University of Southern Queensland

Faculty of Engineering and Surveying

The Finite Difference Time Domain Method for Computational Electromagnetics

A dissertation submitted by

CHAN, Auc Fai

in fulfillment of the requirements of

Courses ENG4111 and 4112 Research Project

towards the degree of

Bachelor of Engineering (Electrical and Electronic)

Submitted: November, 2006

Abstract

Three computer programs are presented in this dissertation. The first one is a one-dimensional (1D) program, simulating a sinusoidal signal propagating along the x-axis. The second one is a two-dimensional (2D) program, simulating radiation from a narrow slit. The third one is also a 2D program, which is a simulation of a Time Domain Reflectometer (TDR) probe.

A graphical user interface (GUI) is included in the 1D program, which makes the computer program user-friendly. A flowchart with details is included in the appendices.

When applying FDTD to problems where solution regions are unbounded, difficulty arises. Since no computers can store unlimited amount of data, it is necessary to somehow, limit the solution regions. Mur’s first-order absorbing boundary condition (ABC) is a method to achieve this. The derivation of Mur’s ABC is given in details in Chapter Four, and implemented in the 1D program. This program serves as an introduction to FDTD.

The 2D program simulating radiation from a narrow slit has two interesting aspects. The perfectly matched layer (PML) is expected to absorb all electromagnetic (EM) waves. No EM waves are supposed to pass through a perfect electric conductor (PEC). Surprisingly, EM waves do penerate through the connecting points between PEC and PML. The reason and solution for this ‘leakage’ are given in this dissertation. Also one might find it a bit confusing to observe that in this program, the x-component of electric field seems to be propagating along the y-direction. The clarification is given in this dissertation.

TDR can be used to measure the amount of moisture in soil. In the TDR program, it is required to simulate a 2D model of a TDR probe of given geometry, and to specify the time constant of a raised cosine function. A plane wave using the raised cosine function is then introduced into the TDR probe.

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

CHAN, Auc Fai

Student Number: 0031138976

____
Signature

____
Date

Acknowledgements

The computer programs in this dissertation are modifications of programs written by Dr. Susan C. Hagness supplied with the book Computational Electrodynamics. The authors of the book are Dr. Allen Taflove and Dr. Susan C. Hagness.

I wish to express my grateful thanks to Dr. Jim Ball for his careful readings of the manuscript, and for many comments and suggestions which have substantially improved the dissertation.

Table of Contents

List of Figures

List of Appendices

Abbreviations

1D :	One-Dimensional
2D :	Two-Dimensional
3D :	Three-Dimensional
ABC :	Absorbing Boundary Condition
EM:	Electromagnetic
FDTD :	Finite-Difference Time-Domain
FEM:	Finite Element Method
GUI :	Graphical User Interface
MOM :	Method of Moments
PEC :	Perfect Electric Conductor
PML :	Perfectly Matched Layer
RF:	Radio Frequency
TDR :	Time Domain Reflectometer

Chapter 1: Introduction

1.1 Background Information

James Clerk Maxwell formulated Maxwell’s equations in 1873 (Umashankar & Taflove, 1995). Exact analytical solutions of Maxwell’s equations were limited to simple and symmetrical structures. Some classical methods of obtaining analytical solutions included separations of variables and conformal transformations.

Separation of variables is applied only to electrically small shapes that can be fitted into ortho-normal coordinate systems. For this reason, separation of variables has limited usage on solving electromagnetic problems.

Conformal transformation is useful for analytical solution of Laplacian fields with complicated shape. The chief limitation in this method is that equipotential boundaries coincident with flux line are assumed. The other major disadvantage of this method is to find a suitable transformation equation that converts a simple boundary into the complicated boundary containing the required field to be solved.

In 1881, Rayleigh succeeded in solving exactly, the scattering of an electromagnetic wave by a perfect conducting cylinder. In 1891, an approximate method was devised by Kirchoff for the diffraction through a hole in a thin screen.

From 1930s to 1950s, asymptotic series were used to solve some of the electromagnetic problems. In 1962, Keller developed the geometrical theory of diffraction (GTD), by using asymptotic series. However, GTD has many disadvantages. Nonphysical field infinities at focal points, caustics and shadow boundaries are some of the defects in GTD. Different diffraction coefficients are needed for different structures. GTD is not suitable for structures in which rays can bounce repeatedly.

In 1960s, the advancement of computer technology and the increase of military defense and industrial needs prompted the researchers to investigate the use of numerical methods on solving electromagnetic problems (Umashankar & Taflove, 1995). Some of the popular numerical methods are methods of moments (MOM), finite element methods (FEM) and finite difference time domain method (FDTD).

1.1.1 Classical methods to solve Maxwell's equations

The classical methods to solve Maxwell's equations are conformal transformations and separation of variables.

a) Conformal Transformations

The analytic function,

$$z = f(t) = x(u, v) + jy(u, v)$$

defines a complex variable z = x+jy as a function of another complex variable t = u+jv.

Transformations described by the above general equation are called conformal transformations.

A detailed description of conformal transformations is given in Chapter 2: Literature Review.

Here in this section, only the limitations and difficulty of conformal transformations are emphasized.

The limitation of conformal transformations is that, boundaries have to coincide with equipotential lines or flux lines inclusively; i.e. boundaries have to either coincide with equipotential lines, or coincide with flux lines, or coincide with both equipotential and flux lines (Binns, Lawrenson & Trowbridge 1994, pp. 121-122).

Furthermore, conformal transformations are limited to solve two-dimensional electrostatic problems only.

It is also rather difficult to find the transformation equation that converts a simple boundary into more complicated boundary containing the field which is required to solve (Binns, Lawrenson & Trowbridge 1994, p.125).

b) Separation of variables

To obtain the analytical solution of Laplace's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 1,$$

let u to be the product of two functions,

$$u(x,y) = X(x)\ Y(y)$$

Separation of variables means assuming that, the solution for u is the product of two functions, one of which is a function of x only, and the other is a function of y only.

Again, detailed description of separation of variables is given in Chapter 2: Literature Review, and only the limitation is emphasized here.

The separation of variables can only be applied to problems that fit into orthonormal coordinate systems. This type of problems are limited in number (Umashankar & Taflove 1993, p.2).

1.1.2 Numerical Methods to solve Maxwell's equations

In order to overcome the limitations of classical methods, numerical methods were developed. The commonly used methods are finite element methods (FEM), methods of moments (MOM) and finite difference time domain (FDTD) methods.

1.2 Finite Element Method (FEM)

The origin of FEM traced back to Courant in 1943, who used triangular elements and the principle of minimum potential energy, in the appendix of his paper (Huebner, Thornton & Byron 1995, p.9) also (Volakis, Chatterjee & Kempel 1998, p.65).

In 1954, Argyris and his collaborators began to publish a series of papers on FEM covering linear structure analysis (Huebner, Thornton & Byron 1995, p.11).

The advantage of FEM is its capability to handle problems of complex geometries and inhomogeneous materials (Sadiku 2003, p.377).

FEM analysis of problems involves the following steps:

- defining the problem's computational domain;
- choosing the shape of discrete elements;
- generation of mesh (preprocessing);
- applying wave equation on each element;
- for statics, applying Laplace's/Poisson's equations on each element;
- applying boundary conditions;
- assembling element matrices to form overall sparse matrix;
- solving matrix system;
- postprocessing field data to extract parameters, such as capacitance, impedance, radar cross section and so on (Volakis, Chatterjee & Kempel 1998, p.68).

1.3 Method of Moments (MOM)

Consider the following inhomogeneous equation

$$L\varphi = g$$

where L is an operator which can be differential or integral; g is the known excitation or source function; and φ is the response or unknown function to be found. It is assumed that L and g are given and the task is to determine φ (Harrington 1993, pp. 1-2).

The procedure of solving the above equation as presented by Roger F. Harrington is called method of moments. The procedure expands φ as a series of functions

$$\varphi = \sum_n a_n \varphi_n$$

and multiples φ by a set of weighting functions w_n and finally solves for a_n (Harrington 1993, pp. 5-6).

In other words, MOM reduces $L\varphi = g$ into matrix equations by using a method known as the method of weighted residuals.

MOM has been successfully used to model:

- scattering by wires and rods;
- scattering by two-dimensional (2D) metallic and dielectric cylinders;
- scattering by three-dimensional (3D) metallic and electric objects of arbitrary shapes; and
- many other scattering problems.

MOM requires significant computer memories (Umashankar & Taflove 1993, p.4) also (Sadiku 2003, p.285).

MOM is not suitable for analyzing the interior of conductive enclosures (Todd 1991, p.7).

1.4 Finite Difference Time Domain (FDTD) method

The Finite Difference Time Domain (FDTD) method is an application of the finite difference method, commonly used in solving differential equations, to solve Maxwell's equations. In FDTD, space is divided into small portions called cells. On the surfaces of each cell, there are assigned points. Each point in the cell is required to satisfy Maxwell's equations. In this way, electromagnetic waves are simulated to propagate in a numerical space, almost as they do in real physical world. FDTD is one of the commonly used methods to analyse electromagnetic phenomena at radio and microwave frequencies. Computer programs written in MATLAB can display electromagnetic phenomena in movies.

The basic algorithm of FDTD was first proposed by K.S. Yee in 1966. In 1975, Allen Taflove and M.E. Brodwin obtained the correct criterion for numerical stability for Yee's algorithm and, firstly solved the sinusoidal steady-state electromagnetic scattering problems, in two- and three-dimensions. This solution becomes a classical computer program example in many FDTD textbooks. Many researchers follow; and among them are G. Mur and J.P. Berenger. Mur published the first numerically stable absorbing boundary condition (ABC) in 1981. Berenger introduced, the best ABC for the time being, the perfectly matched layer (PML) in 1994.

1.4.1 Comparison between FDTD and FEM

FDTD is good for Radio Frequency (RF) and microwave applications. It is not too difficult to write computer programs to implement FDTD and there is no need to solve matrix equations. The final results of FDTD computer programs are displayed in movies and therefore easy to understand, and good for physical insight. As a result, FDTD is suitable for teaching purposes.

FEM is more suitable for low frequency of 50 Hz. It uses fewer elements than FDTD. FEM needs more preprocessing and postprocessing, as mentioned in section 1.2.

1.4.2 Comparison between FDTD and MOM

FDTD permits the modeling of electromagnetic wave interactions with a level of detail as high as that of MOM. For FDTD, the overall computer storage and running time requirements are linearly proportional to N, the number of field unknowns, but MOM requires more storage and running time. MOM leads to systems of linear equations having dense, full coefficient matrices, which require computer storage proportional to N^2, N being the number of field unknowns. The computer execution time to invert the MOM system matrix has N^2 to N^3 dependence (Umashankar & Taflove 1993, pp. 4-5).

Although FDTD has computer running time proportional to N, and MOM has computer execution time of N^2 to N^3 dependence, FDTD has very large N, and MOM can have quite small N.

1.4.3 Comparison between FDTD and Frequency-Domain Techniques

After 1960, the advent of digital computers enabled the frequency-domain techniques to become quite sophisticated. In this period, the main computational approaches for Maxwell's equations were high-frequency asymptotic methods and integral equations, both being frequency-domain techniques.

However, these frequency-domain techniques have many shortcomings.
Asymptotic method is suitable for modeling the scattering properties of electrically large complex shapes. However, if has difficulty in treating nonmetallic material composition, and volumetric complexity of a structure.

Integral equations can handle material and structural complexity. However, they lead to systems of linear equations, which limits the electrical size of possible models.

Although progress has been made in solving huge systems of equations generated by frequency-domain integral equations, the capabilities of even the latest technologies to solve them, are exhausted by many structures of engineering interest (Taflove & Hagness 2000, pp. 3-4).

1.4.4 Justification for the Use of FDTD

In 1970s and 1980s, researchers realized the limitations of frequency-domain techniques. This led to an alternative approach: the FDTD.

The advantages of FDTD are as follows:

- Frequency-domain integral-equation and finite-element methods can model 10^6 electromagnetic field unknowns. FDTD can model 10^9 unknowns.
- In FDTD, specifying a new structure is only a problem of mesh generation, which is much easier than the complex reformulation of an integral equation.
- The rapid development of computer visualization capabilities is beneficial to FDTD. FDTD generates time-marched arrays of fields and is suitable to be displayed as movies (Taflove & Hagness 2000, pp. 3-4).

1.5 Broad Aims and Specific Objectives

The aims of this project are to search in the Internet for available software for solving electromagnetic problems, and to write computer programs to display the phenomena of electromagnetic wave propagations in the form of movies for the purpose of teaching.

The specific objectives are:

(a) to write a one-dimensional Finite Difference Time Domain (FDTD) simulation of Absorbing Boundary Condition;

(b) to write a two-dimensional FDTD simulation modeling the radiation from a narrow slit excited by a plane wave which is a bandpass pulse;

(c) to write a two-dimensional FDTD simulation of a Time Domain Reflectometer probe (TDR) excited by a DC pulsed plane wave having a raised cosine time waveform.

1.6 Dissertation Overview

The first chapter is to provide the background information about the historical development of solutions for Maxwell’s equations. The project aims and objectives are also specified.

Chapter 2: Literature Review is to acquire enough knowledge about FDTD to support the computer programming in FDTD.

Chapter 3: Review of Available FDTD Software Packages is a survey on some commercial and educational packages, focusing on general features and typical processing times

Chapter 4: Theory of FDTD is a brief description of all the theories required before the programs are designed and constructed.

Chapter 5: Application of FDTD to one-dimensional ABC is about 1D-computer programs. It is a good introduction to FDTD, and is useful as a training exercise.

Chapter 6: Application of FDTD to Radiation from a Narrow Slit is to write a program to show a 2D model of radiation from a narrow slit excited by a plane wave, which is a bandpass pulse.

Chapter 7: Application of FDTD to Time Domain Reflectometer (TDR) is to simulate a 2D model of a TDR probe of given geometry.

Chapter 8: Suggestions of Further Work and Conclusions is about further work that can be done directly following this project. More researches on FDTD will be carried out in the future.

Chapter 2: Literature Review of FDTD

2.1 Introduction

The objectives of this literature review are to acquire enough knowledge about FDTD to support the computer programming in FDTD, and to acquire knowledge relevant to this project.

The central part of this literature review, the theory of FDTD, because of its importance, deserves a single chapter. It is described in Chapter 4.

A summary of sections in this review are given below :
Section 2.2 Conformal Transformations, and section 2.3 Separation of Variables, are analytical methods for solving Maxwell’s equations. These two sections present introductions to these two methods.

Section 2.4 Numerical Dispersion and Stability, and section 2.5 Incident Wave Source Conditions, are book reviews on Computational Electrodynamics, by A. Taflove and S.C. Hagness.

Allen Taflove has pioneered the FDTD method since 1975 (the year in that he published his PhD dissertation), and is a leading authority in FDTD. He is one of the most-cited researchers in FDTD (Taflove & Hagness, 2000).

Therefore reading of Taflove’s book is essential. Two of the chapters from the book are presented in literature review.

Section 2.6 Soil Water Content Using TDR, is a review on a research paper. The paper “Electromagnetic Determination of Soil Water Content Using TDR,” by Topp et al., 1982, inspired the part on TDR, of this project.

Section 2.7 Recent Developments of FDTD, presents recent trends of developments, based on seven journal papers.

2.2 Conformal Transformations

When there are no charges in space, the differential form of Gauss' law is,

Div **E** = 0

Furthermore, in a purely electrostatic field, **E** may be expressed as minus the gradient of the potential U:

E = -**grad** U

Therefore div **grad** U = 0

It is convenient to think of div **grad** as a single differential operator, which is called the Laplacian.

Laplacian U = 0,

which is Laplace's equation.

Laplace's equation can be solved analytically by conformal transformations.

The analytic function,

$$z = f(t) = x(u,v) + jy(u,v)$$

defines a complex variable z = x+jy as a function of another complex variable t = u+jv.

Transformations described by the above general equation, are called conformal transformations.

Let us consider an example of conformal transformation.

Suppose z = sin t

It has already been defined that z = x+jy and t = u+jv

Therefore x+jy = sin t

= sin(u+jv)

= sin u cos jv + cos u sin jv

= sin u cosh v + jcos u sinh v

Equating the real and imaginary parts,

x = sin u cosh v

y = cos u sinh v

Squaring and adding these equations eliminates u, giving

$$\frac{x^2}{\cosh^2 v} + \frac{y^2}{\sinh^2 v} = 1,$$

which is the equation of an ellipse.

Squaring and subtracting them eliminates v, so that

$$\frac{x^2}{\sin^2 u} - \frac{y^2}{\cos^2 u} = 1,$$

which is the equation of a hyperbola.

Therefore a straight line described by v = constant is transformed into an ellipse; while a straight line described by u = constant is transformed into a hyperbola.

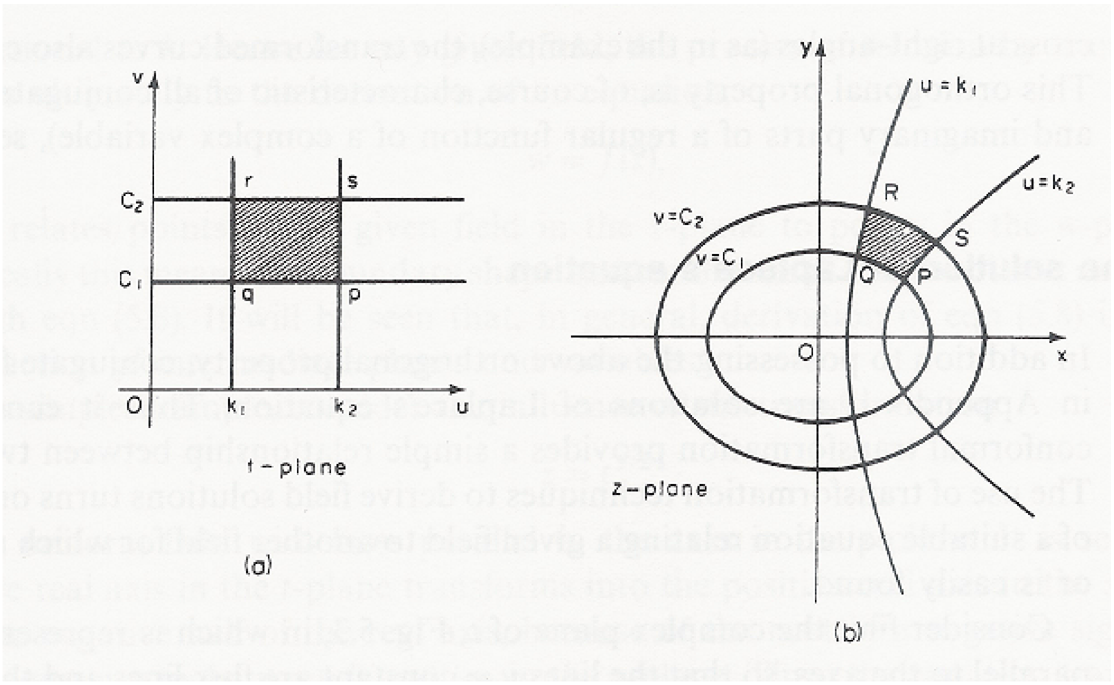


Figure 2.1 Conformal Transformation

Properties of conformal transformations are summarized below:

- For any two intersecting curves, the angle of intersection is unaltered by the conformal transformation.
- To pass from one arm of the angle to the other arm, one must travel around the point of intersection in the same sense.
- In other words, if two curves intersect at a given angle in the t-plane, the transformed curves in the z-plane intersect at the same angle, and in such a way that the senses of the two angles are the same.

- If an analytic function satisfies Laplace's equation in some region R' of the t-plane, the transformation of that analytic function still satisfies Laplace's equation in the appropriate region R of the z-plane.
- Expressing in another way, if possible to find a solution of Laplace's equation in a region of the t-plane, then the same function expressed in terms of x, y in the z-plane will be a solution of Laplace's equation in the transformed region in the z-plane.

(Binns, Lawrenson & Trowbridge 1995, pp.117-122)
(Sokolnikoff & Redheffer 1958, pp.576-586)
(Jeffreys & Swirles 1966, pp.409-411)

2.2.1 Example of Conformal Transformation

Let us have an example to illustrate how conformal transformation can be applied to solve the flux fringing at the end of a parallel plate capacitor.

Consider first the complex plane t where there is a uniform electric field. The lines u = constant are flux lines, while the lines v = constant are equipotential lines. This uniform field is the simplest Laplacian field to which all other solutions are related.

The following conformal transformation will transform the uniform field to flux fringing at the end of the capacitor.

$z = t + e^t$ (This transformation is stated without proof here. Proof is given in *The Analytical and Numerical Solution of Electric and Magnetic Fields*, but it is beyond the scope of this dissertation. The proof presented in the Book, and also the equation log z = log t + t log e = log t + t useful.)

Therefore $x + jy = (u + jv) + e^{u+jv}$

$= (u + jv) + (e^u)(e^{jv})$

$= (u + jv) + (e^u)(\cos v + j\sin v)$

$= (u + e^u \cos v) + j(v + e^u \sin v)$

Equating the real and imaginary parts,

$$x = u + e^{u} \cos v$$

$$y = v + e^{u} \sin v$$

A plot of x, y gives the fringings.

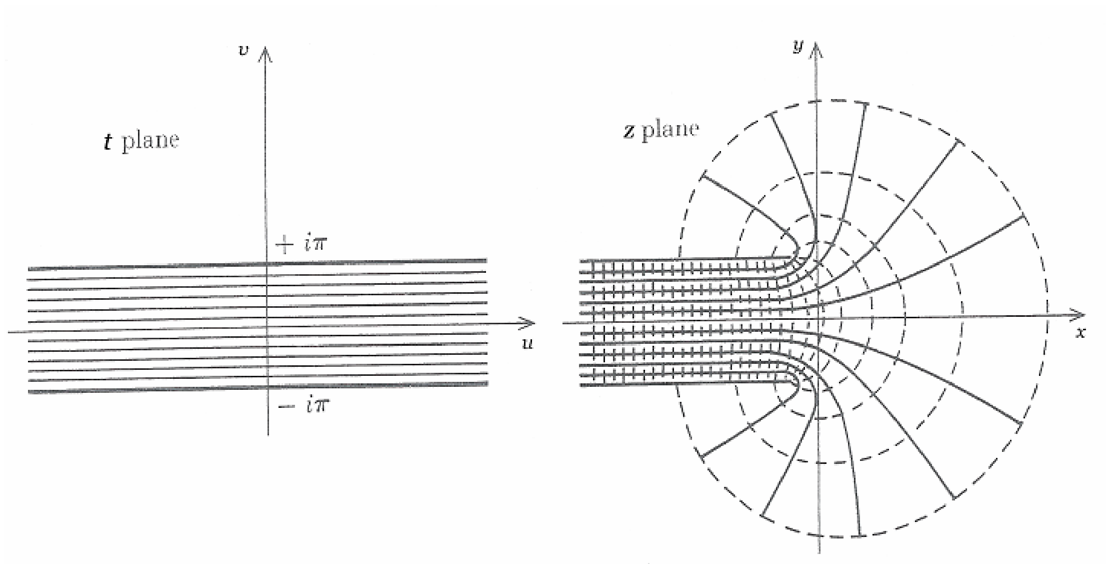


Figure 2.2 Conformal Transformation from a uniform field to fringing of capacitor

(Sokolnikoff & Redheffer 1958, p.586; Binns, Lawrenson & Trowbridge 1995, pp.155-157)

2.3 Separation of Variables

Unlike conformal transformation, which can only be found in textbooks of advanced level, separation of variables can be found in almost any textbooks on electromagnetics.

To obtain the analytical solution of Laplace’s equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 1,$$

let u to be the product of two functions,

$$\mathbf{u(x,y) = X(x)\ Y(y)}$$

Hence $\frac{1}{X}\frac{d^2X}{dx^2}+\frac{1}{Y}\frac{d^2Y}{dy^2}=0$

Putting

$$\frac{1}{X}\frac{d^2X}{dx^2}=-\varpi^2$$

and

$$\frac{1}{Y}\frac{d^2Y}{dy^2}=\varpi^2$$

where ϖ is any real number,

$$X = A\cos\varpi x + B\sin\varpi x,$$
$$Y = C\cosh\varpi y + D\sinh\varpi y,$$

A, B, C and D being arbitrary constants.

The solution is

$$\begin{aligned} u(x,y) &= XY \\ &= (A\cos\varpi x + B\sin\varpi x)(C\cosh\varpi y + D\sinh\varpi y) \end{aligned}$$

(Rao 2004, pp.726-728; Binns, Lawrenson & Trowbridge 1995, pp.57-58)

2.4 Numerical Dispersion and Stability

This is a summary of Chapter 4, of *Computational Electrodynamics* by A. Taflove and S.C. Hagness.

In the FDTD algorithm, numerical dispersion and stability are the two main factors that affect the choice of time step, Δt and lattice space increments, Δx, Δy and Δz.

In a uniform one-dimensional grid, numerical dispersion is reduced to ideal dispersion, when the magic time-step is used. That is, the Courant factor S, is 1. Under this condition, the solution of a continuous one-dimensional wave equation, given by the Yee algorithm, is exact.

For two-or three-dimensional Yee space lattice, the direction of wave propagation affects the numerical dispersion. The space lattice is an anisotropic medium because the phase velocity is different at different direction of wave propagation.

Along the diagonal of a two-dimensional square grid, if S is $1/\sqrt{2}$, numerical dispersion reduces to ideal dispersion.

Similarly, along the diagonal of a three-dimensional cube grid, if S is $1/\sqrt{3}$, numerical dispersion reduces to ideal dispersion.

The numerical dispersion error can be quantified by the physical phase-velocity error and the velocity-anisotropy error.

Physical phase-velocity error measures the amount of phase lead or lag. Velocity-anisotropy error measures the wavefront distortion due to the anisotropy of the space lattice. Physical phase-velocity error can be reduced by proper choice of grid discretization.

Stability problem becomes complex when the Yee algorithm is used with :

1. boundary conditions;
2. variable and unstructured meshing; and
3. lossy, dispersive, nonlinear, and gain materials.

In this book, four strategies are suggested to reduce the effect of numerical dispersion in the Yee algorithm.

These strategies are:

1. to create a specific numerical phase-velocity curve about c;
2. to use fourth-order-accurate spatial differences;
3. to use hexagonal grids; and
4. to use Discrete Fourier Transforms to calculate the spatial derivatives.

Short descriptions on these four strategies are given below.

§ **Centre a specific numerical phase-velocity curve about c**

The physical phase-velocity can be reduced by shifting the phase-velocity curve so that it is centred about c, the speed of light in free-space. However, the velocity-anisotropy remains unchanged.

§ **Fourth-order accurate spatial differences**

The phase-velocity anisotropy error can be reduced by using fourth-order-accurate finite difference scheme in the Yee algorithm. Also, under this situation, the computer storage is greatly reduced. The only drawback is large stencil needed to calculate fourth-order spatial differences when dealing with material interfaces.

§ **Use Hexagonal grids**

Hexagonal grids exhibit a fourth-order dependence, in spite of the second-order accuracy of spatial difference being used.

Difficulties arise when hexagonal grids are used in three dimensions. An automated computer-based mesh generation is needed.

Hexagonal grid, in comparison with fourth-order-accurate spatial differences, has lower numerical dispersion. Hence, less computer storage is required.

§ **Use Discrete Fourier Transform to calculate the spatial derivatives**

Pseudo-spectral time-domain (PSTD) method is used on unstaggered, collocated Cartesian space lattices. The velocity-anisotropy error is reduced to zero. However, PSTD can only be used in structures comprised entirely of dielectrics.

2.5 Incident Wave Source Conditions

This is a summary of Chapter 5, of *Computational Electrodynamics*, by A. Taflove and S.C. Hagness.
In this chapter, different electromagnetic wave sources are modelled.

Four kinds of electromagnetic wave sources are discussed. They are:

- hard-sourced **E** and **H** fields in one-and two-dimensional grids;
- **J** and **M** sources in three-dimensional lattices;
- total-field/scattered-field formulation for plane-wave excitation in one, two, and three dimensions; and
- waveguide sources.

2.5.1 Hard sources in one dimension

A hard source can be set up by using an analytical time-domain function at specific components of **E** and **H** in the FDTD space lattice.

For a pulsed hard source or continuous sinusoidal wave, established at grid-point $\mathbf{i_s}$, the reflected wave from a material structure will eventually return to $\mathbf{i_s}$.

At $\mathbf{i_s}$ the total tangential **E**-field is specified without considering any possible reflected waves in the grid. The hard source generates purious, nonphysical/retroreflection waves at $\mathbf{i_s}$**,** back to the material structure.

This problem can be solved by turning off the hard source before any reflected waves reach the hard source. However, this will limit the maximum number of time steps in FDTD simulation.

2.5.2 Hard sources in two dimensions

After careful analysis, it is found that a "pointwise" hard source in two dimensions is not a point. The hard source has a finite effective radius of approximately 0.2 grid cell in two-dimensional FDTD models, when the grid cell size, Δ is much less than $\frac{\lambda_0}{10}$.

2.5.3 J and M current sources in three dimensions

Yee space lattice is divergence-free in free-space regions when there is no **J** and **M** sources. However, current sources having specific geometry and temporal property can deposit charges in the Yee lattice.

Since Yee lattice can deposit electric and magnetic charges, it is possible to calculate the capacitance and inductance between cells of the space lattice.

After some calculation, the intrinsic lattice capacitance and inductance between adjacent FDTD space lattice in free space are found to be $3\,\varepsilon_0\Delta$ and $\frac{\mu_0\Delta}{4}$ respectively, where Δ is the length of a cubic-cell.

When a lumped-element capacitor or a lumped-element inductor is added at a specific point within the FDTD space lattice, the total capacitance or inductance at that point is the sum of the intrinsic lattice capacitance or inductance plus the lumped-element capacitance or inductance.

2.5.4 The total-field/scattered-field technique

The total-field/scattered-field (TF/SF) attempt to model a plane-wave source by using the linearity of Maxwell's equations.

The total electric and magnetic fields $\mathbf{E}_{\text{total}}$ and $\mathbf{H}_{\text{total}}$ are assumed to be decomposed in the following manner:

$$\mathbf{E}_{\textbf{total}} = \mathbf{E}_{\textbf{inc}} + \mathbf{E}_{\textbf{scat}}$$
$$\mathbf{H}_{\text{total}} = \mathbf{H}_{\text{inc}} + \mathbf{H}_{\text{scat}}$$

$\mathbf{E}_{\text{inc}}$ and $\mathbf{H}_{\text{inc}}$ are the known incident-wave fields that exist in vacuum without any materials in the space lattice.

In TF/SF formulation, the FDTD space lattice is separated into two regions. Region 1 contains only the total fields and the interacting structure. Region 2 (surrounding region 1) contains only the scattered fields. The absorbing boundary condition is applied at the outer layer of Region 2. The interface of Region 1 and Region 2 contains **E** and **H** components.

When the Yee algorithm is applied across the interface, inconsistency arises because of the arithmetic difference between scattered- and total-field values. Correction can be made by using the fact that the total-field value is the sum of the incident-field and scattered-field values.

For two-dimensional and three-dimensional problems, the calculation of the incident field is simplified by a table look-up procedure.

2.5.5 Total-field/reflected-field model formulation for waveguide system

The method used in waveguide system is similar to that used in the TF/SF technique. Region 1 is the total-field region and it contains the interacting structures of interest. Region 2 contains only the reflected-field and the absorbing boundary condition. The consistency condition at the interface between Region 1 and Region 2 is corrected by using the similar procedures in TF/SF technique.

The total-field/reflected-field programming for a waveguide can be adapted from the TF/SF technique.

2.6 Soil Water Content Using TDR

The technology of Time-Domain Reflectometry (TDR) was applied in vertical transmission lines constructed by using parallel wires. The transmission lines were inserted vertically into soil. It was found that this method gave a good measure of the soil water content (Topp et al. 1982, p.678).
This paper inspired the third computer programming on TDR in this project

More details will be given in Chapter 7, Application of FDTD to TDR.

2.7 Recent Developments of FDTD

Descriptions of seven journal papers were given below.

i) The first paper, A New FDTD Algorithm Based on Alternating-Direction Implicit (ADI) Method, by Takefumi Namiki, was published in 1999. The ADI method removes the Courant-Friedrich-Levy (CFL) condition. Two simulation examples are given.

The first example is a 2D free-space simulation, with Mur's first-order ABC and a sine-wave (squared and then added to the source position, i.e. soft source). The results are given below.

For conventional FDTD:

delta t = 9.4 ps

number of time-steps = 5000

CPU time = 215.5s

computer memory = 429.8 Kb

For ADI:

delta t = 235.0 ps

number of time-steps = 200

CPU time = 27.5s

computer memory = 566.7 Kb

The ADI, after removing Courant condition, has larger delta t, so less time-steps and CPU time. It is faster.

The second example is a 2D parallel-plate waveguide model, which gives similar results.

ii) The second paper, Toward the Development of a Three-Dimensional Unconditionally Stable Finite-Difference Time-Domain Method, by Fenghua Zheng, Zhizhang Chen, and Jiazong Zhang, was published in 2000. This is a modification of ADI, and an extension from 2D to 3D.

iii) The third paper, the Concurrent Complementary Operators Method for FDTD Mesh Truncation, by Omar M. Ramahi, was published in 1998. This paper introduced a new absorbing boundary condition.

iv) The fourth paper, Numerical Dispersion Analysis of the Unconditionally Stable 3-D ADI-FDTD Method, by Fenghua Zheng and Zhizhang Chen, was published in 2001. This is related to the second paper. It continues to discuss the dispersion of the 3-D ADI introduced in the second paper.

v) The fifth paper, FDTD Dispersion Revisited: Faster-Than-Light Propagation, by John B. Schneider and Christopher L. Wagner was published in 1999. It discusses the dispersion relation for the second-order FDTD, first derived by Taflove.

vi) The sixth paper, A Conformal FDTD Software Package Modeling Antennas and Microstrip Circuit Components, was written by Wenhua Yu and Raj Mittra.

Conventional FDTD proposed by K.S. Yee uses cubical cells in three-dimensions and square cells in two-dimensions. Consequently the errors in modeling perfect electric conductors (PECs) with curved surfaces will be significant. Reducing the size of the cells to better fit the curve will increase computer timing.

Wenhua and Raj Mittra proposed Conformal FDTD to handle PECs with curved surfaces and developed a software package on Conformal FDTD

vii) The seventh paper, An Effective PML for the Absorption of Evanescent Waves in Waveguides, by Jean-Pierre Berenger, was published in 1998.

Berenger has published his PML in 1994, and in 1998 he proposed another PML, called PML-D

The abstract of this paper states that, as emphasized by several authors in literature, the evanescent modes are not absorbed by usual PML terminating waveguiding structures. A new version of PML, denoted as PML-D, is presented in this paper, which substantially absorbs such waves.

2.8 Recent Trends on Development of FDTD

Based on the journal papers, the recent trends are:

- reducing CPU time;
- searching for new ABCs;
- re-examining dispersion;
- modification of conventional FDTD to handle PECs with curved surfaces;
- development of software packages on FDTD

More and more engineers and scientists are interested in FDTD and consequently, more researches on FDTD will be carried out. This claim is supported by the observed fact, that hundreds of research papers in this area are currently published worldwide each year. In 1985, fewer than ten papers were published (Taflove & Hagness 2000, p.4).

Chapter 3: Review of Available FDTD Software Package

3.1 Software development in FDTD

Since the introduction of FDTD approach nearly thirty years ago, FDTD had become a popular computer technique for solving electromagnetic field problems in many applications, ranging from the design of antennas to biophotonics.

FDTD was a relatively new method when it was compared to other computational methods, such as finite element method or the method of moments.

FDTD commercial packages appeared later than those in FEM due to the fact that satisfactory absorbing boundary was not available until 1994.

Many companies and educational institutes offered FDTD software packages with free or limited usage. It was possible to convert some of the free FDTD software with limited capability to a software with complete features after paying the license fee. The license fee can be a yearly license or a one-time license.

Over the last decade, the development of personal computer had a great progress, therefore the speed of computation, the accuracy of FDTD solution and the electrical size of the electromagnetic problem increased. The speed of a personal computer increased from 133MHz to 2.93GHz. The number of bits that a computer could handle rose from 16 bits to 32 bits. The random access memory (RAM) increased from 8MB to 8GB.

The high processor speed was able to provide faster computation, which helped to reduce time in designing microwave products. This in turn helped to reduce the cost of production.

The increase in number of bits that a computer could handle helped to reduce the error induced by truncation. Thus the accuracy of the FDTD simulation results had been improved.

The electrical size of an electromagnetic problem depended on the size of RAM. With the increase in the size of RAM, more unknown FDTD variables could be handled. The size of a grid cell could be made smaller because of more RAM available.

Besides the new development in personal computers, dedicated hardware for FDTD was available to improve the performance of FDTD. However, dedicated hardware was expensive.

An alternative way to solve general-purpose FDTD problem was to use the graphic card for FDTD computation. High performance graphic card was much cheaper than dedicated FDTD hardware. The only problem for this approach was that pre and pro processing were needed in transforming between FDTD data and image data.

3.2 Review on FDTD software packages

The review was given below after a survey on available FDTD software packages.

