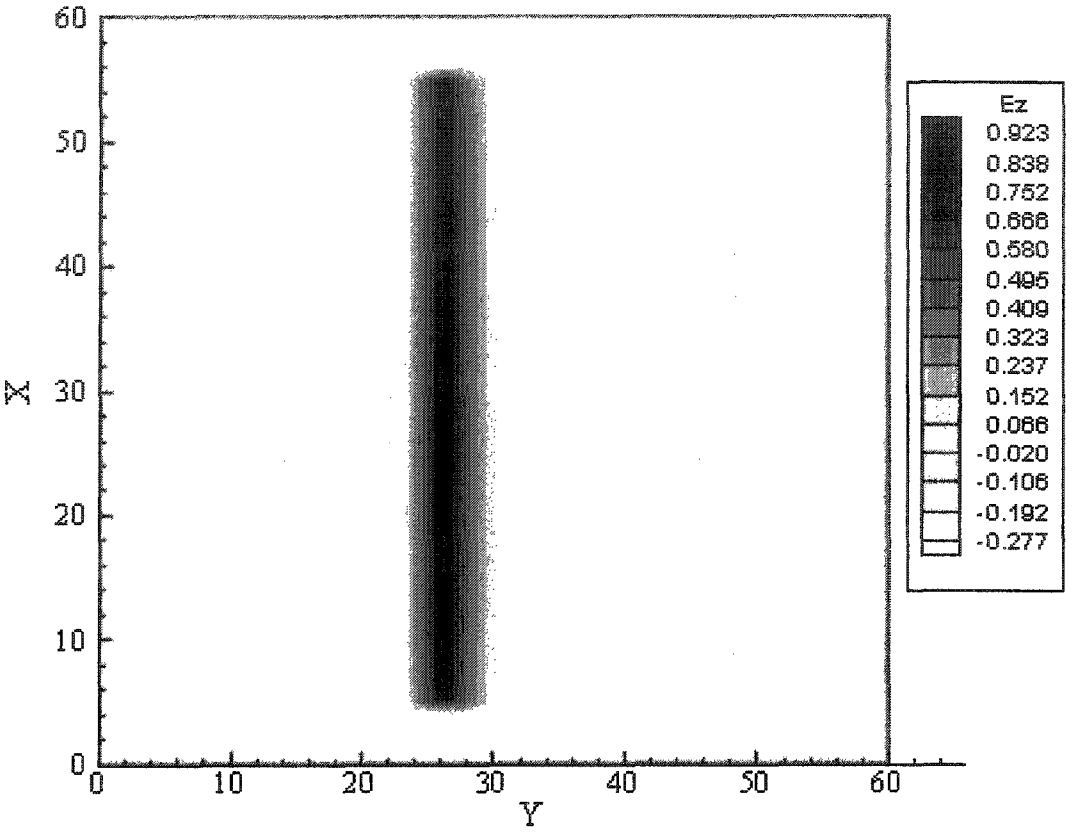
(c)

(Figure 5.5, Continued)



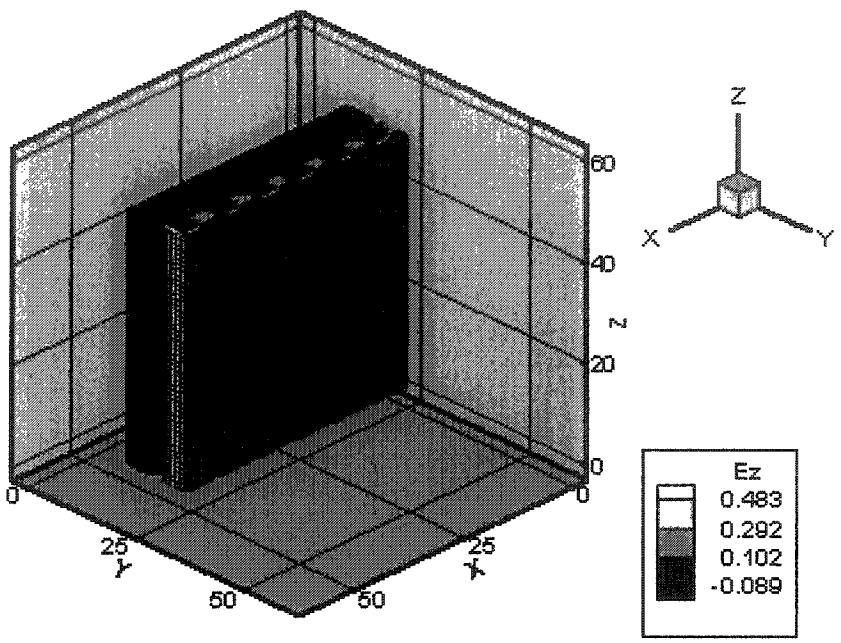
(a)

Figure 5.6.—(a), (b), (c) Snapshots of E_z versus position for the pulse at 150 time steps in: (a) one-dimensional space; (b) two-dimensional space; (c) three-dimensional space.



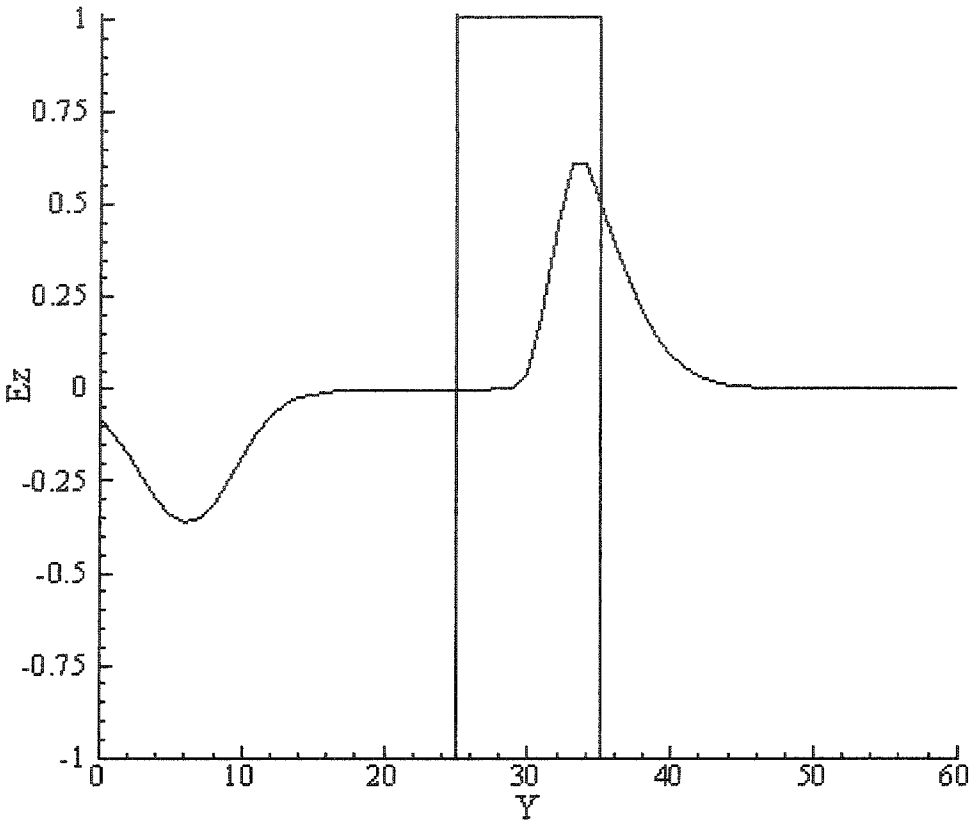
(b)

(Figure 5.6, Continued)



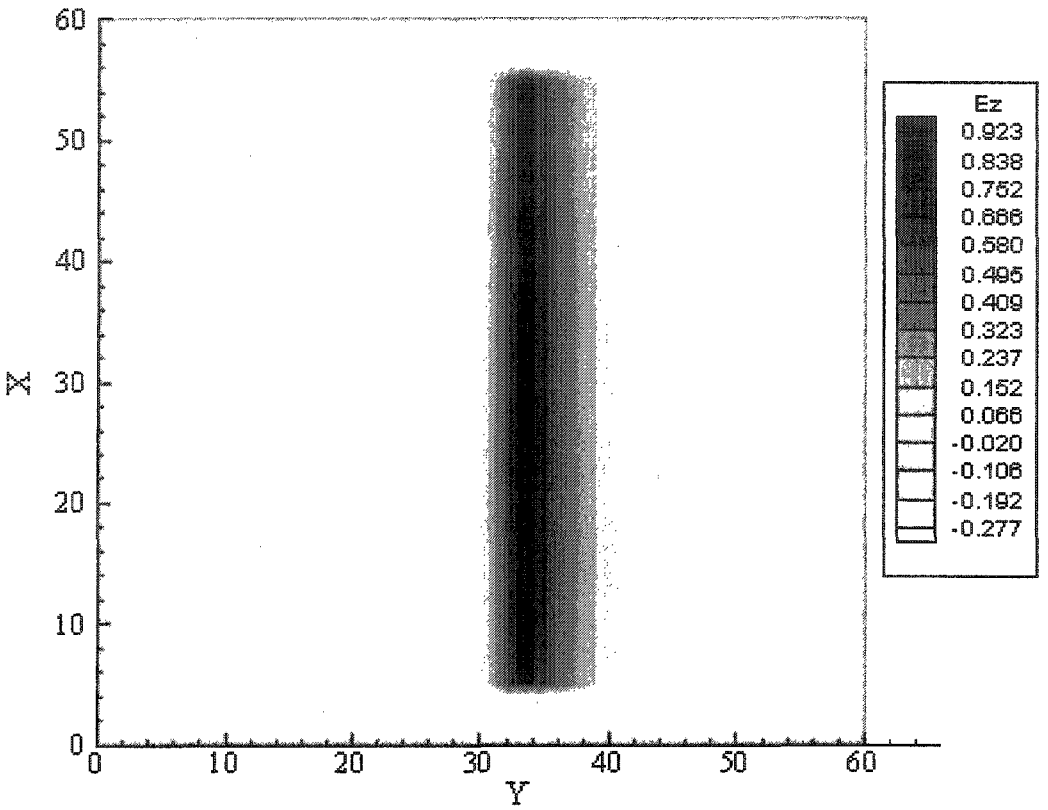
(c)

(Figure 5.6, Continued)



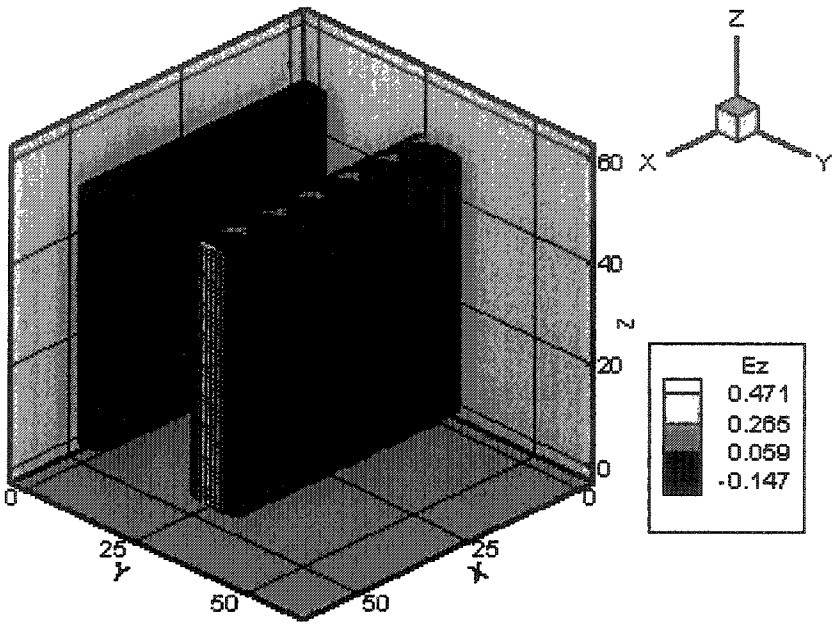
(a)

Figure 5.7.—(a), (b), (c) Snapshots of E_z versus position for the pulse at 180 time steps in: (a) one-dimensional space; (b) two-dimensional space; (c) three-dimensional space.



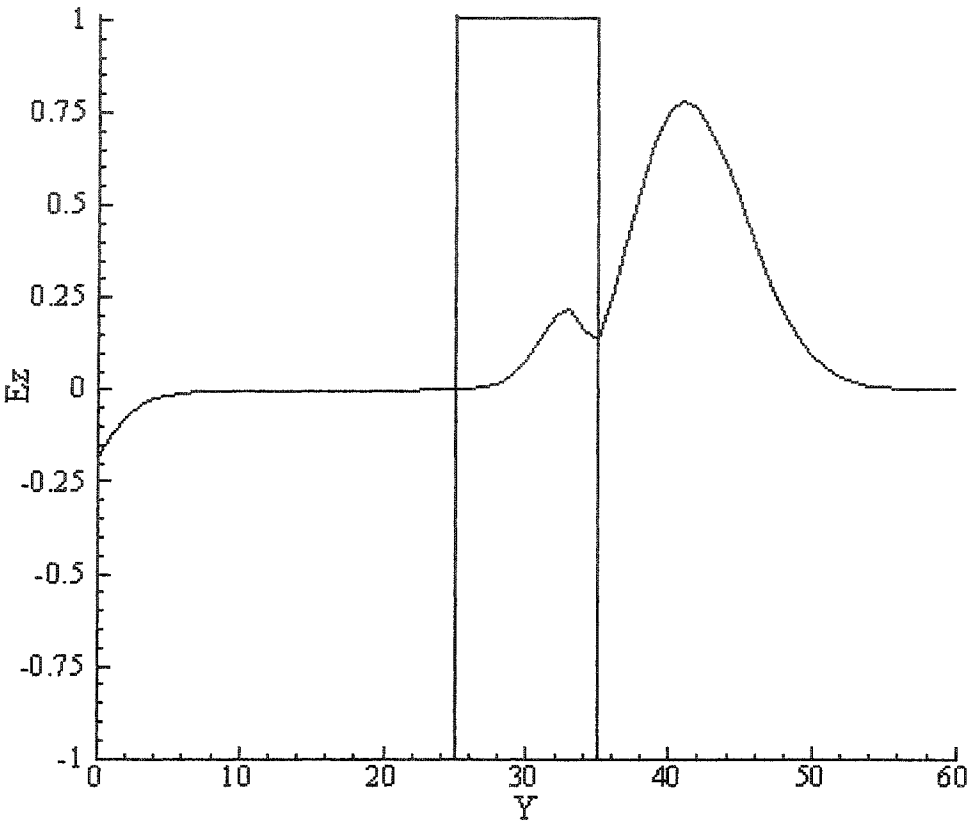
(b)

(Figure 5.7, Continued)



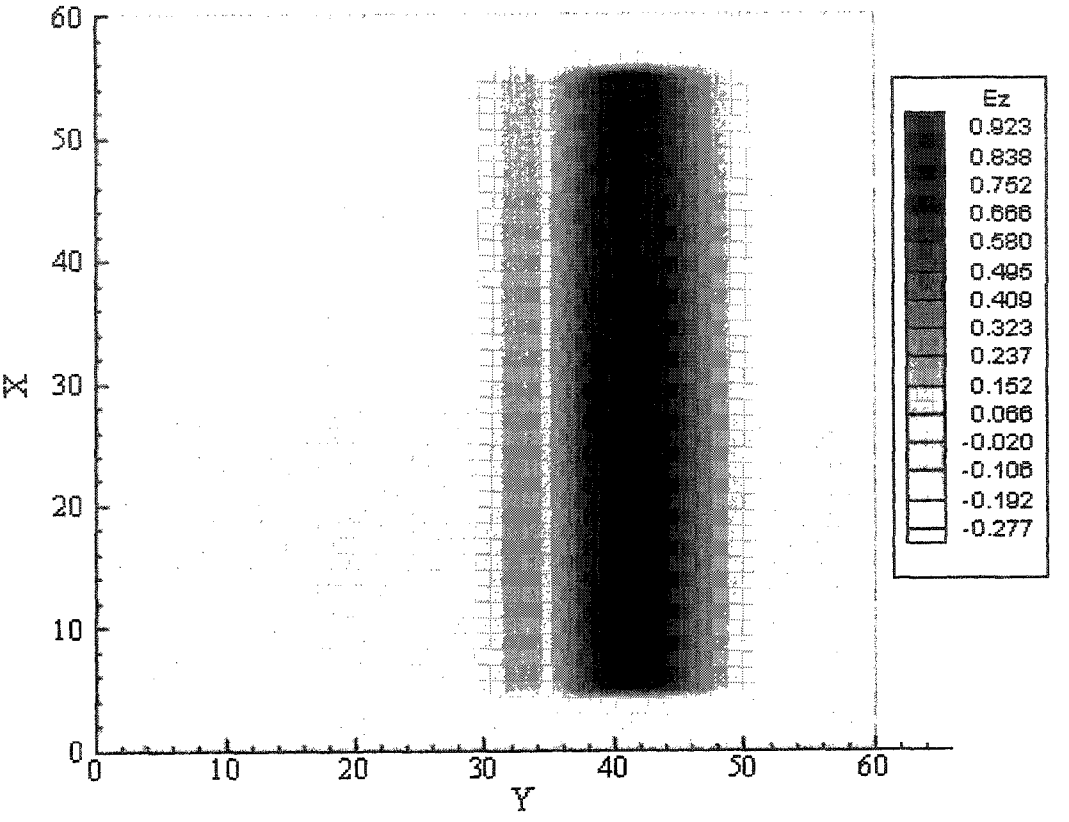
(c)

(Figure 5.7, Continued)



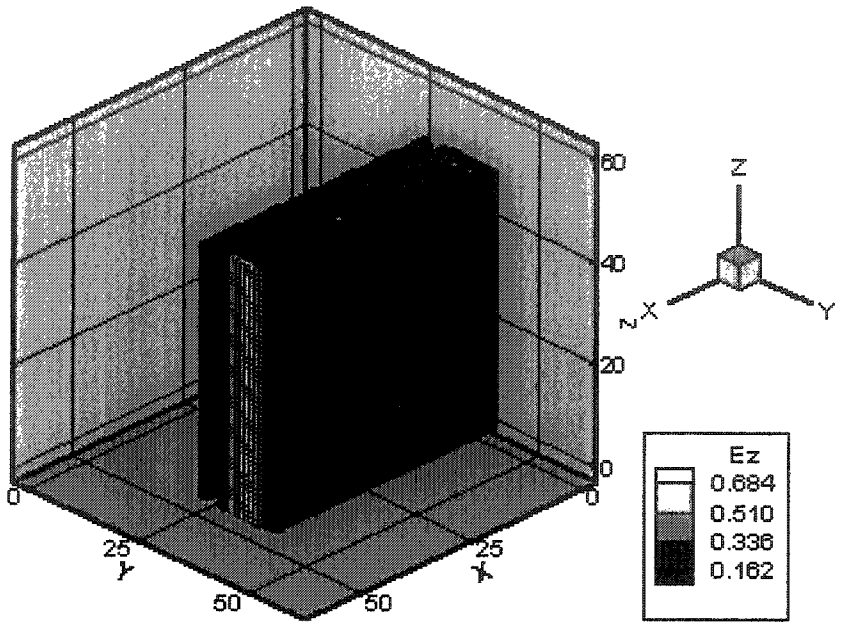
(a)

Figure 5.8.—(a), (b), (c) Snapshots of E_z versus position for the pulse at 200 time steps in: (a) one-dimensional space; (b) two-dimensional space; (c) three-dimensional space.



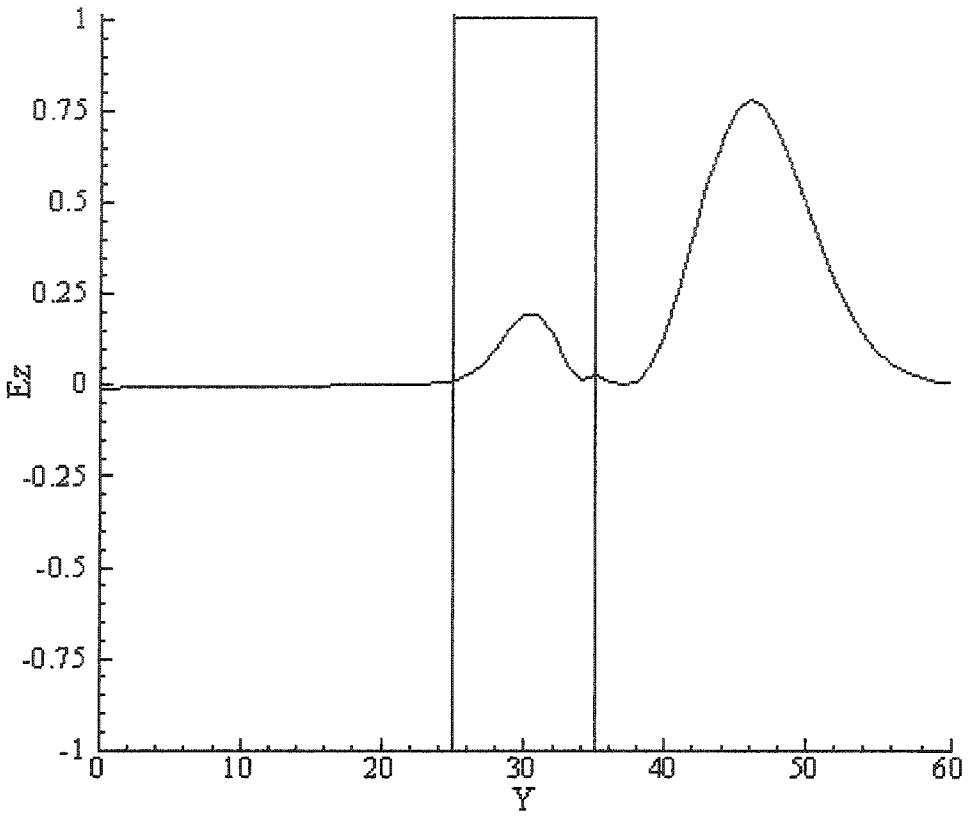
(b)

(Figure 5.8, Continued)



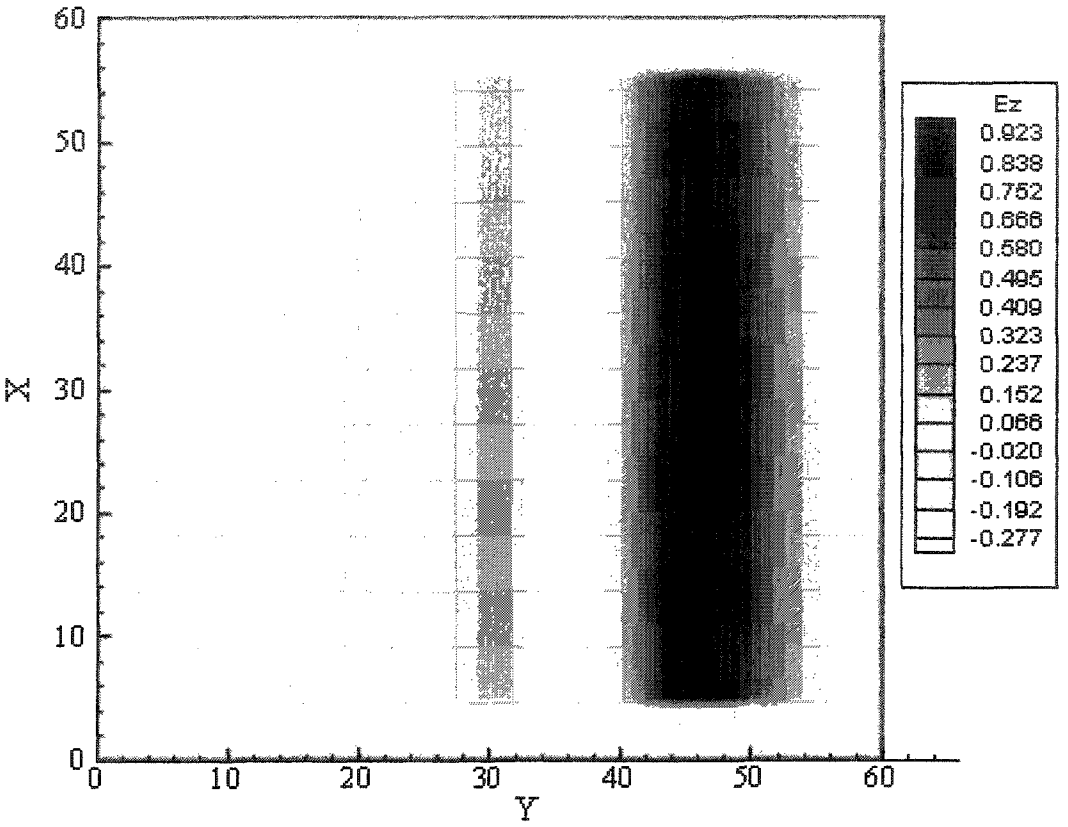
(c)

(Figure 5.8, Continued)



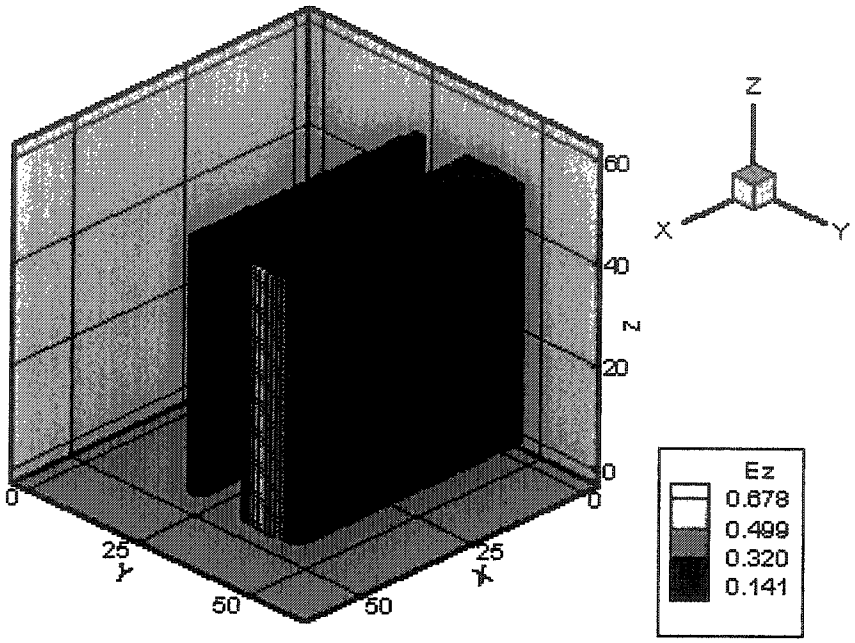
(a)

Figure 5.9.—(a), (b), (c) Snapshots of E_z versus position for the pulse at 210 time steps in: (a) one-dimensional space; (b) two-dimensional space; (c) three-dimensional space.