- **EZ-FDTD software package**

EZ-FDTD needs at least Windows 95 operating system for installation. The permanent single-seat license is $5795 USD and yearly support and software upgrade is $595 USD. The trial version is valid for 30 days only. This package has a user friendly graphical interface for constructing simple FDTD model. EZ-FDTD is able to generate error messages for illogical FDTD problems. It also provides far-field monitor points and movies for E fields and H fields. However, EZ-FDTD does not provide standard structure, like waveguide and antenna. (http://www.ems-plus.com)

- **Fidelity- FDTD software module**

Fidelity is a software module in Zealand application package. It is a software module using FDTD to solve problems. The evaluation version of this software is network-based software. It can only be used online. It is advisable to have internet connection speed of more than 56Kbps if the evaluation version is used. Fidielty provides some standard structure like waveguide, antenna, coaxial port and microstrip port. Fidielty also provides point source and plane wave source. Fidielty can have far and short field view. (http://www.zeland.com)

- **Empire-FDTD software package**

Empire software package can be run on Windows and Linux platform. Empire software also supports 64-bit operation. 64MB of RAM and 400MB of hard disk space are needed for installation. For optimum operation, an Intel Pentium IV processor with chipset, 1 GB DDR RAM at 400MHz and a graphic card with 128MB are recommended. Near and far field plots are available. AutoCAD import and export of design is possible. (http://www.empire.de)

- **Concerto-FDTD software module**

Concerto is a software application package with a FDTD module. Concerto can only be run on Windows operating system. The FDTD module incorporates conforming mesh technology to improve the accuracy. Concerto can perform 3D design of RF and microwave devices. (http://www.vectorfields.com)

- **XFDTD-software package**

XFDTD can be run on multiple processor computers and it is able to have parallel computation. XFDTD requires Windows 2000/XP for 32-bit operating systems. It needs minimum 256 MB RAM. However, 512 MB (or more) RAM is recommended. 16 bit color OpenGL compliant video card (Matrox cards not supported, ATI or NVidia cards recommended) with 1024x768 display resolution is needed. A hard disk of 500 MB space is needed for full installation. XFDTD provides high fidelity human meshes for biological electromagnetic simulations. The meshes can be stretched and rotated. CAD design can be import for FDTD simulation. Utilizing the ability of the GPU (Graphics Processing Unit) in modern computer graphics cards to stream floating point calculations, XFDTD achieves extremely fast calculation speeds. (http://www.remcom.com)

3.3 Conclusion

In general, most of the FDTD software packages can only be run on Windows operating system environments, and they require at least 64 MB of RAM for FDTD computation. In some cases, free or evaluation version of FDTD software are provided. AutoCAD design can be imported into some FDTD software for calculation. All FDTD software can handle 2D and 3D electromagnetic problems. Most of the software packages provide “movie” of field values changing as a function of time. This feature can be served as an educational tool. In particular, XFDTD can be operated on multiple processor computers with parallel computation capability. In XFDTD, human meshes are provided for biological electromagnetic simulations. Graphic card is being used as hardware acceleration for FDTD calculation in XFDTD.

Chapter 4: Theory of FDTD

4.1 Yee's Algorithm

In 1966, K.S. Yee published his classic research paper "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media." (Yee 1966).

Yee's paper introduced a set of finite-difference equations for the time-dependent Maxwell's curl equations. Using the two Maxwell's curl equations, Yee solved both electric and magnetic fields in time and space, in a leapfrog manner. Yee also introduced the Yee's lattice, which is shown on next page Figure 4.1.

Sadiku (2003, pp. 160-163) presented the following derivation.
Maxwell's equations in an isotropic medium are as below

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} \tag{4.1a}$$

$$\nabla \times \mathbf{H} = \sigma \mathbf{E} + \epsilon \frac{\partial \mathbf{E}}{\partial t} \tag{4.1b}$$

The six scalar equations, expressed in rectangular coordinate system are:

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right), \tag{4.2a}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right), \tag{4.2b}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right), \tag{4.2c}$$

$$\frac{\partial E_x}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - \sigma E_x\right), \tag{4.2d}$$

$$\frac{\partial E_y}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} - \sigma E_y\right), \tag{4.2e}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \sigma E_z\right) \tag{4.2f}$$

A grid point in the solution region is defined as

$$(i, j, k) \equiv (i\Delta x, j\Delta y, k\Delta z) \tag{4.3}$$

and functions of space and time are defined as

$$F^n(i, j, k) \equiv F(i\delta, j\delta, k\delta, n\Delta t) \tag{4.4}$$

where $\delta = \Delta x = \Delta y = \Delta z$ is the space increment; Δt is the time increment; *i, j, k,* and *n* are integers. Using central finite difference approximation for space and time derivatives, it becomes:

$$\frac{\partial F^n(i, j, k)}{\partial x} = \frac{F^n(i + 1/2, j, k) - F^n(i - 1/2, j, k)}{\delta} + O\left(\delta^2\right) \tag{4.5}$$

$$\frac{\partial F^n(i, j, k)}{\partial t} = \frac{F^{n+1/2}(i, j, k) - F^{n-1/2}(i, j, k)}{\Delta t} + O\left(\Delta t^2\right) \tag{4.6}$$

Yee positioned the components of **E** and **H** about a unit cell of the lattice as shown in Figure 4.1. Every **E** components is surrounded by four **H** components, and every **H** component is surrounded by four **E** components.

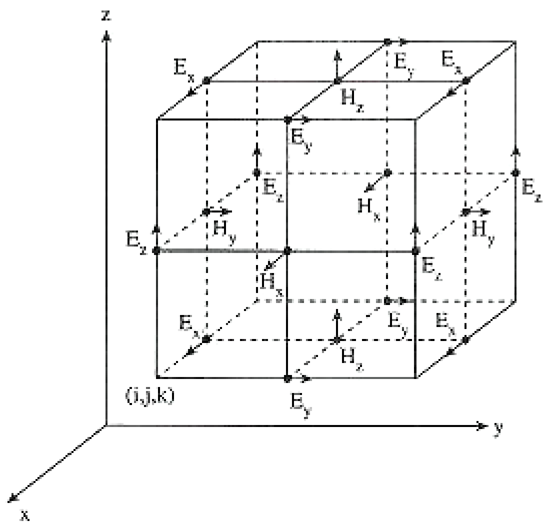


Figure 4.1 Positions of the field components in a unit cell of the Yee's lattice

The **E** and **H** components are evaluated at leapfrog steps. Hence the explicit finite difference approximations are:

$$\boxed{\begin{aligned} H_x^{n+1/2}\ &(i, j+1/2, k+1/2) = H_x^{n-1/2}(i, j+1/2, k+1/2) \\ &+\frac{\delta t}{\mu(i, j+1/2, k+1/2)\delta}\Big[E_y^n(i, j+1/2, k+1) \\ &-E_y^n(i, j+1/2, k) \\ &+E_z^n(i, j, k+1/2) - E_z^n(i, j+1, k+1/2)\Big], \end{aligned}} \tag{4.7a}$$

$$\boxed{\begin{aligned} H_y^{n+1/2}\ &(i+1/2, j, k+1/2) = H_y^{n-1/2}(i+1/2, j, k+1/2) \\ &+\frac{\delta t}{\mu(i+1/2, j, k+1/2)\delta}\Big[E_z^n(i+1, j, k+1/2) \\ &-E_z^n(i, j, k+1/2) \\ &+E_x^n(i+1/2, j, k) - E_x^n(i+1/2, j, k+1)\Big], \end{aligned}} \tag{4.7b}$$

$$\boxed{\begin{aligned} H_z^{n+1/2}\ &(i+1/2, j+1/2, k) = H_z^{n-1/2}(i+1/2, j+1/2, k) \\ &+\frac{\delta t}{\mu(i+1/2, j+1/2, k)\delta}\Big[E_x^n(i+1/2, j+1, k) \\ &-E_x^n(i+1/2, j, k) \\ &+E_y^n(i, j+1/2, k) - E_y^n(i+1, j+1/2, k\Big]. \end{aligned}} \tag{4.7c}$$

$$\boxed{\begin{aligned} E_x^{n+1}\ &(i+1/2, j, k) = \left(1 - \frac{\sigma(i+1/2, j, k)\delta t}{\epsilon(i+1/2, j, k)}\right)\cdot \\ &\quad E_x^n(i+1/2, j, k) \\ &+\frac{\delta t}{\epsilon(i+1/2, j, k)\delta}\Big[H_z^{n+1/2}(i+1/2, j+1/2, k) \\ &-H_z^{n+1/2}(i+1/2, j-1/2, k) \\ &+H_y^{n+1/2}(i+1/2, j, k-1/2) \\ &-H_y^{n+1/2}(i+1/2, j, k+1/2)\Big], \end{aligned}} \tag{4.7d}$$

$$\begin{aligned} H_z^{n+1/2}&(i+1/2, j+1/2, k) = H_z^{n-1/2}(i+1/2, j+1/2, k) \\ &+ \frac{\delta t}{\mu(i+1/2, j+1/2, k)\delta}\Big[E_x^n(i+1/2, j+1, k) \\ &- E_x^n(i+1/2, j, k) \\ &+ E_y^n(i, j+1/2, k) - E_y^n(i+1, j+1/2, k\Big] \, . \end{aligned} \tag{4.7e}$$

$$\begin{aligned} E_x^{n+1}&(i+1/2, j, k) = \left(1 - \frac{\sigma(i+1/2, j, k)\delta t}{\epsilon(i+1/2, j, k)}\right) \cdot \\ &\quad E_x^n(i+1/2, j, k) \\ &+ \frac{\delta t}{\epsilon(i+1/2, j, k)\delta}\Big[H_z^{n+1/2}(i+1/2, j+1/2, k) \\ &- H_z^{n+1/2}(i+1/2, j-1/2, k) \\ &+ H_y^{n+1/2}(i+1/2, j, k-1/2) \\ &- H_y^{n+1/2}(i+1/2, j, k+1/2)\Big] \, , \end{aligned} \tag{4.7f}$$

Equations (4.1) to (4.7) are direct reproductions from Sadiku’s book.

4.2 Stability Condition

The following is based on the book *Time-Domain Computer Analysis of Nonlinear Hybrid Systems* by Wenquan Sui.

Sui (2001, pp.154-156) expressed that FDTD is an explicit finite-difference method. Therefore all unknowns at present time-step are evaluated from known values at previous time-step.

The time increment Δt in FDTD is limited by the stability requirement known as the Courant-Friedrichs-Lewy (CFL) condition (Taflove & Hagness, 2000).

For 3D FDTD in a uniform grid, the stability requirement is:

$$\Delta t \le \frac{1}{v\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}}}$$

where v is the propagation speed of electromagnetic wave inside the medium.

When all the cell sizes are the same,

$$\Delta x = \Delta y = \Delta z = \Delta$$

$$\Delta t \le \frac{\Delta}{v\sqrt{3}} = \frac{0.577\Delta}{v}$$

For 2D FDTD, with the same cell sizes, the stability requirement is

$$\Delta t \le \frac{\Delta}{v\sqrt{2}} = \frac{0.71\Delta}{v}$$

For 1D FDTD, with the same cell sizes, the stability requirement is:

$$\Delta t \le \frac{\Delta}{v}$$

A convenient time-step that satisfies the stability requirement for 1D, 2D and 3D is:

$$\Delta t = \frac{\Delta}{2v} = \frac{0.5\Delta}{v}$$

The equal sign "=" but not "$\le$" is used here

and $0.5 < 0.577 < 0.71 < 1$

Δt can be smaller, but cannot be bigger, than the stability requirement.

To obtain higher accuracy in solution, the grid size has to be smaller. Hence the time-step will be smaller.

The spatial sampling rate is required to be between 10 samples per wavelength to 20 samples per wavelength.

For a system excited by a sinusoidal source of 3GHz, the wavelength is 0.1 meter. If the sampling rate is 20, the cell size will be 0.1/20 = 0.5 cm.

A convenient time-step for 1D, 2D and 3D will be

$$\Delta t = \frac{0.1/20}{2\times 3\times 10^{8}} = 8.3\times 10^{-12}\,\text{second}$$

where $v = 3 \times 10^8$ m/s is the speed of light.

4.3 Absorbing Boundary Conditions (ABCs)

This section consists of three sub-sections, namely the classification of ABCs, Mur's first-order ABC, and the perfectly matched layer (PML).

4.3.1 Classification of ABCs

Sadiku in his book, *Numerical Techniques in Electromagnetics*, page164, explains the need to use ABCs.

Taflove in his book, Advances in Computational Electrodynamics, page 3, presents a concise classification of ABCs.

When applying FDTD to problems where the solution regions are unbounded, difficulty arises. Since no computers can store unlimited amount of data, it is required to somehow, limit our solution regions. ABC is the method to limit the solution regions and it creates the numerical illusion of an infinite space.

Most of the Absorbing Boundary Conditions (ABCs) can be grouped into two categories. One category is ABCs derived from differential equations. The other category is ABCs using material absorbers.

Mur's first-order and second-order absorbing boundary conditions are examples of ABCs derived from differential equations.

The Perfectly Matched Layer (PML), which uses a material absorber, first proposed by Berenger in 1994, is a breakthrough in extending FDTD capability to achieve high accuracy. After many years of applications, the PML has been shown to give better accuracy than other ABCs. Now, the PML has become a standard to which other ABCs are compared with.

4.3.2 Mur's first-order ABC

(a) Mur's first-order ABC by Common Sense Approach

Sullivan in his book, Electromagnetic Simulation Using the FDTD Method, page 4, presents what he said '***a common sense approach***'.

Suppose a wave is travelling at c, the speed of light, along the negative x direction, towards a boundary at the end of the computational domain.

If $\Delta t = \Delta x/c$, then it takes one time step for a wave front to cross Δx, one cell.

Therefore, a common sense approach is:

u at next time step = u at present time step

Written in symbols, it is

(u at time step n+1, at spatial step 0) = (u at time step n, at spatial step 1)

Note that the wave is travelling from spatial step 1 to spatial step 0, along the negative x direction.

That is $u^{n+1}(0) = u^n(1)$

It is easy to implement this ABC. Simply store the value $u^n(1)$, and then put it at $u^{n+1}(0)$.

(b) Mur's first-order ABC by Derivation

The following derivation is based on the book *Computational Methods for Electromagnetics* by A.F. Peterson, S.L. Ray and R. Mittra, page 511 to page 513.

The first order ABC developed by Mur is discussed here.

The one-dimensional wave equation is

$$\frac{\partial^2 u}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = 0$$

The wave equation can be factored by treating the differential operators algebraically.

$$\left(\frac{\partial u}{\partial x}-\frac{1}{c}\frac{\partial u}{\partial t}\right)\left(\frac{\partial u}{\partial x}+\frac{1}{c}\frac{\partial u}{\partial t}\right)=0$$

The two brackets can be equated to zero to give the one-way wave equation

$$\left(\frac{\partial u}{\partial x}-\frac{1}{c}\frac{\partial u}{\partial t}\right)=0 \tag{4.8}$$

$$\text{or } \left(\frac{\partial u}{\partial x}+\frac{1}{c}\frac{\partial u}{\partial t}\right)=0 \tag{4.9}$$

Solutions to Equation (4.8), u(x+ct) are –x travelling waves, while solutions to Equation (4.9), u(x-ct) are +x travelling waves. Thus, to simulate an open-region problem, Equations (4.8) and (4.9) are used as boundary conditions on the left and right boundaries respectively. The above Book then jumps to the result, Equation (4.11).

The elaborations to find the final result Equation (4.11) were given below.

Discretizing Equation (4.8), the first-order equation for negative x direction propagating waves, is:

$$\left(\frac{1}{\Delta x}\right)\left\{\left(\frac{1}{2}\right)\left[u^{n+1}(1)+u^{n}(1)\right]-\left(\frac{1}{2}\right)\left[u^{n+1}(0)+u^{n}(0)\right]\right\}$$

$$-\left(\frac{1}{c}\right)\left(\frac{1}{\Delta x}\right)\left\{\left(\frac{1}{2}\right)\left[u^{n+1}(1)+u^{n+1}(0)\right]-\left(\frac{1}{2}\right)\left[u^{n}(1)+u^{n}(0)\right]\right\}=0 \tag{4.10}$$

Multiple both sides by $2c(\Delta t)(\Delta x)$

$$c(\Delta t)u^{n+1}(1)+c(\Delta t)u^{n}(1)-c(\Delta t)u^{n+1}(0)$$

$$-c(\Delta t)u^{n}(0)-(\Delta x)u^{n+1}(1)-(\Delta x)u^{n+1}(0)+(\Delta x)u^{n}(1)+(\Delta x)u^{n}(0)=0$$

Grouping like terms together to give

$$c(\Delta t)u^{n}(1)+(\Delta x)u^{n}(1)+c(\Delta t)u^{n+1}(1)-(\Delta x)u^{n+1}(1)+(\Delta x)u^{n}(0)-c(\Delta t)u^{n}(0)$$
$$=c(\Delta t)u^{n+1}(0)+(\Delta x)u^{n+1}(0)$$

Factorizing to get

$$u^{n+1}(0)[c(\Delta t)+\Delta x]\ =u^{n}(1)[c(\Delta t)+\Delta x]+u^{n+1}(1)[c(\Delta t)-\Delta x]-u^{n}(0)[c(\Delta t)-\Delta x]$$

$$u^{n+1}(0)[c(\Delta t)+\Delta x]=u^{n}(1)[c(\Delta t)+\Delta x]+[c(\Delta t)-\Delta x]\ \left[u^{n+1}(1)-u^{n}(0)\right]$$

$$u^{n+1}(0)=u^{n}(1)+\left\{[c(\Delta t)-\Delta x]/[c(\Delta t)+\Delta x]\right\}*\ \left[u^{n+1}(1)-u^{n}(0)\right] \qquad (4.11)$$

If $c(\Delta t)-\Delta x=0$, the second term on the right hand side of Equation (4.11) will vanish.

Under the condition $\Delta t=\Delta x/c$

i.e. the magic step,

$$u^{n+1}(0)=u^{n}(1)$$

which is the same as the result obtained by common sense approach.

4.3.3 The Perfectly Matched Layer (PML)

PML is realized by surrounding the computational domain with a lossy material that dampens the outgoing fields.

With properly designed parameters of the PML, incident waves, with any incident angle, will penetrate into the PML without reflection, and are rapidly attenuated inside the PML. The reflection coefficient is zero at boundaries if the parameters of the PML have the following relation:

$$\frac{\sigma_e}{\varepsilon}=\frac{\sigma_m}{\mu}$$

where σ_e is the electric conductivity,

ε is the permittivity,

σ_m is the magnetic conductivity, and

μ is the permeability.

The outer boundary of the PML has a perfect electric conductor (PEC) wall, which will reflect waves back into the PML.

When the above relation is satisfied, the 2D TE mode field inside the PML is found to be

$$F = F_o\, pqr$$

where F_o is the initial field

$$p \text{ is } \exp\left[jwt - \left(\frac{jw}{c}\right)(x\cos\phi + y\sin\phi) \right]$$

$$q \text{ is } \exp\left[\left(-\frac{x}{c}\right)\left(\frac{\sigma_x}{\varepsilon}\right)\cos\phi \right]$$

$$r \text{ is } \exp\left[\left(-\frac{y}{c}\right)\left(\frac{\sigma_y}{\varepsilon}\right)\sin\phi \right]$$

and $$Z = \left(\frac{\mu}{\varepsilon}\right)^{\frac{1}{2}}$$

The field inside the PML decreases exponentially. Since the impedance of the PML, Z, is the same as the impedance just outside the PML, incident waves, at any angle, will transmit into the PML without reflection (Sui 2001, pp.162-163).

Summarizing, plane waves of arbitrary incidence, polarization and frequency are matched and consequently absorbed at the boundary of PML without reflections.

4.4 Determination of Sampling Rate

This section is based on the book, *Finite Difference Time Domain Method for Electromagnetics*, by K.S. Kunz and R. J. Luebbers.

The sampling rate is usually between 10 samples per wavelength to 20 samples per wavelength. The source, such as Gaussian pulse, may contain many wavelengths. The wavelength in sampling rate should be the shortest wavelength in the source.

For accurate determination, such as of radar scattering cross-section, 20 samples per wavelength are necessary. For other cases, reasonable results may be obtained by 10 samples per wavelength.

The paragraphs below present a more detailed description on sampling rate.

According to Nyquist sampling theorem, there must be at least two samples per wavelength in order for the spatial information to be obtained.

In addition to Nyquist theorem, the grid dispersion error has to be considered. In FDTD, waves of different frequencies propagate at slightly different speeds through the grid. The propagation speed also depends on the direction of propagation relative to the grid. A sampling rate of 10 to 20 samples per wavelength, will give accurate results, under dispersion error, and satisfy Nyquist sampling theorem.

Another consideration is the geometry of the object to be modeled. The “staircase” errors of modeling a curved surface with square cells, may be significant. 20 samples per wavelength may give reasonable results in these situations.

Chapter 5: Application of FDTD to One-Dimensional ABC

5.1 Introduction

This program serves to illustrate the necessity of ABC when using FDTD. The 1D computer program is a suitable learning process for undergraduates, who have no idea of what FDTD is. It is a good introduction to FDTD, and has been useful as a training exercise during this project.

The fdtd1D computer program written by Dr. Hagness (2000), is a direct implementation of the FDTD equations. Therefore, the source codes are easy to understand. In this program, Mur's 1st order absorbing boundary is used.

5.2 Modification

Four modifications have been made to the program fdtd1D.

(a) Renaming of variables

Variables are given names that tell what they are. For examples,

cc is changed to light_velocity_freespace

epz is changed to epsilon

eps is changed to epsilon_r (i.e. relative epsilon)

After renaming, the program is easier to read.

(b) Rewriting sections on Movie Initialization and Visualize Fields

The fdtd1D program is written in MATLAB 5, yet it still can be run in MATLAB 7. After studying the documentation in MATLAB 7, it is possible to remove some of the obsolete codes to improve the efficiency of the program.

(c) Adding different sources

Gaussian pulse and half-sine pulse are not supplied in MATLAB. User defined functions for Gaussian pulse and half-sine pulse are created.

(d) Adding a Graphical User Interface (GUI)

The adding of the GUI makes the program fdtd1D very long. The GUI allows users to key in the electric conductivity and the duration of the run time. The program is carefully debugged. It displays appropriate error messages. Users can save the demonstration (i.e. the running result of part of the program) in files, which can be played in MATLAB or other video player.

5.3 User Guide

The screen shows a main menu after the running of the program. There are five push-buttons in the main menu.

5.3.1 Main Menu

Clicking the first push-button leads to an introduction to FDTD, which tells briefly something about FDTD.

The second push-button gives a demonstration of 1-D FDTD using a sine wave of 1GHz.

The third push-button gives a demonstration of 1-D FDTD using a half-sine pulse.

The fourth push-button gives a demonstration of 1-D FDTD using a Gaussian pulse.

The last push-button closes the main menu screen and exits to MATLAB.

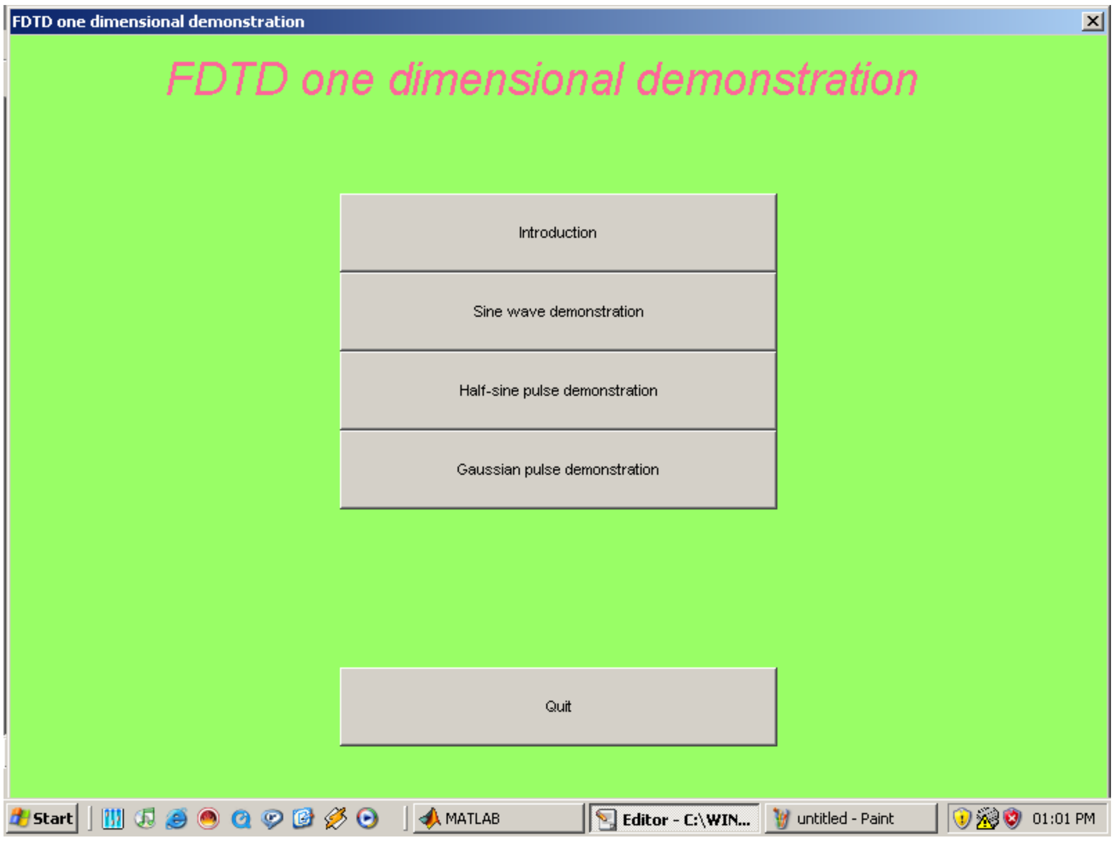
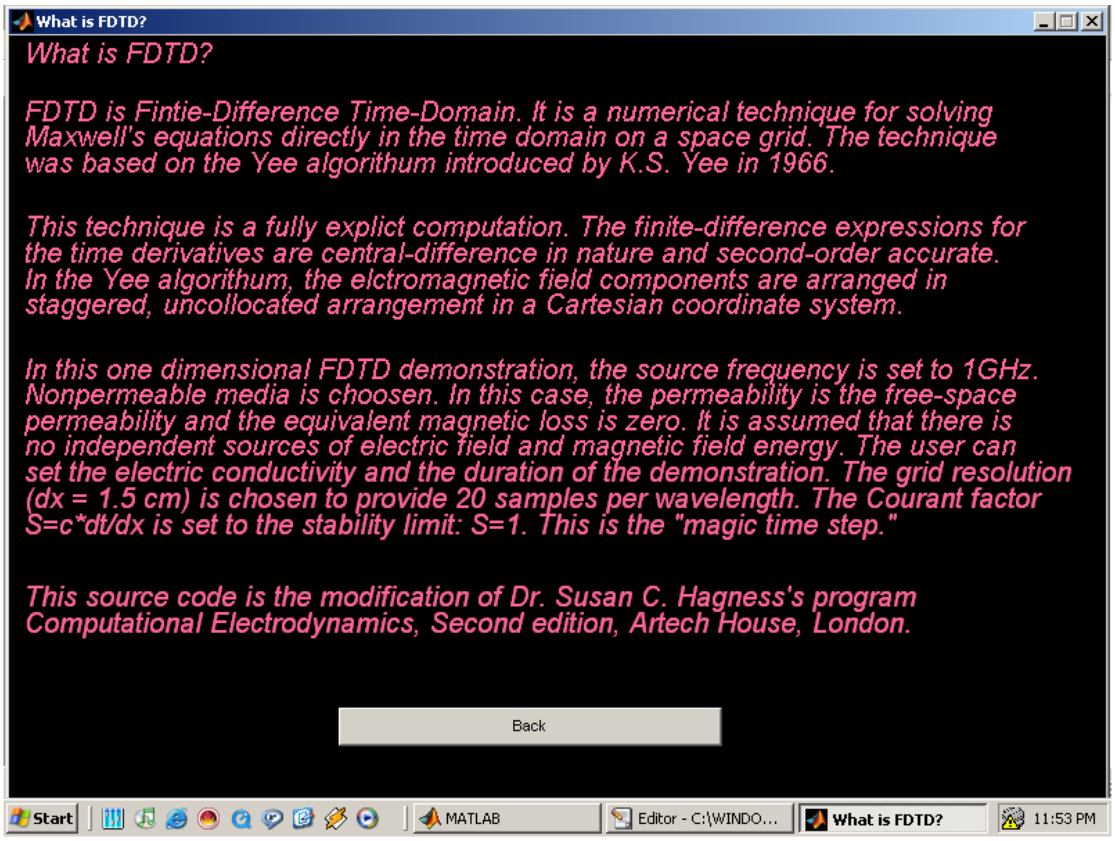

FDTD one dimensional demonstration
FDTD one dimensional demonstration
Introduction
Sine wave demonstration
Half-sine pulse demonstration
Gaussian pulse demonstration
Quit
Start
MATLAB
Editor - C:\WIN...
untitled - Paint
01:01 PM

Figure 5.1 Main screen

5.3.2 First push-button

After clicking the first push-button, the main menu will be closed and a new screen is shown. In the new screen, there is an introduction to FDTD. The user can press the '**Back**' button to return to the main menu.



What is FDTD?

FDTD is Fintie-Difference Time-Domain. It is a numerical technique for solving Maxwell's equations directly in the time domain on a space grid. The technique was based on the Yee algorithum introduced by K.S. Yee in 1966.

This technique is a fully explict computation. The finite-difference expressions for the time derivatives are central-difference in nature and second-order accurate. In the Yee algorithum, the elctromagnetic field components are arranged in staggered, uncollocated arrangement in a Cartesian coordinate system.

In this one dimensional FDTD demonstration, the source frequency is set to 1GHz. Nonpermeable media is choosen. In this case, the permeability is the free-space permeability and the equivalent magnetic loss is zero. It is assumed that there is no independent sources of electric field and magnetic field energy. The user can set the electric conductivity and the duration of the demonstration. The grid resolution (dx = 1.5 cm) is chosen to provide 20 samples per wavelength. The Courant factor S=c*dt/dx is set to the stability limit: S=1. This is the "magic time step."

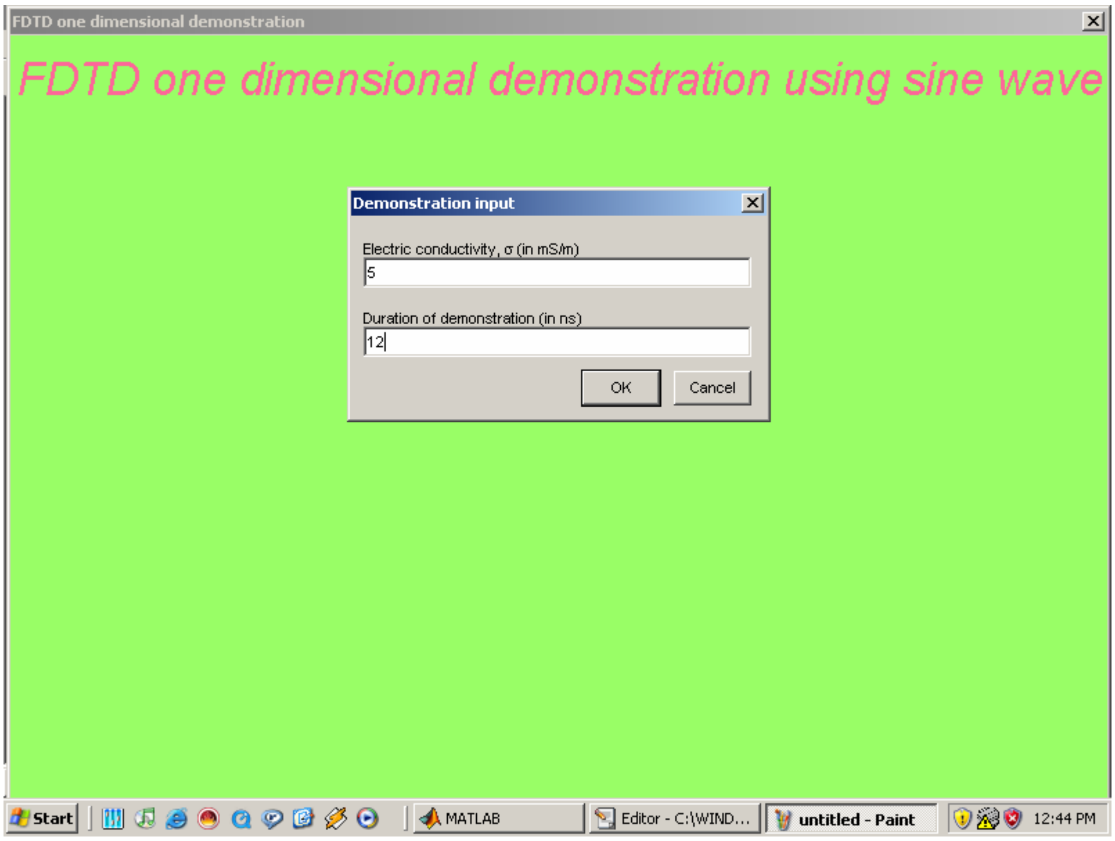
This source code is the modification of Dr. Susan C. Hagness's program Computational Electrodynamics, Second edition, Artech House, London.

Figure 5.2 **Flow chart of main menu**

5.3.3 Second push-button

The second push-button leads to a sub-menu for the demonstration in sine wave.

The user will be asked to key in the value of sigma and the end time of the demonstration. If the user clicks the '**Cancel**' button, the sub-menu will be closed and the main menu will be shown.



FDTD one dimensional demonstration

FDTD one dimensional demonstration using sine wave

Demonstration input

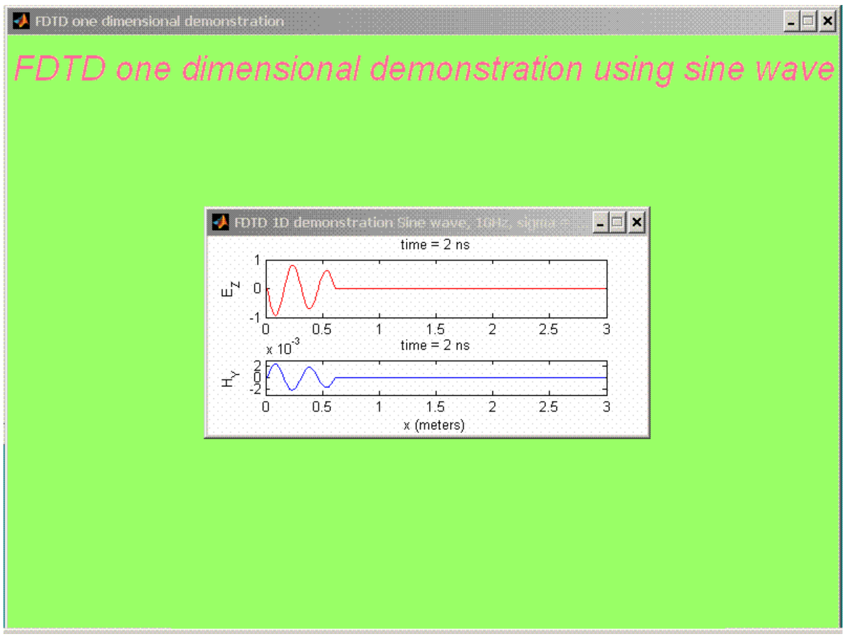
Electric conductivity, σ (in mS/m)

5

Duration of demonstration (in ns)

12

Figure 5.3 Sine wave menu

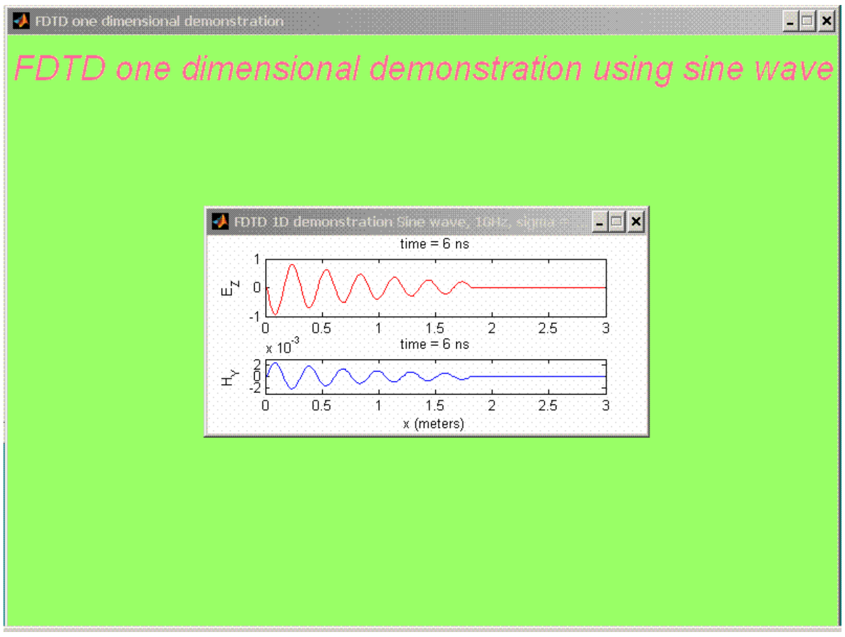


FDTD one dimensional demonstration

FDTD one dimensional demonstration using sine wave

FDTD 1D demonstration Sine wave, 1GHz, sigma =

(a)


FDTD one dimensional demonstration
FDTD one dimensional demonstration using sine wave
FDTD 1D demonstration Sine wave, 1GHz, sigma =
time = 6 ns
E_Z
1
0
-1
0
0.5
1
1.5
2
2.5
3
x 10^-3
time = 6 ns
H_Y
2
0
-2
x (meters)

(b)

Figure 5.4 Sample waveforms

The values of sigma and the end time of demonstration will be checked. If the two values are acceptable, movie will be played; otherwise error messages will displayed, and the user will be asked if he wishes to enter the values again.