(b)

(Figure 5.9, Continued)



(c)

(Figure 5.9, Continued)

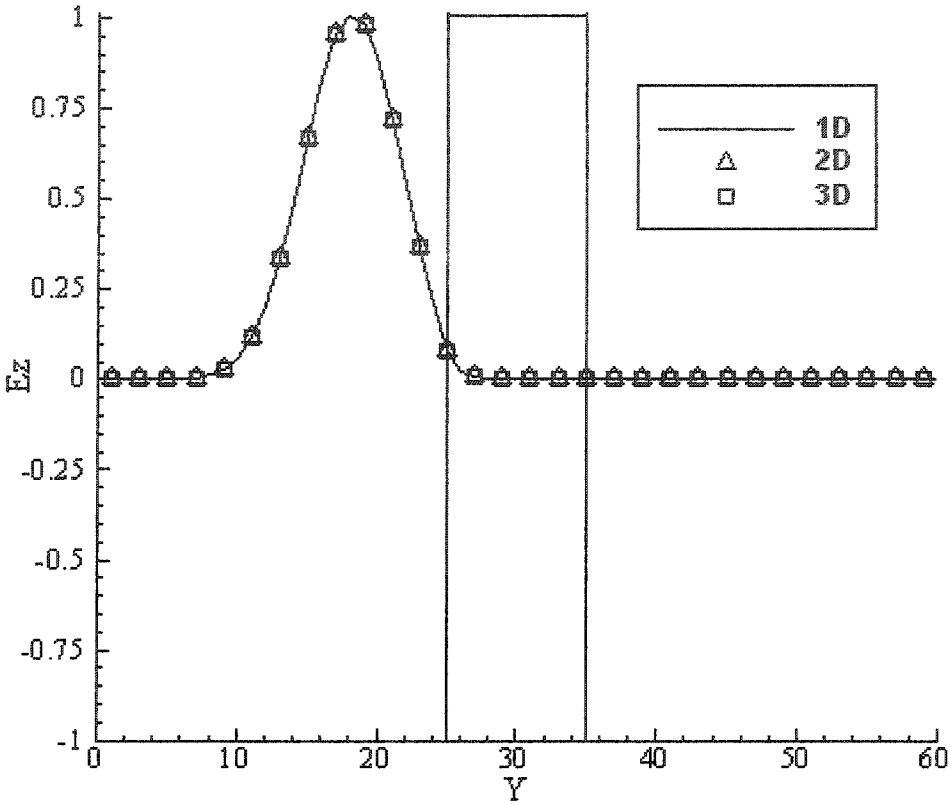
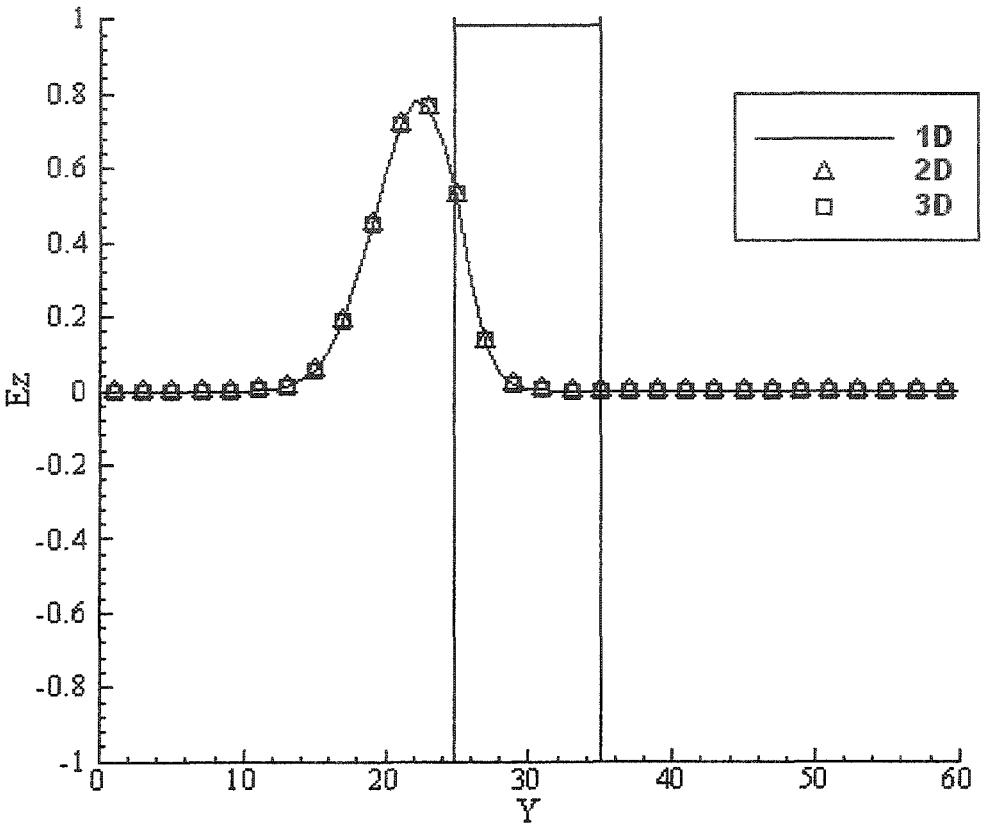


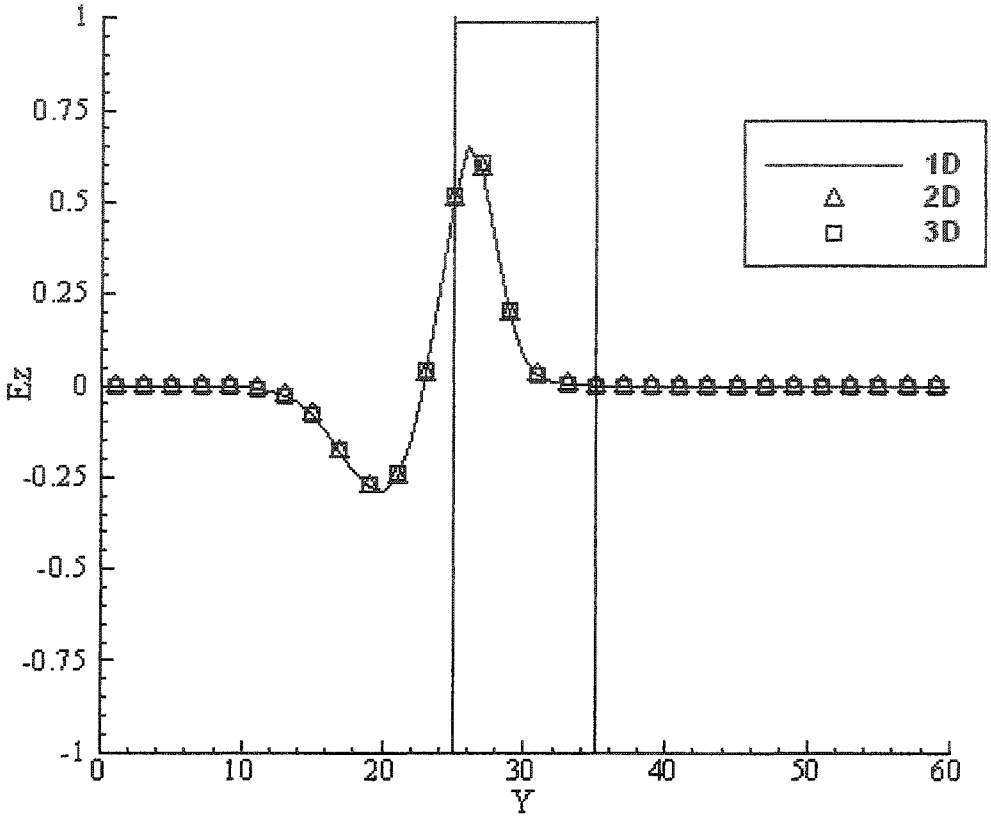
Figure 5.10(a) 130 Timeteps

Figure 5.10.—(a)-(f) A sequence of snapshots of the values of E_z versus y position on the symmetric axis in 1D, 2D and 3D at different time steps: (a) 130 time steps; (b) 140 time steps; (c) 150 time steps; (d) 180 time steps; (e) 200 time steps; (f) 210 time steps.



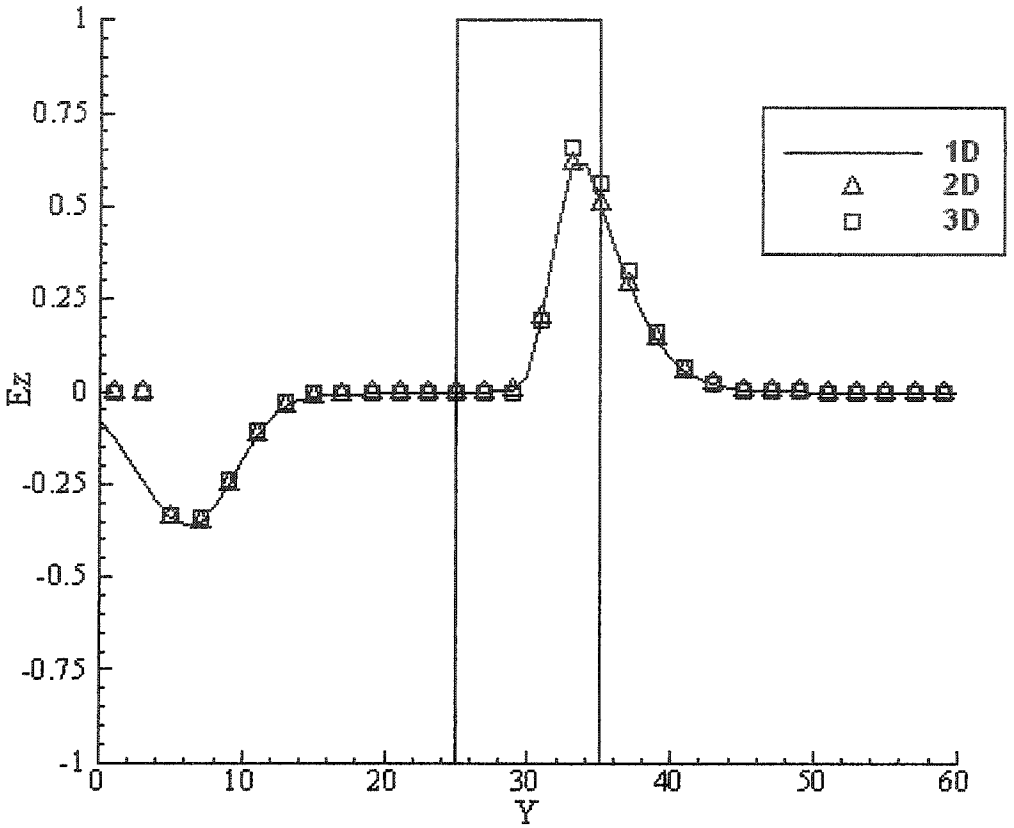
(b)

(Figure 5.10, Continued)



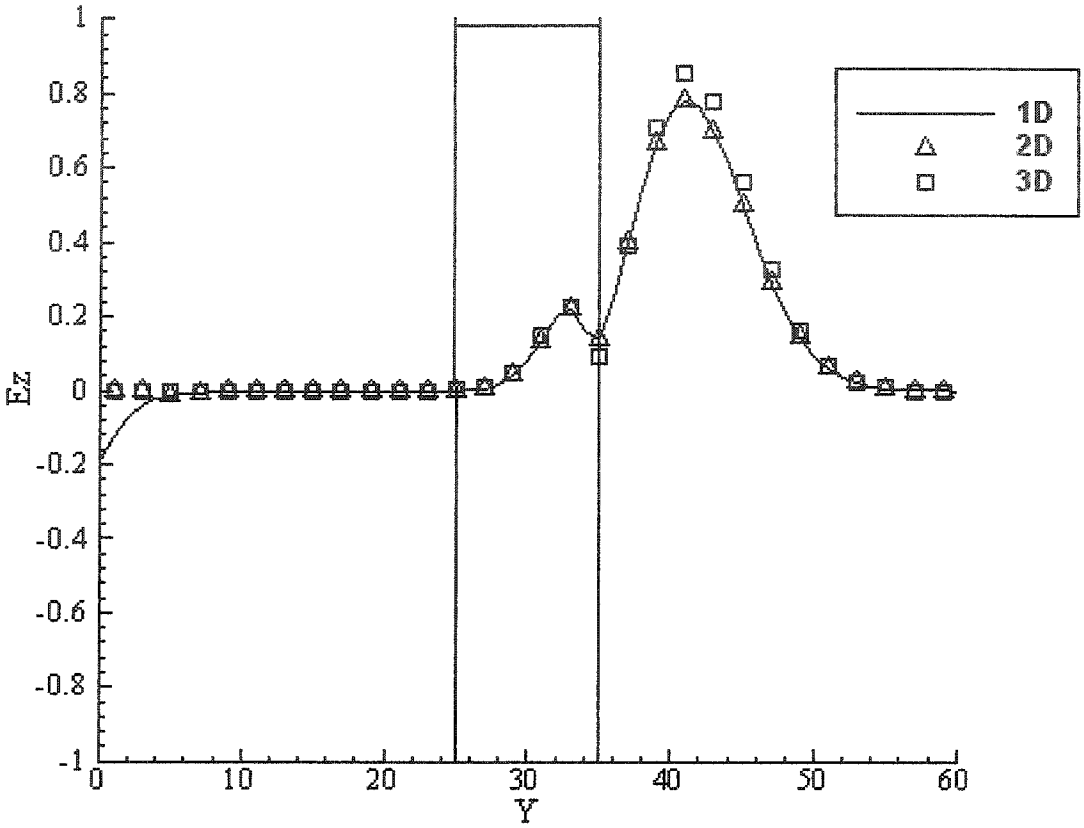
(c)

(Figure 5.10, Continued)



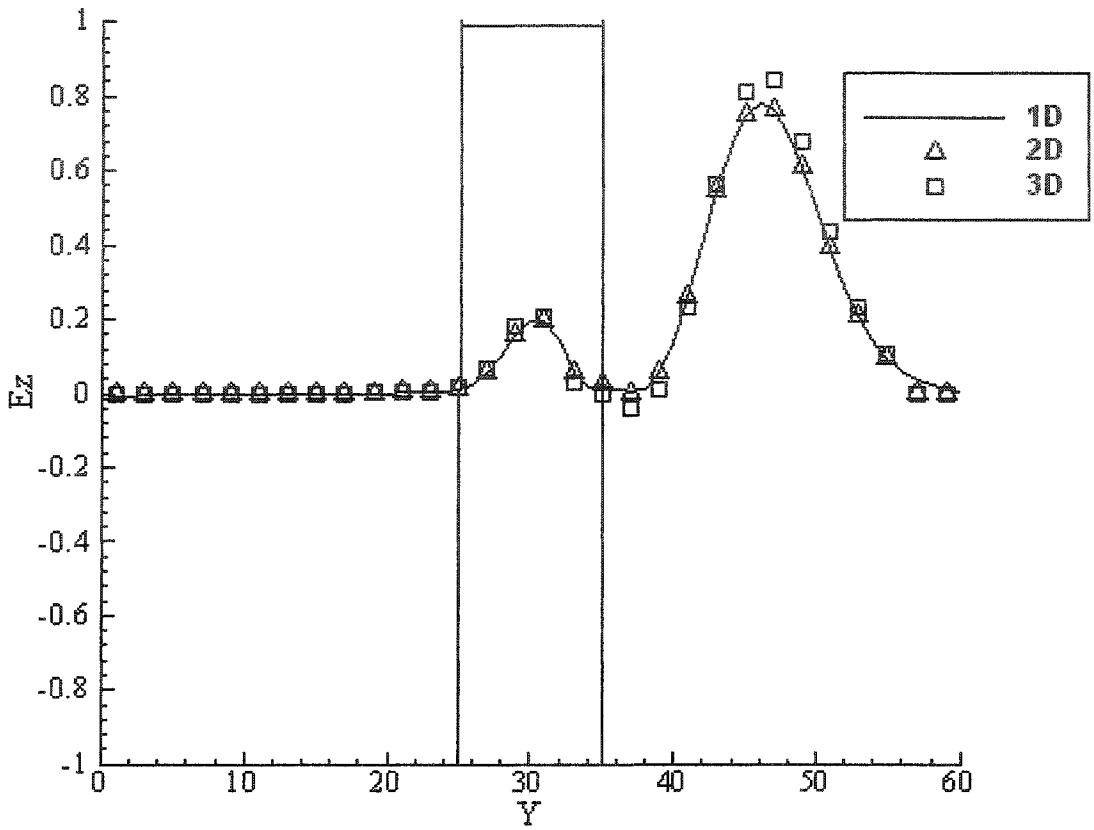
(d)

(Figure 5.10, Continued)



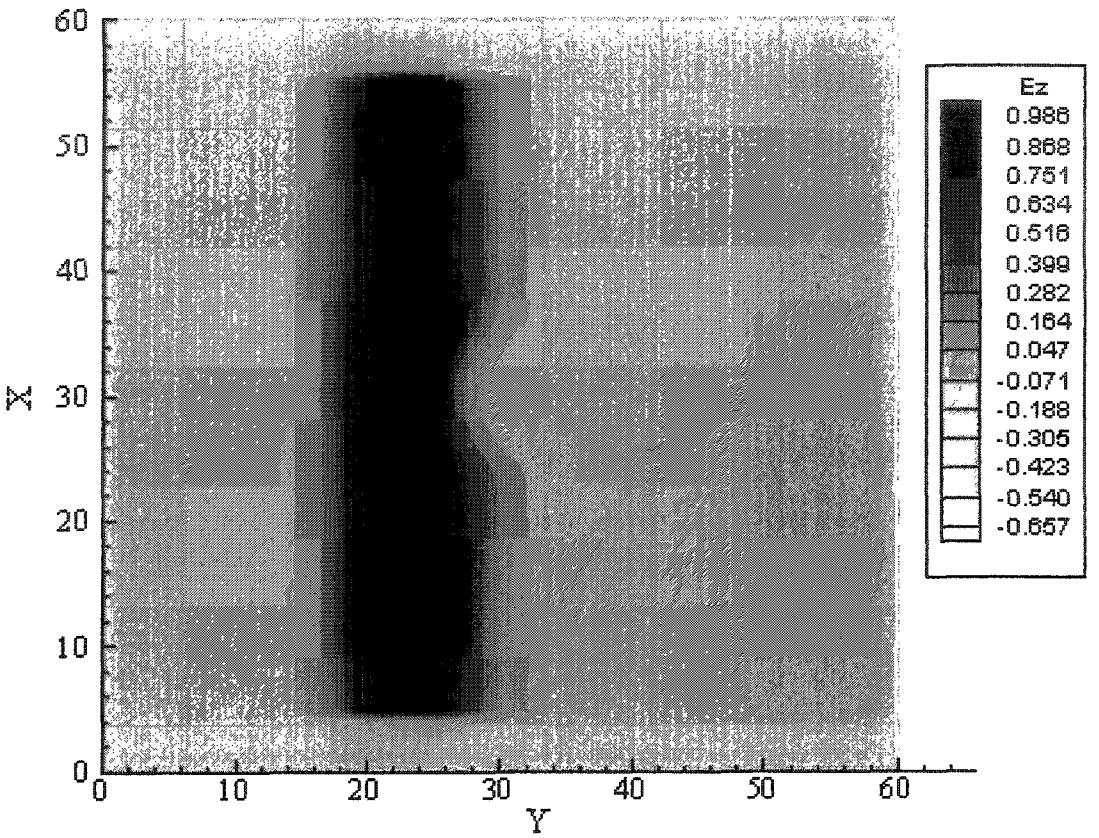
(e)

(Figure 5.10, Continued)



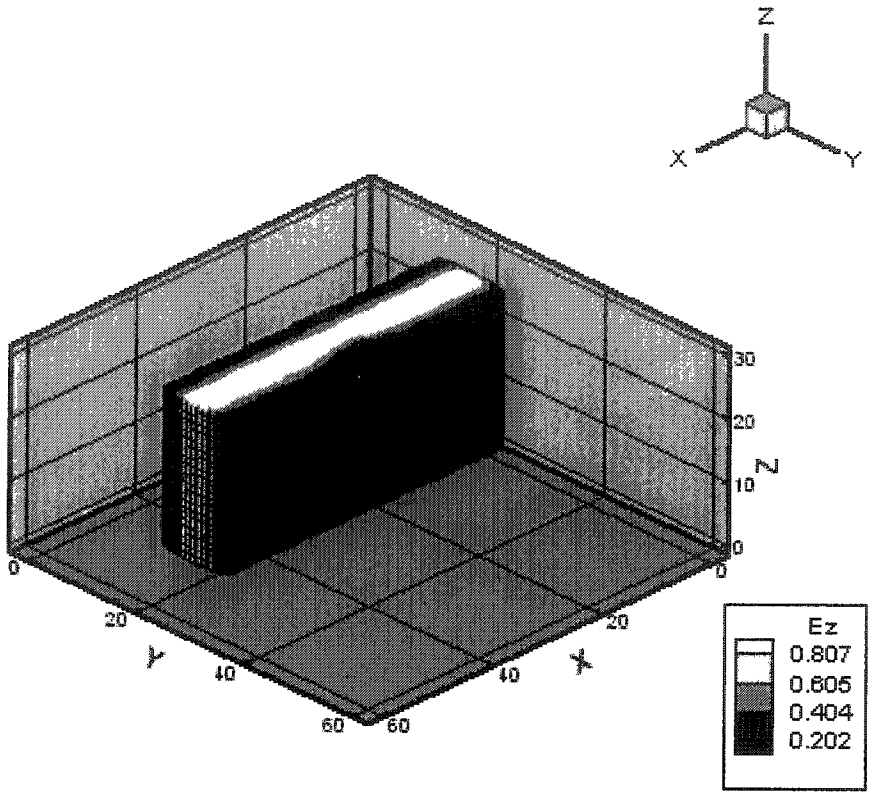
(f)

(Figure 5.10, Continued)



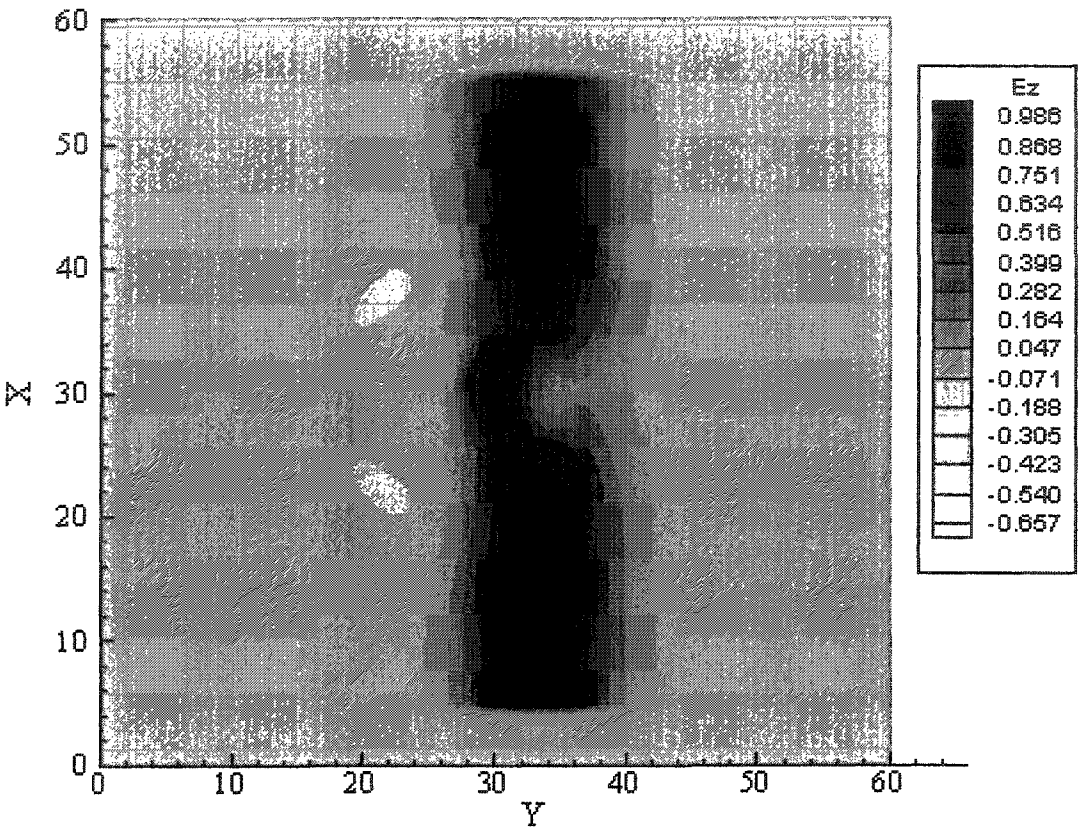
(a)

Figure 5.11.—(a), (b) Contours of E_z at 140 time steps in: (a) the xy cross section at $z = 30$; (b) the half cube where $z = 0 \sim 30$.



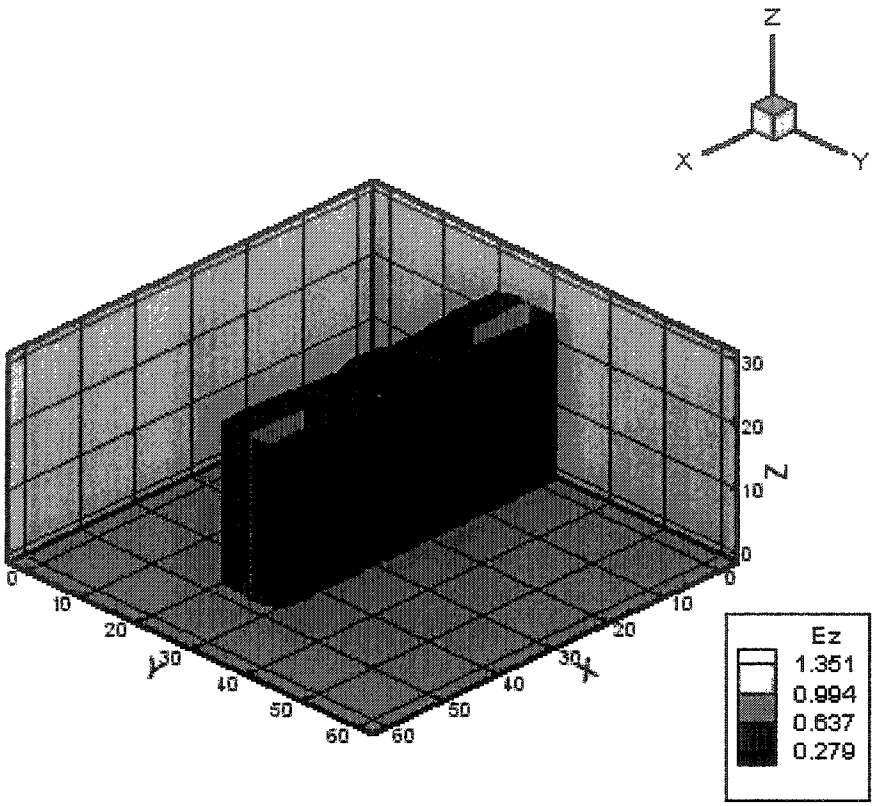
(b)

(Figure 5.11, Continued)



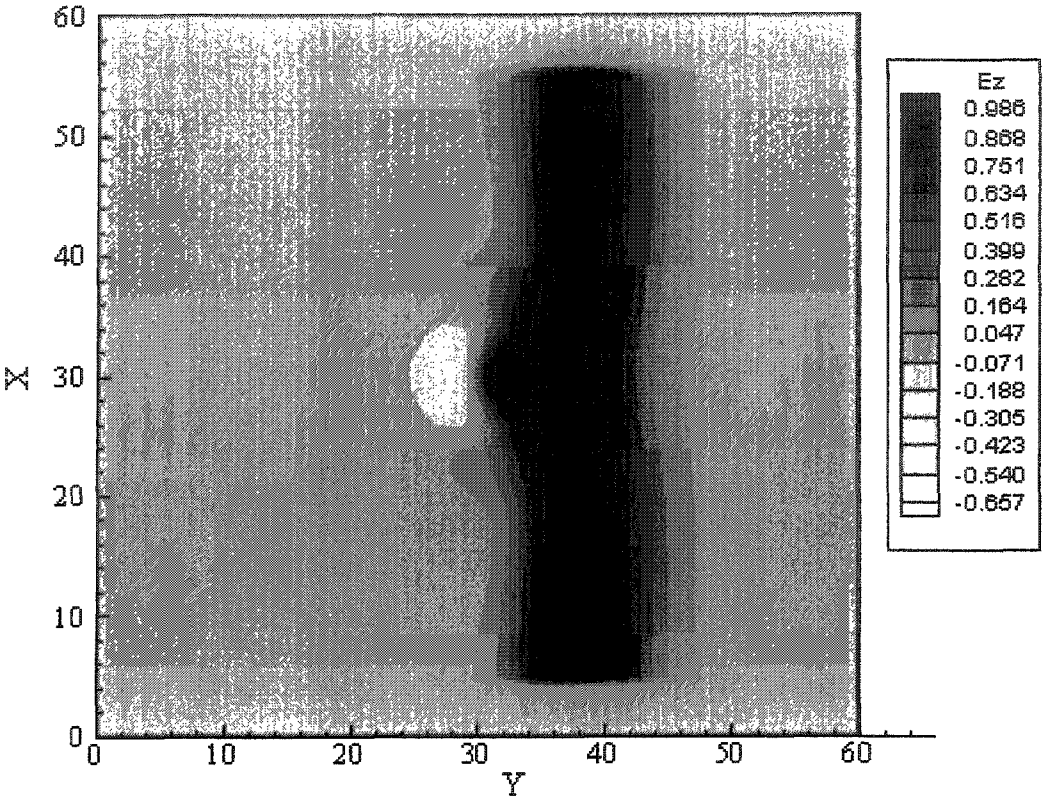
(a)

Figure 5.12.—(a), (b) Contours of E_z at 160 time steps in: (a) the xy cross section at $z = 30$; (b) the half cube where $z = 0 \sim 30$.