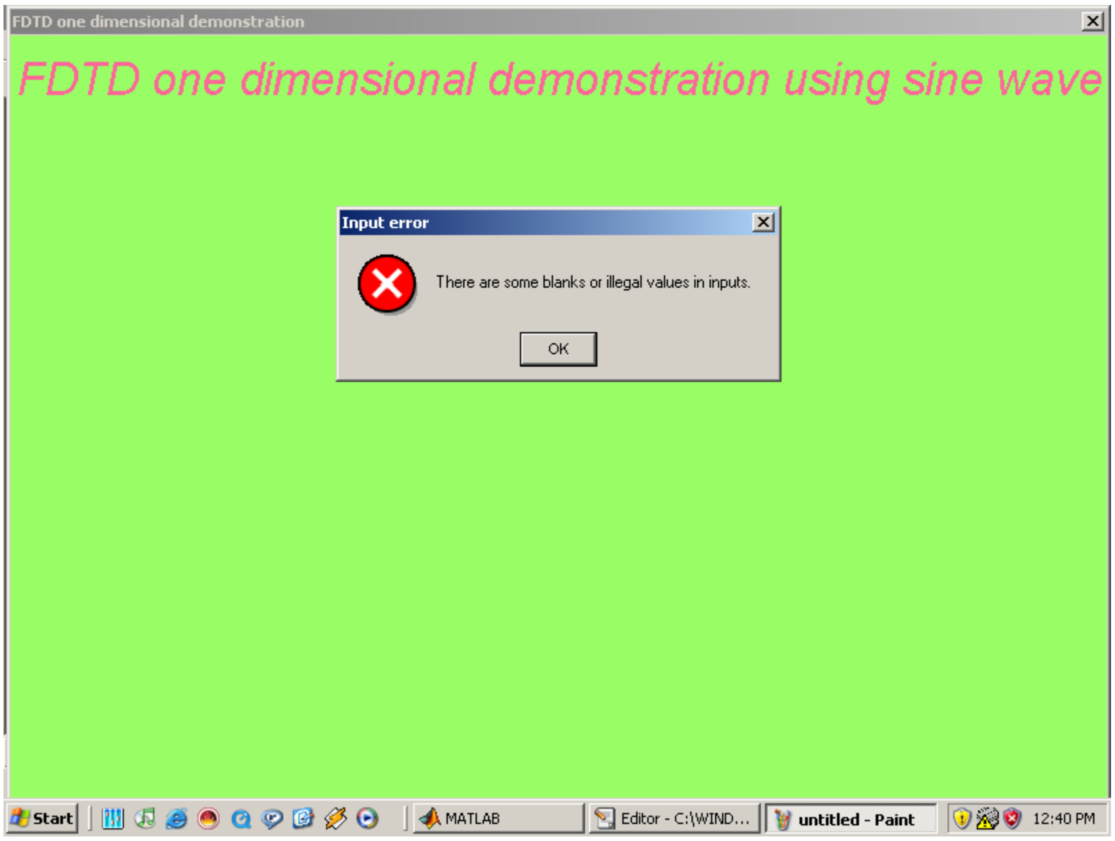

FDTD one dimensional demonstration
FDTD one dimensional demonstration using sine wave
Input error
There are some blanks or illegal values in inputs.
OK
Start
MATLAB
Editor - C:\WIND...
untitled - Paint
12:40 PM

Figure 5.5 Illegal inputs

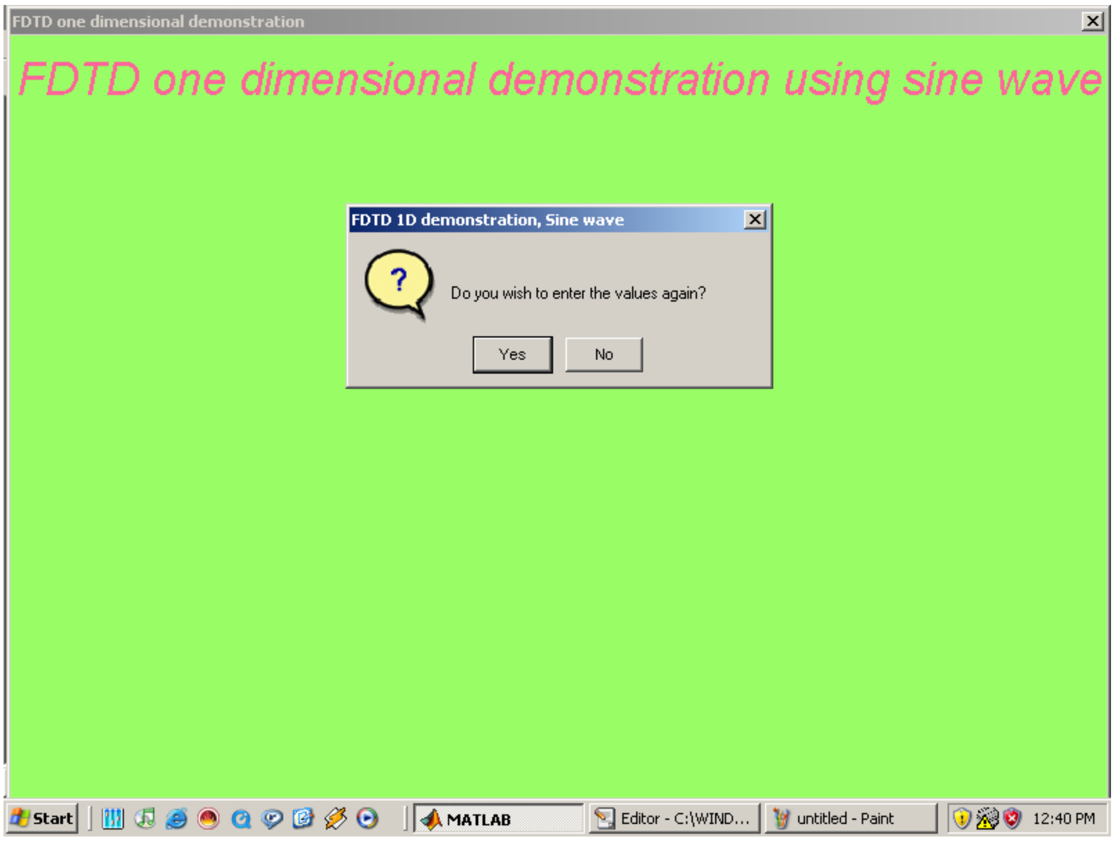
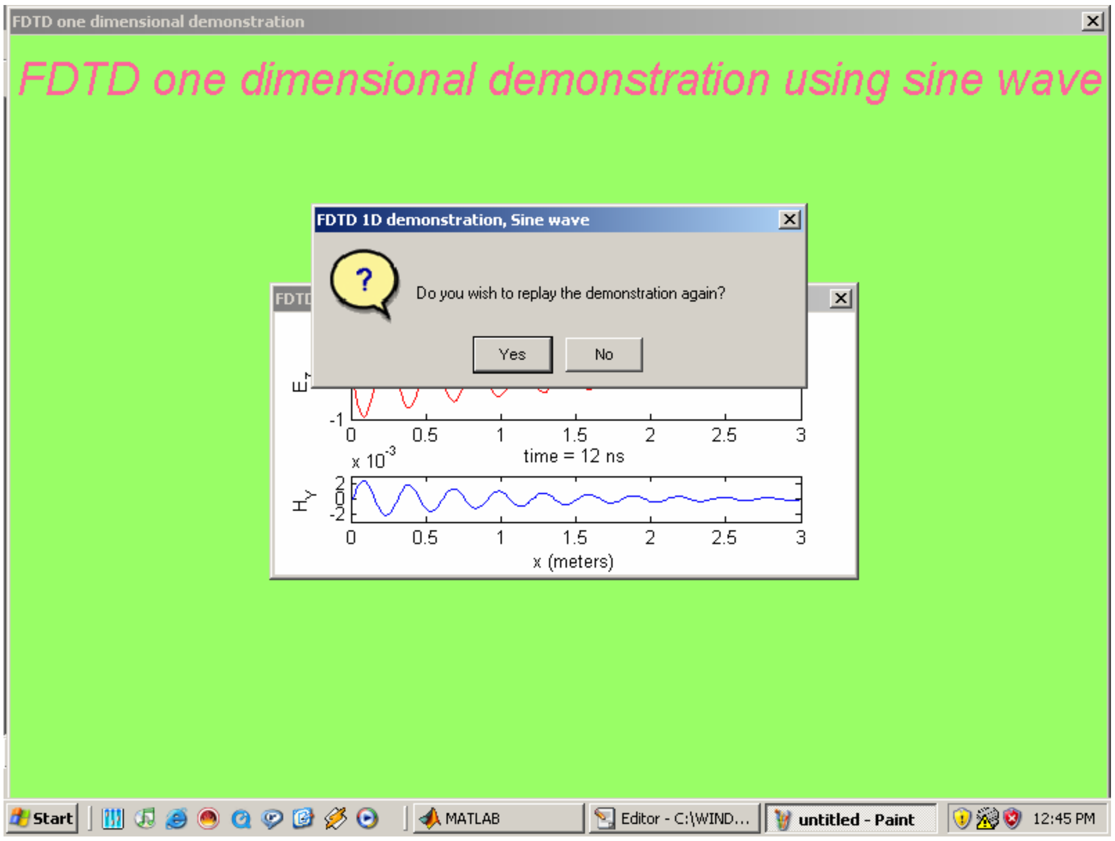

FDTD one dimensional demonstration
FDTD one dimensional demonstration using sine wave
FDTD 1D demonstration, Sine wave
Do you wish to enter the values again?
Yes
No
Start
MATLAB
Editor - C:\WIND...
untitled - Paint
12:40 PM

Figure 5.6 Request dialog

If the user does not wish to enter the values again, the sub-menu will be closed and the main menu will be displayed.

After the movie is played, the user will be asked if he wishes to play the movie again.


FDTD one dimensional demonstration
FDTD one dimensional demonstration using sine wave
FDTD 1D demonstration, Sine wave
Yes
No
time = 12 ns
x (meters)
Start
MATLAB
Editor - C:\WIND...
untitled - Paint
12:45 PM

Do you wish to replay the demonstration again?

Figure 5.7 Replay dialog

If the user does not wish to play the movie again, he will be asked if he wishes to save the movie. If he does not wish to save the movie, he will be asked if he wishes to leave the sub-menu and return to the main menu.

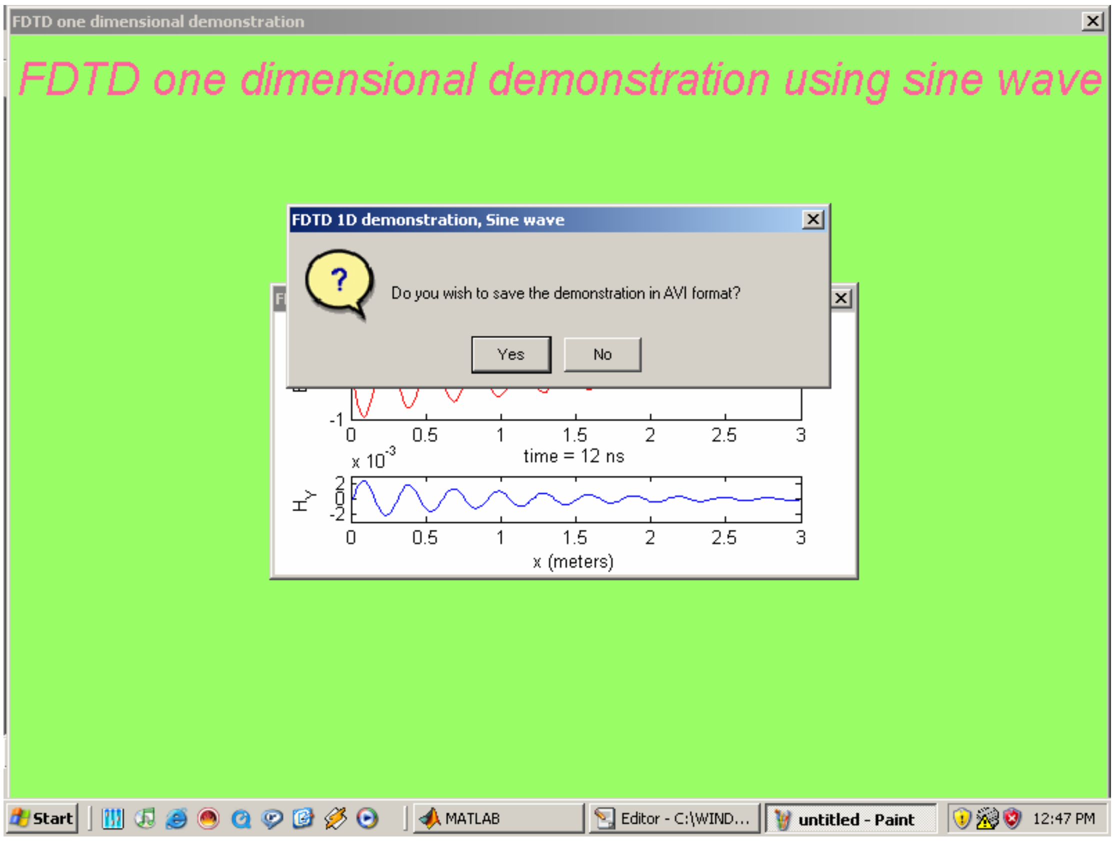

FDTD one dimensional demonstration
FDTD one dimensional demonstration using sine wave
FDTD 1D demonstration, Sine wave
Do you wish to save the demonstration in AVI format?
Yes
No
time = 12 ns
x (meters)
Start
MATLAB
Editor - C:\WIND...
untitled - Paint
12:47 PM

Figure 5.8 Prompt for saving demonstration

If he wishes to save the movie, he will be asked to enter a file name, If no file name is given, a default file name will be used.

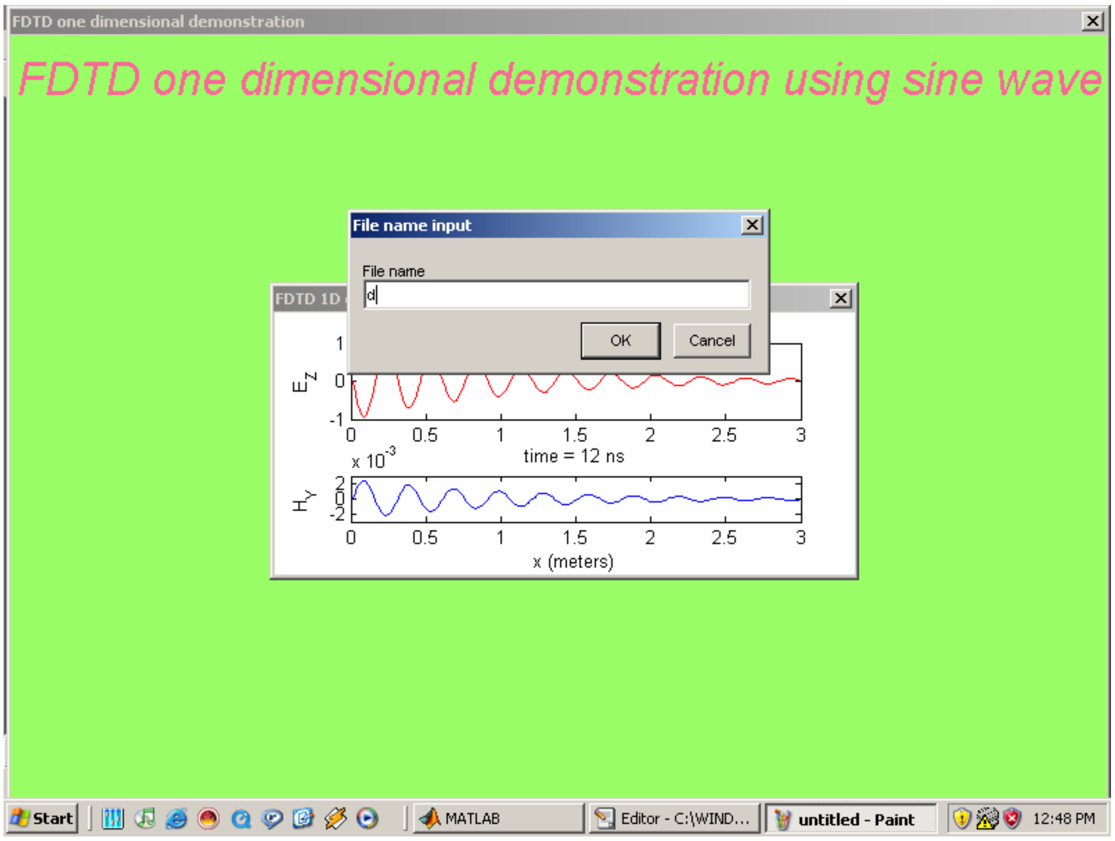
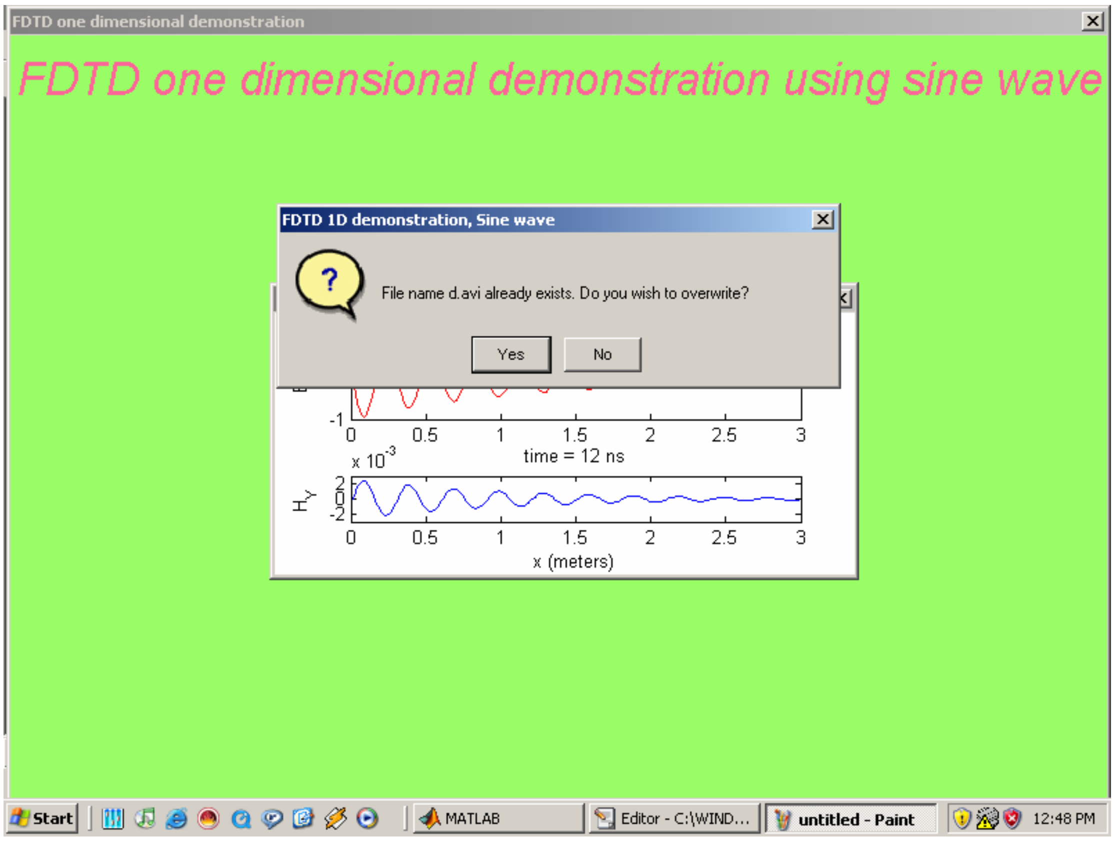


Figure 5.9 Dialog for entering file name

If the file name given by the user already exists, he will be asked if he wishes to enter a new file name, or to over write the existing file.


FDTD one dimensional demonstration
FDTD one dimensional demonstration using sine wave
FDTD 1D demonstration, Sine wave
Yes
No
-1
0 0.5 1 1.5 2 2.5 3
x 10^-3
H_y
2
0
-2
x (meters)
Start
MATLAB
Editor - C:\WIND...
untitled - Paint
12:48 PM

File name d.avi already exists. Do you wish to overwrite?

time = 12 ns

Figure 5.10 Request for overwriting existing file

After the file is saved, the user will be asked if he wishes to exit the sub-menu, or continue with another new demonstration.

5.3.4 Third push-button

The third push-button will show the sub-menu for the demonstration of half-sine pulse. The sub-menu works in the same way as in the sine-wave demonstration.

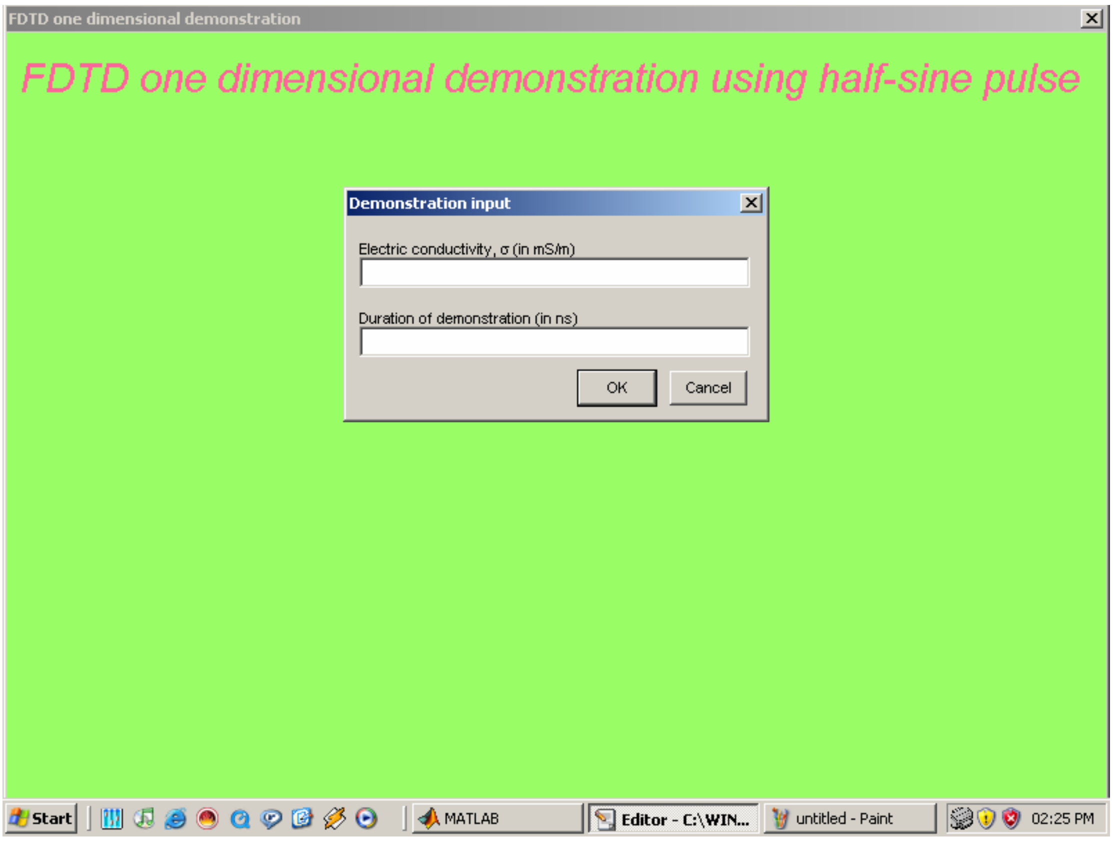
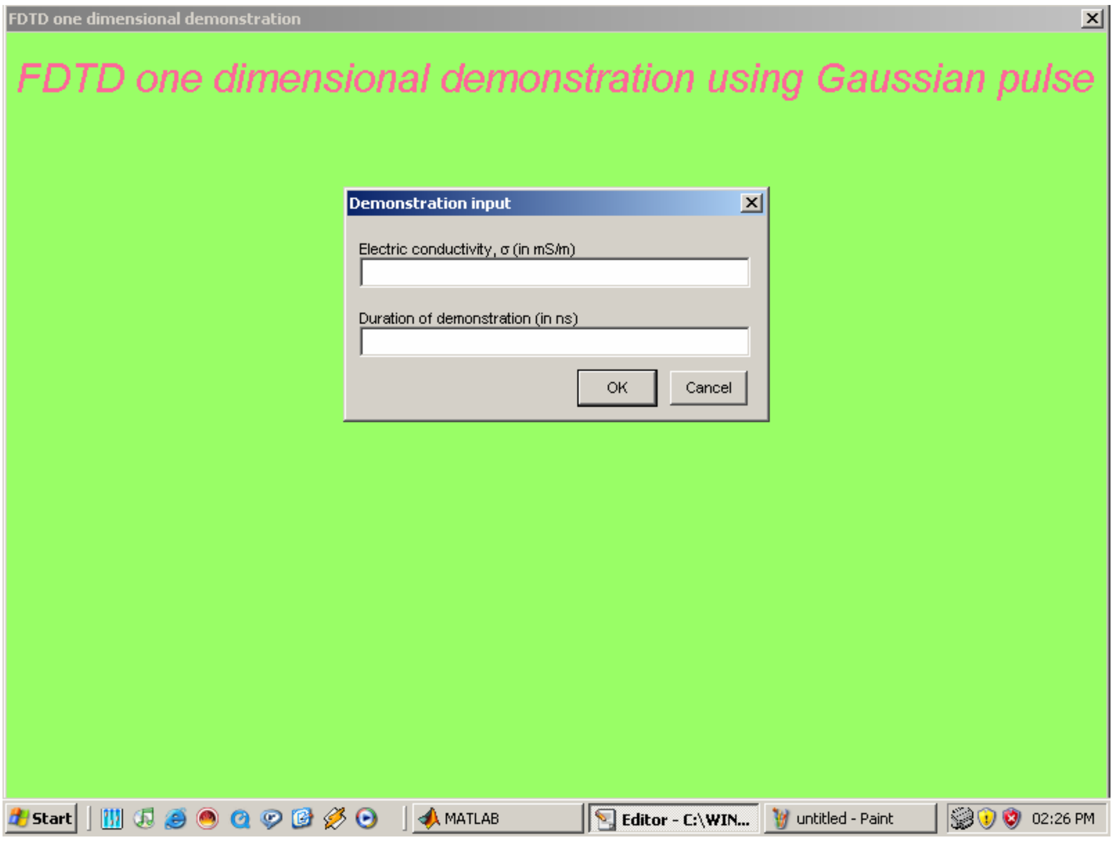

FDTD one dimensional demonstration
FDTD one dimensional demonstration using half-sine pulse
Demonstration input
Electric conductivity, σ (in mS/m)
Duration of demonstration (in ns)
OK
Cancel
Start
MATLAB
Editor - C:\WIN...
untitled - Paint
02:25 PM

Figure 5.11 Half-sine pulse menu

5.3.5 Fourth push-button

The fourth push button will show the sub-menu for the demonstration of Gaussian pulse. The sub-menu works exactly like the sub-menu of the sine-wave demonstration.


FDTD one dimensional demonstration
FDTD one dimensional demonstration using Gaussian pulse
Demonstration input
OK
Cancel
Start
MATLAB
Editor - C:\WIN...
untitled - Paint
02:26 PM

Electric conductivity, σ (in mS/m)

Duration of demonstration (in ns)

Figure 5.12 Gaussian pulse menu

5.4 The Graphical User Interface (GUI)

There is a GUI in this 1D program. A flowchart of the main menu is given below. If undergraduates want to transfer this GUI to their own computer programs to make them user-friendly, they can do so. More flowcharts can be found in the appendix.

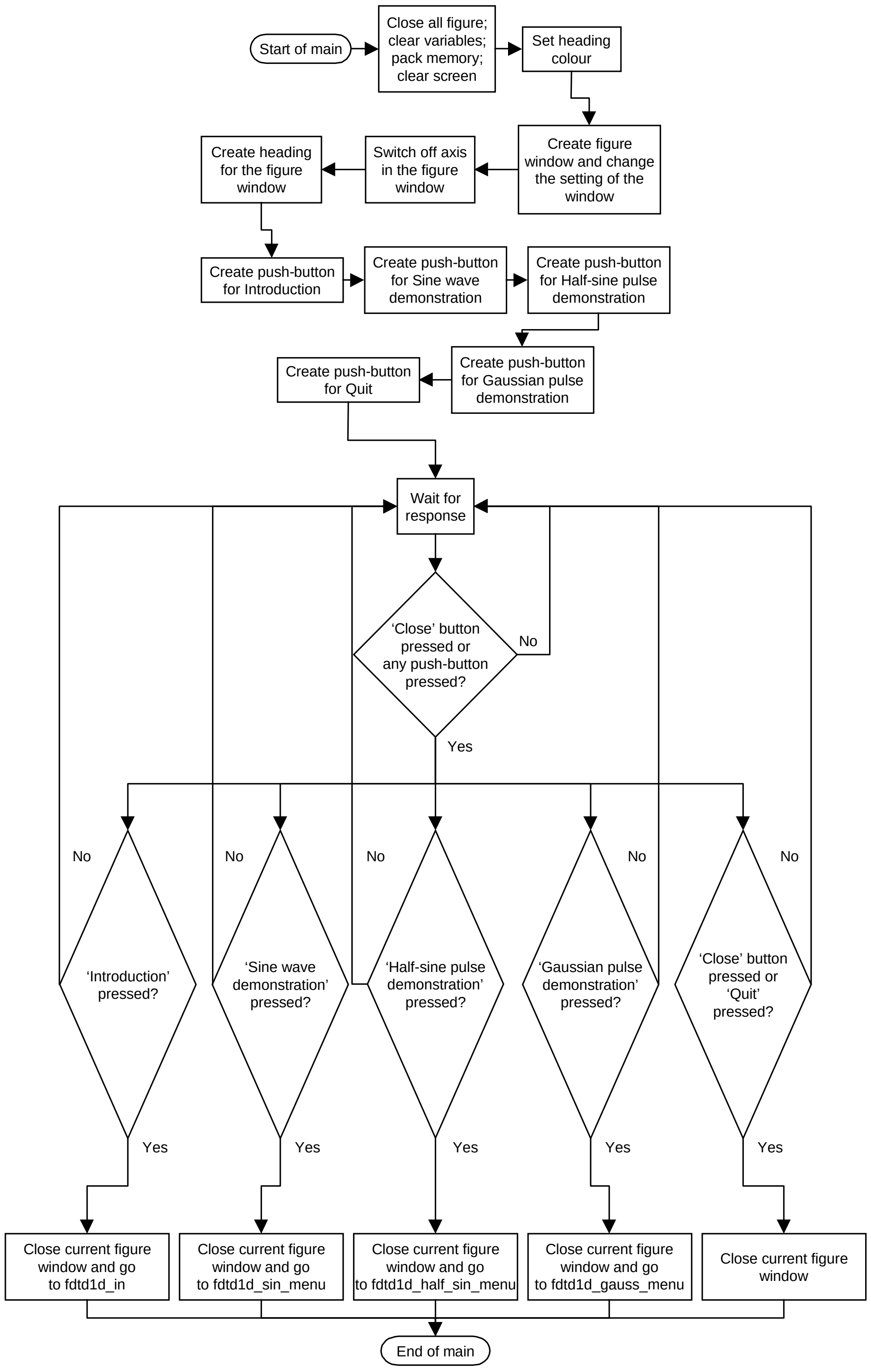

Start of main
Close all figure; clear variables; pack memory; clear screen
Set heading colour
Create figure window and change the setting of the window
Switch off axis in the figure window
Create heading for the figure window
Create push-button for Introduction
Create push-button for Sine wave demonstration
Create push-button for Half-sine pulse demonstration
Create push-button for Gaussian pulse demonstration
Create push-button for Quit
Wait for response
'Close' button pressed or any push-button pressed?
No
Yes
'Introduction' pressed?
'Sine wave demonstration' pressed?
'Half-sine pulse demonstration' pressed?
'Gaussian pulse demonstration' pressed?
'Close' button pressed or 'Quit' pressed?
No
Yes
Close current figure window and go to fdtd1d_in
Close current figure window and go to fdtd1d_sin_menu
Close current figure window and go to fdtd1d_half_sin_menu
Close current figure window and go to fdtd1d_gauss_menu
Close current figure window
End of main

Figure 5.13 Flowchart of main menu

Chapter 6: Application of FDTD to Radiation from a Narrow Slit

6.1 Introduction

Seven programs are adapted from the program fdtd2D written by Dr. Hagness (2000) for the purpose of showing the radiation from a narrow slit. The most important one is the second program, onewl2, a two dimensional model of radiation from a narrow slit excited by a bandpass pulse. This program will be used to develop the program, fdtd2d_gaussian_planewave to provide a 2D model of radiation from a narrow slit excited by a plane wave which is a bandpass pulse. This will fulfill one of the requirements of this project.

6.2 General Features of the Seven Programs

The seven programs have the following features in common. Four different sources are used in these seven programs.

§ Source 1 is a sinusoidal wave of 10 GHz.

§ Source 2 is a bandpass pulse with a carrier frequency of 5 GHz.

§ Source 3 is a plane wave generated by applying Huygens' principle on sinusoidal waves of 10 GHz.

§ Source 4 is plane wave obtained by setting source = 1 and source = -1 alternatively.

The object in fdtd2D, the metallic cylinder, is replaced by a perfect electric conductor (PEC). Then, a slit of width equals 10 cells is specified on the PEC, followed by width of 20 cells and width of 30 cells. Slit-width is taken to be equal to 10 cells, because a sinusoidal wave of 10 GHz has wavelength of 10 cells. The reason for the choice of the width of the slits will be shown in section 6.3.1.

Four different objects are used in the seven programs. They are:

- slit of width = 1 lambda = 1 wavelength = 10 cells;
- slit of width = 2 lambda = 2 wavelength = 20 cells;
- slit of width = 3 lambda = 3 wavelength = 30 cells; and
- perfect electric conductor (PEC).

The names of the seven programs are:

- onewl1;
- onewl2;
- onewl3;
- onewl4;
- twowl2;
- threewl2; and
- PEC2

Wavelength is abbreviated to wl. The last digits 1, 2, 3, 4 in the program names stand for source 1 (sinusoidal wave); source 2 (bandpass pulse); source 3 (Huygens' principle plane wave); and source 4 (source = 1 and source = -1).

In fdtd2D, E and H fields are displayed at every 4th time step. In these seven programs, E and H fields are displayed at every 8th time step in order to give faster display of the E and H fields.

6.2.1 The Program onewl1

The object in this program is a slit with slit-width equals to one wavelength. The source of excitation is source 1, a sinusoidal wave of 10 GHz. It will be shown in section 6.3.1 that a sinusoidal wave of 10 GHz has wavelength which is equal to 10 cells. Therefore, the slit-width is set at 10 cells.

In H_z, the concentric circles of different colours are clearly seen. When these circles reach the slit, they are almost straight lines. After these straight lines pass through the slit, they turn into semi-circles. It will be shown in section 6.3.3 that this phenomenon is correctly predicted by Huygens' principle.

In E_x and E_y, spots are found at the location of source of excitation after the source is switched off. This phenomenon is common in FDTD. The Yee's space lattice can have charge deposition if the source has a DC component. However, the source is a pure sine wave. This contradiction can be explained by the fact that charge is an integration of current over a period of time. For a pure sine wave, the charge is $Q(t) = 1 - \cos \omega t$. Therefore, a DC component is generated because the source is switched off at the time when the sine wave is not at 2π radian.

Some snapshots of the output from this program are given below.

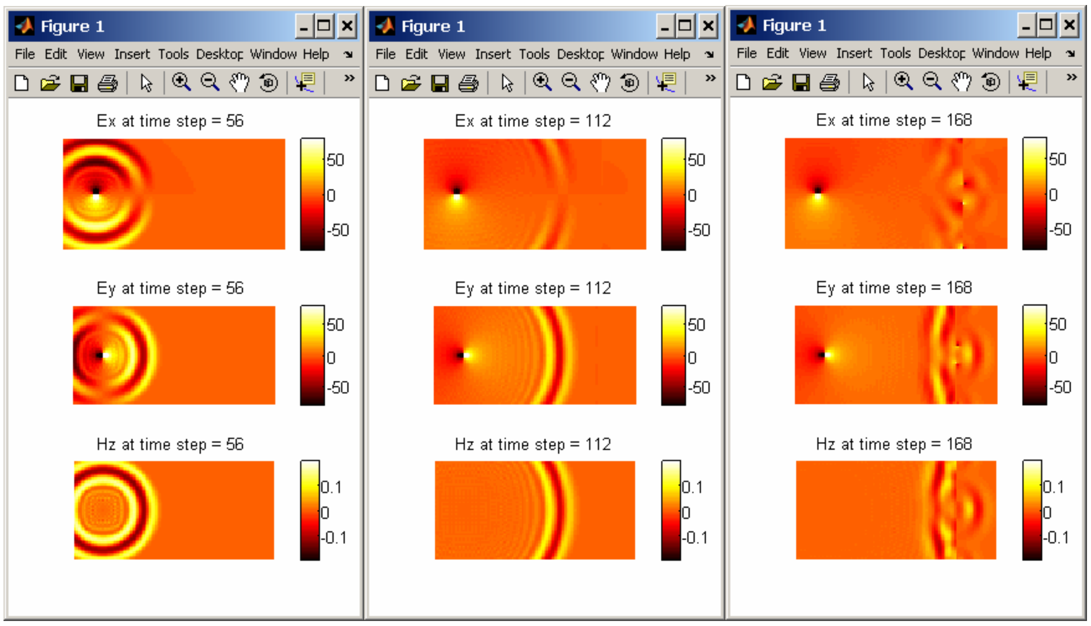


Figure 6.1 Snapshots of movie from onewl1.m

6.2.2 The Program onewl2

The slit-width is equal to one wavelength and the source is source 2, a bandpass pulse of 5 GHz at centre. The rest of the program is same as the program onewl1.

In E_x and E_y, spots are also found at the location of source of excitation after the source is switched off. Compared to the results from onewl1, the intensity of the spots is much smaller. This implies that the pulse has a very small DC component over a period of time.

Some snapshots of the output from this program are given below.