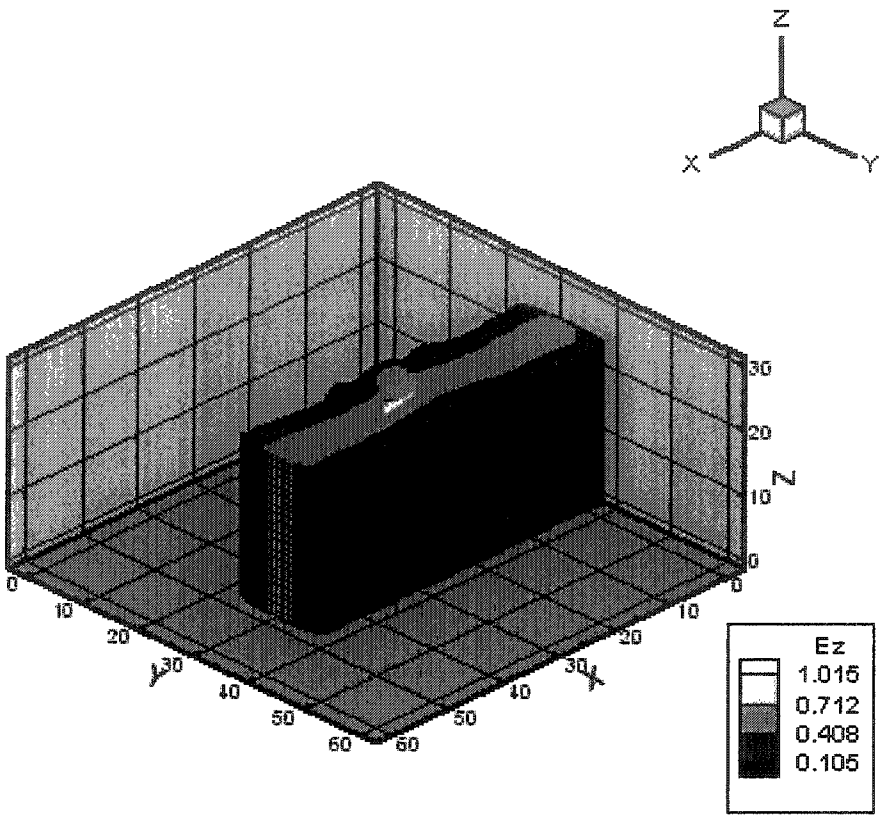
(b)

(Figure 5.12, Continued)



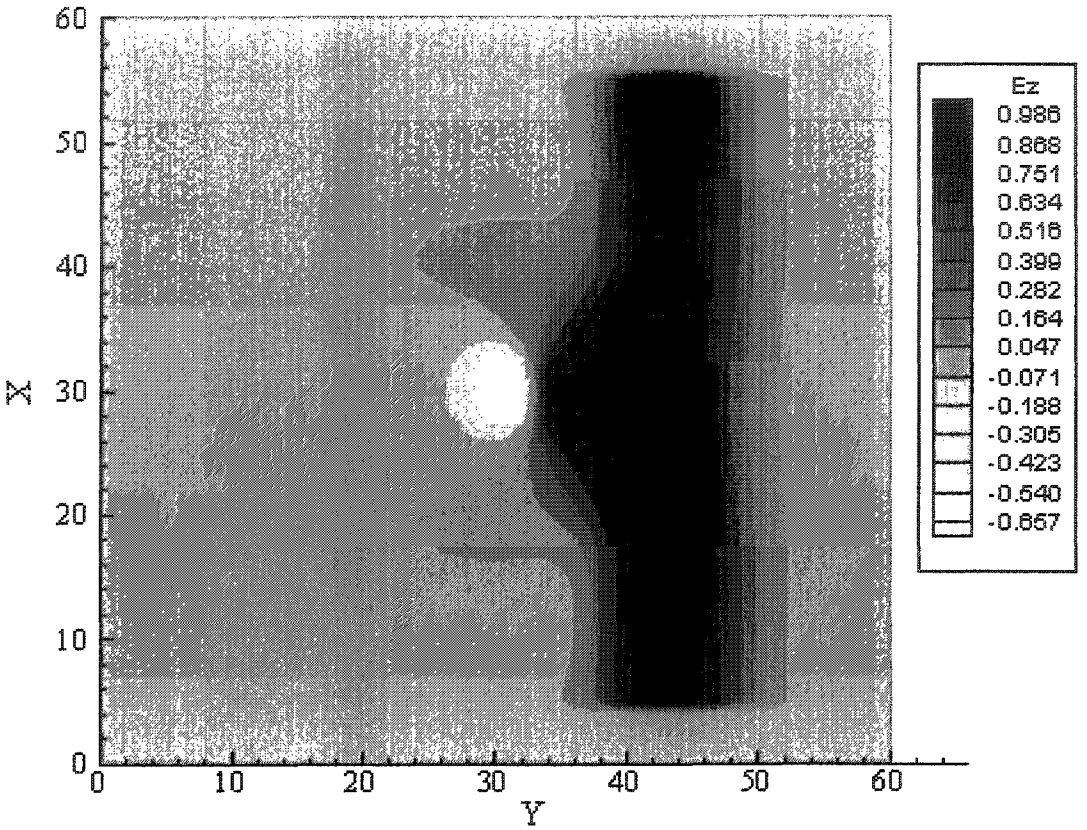
(a)

Figure 5.13.—(a), (b) Contours of E_z at 170 time steps in: (a) the xy cross section at $z = 30$; (b) the half cube where $z = 0 \sim 30$.



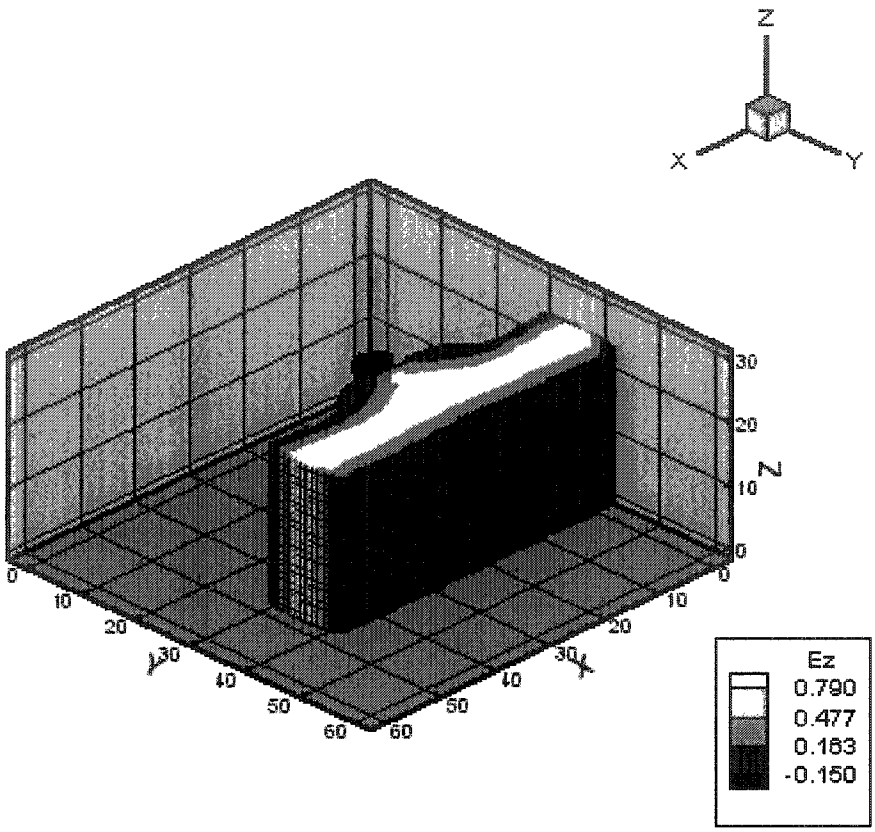
(b)

(Figure 5.13, Continued)



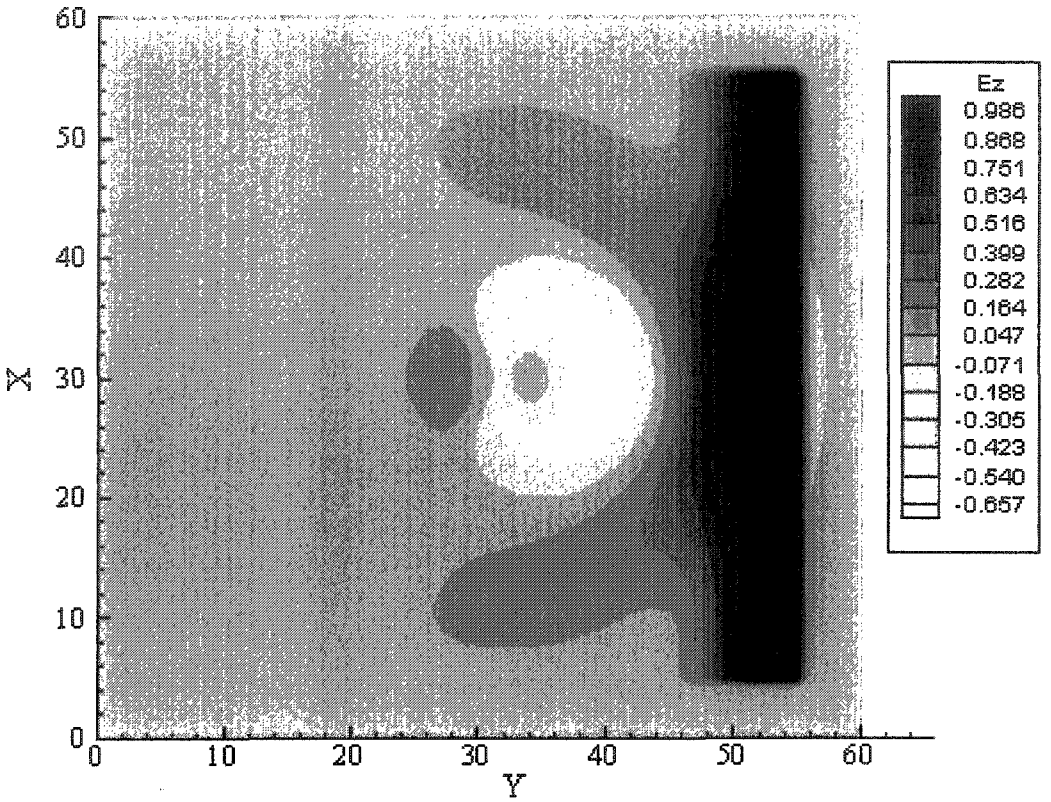
(a)

Figure 5.14.—(a), (b) Contours of E_z at 180 time steps in: (a) the xy cross section at $z = 30$; (b) the half cube where $z = 0 \sim 30$.



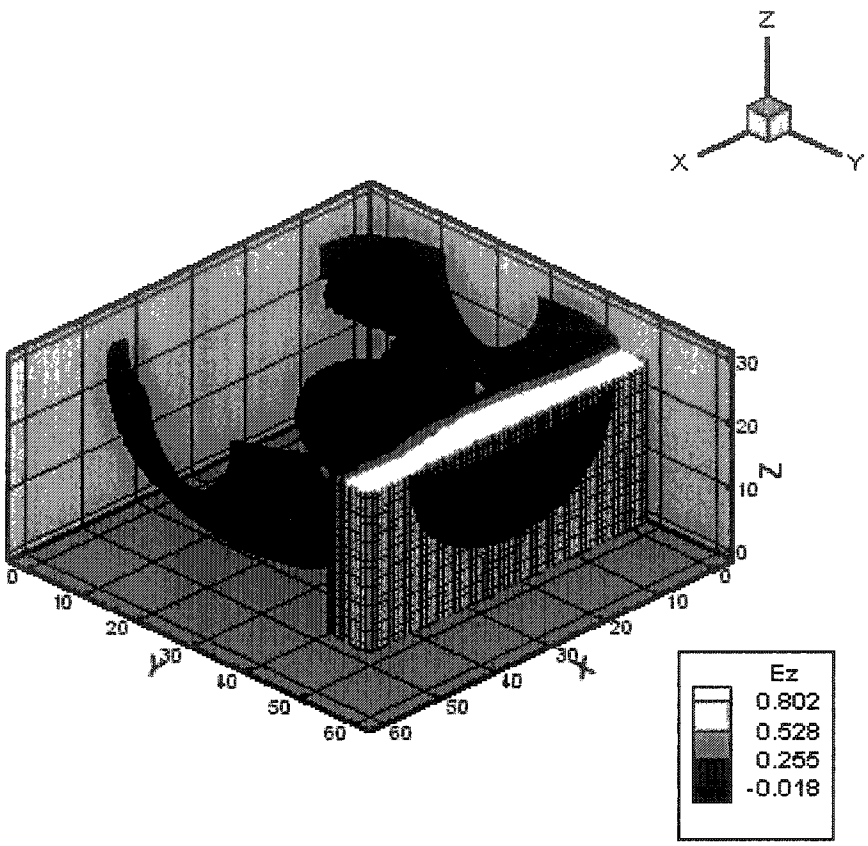
(b)

(Figure 5.14, Continued)



(a)

Figure 5.15.—(a), (b) Contours of E_z at 200 time steps in: (a) the xy cross section at $z = 30$; (b) the half cube where $z = 0 \sim 30$.



(b)

(Figure 5.15, Continued)

5.3 Penetration of a Pulse

We have further studied application of the new method to penetration of a Gaussian pulse into biological matter. The grid size was $\Delta x = 0.1mm$ and the time increment $\Delta t = \frac{\Delta x}{2c_0} = 1.6667 \times 10^{-13} s$. The pulses were chosen as follows:

$$E_z = 10^5 e^{-\frac{(5w-n)^2}{w \times w}} V/m \quad , \tag{5.1}$$

where w is the width of the pulse set to a value between $100\Delta t$ and $2000\Delta t$. A uniform dielectric slab of thickness $1mm - 1cm$ was used to model biological tissues. The parameters for the five different cases are shown in Table 5.2. Transmission, reflection, and absorption were calculated as a function of pulse width for each case. Figures 5.16-5.20 show the energy rate versus the pulse width.

In case 1 the material had $\varepsilon_\infty = 4.0$ but was not dispersive. Consequently no energy was absorbed. Figure 5.16 shows how transmission and reflection varied with pulse width and slab thickness. Most of the energy passed through the slab. Penetration, however, increased with pulse width and decrease with slab thickness. Reflection was simply the complement of transmission; no energy is absorbed by a non-dispersive medium.

In case 2 the material was dispersive and had $\varepsilon_\infty = 4.0$ and $\sigma_s = 1.0$; the remaining dielectric properties were 0. Figure 5.17 shows transmission, reflection, and absorption as a function of pulse width and slab thickness. Evidently the extent of penetration decreased with slab thickness. Reflection increased with slab thickness, regardless of pulse width. For larger thickness of slab, it increased with pulse width

while for the smaller ones it showed an opposite behavior. On the contrary, absorption increased with pulse width for thinner slab while behaved oppositely for the thicker one.

In case 3 the dielectric slab had the properties of mammalian blood. One can see from Figure 5.18 that transmission decreased with increasing in slab thickness. The portion of the initial pulse energy penetrated the thick slab was relatively so small that it may hint an insufficiency to treat blood-borne pathogens. For reflection and absorption, the shapes of curves changed with the slab thickness. Whether it is caused by a real physical effect or numerical influence is still unknown at this time. Theoretical study is needed to explain these behaviors.

In case 4 the slab was considered to be sclera, i.e. the sclerotic coat of the mammalian eyeball. The results shown in Figure 5.19, are similar to those of case 3. The calculation may be useful for deciding whether eye protection should be worn in the vicinity of a nanopulse generator.

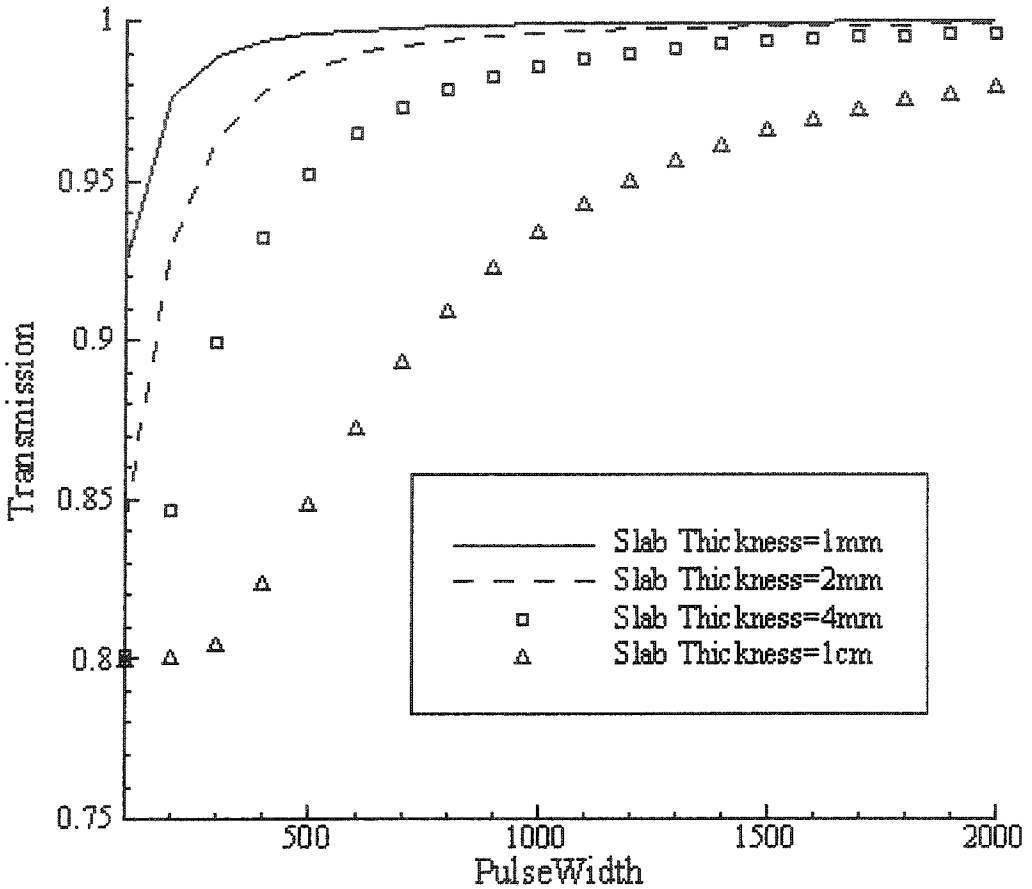
In case 5 the slab was modeled as breast fat. Figure 5.20 illustrates that a large portion of energy passed through the slab. Penetration decreased with slab thickness. A very small portion of the energy was reflected, and reflection increased with either pulse width or slab thickness. Absorption increased with either pulse width or slab thickness, in contrast to the behavior of blood and sclera. This analysis may be pertinent to interest in the use of UWB pulse in breast cancer detection [46].

As to the physics of the time-dependence of the applied pulse, it is not very clear why pulse width or rise time should result in bioeffects. Of course, the energy deposited in a sample will depend on pulse width, but when the pulse is so short, the temperature

rise will be insignificant [47]. Further experimental study is needed to elucidate mechanisms of non-thermal effects.

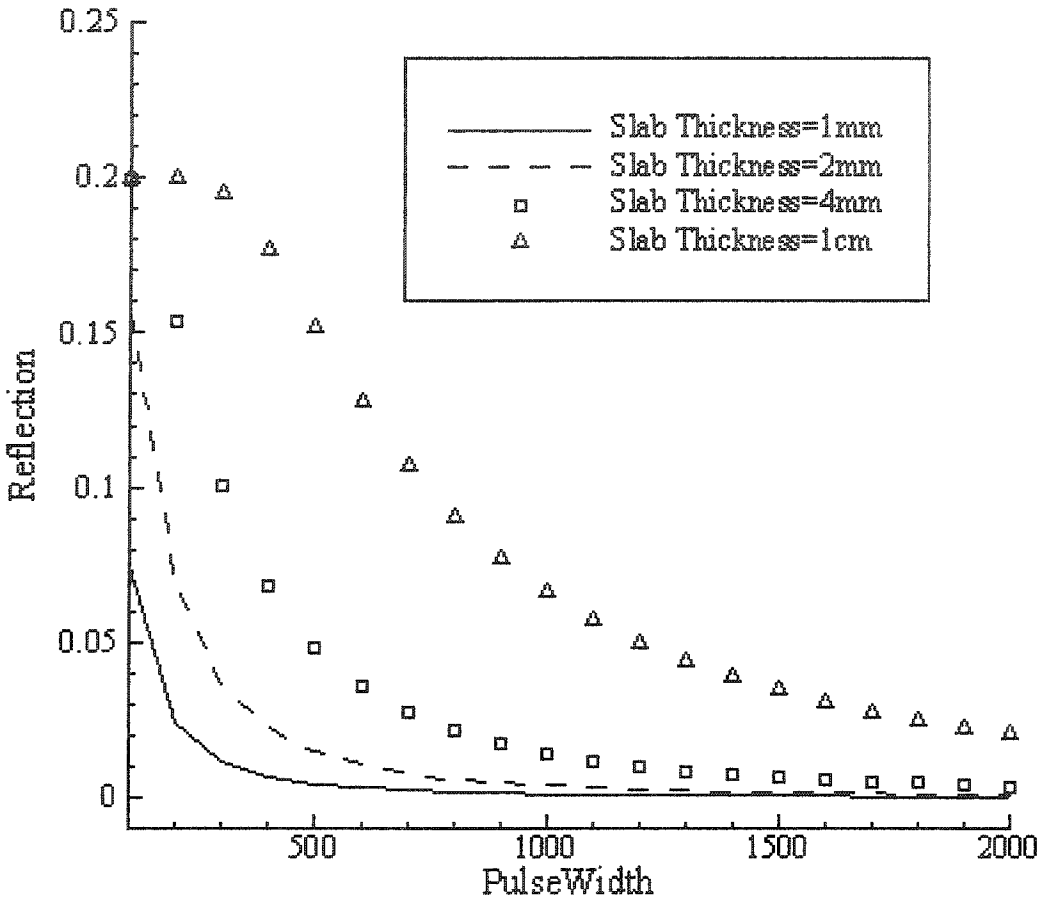
Table 5.2 Dielectric Properties [20, 21, 22]

	Case 1	Case 2	Case 3	Case 4	Case 5
	(Non-dispersive)	(Dispersive)	(Blood)	(Sclera)	(Breast Fat)
ε_∞	4.0	4.0	4.0	4.0	2.5
σ_s	0.0	1.0	0.7	0.5	0.01
$\Delta\varepsilon_1$	0	0	56	50	3
$\Delta\varepsilon_2$	0	0	5200	4000	15
$\Delta\varepsilon_3$	0	0	0	10^5	5×10^4
$\Delta\varepsilon_4$	0	0	0	5×10^6	5×10^7
$\tau_1(ps)$	0	0	8.377	7.958	17.680
$\tau_2(ns)$	0	0	132.629	159.155	63.660
$\tau_3(\mu s)$	0	0	159.155	159.155	454.700
$\tau_4(ms)$	0	0	15.915	15.915	13.260
α_1	0	0	0.1	0.1	0.1
α_2	0	0	0.1	0.1	0.1
α_3	0	0	0.2	0.2	0.1
α_4	0	0	0	0	0



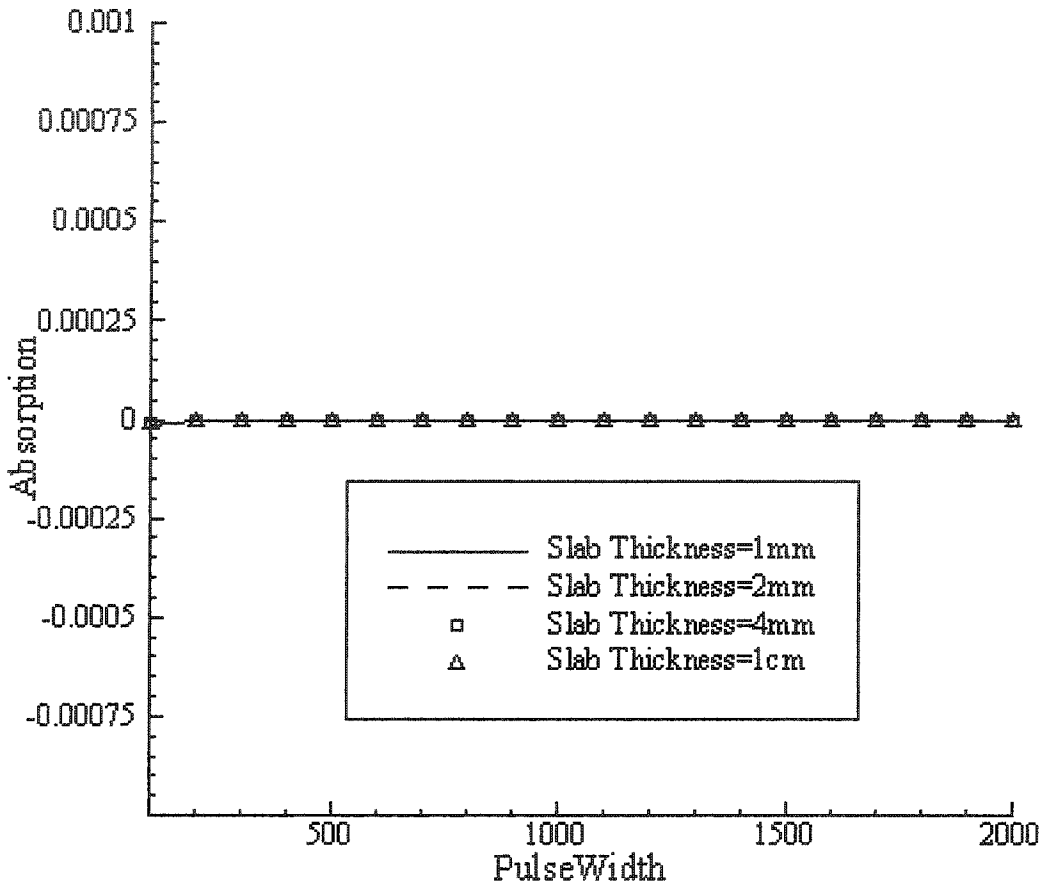
(a)

Figure 5.16.—(a), (b), (c) Transmission, reflection and absorption in case 1.



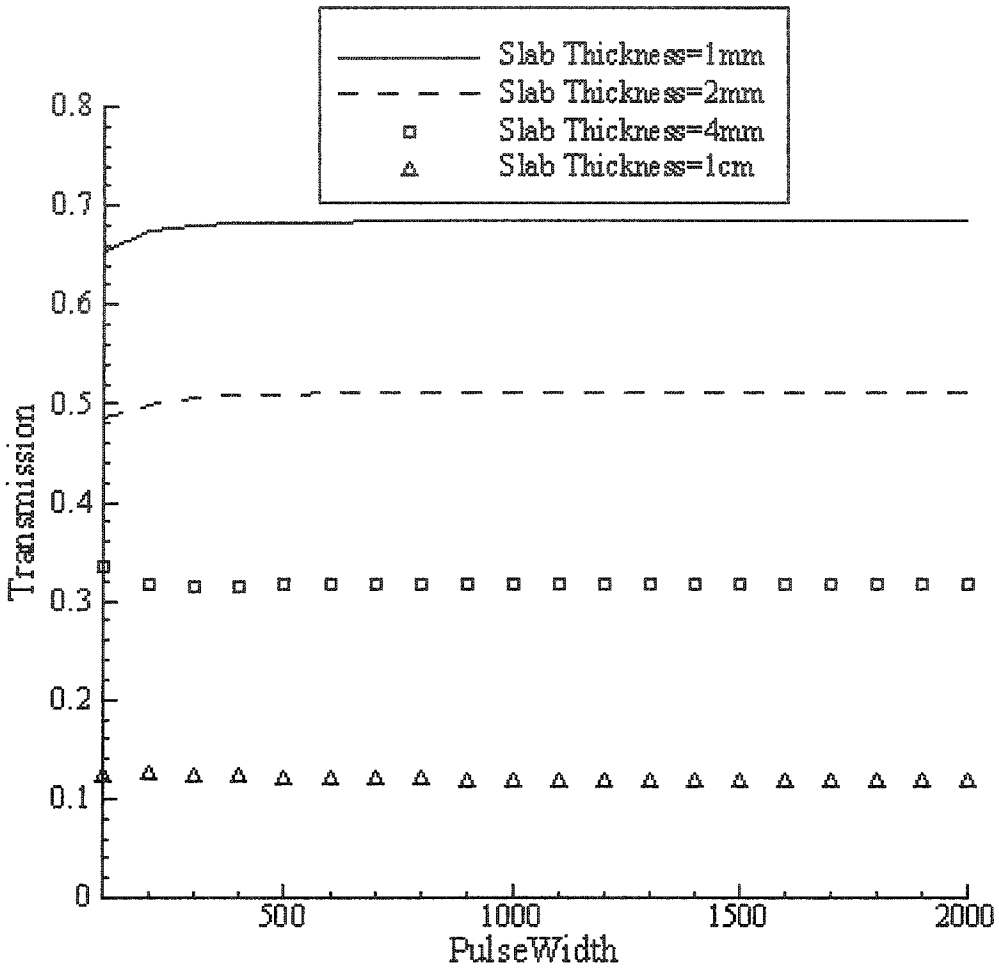
(b)

(Figure 5.16, Continued)



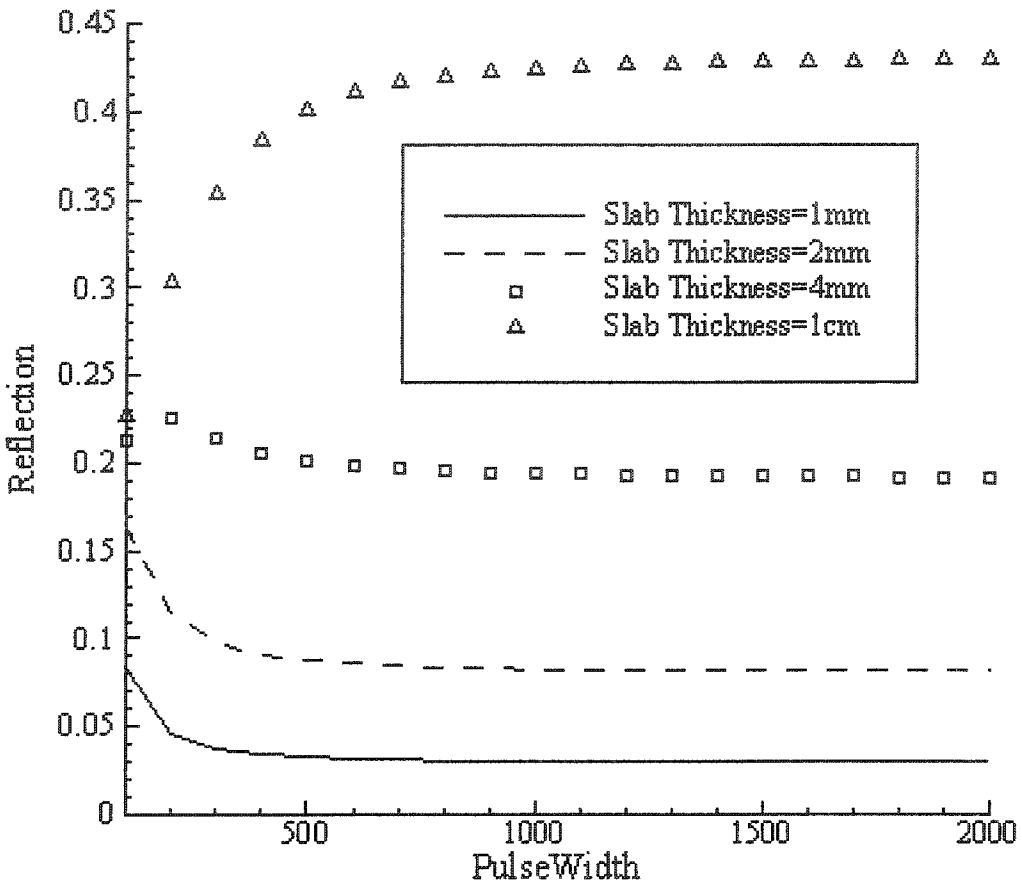
(c)

(Figure 5.16, Continued)



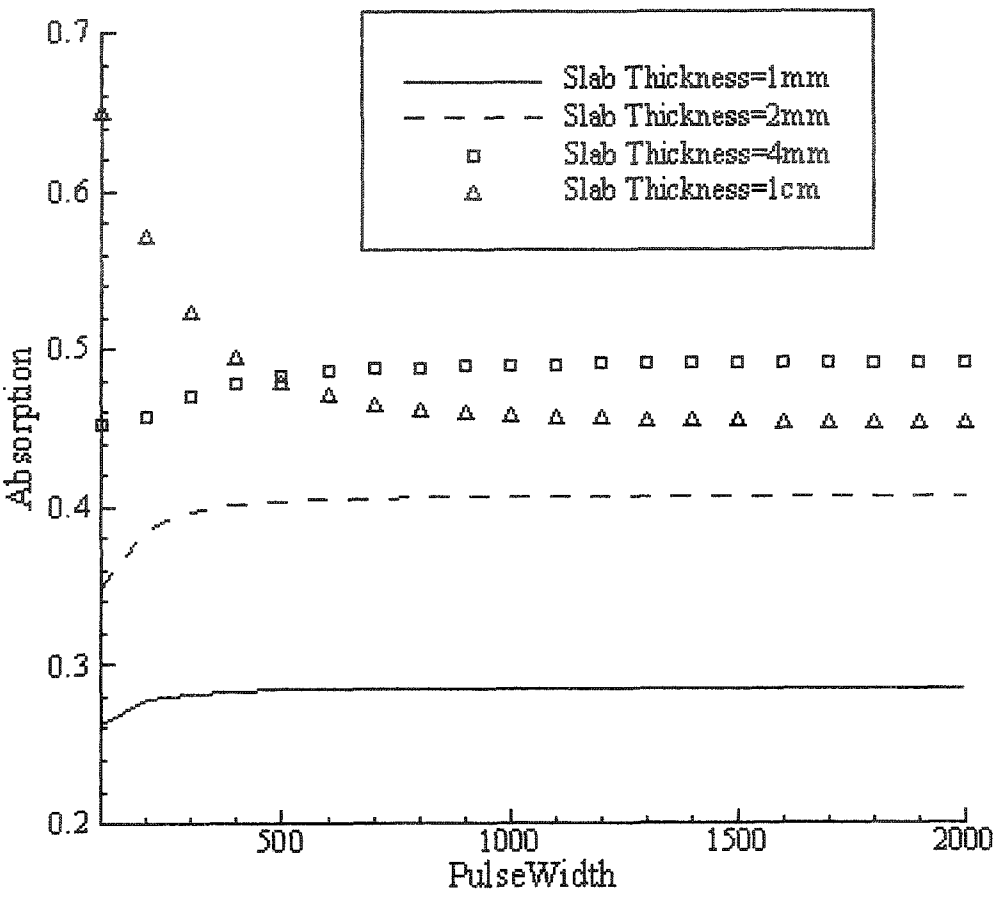
(a)

Figure 5.17.—(a), (b), (c) Transmission, reflection and absorption in case 2.



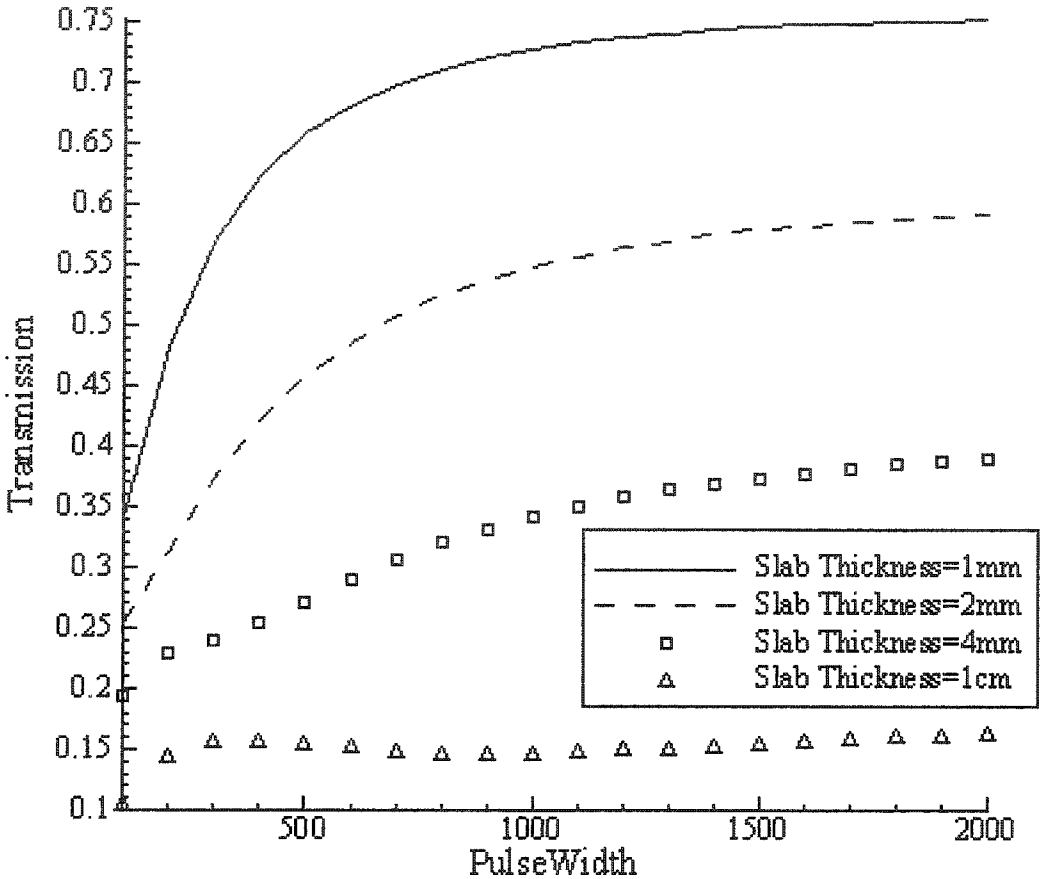
(b)

(Figure 5.17, Continued)



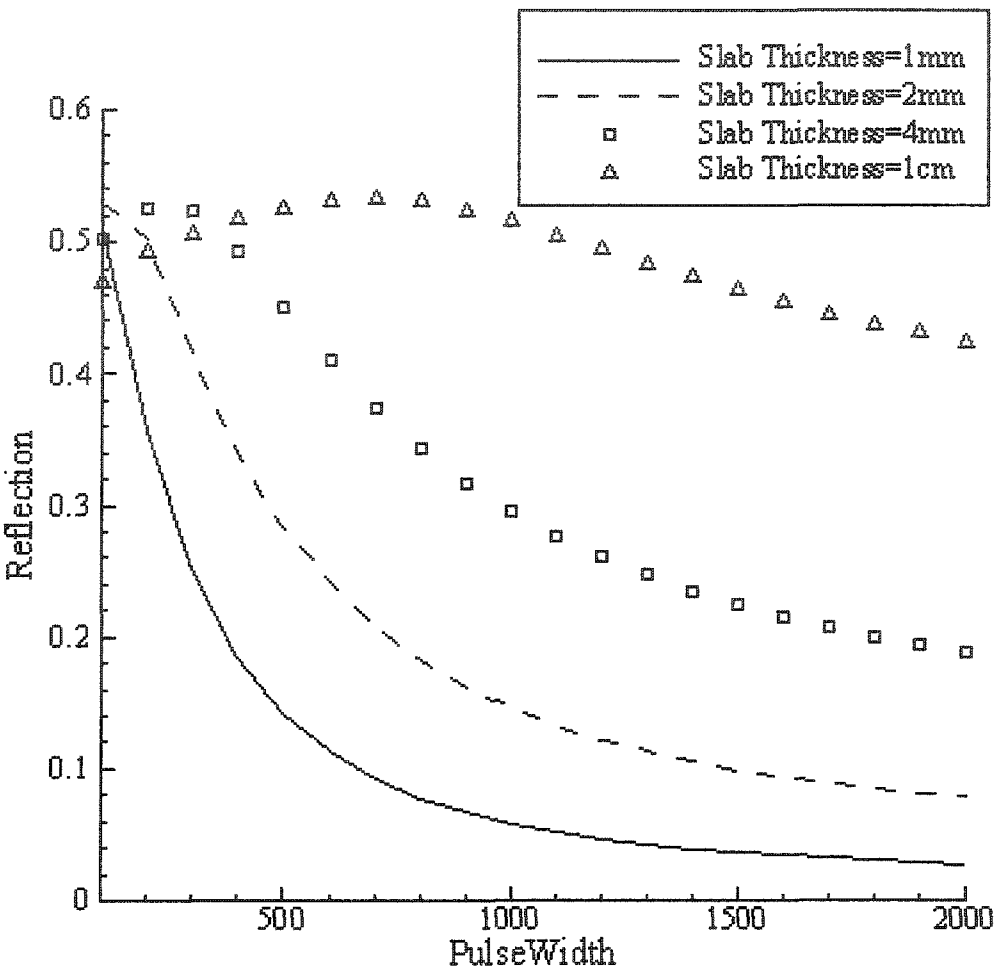
(c)

(Figure 5.17, Continued)



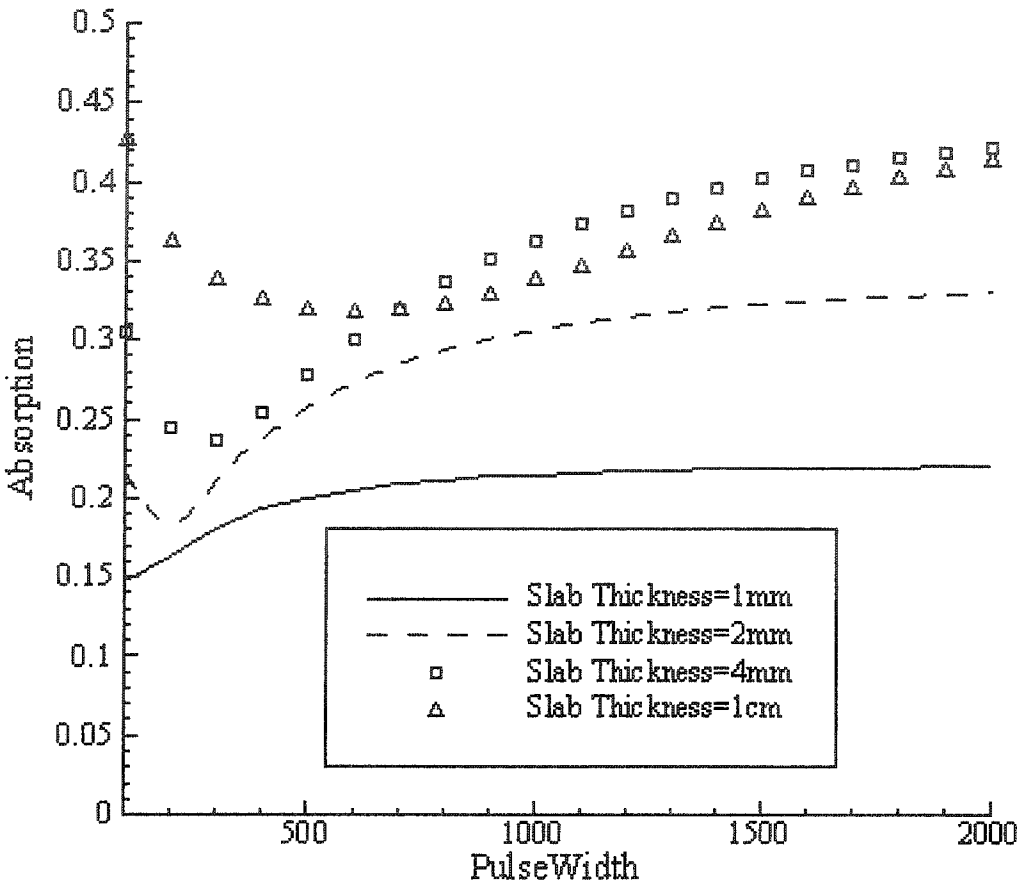
(a)

Figure 5.18.—(a), (b), (c) Transmission, reflection and absorption in case 3.



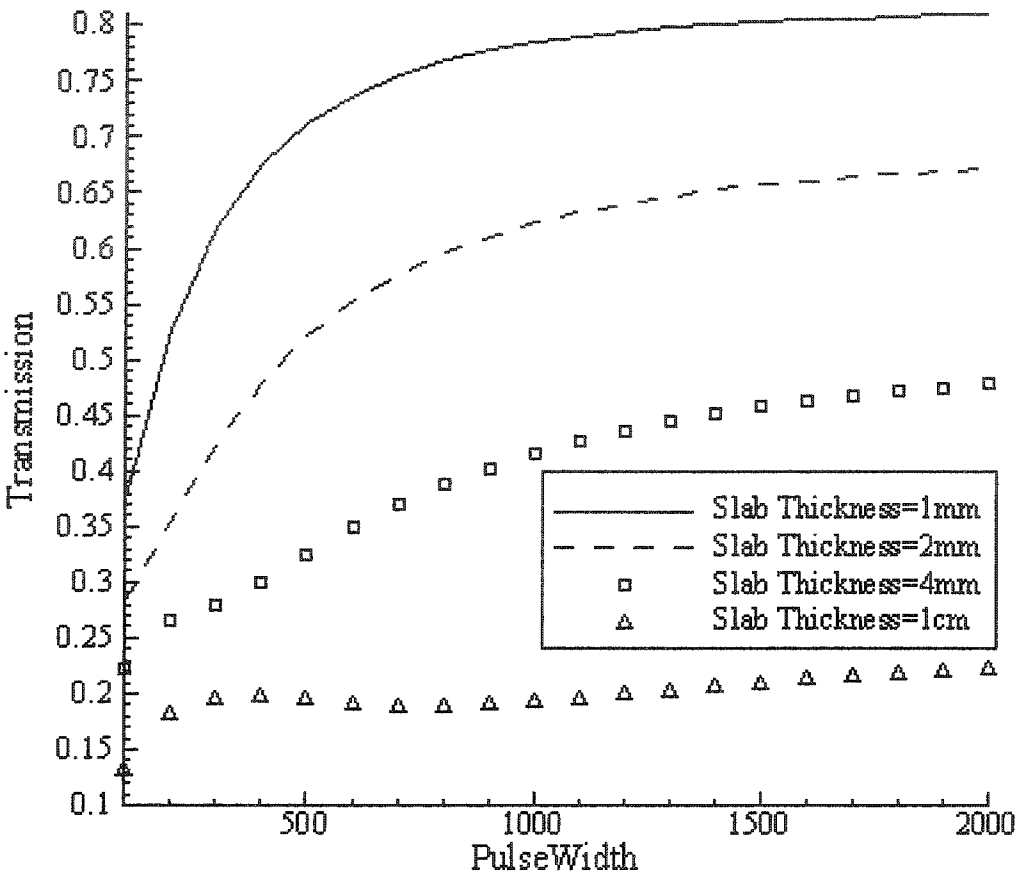
(b)

(Figure 5.18, Continued)



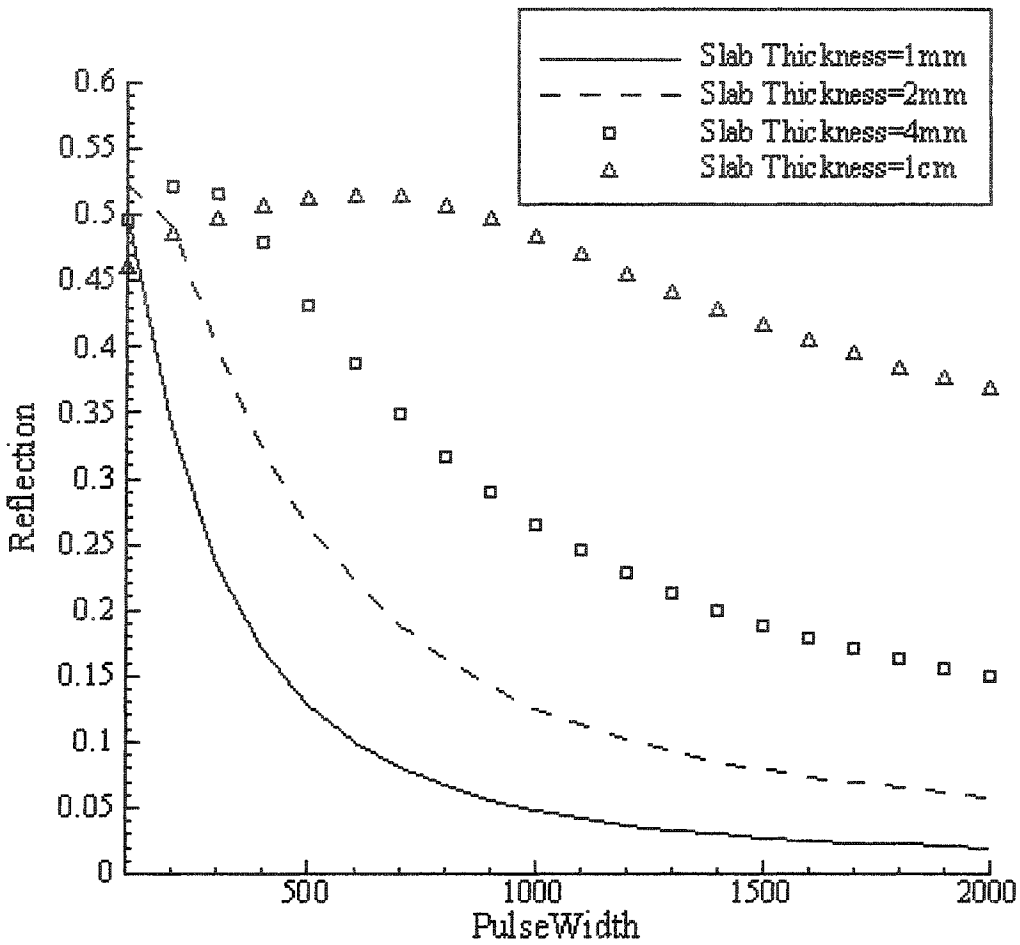
(c)

(Figure 5.18, Continued)



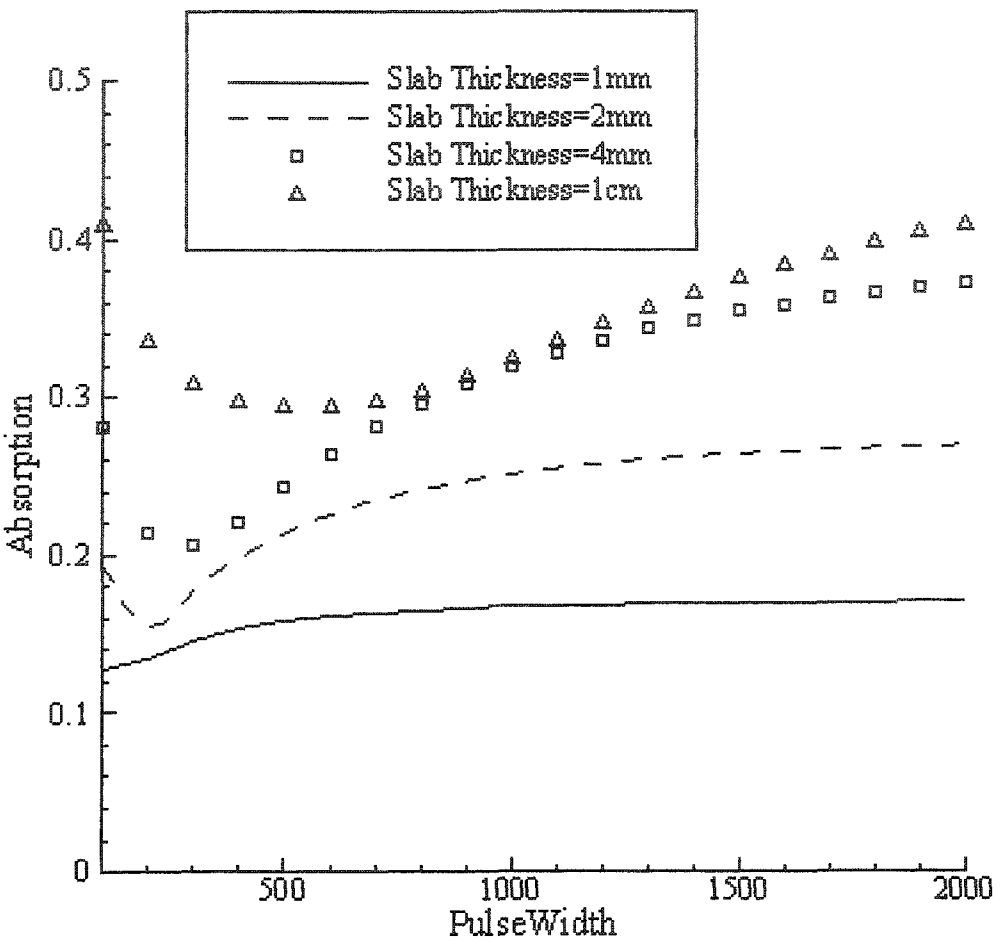
(a)

Figure 5.19.—(a), (b), (c) Transmission, reflection and absorption in case 4.



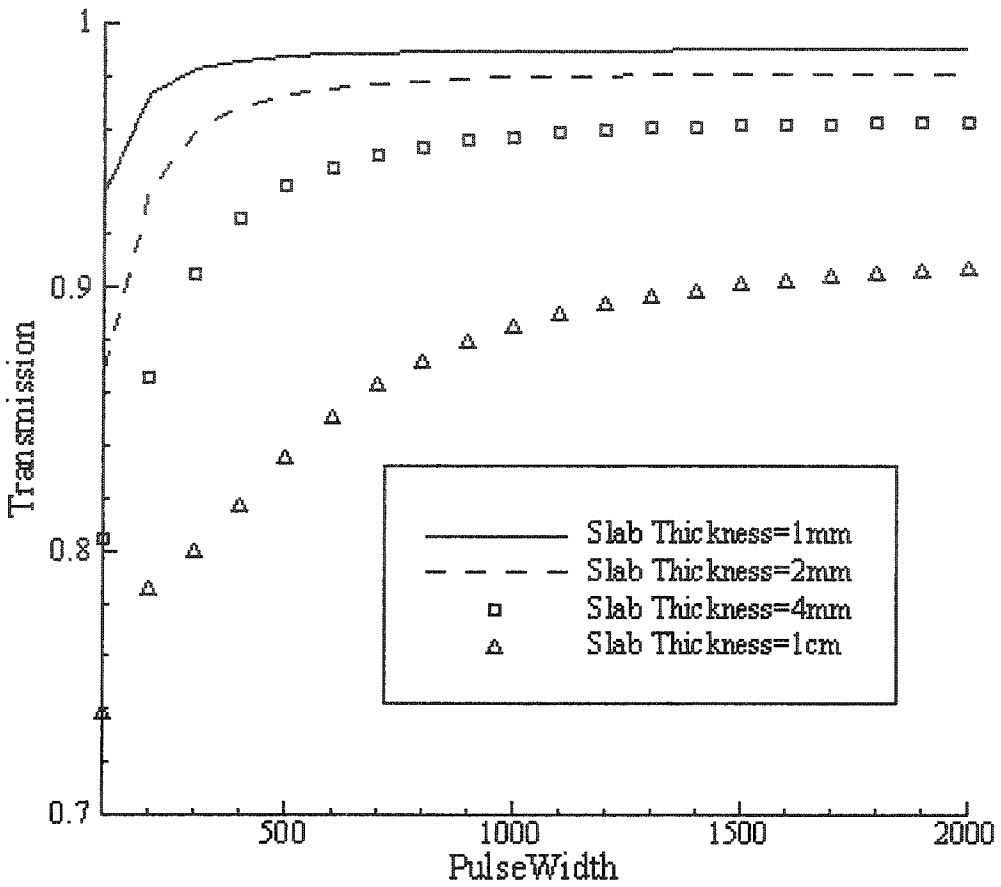
(b)

(Figure 5.19, Continued)



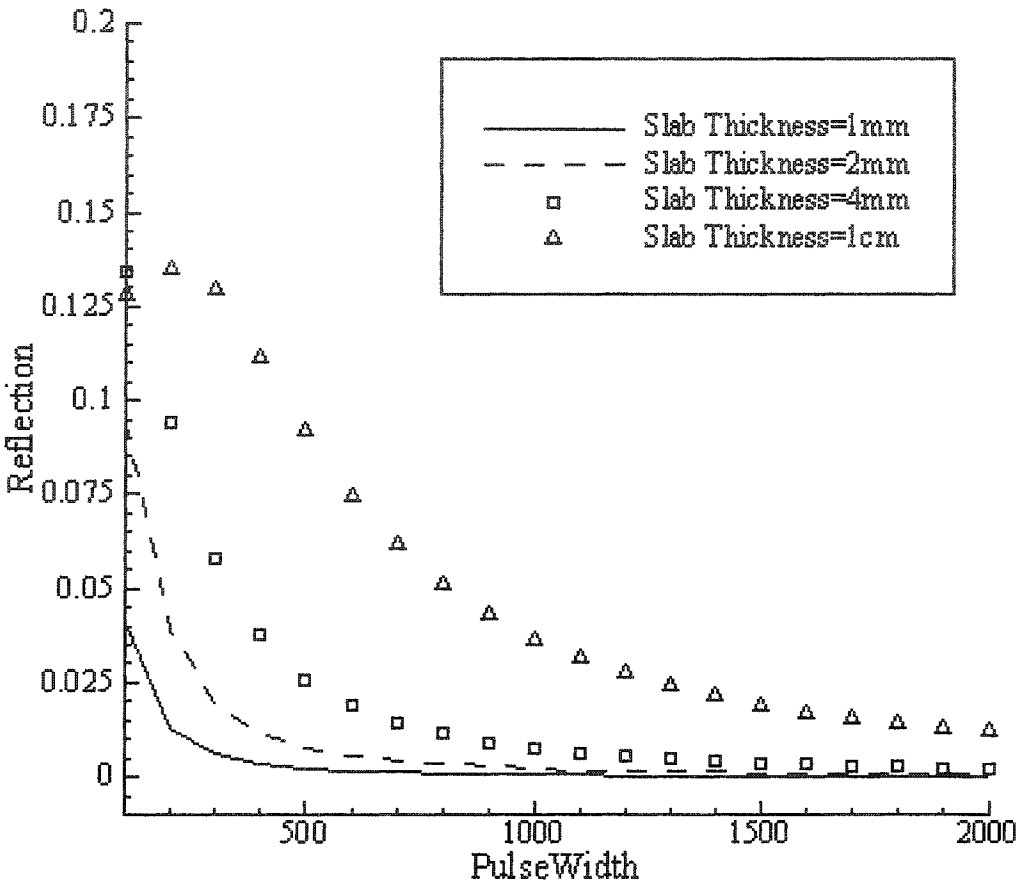
(c)

(Figure 5.19, Continued)



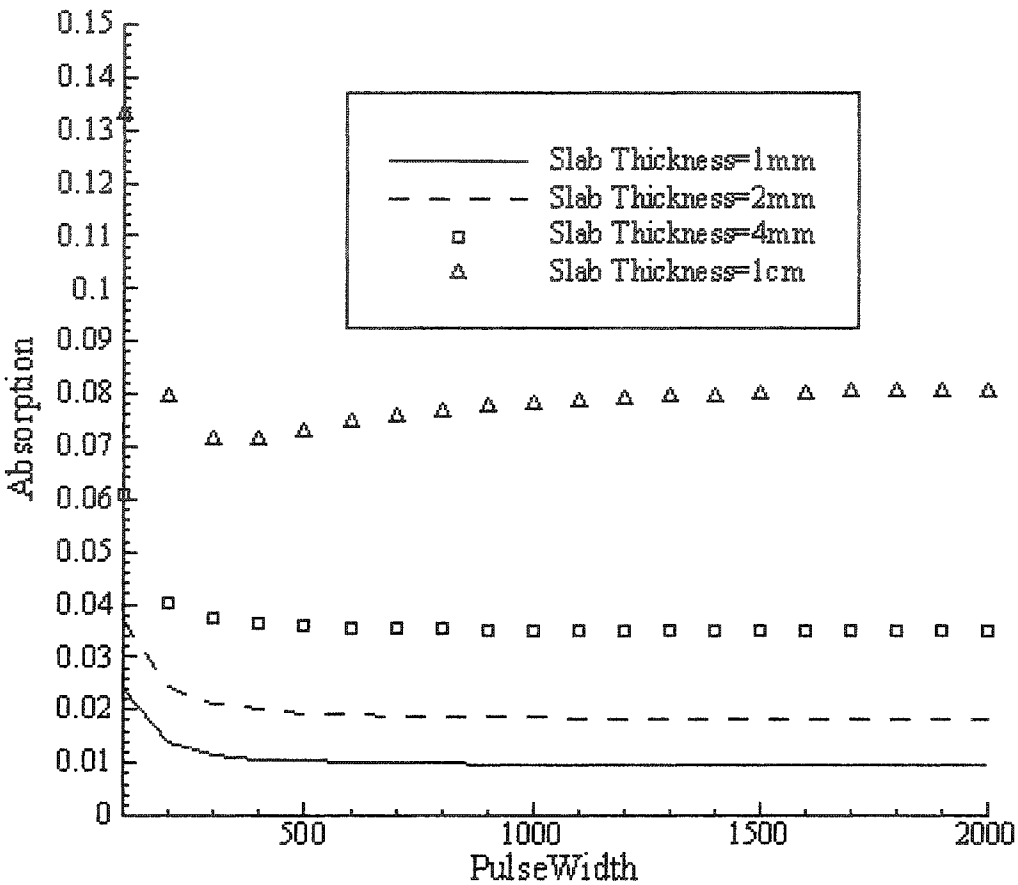
(a)

Figure 5.20.—(a), (b), (c) Transmission, reflection and absorption in case 5.