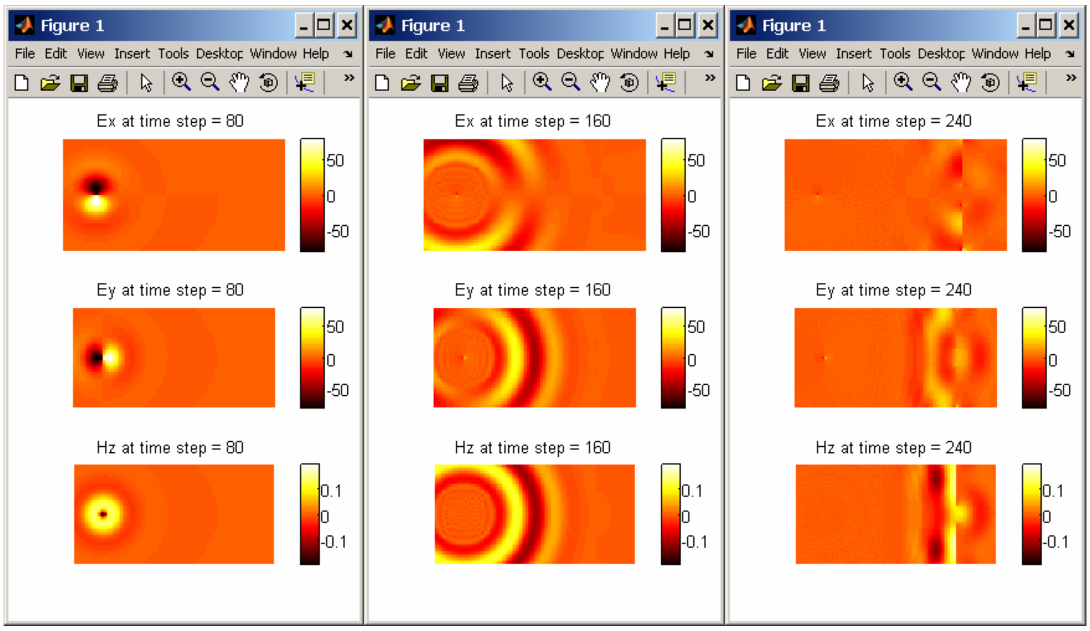


Figure 6.2 Snapshots of movie from onewl2.m

6.2.3 The Program onewl3

Huygens' principle is applied in this program to obtain a plane wave. 5 point sources of sine wave are used. In E_x, bright spots can be seen clearly at the tips of the slit. This can be explained by the fact that some structures, in FDTD, can deposit charges. In fact, in all of the slit programs, this phenomenon exists. After some observation, it is found that structures with sharp corners tend to accumulate charges. This may be due to a great discontinuity in the structure. The metallic cylinder, used in fdtd2D, does not accumulate any charges.

Some snapshots of the output from this program are given below.

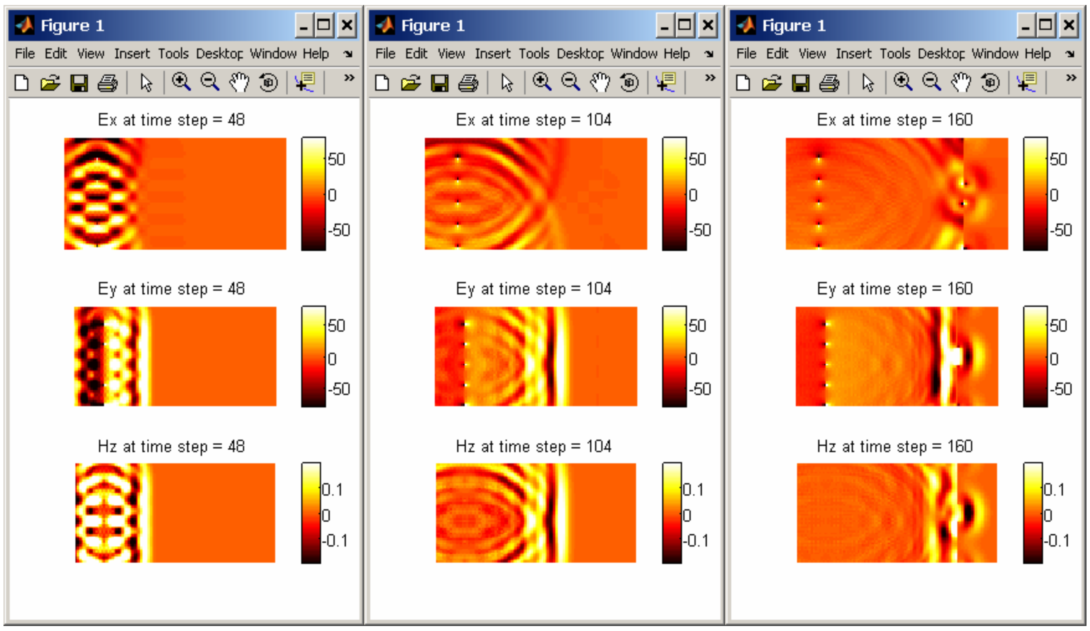


Figure 6.3 Snapshots of movie from onewl3.m

6.2.4 The Program onewl4

In this program, pseudo plane wave is created using vertical lines. A white vertical line represents 1; and a black vertical line represents –1. The source sends out four vertical lines in the order of white, black, white, black. After passing through the slit, these vertical straight lines become circular, which turns out to be consistent with Huygens' principle. No books have ever described this method of setting source = 1 and source = -1 alternatively. Theory on supporting this phenomenon is not found in any book. Therefore, this program, onewl4, should be looked upon as something interesting.

Some snapshots of the output from this program are given below.

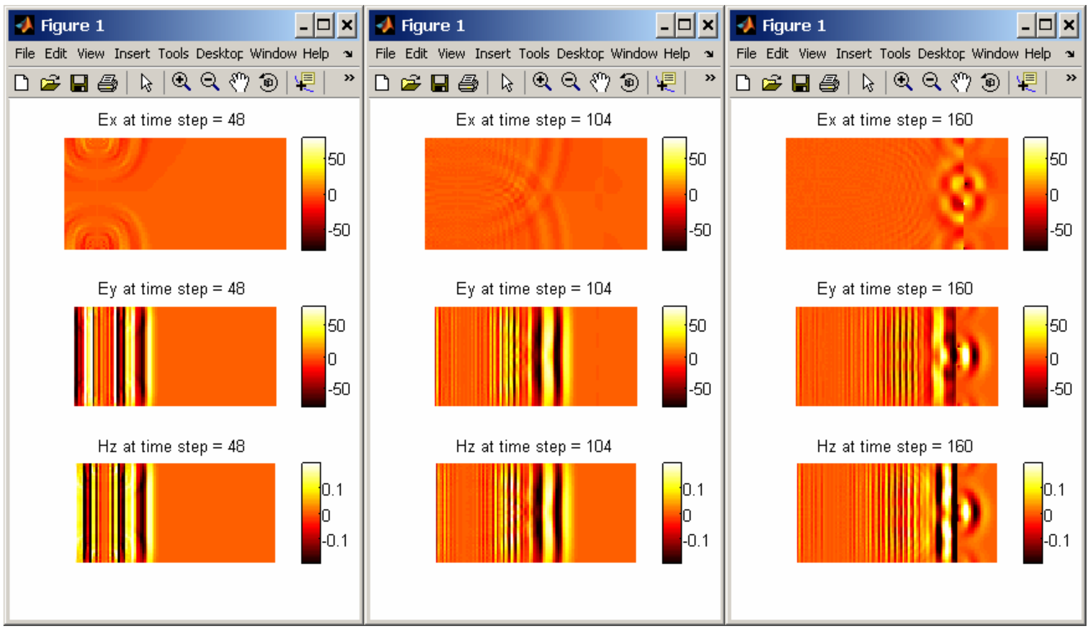


Figure 6.4 Snapshots of movie from onewl4.m

6.2.5 The Program twowl2

The slit-width is 20 cells and the source is a bandpass pulse. The rest of the program is same as those in program onewl2. When the width of the slit is more than one wavelength, the wave becomes less circular after it passes through the slit.

Some snapshots of the output from this program are given below.

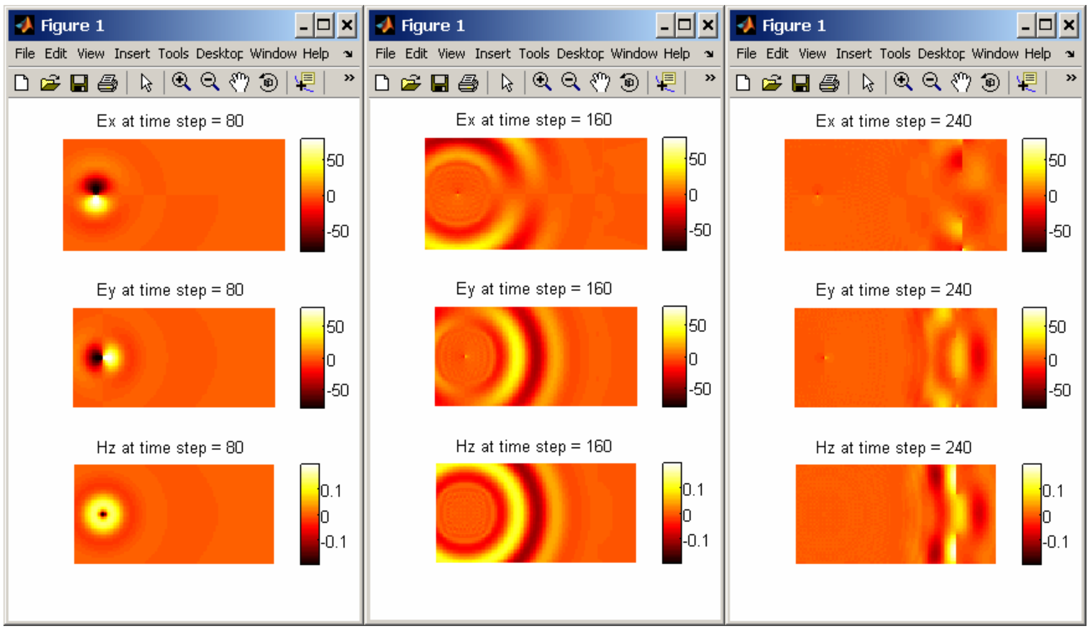


Figure 6.5 Snapshots of movie from twowl2.m

6.2.6 The Program threewl2

The slit-width is 30 cells and the source is a bandpass pulse. The rest of the program is same as those in program onewl2. Same phenomenon is observed, as in twowl2.

Some snapshots of the output from this program are given below.

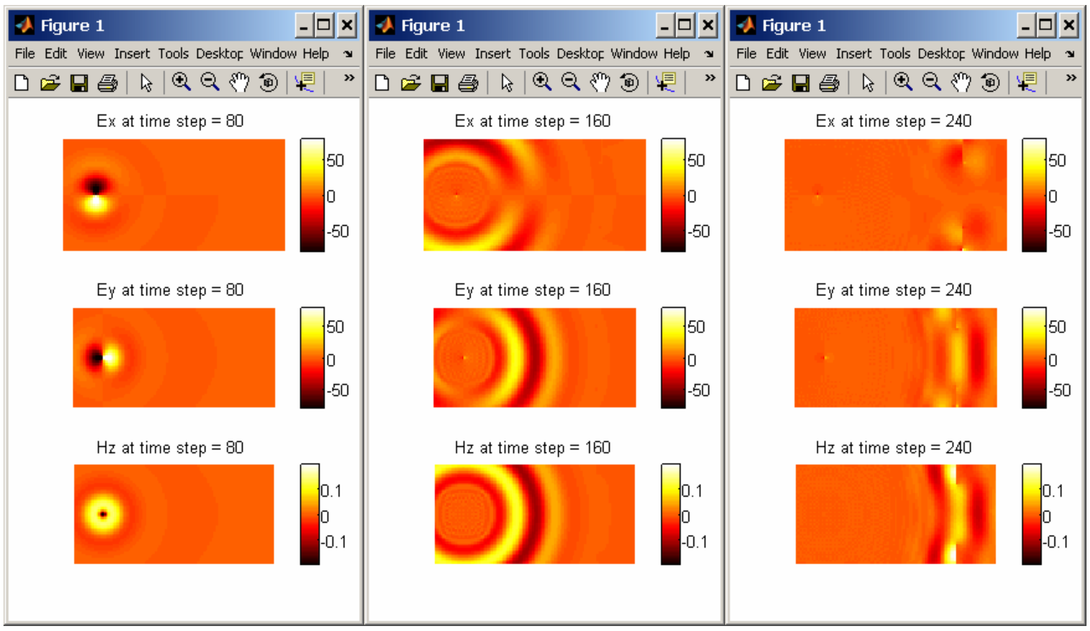


Figure 6.6 Snapshots of movie from threewl3.m

6.2.7 The Program PEC2

The source is a bandpass pulse. After running program PEC2, all waves are expected to be reflected back to the left of the PEC, and nothing should be on the right of the PEC. However, unexpected circular waves can be seen clearly on the upper and lower ends of the PEC. The snapshot of H_z at time step 256, clearly show the unexpected waveforms on the lower and upper ends of the right side of the PEC.

Some snapshots of the output from this program are given below.

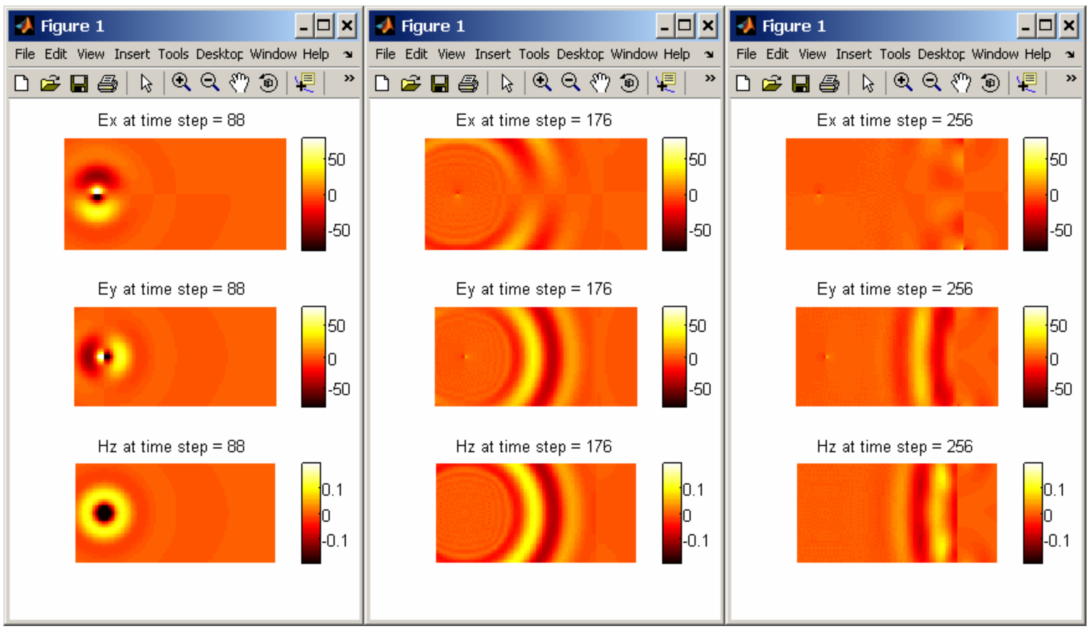


Figure 6.7 Snapshots of movie from PEC2.m

6.2.8 Discussion of Results

The seven programs correctly generate the expected waveforms. However, unexpected waveforms appear on the upper and lower ends of the right side of the PEC. Below is the enlarged view of the result from the program onewl1.

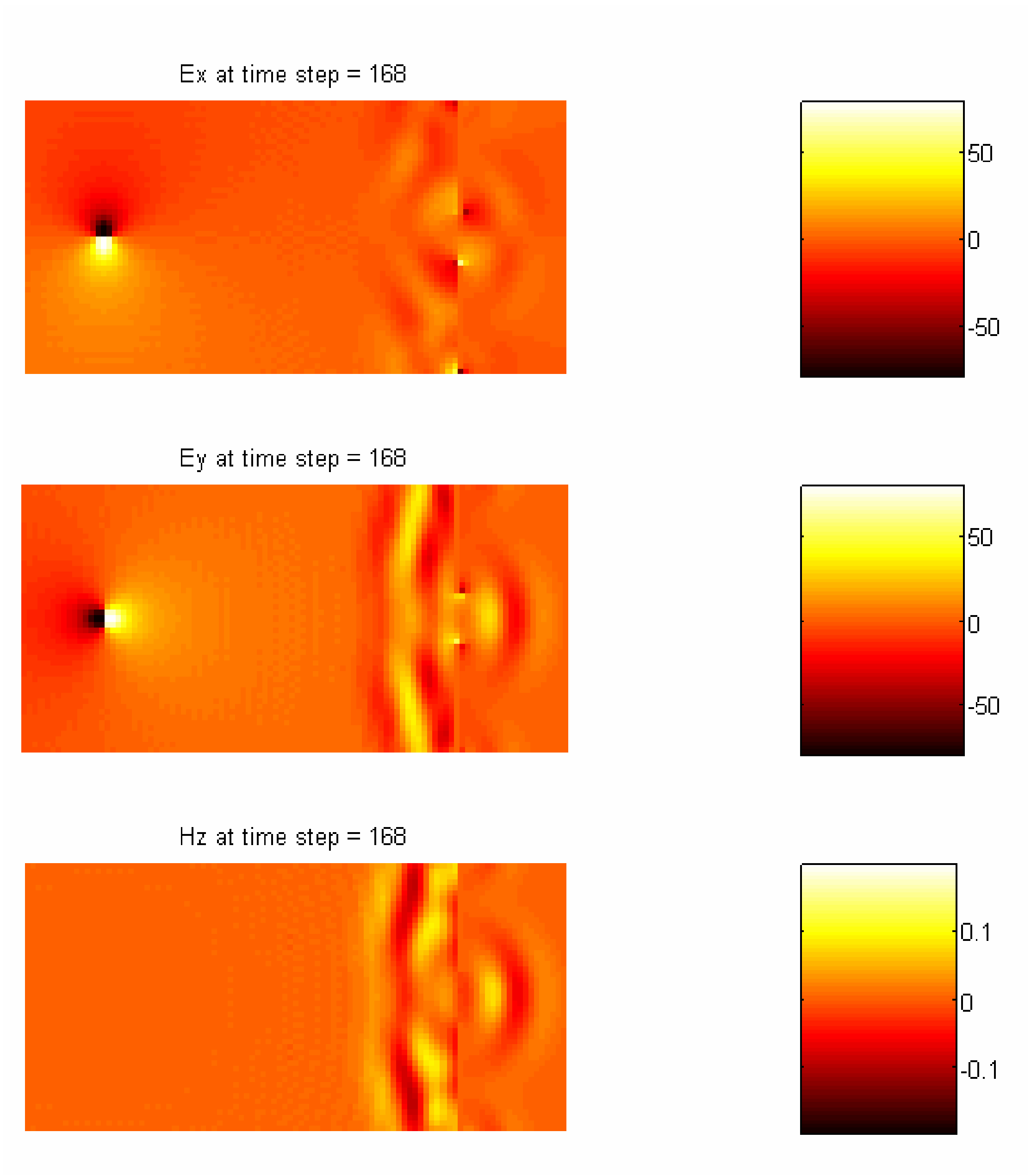


Figure 6.8 Leakage from PEC

In the above figure, leakage of red circumference, appears at the lower end of the metallic sheet in E_x, E_y and H_z.

6.3 Explanation of Slit Width and Huygen's Principle

This section explains the choice of the slit-widths and the frequency of the sinusoidal source. Then, Huygens' principle and the way to implement it on generating plane wave are described.

6.3.1 Calculation of Slit Width

The choice of a suitable wavelength for the sinusoidal source is given in the following steps.

§ In the program fdtd2D written by Dr. Hagness, one cell has length of 0.3cm. In order to reduce the numerical dispersion, there should be at least ten samplings in one wavelength. Ten samplings is 3cm. In other words, when one cell is 0.3cm in length, 3cm will contain 10 cells, that is, 10 samplings. If the wavelength of the source is chosen to be 1.5cm, with one cell of 0.3cm in length, there will be only 5 samplings in one wavelength, which is unacceptable. Hence, the shortest wavelength permitted is 3cm.

§ In fdtd2D, j is from 1 to 50. Results of using slit widths of one wavelength, two wavelengths, and three wavelengths will be investigated. If wavelength is chosen to be 3cm, 3 wavelengths will be 9cm. 9cm is 30 cells (1 cell equals 0.3cm), which occupies more than half of the 50 cells in j. If wavelength is too long, 3 wavelengths will be longer than 50 cells.

§ Therefore, the wavelength of the source is 3cm.

The structure of the slit is given below.

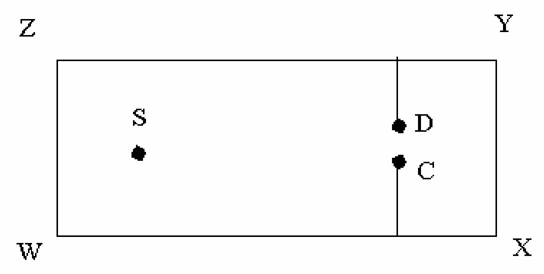


Figure 6.9 Structure with a slit

The coordinates are:

W(1,1)

Y(100,50)

S(15,25)

C(80,20)

D(80,30)

WXYZ is the computational region without PML.

S is the point where the source is put.

The slit-width CD is calculated as below.

From the program fdtd2D, the diameter of the cylinder is 6cm, and it occupies 20 cells. Therefore,

1 cell width = 6 cm /20 =0.3cm

Slit width = 0.3cm * (30-20) = 3cm

If wavelength is 3cm, the frequency of the source will be 1 * 10^{10} Hz. A sinusoidal wave of 10 GHz has wavelength which is equal to 3cm. 3cm is equivalent to 10 cells.

The steps on creating the slit in PEC are:

- initiate WXYZ as free space;
- for i= 80 and $1 \le j \le 20$, set the medium to be 2;
- for i = 80 and $30 \le j \le 50$, set the medium to be 2;

where medium 2 is a medium of high conductivity (that is, medium 2 is a metal)

In the first program, onewl1, a sinusoidal source is used. The codes that control the source are given below.

```
source(n) = sin(2 * pi * frequency * dt * n)=sin(omega * dt * n)

hz(is, js) = source(n), where is = 15, js = 25.
```

6.3.2 Setting the property of the medium to be metal

In the program fdtd2D written by Dr. Hagness, a metal cylinder is used in the program. Therefore, it is possible to create a metal sheet with a slit by just changing the coding describing the metal cylinder. The coding that describes a metal sheet with a slit in onewl1 is given below.

```
caex(80, [1:20]) = ca(2)

caex(80, [30:50]) = ca(2)

cbex(80, [1:20]) = cb(2)

cbex(80, [30:50]) = cb(2)
```

The logic of the program fdtd2D is not affected by the above changes, so the result should be correct. However, after running the program, unexpected circular waves are clearly seen on the upper and lower ends of the right side of the PEC.

6.3.3 Huygens' Principle and Diffraction

In 1678, the Dutch physicist Christian Huygens proposed his Huygens' principle. In this principle, which states that all points on a wavefront serve as point sources of spherical secondary wavelets. After a time t, the new position of the wavefront will be that of a surface tangent to these secondary wavelets. The principle is accepted by the community of scientists and remains useful today.

Huygens' principle provides a method of geometrical construction, to locate the position of a wavefront in future time, if the present position of a given wavefront is known.

When a plane wave of wavelength λ falls on a slit of width a, the plane wave spreads out into the geometrical shadow of the slit. This phenomenon is called diffraction. Diffraction is the property of all waves, including light wave, sound wave, water wave and electromagnetic wave.

As a/λ tends to zero, diffraction becomes more prominent. A light ray is an idealized construction for convenience, but it is impossible to produce a light ray physically. If a light ray is attempted to be produced by sending it through a narrow slit, the light always spreads out. The narrower the slit, the greater the spreading will be.

According to Huygens' principle, if the wavefront of a plane wave arrives on the left of a screen with a slit at the middle, every point on the wavefront can be viewed as an expanding spherical wavelet. The screen blocks off most of the wavelets, leaving a small portion of wavelets to pass through the slit. When the slit is reduced to a point, only one wavelet can pass through it. By Huygens' principle, the wavelet that passes through the slit is spherical, so there is no more plane wave on the right of the screen. Therefore, the phenomenon of diffraction is consistent with Huygens' principle.

6.3.4 Implementation of Plane Wave using Huygens' principle

Sadiku (2003) applied the Huygens' principle in a computer program written in FORTRAN to generate a plane-wave source. By adapting the FORTRAN coding, a MATLAB program, onewl3 is written. Below is the code generating a plane-wave source using the Huygens' principle.

```
source(n) = sin(omega * dt * n)

hz(15,1:10:je) = source(n)
```

15 is the location of the source along the x-axis, and the location of the point sources along the y-axis is from 1 to je (which is 50 in this program), separated at an interval of 10.

6.3.5 Discussion of Results

A PEC is implemented in computer program. In the middle of the PEC, slits of one wavelength, two wavelengths and three wavelengths are created. Waves are sent through the slits and the shape of the waves is distorted. This phenomena observed on the computer screen are consistent with Huygens' principle.

When setting medium to 2, this logical and straight forward approach leads to an unexpected result. Circular waves are clearly seen on the upper and lower ends of the right side of the PEC. This problem is solved in section 6.4.

6.4 Further improvement in programs

In section 6.2, a PEC is added in the programs by following the same method as specifying a metal cylinder in the program, fdtd2D. Since this approach does not change the rest of the coding in fdtd2D, the programs in section 6.2 should be free of error. Unfortunately, unexpected circular waves appear on the upper and lower ends of the right side of the PEC. In this section, the cause and the method of removal of the leakage are explained.

After the leakage is removed, four additional programs are written. These programs are designed to let the simulation run longer so that the reflections become more visible. In these programs, E and H fields are displayed at every 4th time step.

6.4.1 Setting the E-field and H-field to zero

Initially, it is suspected that the leakage is caused by E-field and H-field not set to zero on the PEC surfaces. Sevgi (2003) states that a thin wire, starting from (i, j, k) and having length of four Δz, located along z-axis, may be simulated in FDTD by setting

Ez (i, j, k) = 0

Ez (i, j, k+1) = 0

Ez (i, j, k+2) = 0

Ez (i, j, k+3) = 0

Setting the fields to zero may change the logic of the program fdtd2D, so it is not implemented at first, which makes the method of setting medium to be 2 (that is, setting the medium to be metal), become the first choice initially.

Since there are unexpected leakages after setting medium to be 2, the following lines are added to program onewl1.

```
ex(80, 2:20) = 0
ex(80, 30:je) = 0
ey(80, 1:20) = 0
ey(80, 30:je) = 0
hz(80, 1:20) = 0
hz(80, 30:je) = 0 , where je = 50.
```

With these six lines added, the leakage still exists but less obvious. Nevertheless, these six lines improve program onewl1. Also, they make the method of setting medium to be 2 redundant. Either method can create a PEC. Both methods at the same time are not necessary.

The PEC is initially suspected to be too short. This may be the cause of the leakage. However, changing the length of PEC to ex(80, 1:20) or ex(80, 30:51) will cause error in calculation and the computer stops.

6.4.2 Cause of Leakage

The figure below explains the cause of the leakage. The origin of the PML in fdtd2D is moved to the left of the computational region by 8 cells.

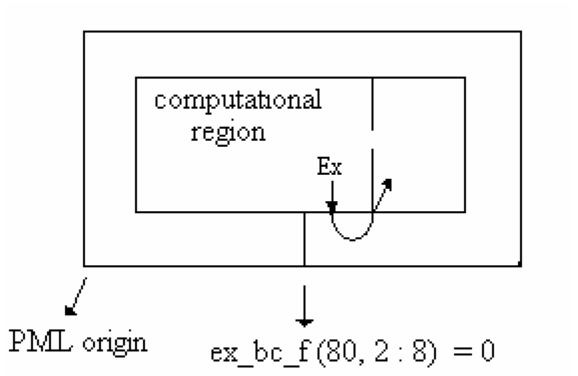


Figure 6.10 Position of PML origin and propagation of E_x

Once the position of the PML origin is known, the reason for leakages is now clear. The sine wave propagates into the PML region between ex_bc_f(80, 2:8) = 0 and the metal sheet. The sine wave is not totally absorbed by the PML. The weakened sine wave propagates back to the computational region behind the metal sheet.

Therefore, the removal of leakages is achieved by shifting ex_bc_f(80,2:8) = 0 to the right by 8 cells. That is to set ex_bc_f(80+8,2:8) = 0. However, this will only remove the leakage in the front PML in E_x. In order to completely remove the leakage, the same principle will be applied to the back of PML in E_x, the front PML in E_y, and the back PML in E_y.

In short, adding a metal sheet (i.e. PEC) in the computational region is not enough. The metal sheet is needed to be extended into the front and back of PML by 8 cells.

6.4.3 Removal of Leakage

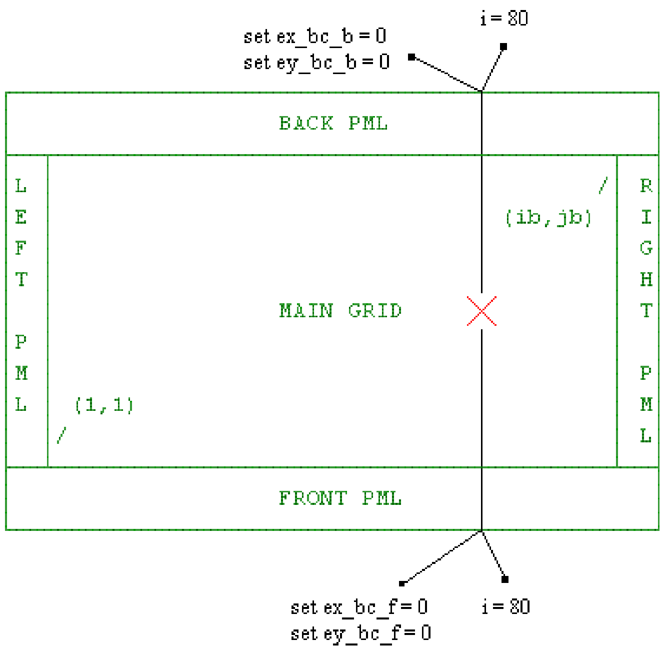


Figure 6.11 Structure with a single slit

The method of removal of leakages is simple. From the above figure, the matrices ex_bc_f, ex_bc_b, ey_bc_f, and ey_bc_b are set to zero at a particular location. The coding that removes leakage is given below.

ex_bc_f(wall_at_x_axis + PML_FB, 2: PML_FB) = 0

ex_bc_b(wall_at_x_axis + PML_FB, 2: PML_FB-1) = 0

ey_bc_f(wall_at_x_axis + PML_FB, :) = 0

ey_bc_b(wall_at_x_axis + PML_FB, :) = 0

wall_at_x_axis is set to 80, and PML_FB is set to 8. wall_at_x_axis is the location of the PEC along the x-axis. PML_FB represents the thickness of front and back of the PML regions.

6.4.4 The Program onewl1_new

Program onewl1_new is the same as program onewl1 except that the leakages are removed.

Some snapshots of the output from this program are given below.

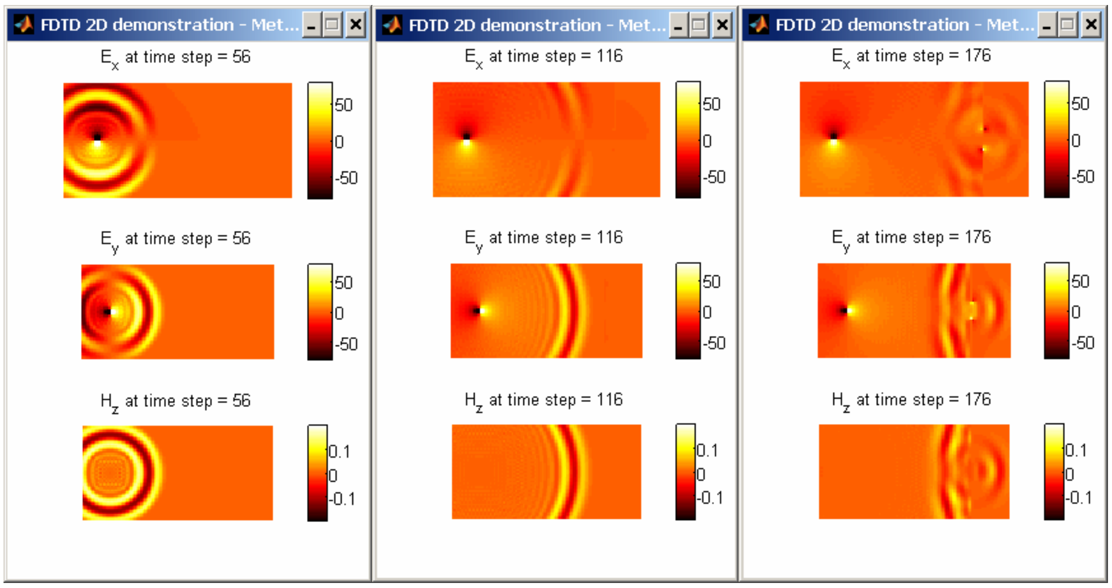


Figure 6.12 Snapshots of movie from onewl1_new.m

6.4.5 The Program onewl2_new

This program is the same as onewl1_new, except that the source is changed to a bandpass Gaussian pulse. At time step 248, on the left side of the PEC is the reflected wave, while on the right side of the PEC is the diffracted circular wave.

Some snapshots of the output from this program are given below.

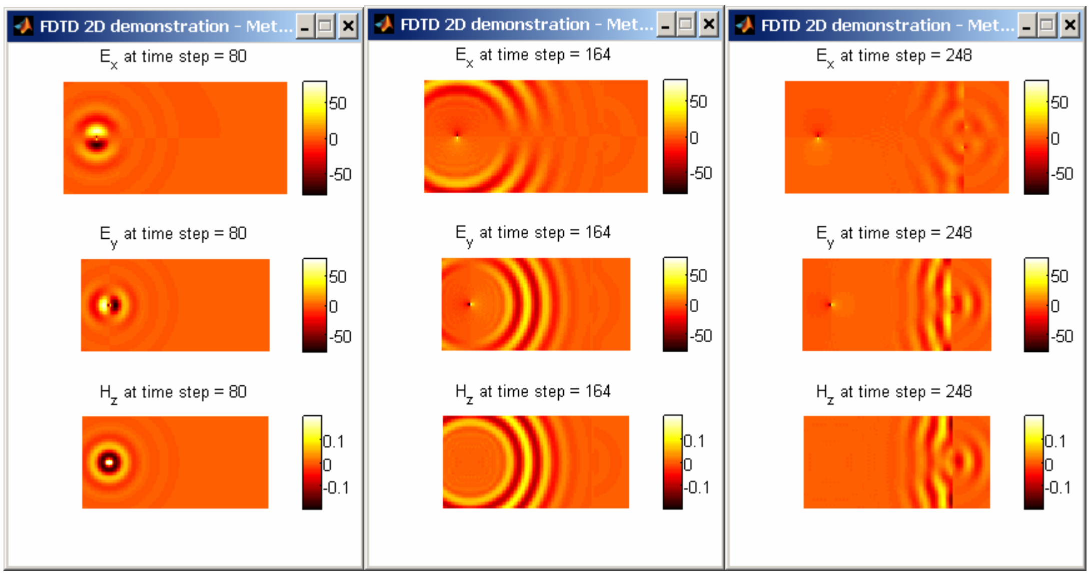


Figure 6.13 Snapshots of movie from onewl2_new.m

6.4.6 The Program onewl1_wall

The source is a sine wave. The object is a wall of 10 cells in thickness.

In the middle picture, E_y, the wall can be clearly seen. The slit at the middle is yellow in colour. The wall is from i = 70 to i = 80. The rest is the same as onewl1_new.

Some snapshots of the output from this program are given below.

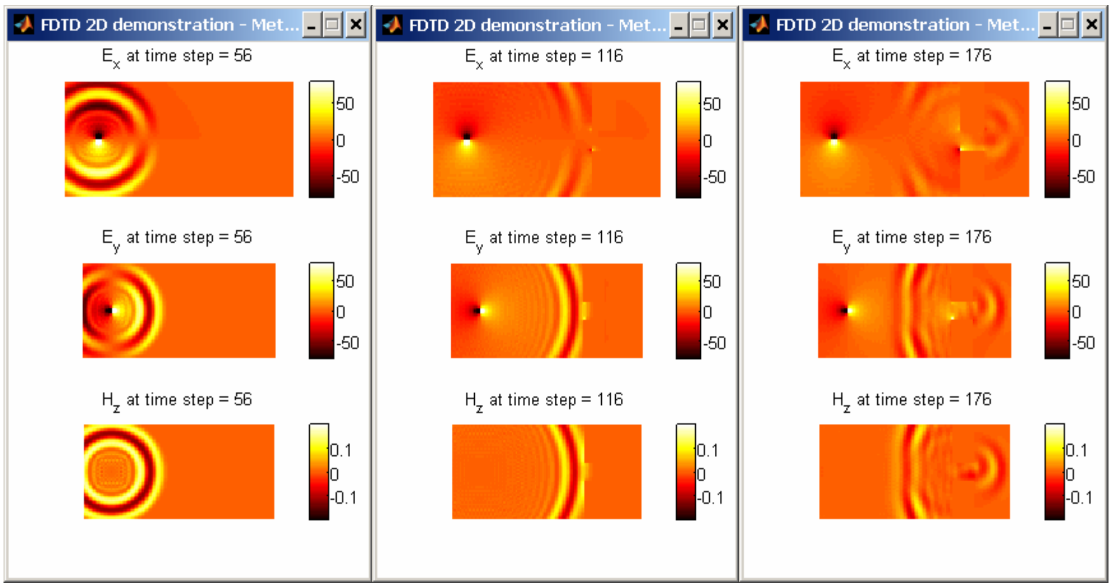


Figure 6.14 Snapshots of movie from onewl1_wall.m

6.4.7 The Program onewl2_wall

The source is a bandpass Gaussian Pulse. The object has a PEC thickness of 10 cells, from i = 70 to i = 80. The PEC can be clearly seen in E_x, E_y and H_z.