(b)

(Figure 5.20, Continued)



(c)

(Figure 5.20, Continued)

CHAPTER 6

CONCLUSION

In this study, we have developed a novel approach for simulating the propagation of a pulse electric field in biological matter subjected to a nanopulse. A new approximation of the Cole-Cole expression for the frequency dependence of the dielectric properties of tissue is used to improve the traditional Debye model. This method, based on a z-transformation and a second-order Taylor Approximation of the Cole-Cole expression, simplifies conversion from the frequency domain to the time domain. Maxwell's equations are then calculated using the finite difference time domain method coupled with the perfectly matched layer method in order to eliminate reflections from the boundary.

The new approach has been applied to investigate of the penetration of electromagnetic Gaussian nanopluses into biological matter. Transmission, reflection, and absorption of energy were calculated as a function of pulse width for different biological tissues. It is found that these properties depend substantially on pulse width as well as type of tissue. Absorption of energy from exposure to nanopulses can have a damaging effect on dells and tissues.

Future work in this direction should examine the role of pulse rise time in transmission, reflection, and absorption, as well the relevance of pulse shape to tissue

penetration. Moreover, the temperature distribution in biological tissues exposed to electromagnetic waves could be investigated [49]. Such study could help to elucidate non-thermal mechanisms of nanopulse bioeffects.

APPENDIX A

Source Codes for 1D Simulation

```
// PML.cpp : Defines the entry point for the console application.
//
/*Fd3d_4.3.c. 3D FDTD, plane wave on a dielectric sphere. */
/* this program simulates the plane wave in 3 dimensions impinging on a dielectric cylinder
the input to the program are, npml = 14, number spheres = 1, radius of cylinder = 20,
epsilon = 30, conductivity = .3, and nsteps = 30, 40, 75, 100*/

#include "stdafx.h"
# include <math.h>
# include <stdlib.h>
# include <stdio.h>

#define IE  151

#define Base1 60
#define Base2 600


double amp0,amp1,amp2;
int ww;
double  dx[IE];
double  ex[IE];
double  hy[IE];
double  ix[IE];
double  gax1[IE];
double  gax2[IE];
double  gax3[IE];
double  gax4[IE];
double  gbx1[IE];
double  gbx2[IE];
double  gbx3[IE];
double  gbx4[IE];

double  gdx1[IE];
double  gdx2[IE];
double  gdx3[IE];
double  gdx4[IE];
double  gcx[IE];
double  gx[IE];
double  sx1[IE][3];
double  sx2[IE][3];
double  sx3[IE][3];
double  sx4[IE][3];


double  inc_dx[IE];
double  inc_ex[IE];
double  inc_hy[IE];
double  inc_ix[IE];
double  inc_gax1[IE];
double  inc_gax2[IE];
double  inc_gax3[IE];
double  inc_gax4[IE];
double  inc_gbx1[IE];
```

```
double  inc_gbx2[IE];
double  inc_gbx3[IE];
double  inc_gbx4[IE];

double  inc_gdx1[IE];
double  inc_gdx2[IE];
double  inc_gdx3[IE];
double  inc_gdx4[IE];
double  inc_gcx[IE];
double  inc_gx[IE];
double  inc_sx1[IE][3];
double  inc_sx2[IE][3];
double  inc_sx3[IE][3];
double  inc_sx4[IE][3];


double  thick;
double  pw;
double  tran[60],ref[60],absp[60];
int cor[60];


int main(int argc, char* argv[])
{



   int cn=0;
   int flag1=0;
    int flag2=0;
    int flag3=0;

    int n,i,ic,nsteps;
   double  ddx,dt,epsz,muz,pi,T;
   int numsph;
    int    NCUR,NPR2,NPR1;

   double  t0,spread,pulse;
   double  ez_inc[IE],hx_inc[IE];


   double ex_low_m1,ex_low_m2,ex_high_m1,ex_high_m2;
   double inc_ex_low_m1,inc_ex_low_m2,inc_ex_high_m1,inc_ex_high_m2;
    double energy1;
    double energy2;
   double energy3;

    int count1;
    int count2;
    int count3;



  double  radius[10],epsilon[10],sigma[10],eps,cond;
   double  dell1[10],dell2[10],dell3[10],dell4[10];
```

```
    double  tau1[10],tau2[10],tau3[10],tau4[10];
    double   alpha1[10],alpha2[10],alpha3[10],alpha4[10];
    double   A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4;
   double  dist;


 //  double  real_pt[NFREQS][IE][JE],imag_pt[NFREQS][IE][JE];

   FILE *fp1,*fp2,*fp3,*fp4;
   ic = (IE-1)/2  ;

   pi = 3.1415926;
   epsz = 8.8e-12;
   muz  = 4*pi*1.e-8;
   ddx =1.e-4;             /* Cell size */
   dt = ddx/6e8;           /* Time steps */

//    fp1 = fopen("0.dat","w");
//     fprintf(fp1,"Thickness  Pulse Distance  Pulse Num   Crossing     Ref     Absorb    peneration  \n");
     fp1 = fopen("transmission.dat","w");
      fprintf(fp1,"TITLE=ELECTRIC FIELD\n");
      fprintf(fp1,"VARIABLES=PulseWidth, Transmission\n");
     fp2 = fopen("ref.dat","w");
      fprintf(fp2,"TITLE=ELECTRIC FIELD\n");
      fprintf(fp2,"VARIABLES=PulseWidth, Reflection\n");
     fp3 = fopen("Absportion.dat","w");
      fprintf(fp3,"TITLE=ELECTRIC FIELD\n");
      fprintf(fp3,"VARIABLES=PulseWidth, Absorption\n");
    fp4 = fopen("1dEz-t.dat","w");
      fprintf(fp4,"TITLE=ELECTRIC FIELD\n");
      fprintf(fp4,"VARIABLES=Y, Ez\n");
      /* Initialize the arrays */

  // Dis=ee*10000;
    for(int wall=4;wall<=4;wall++)
    {

            if(wall==1)
                    thick=5.0;
            else if(wall==2)
                    thick=10.0;
            else if(wall==3)
                    thick=20.0;
            else if(wall==4)
                    thick=50.0;
    for(int width=1;width<=20;width++)
    {
    ww=width*100;
    pulse=0.0;
   amp0=0.0;
   amp1=0.0;
    amp2=0.0;
    flag1=0;
    flag2=0;
    flag3=0;
    energy1=0.0;
```

```
 energy2=0.0;
energy3=0.0;
 count1=0;
 count2=0;
 count3=0;
 NCUR=2;
 NPR1=1;
 NPR2=0;
 for ( i=0; i < IE; i++ ) {
  ez_inc[i] = 0.;
  hx_inc[i] = 0.;


     ex[i]= 0.0 ;
     dx[i]= 0.0 ;
     ix[i]=0.0;
     hy[i]= 0.0 ;
     gcx[i]= 1.0;
              gx[i]=0.0;
              gax1[i]=0.0;
              gax2[i]=0.0;
              gax3[i]=0.0;
              gax4[i]=0.0;
              gbx1[i]=0.0;
              gbx2[i]=0.0;
              gbx3[i]=0.0;
              gbx4[i]=0.0;

              gdx1[i]=0.0;
              gdx2[i]=0.0;
              gdx3[i]=0.0;
              gdx4[i]=0.0;


}

 for ( i=0; i < IE; i++ ) {
 ez_inc[i] = 0.;
 hx_inc[i] = 0.;


    inc_ex[i]= 0.0 ;
    inc_dx[i]= 0.0 ;
    inc_ix[i]= 0.0;
    inc_hy[i]= 0.0 ;
    inc_gcx[i]= 1.0;
             inc_gx[i]=0.0;
             inc_gax1[i]=0.0;
             inc_gax2[i]=0.0;
             inc_gax3[i]=0.0;
             inc_gax4[i]=0.0;
             inc_gbx1[i]=0.0;
             inc_gbx2[i]=0.0;
             inc_gbx3[i]=0.0;
             inc_gbx4[i]=0.0;
```

```
            inc_gdx1[i]=0.0;
            inc_gdx2[i]=0.0;
            inc_gdx3[i]=0.0;
            inc_gdx4[i]=0.0;


    }


    ex_low_m2=0.;
    ex_low_m1=0.;
    ex_high_m2=0.;
   ex_high_m1=0.;

    inc_ex_low_m2=0.;
    inc_ex_low_m1=0.;
    inc_ex_high_m2=0.;
  inc_ex_high_m1=0.;


////////////////


    for ( i=0; i < IE; i++ ) {

                    for(int temp=0;temp<3;temp++){

        sx1[i][temp]=0.0;
                sx2[i][temp]=0.0;
                        sx3[i][temp]=0.0;
                        sx4[i][temp]=0.0;

    }}

    ///
  for ( i=0; i < IE; i++ ) {

                    for(int temp=0;temp<3;temp++){

        inc_sx1[i][temp]=0.0;
                inc_sx2[i][temp]=0.0;
                        inc_sx3[i][temp]=0.0;
                        inc_sx4[i][temp]=0.0;

    }}


  /* Specify the dielectric sphere */

    epsilon[0] = 1.;
    sigma[0]   = 0.;
     dell1[0]=0.;
     dell2[0]=0.;
     dell3[0]=0.;
     dell4[0]=0.;
```

```
 tau1[0]=0.;
 tau2[0]=0.;
 tau3[0]=0.;
 tau4[0]=0.;
alpha1[0]=0.;
 alpha2[0]=0.;
 alpha3[0]=0.;
 alpha4[0]=0.;

  numsph=1;

   radius[1]=thick;

        ////////////// blood

        epsilon[1]=4.;
        sigma[1]=0.7;
        dell1[1]=56;
        dell2[1]=5200;
        dell3[1]=0.;
        dell4[1]=0.;
        tau1[1]=8.377e-12;
        tau2[1]=132.629e-9;
        tau3[1]=159.155e-6;
        tau4[1]=15.915e-3;
        alpha1[1]=0.1;
        alpha2[1]=0.1;
        alpha3[1]=0.2;
        alpha4[1]=0.;


/*      Calculate gax,gbx  */
  for ( i = 0; i < IE; i++ ) {

  eps = epsilon[0];
  cond = sigma[0];
   C1=pow(tau1[0]/dt,1-alpha1[0]);
   C2=pow(tau2[0]/dt,1-alpha2[0]);
   C3=pow(tau3[0]/dt,1-alpha3[0]);
   C4=pow(tau4[0]/dt,1-alpha4[0]);
   B1=alpha1[0];
   B2=alpha2[0];
   B3=alpha3[0];
   B4=alpha4[0];
   A1=dell1[0]/(1+C1);
   A2=dell2[0]/(1+C2);
   A3=dell3[0]/(1+C3);
   A4=dell4[0]/(1+C4);


       dist = (ic-i);

   //     dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                              dist=sqrt(pow(dist,2.));
       for (n=1; n<= numsph; n++) {
                /*
```

```
                        if(n==3)
                        {
                                ydist=ydist-3;
                        xdist=xdist-3;
                                dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                        }
                        */
        if( dist <= radius[n]) {
        eps = epsilon[n];
        cond = sigma[n] ;
                        C1=pow(tau1[n]/dt,1-alpha1[n]);
                        C2=pow(tau2[n]/dt,1-alpha2[n]);
                        C3=pow(tau3[n]/dt,1-alpha3[n]);
                        C4=pow(tau4[n]/dt,1-alpha4[n]);
                        B1=alpha1[n];
                        B2=alpha2[n];
                        B3=alpha3[n];
                        B4=alpha4[n];
                        A1=dell1[n]/(1+C1);
                        A2=dell2[n]/(1+C2);
                        A3=dell3[n]/(1+C3);
                        A4=dell4[n]/(1+C4);
        }

      }
                      gcx[i]=1./(eps+(cond*dt/epsz)+A1+A2+A3+A4);
      gax1[i]=(1-B1)*C1/(1+C1);
                      gax2[i]=(1-B2)*C2/(1+C2);
                      gax3[i]=(1-B3)*C3/(1+C3);
                      gax4[i]=(1-B4)*C4/(1+C4);
                      gbx1[i]=0.5*(1-B1)*B1*C1/(1+C1);
                      gbx2[i]=0.5*(1-B2)*B2*C2/(1+C2);
                      gbx3[i]=0.5*(1-B3)*B3*C3/(1+C3);
                      gbx4[i]=0.5*(1-B4)*B4*C4/(1+C4);
                      gdx1[i]=A1;
                      gdx2[i]=A2;
      gdx3[i]=A3;
      gdx4[i]=A4;
      gx[i]=cond*dt/epsz;



  }



t0 = 40.0;
spread = 10.0;
T = 0;
```

```
  //while ( nsteps > 0 ) {
  //   printf( "nsteps --> ");
  //   scanf("%d", &nsteps);
  //   printf("%d \n", nsteps);
   nsteps= 60*ww;

   for ( n=1; n <=nsteps ; n++)  {
     T = T + 1;
      NPR2=NPR1;
             NPR1=NCUR;
             NCUR=(NCUR+1)%3;
/*  ----  Start of the Main FDTD loop ----  */



             pw=(double)ww;
       pulse=exp(-(T-5*pw)*(T-5*pw)/(pw*pw));



    /* Calculate the Dx field */


    for ( i=0; i < IE; i++ ) {

         dx[i] =dx[i]+0.5*( hy[i-1]- hy[i]) ;

    }

     dx[5]=dx[5]+pulse;



    for ( i=1; i < IE-1; i++ ) {

       B1=gax1[i]*sx1[i][NPR1]+gbx1[i]*sx1[i][NPR2];
                  B2=gax2[i]*sx2[i][NPR1]+gbx2[i]*sx2[i][NPR2];
       B3=gax3[i]*sx3[i][NPR1]+gbx3[i]*sx3[i][NPR2];
            B4=gax4[i]*sx4[i][NPR1]+gbx4[i]*sx4[i][NPR2];
       ex[i] = gcx[i]*(dx[i] - ix[i]-B1-B2-B3-B4);
       ix[i]= ix[i] + gx[i]*ex[i];
       sx1[i][NCUR]=B1+gdx1[i]*ex[i];
                  sx2[i][NCUR]=B2+gdx2[i]*ex[i];
                  sx3[i][NCUR]=B3+gdx3[i]*ex[i];
                  sx4[i][NCUR]=B4+gdx4[i]*ex[i];


                 }

    ex[0]=ex_low_m2;
          ex_low_m2=ex_low_m1;
          ex_low_m1=ex[1];

          ex[IE-1]=ex_high_m2;
          ex_high_m2=ex_high_m1;
          ex_high_m1=ex[IE-2];
```

```
for(  i=0;i<IE-1;i++)
 {
          hy[i]=hy[i]+ 0.5*(ex[i]-ex[i+1]);
 }


 ///////

 /* Calculate the Dx field */


 for ( i=0; i < IE; i++ ) {

       inc_dx[i] =inc_dx[i]+0.5*( inc_hy[i-1]- inc_hy[i]) ;

 }

  inc_dx[5]=inc_dx[5]+pulse;



 for ( i=1; i < IE-1; i++ ) {

     B1=inc_gax1[i]*inc_sx1[i][NPR1]+inc_gbx1[i]*inc_sx1[i][NPR2];
                  B2=inc_gax2[i]*inc_sx2[i][NPR1]+inc_gbx2[i]*inc_sx2[i][NPR2];
     B3=inc_gax3[i]*inc_sx3[i][NPR1]+inc_gbx3[i]*inc_sx3[i][NPR2];
           B4=inc_gax4[i]*inc_sx4[i][NPR1]+inc_gbx4[i]*inc_sx4[i][NPR2];
     inc_ex[i] = inc_gcx[i]*(inc_dx[i] - inc_ix[i]-B1-B2-B3-B4);
     inc_ix[i]= inc_ix[i] + inc_gx[i]*inc_ex[i];
     inc_sx1[i][NCUR]=B1+inc_gdx1[i]*inc_ex[i];
                  inc_sx2[i][NCUR]=B2+inc_gdx2[i]*inc_ex[i];
                  inc_sx3[i][NCUR]=B3+inc_gdx3[i]*inc_ex[i];
                  inc_sx4[i][NCUR]=B4+inc_gdx4[i]*inc_ex[i];


                 }

 inc_ex[0]=inc_ex_low_m2;
         inc_ex_low_m2=inc_ex_low_m1;
         inc_ex_low_m1=inc_ex[1];

         inc_ex[IE-1]=inc_ex_high_m2;
         inc_ex_high_m2=inc_ex_high_m1;
         inc_ex_high_m1=inc_ex[IE-2];


for(  i=0;i<IE-1;i++)
 {
          inc_hy[i]=inc_hy[i]+ 0.5*(inc_ex[i]-inc_ex[i+1]);
 }
```

```
if(n%100==0)
        {

                printf("Thickness=%f, width=%f, n= %d\n",thick,pw,n);


    fprintf(fp4,"ZONE \n");
        for( int s=0;s<IE;s++)
                fprintf(fp4,"%d   %20.12lf\n",s,ex[s]);

        }



        amp0=inc_ex[8];
        amp1=ex[8]-inc_ex[8];
        amp2=ex[IE-8];


// calculate energy

                if(count1%2==0)
                {
                        energy1=energy1+4*amp0*amp0;
                }
                else
                {
                        energy1=energy1+2*amp0*amp0;
                }
                count1++;
//          }

          if(flag2==0)
          {
    if(amp2!=0)
                flag2=1;

          }
//

          if(flag2==1)
          {
                if(count2%2==0)
                {
                        energy2=energy2+4*amp2*amp2;
                }
                else
                {
                        energy2=energy2+2*amp2*amp2;
                }
                count2++;
          }


           if(flag3==0)
          {
```

```
        if(amp1<0.0)
                flag3=1;

        }


        if(flag3==1)
        {
                if(count3%2==0)
                {
                        energy3=energy3+4*amp1*amp1;
                }
                else
                {
                        energy3=energy3+2*amp1*amp1;
                }
                count3++;
        }



    }//for
     tran[width]=energy2/energy1;
     ref[width]=energy3/energy1;
     absp[width]=(energy1-energy2-energy3)/energy1;
     cor[width]=ww;

     }//for
         fprintf(fp1,"ZONE \n");
             for(  cn=1;cn<=20;cn++)
                     fprintf(fp1,"%d   %20.12lf\n",cor[cn],tran[cn]);
                     fprintf(fp2,"ZONE \n");
             for(  cn=1;cn<=20;cn++)
                     fprintf(fp2,"%d   %20.12lf\n",cor[cn],ref[cn]);
                     fprintf(fp3,"ZONE \n");
             for( cn=1;cn<=20;cn++)
                     fprintf(fp3,"%d   %20.12lf\n",cor[cn],absp[cn]);
     }//for



     return 0;
}
```

APPENDIX B

Source Codes for 2D Simulation

```
// PML.cpp : Defines the entry point for the console application.
//
/*Fd3d_4.3.c. 3D FDTD, plane wave on a dielectric sphere. */
/* this program simulates the plane wave in 3 dimensions impinging on a dielectric cylinder
the input to the program are, npml = 14, number spheres = 1, radius of cylinder = 20,
epsilon = 30, conductivity = .3, and nsteps = 30, 40, 75, 100*/

#include "stdafx.h"
# include <math.h>
# include <stdlib.h>
# include <stdio.h>

#define IE  61
#define JE  61
#define KE  61
#define ia  5
#define ja  5
#define ka  5
#define NFREQS 10
#define Emp .33
double amp0,amp1,amp2,ww;
double energy1,energy2,energy3;
int flag1,flag2,flag3;
int count1,count2,count3;

double  dz[IE][JE],ez[IE][JE];
double  hx[IE][JE],hy[IE][JE];
double  iz[IE][JE];
double  gaz1[IE][JE],gaz2[IE][JE],gaz3[IE][JE],gaz4[IE][JE];
double  gbz1[IE][JE],gbz2[IE][JE],gbz3[IE][JE],gbz4[IE][JE];
double  gdz1[IE][JE],gdz2[IE][JE],gdz3[IE][JE],gdz4[IE][JE];
double  gcz[IE][JE];
double  gz[IE][JE];
double  sz1[IE][JE][3],sz2[IE][JE][3],sz3[IE][JE][3],sz4[IE][JE][3];
double  ihx[IE][JE],ihy[IE][JE];
double  ez_inc[JE],hx_inc[JE],dz_inc[JE],iz_inc[JE];
double  ez_low_m1,ez_low_m2,ez_high_m1,ez_high_m2;

double  inc_gaz1[JE];
double  inc_gaz2[JE];
double  inc_gaz3[JE];
double  inc_gaz4[JE];
double  inc_gbz1[JE];
double  inc_gbz2[JE];
double  inc_gbz3[JE];
double  inc_gbz4[JE];

double  inc_gdz1[JE];
double  inc_gdz2[JE];
double  inc_gdz3[JE];
double  inc_gdz4[JE];
double  inc_gcz[JE];
double  inc_gz[JE];
double  inc_sz1[JE][3];
```

```
double inc_sz2[JE][3];
double inc_sz3[JE][3];
double inc_sz4[JE][3];
//////////////////

double org_dz[IE][JE], org_ez[IE][JE];
double org_hx[IE][JE], org_hy[IE][JE];
double org_iz[IE][JE];
double org_gaz1[IE][JE],org_gaz2[IE][JE],org_gaz3[IE][JE],org_gaz4[IE][JE];
double org_gbz1[IE][JE],org_gbz2[IE][JE],org_gbz3[IE][JE],org_gbz4[IE][JE];
double org_gdz1[IE][JE],org_gdz2[IE][JE],org_gdz3[IE][JE],org_gdz4[IE][JE];
double org_gcz[IE][JE];
double org_gz[IE][JE];
double org_sz1[IE][JE][3],org_sz2[IE][JE][3],org_sz3[IE][JE][3],org_sz4[IE][JE][3];
double org_ihx[IE][JE],org_ihy[IE][JE];
double org_ez_inc[JE],org_hx_inc[JE],org_dz_inc[JE],org_iz_inc[JE];
double org_ez_low_m1,org_ez_low_m2,org_ez_high_m1,org_ez_high_m2;
double org_inc_gaz1[JE];
double org_inc_gaz2[JE];
double org_inc_gaz3[JE];
double org_inc_gaz4[JE];
double org_inc_gbz1[JE];
double org_inc_gbz2[JE];
double org_inc_gbz3[JE];
double org_inc_gbz4[JE];

double org_inc_gdz1[JE];
double org_inc_gdz2[JE];
double org_inc_gdz3[JE];
double org_inc_gdz4[JE];
double org_inc_gcz[JE];
double org_inc_gz[JE];
double org_inc_sz1[JE][3];
double org_inc_sz2[JE][3];
double org_inc_sz3[JE][3];
double org_inc_sz4[JE][3];

//////////////////////
int main(int argc, char* argv[])
{
  int n,i,j,ic,jc,kc,nsteps,n_pml;
  double ddx,dt,epsz,muz,pi,npml,T;
  int ib,jb,kb,numsph;
       int   NCUR,NPR2,NPR1;
  double xn,xxn,curl_e;
  double t0,spread,pulse,thick;

   double gi2[IE],gi3[IE];
   double gj2[JE],gj3[JE];
   double fi1[IE],fi2[IE],fi3[IE];
   double fj1[JE],fj2[JE],fj3[JE];


  double radius[10],epsilon[10],sigma[10],eps,cond;
       double dell1[10],dell2[10],dell3[10],dell4[10];
       double tau1[10],tau2[10],tau3[10],tau4[10];
```

```
        double  alpha1[10],alpha2[10],alpha3[10],alpha4[10];
        double  A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4;
double  dist,xdist,ydist;
        double alfa,beta,rise,x,peak,ratio;


FILE *fp,*fp1,*fp2,*fp3,*fp5,*fp6,*fp7,*fp8,*fp9,*fp10;
ic = (IE-1)/2  ;
jc = (JE-1)/2 ;
kc = (KE-1)/2  ;
ib = IE - ia - 1;
jb = JE - ja - 1;
kb = KE - ka - 1;
pi = 3.14159;
epsz = 8.8e-12;
muz  = 4*pi*1.e-7;
ddx = 1e-6;             /* Cell size */
dt = ddx/6e8;           /* Time steps */
        alfa=0.1;
        beta=0.08;
        rise=(log(alfa)-log(beta))/(alfa-beta);
        x=(log(alfa)-log(beta))/(alfa-beta);
        peak=exp(-alfa*x)-exp(-beta*x);
        ratio=1.0/peak;

        fp = fopen("Ez-t.dat","w");
          fprintf(fp,"TITLE=ELECTRIC FIELD\n");
          fprintf(fp,"VARIABLES=X, Y, Ez\n");
                fp5 = fopen("130.dat","w");
        fprintf(fp5,"TITLE=ELECTRIC FIELD\n");
        fprintf(fp5,"VARIABLES= Y, Ez\n");
                fp6 = fopen("140.dat","w");
        fprintf(fp6,"TITLE=ELECTRIC FIELD\n");
        fprintf(fp6,"VARIABLES= Y, Ez\n");
                fp7 = fopen("150.dat","w");
        fprintf(fp7,"TITLE=ELECTRIC FIELD\n");
        fprintf(fp7,"VARIABLES= Y, Ez\n");
                fp8 = fopen("180.dat","w");
        fprintf(fp8,"TITLE=ELECTRIC FIELD\n");
        fprintf(fp8,"VARIABLES= Y, Ez\n");
                fp9 = fopen("200.dat","w");
        fprintf(fp9,"TITLE=ELECTRIC FIELD\n");
        fprintf(fp9,"VARIABLES= Y, Ez\n");
                fp10 = fopen("210.dat","w");
        fprintf(fp10,"TITLE=ELECTRIC FIELD\n");
        fprintf(fp10,"VARIABLES= Y, Ez\n");

          fp1 = fopen("data.dat","w");
          fprintf(fp1,"Rising Time  Thickness  Crossing     Ref     Absorb    peneration  \n");

          fp2 = fopen("1dEz-t.dat","w");
          fprintf(fp2,"TITLE=ELECTRIC FIELD\n");
          fprintf(fp2,"VARIABLES=Y, Ez\n");


          fp3 = fopen("Ez-t1.dat","w");
```

```
        fprintf(fp3,"TITLE=ELECTRIC FIELD\n");
        fprintf(fp3,"VARIABLES=Y, Ez\n");


/* Initialize the arrays */
                for(int wall=1;wall<=1;wall++)
        {
                thick=(double)wall*2.0;


                for(int width=8;width<=8;width++)
                {


                        //////////
      amp0=0.0;
                        amp1=0.0;
                        amp2=0.0;
                        flag1=0;
                        flag2=0;
                        flag3=0;
                        energy1=0.0;
                        energy2=0.0;
                  energy3=0.0;
                        count1=0;
                        count2=0;
                        count3=0;
                        NCUR=2;
                        NPR1=1;
                        NPR2=0;

                        ////////
                 for ( j=0; j < JE; j++ ) {

    for ( i=0; i < IE; i++ ) {
     ez[i][j]= 0.0 ;
     dz[i][j]= 0.0 ;
     hx[i][j]= 0.0 ;
     hy[i][j]= 0.0 ;
     iz[i][j]= 0.0 ;
                  gcz[i][j]= 1.0;
                  gz[i][j]=0.0;
                  gaz1[i][j]=0.0;
                  gaz2[i][j]=0.0;
                  gaz3[i][j]=0.0;
                  gaz4[i][j]=0.0;

                  gbz1[i][j]=0.0;
                  gbz2[i][j]=0.0;
                  gbz3[i][j]=0.0;
                  gbz4[i][j]=0.0;

                  gdz1[i][j]=0.0;
                  gdz2[i][j]=0.0;
                  gdz3[i][j]=0.0;
                  gdz4[i][j]=0.0;
```

```
        ihx[i][j] = 0.0;
        ihy[i][j] = 0.0;

                    } }

////////////////


          for ( j=0; j < JE; j++ ) {
        for ( i=0; i < IE; i++ ) {
                                  for(int temp=0;temp<3;temp++){


                                        sz1[i][j][temp]=0.0;
                                sz2[i][j][temp]=0.0;
                                        sz3[i][j][temp]=0.0;
                                        sz4[i][j][temp]=0.0;
          } } }

          ///


for(j=0;j<JE;j++)
{
          ez_inc[j]=0;
          dz_inc[j]=0;
          hx_inc[j]=0;
          iz_inc[j]=0;
          inc_gcz[j]= 1.0;
          inc_gz[j]=0.0;
          inc_gaz1[j]=0.0;
          inc_gaz2[j]=0.0;
          inc_gaz3[j]=0.0;
          inc_gaz4[j]=0.0;
          inc_gbz1[j]=0.0;
          inc_gbz2[j]=0.0;
          inc_gbz3[j]=0.0;
  inc_gbz4[j]=0.0;
          inc_gdz1[j]=0.0;
          inc_gdz2[j]=0.0;
          inc_gdz3[j]=0.0;
          inc_gdz4[j]=0.0;
          }

          ez_low_m1=0;
          ez_low_m2=0;
          ez_high_m1=0;
          ez_high_m2=0;



         /////
  for ( j=0; j < JE; j++ ) {

    for ( i=0; i < IE; i++ ) {
      org_ez[i][j]= 0.0 ;
```

```
org_dz[i][j]= 0.0 ;
org_hx[i][j]= 0.0 ;
org_hy[i][j]= 0.0 ;
org_iz[i][j]= 0.0 ;
                org_gcz[i][j]= 1.0;
                org_gz[i][j]=0.0;
                org_gaz1[i][j]=0.0;
                org_gaz2[i][j]=0.0;
                org_gaz3[i][j]=0.0;
                org_gaz4[i][j]=0.0;

                org_gbz1[i][j]=0.0;
                org_gbz2[i][j]=0.0;
                org_gbz3[i][j]=0.0;
                org_gbz4[i][j]=0.0;

                org_gdz1[i][j]=0.0;
                org_gdz2[i][j]=0.0;
                org_gdz3[i][j]=0.0;
                org_gdz4[i][j]=0.0;

                org_ihx[i][j] = 0.0;
      org_ihy[i][j] = 0.0;

              } }

//////////////

    for ( j=0; j < JE; j++ ) {
   for ( i=0; i < IE; i++ ) {
                        for(int temp=0;temp<3;temp++){


                              org_sz1[i][j][temp]=0.0;
                      org_sz2[i][j][temp]=0.0;
                              org_sz3[i][j][temp]=0.0;
                              org_sz4[i][j][temp]=0.0;
    } } }

    ///


for(j=0;j<JE;j++)
{
        org_ez_inc[j]=0;
        org_dz_inc[j]=0;
        org_hx_inc[j]=0;
        org_iz_inc[j]=0;
        org_inc_gcz[j]= 1.0;
        org_inc_gz[j]=0.0;
        org_inc_gaz1[j]=0.0;
        org_inc_gaz2[j]=0.0;
        org_inc_gaz3[j]=0.0;
        org_inc_gaz4[j]=0.0;
        org_inc_gbz1[j]=0.0;
```

```
        org_inc_gbz2[j]=0.0;
        org_inc_gbz3[j]=0.0;
org_inc_gbz4[j]=0.0;
        org_inc_gdz1[j]=0.0;
        org_inc_gdz2[j]=0.0;
        org_inc_gdz3[j]=0.0;
        org_inc_gdz4[j]=0.0;
        }

        org_ez_low_m1=0;
        org_ez_low_m2=0;
        org_ez_high_m1=0;
        org_ez_high_m2=0;


                ////////

/* Parameters for the Fourier Transforms */




/*  Boundary Conditions */

  for ( i=0; i < IE; i++ ) {

  fi1[i] = 0.;
  gi2[i] = 1.;
  fi2[i] = 1.;
  gi3[i] = 1.;
  fi3[i] = 1.;
  }

  for ( j=0; j < JE; j++ ) {

  fj1[j] = 0.;
  gj2[j] = 1.;
  fj2[j] = 1.;
  gj3[j] = 1.;
  fj3[j] = 1.;
  }

   n_pml =5;
   npml=5.0;
  for ( i=0; i < n_pml; i++ ) {
    xxn = (npml-i)/npml;
    xn  =Emp*pow(xxn,3.);
    gi2[i] = 1./(1.+xn);
    gi2[IE-i-1] = 1./(1.+xn);
    gi3[i] = (1.-xn)/(1.+xn);
    gi3[IE-i-1] = (1.-xn)/(1.+xn);
   xxn = (npml-i-.5)/npml;
    xn  = Emp*pow(xxn,3.);
```

```
  fi1[i] = xn;
  fi1[IE-i-2] = xn;
  fi2[i] = 1./(1.+xn);
  fi2[IE-i-2] = 1./(1.+xn);
  fi3[i] = (1.-xn)/(1.+xn);
  fi3[IE-i-2] = (1.-xn)/(1.+xn);
 }

 for ( j=0; j < n_pml; j++ ) {
  xxn = (npml-j)/npml;
  xn  = Emp*pow(xxn,3.);

  gj2[j] = 1./(1.+xn);
  gj2[JE-j-1] = 1./(1.+xn);
  gj3[j] = (1.-xn)/(1.+xn);
  gj3[JE-j-1] = (1.-xn)/(1.+xn);
 xxn = (npml-j-.5)/npml;
  xn  = Emp*pow(xxn,3.);
  fj1[j] = xn;
  fj1[JE-j-2] = xn;
  fj2[j] = 1./(1.+xn);
  fj2[JE-j-2] = 1./(1.+xn);
  fj3[j] = (1.-xn)/(1.+xn);
  fj3[JE-j-2] = (1.-xn)/(1.+xn);
 }



/* Specify the dielectric sphere */

  epsilon[0] = 1.;
  sigma[0]   = 0.;
        dell1[0]=0.;
        dell2[0]=0.;
        dell3[0]=0.;
        dell4[0]=0.;
        tau1[0]=0.;
        tau2[0]=0.;
        tau3[0]=0.;
        tau4[0]=0.;
  alpha1[0]=0.;
        alpha2[0]=0.;
        alpha3[0]=0.;
        alpha4[0]=0.;

    radius[1]=5;
                epsilon[1]=4.;
                sigma[1]=0.7;
                dell1[1]=56;
                dell2[1]=5200;
                dell3[1]=0.;
                dell4[1]=0.;
                tau1[1]=8.377e-12;
                tau2[1]=132.629e-9;
```

```
                tau3[1]=159.155e-6;
                tau4[1]=15.915e-3;
                alpha1[1]=0.1;
                alpha2[1]=0.1;
                alpha3[1]=0.2;
                alpha4[1]=0.;
                numsph=1;

/*      Calculate gaz,gbz  */
   for ( i = 0; i < IE; i++ ) {
  for ( j = 0; j < JE; j++ ) {
  eps = epsilon[0];
  cond = sigma[0];
         C1=pow(tau1[0]/dt,1-alpha1[0]);
         C2=pow(tau2[0]/dt,1-alpha2[0]);
         C3=pow(tau3[0]/dt,1-alpha3[0]);
         C4=pow(tau4[0]/dt,1-alpha4[0]);
         B1=alpha1[0];
         B2=alpha2[0];
         B3=alpha3[0];
         B4=alpha4[0];
         A1=dell1[0]/(1+C1);
         A2=dell2[0]/(1+C2);
         A3=dell3[0]/(1+C3);
         A4=dell4[0]/(1+C4);
        xdist = (ic-i);
        ydist = (jc-j);
      //  dist = sqrt(pow(xdist,2.) + pow(ydist,2.));
                                dist=sqrt(pow(ydist,2.));
        for (n=1; n<= numsph; n++) {
                                /*          if(n==3)
                                  {
                                        ydist=ydist-3;
                                   xdist=xdist-3;
                                        dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                                  }
                                  */
          if( dist <=radius[n]) {
          eps  =  epsilon[n] ;
          cond =  sigma[n];
                                  C1=pow(tau1[n]/dt,1-alpha1[n]);
                                  C2=pow(tau2[n]/dt,1-alpha2[n]);
                                  C3=pow(tau3[n]/dt,1-alpha3[n]);
                                  C4=pow(tau4[n]/dt,1-alpha4[n]);
                                  B1=alpha1[n];
                                  B2=alpha2[n];
                                  B3=alpha3[n];
                                  B4=alpha4[n];
                                  A1=dell1[n]/(1+C1);
                                  A2=dell2[n]/(1+C2);
                                  A3=dell3[n]/(1+C3);
                                  A4=dell4[n]/(1+C4);
        } }
        gcz[i][j]=1./(eps+(cond*dt/epsz)+A1+A2+A3+A4);
        gaz1[i][j]=(1-B1)*C1/(1+C1);
                                gaz2[i][j]=(1-B2)*C2/(1+C2);
```

```
                                gaz3[i][j]=(1-B3)*C3/(1+C3);
                                gaz4[i][j]=(1-B4)*C4/(1+C4);
                                gbz1[i][j]=0.5*(1-B1)*B1*C1/(1+C1);
                                gbz2[i][j]=0.5*(1-B2)*B2*C2/(1+C2);
                                gbz3[i][j]=0.5*(1-B3)*B3*C3/(1+C3);
                                gbz4[i][j]=0.5*(1-B4)*B4*C4/(1+C4);
                                gdz1[i][j]=A1;
                                gdz2[i][j]=A2;
        gdz3[i][j]=A3;
        gdz4[i][j]=A4;
                                gz[i][j]=cond*dt/epsz;
}}
                ////////

                for ( j = 0; j < JE; j++ ) {

eps = epsilon[0];
cond = sigma[0];
        C1=pow(tau1[0]/dt,1-alpha1[0]);
        C2=pow(tau2[0]/dt,1-alpha2[0]);
        C3=pow(tau3[0]/dt,1-alpha3[0]);
        C4=pow(tau4[0]/dt,1-alpha4[0]);
        B1=alpha1[0];
        B2=alpha2[0];
        B3=alpha3[0];
        B4=alpha4[0];
        A1=dell1[0]/(1+C1);
        A2=dell2[0]/(1+C2);
        A3=dell3[0]/(1+C3);
        A4=dell4[0]/(1+C4);


      ydist = (jc-j);
    //  dist = sqrt(pow(xdist,2.) + pow(ydist,2.));
                                dist=sqrt(pow(ydist,2.));

  //      dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                                 for (n=1; n<= numsph; n++) {

        if( dist<=radius[n]) {
        eps  =  epsilon[n];
        cond =  sigma[n] ;
                                  C1=pow(tau1[n]/dt,1-alpha1[n]);
                                  C2=pow(tau2[n]/dt,1-alpha2[n]);
                                  C3=pow(tau3[n]/dt,1-alpha3[n]);
                                  C4=pow(tau4[n]/dt,1-alpha4[n]);
                                  B1=alpha1[n];
                                  B2=alpha2[n];
                                  B3=alpha3[n];
                                  B4=alpha4[n];
                                  A1=dell1[n]/(1+C1);
                                  A2=dell2[n]/(1+C2);
                                  A3=dell3[n]/(1+C3);
                                  A4=dell4[n]/(1+C4);
        }
```

```
        }
                                inc_gcz[j]=1./(eps+(cond*dt/epsz)+A1+A2+A3+A4);
        inc_gaz1[j]=(1-B1)*C1/(1+C1);
                                inc_gaz2[j]=(1-B2)*C2/(1+C2);
                                inc_gaz3[j]=(1-B3)*C3/(1+C3);
                                inc_gaz4[j]=(1-B4)*C4/(1+C4);
                                inc_gbz1[j]=0.5*(1-B1)*B1*C1/(1+C1);
                                inc_gbz2[j]=0.5*(1-B2)*B2*C2/(1+C2);
                                inc_gbz3[j]=0.5*(1-B3)*B3*C3/(1+C3);
                                inc_gbz4[j]=0.5*(1-B4)*B4*C4/(1+C4);
                                inc_gdz1[j]=A1;
                                inc_gdz2[j]=A2;
        inc_gdz3[j]=A3;
        inc_gdz4[j]=A4;
        inc_gz[j]=cond*dt/epsz;



    }


  t0 = 40.0;
  spread = 10.0;
  T = 0;
  nsteps = 400;


  for ( n=1; n <=nsteps ; n++)  {
        //          pulse=1000*ratio*(exp(-alfa*(T))-exp(-beta*(T)));
    T = T + 1;
          pulse=exp(-(T-100)*(T-100)/100.0);


    NPR2=NPR1;
                NPR1=NCUR;
                NCUR=(NCUR+1)%3;
////////////////////////  with slab
                  for ( j=0; j < JE; j++ ) {

        dz_inc[j] =dz_inc[j]+0.5*( hx_inc[j-1]- hx_inc[j]);

    }


              dz_inc[3]=pulse+dz_inc[3];



        ////////////

  /* Calculate the Dz field */

  for ( i=1; i < IE; i++ ) {
    for ( j=1; j < JE; j++ ) {
```

```
        dz[i][j] = gi3[i]*gj3[j]*dz[i][j]
        + gi2[i]*gj2[j]*.5*(hy[i][j]- hy[i-1][j]- hx[i][j]+ hx[i][j-1]);
    } }



    /* Incident Dz */

    for ( i=ia; i <= ib; i++ ) {

        dz[i][ja] = dz[i][ja] + .5*hx_inc[ja-1];
        dz[i][jb] = dz[i][jb] - .5*hx_inc[jb];
    }

            for ( j=1; j < JE-1; j++ ) {

        B1=inc_gaz1[j]*inc_sz1[j][NPR1]+inc_gbz1[j]*inc_sz1[j][NPR2];
                                B2=inc_gaz2[j]*inc_sz2[j][NPR1]+inc_gbz2[j]*inc_sz2[j][NPR2];
        B3=inc_gaz3[j]*inc_sz3[j][NPR1]+inc_gbz3[j]*inc_sz3[j][NPR2];
                        B4=inc_gaz4[j]*inc_sz4[j][NPR1]+inc_gbz4[j]*inc_sz4[j][NPR2];
        ez_inc[j] = inc_gcz[j]*(dz_inc[j] - iz_inc[j]-B1-B2-B3-B4);
        iz_inc[i]= iz_inc[i] + inc_gz[i]*ez_inc[i];
        inc_sz1[j][NCUR]=B1+inc_gdz1[j]*ez_inc[j];
                                inc_sz2[j][NCUR]=B2+inc_gdz2[j]*ez_inc[j];
                                inc_sz3[j][NCUR]=B3+inc_gdz3[j]*ez_inc[j];
                                inc_sz4[j][NCUR]=B4+inc_gdz4[j]*ez_inc[j];


                               }

         //                    ez_inc[3] =ez_inc[3]+pulse;

                    //         ez_inc[3] =pulse;
   /* Boundary conditions for the incident buffer*/

    ez_inc[0] = ez_low_m2;
    ez_low_m2 = ez_low_m1;
    ez_low_m1 = ez_inc[1];

    ez_inc[JE-1]  = ez_high_m2;
    ez_high_m2 = ez_high_m1;
    ez_high_m1 = ez_inc[JE-2];

                                for ( j=0; j < JE-1; j++ ) {
      hx_inc[j] = hx_inc[j] + .5*( ez_inc[j] - ez_inc[j+1] );
     }


//          pulse =  exp(-.5*(pow((t0-T)/spread,2.0) ));
//                  pulse=100000*exp(-(T-100)*(T-100)/100.0);



    /* Calculate the E from D field */
    /* Remember: part of the PML is E=0 at the edges */
    for ( i=1; i < IE; i++ ) {
```

```
  for ( j=1; j < JE; j++ ) {


    B1=gaz1[i][j]*sz1[i][j][NPR1]+gbz1[i][j]*sz1[i][j][NPR2];
                    B2=gaz2[i][j]*sz2[i][j][NPR1]+gbz2[i][j]*sz2[i][j][NPR2];
    B3=gaz3[i][j]*sz3[i][j][NPR1]+gbz3[i][j]*sz3[i][j][NPR2];
              B4=gaz4[i][j]*sz4[i][j][NPR1]+gbz4[i][j]*sz4[i][j][NPR2];
    ez[i][j] = gcz[i][j]*(dz[i][j] - iz[i][j]-B1-B2-B3-B4);
    iz[i][j] = iz[i][j] + gz[i][j]*ez[i][j];
    sz1[i][j][NCUR]=B1+gdz1[i][j]*ez[i][j];
                    sz2[i][j][NCUR]=B2+gdz2[i][j]*ez[i][j];
                    sz3[i][j][NCUR]=B3+gdz3[i][j]*ez[i][j];
                    sz4[i][j][NCUR]=B4+gdz4[i][j]*ez[i][j];

} }

/* Calculate the Hx field */

for ( i=0; i < IE; i++ ) {
  for ( j=0; j < JE-1; j++ ) {
      curl_e = ( -ez[i][j+1] + ez[i][j]) ;
       ihx[i][j] = ihx[i][j]  + curl_e;
        hx[i][j] = fj3[j]*hx[i][j]
       + fj2[j]*.5*( curl_e + fi1[i]*ihx[i][j] );
} }


/* Incident Hx */

for ( i=ia; i <= ib; i++ ) {

    hx[i][ja-1] = hx[i][ja-1] + .5*ez_inc[ja];
    hx[i][jb]   = hx[i][jb]   - .5*ez_inc[jb];
}


/* Calculate the Hy field */

for ( i=0; i < IE-1; i++ ) {
  for ( j=0; j < JE; j++ ) {

      curl_e = ez[i+1][j] - ez[i][j];
      ihy[i][j] = ihy[i][j] + curl_e ;
      hy[i][j] = fi3[i]*hy[i][j]
      + fi2[i]*.5*( curl_e + fj1[j]*ihy[i][j] );
} }


/* Incident Hy */


for ( j=ja; j <= jb; j++ ) {

    hy[ia-1][j] = hy[ia-1][j] - .5*ez_inc[j];
    hy[ib][j]   = hy[ib][j]   + .5*ez_inc[j];
 }
```

```
////

        if((n%2)==0)
                {
                        printf("%d   \n ",n);


    fprintf(fp,"ZONE I=61, J=61, K=1, F=POINT\n");

                  int s;
                        for(int ys=0;ys<JE;ys++)
                                for(int xs=0;xs<IE;xs++)
                                fprintf(fp,"%d    %d    %lf \n",xs,ys,ez[ys][xs]);
                                fprintf(fp2,"ZONE \n");
                for(  s=0;s<JE;s++)
                        fprintf(fp2,"%d   %20.12lf\n",s,ez[30][s]);
                        fprintf(fp3,"ZONE \n");
                        for(  s=0;s<JE;s++)
                        fprintf(fp3,"%d   %20.12lf\n",s,ez_inc[s]);

                        if(n==130)
                        {
     for(  s=0;s<JE;s++)
                        fprintf(fp5,"%d   %20.12lf\n",s,ez_inc[s]);
                        }
    if(n==140)
                        {
     for(  s=0;s<JE;s++)
                        fprintf(fp6,"%d   %20.12lf\n",s,ez_inc[s]);
                        }
                if(n==150)
                        {
    for(  s=0;s<JE;s++)
                        fprintf(fp7,"%d   %20.12lf\n",s,ez_inc[s]);
                        }
                if(n==180)
                        {
     for(  s=0;s<JE;s++)
                        fprintf(fp8,"%d   %20.12lf\n",s,ez_inc[s]);
                        }
                if(n==200)
                        {
     for(  s=0;s<JE;s++)
                        fprintf(fp9,"%d   %20.12lf\n",s,ez_inc[s]);
                        }
                if(n==210)
                        {
     for(  s=0;s<JE;s++)
                        fprintf(fp10,"%d   %20.12lf\n",s,ez_inc[s]);
                        }
```

```
    }

//////////////////////////////with slab
//////////////////////////////without slab

          for ( j=0; j < JE; j++ ) {

        org_dz_inc[j] = org_dz_inc[j]+0.5*( org_hx_inc[j-1]-  org_hx_inc[j]);

    }


                org_dz_inc[3]=pulse+org_dz_inc[3];



        ////////////

    /* Calculate the Dz field */

    for ( i=1; i < IE; i++ ) {
      for ( j=1; j < JE; j++ ) {

        org_dz[i][j] = gi3[i]*gj3[j]* org_dz[i][j]
       + gi2[i]*gj2[j]*.5*( org_hy[i][j]-  org_hy[i-1][j]-  org_hx[i][j]+  org_hx[i][j-1]);
   } }



    /* Incident Dz */

    for ( i=ia; i <= ib; i++ ) {

      org_dz[i][ja] =  org_dz[i][ja] + .5* org_hx_inc[ja-1];
      org_dz[i][jb] =  org_dz[i][jb] - .5* org_hx_inc[jb];
    }

          for ( j=1; j < JE-1; j++ ) {

       B1= org_inc_gaz1[j]* org_inc_sz1[j][NPR1]+ org_inc_gbz1[j]* org_inc_sz1[j][NPR2];
                        B2= org_inc_gaz2[j]* org_inc_sz2[j][NPR1]+ org_inc_gbz2[j]*
org_inc_sz2[j][NPR2];
       B3= org_inc_gaz3[j]* org_inc_sz3[j][NPR1]+ org_inc_gbz3[j]* org_inc_sz3[j][NPR2];
                B4= org_inc_gaz4[j]* org_inc_sz4[j][NPR1]+ org_inc_gbz4[j]*
org_inc_sz4[j][NPR2];
       org_ez_inc[j] = org_inc_gcz[j]*(org_dz_inc[j] - org_iz_inc[j]-B1-B2-B3-B4);
       org_iz_inc[i]= org_iz_inc[i] + org_inc_gz[i]*org_ez_inc[i];
       org_inc_sz1[j][NCUR]=B1+org_inc_gdz1[j]*org_ez_inc[j];
                        org_inc_sz2[j][NCUR]=B2+org_inc_gdz2[j]*org_ez_inc[j];
                        org_inc_sz3[j][NCUR]=B3+org_inc_gdz3[j]*org_ez_inc[j];
                        org_inc_sz4[j][NCUR]=B4+org_inc_gdz4[j]*org_ez_inc[j];


                      }
```

```
        //                  ez_inc[3] =ez_inc[3]+pulse;

                    //          ez_inc[3] =pulse;
/* Boundary conditions for the incident buffer*/

 org_ez_inc[0] = org_ez_low_m2;
 org_ez_low_m2 = org_ez_low_m1;
 org_ez_low_m1 = org_ez_inc[1];

 org_ez_inc[JE-1]  = org_ez_high_m2;
 org_ez_high_m2 = org_ez_high_m1;
 org_ez_high_m1 = org_ez_inc[JE-2];

                          for ( j=0; j < JE-1; j++ ) {
    org_hx_inc[j] = org_hx_inc[j] + .5*( org_ez_inc[j] - org_ez_inc[j+1] );
   }


 for ( i=1; i < IE; i++ ) {
   for ( j=1; j < JE; j++ ) {


     B1=org_gaz1[i][j]*org_sz1[i][j][NPR1]+org_gbz1[i][j]*org_sz1[i][j][NPR2];
                          B2=org_gaz2[i][j]*org_sz2[i][j][NPR1]+org_gbz2[i][j]*org_sz2[i][j][NPR2];
     B3=org_gaz3[i][j]*org_sz3[i][j][NPR1]+org_gbz3[i][j]*org_sz3[i][j][NPR2];
                 B4=org_gaz4[i][j]*org_sz4[i][j][NPR1]+org_gbz4[i][j]*org_sz4[i][j][NPR2];
     org_ez[i][j] = org_gcz[i][j]*(org_dz[i][j] - org_iz[i][j]-B1-B2-B3-B4);
     iz[i][j] = iz[i][j] + gz[i][j]*ez[i][j];
     org_sz1[i][j][NCUR]=B1+org_gdz1[i][j]*org_ez[i][j];
                          org_sz2[i][j][NCUR]=B2+org_gdz2[i][j]*org_ez[i][j];
                          org_sz3[i][j][NCUR]=B3+org_gdz3[i][j]*org_ez[i][j];
                          org_sz4[i][j][NCUR]=B4+org_gdz4[i][j]*org_ez[i][j];

 } }
      /* Calculate the Fourier transform of Ex. */

 /* Calculate the incident field */


 /* Calculate the Hx field */

for ( i=0; i < IE; i++ ) {
  for ( j=0; j < JE-1; j++ ) {
       curl_e = ( -org_ez[i][j+1] + org_ez[i][j]) ;
        org_ihx[i][j] = org_ihx[i][j]  + curl_e;
         org_hx[i][j] = fj3[j]*org_hx[i][j]
        + fj2[j]*.5*( curl_e + fi1[i]*org_ihx[i][j] );
} }


 /* Incident Hx */

 for ( i=ia; i <= ib; i++ ) {

     org_hx[i][ja-1] = org_hx[i][ja-1] + .5*org_ez_inc[ja];
     org_hx[i][jb]   = org_hx[i][jb]   - .5*org_ez_inc[jb];
```

```
    }

    /* Calculate the Hy field */

    for ( i=0; i < IE-1; i++ ) {
      for ( j=0; j < JE; j++ ) {

          curl_e = org_ez[i+1][j] - org_ez[i][j];
          org_ihy[i][j] = org_ihy[i][j] + curl_e ;
          org_hy[i][j] = fi3[i]*org_hy[i][j]
          + fi2[i]*.5*( curl_e + fj1[j]*org_ihy[i][j] );
    } }

    /* Incident Hy */

    for ( j=ja; j <= jb; j++ ) {

        org_hy[ia-1][j] = org_hy[ia-1][j] - .5*org_ez_inc[j];
        org_hy[ib][j]   = org_hy[ib][j]   + .5*org_ez_inc[j];
      }



////

//////////////////////////////////////////////////////////////////////////////////without slab

  //////////energy
//
        amp0=org_ez[30][8];
                amp1=ez[30][8]-org_ez[30][8];
                amp2=ez[30][52];



        // calculate energy

                        if(count1%2==0)
                        {
                                for(i=ia;i<=ib;i++)
                                energy1=energy1+4*org_ez[30][8]*org_ez[30][8];
                        }
                        else
                        {
                                 for(i=ia;i<=ib;i++)
                                energy1=energy1+2*org_ez[30][8]*org_ez[30][8];
                        }
                        count1++;
        //          }

                    if(flag2==0)
                    {
```

```
        if(amp2!=0)
                flag2=1;

        }
//

        if(flag2==1)
        {
                if(count2%2==0)
                {
                        for(i=ia;i<=ib;i++)
                        energy2=energy2+4*ez[30][52]*ez[30][52];
                }
                else
                {
                        for(i=ia;i<=ib;i++)
                        energy2=energy2+2*ez[30][52]*ez[30][52];
                }
                count2++;
        }


         if(flag3==0)
        {
    if(amp1<0.0)
                flag3=1;

        }


        if(flag3==1)
        {
                if(count3%2==0)
                {
                        for(i=ia;i<=ib;i++)
                        {
                                amp1=ez[i][8]-org_ez[i][8];
                    energy3=energy3+4*amp1*amp1;
                        }
                }
                else
                {
                        for(i=ia;i<=ib;i++)
                        {
                                amp1=ez[i][8]-org_ez[i][8];
                                energy3=energy3+2*amp1*amp1;
                        }
                }
                count3++;
        }
        ////////////////energy


}
```

```
        fprintf(fp1,"%10.8f %10.8f %10.8f %10.8f %10.8f  %10.8f\n",(double)ww/60000.0,
2.0*thick*ddx,energy2/energy1,energy3/energy1,(energy1-energy2-energy3)/energy1,1-energy3/energy1);

        printf("%10.8f %10.8f %10.8f %10.8f %10.8f  %10.8f\n",(double)ww/60000.0,
2.0*thick*ddx,energy2/energy1,energy3/energy1,(energy1-energy2-energy3)/energy1,1-energy3/energy1);

}
}

        return 0;
}
```