Some snapshots of the output from this program are given below.

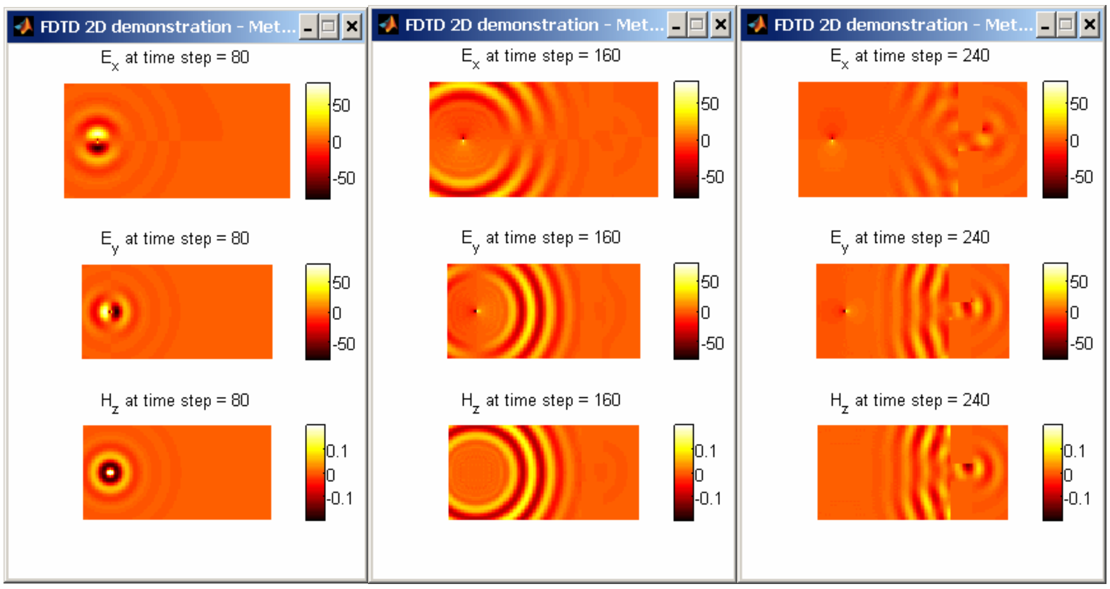


Figure 6.15 Snapshots of movie from onewl2_wall.m

6.5 The Program fdtd2d_gaussian_planewave

In this section, fdtd2d_gaussian_planewave is created to meet the requirement of the specification. This program uses Huygens' principle to generate a plane wave. In this program, 5 bandpass Gaussian pulse sources are used to create a plane wave. The metal sheet with a slit, has a finite thickness. When the plane wave passes through the slit, the slit behaves like a waveguide. After passing through the slit, the plane wave becomes a circular wave, as predicted by Huygens' principle. No leakage is observed after the plane wave passes through the metal sheet.

Some snapshots of the output from this program are given below.

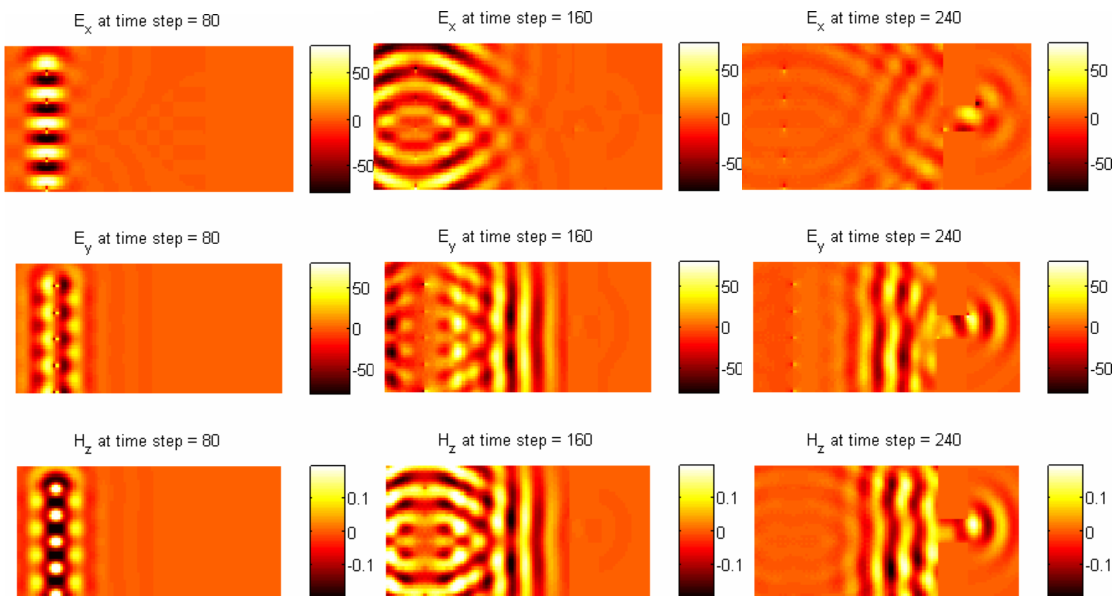


Figure 6.16 Plane wave using bandpass Gaussian pulse

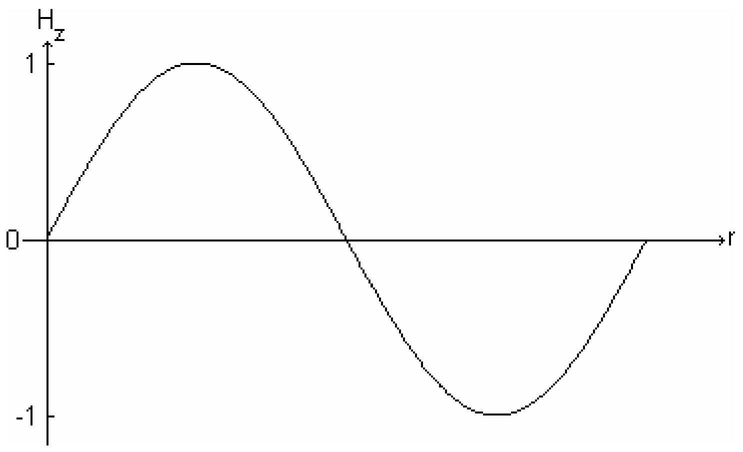
6.6 Directions of Propagation

From the three movies of E_x, E_y and H_z in the programs showing the waves passing through a slit, E_x propagates towards the bottom and top, and E_y propagates towards the left and right. It seems that the x-axis is from bottom to top, and the y-axis is from left to right. However, this is only an illusion which is not true.

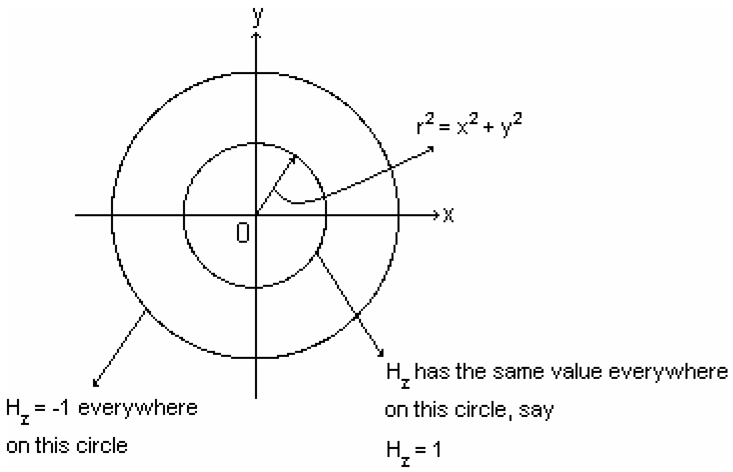
Although E_x looks like propagating along the bottom to top direction, the E_x vector is actually pointing from left to right. In this section, E_x, E_y, and H_z are solved analytically and graphs are plotted to see its shape.

6.6.1 Analytical Solutions of Fields

A one-dimensional H_z = sin r is like this.



A two-dimensional $H_z = \sin r$, where $r^2 = x^2 + y^2$ is like this.


y
x
0
r² = x² + y²
H_z = -1 everywhere on this circle
H_z has the same value everywhere on this circle, say H_z = 1

From Maxwell's equations in two dimension without any external sources,

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} \tag{6.1}$$

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} \tag{6.2}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left[\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right]$$

Let $H_z = \sin \omega t$

For 2D sine function, $\omega t = \sqrt{x^2 + y^2}$, $t = \dfrac{\sqrt{\left(x^2 + y^2\right)}}{\omega}$ (6.3)

$$H_z = \sin\left(\sqrt{\left(x^2 + y^2\right)}\right) \tag{6.4}$$

From (6.4), $\dfrac{\partial H_z}{\partial y} = \cos\left(\left(x^2 + y^2\right)^{1/2}\right)\left(\dfrac{1}{2}\right)\left(x^2 + y^2\right)^{-1/2}(2y)$

From (6.1), $\dfrac{\partial E_x}{\partial t} = \left(\dfrac{y}{\varepsilon}\right)\left(x^2 + y^2\right)^{-1/2}\cos\left(\left(x^2 + y^2\right)^{1/2}\right)$

Integrate $\dfrac{\partial E_x}{\partial t}$, $E_x = \left(\dfrac{yt}{\varepsilon}\right)\left(x^2 + y^2\right)^{-1/2}\cos\left(\left(x^2 + y^2\right)^{1/2}\right)$

From (6.3), $t = \dfrac{\sqrt{\left(x^2 + y^2\right)}}{\omega}$

$$E_x = \left(\frac{y\left(x^2+y^2\right)^{1/2}}{\omega\varepsilon}\right)\left(x^2+y^2\right)^{-1/2}\cos\left(\left(x^2+y^2\right)^{1/2}\right)$$

$$E_x = \left(\frac{y}{\omega\varepsilon}\right)\cos\left(\left(x^2+y^2\right)^{1/2}\right)$$

From (6.2), $\frac{\partial E_y}{\partial t} = \left(\frac{-x}{\varepsilon}\right)\left(x^2+y^2\right)^{-1/2}\cos\left(\left(x^2+y^2\right)^{1/2}\right)$

Integrate $\frac{\partial E_y}{\partial t}$, $E_y = \left(\frac{-xt}{\varepsilon}\right)\left(x^2+y^2\right)^{-1/2}\cos\left(\left(x^2+y^2\right)^{1/2}\right)$

From (6.3), $E_y = \left(\frac{-x\left(x^2+y^2\right)^{1/2}}{\omega\varepsilon}\right)\left(x^2+y^2\right)^{-1/2}\cos\left(\left(x^2+y^2\right)^{1/2}\right)$

$$E_y = \left(\frac{-x}{\omega\varepsilon}\right)\cos\left(\left(x^2+y^2\right)^{1/2}\right)$$

Summarizing,

if $H_z = \sin\left(\sqrt{\left(x^2+y^2\right)}\right)$ is a 2D sine function, then

$$E_x = \left(\frac{y}{\omega\varepsilon}\right)\cos\left(\left(x^2+y^2\right)^{1/2}\right)$$

$$E_y = \left(\frac{-x}{\omega\varepsilon}\right)\cos\left(\left(x^2+y^2\right)^{1/2}\right)$$

A program fdtd2d_fix_screen is written to plot the three final equations.

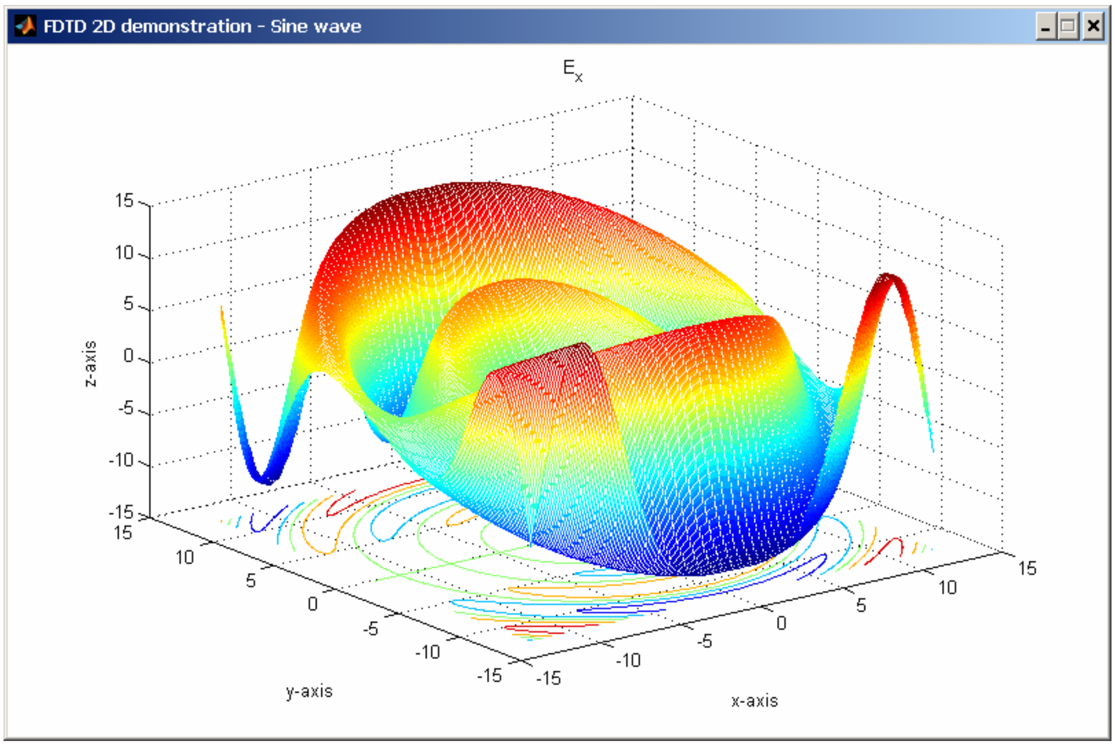


Figure 6.17 E_x in free space

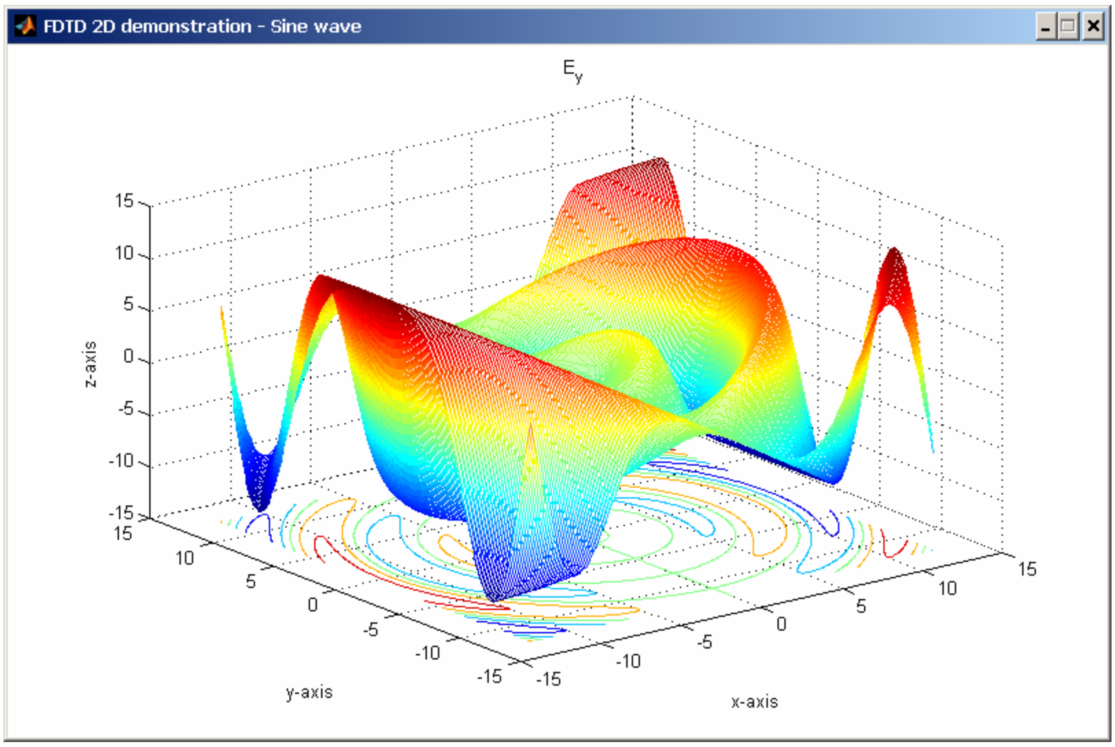


Figure 6.18 E_y in free space

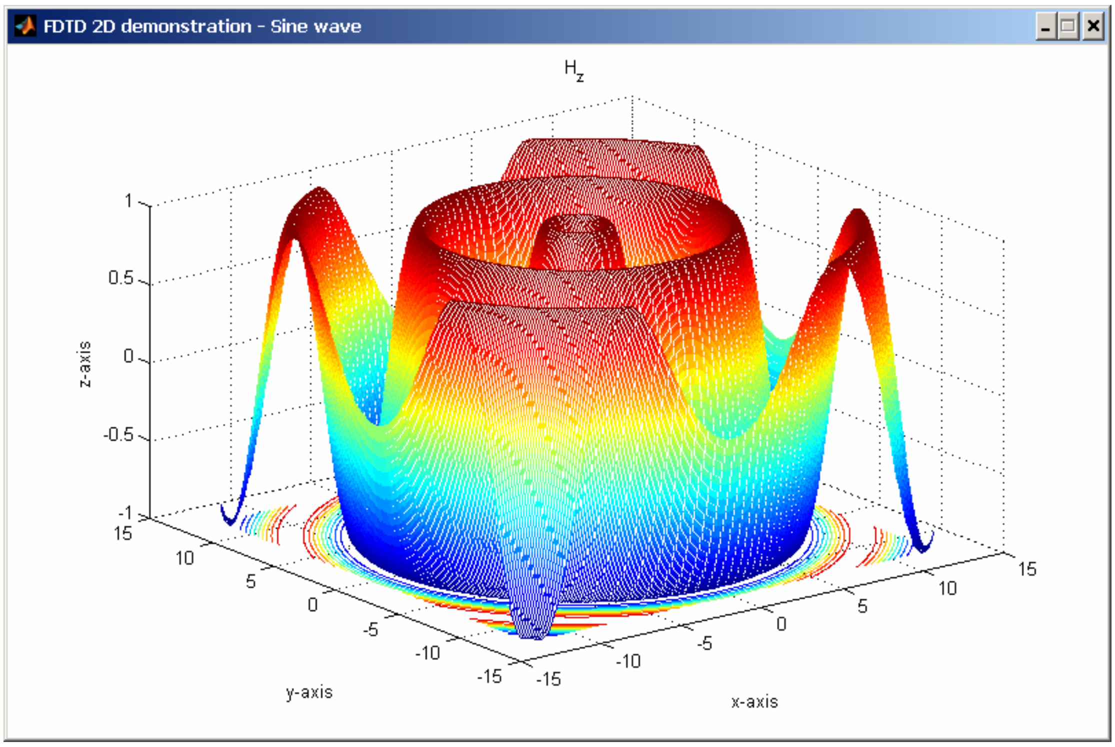


Figure 6.19 H_z in free space

6.6.2 Summary of Plots of the Analytical Solutions

The directions of H_z, E_x and E_y are determined by the directions used in equations, not by their shapes in their graphs. Some significant points are noted in the graphs of H_z, E_x and E_y.

The projections of H_z on x-y plane are concentric circles, same as those in slit programs using sine wave source.

The projections of E_x on x-y plane seems to propagates along y-axis, but E_x is actually defined in the equation to point along x-axis.

Since, the deduced equation of E_x contains y, an exchange of x-axis and y-axis in the graph (i.e. set x-axis points in up and down and y-axis points in left and right), will be followed by an exchange in E_x direction. Therefore, E_x always looks as if propagating along y-axis, no matter how to put the x-axis and y-axis in the graph.

Chapter 7: Application of FDTD to Time Domain Reflectometer (a TDR probe)

7.1 Introduction

TDR can be used to measure the amount of moisture in soil (Topp et al. 1982). A TDR probe, consisting of two parallel wires, is pushed into the soil. The TDR probe can now be treated as a transmission line, filled with soil and terminated by an open circuit. A pulse is generated by a step generator, and the time taken for the reflection to return to the source is measured. The velocity of the wave from the source and the apparent dielectric constant of the soil can then be calculated from the time. From the apparent dielectric constant, the soil moisture can be found from the look-up table supplied with the TDR. A typical TDR system is given in the figure below.

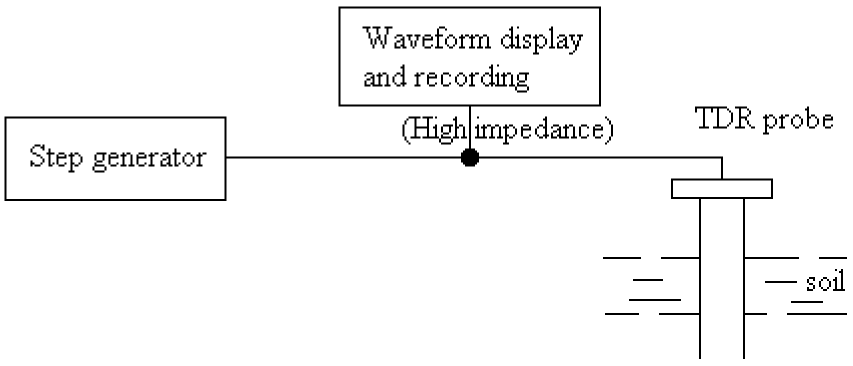


Figure 7.1 A TDR System

In this project, it is required to simulate a 2D model of a TDR probe of a given geometry. A raised cosine plane wave is introduced into the TDR probe.

For the raised cosine, it is required to specify the time constant which sets the rise time. Then a computer program is written to implement the simulation.

7.2 The TDR Probe Structure

The following is a brief description of the structure of TDR probe. Below is the structure of the probe used for simulation.

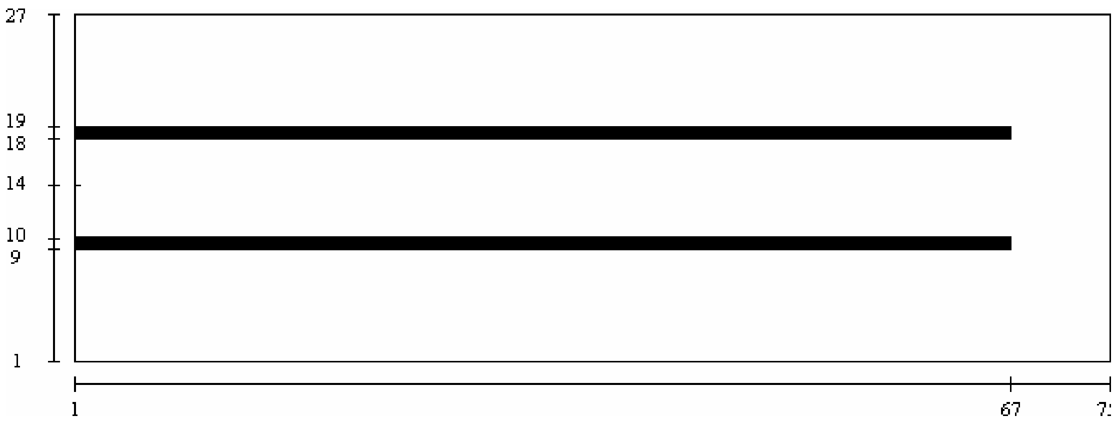


Figure 7.2 The TDR Probe Structure

It is already known that 3mm = 1 cell. Therefore, 200mm is approximated by 67 cells (67 x 3 = 201), and 25mm is approximated by 8 cells (8 x 3 = 24).

The two metallic plates are from j = 9 to 10, and j = 18 to 19. In x-direction, i = 200mm + 25mm = 67 cells + 8 cells = 75 cells. In y-direction, j = 25mm + 3mm + 25m + 3mm + 25mm = 81mm = 27 cells.

7.3 Calculation of the Time-Constant, T

The time constant, T of the source is chosen in such a way that the wave just reaches one of the metal plates of the TDR probe.

Below is the calculation of the time for the wave to travel from the source to one of the metal plates is calculated.

$$dx = 3\times10^{-3}\,m = 3mm$$

$$dt = \frac{dx}{2\times c}$$
$$dt = \frac{3\times10^{-3}}{2\times3\times10^{8}}$$
$$dt = 5\times10^{-12}\,s$$

Time for $1\,dx$ is $10\times10^{-12}\,s$ (= 2dt)

$\therefore$ Time for 4 dx is $40\times10^{-12}\,s$ (from j = 14 to j = 18)

The source being used for this simulation is a step function using raised-cosine function.

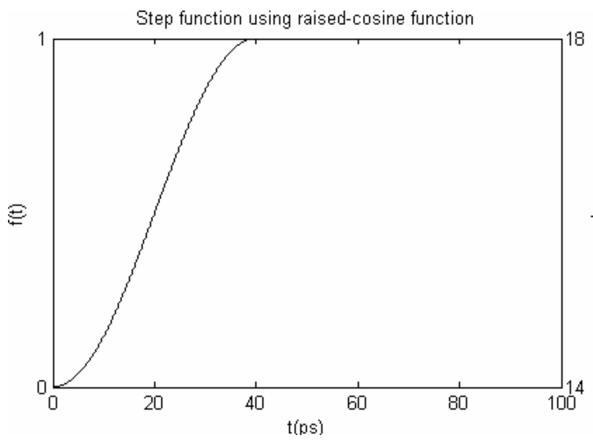


Figure 7.3 Raised-cosine step function

The raised-cosine function is f(t) = 0.5[1-cos(pi*t/T)].

If f(t) = 1 when t = $40*10^{-12}$, then

1 = 0.5[1-cos(pi*40*10^{-12}/T)]

T = 40*10^{-12}s

Therefore, the time-constant for the raise cosine is T = 40 picoseconds.

7.4 Program Result

A program, fdtd2d_tdr_ver1 is written to simulate the TDR probe. Some snapshots of the output from this program are given below.

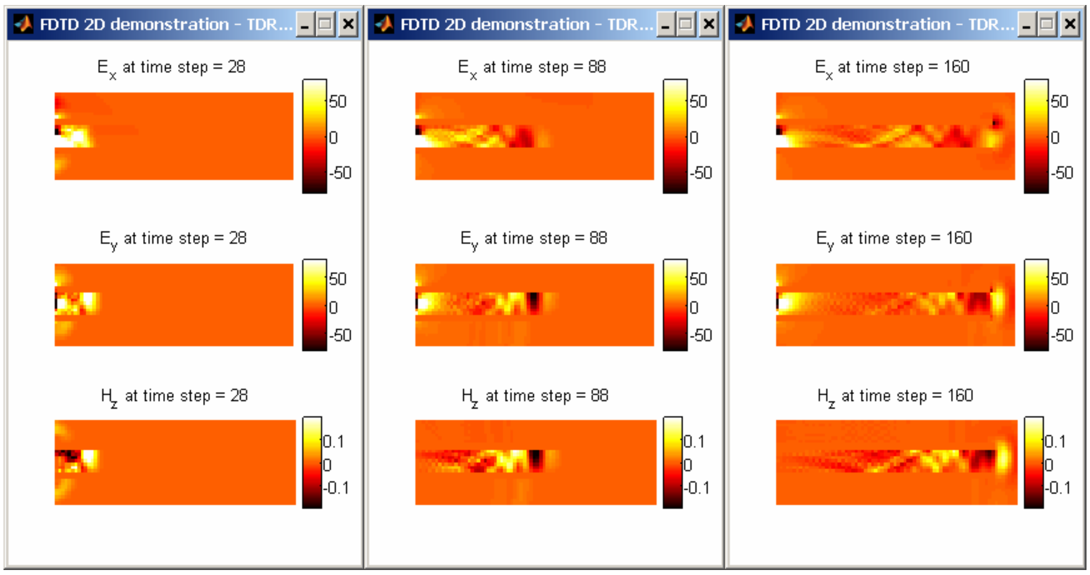


Figure 7.4 Snapshots of movie from fdtd2d_tdr_ver1.m

In this program, leakage can be seen clearly at the sides of the metal plates of the TDR probe. Also, the source fails to generate a plane wave. Below is the enlarged view of the result from the program fdtd2d_tdr_ver1.

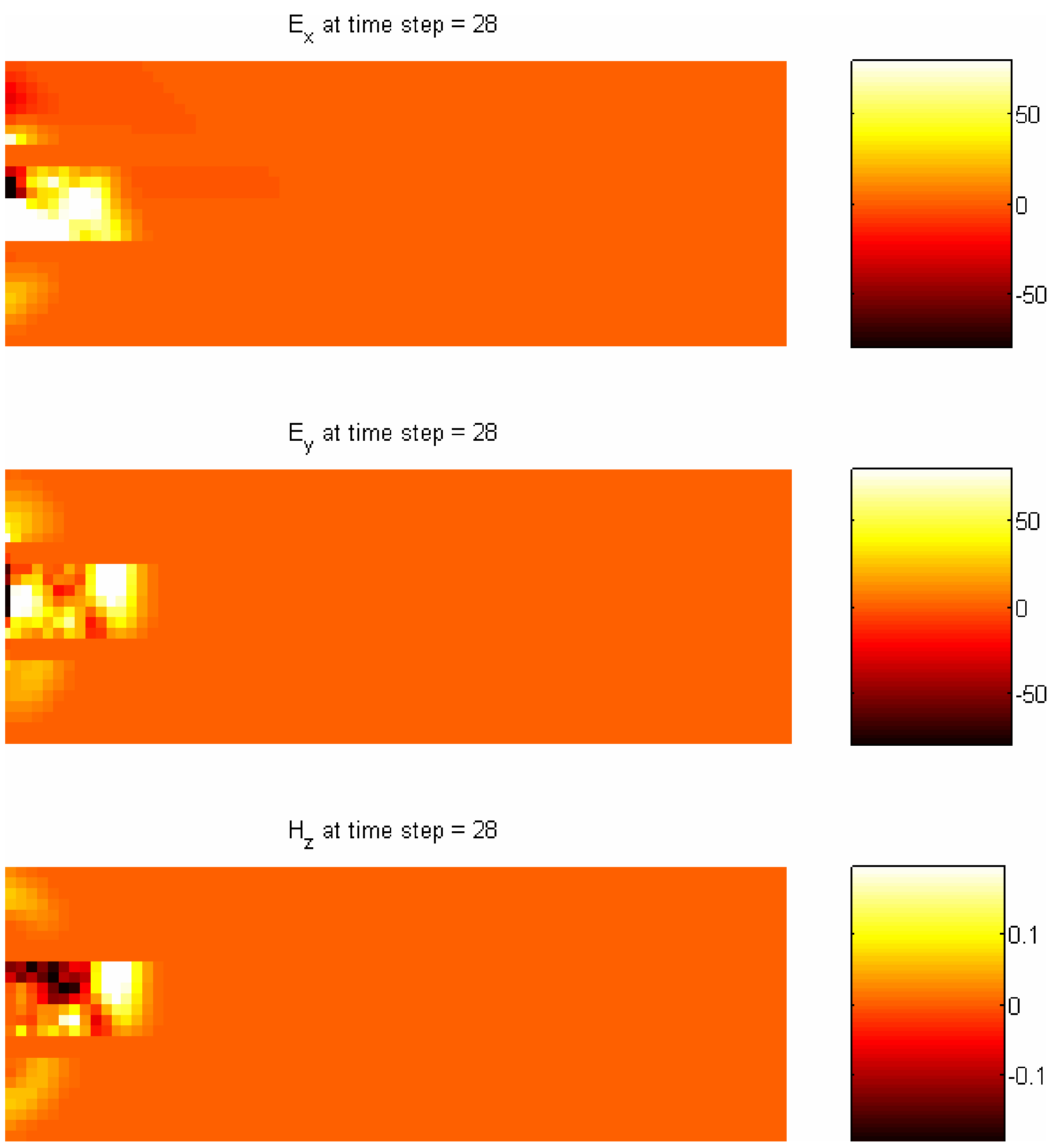


Figure 7.5 Leakage from TDR probe

The leakage cannot be explained by using the same theory in section 6.4.3. After reading the program code, it is found that the left PML cannot absorb the leakage whenever the source starts.

The source fails to generate plane wave. When the waves from the source reach the metal plates, some of them reflect back to the source. Since hard source is used, the reflected waves corrupt the source.

In E_x and E_y, there is a large amount of charge deposition at the source point. It is because the source has a strong DC component.

7.5 Removal of Leakage in TDR probe

The leakage problem can be solved by using the similar approach in section 6.4.3 by setting the appropriate PML layers to zero. Instead of setting a particular location in the left PML in E_x and the left PML in E_y to zero, both of them are entirely set to zero. Below is the code added to remove the leakage.

```
ex_bc_l(:,:) = 0;
ey_bc_l(2:PML_LR,:) = 0;
```

A program, fdtd2d_tdr_ver2 is written to show the removal of leakage. Some snapshots of the output from this program are given below.

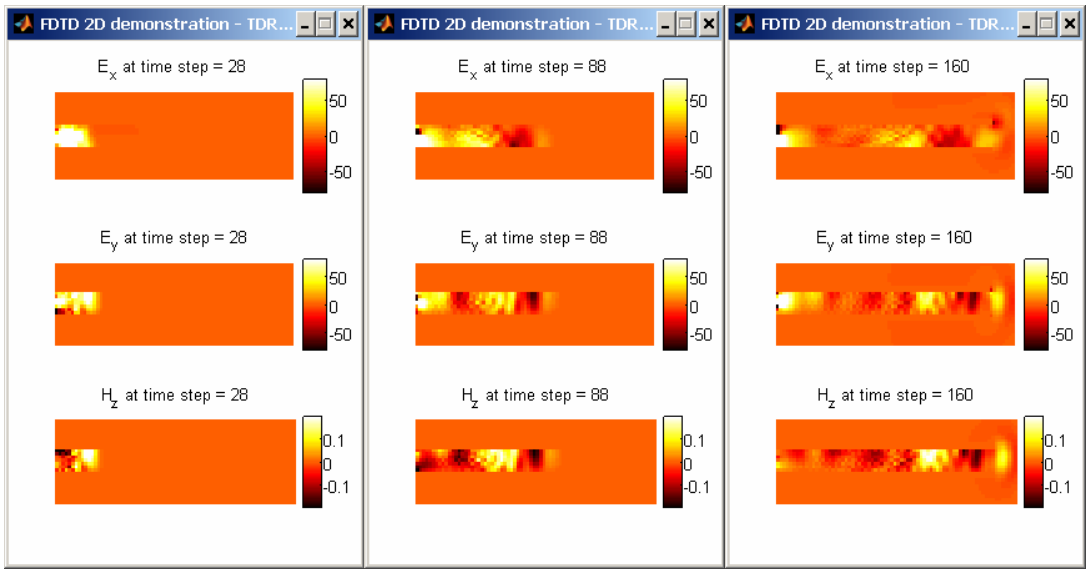


Figure 7.6 Snapshots of movie from fdtd2d_tdr_ver2.m

The leakage is successfully removed. The problem of generating a plane wave will be dealt with in the next section.

7.6 Generation of a Plane Wave using Point Source

In this project, plane waves can be generated by using either Huygens' principle or having a point source placed far enough to make the waves look like plane waves. For this TDR probe simulation, the latter method is chosen for the generation of plane waves.

In the previous two programs, the source is too close to the two metal plates of the TDR probe. One of the solutions is to place the source far from the probe. Alternatively, the probe can be shortened to allow the waves from the source propagate like plane waves. The second option is preferred because the entire area of the FDTD computation domain remains unchanged. The FDTD computation time increases if the first option is implemented. The first option needs the area of the FDTD computation domain to increase, so that the source can be far enough from the TDR probe. Below is the modified TDR probe structure.

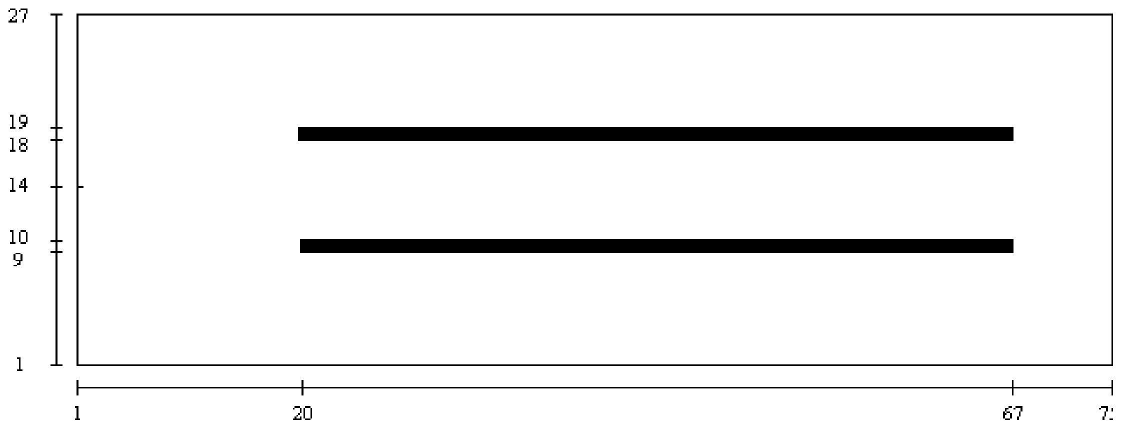


Figure 7.7 The Modified TDR Probe Structure

A program fdtd2d_tdr_ver3 is written to simulate the TDR probe using a plane wave source having a raised cosine step function. Some snapshots of the output from this program are given below.

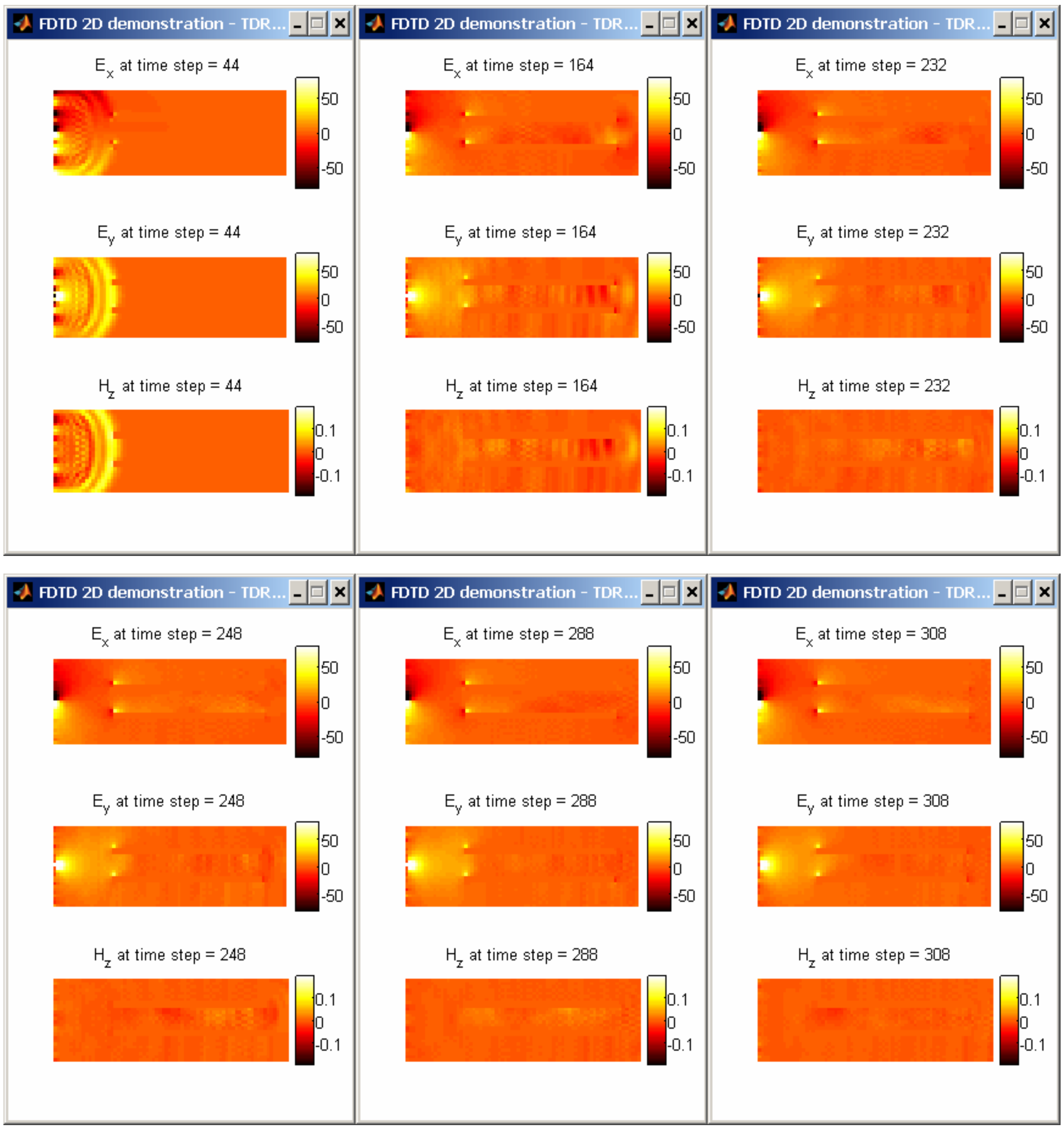


Figure 7.8 Snapshots of movie from fdtd2d_tdr_ver3.m

From the above diagrams, the waves from the source behave like plane waves at time step 44. The waves radiate from the TDR probe. At about time step 232, some waves are reflected back to the source from the end of the TDR probe. This phenomenon can be explained by treating the TDR probe as if it is a transmission line without a matched load. When a transmission line is not terminated by a correct load, some waves will be reflected from the point where the mismatch occurs. Similarly, the ends of the TDR probe are not connected to anything. Therefore, reflection will occur in this demonstration.

Chapter 8: Suggestions for Further Work and Conclusions

8.1 Suggestions for Further Work

It will be good to implement the following suggestions to improve the educational values of this project.

8.1.1 Suggestions for 1D

Figure 8.1 shows the result of a simulation where the computation region is divided into two equal parts. The left side is free space (relative dielectric constant = 1), and the right side is a dielectric medium with relative dielectric constant = 4.

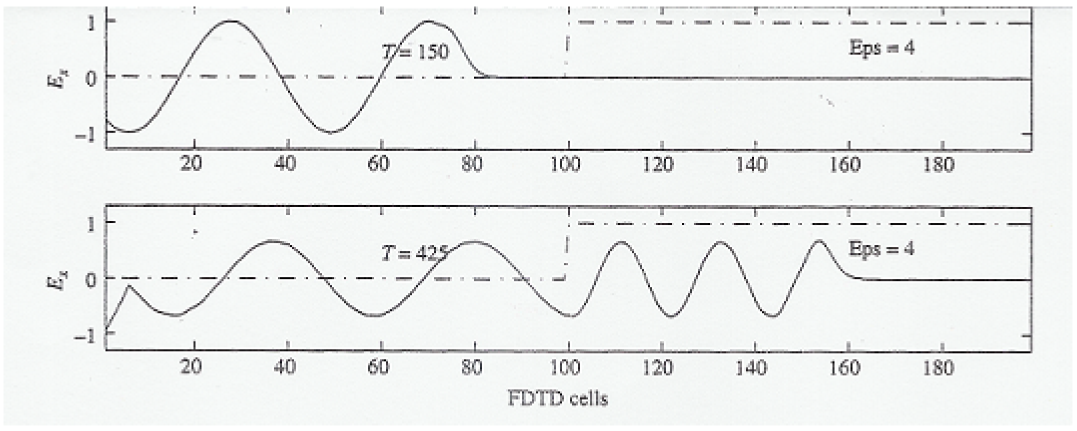


Figure 8.1 Simulation of a sinusoidal wave hitting a dielectric medium

An example related to Figure 8.1 is light traveling from air into glass. The wavelength of light will become shorter after entering into glass. The upper figure shows simulation at time-step T=150, and the lower figure at T = 425.

Figure 8.2 shows the simulation of a sinusoidal wave hitting a lossy dielectric medium. The lossy dielectric medium has relative dielectric constant = 4 and conductivity = 0.04 (S/m).

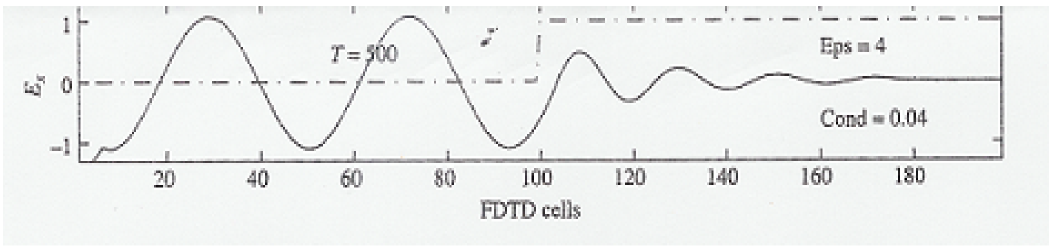


Figure 8.2 Simulation of a sinusoidal wave hitting lossy dielectric medium

(Sullivan 2004, pp. 5-10).

8.1.2 Suggestion for 2D narrow slit

The suggestion is to obtain the far field from the near field, which is already found in this project, by near-field-to-far-field transformation.

Using the near-field data obtained, the Transformation is used to calculate the complete far-field radiation pattern of the narrow slit. There is no need to extend the FDTD computational region to cover the far field region.

Taflove and Hagness (2000) provided the details of the Transformation. Although partial programming in FORTRAN is given, it is still difficult to understand.

8.1.3 Suggestion for 2D TDR probe

i) A hard source of raised cosine was used in this project. The suggestion is to use a plane-wave soft source.

ii) The 2D TDR probe is an open-end transmission line. The suggestion is to write a passage to explain the relation between the static capacitance at the open end, and the time delay of the reflection from the open end back to the source point, (i.e. the relation between capacitance and time delay).

iii) The final suggestion is to extend the 2D TDR probe to 3D TDR probe.

8.2 Project Conclusions

In this project, most of the objectives in the specification are met.

i) From the survey of available software for solving electromagnetic problems, it was found that some companies and educational institutions offer free download FDTD software or trial versions of software packages for a limited period of time. The cost of licensed FDTD software is about USD 5000 to USD 6000.

The computer hardware requirement depends on the FDTD software. A computer with Intel Pentium 4 as CPU, 512 MB of RAM, and a moderate performance graphic card, is recommended for running moderate FDTD software.

ii) The specific objectives to write three computer programs for displaying the phenomena of electromagnetic wave propagations are met.

For the 1D FDTD simulation, the need to simulate an infinite space when using FDTD was pointed out. ABC is a method to meet this need. Mur’s first-order ABC was, step by step, derived from wave equation in details. Mur’s first-order ABC is simple in concept, easy to implement in computer program and works very well in 1D. A GUI is written for the 1D FDTD simulation, which allows users to choose different sources and enter different values of time-duration and conductivity, when running the program. The GUI also allows users to save the simulation results in video format.

iii) For the 2D simulation of radiation from a narrow slit, unexpected phenomenon appeared. The PML was expected to absorb all electromagnetic (EM) waves, and EM waves could not pass through PEC. Surprisingly, EM wave was seen to pass through the connecting points between PEC and PML. This phenomenon was investigated and the ‘passing” was stopped. This 2D simulation correctly showed the diffraction phenomenon. Huygens’ principle was applied to generate plane-waves in the slit programs.

iv) For the 2D simulation of TDR probe, a hard source is located in the middle of two metallic plates that comprise the probe. The waves from the hard source are reflected back by the two metallic plates to the hard source. The hard source prevents the movement of the reflected waves to pass through the source position to reach the PML. Therefore the hard source fails to simulate an incident wave properly (Taflove & Hagness, 2000).

The metallic plates are shortened on the left side so that the probe is a distance from the source. Waves reaching the probe could then be approximated as plane-waves and no more reflected waves are observed.

In conclusion, this project gives me a good learning experience in appreciating such an advanced topic in the area of electromagnetics. The application of FDTD can be found everywhere, and research on FDTD still continues. Therefore, to have a good understanding in FDTD will be beneficial to students.

List of References

Books

Binns, K.J., Lawrenson, P.J., & Trowbridge, C.W., 1995, *The Analytical and Numerical Solution of Electric and Magnetic Fields*, John Wiley & Sons, Inc., New York.

Harrington, R.F. 1993, *Field Computation by Moment Methods*, IEEE Press, New York.

Hilliday, D., Resnick, R. & Walker, J. 2000, *Fundamentals of Physics*, Fifth Edition, John Wiley & Sons, Inc., New York.

Huebner, K.H., Thornton, E. A. & Bryom, T.G.. 1995, *The Finite Element Method for Engineers, Third Edition*, John Wiley & Sons, Inc., New York.

Jeffreys, H. & Swirles, B., 1966, *Methods of Mathematical Physics*, Cambridge University Press, Cambridge.

Kunz, K.S. & Luebbers, R.J. 1992, *The Finite Difference Time Domain Method for Electromagnetics*, CRC Press, London.

Peterson A.F., Ray S.L. & Mittra R. 2002, *Computational Methods for Electromagnetics*, IEEE Press, New York.

Rao, N.N., 2004, *Elements of Engineering Electromagnetics, sixth edition*, Prentice Hall, New Jersey.

Rao, S.M. 1995, *Time Domain Electromagnetics*, IEEE Press, New York.

Sadiku, M.N.O. 2003, *Numerical Techniques in Electromagnetics*, Second Edition, CRC Press, London.

Sevgi, L. 2003, *Complex Electromagnetic Problems and Numerical Simulation Approaches*, IEEE Press, New York.

Sokolnikoff, I.S. & Redheffer, R.M., 1958, *Mathematics of Physics and Modern Engineering*, McGraw Hill, New York.

Sui, W. 2001, *Time-Domain Computer Analysis of Nonlinear Hybrid Systems*, CRC Press, London.

Sullivan, D.M. 2004, *Electromagnetic Simulation Using the FDTD Method*, IEEE Press, New York.

Taflove, A. 2004, *Advances in Computational Electrodynamics*, Artech House, London.

Umashankar, K. & Taflove, A. 1993, *Computational Electromagnetics*, Artech House, London.

Volakis, J.L., Chatterjee, A. & Kempel, L.C. 1998, *Finite Element Method for Electromagnetics*, IEEE Press, New York.

Journals

F. Zheng and Z. Chen, “Numerical Dispersion Analysis of the Unconditionally Stable 3-D ADI-FDTD Method,” IEEE Transactions on Microwave Theory and Techniques, Vol. 49, No.5, pp.1006-1009, May, 2001.

F. Zheng, Z. Chen and J. Zhang, “Toward the Development of a Three-Dimensional Unconditionally Stable Finite-Difference Time-Domain Method,” IEEE Transactions on Microwave Theory and Techniques, Vol. 48, No.9, pp.1550-1558, September, 2000.

G.C. Topp, J.L. Davis and A.P. Annan, “Electromagnetic Determination of Soil Water Content Using TDR,” Soil Sci. Soc. Am. J., Vol.46, 1982.

H.H. Todd, “Survey of Numerical Electromagnetic Modeling Techniques,” University of Missouri-Rolla, September, 1991.

J.B. Schneider and C.L. Wagner, “FDTD Dispersion Revisited: Faster-Than-Light Propagation,” IEEE Microwave and Guided Wave Letters, Vol.9, No.2, pp.54-56, February, 1999.

J.P. Berenger, “An Effective PML for the Absorption of Evanescent Waves in Waveguides,” IEEE Microwave and guided wave letters, Vol. 8, No.5, May 1998.

O.M. Ramahi, “The Concurrent Complementary Operators Method for FDTD Mesh Truncation,” IEEE Transactions on Antennas and Propagation, Vol. 46, No.10, pp.1475-1482, October, 1998.

T. Namiki, “A New FDTD Algorithm Based on Alternating-Direction Implicit Method,” IEEE Transactions on Microwave Theory and Techniques, Vol. 47, No.10, pp.2003-2007, October, 1999.

W. Yu & R. Mittra, “A Conformal FDTD Software Package Modeling Antennas and Microstrip Circuit Components,” The Pennsylvania State University.

Websites

Celia Home Page [Online], 2005. Available: http://www.virtual-science.co.uk/celia/Celia_code/celia_home.htm [Accessed 11 March 2006].

Educational resources available from Field Precision [Online], 2006. Available: http://www.fieldp.com/educa.html [Accessed 07 Mar. 2006].

EMS Plus [Homepage of EMS-Plus] [Online], 07 May 2005 - last update. Available: http://www.ems-plus.com/ [Accessed 11 Mar. 2006].

FDTD Method [Online], 2005. Available: http://www.electromagnetics.info/fdtd/theory.htm [Accessed 13 Mar. 2006].

FDTD.org Home [Homepage of FDTD.org] [Online], 05 Feb. 2006 - last update. Available: http://www.fdtd.org/ [Accessed 07 Mar. 2006].

Field Precision Ordering Information - Finite Element Engineering Software for Magnetic and Electric Field Analysis [Online], 2006. Available: http://www.fieldp.com/order.html [Accessed 07 Mar. 2006].

Maxwell's Equations [Online], 2006. Available: http://www.electromagnetics.info/maxwell.htm [Accessed 13 Mar. 2006].

WestGrid Software [Online], 2005. Available: http://www.westgrid.ca/support/software/ [Accessed 11 Mar. 2006].

FDTD in electromagnetics [Online], 2005. Available: http://www.electromagnetics.info/fdtd/ [Accessed 07 Mar. 2006].

Zeland Software, Inc. [Homepage of Zeland Software, Inc.] [Online], 10 Jan. 2006 - last update. Available: http://www.zeland.com/ [Accessed 07 Mar. 2006].

Zeland Software, Inc-Fidelity [Online], 2004. Available:
http://www.zeland.com/fidelity.html [Accessed 11 Mar. 2006].

Appendices

Appendix A

Project Specification

University of Southern Queensland
Faculty of Engineering and Surveying

ENG 4111/2 Research Project
PROJECT SPECIFICATION

FOR:	Chan Auc Fai
TOPIC:	The Finite Difference Time Domain method for Computational Electromagnetics
SUPERVISOR:	Dr Jim Ball
SPONSORSHIP:	Faculty of Engineering and Surveying
PROJECT AIM:	The Finite Difference Time Domain method is an extension of the well-known finite difference method to time varying situations. It is one of the most popular methods for the analysis of electromagnetic fields at radio and microwave frequencies. The objective of this project is to adapt available software to the solution of simple problems, and create example programs which may be useful for teaching purposes.

PROGRAMME: Issue A, 24 March 2006

1. Literature review
2. Survey of available software, costs and free downloads
3. One-dimensional FDTD simulation of Absorbing Boundary Conditions
4. Modify fdtd2D to provide a 2D model of radiation from a narrow slit excited by a plane wave which is a bandpass pulse
5. Modify fdtd2D into a parallel plate model of a TDR probe excited by a DC pulsed plane wave having a raised cosine time waveform
6. Write first draft of dissertation and provide to supervisor for comment and advice
7. Complete final draft of dissertation

As time permits

8. Modify existing code to provide a 3D model of a probe excited cavity

AGREED: __________ (Student) __________ (Supervisor) ____/____/____

Appendix B

One-dimensional FDTD simulation of ABC Source Code

```
%Filename: fdtd1d_main.m
%Date of creation: November 2006
%Main menu for showing the FDTD demonstration in 1D

clear all
close all
pack
clc

pink1 = [1 0.4 0.6];

hd_fig = figure('Units','normalized',...
            'Color',[0.6 1 0.4],...
            'Name','FDTD one dimensional demonstration',...
            'NumberTitle','off',...
   'Units','normalized',...
            'Position',[0.005 0.015 0.99 0.95],...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'DoubleBuffer','off',...
   'WindowStyle','modal');

axis off

hd_txt1 = text('Units','normalized',...
            'Position',[0.015 1],...
            'Color',pink1,...
            'String',['FDTD one dimensional demonstration'],...
            'Fontsize',25,...
            'Fontweight','normal',...
            'Fontangle','italic',...
            'VerticalAlignment','baseline',...
            'HorizontalAlignment','left');


hd_ctrl_btn1 = uicontrol('Parent',hd_fig,...
                        'Units','normalized',...
                        'Position',[0.3 0.7 0.4 0.1],...
                        'String','Introduction',...
                        'Style','pushbutton',...
                        'Callback','close;fdtd1d_in');

hd_ctrl_btn2 = uicontrol('Parent',hd_fig,...
                        'Units','normalized',...
                        'Position',[0.3 0.6 0.4 0.1],...
                        'String','Sine wave demonstration',...
                        'Style','pushbutton',...
                        'Callback','close;fdtd1d_sin_menu');

hd_ctrl_btn3 = uicontrol('Parent',hd_fig,...
                        'Units','normalized',...
                        'Position',[0.3 0.5 0.4 0.1],...
                        'String','Half-sine pulse demonstration',...
                        'Style','pushbutton',...
                        'Callback','close;fdtd1d_half_sin_menu');

hd_ctrl_btn4 = uicontrol('Parent',hd_fig,...
                        'Units','normalized',...
                        'Position',[0.3 0.4 0.4 0.1],...
                        'String','Gaussian pulse demonstration',...
                        'Style','pushbutton',...
                        'Callback','close;fdtd1d_gauss_menu');

hd_ctrl_btn5 = uicontrol('Parent',hd_fig,...
                        'Units','normalized',...
                        'Position',[0.3 0.1 0.4 0.1],...
                        'String','Quit',...
                        'Style','pushbutton',...
                        'Callback','close');
```

```
%Filename: fdtd1d_in.m
%Date of creation: November 2006
%Show a brief introduction of FDTD

clear all
close all
pack
clc

pink1 = [1 0.4 0.6];
y_spacing = 0.04;
x_margin = -0.15;

hd_fig = figure('Units','normalized',...
            'Color',[0 0 0],...
            'Name','What is FDTD?',...
            'NumberTitle','off',...
  'Units','normalized',...
            'Position',[0.005 0.015 0.99 0.95],...
  'MenuBar','none',...
  'ToolBar','none',...
  'Resize','off',...
  'DoubleBuffer','off',...
  'DeleteFcn','fdtd1d_main',...
  'WindowStyle','normal');

axis off

hd_txt1 = text('Units','normalized',...
            'Position',[x_margin 1.05],...
            'Color',pink1,...
            'String',['What is FDTD?'],...
            'Fontsize',15,...
            'Fontweight','normal',...
            'Fontangle','italic',...
            'VerticalAlignment','baseline',...
            'HorizontalAlignment','left');

text1 = ['FDTD is Fintie-Difference Time-Domain. It is a numerical technique for solving';
     'Maxwell''s equations directly in the time domain on a space grid. The technique';
     'was based on the Yee algorithum introduced by K.S. Yee in 1966.              '];

hd_txt2 = imtext1(x_margin,0.96,text1,y_spacing,'left',...
            15,'italic','normal',pink1);

text2 = ['This technique is a fully explict computation. The finite-difference expressions for';
     'the time derivatives are central-difference in nature and second-order accurate.   ';
     'In the Yee algorithum, the elctromagnetic field components are arranged in         ';
     'staggered, uncollocated arrangement in a Cartesian coordinate system.              '];

hd_txt3 = imtext1(x_margin,0.78,text2,y_spacing,'left',...
            15,'italic','normal',pink1);

text3 = ['In this one dimensional FDTD demonstration, the source frequency is set to 1GHz.       ';
     'Nonpermeable media is choosen. In this case, the permeability is the free-space        ';
     'permeability and the equivalent magnetic loss is zero. It is assumed that there is     ';
     'no independent sources of electric field and magnetic field energy. The user can       ';
     'set the electric conductivity and the duration of the demonstration. The grid resolution';
     '(dx = 1.5 cm) is chosen to provide 20 samples per wavelength. The Courant factor       ';
     'S=c*dt/dx is set to the stability limit: S=1. This is the "magic time step."          '];

hd_txt4 = imtext1(x_margin,0.56,text3,y_spacing,'left',...
            15,'italic','normal',pink1);

text4 = ['This source code is the modification of Dr. Susan C. Hagness''s program';
     'Computational Electrodynamics, Second edition, Artech House, London.  '];

hd_txt5 = imtext1(x_margin,0.21,text4,y_spacing,'left',...
            15,'italic','normal',pink1);

%Back button

hd_ctrl_btn1 = uicontrol('Parent',hd_fig,...
                    'Units','normalized',...
                    'Position',[0.3 0.1 0.35 0.05],...
                    'String','Back',...
                    'Style','pushbutton',...
                    'Callback','close;fdtd1d_main');
```

```
%Filename: fdtd1d_sin_menu.m
%Date of creation: November 2006
%Menu for showing the sine wave using FDTD

clear all
close all
pack
clc

%************************************************************************
%    Input dialog box setting
%************************************************************************

%Note: The output of inputdlg1 gives a cell array.
prompt_strings1 = {'Electric conductivity, \sigma (in mS/m)',...
   'Duration of demonstration (in ns)'};
inputdlg_dialog_title1 = 'Demonstration input';
default_inputs1 = {'',''};
num_of_ans1 = 2;

prompt_strings2 = {'File name'};
inputdlg_dialog_title2 = 'File name input';
default_inputs2 = {''};
num_of_ans2 = 1;

number_of_lines_for_each_answer = 1;
answer_cell_array = {};
answer_numeric_array = [];
answer_char_array = '';

inputdlg_options.Resize = 'off';
inputdlg_options.WindowStyle = 'modal';
inputdlg_options.Interpreter = 'tex';

%************************************************************************
%    Message dialog box setting
%************************************************************************

info_msg_dialog_title = 'Demonstration saved';
info_msg_icon = 'help';

err_msg_dialog_title = 'Input error';
err_msg_icon = 'error';

msgbox_options.WindowStyle = 'modal';
msgbox_options.Interpreter = 'tex';

sigma_err_msg = 'Electric conductivity, \sigma cannot be negative.';
end_time_err_msg = 'Duration of demonstration cannot be less than zero.';
empty_err_msg = 'There are some blanks or illegal values in inputs.';

err_msg_true = true(1);

info_msg = 'The demonstration is saved as ';

%************************************************************************
%    Question dialog box setting
%************************************************************************

%Note: questdlg gives empty char array when the user click
%      the 'X' button on the top right hand corner
quest_dialog_title = 'FDTD 1D demonstration, Sine wave';

yes_button = 'Yes';
no_button = 'No';
default_button = 'Yes';
quest_msg1_answer = default_button;
quest_msg2_answer = default_button;
quest_msg3_answer = default_button;
quest_msg4_answer = no_button;
quest_msg5_answer = no_button;

quest_msg1 = 'Do you wish to enter the values again?';
quest_msg2 = 'Do you wish to replay the demonstration again?';
quest_msg3 = 'Do you wish to save the demonstration in AVI format?';
```

```
quest_msg4 = 'File name ';
quest_msg5 = 'Do you wish to exit the sub-menu?';

%**********************************************************************
%     Main program
%**********************************************************************

pink1 = [1 0.4 0.6];

hd_fig = figure('Units','normalized',...
             'Color',[0.6 1 0.4],...
             'Name','FDTD one dimensional demonstration',...
             'NumberTitle','off',...
   'Units','normalized',...
             'Position',[0.005 0.015 0.99 0.95],...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'DoubleBuffer','off',...
   'WindowStyle','normal');

hd_axis = gca;
axis off

hd_txt1 = text('Units','normalized',...
             'Position',[-0.16 1],...
             'Color',pink1,...
             'String',['FDTD one dimensional demonstration using sine wave'],...
             'Fontsize',25,...
             'Fontweight','normal',...
             'Fontangle','italic',...
             'VerticalAlignment','baseline',...
             'HorizontalAlignment','left');

while (strcmp(quest_msg5_answer,no_button)),
   %Reset the message box answers if the user
   %does not wish to exit the sub_menu
   quest_msg1_answer = default_button;
   quest_msg2_answer = default_button;
   quest_msg3_answer = default_button;
   quest_msg4_answer = no_button;
   quest_msg5_answer = no_button;
   err_msg_true = true(1);

   %If the input values are illegal or there is no input,
   %and if the user wishes to re-enter the values, the sub-menu
   %will remain acive.
   while (strcmp(quest_msg1_answer,default_button)&&(err_msg_true)),
      %If the user presses 'Cancel', or the sub-menu
      %is terminated, the main menu is invoked.
      %If the user does not enter any input, the inputdlg1
      %will return a empty cell array.
      while ((~(strcmp(answer_char_array,'Cancel')))&&...
            ((isempty(answer_numeric_array)))),
         answer_cell_array = inputdlg1(prompt_strings1,inputdlg_dialog_title1,...
            number_of_lines_for_each_answer,default_inputs1,inputdlg_options);

         answer_char_array = char(answer_cell_array);

         answer_numeric_array = str2num(answer_char_array);

         if (strcmp(answer_char_array,'Cancel')),
            fdtd1d_main
            return
         end

         if (isempty(answer_numeric_array)),
            answer_char_array = 'Cancel';
         end
      end

      %If the user does not enter all the inputs, error
      %message is given. Otherwise, we will check
      %for illegal inputs.
      if ((length(answer_numeric_array) ~= num_of_ans1)),
         h_msgbox = msgbox(empty_err_msg,err_msg_dialog_title,...
            err_msg_icon,msgbox_options);
         uiwait(h_msgbox)
      else
         %This part is excuted only when user
         %enters all inputs
         sigma = answer_numeric_array(1)*(1e-3);
         end_time = answer_numeric_array(2)*(1e-9);

         %If all the inputs are reasonable, movie is shown.
         %Otherwise, error message is given.
         if ((sigma >= 0)&&(end_time > 0)),
            err_msg_true = (~err_msg_true);

            hd_txt2 = text('Parent',hd_axis,...
            'Units','normalized',...
            'Position',[0.28 0.2],...
            'Color',pink1,...
            'String',['Please do not move the screen during display'],...
            'Fontsize',10,...
            'Fontweight','normal',...
            'Fontangle','italic',...
            'VerticalAlignment','baseline',...
            'HorizontalAlignment','left');

            [h1,M,replay_counter,...
               frames_per_second,rect] = fdtd1d_sin(sigma,end_time);
            delete(hd_txt2)
         else
            if sigma < 0,
               h_msgbox = msgbox(sigma_err_msg,err_msg_dialog_title,...
                  err_msg_icon,msgbox_options);
               uiwait(h_msgbox)
            end

            if end_time <= 0,
               h_msgbox = msgbox(end_time_err_msg,err_msg_dialog_title,...
                  err_msg_icon,msgbox_options);
               uiwait(h_msgbox)
            end
         end
      end

      %This part is excuted only when there is no input
      %or the inputs are illegal
      if (err_msg_true),
         %Ask user if user wishes to re-enter values
         quest_msg1_answer = questdlg(quest_msg1,quest_dialog_title,...
            yes_button,no_button,default_button);
      end

      %If user does not wish to re-enter values,
      %main menu is invoked. Otherwise, reset
      %some values
      if (strcmp(quest_msg1_answer,no_button)||...
            (isempty(quest_msg1_answer))),
         fdtd1d_main
         return
      else
         %Reset values
         answer_cell_array = {};
         answer_numeric_array = [];
         answer_char_array = '';
      end
   end

   %Ask the user if he wishes to replay the movie
   while (strcmp(quest_msg2_answer,default_button)),
      movie(h1,M,replay_counter,frames_per_second,rect)
      quest_msg2_answer = questdlg(quest_msg2,quest_dialog_title,...
         yes_button,no_button,default_button);
      %The pause for half a second is needed for slow-response
      %graphic card. Otherwise the question dialog box does not
      %have enough time to disappear.
      pause(0.5)
```

```
end

%Ask the user if he wishes to save the demonstration
quest_msg3_answer = questdlg(quest_msg3,quest_dialog_title,...
  yes_button,no_button,default_button);

%If the user wishes to save the demonstration,
%ask the user for file name.
if (strcmp(quest_msg3_answer,default_button)),
  directory = pwd;
  %If the file name already exists, ask the user
  %if he wishes to overwrite the file.
  while (strcmp(quest_msg4_answer,no_button)),
    filename_cell_array = inputdlg1(prompt_strings2,inputdlg_dialog_title2,...
      number_of_lines_for_each_answer,default_inputs2,inputdlg_options);

    filename_char_array = char(filename_cell_array);

    %If the user clicks the 'X' button on the top right corner,
    %file name is set to blank
    if (strcmp(filename_char_array,'Cancel')),
      filename_char_array = '';
    end

    %If no file name is given, default file name is used
    if isempty(filename_char_array),
      filename_char_array = 'fdtd1d_sin_demo';
      quest_msg4_answer = yes_button;
    else
      %If the user insists to overwrite the file,
      %the old file is deleted to prevent
      %file creation error
      if (exist([filename_char_array,'.avi'])),
        quest_msg4 = [quest_msg4,filename_char_array,...
          '.avi already exists. Do you wish to overwrite?'];
        quest_msg4_answer = questdlg(quest_msg4,quest_dialog_title,...
          yes_button,no_button,default_button);
        quest_msg4 = 'File name ';
        if (strcmp(quest_msg4_answer,yes_button)),
          recycle on%Move deleted files to Recycle folder in Window
          delete([filename_char_array,'.avi'])
        end
      else
        quest_msg4_answer = yes_button;
      end
    end
  end

  %Change the pointer to  watch-shape
  %during the process of movie2avi
  set(hd_fig,'Pointer','watch')
  set(h1,'Pointer','watch')
  %The delay is needed to show
  %the change of pointer shape
  pause(0.5)
  movie2avi(M,filename_char_array)

  %Show the user the file name after saving the demonstration
  info_msg = [info_msg,filename_char_array,'.avi in ',directory];

  %Change the pointer back to arrow-shape
  %after the process of movie2avi
  set(hd_fig,'Pointer','arrow')
  set(h1,'Pointer','arrow')
  %The delay is needed to show
  %the change of pointer shape
  pause(0.5)
  h_msgbox = msgbox(info_msg,info_msg_dialog_title,...
    info_msg_icon,msgbox_options);
  uiwait(h_msgbox)
end

delete(h1)%Close the figure showing the movie
info_msg = 'The demonstration is saved as ';%Reset message
```

```
  %Ask the user if he wishes to exit the sub-menu
  quest_msg5_answer = questdlg(quest_msg5,quest_dialog_title,...
     yes_button,no_button,default_button);
end

fdtd1d_main
```

```
function [h1,M,replay_counter,...
   frames_per_second,rect] = fdtd1d_sin(sigma,end_time)

%FDTD1D_SIN Create a FDTD-movie ouput in one dimension
%          by using sin() function. Nonpermeable media
%          is choosen. It is assumed that there is
%          no independent sources of electric field
%          and magnetic field energy. The Courant factor
%          is set to 1.
%
%The predefined sine wave parameter:
%source_freq = 1GHz
%pulse_max_amplitude = 1
%
%Note : The figure showing the movie, can only be
%      closed by delete command in MATLAB
%
%sigma = Electric conductivity, in Simen/meter
%end_time = The end time of the movie, in second
%h1 = Figure handle of the figure showing the movie
%M = Movie frames
%replay_counter = Number of replays of the movie
%frames_per_second = Frames per second of the movie
%rect = The position property of the figure showing the
%      movie and it is relative to the lower-left corner
%      of the figure handle, h1

%Adapted from fdtd1D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%             Department of Electrical and Computer Engineering
%             University of Wisconsin-Madison
%             1415 Engineering Drive
%             Madison, WI 53706-1691
%             608-265-5739
%             hagness@engr.wisc.edu
%
% Date of this version:  February 2000

%Check the number of input arguments
if nargin < 2,
   error('The number of input arguments is less than 2.');
end

%Check the number of output arguments
if nargout < 5,
   error('The number of input arguments is less than 5.');
end

%***********************************************************************
%     Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 1e9;%source_frequency of source excitation
lambda = light_velocity_freespace/source_freq;%wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

i_end = 200;%number of grid cells in x-direction

ib = i_end+1;

dx = lambda/20;%space increment of 1-D lattice
dt = dx/light_velocity_freespace;%time step
source_omegadt = source_omega*dt;

nmax = round(end_time/dt);%total number of time steps

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

```
%     Material parameters
%***********************************************************************

epsilon_r = 1;
epsilon = epsilon_0*epsilon_r;

%***********************************************************************
%     Updating coefficients for space region with nonpermeable media
%***********************************************************************

scaling_factor = dt/(mu_0*dx);

ca = (1-(dt*sigma)/(2*epsilon))/(1+(dt*sigma)/(2*epsilon));
cb = (dt/(epsilon*dx))/(1+(dt*sigma)/(2*epsilon));
cb_scaled = scaling_factor*cb;

%***********************************************************************
%     Field arrays
%***********************************************************************

e_scaled = zeros(1,ib);
h = zeros(1,i_end);

%***********************************************************************
%     Movie initialization
%***********************************************************************

x = linspace(dx,i_end*dx,i_end);
frames_per_second = 15;
replay_counter = 0;
%Screen size cannot be too large,
%otherwise movie cannot be played properly
screen_size = [195 190 420 190];
%screen_size = [1 29 800 504];

sigma_text = num2str(sigma/(1e-3));
title_text = ['Sine wave, 1GHz, sigma = ',sigma_text, ' mS/m'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 1D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal',...
   'CloseRequestFcn','disp("")');

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:nmax,

%***********************************************************************
%     Update electric fields
%***********************************************************************

e = sin(source_omegadt*n);
e_scaled(1) = scaling_factor*e;

rbc = e_scaled(i_end);
e_scaled(2:i_end) = ca*e_scaled(2:i_end)+...
   cb_scaled*(h(2:i_end)-h(1:i_end-1));
e_scaled(ib) = rbc;

%***********************************************************************
%     Update magnetic fields
%***********************************************************************
```

```
h(1:i_end) = h(1:i_end)+e_scaled(2:ib)-e_scaled(1:i_end);

%***********************************************************************
%     Visualize fields
%***********************************************************************

%Computer cannot show correct movie display
%if all frames are stored to variable M

%Therefore, it is advisable to store
%only some of the frames into variable M
%The lowest value is 1.
%When the value is 1, no frames are skipped
number_of_skip_frame = 2;
if mod(n,number_of_skip_frame) == 0,

   time_label = num2str(round(n*dt/1e-9));

   subplot(2,1,1),plot(x,e_scaled(1:i_end)/scaling_factor,'r')
   axis([0 3 -1 1])
   title(['time = ',time_label,' ns'])
   ylabel('E_Z')

   subplot(2,1,2),plot(x,h,'b')
   axis([0 3 -3e-3 3e-3])
   title(['time = ',time_label,' ns'])
   ylabel('H_Y')
   xlabel('x (meters)')

   rect = get(h1,'Position');
   rect(1:2) = [0 0];%change the position of the frame
                  %with respect to the current figure window

   M(n/number_of_skip_frame) = getframe(h1,rect);

end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end
```

```
%Filename: fdtd1d_half_sin_menu.m
%Date of creation: November 2006
%Menu for showing the half-sine pulse using FDTD in 1D

clear all
close all
pack
clc

%***********************************************************************
%     Input dialog box setting
%***********************************************************************