APPENDIX C

Source Codes for 3D Simulation

```
// PML.cpp : Defines the entry point for the console application.
//
/*Fd3d_4.3.c. 3D FDTD, plane wave on a dielectric sphere. */
/* this program simulates the plane wave in 3 dimensions impinging on a dielectric cylinder
the input to the program are, npml = 14, number spheres = 1, radius of cylinder = 20,
epsilon = 30, conductivity = .3, and nsteps = 30, 40, 75, 100*/

#include "stdafx.h"
# include <math.h>
# include <stdlib.h>
# include <stdio.h>

#define IE  61
#define JE  61
#define KE  61
#define ia  5
#define ja  5
#define ka  5
#define NFREQS 10
#define Emp .33

double  dx[IE][JE][KE],dy[IE][JE][KE],dz[IE][JE][KE];
double  ex[IE][JE][KE],ey[IE][JE][KE],ez[IE][JE][KE];
double  hx[IE][JE][KE],hy[IE][JE][KE],hz[IE][JE][KE];
double  ix[IE][JE][KE],iy[IE][JE][KE],iz[IE][JE][KE];
double  gax1[IE][JE][KE],gay1[IE][JE][KE],gaz1[IE][JE][KE];
double  gax2[IE][JE][KE],gay2[IE][JE][KE],gaz2[IE][JE][KE];
double  gax3[IE][JE][KE],gay3[IE][JE][KE],gaz3[IE][JE][KE];
double  gax4[IE][JE][KE],gay4[IE][JE][KE],gaz4[IE][JE][KE];

double  gbx1[IE][JE][KE],gby1[IE][JE][KE],gbz1[IE][JE][KE];
double  gbx2[IE][JE][KE],gby2[IE][JE][KE],gbz2[IE][JE][KE];
double  gbx3[IE][JE][KE],gby3[IE][JE][KE],gbz3[IE][JE][KE];
double  gbx4[IE][JE][KE],gby4[IE][JE][KE],gbz4[IE][JE][KE];

double  gdx1[IE][JE][KE],gdy1[IE][JE][KE],gdz1[IE][JE][KE];
double  gdx2[IE][JE][KE],gdy2[IE][JE][KE],gdz2[IE][JE][KE];
double  gdx3[IE][JE][KE],gdy3[IE][JE][KE],gdz3[IE][JE][KE];
double  gdx4[IE][JE][KE],gdy4[IE][JE][KE],gdz4[IE][JE][KE];
double  gcx[IE][JE][KE],gcy[IE][JE][KE],gcz[IE][JE][KE];
double  gx[IE][JE][KE],gy[IE][JE][KE],gz[IE][JE][KE];
double  sx1[IE][JE][KE][3],sy1[IE][JE][KE][3],sz1[IE][JE][KE][3];
double  sx2[IE][JE][KE][3],sy2[IE][JE][KE][3],sz2[IE][JE][KE][3];
double  sx3[IE][JE][KE][3],sy3[IE][JE][KE][3],sz3[IE][JE][KE][3];
double  sx4[IE][JE][KE][3],sy4[IE][JE][KE][3],sz4[IE][JE][KE][3];



double  idxl[ia][JE][KE],idxh[ia][JE][KE];
double  ihxl[ia][JE][KE],ihxh[ia][JE][KE];
double  idyl[IE][ja][KE],idyh[IE][ja][KE];
double  ihyl[IE][ja][KE],ihyh[IE][ja][KE];
double  idzl[IE][JE][ka],idzh[IE][JE][ka];
double  ihzl[IE][JE][ka],ihzh[IE][JE][ka];
```

```
double  inc_gaz1[JE];
double  inc_gaz2[JE];
double  inc_gaz3[JE];
double  inc_gaz4[JE];
double  inc_gbz1[JE];
double  inc_gbz2[JE];
double  inc_gbz3[JE];
double  inc_gbz4[JE];

double  inc_gdz1[JE];
double  inc_gdz2[JE];
double  inc_gdz3[JE];
double  inc_gdz4[JE];
double  inc_gcz[JE];
double  inc_gz[JE];
double  inc_sz1[JE][3];
double  inc_sz2[JE][3];
double  inc_sz3[JE][3];
double  inc_sz4[JE][3];

int main(int argc, char* argv[])
{




   int n,i,j,k,ic,jc,kc,nsteps,n_pml;
   double  ddx,dt,epsz,muz,pi,npml,T;
   int ib,jb,kb,numsph;
         int    NCUR,NPR2,NPR1;
   double  xn,xxn,curl_e;
   double  t0,spread,pulse;
   double  ez_inc[JE],hx_inc[JE],dz_inc[JE],iz_inc[JE];

   double ez_low_m1,ez_low_m2,ez_high_m1,ez_high_m2;




    int ixh, jyh, kzh;

    double  gi1[IE],gi2[IE],gi3[IE];
    double  gj1[JE],gj2[JE],gj3[JE];
    double  gk1[KE],gk2[KE],gk3[KE];
    double  fi1[IE],fi2[IE],fi3[IE];
    double  fj1[JE],fj2[JE],fj3[JE];
    double  fk1[KE],fk2[KE],fk3[KE];

  double  radius[10],epsilon[10],sigma[10],eps,cond;
         double  dell1[10],dell2[10],dell3[10],dell4[10];
         double  tau1[10],tau2[10],tau3[10],tau4[10];
         double  alpha1[10],alpha2[10],alpha3[10],alpha4[10];
         double  A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4;
   double  dist,xdist,ydist,zdist,curl_h;
```

```
FILE *fp,*fp1,*fp2,*fp3,*fp5,*fp6,*fp7,*fp8,*fp9,*fp10 ;
ic = (IE-1)/2  ;
jc = (JE-1)/2 ;
kc = (KE-1)/2  ;
ib = IE - ia - 1;
jb = JE - ja - 1;
kb = KE - ka - 1;
pi = 3.14159;
epsz = 8.8e-12;
muz  = 4*pi*1.e-7;
ddx = 1e-6;              /* Cell size */
dt = ddx/6e8;            /* Time steps */
NCUR=2;
       NPR1=1;
       NPR2=0;
/* Initialize the arrays */
for ( j=0; j < JE; j++ ) {


  for ( k=0; k < KE; k++ ) {
     for ( i=0; i < IE; i++ ) {
     ex[i][j][k]= 0.0 ;
     ey[i][j][k]= 0.0 ;
     ez[i][j][k]= 0.0 ;
     dx[i][j][k]= 0.0 ;
     dy[i][j][k]= 0.0 ;
     dz[i][j][k]= 0.0 ;
     hx[i][j][k]= 0.0 ;
     hy[i][j][k]= 0.0 ;
     hz[i][j][k]= 0.0 ;
     ix[i][j][k]= 0.0 ;
     iy[i][j][k]= 0.0 ;
     iz[i][j][k]= 0.0 ;
                  gcx[i][j][k]= 1.0;
                  gcy[i][j][k]= 1.0;
                  gcz[i][j][k]= 1.0;

                  gx[i][j][k]=0.0;
                  gy[i][j][k]=0.0;
                  gz[i][j][k]=0.0;


                  gax1[i][j][k]=0.0;
                  gax2[i][j][k]=0.0;
                  gax3[i][j][k]=0.0;
                  gax4[i][j][k]=0.0;
                  gay1[i][j][k]=0.0;
                  gay2[i][j][k]=0.0;
                  gay3[i][j][k]=0.0;
                  gay4[i][j][k]=0.0;
                  gaz1[i][j][k]=0.0;
                  gaz2[i][j][k]=0.0;
                  gaz3[i][j][k]=0.0;
                  gaz4[i][j][k]=0.0;

                  gbx1[i][j][k]=0.0;
```

```
                    gbx2[i][j][k]=0.0;
                    gbx3[i][j][k]=0.0;
                    gbx4[i][j][k]=0.0;
                    gby1[i][j][k]=0.0;
                    gby2[i][j][k]=0.0;
                    gby3[i][j][k]=0.0;
                    gby4[i][j][k]=0.0;
                    gbz1[i][j][k]=0.0;
                    gbz2[i][j][k]=0.0;
                    gbz3[i][j][k]=0.0;
                    gbz4[i][j][k]=0.0;


                    gdx1[i][j][k]=0.0;
                    gdx2[i][j][k]=0.0;
                    gdx3[i][j][k]=0.0;
                    gdx4[i][j][k]=0.0;
                    gdy1[i][j][k]=0.0;
                    gdy2[i][j][k]=0.0;
                    gdy3[i][j][k]=0.0;
                    gdy4[i][j][k]=0.0;
                    gdz1[i][j][k]=0.0;
                    gdz2[i][j][k]=0.0;
                    gdz3[i][j][k]=0.0;
                    gdz4[i][j][k]=0.0;

  } } }

////////////////


        for ( j=0; j < JE; j++ ) {
      for ( k=0; k < KE; k++ ) {
       for ( i=0; i < IE; i++ ) {
                              for(int temp=0;temp<3;temp++){

        sx1[i][j][k][temp]=0.0;
                        sx2[i][j][k][temp]=0.0;
                                sx3[i][j][k][temp]=0.0;
                                sx4[i][j][k][temp]=0.0;
                                sy1[i][j][k][temp]=0.0;
                        sy2[i][j][k][temp]=0.0;
                                sy3[i][j][k][temp]=0.0;
                                sy4[i][j][k][temp]=0.0;
                                sz1[i][j][k][temp]=0.0;
                        sz2[i][j][k][temp]=0.0;
                                sz3[i][j][k][temp]=0.0;
                                sz4[i][j][k][temp]=0.0;
        } } }}

        ///


 for(j=0;j<JE;j++)
{
        ez_inc[j]=0;
```

```
        dz_inc[j]=0;
        hx_inc[j]=0;
        iz_inc[j]=0;
        inc_gcz[j]= 1.0;
        inc_gz[j]=0.0;
        inc_gaz1[j]=0.0;
        inc_gaz2[j]=0.0;
        inc_gaz3[j]=0.0;
        inc_gaz4[j]=0.0;
        inc_gbz1[j]=0.0;
        inc_gbz2[j]=0.0;
        inc_gbz3[j]=0.0;
 inc_gbz4[j]=0.0;
        inc_gdz1[j]=0.0;
        inc_gdz2[j]=0.0;
        inc_gdz3[j]=0.0;
        inc_gdz4[j]=0.0;
        }

        ez_low_m1=0;
        ez_low_m2=0;
        ez_high_m1=0;
        ez_high_m2=0;


       /////


for ( i=0; i < ia; i++ ) {
  for ( j=0; j < JE; j++ ) {
    for ( k=0; k < KE; k++ ) {
     idxl[i][j][k] = 0.0;
     idxh[i][j][k] = 0.0;
     ihxl[i][j][k] = 0.0;
     ihxh[i][j][k] = 0.0;
}  }  }

for ( i=0; i < IE; i++ ) {
  for ( j=0; j < ja; j++ ) {
    for ( k=0; k < KE; k++ ) {
     idyl[i][j][k] = 0.0;
     idyh[i][j][k] = 0.0;
     ihyl[i][j][k] = 0.0;
     ihyh[i][j][k] = 0.0;
}  }  }

for ( i=0; i < IE; i++ ) {
  for ( j=0; j < JE; j++ ) {
    for ( k=0; k < ka; k++ ) {
     idzl[i][j][k] = 0.0;
     idzh[i][j][k] = 0.0;
     ihzl[i][j][k] = 0.0;
     ihzh[i][j][k] = 0.0;
}  }  }
```

```
/*  Boundary Conditions */

    for ( i=0; i < IE; i++ ) {
    gi1[i] = 0.;
    fi1[i] = 0.;
    gi2[i] = 1.;
    fi2[i] = 1.;
    gi3[i] = 1.;
    fi3[i] = 1.;
    }

    for ( j=0; j < JE; j++ ) {
    gj1[j] = 0.;
    fj1[j] = 0.;
    gj2[j] = 1.;
    fj2[j] = 1.;
    gj3[j] = 1.;
    fj3[j] = 1.;
    }

    for ( k=0; k < IE; k++ ) {
    gk1[k] = 0.;
    fk1[k] = 0.;
    gk2[k] = 1.;
    fk2[k] = 1.;
    gk3[k] = 1.;
    fk3[k] = 1.;
    }
/*
     printf( "npml --> ");
     scanf("%lf", &npml);
     printf("%lf \n", npml);
     n_pml =(int) npml;
*/
            npml=5;
            n_pml=(int) npml;

    for ( i=0; i < n_pml; i++ ) {
     xxn = (npml-i)/npml;
     xn  = Emp*pow(xxn,3.);
     fi1[i] = xn;
     fi1[IE-i-1] = xn;
     gi2[i] = 1./(1.+xn);
     gi2[IE-i-1] = 1./(1.+xn);
     gi3[i] = (1.-xn)/(1.+xn);
     gi3[IE-i-1] = (1.-xn)/(1.+xn);
    xxn = (npml-i-.5)/npml;
     xn  = Emp*pow(xxn,3.);
     gi1[i] = xn;
     gi1[IE-i-2] = xn;
     fi2[i] = 1./(1.+xn);
     fi2[IE-i-2] = 1./(1.+xn);
     fi3[i] = (1.-xn)/(1.+xn);
     fi3[IE-i-2] = (1.-xn)/(1.+xn);
    }
```

```
for ( j=0; j < n_pml; j++ ) {
  xxn = (npml-j)/npml;
  xn  = Emp*pow(xxn,3.);
  fj1[j] = xn;
  fj1[JE-j-1] = xn;
  gj2[j] = 1./(1.+xn);
  gj2[JE-j-1] = 1./(1.+xn);
  gj3[j] = (1.-xn)/(1.+xn);
  gj3[JE-j-1] = (1.-xn)/(1.+xn);
 xxn = (npml-j-.5)/npml;
  xn  = Emp*pow(xxn,3.);
  gj1[j] = xn;
  gj1[JE-j-2] = xn;
  fj2[j] = 1./(1.+xn);
  fj2[JE-j-2] = 1./(1.+xn);
  fj3[j] = (1.-xn)/(1.+xn);
  fj3[JE-j-2] = (1.-xn)/(1.+xn);
}

for ( k=0; k < n_pml; k++ ) {
  xxn = (npml-k)/npml;
  xn  = Emp*pow(xxn,3.);
  fk1[k] = xn;
  fk1[KE-k-1] = xn;
  gk2[k] = 1./(1.+xn);
  gk2[KE-k-1] = 1./(1.+xn);
  gk3[k] = (1.-xn)/(1.+xn);
  gk3[KE-k-1] = (1.-xn)/(1.+xn);
 xxn = (npml-k-.5)/npml;
  xn  = Emp*pow(xxn,3.);
  gk1[k] = xn;
  gk1[KE-k-2] = xn;
  fk2[k] = 1./(1.+xn);
  fk2[KE-k-2] = 1./(1.+xn);
  fk3[k] = (1.-xn)/(1.+xn);
  fk3[KE-k-2] = (1.-xn)/(1.+xn);
}

/* Specify the dielectric sphere */

  epsilon[0] = 1.;
  sigma[0]   = 0.;
        del1[0]=0.;
        del2[0]=0.;
        del3[0]=0.;
        del4[0]=0.;
        tau1[0]=0.;
        tau2[0]=0.;
        tau3[0]=0.;
        tau4[0]=0.;
  alpha1[0]=0.;
        alpha2[0]=0.;
        alpha3[0]=0.;
        alpha4[0]=0.;
```

```
/*
    printf( "Number spheres --> ");
    scanf("%d", &numsph);
      printf( "numsph= %d \n ",numsph);

    for (n = 1; n <= numsph; n++) {
      printf( "Sphere radius (cells), epsilon, sigma --> ");
      scanf("%lf %lf %lf", &radius[n], &epsilon[n], &sigma[n]);
                    printf("Dell1,Dell2,Dell3,Dell4--->");
                    scanf("%lf %lf %lf %lf",&dell1[n],&dell2[n],&dell3[n],&dell4[n]);
                    printf("Tau1,Tau2,Tau3,Tau4---->");
                    scanf("%lf %lf %lf %lf",&tau1[n],&tau2[n],&tau3[n],&tau4[n]);
                    printf("Alpha1,Alpha2,Alpha3,Alpha4--->");
                    scanf("%lf %lf %lf %lf",&alpha1[n],&alpha2[n],&alpha3[n],&alpha4[n]);
      printf( "Radius = %10.2f  Eps = %10.2f Sigma = %10.8f \n ",
        radius[n],epsilon[n],sigma[n]);
    }

    for (n = 1; n <= numsph; n++) {
      printf( "Radius = %10.2f  Eps = %10.2f Sigma = %10.8f \n ",
      radius[n],epsilon[n],sigma[n]);
    }
*/
            radius[1]=10;
                    epsilon[1]=4.;
                    sigma[1]=0.7;
                    dell1[1]=56;
                    dell2[1]=5200;
                    dell3[1]=0.;
                    dell4[1]=0.;
                    tau1[1]=8.377e-12;
                    tau2[1]=132.629e-9;
                    tau3[1]=159.155e-6;
                    tau4[1]=15.915e-3;
                    alpha1[1]=0.1;
                    alpha2[1]=0.1;
                    alpha3[1]=0.2;
                    alpha4[1]=0.;
                    numsph=1;
  /*     Calculate gax,gbx  */
    for ( i = 0; i < IE; i++ ) {
    for ( j = 0; j < JE; j++ ) {
    for ( k = 0; k < KE; k++ ) {
    eps = epsilon[0];
    cond = sigma[0];
          C1=pow(tau1[0]/dt,1-alpha1[0]);
          C2=pow(tau2[0]/dt,1-alpha2[0]);
          C3=pow(tau3[0]/dt,1-alpha3[0]);
          C4=pow(tau4[0]/dt,1-alpha4[0]);
          B1=alpha1[0];
          B2=alpha2[0];
          B3=alpha3[0];
          B4=alpha4[0];
          A1=dell1[0]/(1+C1);
          A2=dell2[0]/(1+C2);
          A3=dell3[0]/(1+C3);
```

```
A4=dell4[0]/(1+C4);

ydist = (jc-j);
xdist = (ic-i-.5);
zdist = (kc-k);
dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                    //          dist=sqrt(pow(ydist,2.));
for (n=1; n<= numsph; n++) {
                /*
                                 if(n==3)
                                 {
                                        ydist=ydist-3;
                                 xdist=xdist-3;
                                        dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                                 }
                                 */
  if( dist <= radius[n]) {
  eps  =  epsilon[n];
  cond =  sigma[n] ;
                                 C1=pow(tau1[n]/dt,1-alpha1[n]);
                                 C2=pow(tau2[n]/dt,1-alpha2[n]);
                                 C3=pow(tau3[n]/dt,1-alpha3[n]);
                                 C4=pow(tau4[n]/dt,1-alpha4[n]);
                                 B1=alpha1[n];
                                 B2=alpha2[n];
                                 B3=alpha3[n];
                                 B4=alpha4[n];
                                 A1=dell1[n]/(1+C1);
                                 A2=dell2[n]/(1+C2);
                                 A3=dell3[n]/(1+C3);
                                 A4=dell4[n]/(1+C4);
  }

}
                                 gcx[i][j][k]=1./(eps+(cond*dt/epsz)+A1+A2+A3+A4);
gax1[i][j][k]=(1-B1)*C1/(1+C1);
                                 gax2[i][j][k]=(1-B2)*C2/(1+C2);
                                 gax3[i][j][k]=(1-B3)*C3/(1+C3);
                                 gax4[i][j][k]=(1-B4)*C4/(1+C4);
                                 gbx1[i][j][k]=0.5*(1-B1)*B1*C1/(1+C1);
                                 gbx2[i][j][k]=0.5*(1-B2)*B2*C2/(1+C2);
                                 gbx3[i][j][k]=0.5*(1-B3)*B3*C3/(1+C3);
                                 gbx4[i][j][k]=0.5*(1-B4)*B4*C4/(1+C4);
                                 gdx1[i][j][k]=A1;
                                 gdx2[i][j][k]=A2;
gdx3[i][j][k]=A3;
gdx4[i][j][k]=A4;
gx[i][j][k]=cond*dt/epsz;




}}}

/*      Calculate gay,gby  */
  for ( i = 0; i < IE; i++ ) {
```

```
for ( j = 0; j < JE; j++ ) {
for ( k = 0; k < KE; k++ ) {
eps = epsilon[0];
cond = sigma[0];
        C1=pow(tau1[0]/dt,1-alpha1[0]);
        C2=pow(tau2[0]/dt,1-alpha2[0]);
        C3=pow(tau3[0]/dt,1-alpha3[0]);
        C4=pow(tau4[0]/dt,1-alpha4[0]);
        B1=alpha1[0];
        B2=alpha2[0];
        B3=alpha3[0];
        B4=alpha4[0];
        A1=dell1[0]/(1+C1);
        A2=dell2[0]/(1+C2);
        A3=dell3[0]/(1+C3);
        A4=dell4[0]/(1+C4);
      xdist = (ic-i);
      ydist = (jc-j-.5);
      zdist = (kc-k);
      dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
              //                          dist=sqrt(pow(ydist,2.));
      for (n=1; n<= numsph; n++) {
                                /*  if(n==3)
                                  {
                                        ydist=ydist-3;
                                   xdist=xdist-3;
                                        dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                                  }
                                  */
        if( dist <= radius[n]) {
        eps  = epsilon[n] ;
        cond = sigma[n] ;
                                  C1=pow(tau1[n]/dt,1-alpha1[n]);
                                  C2=pow(tau2[n]/dt,1-alpha2[n]);
                                  C3=pow(tau3[n]/dt,1-alpha3[n]);
                                  C4=pow(tau4[n]/dt,1-alpha4[n]);
                                  B1=alpha1[n];
                                  B2=alpha2[n];
                                  B3=alpha3[n];
                                  B4=alpha4[n];
                                  A1=dell1[n]/(1+C1);
                                  A2=dell2[n]/(1+C2);
                                  A3=dell3[n]/(1+C3);
                                  A4=dell4[n]/(1+C4);
        }
      }
                                gcy[i][j][k]=1./(eps+(cond*dt/epsz)+A1+A2+A3+A4);
      gay1[i][j][k]=(1-B1)*C1/(1+C1);
                                gay2[i][j][k]=(1-B2)*C2/(1+C2);
                                gay3[i][j][k]=(1-B3)*C3/(1+C3);
                                gay4[i][j][k]=(1-B4)*C4/(1+C4);
                                gby1[i][j][k]=0.5*(1-B1)*B1*C1/(1+C1);
                                gby2[i][j][k]=0.5*(1-B2)*B2*C2/(1+C2);
                                gby3[i][j][k]=0.5*(1-B3)*B3*C3/(1+C3);
                                gby4[i][j][k]=0.5*(1-B4)*B4*C4/(1+C4);
                                gdy1[i][j][k]=A1;
```

```
                                    gdy2[i][j][k]=A2;
            gdy3[i][j][k]=A3;
            gdy4[i][j][k]=A4;
                                    gy[i][j][k]=cond*dt/epsz;

    }}}

/*      Calculate gaz,gbz  */
    for ( i = 0; i < IE; i++ ) {
   for ( j = 0; j < JE; j++ ) {
   for ( k = 0; k < KE; k++ ) {
   eps = epsilon[0];
   cond = sigma[0];
            C1=pow(tau1[0]/dt,1-alpha1[0]);
            C2=pow(tau2[0]/dt,1-alpha2[0]);
            C3=pow(tau3[0]/dt,1-alpha3[0]);
            C4=pow(tau4[0]/dt,1-alpha4[0]);
            B1=alpha1[0];
            B2=alpha2[0];
            B3=alpha3[0];
            B4=alpha4[0];
            A1=dell1[0]/(1+C1);
            A2=dell2[0]/(1+C2);
            A3=dell3[0]/(1+C3);
            A4=dell4[0]/(1+C4);
          xdist = (ic-i);
          ydist = (jc-j);
          zdist = (kc-k-.5);
          dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
         //                                  dist=sqrt(pow(ydist,2.));
          for (n=1; n<= numsph; n++) {
                                      /*          if(n==3)
                                        {
                                                ydist=ydist-3;
                                         xdist=xdist-3;
                                                dist = sqrt(pow(xdist,2.) + pow(ydist,2.) + pow(zdist,2.));
                                        }
                                        */
            if( dist <= radius[n]) {
            eps  =  epsilon[n] ;
            cond =  sigma[n];
                                        C1=pow(tau1[n]/dt,1-alpha1[n]);
                                        C2=pow(tau2[n]/dt,1-alpha2[n]);
                                        C3=pow(tau3[n]/dt,1-alpha3[n]);
                                        C4=pow(tau4[n]/dt,1-alpha4[n]);
                                        B1=alpha1[n];
                                        B2=alpha2[n];
                                        B3=alpha3[n];
                                        B4=alpha4[n];
                                        A1=dell1[n]/(1+C1);
                                        A2=dell2[n]/(1+C2);
                                        A3=dell3[n]/(1+C3);
                                        A4=dell4[n]/(1+C4);
          } }
          gcz[i][j][k]=1./(eps+(cond*dt/epsz)+A1+A2+A3+A4);
          gaz1[i][j][k]=(1-B1)*C1/(1+C1);
```

```
                                gaz2[i][j][k]=(1-B2)*C2/(1+C2);
                                gaz3[i][j][k]=(1-B3)*C3/(1+C3);
                                gaz4[i][j][k]=(1-B4)*C4/(1+C4);
                                gbz1[i][j][k]=0.5*(1-B1)*B1*C1/(1+C1);
                                gbz2[i][j][k]=0.5*(1-B2)*B2*C2/(1+C2);
                                gbz3[i][j][k]=0.5*(1-B3)*B3*C3/(1+C3);
                                gbz4[i][j][k]=0.5*(1-B4)*B4*C4/(1+C4);
                                gdz1[i][j][k]=A1;
                                gdz2[i][j][k]=A2;
        gdz3[i][j][k]=A3;
        gdz4[i][j][k]=A4;
                                gz[i][j][k]=cond*dt/epsz;
  }}}


t0 = 40.0;
spread = 10.0;
T = 0;
nsteps = 1;
  fp = fopen("Ez-t.dat","w");
       fprintf(fp,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp,"VARIABLES=X, Y, Ez\n");

              fp5 = fopen("130.dat","w");
       fprintf(fp5,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp5,"VARIABLES=X, Y, Z, Ez\n");
              fp6 = fopen("140.dat","w");
       fprintf(fp6,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp6,"VARIABLES= X, Y, Z, Ez\n");
              fp7 = fopen("150.dat","w");
       fprintf(fp7,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp7,"VARIABLES= X, Y, Z, Ez\n");
              fp8 = fopen("180.dat","w");
       fprintf(fp8,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp8,"VARIABLES= X, Y, Z, Ez\n");
              fp9 = fopen("200.dat","w");
       fprintf(fp9,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp9,"VARIABLES= X, Y, Z, Ez\n");
              fp10 = fopen("210.dat","w");
       fprintf(fp10,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp10,"VARIABLES= X, Y, Z, Ez\n");


       fp1 = fopen("data.dat","w");
       fprintf(fp1,"Rising Time Thickness Crossing    Ref    Absorb   peneration \n");

       fp2 = fopen("1dEz-t.dat","w");
       fprintf(fp2,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp2,"VARIABLES=Y, Ez\n");


       fp3 = fopen("Ez-t1.dat","w");
       fprintf(fp3,"TITLE=ELECTRIC FIELD\n");
       fprintf(fp3,"VARIABLES=Y, Ez\n");
```

```
//while ( nsteps > 0 ) {
          /*
     printf( "nsteps --> ");
     scanf("%d", &nsteps);
     printf("%d \n", nsteps);
    */
          nsteps=400;
   for ( n=1; n <=nsteps ; n++)  {
     T = T + 1;
      NPR2=NPR1;
                    NPR1=NCUR;
                    NCUR=(NCUR+1)%3;
/*  ----   Start of the Main FDTD loop ----  */

   /* Calculate the incident buffer */
  for ( j=0; j < JE; j++ ) {

          dz_inc[j] =dz_inc[j]+0.5*( hx_inc[j-1]- hx_inc[j]);

     }

                  pulse=exp(-(T-100)*(T-100)/100.0);
                  dz_inc[3]=pulse+dz_inc[3];




          ////////////
    /* Calculate the Dx field */

    for ( i=1; i < ia; i++ ) {
      for ( j=1; j < JE; j++ ) {
        for ( k=1; k < KE; k++ ) {
          curl_h = ( hz[i][j][k] - hz[i][j-1][k]
                - hy[i][j][k] + hy[i][j][k-1]) ;
          idxl[i][j][k] = idxl[i][j][k] + curl_h;
          dx[i][j][k] = gj3[j]*gk3[k]*dx[i][j][k]
        + gj2[j]*gk2[k]*.5*(curl_h + gi1[i]*idxl[i][j][k]);
    } } }

    for ( i=ia; i <= ib; i++ ) {
      for ( j=1; j < JE; j++ ) {
        for ( k=1; k < KE; k++ ) {
          curl_h = ( hz[i][j][k] - hz[i][j-1][k]
                - hy[i][j][k] + hy[i][j][k-1]) ;
          dx[i][j][k] = gj3[j]*gk3[k]*dx[i][j][k]
                + gj2[j]*gk2[j]*.5*curl_h ;
    } } }

    for ( i=ib+1; i < IE; i++ ) {
      ixh = i - ib - 1;
      for ( j=1; j < JE; j++ ) {
        for ( k=1; k < KE; k++ ) {
          curl_h = ( hz[i][j][k] - hz[i][j-1][k]
```

```
                - hy[i][j][k] + hy[i][j][k-1]) ;
        idxh[ixh][j][k] = idxh[ixh][j][k] + curl_h;
        dx[i][j][k] = gj3[j]*gk3[k]*dx[i][j][k]
      + gj2[j]*gk2[k]*.5*(curl_h + gi1[i]*idxh[ixh][j][k]);
} } }

/* Calculate the Dy field */

for ( i=1; i < IE; i++ ) {
  for ( j=1; j < ja; j++ ) {
    for ( k=1; k < KE; k++ ) {
      curl_h = ( hx[i][j][k] - hx[i][j][k-1]
            - hz[i][j][k] + hz[i-1][j][k]) ;
      idyl[i][j][k] = idyl[i][j][k] + curl_h;
      dy[i][j][k] = gi3[i]*gk3[k]*dy[i][j][k]
      + gi2[i]*gk2[k]*.5*( curl_h + gj1[j]*idyl[i][j][k]);
} } }

for ( i=1; i < IE; i++ ) {
  for ( j=ja; j <= jb; j++ ) {
    for ( k=1; k < KE; k++ ) {
      curl_h = ( hx[i][j][k] - hx[i][j][k-1]
            - hz[i][j][k] + hz[i-1][j][k]) ;
      dy[i][j][k] = gi3[i]*gk3[k]*dy[i][j][k]
            + gi2[i]*gk2[k]*.5* curl_h ;
} } }

for ( i=1; i < IE; i++ ) {
  for ( j=jb+1; j < JE; j++ ) {
    jyh = j - jb - 1;
    for ( k=1; k < KE; k++ ) {
      curl_h = ( hx[i][j][k] - hx[i][j][k-1]
            - hz[i][j][k] + hz[i-1][j][k]) ;
      idyh[i][jyh][k] = idyh[i][jyh][k] + curl_h;
      dy[i][j][k] = gi3[i]*gk3[k]*dy[i][j][k]
      + gi2[i]*gk2[k]*.5*( curl_h + gj1[j]*idyh[i][jyh][k]);
} } }

/* Incident Dy */

for ( i=ia; i <= ib; i++ ) {
  for ( j=ja; j <= jb-1; j++ ) {
    dy[i][j][ka]   = dy[i][j][ka]   - .5*hx_inc[j];
    dy[i][j][kb+1] = dy[i][j][kb+1] + .5*hx_inc[j];
} }


/* Calculate the Dz field */

for ( i=1; i < IE; i++ ) {
  for ( j=1; j < JE; j++ ) {
    for ( k=0; k < ka; k++ ) {
      curl_h = ( hy[i][j][k] - hy[i-1][j][k]
            - hx[i][j][k] + hx[i][j-1][k]) ;
      idzl[i][j][k] = idzl[i][j][k] + curl_h;
      dz[i][j][k] = gi3[i]*gj3[j]*dz[i][j][k]
```

```
        + gi2[i]*gj2[j]*.5*( curl_h + gk1[k]*idzl[i][j][k] );
} } }

 for ( i=1; i < IE; i++ ) {
   for ( j=1; j < JE; j++ ) {
     for ( k=ka; k <= kb; k++ ) {
       curl_h = ( hy[i][j][k] - hy[i-1][j][k]
             - hx[i][j][k] + hx[i][j-1][k]) ;
       dz[i][j][k] = gi3[i]*gj3[j]*dz[i][j][k]
              + gi2[i]*gj2[j]*.5* curl_h ;
} } }

 for ( i=1; i < IE; i++ ) {
   for ( j=1; j < JE; j++ ) {
     for ( k=kb+1; k < KE; k++ ) {
     kzh = k - kb - 1;
       curl_h = ( hy[i][j][k] - hy[i-1][j][k]
             - hx[i][j][k] + hx[i][j-1][k]) ;
       idzh[i][j][kzh] = idzh[i][j][kzh] + curl_h;
       dz[i][j][k] = gi3[i]*gj3[j]*dz[i][j][k]
       + gi2[i]*gj2[j]*.5*( curl_h + gk1[k]*idzh[i][j][kzh] );
} } }

 /* Incident Dz */

 for ( i=ia; i <= ib; i++ ) {
   for ( k=ka; k <= kb; k++ ) {
     dz[i][ja][k] = dz[i][ja][k] + .5*hx_inc[ja-1];
     dz[i][jb][k] = dz[i][jb][k] - .5*hx_inc[jb];
 } }

for ( j=1; j < JE-1; j++ ) {

     B1=inc_gaz1[j]*inc_sz1[j][NPR1]+inc_gbz1[j]*inc_sz1[j][NPR2];
                        B2=inc_gaz2[j]*inc_sz2[j][NPR1]+inc_gbz2[j]*inc_sz2[j][NPR2];
     B3=inc_gaz3[j]*inc_sz3[j][NPR1]+inc_gbz3[j]*inc_sz3[j][NPR2];
                B4=inc_gaz4[j]*inc_sz4[j][NPR1]+inc_gbz4[j]*inc_sz4[j][NPR2];
     ez_inc[j] = inc_gcz[j]*(dz_inc[j] - iz_inc[j]-B1-B2-B3-B4);
     iz_inc[i]= iz_inc[i] + inc_gz[i]*ez_inc[i];
     inc_sz1[j][NCUR]=B1+inc_gdz1[j]*ez_inc[j];
                        inc_sz2[j][NCUR]=B2+inc_gdz2[j]*ez_inc[j];
                        inc_sz3[j][NCUR]=B3+inc_gdz3[j]*ez_inc[j];
                        inc_sz4[j][NCUR]=B4+inc_gdz4[j]*ez_inc[j];


                       }
      /* Boundary conditions for the incident buffer*/

 ez_inc[0] = ez_low_m2;
 ez_low_m2 = ez_low_m1;
 ez_low_m1 = ez_inc[1];

 ez_inc[JE-1] = ez_high_m2;
 ez_high_m2 = ez_high_m1;
 ez_high_m1 = ez_inc[JE-2];
 /*  Source */
```

```
//          pulse =  exp(-.5*(pow((t0-T)/spread,2.0) ));
//                  pulse=100000*exp(-(T-100)*(T-100)/100.0);

//          for ( i=0; i < IE; i++ ) {
//       for ( k=0; k < KE; k++ ) {
//                              dz[i][6][k]=pulse;
//                  }
//          }

  /*   pulse =  sin(2*pi*400*1e6*dt*T);
     for ( k=kc-6; k <= kc+6; k++ ) {
       dz[ic][jc][k] = 0.;
     }
    pulse =  exp(-.5*(pow((t0-T)/spread,2.0) ));
     dz[ic][jc][kc] = pulse;*/

   //  printf("%4.0f  %6.2f \n ",T,pulse);

     /* Calculate the E from D field */
     /* Remember: part of the PML is E=0 at the edges */
     for ( i=1; i < IE-1; i++ ) {
       for ( j=1; j < JE-1; j++ ) {
         for ( k=1; k < KE-1; k++ ) {
         B1=gax1[i][j][k]*sx1[i][j][k][NPR1]+gbx1[i][j][k]*sx1[i][j][k][NPR2];
                              B2=gax2[i][j][k]*sx2[i][j][k][NPR1]+gbx2[i][j][k]*sx2[i][j][k][NPR2];
         B3=gax3[i][j][k]*sx3[i][j][k][NPR1]+gbx3[i][j][k]*sx3[i][j][k][NPR2];
                      B4=gax4[i][j][k]*sx4[i][j][k][NPR1]+gbx4[i][j][k]*sx4[i][j][k][NPR2];
         ex[i][j][k] = gcx[i][j][k]*(dx[i][j][k] - ix[i][j][k]-B1-B2-B3-B4);
         ix[i][j][k] = ix[i][j][k] + gx[i][j][k]*ex[i][j][k];
         sx1[i][j][k][NCUR]=B1+gdx1[i][j][k]*ex[i][j][k];
                              sx2[i][j][k][NCUR]=B2+gdx2[i][j][k]*ex[i][j][k];
                              sx3[i][j][k][NCUR]=B3+gdx3[i][j][k]*ex[i][j][k];
                              sx4[i][j][k][NCUR]=B4+gdx4[i][j][k]*ex[i][j][k];


                              B1=gay1[i][j][k]*sy1[i][j][k][NPR1]+gby1[i][j][k]*sy1[i][j][k][NPR2];
                              B2=gay2[i][j][k]*sy2[i][j][k][NPR1]+gby2[i][j][k]*sy2[i][j][k][NPR2];
         B3=gay3[i][j][k]*sy3[i][j][k][NPR1]+gby3[i][j][k]*sy3[i][j][k][NPR2];
                      B4=gay4[i][j][k]*sy4[i][j][k][NPR1]+gby4[i][j][k]*sy4[i][j][k][NPR2];
         ey[i][j][k] = gcy[i][j][k]*(dy[i][j][k] - iy[i][j][k]-B1-B2-B3-B4);
         iy[i][j][k] = iy[i][j][k] + gy[i][j][k]*ey[i][j][k];
         sy1[i][j][k][NCUR]=B1+gdy1[i][j][k]*ey[i][j][k];
                              sy2[i][j][k][NCUR]=B2+gdy2[i][j][k]*ey[i][j][k];
                              sy3[i][j][k][NCUR]=B3+gdy3[i][j][k]*ey[i][j][k];
                              sy4[i][j][k][NCUR]=B4+gdy4[i][j][k]*ey[i][j][k];


         B1=gaz1[i][j][k]*sz1[i][j][k][NPR1]+gbz1[i][j][k]*sz1[i][j][k][NPR2];
                              B2=gaz2[i][j][k]*sz2[i][j][k][NPR1]+gbz2[i][j][k]*sz2[i][j][k][NPR2];
         B3=gaz3[i][j][k]*sz3[i][j][k][NPR1]+gbz3[i][j][k]*sz3[i][j][k][NPR2];
                      B4=gaz4[i][j][k]*sz4[i][j][k][NPR1]+gbz4[i][j][k]*sz4[i][j][k][NPR2];
         ez[i][j][k] = gcz[i][j][k]*(dz[i][j][k] - iz[i][j][k]-B1-B2-B3-B4);
         iz[i][j][k] = iz[i][j][k] + gz[i][j][k]*ez[i][j][k];
         sz1[i][j][k][NCUR]=B1+gdz1[i][j][k]*ez[i][j][k];
                              sz2[i][j][k][NCUR]=B2+gdz2[i][j][k]*ez[i][j][k];
                              sz3[i][j][k][NCUR]=B3+gdz3[i][j][k]*ez[i][j][k];
```

```
sz4[i][j][k][NCUR]=B4+gdz4[i][j][k]*ez[i][j][k];



} } }


    /* Calculate the Fourier transform of Ex. */


/* Calculate the incident field */


  for ( j=0; j < JE-1; j++ ) {
   hx_inc[j] = hx_inc[j] + .5*( ez_inc[j] - ez_inc[j+1] );
  }


/* Calculate the Hx field */

for ( i=0; i < ia; i++ ) {
  for ( j=0; j < JE-1; j++ ) {
    for ( k=0; k < KE-1; k++ ) {
      curl_e = ( ey[i][j][k+1] - ey[i][j][k]
            - ez[i][j+1][k] + ez[i][j][k]) ;
      ihxl[i][j][k] = ihxl[i][j][k]  + curl_e;
       hx[i][j][k] = fj3[j]*fk3[k]*hx[i][j][k]
      + fj2[j]*fk2[k]*.5*( curl_e + fi1[i]*ihxl[i][j][k] );
} } }

 for ( i=ia; i <= ib; i++ ) {
   for ( j=0; j < JE-1; j++ ) {
     for ( k=0; k < KE-1; k++ ) {
       curl_e = ( ey[i][j][k+1] - ey[i][j][k]
             - ez[i][j+1][k] + ez[i][j][k]) ;
       hx[i][j][k] = fj3[j]*fk3[k]*hx[i][j][k]
              + fj2[j]*fk2[k]*.5*curl_e ;
} } }

 for ( i=ib+1; i < IE; i++ ) {
   ixh = i - ib-1;
   for ( j=0; j < JE-1; j++ ) {
     for ( k=0; k < KE-1; k++ ) {
       curl_e = ( ey[i][j][k+1] - ey[i][j][k]
             - ez[i][j+1][k] + ez[i][j][k]) ;
       ihxh[ixh][j][k] = ihxh[ixh][j][k]  + curl_e;
       hx[i][j][k] = fj3[j]*fk3[k]*hx[i][j][k]
       + fj2[j]*fk2[k]*.5*( curl_e + fi1[i]*ihxh[ixh][j][k] );
} } }

/* Incident Hx */

for ( i=ia; i <= ib; i++ ) {
  for ( k=ka; k <= kb; k++ ) {
    hx[i][ja-1][k] = hx[i][ja-1][k] + .5*ez_inc[ja];
```

```
    hx[i][jb][k]   = hx[i][jb][k]   - .5*ez_inc[jb];
} }


/* Calculate the Hy field */

for ( i=0; i < IE-1; i++ ) {
  for ( j=0; j < ja; j++ ) {
    for ( k=0; k < KE-1; k++ ) {
      curl_e = ( ez[i+1][j][k] - ez[i][j][k]
            - ex[i][j][k+1] + ex[i][j][k]) ;
      ihyl[i][j][k] = ihyl[i][j][k] + curl_e ;
      hy[i][j][k] = fi3[i]*fk3[k]*hy[i][j][k]
      + fi2[i]*fk3[k]*.5*( curl_e + fj1[j]*ihyl[i][j][k] );
} } }

for ( i=0; i < IE-1; i++ ) {
  for ( j=ja; j <= jb; j++ ) {
    for ( k=0; k < KE-1; k++ ) {
      curl_e = ( ez[i+1][j][k] - ez[i][j][k]
            - ex[i][j][k+1] + ex[i][j][k]) ;
       hy[i][j][k] = fi3[i]*fk3[k]*hy[i][j][k]
             + fi2[i]*fk3[k]*.5*curl_e ;
} } }

for ( i=0; i < IE-1; i++ ) {
  for ( j=jb+1; j < JE; j++ ) {
  jyh = j - jb-1;
    for ( k=0; k < KE-1; k++ ) {
      curl_e = ( ez[i+1][j][k] - ez[i][j][k]
            - ex[i][j][k+1] + ex[i][j][k]) ;
      ihyh[i][jyh][k] = ihyh[i][jyh][k] + curl_e ;
      hy[i][j][k] = fi3[i]*fk3[k]*hy[i][j][k]
      + fi2[i]*fk3[k]*.5*( curl_e + fj1[j]*ihyh[i][jyh][k] );
} } }

/* Incident Hy */

for ( j=ja; j <= jb; j++ ) {
  for ( k=ka; k <= kb; k++ ) {
    hy[ia-1][j][k] = hy[ia-1][j][k] - .5*ez_inc[j];
    hy[ib][j][k]   = hy[ib][j][k]   + .5*ez_inc[j];
} }


/* Calculate the Hz field */

for ( i=0; i < IE-1; i++ ) {
  for ( j=0; j < JE-1; j++ ) {
    for ( k=0; k < ka; k++ ) {
      curl_e = ( ex[i][j+1][k] - ex[i][j][k]
          -  ey[i+1][j][k] + ey[i][j][k] );
      ihzl[i][j][k] = ihzl[i][j][k] + curl_e;
      hz[i][j][k] = fi3[i]*fj3[j]*hz[i][j][k]
      + fi2[i]*fj2[j]*.5*( curl_e + fk1[k]*ihzl[i][j][k] );
} } }
```

```
for ( i=0; i < IE-1; i++ ) {
  for ( j=0; j < JE-1; j++ ) {
    for ( k=ka; k <= kb; k++ ) {
      curl_e = ( ex[i][j+1][k] - ex[i][j][k]
           - ey[i+1][j][k] + ey[i][j][k] );
      hz[i][j][k] = fi3[i]*fj3[j]*hz[i][j][k]
            + fi2[i]*fj2[j]*.5*curl_e ;
} } }

for ( i=0; i < IE-1; i++ ) {
  for ( j=0; j < JE-1; j++ ) {
    for ( k=kb+1; k < KE; k++ ) {
    kzh = k - kb - 1;
      curl_e = ( ex[i][j+1][k] - ex[i][j][k]
           - ey[i+1][j][k] + ey[i][j][k] );
      ihzh[i][j][kzh] = ihzh[i][j][kzh] + curl_e;
      hz[i][j][k] = fi3[i]*fj3[j]*hz[i][j][k]
      + fi2[i]*fj2[j]*.5*( curl_e + fk1[k]*ihzh[i][j][kzh] );
} } }


////
        if((n%2)==0)
            {


                    printf("%d  \n ",n);
 fprintf(fp,"ZONE I=61, J=61, K=1, F=POINT\n");

                    for(int ys=0;ys<JE;ys++)
                        for(int xs=0;xs<IE;xs++)
                        fprintf(fp,"%d    %d    %lf \n",xs,ys,ez[ys][xs][30]);


                    if(n>=140&&n<=280&&(n%4)==0)
                    {
                    fprintf(fp5,"ZONE I=61, J=61, K=31, F=POINT\n");
                    for(int ks=0;ks<31;ks++)
    for(int ys=0;ys<JE;ys++)
                        for(int xs=0;xs<IE;xs++)
                        fprintf(fp5,"%d    %d   %d    %lf \n",xs,ys,ks,ez[xs][ys][ks]);
                    }
 }
            ////////
            /*
     if(n==140)
                    {
                    fprintf(fp5,"ZONE I=61, J=61, K=31, F=POINT\n");
                    for(int ks=0;ks<31;ks++)
    for(int ys=0;ys<JE;ys++)
                        for(int xs=0;xs<IE;xs++)
                        fprintf(fp5,"%d    %d   %d    %lf \n",xs,ys,ks,ez[xs][ys][ks]);
                    }
                    */
     if(n==160)
                    {
```

```
                    fprintf(fp6,"ZONE I=61, J=61, K=31, F=POINT\n");
                    for(int ks=0;ks<31;ks++)
        for(int ys=0;ys<JE;ys++)
                            for(int xs=0;xs<IE;xs++)
                            fprintf(fp6,"%d   %d   %d   %lf \n",xs,ys,ks,ez[xs][ys][ks]);
                    }
                if(n==170)
                    {
                    fprintf(fp7,"ZONE I=61, J=61, K=31, F=POINT\n");
                    for(int ks=0;ks<31;ks++)
        for(int ys=0;ys<JE;ys++)
                            for(int xs=0;xs<IE;xs++)
                            fprintf(fp7,"%d   %d   %d   %lf \n",xs,ys,ks,ez[xs][ys][ks]);
                    }
        if(n==180)
                    {
                    fprintf(fp8,"ZONE I=61, J=61, K=31, F=POINT\n");
                    for(int ks=0;ks<31;ks++)
        for(int ys=0;ys<JE;ys++)
                            for(int xs=0;xs<IE;xs++)
                            fprintf(fp8,"%d   %d   %d   %lf \n",xs,ys,ks,ez[xs][ys][ks]);
                    }
                    if(n==200)
                    {
                    fprintf(fp9,"ZONE I=61, J=61, K=31, F=POINT\n");
                    for(int ks=0;ks<31;ks++)
        for(int ys=0;ys<JE;ys++)
                            for(int xs=0;xs<IE;xs++)
                            fprintf(fp9,"%d   %d   %d   %lf \n",xs,ys,ks,ez[xs][ys][ks]);
                    }
                    if(n==210)
                    {
                    fprintf(fp10,"ZONE I=61, J=61, K=31, F=POINT\n");
                    for(int ks=0;ks<31;ks++)
        for(int ys=0;ys<JE;ys++)
                            for(int xs=0;xs<IE;xs++)
                            fprintf(fp10,"%d   %d   %d   %lf \n",xs,ys,ks,ez[xs][ys][ks]);
                    }
                ////


    }
        fclose(fp);
/*  ----  End of the main FDTD loop ---- */


        return 0;
}
```