%Note: The output of inputdlg1 gives a cell array.
prompt_strings1 = {'Electric conductivity, \sigma (in mS/m)',...
   'Duration of demonstration (in ns)'};
inputdlg_dialog_title1 = 'Demonstration input';
default_inputs1 = {'',''};
num_of_ans1 = 2;

prompt_strings2 = {'File name'};
inputdlg_dialog_title2 = 'File name input';
default_inputs2 = {''};
num_of_ans2 = 1;

number_of_lines_for_each_answer = 1;
answer_cell_array = {};
answer_numeric_array = [];
answer_char_array = '';

inputdlg_options.Resize = 'off';
inputdlg_options.WindowStyle = 'modal';
inputdlg_options.Interpreter = 'tex';

%***********************************************************************
%     Message dialog box setting
%***********************************************************************

info_msg_dialog_title = 'Demonstration saved';
info_msg_icon = 'help';

err_msg_dialog_title = 'Input error';
err_msg_icon = 'error';

msgbox_options.WindowStyle = 'modal';
msgbox_options.Interpreter = 'tex';

sigma_err_msg = 'Electric conductivity, \sigma cannot be negative.';
end_time_err_msg = 'Duration of demonstration cannot be less than zero.';
empty_err_msg = 'There are some blanks or illegal values in inputs.';

err_msg_true = true(1);

info_msg = 'The demonstration is saved as ';

%***********************************************************************
%     Question dialog box setting
%***********************************************************************

%Note: questdlg gives empty char array when the user click
%      the 'X' button on the top right hand corner
quest_dialog_title = 'FDTD 1D demonstration, Half-sine pulse';

yes_button = 'Yes';
no_button = 'No';
default_button = 'Yes';
quest_msg1_answer = default_button;
quest_msg2_answer = default_button;
quest_msg3_answer = default_button;
quest_msg4_answer = no_button;
quest_msg5_answer = no_button;

quest_msg1 = 'Do you wish to enter the values again?';
quest_msg2 = 'Do you wish to replay the demonstration again?';
quest_msg3 = 'Do you wish to save the demonstration in AVI format?';
```

```
quest_msg4 = 'File name ';
quest_msg5 = 'Do you wish to exit the sub-menu?';

%**********************************************************************
%     Main program
%**********************************************************************

pink1 = [1 0.4 0.6];

hd_fig = figure('Units','normalized',...
              'Color',[0.6 1 0.4],...
              'Name','FDTD one dimensional demonstration',...
              'NumberTitle','off',...
   'Units','normalized',...
              'Position',[0.005 0.015 0.99 0.95],...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'DoubleBuffer','off',...
   'WindowStyle','normal');

hd_axis = gca;
axis off

hd_txt1 = text('Units','normalized',...
              'Position',[-0.15 1],...
              'Color',pink1,...
              'String',['FDTD one dimensional demonstration using half-sine pulse'],...
              'Fontsize',22,...
              'Fontweight','normal',...
              'Fontangle','italic',...
              'VerticalAlignment','baseline',...
              'HorizontalAlignment','left');

while (strcmp(quest_msg5_answer,no_button)),
   %Reset the message box answers if the user
   %does not wish to exit the sub_menu
   quest_msg1_answer = default_button;
   quest_msg2_answer = default_button;
   quest_msg3_answer = default_button;
   quest_msg4_answer = no_button;
   quest_msg5_answer = no_button;
   err_msg_true = true(1);

   %If the input values are illegal or there is no input,
   %and if the user wishes to re-enter the values, the sub-menu
   %will remain acive.
   while (strcmp(quest_msg1_answer,default_button)&&(err_msg_true)),
      %If the user presses 'Cancel', or the sub-menu
      %is terminated, the main menu is invoked.
      %If the user does not enter any input, the inputdlg1
      %will return a empty cell array.
      while ((~(strcmp(answer_char_array,'Cancel')))&&...
            ((isempty(answer_numeric_array)))),
         answer_cell_array = inputdlg1(prompt_strings1,inputdlg_dialog_title1,...
            number_of_lines_for_each_answer,default_inputs1,inputdlg_options);

         answer_char_array = char(answer_cell_array);

         answer_numeric_array = str2num(answer_char_array);

         if (strcmp(answer_char_array,'Cancel')),
            fdtd1d_main
            return
         end

         if (isempty(answer_numeric_array)),
            answer_char_array = 'Cancel';
         end
      end

      %If the user does not enter all the inputs, error
      %message is given. Otherwise, we will check
      %for illegal inputs.
      if ((length(answer_numeric_array) ~= num_of_ans1)),
         h_msgbox = msgbox(empty_err_msg,err_msg_dialog_title,...
            err_msg_icon,msgbox_options);
         uiwait(h_msgbox)
      else
         %This part is excuted only when user
         %enters all inputs
         sigma = answer_numeric_array(1)*(1e-3);
         end_time = answer_numeric_array(2)*(1e-9);

         %If all the inputs are reasonable, movie is shown.
         %Otherwise, error message is given.
         if ((sigma >= 0)&&(end_time > 0)),
            err_msg_true = (~err_msg_true);

            hd_txt2 = text('Parent',hd_axis,...
            'Units','normalized',...
            'Position',[0.28 0.2],...
            'Color',pink1,...
            'String',['Please do not move the screen during display'],...
            'Fontsize',10,...
            'Fontweight','normal',...
            'Fontangle','italic',...
            'VerticalAlignment','baseline',...
            'HorizontalAlignment','left');

            [h1,M,replay_counter,...
               frames_per_second,rect] = fdtd1d_half_sin(sigma,end_time);
            delete(hd_txt2)
         else
            if sigma < 0,
               h_msgbox = msgbox(sigma_err_msg,err_msg_dialog_title,...
                  err_msg_icon,msgbox_options);
               uiwait(h_msgbox)
            end

            if end_time <= 0,
               h_msgbox = msgbox(end_time_err_msg,err_msg_dialog_title,...
                  err_msg_icon,msgbox_options);
               uiwait(h_msgbox)
            end
         end
      end

      %This part is excuted only when there is no input
      %or the inputs are illegal
      if (err_msg_true),
         %Ask user if user wishes to re-enter values
         quest_msg1_answer = questdlg(quest_msg1,quest_dialog_title,...
            yes_button,no_button,default_button);
      end

      %If user does not wish to re-enter values,
      %main menu is invoked. Otherwise, reset
      %some values
      if (strcmp(quest_msg1_answer,no_button)||...
            (isempty(quest_msg1_answer))),
         fdtd1d_main
         return
      else
         %Reset values
         answer_cell_array = {};
         answer_numeric_array = [];
         answer_char_array = '';
      end
   end

   %Ask the user if he wishes to replay the movie
   while (strcmp(quest_msg2_answer,default_button)),
      movie(h1,M,replay_counter,frames_per_second,rect)
      quest_msg2_answer = questdlg(quest_msg2,quest_dialog_title,...
         yes_button,no_button,default_button);
      %The pause for half a second is needed for slow-response
      %graphic card. Otherwise the question dialog box does not
      %have enough time to disappear.
      pause(0.5)
```

```
end

%Ask the user if he wishes to save the demonstration
quest_msg3_answer = questdlg(quest_msg3,quest_dialog_title,...
  yes_button,no_button,default_button);

%If the user wishes to save the demonstration,
%ask the user for file name.
if (strcmp(quest_msg3_answer,default_button)),
  directory = pwd;
  %If the file name already exists, ask the user
  %if he wishes to overwrite the file.
  while (strcmp(quest_msg4_answer,no_button)),
    filename_cell_array = inputdlg1(prompt_strings2,inputdlg_dialog_title2,...
      number_of_lines_for_each_answer,default_inputs2,inputdlg_options);

    filename_char_array = char(filename_cell_array);

    %If the user clicks the 'X' button on the top right corner,
    %file name is set to blank
    if (strcmp(filename_char_array,'Cancel')),
      filename_char_array = '';
    end

    %If no file name is given, default file name is used
    if isempty(filename_char_array),
      filename_char_array = 'fdtd1d_half_sin_demo';
      quest_msg4_answer = yes_button;
    else
      %If the user insists to overwrite the file,
      %the old file is deleted to prevent
      %file creation error
      if (exist([filename_char_array,'.avi'])),
        quest_msg4 = [quest_msg4,filename_char_array,...
          '.avi already exists. Do you wish to overwrite?'];
        quest_msg4_answer = questdlg(quest_msg4,quest_dialog_title,...
          yes_button,no_button,default_button);
        quest_msg4 = 'File name ';
        if (strcmp(quest_msg4_answer,yes_button)),
          recycle on%Move deleted files to Recycle folder in Window
          delete([filename_char_array,'.avi'])
        end
      else
        quest_msg4_answer = yes_button;
      end
    end
  end

  %Change the pointer to  watch-shape
  %during the process of movie2avi
  set(hd_fig,'Pointer','watch')
  set(h1,'Pointer','watch')
  %The delay is needed to show
  %the change of pointer shape
  pause(0.5)
  movie2avi(M,filename_char_array)

  %Show the user the file name after saving the demonstration
  info_msg = [info_msg,filename_char_array,'.avi in ',directory];

  %Change the pointer back to arrow-shape
  %after the process of movie2avi
  set(hd_fig,'Pointer','arrow')
  set(h1,'Pointer','arrow')
  %The delay is needed to show
  %the change of pointer shape
  pause(0.5)
  h_msgbox = msgbox(info_msg,info_msg_dialog_title,...
    info_msg_icon,msgbox_options);
  uiwait(h_msgbox)
end

delete(h1)%Close the figure showing the movie
info_msg = 'The demonstration is saved as ';%Reset message
```

```
  %Ask the user if he wishes to exit the sub-menu
  quest_msg5_answer = questdlg(quest_msg5,quest_dialog_title,...
    yes_button,no_button,default_button);
end

fdtd1d_main
```

```
function [h1,M,replay_counter,...
   frames_per_second,rect] = fdtd1d_half_sin(sigma,end_time)

%FDTD1D_HALF_SIN Create a FDTD-movie ouput in one dimension
%              by using user-defined function,
%              half_sine_pulse(). Nonpermeable media
%              is choosen. It is assumed that there is
%              no independent sources of electric field
%              and magnetic field energy. The Courant factor
%              is set to 1.
%
%The predefined half-sine pulse parameter:
%source_freq = 1GHz
%pulse_max_amplitude = 1
%pulse_center_location = 40*dt
%
%Note : The figure showing the movie, can only be
%       closed by delete command in MATLAB
%
%sigma = Electric conductivity, in Simen/meter
%end_time = The end time of the movie, in second
%h1 = Figure handle of the figure showing the movie
%M = Movie frames
%replay_counter = Number of replays of the movie
%frames_per_second = Frames per second of the movie
%rect = The position property of the figure showing the
%       movie and it is relative to the lower-left corner
%       of the figure handle, h1

%Adapted from fdtd1D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%              Department of Electrical and Computer Engineering
%              University of Wisconsin-Madison
%              1415 Engineering Drive
%              Madison, WI 53706-1691
%              608-265-5739
%              hagness@engr.wisc.edu
%
% Date of this version:  February 2000

%Check the number of input arguments
if nargin < 2,
   error('The number of input arguments is less than 2.');
end

%Check the number of output arguments
if nargout < 5,
   error('The number of input arguments is less than 5.');
end

%***********************************************************************
%     Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 1e9;%source_frequency of source excitation
lambda = light_velocity_freespace/source_freq;%wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

i_end = 200;%number of grid cells in x-direction

ib = i_end+1;

dx = lambda/20;%space increment of 1-D lattice
dt = dx/light_velocity_freespace;%time step
source_omegadt = source_omega*dt;

nmax = round(end_time/dt);%total number of time steps
```

```
%***********************************************************************
%     Half-sine pulse parameter
%***********************************************************************

pulse_max_amplitude = 1;
pulse_center_location = 40*dt;

%***********************************************************************
%     Material parameters
%***********************************************************************

epsilon_r = 1;
epsilon = epsilon_0*epsilon_r;

%***********************************************************************
%     Updating coefficients for space region with nonpermeable media
%***********************************************************************

scaling_factor = dt/(mu_0*dx);

ca = (1-(dt*sigma)/(2*epsilon))/(1+(dt*sigma)/(2*epsilon));
cb = (dt/(epsilon*dx))/(1+(dt*sigma)/(2*epsilon));
cb_scaled = scaling_factor*cb;

%***********************************************************************
%     Field arrays
%***********************************************************************

e_scaled = zeros(1,ib);
h = zeros(1,i_end);

%***********************************************************************
%     Movie initialization
%***********************************************************************

x = linspace(dx,i_end*dx,i_end);
frames_per_second = 15;
replay_counter = 0;
%Screen size cannot be too large,
%otherwise movie cannot be played properly
screen_size = [195 190 420 190];
%screen_size = [1 29 800 504];

sigma_text = num2str(sigma/(1e-3));
title_text = ['Half-sine pulse, 1GHz, sigma = ',sigma_text, ' mS/m'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 1D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal',...
   'CloseRequestFcn','disp("")');

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:nmax,

%***********************************************************************
%     Update electric fields
%***********************************************************************

e = half_sine_pulse(n*dt,pulse_max_amplitude,...
   pulse_center_location,source_freq);
```

```
e_scaled(1) = scaling_factor*e;

rbc = e_scaled(i_end);
e_scaled(2:i_end) = ca*e_scaled(2:i_end)+...
   cb_scaled*(h(2:i_end)-h(1:i_end-1));
e_scaled(ib) = rbc;

%***********************************************************************
%     Update magnetic fields
%***********************************************************************

h(1:i_end) = h(1:i_end)+e_scaled(2:ib)-e_scaled(1:i_end);

%***********************************************************************
%     Visualize fields
%***********************************************************************

%Computer cannot show correct movie display
%if all frames are stored to variable M

%Therefore, it is advisable to store
%only some of the frames into variable M
%The lowest value is 1.
%When the value is 1, no frames are skipped
number_of_skip_frame = 2;
if mod(n,number_of_skip_frame) == 0,

   time_label = num2str(round(n*dt/1e-9));

   subplot(2,1,1),plot(x,e_scaled(1:i_end)/scaling_factor,'r')
   axis([0 3 -1 1])
   title(['time = ',time_label,' ns'])
   ylabel('E_Z')

   subplot(2,1,2),plot(x,h,'b')
   axis([0 3 -3e-3 3e-3])
   title(['time = ',time_label,' ns'])
   ylabel('H_Y')
   xlabel('x (meters)')

   rect = get(h1,'Position');
   rect(1:2) = [0 0];%change the position of the frame
                  %with respect to the current figure window

   M(n/number_of_skip_frame) = getframe(h1,rect);

end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end
```

```
function [y] = half_sine_pulse(x,pulse_max_amplitude,...
   pulse_center_location,source_freq)

%HALF_SINE_PULSE Create a half-sine pulse
%[y] = half_sine_pulse(x,pulse_max_amplitude,
%  pulse_center_location,source_freq)
%y = Half-sine pulse output, in second
%x = Half-sine pulse input range, in second
%pulse_max_amplitude = Maximum amplitude of Half-sine pulse
%pulse_center_location = The location of the center of
%                        half-sine pulse, in second
%source_freq = Frequency of one cycle of the sine wave, in Hz

if nargin ~= 4,
   error('The number of input arguments must be 4');
end

half_pulse_width = 1/(4*source_freq);
upper_limit = pulse_center_location + half_pulse_width;
lower_limit = pulse_center_location - half_pulse_width;
y = 0;

if ((x >= lower_limit)&(x <= upper_limit)),
   y = pulse_max_amplitude*sin((2*pi*source_freq*x)+(pi/2));
end
```

```
%Filename: fdtd1d_gauss_menu.m
%Date of creation: November 2006
%Menu for showing the Gaussian pulse using FDTD in 1D

clear all
close all
pack
clc

%***********************************************************************
%     Input dialog box setting
%***********************************************************************

%Note: The output of inputdlg1 gives a cell array.
prompt_strings1 = {'Electric conductivity, \sigma (in mS/m)',...
   'Duration of demonstration (in ns)'};
inputdlg_dialog_title1 = 'Demonstration input';
default_inputs1 = {'',''};
num_of_ans1 = 2;

prompt_strings2 = {'File name'};
inputdlg_dialog_title2 = 'File name input';
default_inputs2 = {''};
num_of_ans2 = 1;

number_of_lines_for_each_answer = 1;
answer_cell_array = {};
answer_numeric_array = [];
answer_char_array = '';

inputdlg_options.Resize = 'off';
inputdlg_options.WindowStyle = 'modal';
inputdlg_options.Interpreter = 'tex';

%***********************************************************************
%     Message dialog box setting
%***********************************************************************

info_msg_dialog_title = 'Demonstration saved';
info_msg_icon = 'help';

err_msg_dialog_title = 'Input error';
err_msg_icon = 'error';

msgbox_options.WindowStyle = 'modal';
msgbox_options.Interpreter = 'tex';

sigma_err_msg = 'Electric conductivity, \sigma cannot be negative.';
end_time_err_msg = 'Duration of demonstration cannot be less than zero.';
empty_err_msg = 'There are some blanks or illegal values in inputs.';

err_msg_true = true(1);

info_msg = 'The demonstration is saved as ';

%***********************************************************************
%     Question dialog box setting
%***********************************************************************

%Note: questdlg gives empty char array when the user click
%      the 'X' button on the top right hand corner
quest_dialog_title = 'FDTD 1D demonstration, Gaussian pulse';

yes_button = 'Yes';
no_button = 'No';
default_button = 'Yes';
quest_msg1_answer = default_button;
quest_msg2_answer = default_button;
quest_msg3_answer = default_button;
quest_msg4_answer = no_button;
quest_msg5_answer = no_button;

quest_msg1 = 'Do you wish to enter the values again?';
quest_msg2 = 'Do you wish to replay the demonstration again?';
quest_msg3 = 'Do you wish to save the demonstration in AVI format?';
```

```
quest_msg4 = 'File name ';
quest_msg5 = 'Do you wish to exit the sub-menu?';

%**********************************************************************
%     Main program
%**********************************************************************

pink1 = [1 0.4 0.6];

hd_fig = figure('Units','normalized',...
            'Color',[0.6 1 0.4],...
            'Name','FDTD one dimensional demonstration',...
            'NumberTitle','off',...
   'Units','normalized',...
            'Position',[0.005 0.015 0.99 0.95],...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'DoubleBuffer','off',...
   'WindowStyle','normal');

hd_axis = gca;
axis off

hd_txt1 = text('Units','normalized',...
            'Position',[-0.155 1],...
            'Color',pink1,...
            'String',['FDTD one dimensional demonstration using Gaussian pulse'],...
            'Fontsize',22,...
            'Fontweight','normal',...
            'Fontangle','italic',...
            'VerticalAlignment','baseline',...
            'HorizontalAlignment','left');

while (strcmp(quest_msg5_answer,no_button)),
   %Reset the message box answers if the user
   %does not wish to exit the sub_menu
   quest_msg1_answer = default_button;
   quest_msg2_answer = default_button;
   quest_msg3_answer = default_button;
   quest_msg4_answer = no_button;
   quest_msg5_answer = no_button;
   err_msg_true = true(1);

   %If the input values are illegal or there is no input,
   %and if the user wishes to re-enter the values, the sub-menu
   %will remain acive.
   while (strcmp(quest_msg1_answer,default_button)&&(err_msg_true)),
      %If the user presses 'Cancel', or the sub-menu
      %is terminated, the main menu is invoked.
      %If the user does not enter any input, the inputdlg1
      %will return a empty cell array.
      while ((~(strcmp(answer_char_array,'Cancel')))&&...
            ((isempty(answer_numeric_array)))),
         answer_cell_array = inputdlg1(prompt_strings1,inputdlg_dialog_title1,...
            number_of_lines_for_each_answer,default_inputs1,inputdlg_options);

         answer_char_array = char(answer_cell_array);

         answer_numeric_array = str2num(answer_char_array);

         if (strcmp(answer_char_array,'Cancel')),
            fdtd1d_main
            return
         end

         if (isempty(answer_numeric_array)),
            answer_char_array = 'Cancel';
         end
      end

      %If the user does not enter all the inputs, error
      %message is given. Otherwise, we will check
      %for illegal inputs.
      if ((length(answer_numeric_array) ~= num_of_ans1)),
         h_msgbox = msgbox(empty_err_msg,err_msg_dialog_title,...
            err_msg_icon,msgbox_options);
         uiwait(h_msgbox)
      else
         %This part is excuted only when user
         %enters all inputs
         sigma = answer_numeric_array(1)*(1e-3);
         end_time = answer_numeric_array(2)*(1e-9);

         %If all the inputs are reasonable, movie is shown.
         %Otherwise, error message is given.
         if ((sigma >= 0)&&(end_time > 0)),
            err_msg_true = (~err_msg_true);

            hd_txt2 = text('Parent',hd_axis,...
            'Units','normalized',...
            'Position',[0.28 0.2],...
            'Color',pink1,...
            'String',['Please do not move the screen during display'],...
            'Fontsize',10,...
            'Fontweight','normal',...
            'Fontangle','italic',...
            'VerticalAlignment','baseline',...
            'HorizontalAlignment','left');

            [h1,M,replay_counter,...
              frames_per_second,rect] = fdtd1d_gauss(sigma,end_time);
            delete(hd_txt2)
         else
            if sigma < 0,
              h_msgbox = msgbox(sigma_err_msg,err_msg_dialog_title,...
                err_msg_icon,msgbox_options);
              uiwait(h_msgbox)
            end

            if end_time <= 0,
              h_msgbox = msgbox(end_time_err_msg,err_msg_dialog_title,...
                err_msg_icon,msgbox_options);
              uiwait(h_msgbox)
            end
         end
      end

      %This part is excuted only when there is no input
      %or the inputs are illegal
      if (err_msg_true),
         %Ask user if user wishes to re-enter values
         quest_msg1_answer = questdlg(quest_msg1,quest_dialog_title,...
            yes_button,no_button,default_button);
      end

      %If user does not wish to re-enter values,
      %main menu is invoked. Otherwise, reset
      %some values
      if (strcmp(quest_msg1_answer,no_button)||...
            (isempty(quest_msg1_answer))),
         fdtd1d_main
         return
      else
         %Reset values
         answer_cell_array = {};
         answer_numeric_array = [];
         answer_char_array = '';
      end
   end

   %Ask the user if he wishes to replay the movie
   while (strcmp(quest_msg2_answer,default_button)),
      movie(h1,M,replay_counter,frames_per_second,rect)
      quest_msg2_answer = questdlg(quest_msg2,quest_dialog_title,...
         yes_button,no_button,default_button);
      %The pause for half a second is needed for slow-response
      %graphic card. Otherwise the question dialog box does not
      %have enough time to disappear.
      pause(0.5)
```

```
end

%Ask the user if he wishes to save the demonstration
quest_msg3_answer = questdlg(quest_msg3,quest_dialog_title,...
  yes_button,no_button,default_button);

%If the user wishes to save the demonstration,
%ask the user for file name.
if (strcmp(quest_msg3_answer,default_button)),
  directory = pwd;
  %If the file name already exists, ask the user
  %if he wishes to overwrite the file.
  while (strcmp(quest_msg4_answer,no_button)),
    filename_cell_array = inputdlg1(prompt_strings2,inputdlg_dialog_title2,...
      number_of_lines_for_each_answer,default_inputs2,inputdlg_options);

    filename_char_array = char(filename_cell_array);

    %If the user clicks the 'X' button on the top right corner,
    %file name is set to blank
    if (strcmp(filename_char_array,'Cancel')),
      filename_char_array = '';
    end

    %If no file name is given, default file name is used
    if isempty(filename_char_array),
      filename_char_array = 'fdtd1d_gauss_demo';
      quest_msg4_answer = yes_button;
    else
      %If the user insists to overwrite the file,
      %the old file is deleted to prevent
      %file creation error
      if (exist([filename_char_array,'.avi'])),
        quest_msg4 = [quest_msg4,filename_char_array,...
          '.avi already exists. Do you wish to overwrite?'];
        quest_msg4_answer = questdlg(quest_msg4,quest_dialog_title,...
          yes_button,no_button,default_button);
        quest_msg4 = 'File name ';
        if (strcmp(quest_msg4_answer,yes_button)),
          recycle on%Move deleted files to Recycle folder in Window
          delete([filename_char_array,'.avi'])
        end
      else
        quest_msg4_answer = yes_button;
      end
    end
  end

  %Change the pointer to  watch-shape
  %during the process of movie2avi
  set(hd_fig,'Pointer','watch')
  set(h1,'Pointer','watch')
  %The delay is needed to show
  %the change of pointer shape
  pause(0.5)
  movie2avi(M,filename_char_array)

  %Show the user the file name after saving the demonstration
  info_msg = [info_msg,filename_char_array,'.avi in ',directory];

  %Change the pointer back to arrow-shape
  %after the process of movie2avi
  set(hd_fig,'Pointer','arrow')
  set(h1,'Pointer','arrow')
  %The delay is needed to show
  %the change of pointer shape
  pause(0.5)
  h_msgbox = msgbox(info_msg,info_msg_dialog_title,...
    info_msg_icon,msgbox_options);
  uiwait(h_msgbox)
end

delete(h1)%Close the figure showing the movie
info_msg = 'The demonstration is saved as ';%Reset message
```

```
   %Ask the user if he wishes to exit the sub-menu
   quest_msg5_answer = questdlg(quest_msg5,quest_dialog_title,...
      yes_button,no_button,default_button);
end

fdtd1d_main
```

```
function [h1,M,replay_counter,...
   frames_per_second,rect] = fdtd1d_gauss(sigma,end_time)

%FDTD1D_GAUSS Create a FDTD-movie ouput in one dimension
%            by using user-defined function,
%            gaussian_pulse(). Nonpermeable media
%            is choosen. It is assumed that there is
%            no independent sources of electric field
%            and magnetic field energy. The Courant factor
%            is set to 1.
%
%The predefined Gaussian pulse parameter:
%pulse_max_amplitude = 1
%pulse_center_location = 40*dt
%half_pulse_width = 20*dt (at the point 1/e)
%
%Note : The figure showing the movie, can only be
%      closed by delete command in MATLAB
%
%sigma = Electric conductivity, in Simen/meter
%end_time = The end time of the movie, in second
%h1 = Figure handle of the figure showing the movie
%M = Movie frames
%replay_counter = Number of replays of the movie
%frames_per_second = Frames per second of the movie
%rect = The position property of the figure showing the
%      movie and it is relative to the lower-left corner
%      of the figure handle, h1

%Adapted from fdtd1D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%             Department of Electrical and Computer Engineering
%             University of Wisconsin-Madison
%             1415 Engineering Drive
%             Madison, WI 53706-1691
%             608-265-5739
%             hagness@engr.wisc.edu
%
% Date of this version:  February 2000

%Check the number of input arguments
if nargin < 2,
   error('The number of input arguments is less than 2.');
end

%Check the number of output arguments
if nargout < 5,
   error('The number of input arguments is less than 5.');
end

%***********************************************************************
%     Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 1e9;%source_frequency of source excitation
lambda = light_velocity_freespace/source_freq;%wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

i_end = 200;%number of grid cells in x-direction

ib = i_end+1;

dx = lambda/20;%space increment of 1-D lattice
dt = dx/light_velocity_freespace;%time step
source_omegadt = source_omega*dt;

nmax = round(end_time/dt);%total number of time steps
```

```
%***********************************************************************
%     Gaussian pulse parameter
%***********************************************************************

pulse_max_amplitude = 1;
pulse_center_location = 40*dt;
half_pulse_width = 20*dt;

%***********************************************************************
%     Material parameters
%***********************************************************************

epsilon_r = 1;
epsilon = epsilon_0*epsilon_r;

%***********************************************************************
%     Updating coefficients for space region with nonpermeable media
%***********************************************************************

scaling_factor = dt/(mu_0*dx);

ca = (1-(dt*sigma)/(2*epsilon))/(1+(dt*sigma)/(2*epsilon));
cb = (dt/(epsilon*dx))/(1+(dt*sigma)/(2*epsilon));
cb_scaled = scaling_factor*cb;

%***********************************************************************
%     Field arrays
%***********************************************************************

e_scaled = zeros(1,ib);
h = zeros(1,i_end);

%***********************************************************************
%     Movie initialization
%***********************************************************************

x = linspace(dx,i_end*dx,i_end);
frames_per_second = 15;
replay_counter = 0;
%Screen size cannot be too large,
%otherwise movie cannot be played properly
screen_size = [195 190 420 190];
%screen_size = [1 29 800 504];

sigma_text = num2str(sigma/(1e-3));
title_text = ['Gaussian pulse, sigma = ',sigma_text, ' mS/m'];

%Property 'CloseRequestFcn', is set to 'disp('''')'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 1D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal',...
   'CloseRequestFcn','disp('''')');

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:nmax,

%***********************************************************************
%     Update electric fields
%***********************************************************************

e = gaussian_pulse(n*dt,pulse_max_amplitude,...
  pulse_center_location,half_pulse_width);
```

```
e_scaled(1) = scaling_factor*e;

rbc = e_scaled(i_end);
e_scaled(2:i_end) = ca*e_scaled(2:i_end)+...
   cb_scaled*(h(2:i_end)-h(1:i_end-1));
e_scaled(ib) = rbc;

%***********************************************************************
%     Update magnetic fields
%***********************************************************************

h(1:i_end) = h(1:i_end)+e_scaled(2:ib)-e_scaled(1:i_end);

%***********************************************************************
%     Visualize fields
%***********************************************************************

%Computer cannot show correct movie display
%if all frames are stored to variable M

%Therefore, it is advisable to store
%only some of the frames into variable M
%The lowest value is 1.
%When the value is 1, no frames are skipped
number_of_skip_frame = 2;
if mod(n,number_of_skip_frame) == 0,

   time_label = num2str(round(n*dt/1e-9));

   subplot(2,1,1),plot(x,e_scaled(1:i_end)/scaling_factor,'r')
   axis([0 3 -1 1])
   title(['time = ',time_label,' ns'])
   ylabel('E_Z')

   subplot(2,1,2),plot(x,h,'b')
   axis([0 3 -3e-3 3e-3])
   title(['time = ',time_label,' ns'])
   ylabel('H_Y')
   xlabel('x (meters)')

   rect = get(h1,'Position');
   rect(1:2) = [0 0];%change the position of the frame
                  %with respect to the current figure window

   M(n/number_of_skip_frame) = getframe(h1,rect);

end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end
```

```
function [y] = gaussian_pulse(x,pulse_max_amplitude,...
   pulse_center_location,half_pulse_width)

%GAUSSIAN_PULSE Create a Gaussian pulse
%[y] = gaussian_pulse(x,pulse_max_amplitude,
%  pulse_center_location,half_pulse_width)
%y = Gaussian pulse output
%x = Gaussian pulse input range
%pulse_max_amplitude = Maximum amplitude of Gaussian pulse
%pulse_center_location = The location of the center of
%                        Gaussian pulse
%half_pulse_width = The width of half of Gaussian pulse
%                   at the point 1/e

if nargin ~= 4,
   error('The number of input arguments must be 4');
end

y = pulse_max_amplitude*exp(-(((x-pulse_center_location)...
   /(half_pulse_width)).^2));
```

```
function H = imtext1(x,y,T,dely,justify,fontsize,fontangle,fontweight,color)
%IMTEXT1 Place possibly multi-line text as xlabel.
%
%       IMTEXT1(x,y,T) writes T, left justified, with the lower left
%       corner at axis normalized coordinates (x,y).
%
%       IMTEXT1(x,y,T) uses white for color, fontsize of 12,dely of 0.04
%       normal for fontangle, and normal for fontweight.
%
%       fontangle can be 'normal' 'italic' or 'oblique'.
%       fontweight can be 'light' 'normal' 'demi' or 'bold'.
%       dely is the spacing between each sentences.
%
%       IMTEXT1 uses 'baseline' for verticalalignment and 'normalized'
%       for 'units'.
%
%       IMTEXT1(x,y,T,'center') and IMTEXT1(x,y,T,'right') are also possible.
%       H = IMTEXT1(...) returns a vector of handles to the lines of text.
%
%       IMTEXT1 is modified from IMTEXT.
%       IMTEXT is a copyright product of The MathWorks, Inc.
%
%       Auc Fai. Chan, 2-13-2000
%
%Example T=['I am driving to A from B';
%          'It takes several hours  '];
%        a=imtext1(0.5,0.5,T,0.3,'left',15,'italic','bold',[1 0 0]);


if nargin < 9, justify = 'left';
                dely = 0.04;
                fontsize = 12;
                color = [1 1 1];
                fontangle = 'normal';
                fontweight = 'normal';
end
ax = gca;
[m,n] = size(T);
for k = 1:m
   h(k) = text(x,y,T(k,:),...
                          'Units','normalized',...
                          'HorizontalAlignment',justify,...
                          'Fontsize',fontsize,...
                          'Fontangle',fontangle,...
                          'Fontweight',fontweight,...
                          'Color',color,...
                          'VerticalAlignment','baseline');
   y = y - dely;
end
grid on
if nargout > 0
   H = h';
end
```

```
function Answer=inputdlg1(Prompt, Title, NumLines, DefAns, Resize)
%INPUTDLG Input dialog box.
%  ANSWER = INPUTDLG1(PROMPT) creates a modal dialog box that returns user
%  input for multiple prompts in the cell array ANSWER. PROMPT is a cell
%  array containing the PROMPT strings.
%
%  If the user press the 'Cancel' button, ANSWER = {'Cancel'}
%
%  INPUTDLG1 uses UIWAIT to suspend execution until the user responds.
%
%  ANSWER = INPUTDLG1(PROMPT,NAME) specifies the title for the dialog.
%
%  ANSWER = INPUTDLG1(PROMPT,NAME,NUMLINES) specifies the number of lines for
%  each answer in NUMLINES. NUMLINES may be a constant value or a column
%  vector having one element per PROMPT that specifies how many lines per
%  input field. NUMLINES may also be a matrix where the first column
%  specifies how many rows for the input field and the second column
%  specifies how many columns wide the input field should be.
%
%  ANSWER = INPUTDLG1(PROMPT,NAME,NUMLINES,DEFAULTANSWER) specifies the
%  default answer to display for each PROMPT. DEFAULTANSWER must contain
%  the same number of elements as PROMPT and must be a cell array of
%  strings.
%
%  ANSWER = INPUTDLG1(PROMPT,NAME,NUMLINES,DEFAULTANSWER,OPTIONS) specifies
%  additional options. If OPTIONS is the string 'on', the dialog is made
%  resizable. If OPTIONS is a structure, the fields Resize, WindowStyle, and
%  Interpreter are recognized. Resize can be either 'on' or
%  'off'. WindowStyle can be either 'normal' or 'modal'. Interpreter can be
%  either 'none' or 'tex'. If Interpreter is 'tex', the prompt strings are
%  rendered using LaTeX.
%
%  Examples:
%
%  prompt={'Enter the matrix size for x^2:','Enter the colormap name:'};
%  name='Input for Peaks function';
%  numlines=1;
%  defaultanswer={'20','hsv'};
%
%  answer=inputdlg1(prompt,name,numlines,defaultanswer);
%
%  options.Resize='on';
%  options.WindowStyle='normal';
%  options.Interpreter='tex';
%
%  answer=inputdlg1(prompt,name,numlines,defaultanswer,options);
%
%  See also TEXTWRAP, QUESTDLG, UIWAIT.