REFERENCES

1. Albanese, R., Blaschak, J., Medina, R. and Penn, J., Ultrashort electromagnetic signals: Biophysical questions, safety issues, and medical opportunities, *Aviat. Space Environ. Med.,* 1994, *65*, A116-120.

2. Lin, J.C., Interaction of electromagnetic transient radiation with biological materials, *IEEE Transactions on Electromagnetic Compatibility*, 1976, *17*, 93-97.

3. Lin, J.C., Electromagnetic pulse interaction with mammalian cranial structures, *IEEE Transactions of Biomedical Engineering*, 1976, *23*,61-65.

4. Lin, J.C., Lam, C.K., Coupling of Gaussian electromagnetic pulse into musle-bone model of biological structure, *Journal of Microwave Power*, 1976, *11*, 67-75.

5. Samn, S., Mathur, S.P., A mathematical model of a gigahertz transverse electromagnetic cell, *I. AFRL-HE-TR-1999-0219*, 1999,1-18.

6. Kunz, K.S., Luebbers, R.J., The Finite Difference Time Domain Method for Electromagnetics, 1993, Boca Raton, Florida: CRC Press.

7. Polk, C., Postow, E., CRC Handbook of Biololgical Effects of Electromagnetic Fields, 2nd ed., 1995, Boca Raton, Florida: CRC Press.

8. Sadiku, M.N.O., Numerical Techniques in Electromagnetics, 1992, Boca Raton, Florida: CRC Press.

9. Yee, K.S., Numerical solution of initial boundary value problems involving Maxwell's equations in isopropic media, *IEEE Trans. Antennas and propagation*, 1966, *14*, 302-307.

10. Bergenger, J.P., A perfectly matched layer for the absorption of electromagnetic waves, *J. Comp. Phys.*, 1994, *114*, 185-200.

11. Berenger, J.P., Three-dimensional perfectly matched layer for the absorption of electromagnetic waves, *J. Comp.Phys.*, 1996, *127*, 363-379.

12. Schoenbach, K.H., Beebe, S.J., Beuscher, E.S., Intracellular effect of ultrashort electrical pulses, *Bioelectromagnetics*, 2001, *22*, 440-448.

13. Foster, K.R., Thermal and nonthermal mechanisms of interaction of radio-frequency energy with biological systems, *IEEE Trans. Plasma Sci.*, 2000, *28*, 15-23.

14. Joshi, R.P., Schoenbach, K.H., Electroporation dynamics in biological cels subjected to ultrafast electrical pulses: a numerical simulation study, *Phy. Rev. E*, 2000, *62B*, 1025-1033.

15. Joshi, R.P., Hu, Q., Aly, R., Schoenbach, K.H., Hjalmarson, H.P., Self-consistent simulations of electroporation dynamics in biological cells su bjected to ultrashort electrical pulses. *Phys. Rev. E*, 2001, *64*, 011913.

16. Joshi, R.P., Hu, Q., Aly, R., Schoenbach, K.H., Hjalmarson, H.P., Self-consistent simulations of electroporation dynamics in biological cells su bjected to ultrashort electrical pulses. *Phys. Rev. E*, 2001, *64*, 011913.

17. Cole, K.S., Cole, R.H., Dispersion and absorption in dielectrics I. Alternating current characteristics, *J. Chemical Physics,* 1941, *9*, 341-351.

18. Gabriel, C., Gabriel, S., Corthout, E., The dielectric properties of biological tissues: I. Literature survey, *Phys. Med. Biol.*, 1996, *41*, 2231-2249.

19. Gabriel, S., Lau, R.W., Gabriel, C., The dielectric properties of biological tissues: II. Measurements in the frequency range 10Hz to 20GHz, *Phys. Med. Biol.*, 1996, *41*, 2251-2269.

20. Gabriel, S., Lau, R.W., Gabriel, C., The dielectric properties of biological tissues: III. Parametric models for the dielectric spectrum of tissues, *Phys. Med. Biol.*, 1996, *41*, 2271-2293.

21. Gabriel, C., Gabriel, S., Compilation of the dielectric properties of body tissues at RF and microwave frequencies, 1996, Preprint Al/OE-TR-1996-0037, Armstrong Laboratory Brooks AFB, San Antonio, Texas.

22. http://www.brooks.af.mil/AFRL/HED/hedr/reports/dielectric/Appendix.C/AppendixC.h

23. Hunt, W.D., Multiterm Debye dispersion relations for permittivity of muscle, *IEEE Trans. Biomed. Eng.*, 1985, *32*, 60-63.

24. Stoykov, N.S., Kuiken, T.A, Lowery, M.M., Taflove, A., Finite-element time domain algorithms fo rmodeling linear Debye and lorentz dielectric dispersion at lowfrequencies, *IEEE Transactions on Biomedical Engineering*, 2003, *50*, 1100-1107.

25. Taflove, A., Hagness, S.C., Computational Electrodynamics: The Finite-Difference Time-Domain Method, 2nd ed., Norwood, MA: Artech House, 2000,373-410.

26. Ida, N. Numerical Modeling for Electromagnetic Non-Destructive Evaluation, 1995, Newyork, Chapman&Hall.

27. Taflove, A., Brodwin, M.E., Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations, *IEEE Micro. Theo. Tech.*, Aug. 1975, *MTT-23, no.8*, 623-630.

28. Sullivan D.M., Electromagnetic Simulation Using the FDTD Method, 1999, New York: IEEE.

29. Taflove, A., Umashankar, K.R., Solution of complex electromagnetic penetration and scattering problems in unbounded region, in Kalinowski A.J.,*Computational Method for Infinite Domain Media-structure Interaction*. Washington, DC: ASME, 1981, *46*, 83-113.

30. Taflove, A., Application of the finite-difference time-domain method to sinusoidal steady-state electromagnetic-penetration problems, *IEEE Trans. EM Comp.*, Aug 1980, *EMC-22, no. 3*, 191-202.

31. Taflove, A., Brodwin, M.E., Computation of the electromagnetic fields and induced temperature within a model of the microwave irradiated human eye, *IEEE Micro. Theo. Tech.*, Aug. 1975, *MTT-23, no.11*, 888-896.

32. Taflove, A., Umashankar, K.R., A hybrid moment method/finite difference time domain approach to electromagnetic coupling and aperture penetration into complex geometries, *IEEE Trans. Ant. Prop.*, July 1982, *Ap-30, no. 4*, 617-627.

33. Umashankar, A., Taflove, A., A novel method to analyze electromagnetic scattering of complex objects, *IEEE Trans. EM Comp.*, Nov.1982, *EMC-24, no.4*, 397-405.

34. Taflove, A., Umashankar, K.R., The finite difference time domain method for numerical modeling for electromagnetic wave interactions, *Electromagnetics*, 1990, *10*, 105-126.

35. Taflove, A., Computation Electrodynamics: The Finite-Difference Time-Domain Method, 1995, Boston MA: Artech House.

36. Foster, K.R., Schwan, H.P., Dielectric properties of tissues and biological materials: a critical review, *Crit. Rev. Biomed. Eng.*, 1989, *17*, 25-104.

37. Joshi, R.P., Schoenbach, K.H., Mechanism for membrane electroporation irreversibility under high-intensity, ultrashort electrical pulse conditions, *Phys. Rev. E.*, 2002,*66*,052901.

38. Sullivan, D.M., Z transform theory and the FDTD method, *IEEE Trans.on Antennas and Propagat.*, Jan. 1996, *AP-44*, 28-34.

39. Sullivan, D.M., Nonlinear FDTD formulations using Z transforms, *IEEE Trans. Micorwave Theory and Tech.*, Mar. 1995, *MTT-43*, 676-682.

40. Simicevic, N., Haynie, D.T., FDTD Simulation of Exposure of Biological Material to Electromagnetic Nanopulses, *arXiv:physics/0407054*, 2004,1

41. Cheng, D.K., Field and Wave Electromagnetics, 1992, Menlo Park, CA: Addison-Wesley.

42. Sullivan, D.M., A simplified PML for use with the FDTD method, *IEEE Microwave and Guided Wave Letters*, 1996, *6*, 97-99.

43. Sullivan, D.M., An unsplit step 3-D PML for use with the FDTD method, *IEEE Microwave and Guided Wave Letters*, 1997, *7*, 184-186.

44. Sacks, Z.S., Kingsland, D.M., Lee, R. Lee, J.F.,A perfectly matched anisotropic absorber for use as an absorbing boundary condition, *IEEE Trans. Anten. And Prop.*, 1995, *43*, 1460-1463.

45. Cook, H.F., The dielectric behavior of some types of human tissues at microwave frequencies, *British J. Appl. Physics*, 1951, *2*, 295-300.

46. Li, X., Hagness, S.C., Choi, M.K., van der Weide, D.W., Numerical and experimental investigation of an ultrawideband ridged pyramical horn antenna with curved launching plane for pulse radiation, *IEEE Antennas and Wireless Propagation Letters,* 2003, *2*, 259-262.

47. Vernier, P.T., Sun Y., Marcu, L., Craft, C.M., Gundersen, M.A., Nanoelectropulse-induced phosphatidylserine translocation, *Biophys.J.*, 2004, *86*,4040-4048.

48. Seaman, R.L., Belt, M.L., Doyle, J.M., Mathur, S.P., Ultra-wideband electromagnetic pulses and morphine-induced changes in nociception and activity in mice, *Physiol. Behav.*, 1998, *65*, 263-270.

49. Young L.A., Boehm R.F., A Finite Difference Heat Transfer Analysis of a Percutaneous Transluminal Microwave Angioplasty System, *J. Biomechanical Engineering*, 1993, *115*, 441-446