%  INPUTDLG1 is modified from INPUTDLG.
%  INPUTDLG is a copyright product of The MathWorks, Inc.
%  Auc Fai. Chan, 4-24-2006

%%%%%%%%%%%%%%%%%%%%
%%% Nargin Check %%%
%%%%%%%%%%%%%%%%%%%%
error(nargchk(1,5,nargin));
error(nargoutchk(1,1,nargout));

if nargin==1,
   Title=' ';
end

if nargin<=2
   NumLines=1;
end

if ~iscell(Prompt)
   Prompt={Prompt};
end

NumQuest=prod(size(Prompt));

if nargin<=3
   DefAns=cell(NumQuest,1);
```

```
  for lp=1:NumQuest
    DefAns{lp}='';
  end
end

WindowStyle='modal';
Interpreter='none';

if nargin<5
  Resize = 'off';
end

Options = struct([]);

if nargin==5 && isstruct(Resize)
  Options = Resize;
  Resize  = 'off';
  if isfield(Options,'Interpreter'), Interpreter=Options.Interpreter; end
  if isfield(Options,'WindowStyle'), WindowStyle=Options.WindowStyle; end
  if isfield(Options,'Resize'),      Resize=Options.Resize;           end
end

[rw,cl]=size(NumLines);
OneVect = ones(NumQuest,1);
if (rw == 1 & cl == 2)
  NumLines=NumLines(OneVect,:);
elseif (rw == 1 & cl == 1)
  NumLines=NumLines(OneVect);
elseif (rw == 1 & cl == NumQuest)
  NumLines = NumLines';
elseif rw ~= NumQuest | cl > 2,
  error('NumLines size is incorrect.')
end

if ~iscell(DefAns),
  error('Default Answer must be a cell array of strings.');
end

%%%%%%%%%%%%%%%%%%%%%%%
%%% Create InputFig %%%
%%%%%%%%%%%%%%%%%%%%%%%
FigWidth=300;
FigHeight=100;
FigPos(3:4)=[FigWidth FigHeight];
FigColor=get(0,'Defaultuicontrolbackgroundcolor');

InputFig=dialog(                   ...
  'Visible'          ,'off'      , ...
  'KeyPressFcn'      ,@doFigureKeyPress, ...
  'Name'             ,Title      , ...
  'Pointer'          ,'arrow'    , ...
  'Units'            ,'pixels'   , ...
  'UserData'         ,'Cancel'   , ...
  'Tag'              ,Title      , ...
  'HandleVisibility' ,'callback' , ...
  'Color'            ,FigColor   , ...
  'NextPlot'         ,'add'      , ...
  'WindowStyle'      ,WindowStyle, ...
  'DoubleBuffer'     ,'on'       , ...
  'Resize'           ,Resize       ...
  );


%%%%%%%%%%%%%%%%%%%%%
%%% Set Positions %%%
%%%%%%%%%%%%%%%%%%%%%
DefOffset   = 10;

DefBtnWidth = 56;
BtnHeight   = 22;
BtnYOffset  = DefOffset;
BtnFontSize = get(0,'FactoryUIControlFontSize');
BtnWidth    = DefBtnWidth;

TextInfo.Units              = 'pixels'   ;
TextInfo.FontSize           = BtnFontSize;
TextInfo.HorizontalAlignment= 'left'     ;
TextInfo.HandleVisibility   = 'callback' ;

StInfo=TextInfo;
StInfo.Style              = 'text'     ;
StInfo.BackgroundColor    = FigColor;

TextInfo.VerticalAlignment= 'bottom';

EdInfo=StInfo;
EdInfo.Style           = 'edit';
EdInfo.BackgroundColor = 'w';

BtnInfo=StInfo;
BtnInfo.Style               = 'pushbutton';
BtnInfo.HorizontalAlignment = 'center';

% adjust button height and width
btnMargin=1.4;
ExtControl=uicontrol(InputFig   ,BtnInfo     , ...
                     'String'   ,'OK'        , ...
                     'Visible'  ,'off'         ...
                     );
BtnExtent = get(ExtControl,'Extent');
BtnWidth  = max(BtnWidth,BtnExtent(3)+8);
BtnHeight = max(BtnHeight,BtnExtent(4)*btnMargin);
delete(ExtControl);

TxtWidth=FigWidth-2*DefOffset;

% Determine # of lines for all Prompts
ExtControl=uicontrol(InputFig   ,StInfo     , ...
                     'String'   ,''         , ...
                     'Position' ,[ DefOffset DefOffset 0.96*TxtWidth BtnHeight ] , ...
                     'Visible'  ,'off'        ...
                     );

WrapQuest=cell(NumQuest,1);
QuestPos=zeros(NumQuest,4);

for ExtLp=1:NumQuest
   if size(NumLines,2)==2
      [WrapQuest{ExtLp},QuestPos(ExtLp,1:4)]= ...
         textwrap(ExtControl,Prompt(ExtLp),NumLines(ExtLp,2));
   else
      [WrapQuest{ExtLp},QuestPos(ExtLp,1:4)]= ...
         textwrap(ExtControl,Prompt(ExtLp),80);
   end
end % for ExtLp

delete(ExtControl);
QuestHeight=QuestPos(:,4);

TxtHeight=QuestHeight(1)/size(WrapQuest{1,1},1);
EditHeight=TxtHeight*NumLines(:,1);
EditHeight(NumLines(:,1)==1)=EditHeight(NumLines(:,1)==1)+4;

FigHeight=(NumQuest+2)*DefOffset    + ...
          BtnHeight+sum(EditHeight) + ...
          sum(QuestHeight);

TxtXOffset=DefOffset;

QuestYOffset=zeros(NumQuest,1);
EditYOffset=zeros(NumQuest,1);
QuestYOffset(1)=FigHeight-DefOffset-QuestHeight(1);
EditYOffset(1)=QuestYOffset(1)-EditHeight(1);

for YOffLp=2:NumQuest,
   QuestYOffset(YOffLp)=EditYOffset(YOffLp-1)-QuestHeight(YOffLp)-DefOffset;
   EditYOffset(YOffLp)=QuestYOffset(YOffLp)-EditHeight(YOffLp);
end % for YOffLp

QuestHandle=[];
```

```
EditHandle=[];
FigWidth =1;

AxesHandle=axes('Parent',InputFig,'Position',[0 0 1 1],'Visible','off');

for lp=1:NumQuest,
  EditTag=['Edit' num2str(lp)];
  if ~ischar(DefAns{lp}),
    delete(InputFig);
    error('Default Answer must be a cell array of strings.');
  end

  EditHandle(lp)=uicontrol(InputFig    , ...
                EdInfo      , ...
                'Max'       ,NumLines(lp,1)      , ...
                'Position'   ,[ TxtXOffset EditYOffset(lp) TxtWidth EditHeight(lp) ], ...
                'String'     ,DefAns{lp}          , ...
                'Tag'        ,'Edit'              ...
                );

  QuestHandle(lp)=text('Parent'     ,AxesHandle, ...
              TextInfo     , ...
              'Position'   ,[ TxtXOffset QuestYOffset(lp)], ...
              'String'     ,WrapQuest{lp}                , ...
              'Color'      ,get(EditHandle(lp),'ForegroundColor')   , ...
              'Interpreter',Interpreter                  , ...
              'Tag'        ,'Quest'                       ...
              );

  if size(NumLines,2) == 2,
    set(EditHandle(lp),'String',char(ones(1,NumLines(lp,2))*'x'));
    Extent = get(EditHandle(lp),'Extent');
    NewPos = [TxtXOffset EditYOffset(lp)  Extent(3) EditHeight(lp) ];

    NewPos1= [TxtXOffset QuestYOffset(lp)];
    set(EditHandle(lp),'Position',NewPos,'String',DefAns{lp})
    set(QuestHandle(lp),'Position',NewPos1)

    FigWidth=max(FigWidth,Extent(3)+2*DefOffset);
  else
    FigWidth=max(175,TxtWidth+2*DefOffset);
  end

end % for lp

FigPos=get(InputFig,'Position');

FigWidth=max(FigWidth,2*(BtnWidth+DefOffset)+DefOffset);
FigPos(1)=0;
FigPos(2)=0;
FigPos(3)=FigWidth;
FigPos(4)=FigHeight;

set(InputFig,'Position',getnicedialoglocation(FigPos,get(InputFig,'Units')));

OKHandle=uicontrol(InputFig     ,              ...
          BtnInfo      , ...
          'Position'   ,[ FigWidth-2*BtnWidth-2*DefOffset DefOffset BtnWidth BtnHeight ] , ...
          'KeyPressFcn',@doControlKeyPress , ...
          'String'     ,'OK'        , ...
          'Callback'   ,@doCallback , ...
          'Tag'        ,'OK'        , ...
          'UserData'   ,'OK'         ...
          );

BtnPos = get(OKHandle,'Position');
h = uicontrol(InputFig,'BackgroundColor', 'k', ...
        'Style','frame','Position',[ BtnPos(1)-1 BtnPos(2)-1 BtnPos(3)+2 BtnPos(4)+2 ]);
uistack(h,'bottom')

CancelHandle=uicontrol(InputFig     ,              ...
            BtnInfo      , ...
            'Position'   ,[ FigWidth-BtnWidth-DefOffset DefOffset BtnWidth BtnHeight ]           , ...
            'KeyPressFcn',@doControlKeyPress            , ...
            'String'     ,'Cancel'    , ...
```

```
                  'Callback'   ,@doCallback , ...
                  'Tag'        ,'Cancel'    , ...
                  'UserData'   ,'Cancel'      ...
                  );

handles = guihandles(InputFig);
handles.MinFigWidth = FigWidth;
handles.FigHeight   = FigHeight;
handles.TextMargin  = 2*DefOffset;
guidata(InputFig,handles);
set(InputFig,'ResizeFcn',@doResize);

% make sure we are on screen
movegui(InputFig)

% if there is a figure out there and it's modal, we need to be modal too
if ~isempty(gcbf) && strcmp(get(gcbf,'WindowStyle'),'modal')
    set(InputFig,'WindowStyle','modal');
end

set(InputFig,'Visible','on');
drawnow;

uiwait(InputFig);

if ishandle(InputFig)
    Answer={'Cancel'};
    if strcmp(get(InputFig,'UserData'),'OK'),
        Answer=cell(NumQuest,1);
        for lp=1:NumQuest,
            Answer(lp)=get(EditHandle(lp),{'String'});
        end
    end
    delete(InputFig);
else
    Answer={'Cancel'};
end

function doFigureKeyPress(obj, evd)
switch(evd.Key)
 case {'return','space'}
  set(gcbf,'UserData','OK');
  uiresume(gcbf);
 case {'escape'}
  delete(gcbf);
end

function doControlKeyPress(obj, evd)
switch(evd.Key)
 case {'return'}
  if ~strcmp(get(obj,'UserData'),'Cancel')
      set(gcbf,'UserData','OK');
      uiresume(gcbf);
  else
      delete(gcbf)
  end
 case 'escape'
  delete(gcbf)
end

function doCallback(obj, evd)
if ~strcmp(get(obj,'UserData'),'Cancel')
    set(gcbf,'UserData','OK');
    uiresume(gcbf);
else
    delete(gcbf)
end

function doResize(FigHandle, evd)
Data=guidata(FigHandle);

resetPos = false;

FigPos = get(FigHandle,'Position');
FigWidth = FigPos(3);
```

```
FigHeight = FigPos(4);

if FigWidth < Data.MinFigWidth
    FigWidth  = Data.MinFigWidth;
    FigPos(3) = Data.MinFigWidth;
    resetPos = true;
end

% make sure edit fields use all available space
for lp = 1:length(Data.Edit)
    EditPos = get(Data.Edit(lp),'Position');
    EditPos(3) = FigWidth - Data.TextMargin;
    set(Data.Edit(lp),'Position',EditPos);
end

if FigHeight ~= Data.FigHeight
    FigPos(4) = Data.FigHeight;
    resetPos = true;
end

if resetPos
    set(FigHandle,'Position',FigPos);
end

function figure_size = getnicedialoglocation(figure_size, figure_units)
% adjust the specified figure position to fig nicely over GCBF
% or into the upper 3rd of the screen

%  Copyright 1999-2002 The MathWorks, Inc.
%  $Revision: 1.1.6.1 $

parentHandle = gcbf;
propName = 'Position';
if isempty(parentHandle)
    parentHandle = 0;
    propName = 'ScreenSize';
end

old_u = get(parentHandle,'Units');
set(parentHandle,'Units',figure_units);
container_size=get(parentHandle,propName);
set(parentHandle,'Units',old_u);

figure_size(1) = container_size(1)  + 1/2*(container_size(3) - figure_size(3));
figure_size(2) = container_size(2)  + 2/3*(container_size(4) - figure_size(4));
```

Appendix C

Two-dimensional FDTD simulation of radiation from a narrow slit excited by a bandpass pulse
Source Code

```
%Filename: onewl1.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all %(added)
close all %(added)
pack %(added)
clc %(added)

%***********************************************************************
%    Fundamental constants
%***********************************************************************

cc=2.99792458e8;           %speed of light in free space
muz=4.0*pi*1.0e-7;         %permeability of free space
epsz=1.0/(cc*cc*muz);      %permittivity of free space

freq=10.0e+9;              %center frequency of source excitation
lambda=cc/freq;            %center wavelength of source excitation
omega=2.0*pi*freq;

%***********************************************************************
%    Grid parameters
%***********************************************************************

ie=100;          %number of grid cells in x-direction
je=50;           %number of grid cells in y-direction

ib=ie+1;
jb=je+1;

is=15;           %location of z-directed hard source
js=je/2;         %location of z-directed hard source

dx=3.0e-3;       %space increment of square lattice
dt=dx/(2.0*cc);  %time step

nmax=168; %(changed)       %total number of time steps

iebc=8;          %thickness of left and right PML region
jebc=8;          %thickness of front and back PML region
rmax=0.00001;
orderbc=2;
ibbc=iebc+1;
jbbc=jebc+1;
iefbc=ie+2*iebc;
jefbc=je+2*jebc;
ibfbc=iefbc+1;
jbfbc=jefbc+1;

%***********************************************************************
%    Material parameters
%***********************************************************************

media=2;

eps=[1.0 1.0];
sig=[0.0 1.0e+7];
mur=[1.0 1.0];
sim=[0.0 0.0];

%***********************************************************************
%    Wave excitation
%***********************************************************************
```

```
rtau=160.0e-12;
tau=rtau/dt;
delay=3*tau;

source=zeros(1,nmax);
for n=1:1*tau %(added)
  source(n)=sin(omega*dt*n); %(added)
end %(added)

%***********************************************************************
%     Field arrays
%***********************************************************************

ex=zeros(ie,jb);           %fields in main grid
ey=zeros(ib,je);
hz=zeros(ie,je);

exbcf=zeros(iefbc,jebc);   %fields in front PML region
eybcf=zeros(ibfbc,jebc);
hzxbcf=zeros(iefbc,jebc);
hzybcf=zeros(iefbc,jebc);

exbcb=zeros(iefbc,jbbc);   %fields in back PML region
eybcb=zeros(ibfbc,jebc);
hzxbcb=zeros(iefbc,jebc);
hzybcb=zeros(iefbc,jebc);

exbcl=zeros(iebc,jb);      %fields in left PML region
eybcl=zeros(iebc,je);
hzxbcl=zeros(iebc,je);
hzybcl=zeros(iebc,je);

exbcr=zeros(iebc,jb);      %fields in right PML region
eybcr=zeros(ibbc,je);
hzxbcr=zeros(iebc,je);
hzybcr=zeros(iebc,je);

%***********************************************************************
%     Updating coefficients
%***********************************************************************

for i=1:media
  eaf  =dt*sig(i)/(2.0*epsz*eps(i));
  ca(i)=(1.0-eaf)/(1.0+eaf);
  cb(i)=dt/epsz/eps(i)/dx/(1.0+eaf);
  haf  =dt*sim(i)/(2.0*muz*mur(i));
  da(i)=(1.0-haf)/(1.0+haf);
  db(i)=dt/muz/mur(i)/dx/(1.0+haf);
end

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

caex(1:ie,1:jb)=ca(1);
cbex(1:ie,1:jb)=cb(1);

caey(1:ib,1:je)=ca(1);
cbey(1:ib,1:je)=cb(1);

dahz(1:ie,1:je)=da(1);
dbhz(1:ie,1:je)=db(1);

%     Add PEC with slit of width equals one lambda
caex(80,[1:20])=ca(2); %(added)
caex(80,[30:je])=ca(2); %(added)
cbex(80,[1:20])=cb(2); %(added)
cbex(80,[30:je])=cb(2); %(added)
caey(80,[1:20])=ca(2); %(added)
caey(80,[30:je])=ca(2); %(added)
cbey(80,[1:20])=cb(2); %(added)
cbey(80,[30:je])=cb(2); %(added)
```

```
%***********************************************************************
%     Fill the PML regions
%***********************************************************************

delbc=iebc*dx;
sigmam=-log(rmax/100.0)*epsz*cc*(orderbc+1)/(2*delbc);
bcfactor=eps(1)*sigmam/(dx*(delbc^orderbc)*(orderbc+1));

%     FRONT region

caexbcf(1:iefbc,1)=1.0;
cbexbcf(1:iefbc,1)=0.0;
for j=2:jebc
  y1=(jebc-j+1.5)*dx;
  y2=(jebc-j+0.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1.0-ca1)/(sigmay*dx);
  caexbcf(1:iefbc,j)=ca1;
  cbexbcf(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,1)=ca1;
cbex(1:ie,1)=cb1;
caexbcl(1:iebc,1)=ca1;
cbexbcl(1:iebc,1)=cb1;
caexbcr(1:iebc,1)=ca1;
cbexbcr(1:iebc,1)=cb1;

for j=1:jebc
  y1=(jebc-j+1)*dx;
  y2=(jebc-j)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcf(1:iefbc,j)=da1;
  dbhzybcf(1:iefbc,j)=db1;
  caeybcf(1:ibfbc,j)=ca(1);
  cbeybcf(1:ibfbc,j)=cb(1);
  dahzxbcf(1:iefbc,j)=da(1);
  dbhzxbcf(1:iefbc,j)=db(1);
end

%     BACK region

caexbcb(1:iefbc,jbbc)=1.0;
cbexbcb(1:iefbc,jbbc)=0.0;
for j=2:jebc
  y1=(j-0.5)*dx;
  y2=(j-1.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmay*dx);
  caexbcb(1:iefbc,j)=ca1;
  cbexbcb(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,jb)=ca1;
cbex(1:ie,jb)=cb1;
caexbcl(1:iebc,jb)=ca1;
cbexbcl(1:iebc,jb)=cb1;
caexbcr(1:iebc,jb)=ca1;
cbexbcr(1:iebc,jb)=cb1;

for j=1:jebc
  y1=j*dx;
  y2=(j-1)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcb(1:iefbc,j)=da1;
  dbhzybcb(1:iefbc,j)=db1;
  caeybcb(1:ibfbc,j)=ca(1);
  cbeybcb(1:ibfbc,j)=cb(1);
  dahzxbcb(1:iefbc,j)=da(1);
  dbhzxbcb(1:iefbc,j)=db(1);
end

%     LEFT region

caeybcl(1,1:je)=1.0;
cbeybcl(1,1:je)=0.0;
for i=2:iebc
  x1=(iebc-i+1.5)*dx;
  x2=(iebc-i+0.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcl(i,1:je)=ca1;
  cbeybcl(i,1:je)=cb1;
  caeybcf(i,1:jebc)=ca1;
  cbeybcf(i,1:jebc)=cb1;
  caeybcb(i,1:jebc)=ca1;
  cbeybcb(i,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(1,1:je)=ca1;
cbey(1,1:je)=cb1;
caeybcf(iebc+1,1:jebc)=ca1;
cbeybcf(iebc+1,1:jebc)=cb1;
caeybcb(iebc+1,1:jebc)=ca1;
cbeybcb(iebc+1,1:jebc)=cb1;

for i=1:iebc
  x1=(iebc-i+1)*dx;
  x2=(iebc-i)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcl(i,1:je)=da1;
  dbhzxbcl(i,1:je)=db1;
  dahzxbcf(i,1:jebc)=da1;
  dbhzxbcf(i,1:jebc)=db1;
  dahzxbcb(i,1:jebc)=da1;
  dbhzxbcb(i,1:jebc)=db1;
  caexbcl(i,2:je)=ca(1);
  cbexbcl(i,2:je)=cb(1);
  dahzybcl(i,1:je)=da(1);
  dbhzybcl(i,1:je)=db(1);
end

%     RIGHT region

caeybcr(ibbc,1:je)=1.0;
cbeybcr(ibbc,1:je)=0.0;
for i=2:iebc
  x1=(i-0.5)*dx;
  x2=(i-1.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcr(i,1:je)=ca1;
  cbeybcr(i,1:je)=cb1;
  caeybcf(i+iebc+ie,1:jebc)=ca1;
  cbeybcf(i+iebc+ie,1:jebc)=cb1;
  caeybcb(i+iebc+ie,1:jebc)=ca1;
  cbeybcb(i+iebc+ie,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
```

```
caey(ib,1:je)=ca1;
cbey(ib,1:je)=cb1;
caeybcf(iebc+ib,1:jebc)=ca1;
cbeybcf(iebc+ib,1:jebc)=cb1;
caeybcb(iebc+ib,1:jebc)=ca1;
cbeybcb(iebc+ib,1:jebc)=cb1;

for i=1:iebc
  x1=i*dx;
  x2=(i-1)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcr(i,1:je) = da1;
  dbhzxbcr(i,1:je) = db1;
  dahzxbcf(i+ie+iebc,1:jebc)=da1;
  dbhzxbcf(i+ie+iebc,1:jebc)=db1;
  dahzxbcb(i+ie+iebc,1:jebc)=da1;
  dbhzxbcb(i+ie+iebc,1:jebc)=db1;
  caexbcr(i,2:je)=ca(1);
  cbexbcr(i,2:je)=cb(1);
  dahzybcr(i,1:je)=da(1);
  dbhzybcr(i,1:je)=db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

subplot(3,1,1),pcolor(ex');
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = 0']);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = 0']);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = 0']);

rect=get(gcf,'Position');
rect(1:2)=[0 0];

M=moviein(nmax/8,gcf,rect);

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n=1:nmax

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

ex(:,2:je)=caex(:,2:je).*ex(:,2:je)+...
           cbex(:,2:je).*(hz(:,2:je)-hz(:,1:je-1));
```

```
ey(2:ie,:)=caey(2:ie,:).*ey(2:ie,:)+...
          cbey(2:ie,:).*(hz(1:ie-1,:)-hz(2:ie,:));

ex(80,2:20)=0; %(added)
ex(80,30:je)=0; %(added)
ey(80,1:20)=0; %(added)
ey(80,30:je)=0; %(added)
%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

exbcf(:,2:jebc)=caexbcf(:,2:jebc).*exbcf(:,2:jebc)-...
  cbexbcf(:,2:jebc).*(hzxbcf(:,1:jebc-1)+hzybcf(:,1:jebc-1)-...
                   hzxbcf(:,2:jebc)-hzybcf(:,2:jebc));
ex(1:ie,1)=caex(1:ie,1).*ex(1:ie,1)-...
  cbex(1:ie,1).*(hzxbcf(ibbc:iebc+ie,jebc)+...
              hzybcf(ibbc:iebc+ie,jebc)-hz(1:ie,1));

%     BACK

exbcb(:,2:jebc-1)=caexbcb(:,2:jebc-1).*exbcb(:,2:jebc-1)-...
  cbexbcb(:,2:jebc-1).*(hzxbcb(:,1:jebc-2)+hzybcb(:,1:jebc-2)-...
                     hzxbcb(:,2:jebc-1)-hzybcb(:,2:jebc-1));
ex(1:ie,jb)=caex(1:ie,jb).*ex(1:ie,jb)-...
  cbex(1:ie,jb).*(hz(1:ie,jb-1)-hzxbcb(ibbc:iebc+ie,1)-...
               hzybcb(ibbc:iebc+ie,1));

%     LEFT

exbcl(:,2:je)=caexbcl(:,2:je).*exbcl(:,2:je)-...
  cbexbcl(:,2:je).*(hzxbcl(:,1:je-1)+hzybcl(:,1:je-1)-...
                  hzxbcl(:,2:je)-hzybcl(:,2:je));
exbcl(:,1)=caexbcl(:,1).*exbcl(:,1)-...
  cbexbcl(:,1).*(hzxbcf(1:iebc,jebc)+hzybcf(1:iebc,jebc)-...
               hzxbcl(:,1)-hzybcl(:,1));
exbcl(:,jb)=caexbcl(:,jb).*exbcl(:,jb)-...
  cbexbcl(:,jb).*(hzxbcl(:,je)+hzybcl(:,je)-...
                hzxbcb(1:iebc,1)-hzybcb(1:iebc,1));

%     RIGHT

exbcr(:,2:je)=caexbcr(:,2:je).*exbcr(:,2:je)-...
  cbexbcr(:,2:je).*(hzxbcr(:,1:je-1)+hzybcr(:,1:je-1)-...
                  hzxbcr(:,2:je)-hzybcr(:,2:je));
exbcr(:,1)=caexbcr(:,1).*exbcr(:,1)-...
  cbexbcr(:,1).*(hzxbcf(1+iebc+ie:iefbc,jebc)+...
               hzybcf(1+iebc+ie:iefbc,jebc)-...
               hzxbcr(:,1)-hzybcr(:,1));
exbcr(:,jb)=caexbcr(:,jb).*exbcr(:,jb)-...
  cbexbcr(:,jb).*(hzxbcr(:,je)+hzybcr(:,je)-...
                hzxbcb(1+iebc+ie:iefbc,1)-...
                hzybcb(1+iebc+ie:iefbc,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

eybcf(2:iefbc,:)=caeybcf(2:iefbc,:).*eybcf(2:iefbc,:)-...
  cbeybcf(2:iefbc,:).*(hzxbcf(2:iefbc,:)+hzybcf(2:iefbc,:)-...
                    hzxbcf(1:iefbc-1,:)-hzybcf(1:iefbc-1,:));

%     BACK

eybcb(2:iefbc,:)=caeybcb(2:iefbc,:).*eybcb(2:iefbc,:)-...
  cbeybcb(2:iefbc,:).*(hzxbcb(2:iefbc,:)+hzybcb(2:iefbc,:)-...
                    hzxbcb(1:iefbc-1,:)-hzybcb(1:iefbc-1,:));

%     LEFT

eybcl(2:iebc,:)=caeybcl(2:iebc,:).*eybcl(2:iebc,:)-...
  cbeybcl(2:iebc,:).*(hzxbcl(2:iebc,:)+hzybcl(2:iebc,:)-...
                 hzxbcl(1:iebc-1,:)-hzybcl(1:iebc-1,:));
ey(1,:)=caey(1,:).*ey(1,:)-...
  cbey(1,:).*(hz(1,:)-hzxbcl(iebc,:)-hzybcl(iebc,:));

%     RIGHT

eybcr(2:iebc,:)=caeybcr(2:iebc,:).*eybcr(2:iebc,:)-...
  cbeybcr(2:iebc,:).*(hzxbcr(2:iebc,:)+hzybcr(2:iebc,:)-...
                 hzxbcr(1:iebc-1,:)-hzybcr(1:iebc-1,:));
ey(ib,:)=caey(ib,:).*ey(ib,:)-...
  cbey(ib,:).*(hzxbcr(1,:)+hzybcr(1,:)- hz(ie,:));


%***********************************************************************
%     Update magnetic fields (HZ) in main grid
%***********************************************************************

hz(1:ie,1:je)=dahz(1:ie,1:je).*hz(1:ie,1:je)+...
            dbhz(1:ie,1:je).*(ex(1:ie,2:jb)-ex(1:ie,1:je)+...
                           ey(1:ie,1:je)-ey(2:ib,1:je));

hz(is,js)=source(n);

hz(80,1:20)=0; %(added)
hz(80,30:je)=0; %(added)

%***********************************************************************
%     Update HZX in PML regions
%***********************************************************************

%     FRONT

hzxbcf(1:iefbc,:)=dahzxbcf(1:iefbc,:).*hzxbcf(1:iefbc,:)-...
  dbhzxbcf(1:iefbc,:).*(eybcf(2:ibfbc,:)-eybcf(1:iefbc,:));

%     BACK

hzxbcb(1:iefbc,:)=dahzxbcb(1:iefbc,:).*hzxbcb(1:iefbc,:)-...
  dbhzxbcb(1:iefbc,:).*(eybcb(2:ibfbc,:)-eybcb(1:iefbc,:));

%     LEFT

hzxbcl(1:iebc-1,:)=dahzxbcl(1:iebc-1,:).*hzxbcl(1:iebc-1,:)-...
  dbhzxbcl(1:iebc-1,:).*(eybcl(2:iebc,:)-eybcl(1:iebc-1,:));
hzxbcl(iebc,:)=dahzxbcl(iebc,:).*hzxbcl(iebc,:)-...
  dbhzxbcl(iebc,:).*(ey(1,:)-eybcl(iebc,:));

%     RIGHT

hzxbcr(2:iebc,:)=dahzxbcr(2:iebc,:).*hzxbcr(2:iebc,:)-...
  dbhzxbcr(2:iebc,:).*(eybcr(3:ibbc,:)-eybcr(2:iebc,:));
hzxbcr(1,:)=dahzxbcr(1,:).*hzxbcr(1,:)-...
  dbhzxbcr(1,:).*(eybcr(2,:)-ey(ib,:));

%***********************************************************************
%     Update HZY in PML regions
%***********************************************************************

%     FRONT

hzybcf(:,1:jebc-1)=dahzybcf(:,1:jebc-1).*hzybcf(:,1:jebc-1)-...
  dbhzybcf(:,1:jebc-1).*(exbcf(:,1:jebc-1)-exbcf(:,2:jebc));
hzybcf(1:iebc,jebc)=dahzybcf(1:iebc,jebc).*hzybcf(1:iebc,jebc)-...
  dbhzybcf(1:iebc,jebc).*(exbcf(1:iebc,jebc)-exbcl(1:iebc,1));
hzybcf(iebc+1:iebc+ie,jebc)=...
  dahzybcf(iebc+1:iebc+ie,jebc).*hzybcf(iebc+1:iebc+ie,jebc)-...
  dbhzybcf(iebc+1:iebc+ie,jebc).*(exbcf(iebc+1:iebc+ie,jebc)-...
                            ex(1:ie,1));
hzybcf(iebc+ie+1:iefbc,jebc)=...
  dahzybcf(iebc+ie+1:iefbc,jebc).*hzybcf(iebc+ie+1:iefbc,jebc)-...
  dbhzybcf(iebc+ie+1:iefbc,jebc).*(exbcf(iebc+ie+1:iefbc,jebc)-...
                            exbcr(1:iebc,1));

%     BACK
```

```
hzybcb(1:iefbc,2:jebc)=dahzybcb(1:iefbc,2:jebc).*hzybcb(1:iefbc,2:jebc)-...
  dbhzybcb(1:iefbc,2:jebc).*(exbcb(1:iefbc,2:jebc)-exbcb(1:iefbc,3:jbbc));
hzybcb(1:iebc,1)=dahzybcb(1:iebc,1).*hzybcb(1:iebc,1)-...
  dbhzybcb(1:iebc,1).*(exbcl(1:iebc,jb)-exbcb(1:iebc,2));
hzybcb(iebc+1:iebc+ie,1)=...
  dahzybcb(iebc+1:iebc+ie,1).*hzybcb(iebc+1:iebc+ie,1)-...
  dbhzybcb(iebc+1:iebc+ie,1).*(ex(1:ie,jb)-exbcb(iebc+1:iebc+ie,2));
hzybcb(iebc+ie+1:iefbc,1)=...
  dahzybcb(iebc+ie+1:iefbc,1).*hzybcb(iebc+ie+1:iefbc,1)-...
  dbhzybcb(iebc+ie+1:iefbc,1).*(exbcr(1:iebc,jb)-...
                              exbcb(iebc+ie+1:iefbc,2));

%     LEFT

hzybcl(:,1:je)=dahzybcl(:,1:je).*hzybcl(:,1:je)-...
  dbhzybcl(:,1:je).*(exbcl(:,1:je)-exbcl(:,2:jb));

%     RIGHT

hzybcr(:,1:je)=dahzybcr(:,1:je).*hzybcr(:,1:je)-...
  dbhzybcr(:,1:je).*(exbcr(:,1:je)-exbcr(:,2:jb));

%***********************************************************************
%     Visualize fields
%***********************************************************************

if mod(n,8)==0;

timestep=int2str(n);

subplot(3,1,1),pcolor(ex');
colormap hot %(added)
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = ',timestep]);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = ',timestep]);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = ',timestep]);

nn=n/8;
M(:,nn)=getframe(gcf,rect);

end;

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(gcf,M,0,10,rect);
```

```
%Filename: onewl2.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all %(added)
close all %(added)
pack %(added)
clc %(added)

%***********************************************************************
%   Fundamental constants
%***********************************************************************

cc=2.99792458e8;          %speed of light in free space
muz=4.0*pi*1.0e-7;        %permeability of free space
epsz=1.0/(cc*cc*muz);     %permittivity of free space

freq=5.0e+9;              %center frequency of source excitation
lambda=cc/freq;           %center wavelength of source excitation
omega=2.0*pi*freq;

%***********************************************************************
%   Grid parameters
%***********************************************************************

ie=100;         %number of grid cells in x-direction
je=50;          %number of grid cells in y-direction

ib=ie+1;
jb=je+1;

is=15;          %location of z-directed hard source
js=je/2;        %location of z-directed hard source

dx=3.0e-3;      %space increment of square lattice
dt=dx/(2.0*cc);   %time step

nmax=240; %(changed)      %total number of time steps

iebc=8;         %thickness of left and right PML region
jebc=8;         %thickness of front and back PML region
rmax=0.00001;
orderbc=2;
ibbc=iebc+1;
jbbc=jebc+1;
iefbc=ie+2*iebc;
jefbc=je+2*jebc;
ibfbc=iefbc+1;
jbfbc=jefbc+1;

%***********************************************************************
%   Material parameters
%***********************************************************************

media=2;

eps=[1.0 1.0];
sig=[0.0 1.0e+7];
mur=[1.0 1.0];
sim=[0.0 0.0];

%***********************************************************************
%   Wave excitation
%***********************************************************************
```

```
rtau=160.0e-12;
tau=rtau/dt;
delay=3*tau;

source=zeros(1,nmax);
for n=1:4*tau
  source(n)=sin(omega*(n-delay)*dt)*exp(-((n-delay)^2/tau^2));
end

%***********************************************************************
%     Field arrays
%***********************************************************************

ex=zeros(ie,jb);           %fields in main grid
ey=zeros(ib,je);
hz=zeros(ie,je);

exbcf=zeros(iefbc,jebc);   %fields in front PML region
eybcf=zeros(ibfbc,jebc);
hzxbcf=zeros(iefbc,jebc);
hzybcf=zeros(iefbc,jebc);

exbcb=zeros(iefbc,jbbc);   %fields in back PML region
eybcb=zeros(ibfbc,jebc);
hzxbcb=zeros(iefbc,jebc);
hzybcb=zeros(iefbc,jebc);

exbcl=zeros(iebc,jb);      %fields in left PML region
eybcl=zeros(iebc,je);
hzxbcl=zeros(iebc,je);
hzybcl=zeros(iebc,je);

exbcr=zeros(iebc,jb);      %fields in right PML region
eybcr=zeros(ibbc,je);
hzxbcr=zeros(iebc,je);
hzybcr=zeros(iebc,je);

%***********************************************************************
%     Updating coefficients
%***********************************************************************

for i=1:media
  eaf  =dt*sig(i)/(2.0*epsz*eps(i));
  ca(i)=(1.0-eaf)/(1.0+eaf);
  cb(i)=dt/epsz/eps(i)/dx/(1.0+eaf);
  haf  =dt*sim(i)/(2.0*muz*mur(i));
  da(i)=(1.0-haf)/(1.0+haf);
  db(i)=dt/muz/mur(i)/dx/(1.0+haf);
end

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

caex(1:ie,1:jb)=ca(1);
cbex(1:ie,1:jb)=cb(1);

caey(1:ib,1:je)=ca(1);
cbey(1:ib,1:je)=cb(1);

dahz(1:ie,1:je)=da(1);
dbhz(1:ie,1:je)=db(1);

%     Add PEC with slit of width equals one lambda
caex(80,[1:20])=ca(2); %(added)
caex(80,[30:je])=ca(2); %(added)
cbex(80,[1:20])=cb(2); %(added)
cbex(80,[30:je])=cb(2); %(added)
caey(80,[1:20])=ca(2); %(added)
caey(80,[30:je])=ca(2); %(added)
cbey(80,[1:20])=cb(2); %(added)
cbey(80,[30:je])=cb(2); %(added)
```

```
%***********************************************************************
%     Fill the PML regions
%***********************************************************************

delbc=iebc*dx;
sigmam=-log(rmax/100.0)*epsz*cc*(orderbc+1)/(2*delbc);
bcfactor=eps(1)*sigmam/(dx*(delbc^orderbc)*(orderbc+1));

%     FRONT region

caexbcf(1:iefbc,1)=1.0;
cbexbcf(1:iefbc,1)=0.0;
for j=2:jebc
  y1=(jebc-j+1.5)*dx;
  y2=(jebc-j+0.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1.0-ca1)/(sigmay*dx);
  caexbcf(1:iefbc,j)=ca1;
  cbexbcf(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,1)=ca1;
cbex(1:ie,1)=cb1;
caexbcl(1:iebc,1)=ca1;
cbexbcl(1:iebc,1)=cb1;
caexbcr(1:iebc,1)=ca1;
cbexbcr(1:iebc,1)=cb1;

for j=1:jebc
  y1=(jebc-j+1)*dx;
  y2=(jebc-j)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcf(1:iefbc,j)=da1;
  dbhzybcf(1:iefbc,j)=db1;
  caeybcf(1:ibfbc,j)=ca(1);
  cbeybcf(1:ibfbc,j)=cb(1);
  dahzxbcf(1:iefbc,j)=da(1);
  dbhzxbcf(1:iefbc,j)=db(1);
end

%     BACK region

caexbcb(1:iefbc,jbbc)=1.0;
cbexbcb(1:iefbc,jbbc)=0.0;
for j=2:jebc
  y1=(j-0.5)*dx;
  y2=(j-1.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmay*dx);
  caexbcb(1:iefbc,j)=ca1;
  cbexbcb(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,jb)=ca1;
cbex(1:ie,jb)=cb1;
caexbcl(1:iebc,jb)=ca1;
cbexbcl(1:iebc,jb)=cb1;
caexbcr(1:iebc,jb)=ca1;
cbexbcr(1:iebc,jb)=cb1;

for j=1:jebc
  y1=j*dx;
  y2=(j-1)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcb(1:iefbc,j)=da1;
  dbhzybcb(1:iefbc,j)=db1;
  caeybcb(1:ibfbc,j)=ca(1);
  cbeybcb(1:ibfbc,j)=cb(1);
  dahzxbcb(1:iefbc,j)=da(1);
  dbhzxbcb(1:iefbc,j)=db(1);
end

%     LEFT region

caeybcl(1,1:je)=1.0;
cbeybcl(1,1:je)=0.0;
for i=2:iebc
  x1=(iebc-i+1.5)*dx;
  x2=(iebc-i+0.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcl(i,1:je)=ca1;
  cbeybcl(i,1:je)=cb1;
  caeybcf(i,1:jebc)=ca1;
  cbeybcf(i,1:jebc)=cb1;
  caeybcb(i,1:jebc)=ca1;
  cbeybcb(i,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(1,1:je)=ca1;
cbey(1,1:je)=cb1;
caeybcf(iebc+1,1:jebc)=ca1;
cbeybcf(iebc+1,1:jebc)=cb1;
caeybcb(iebc+1,1:jebc)=ca1;
cbeybcb(iebc+1,1:jebc)=cb1;

for i=1:iebc
  x1=(iebc-i+1)*dx;
  x2=(iebc-i)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcl(i,1:je)=da1;
  dbhzxbcl(i,1:je)=db1;
  dahzxbcf(i,1:jebc)=da1;
  dbhzxbcf(i,1:jebc)=db1;
  dahzxbcb(i,1:jebc)=da1;
  dbhzxbcb(i,1:jebc)=db1;
  caexbcl(i,2:je)=ca(1);
  cbexbcl(i,2:je)=cb(1);
  dahzybcl(i,1:je)=da(1);
  dbhzybcl(i,1:je)=db(1);
end

%     RIGHT region

caeybcr(ibbc,1:je)=1.0;
cbeybcr(ibbc,1:je)=0.0;
for i=2:iebc
  x1=(i-0.5)*dx;
  x2=(i-1.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcr(i,1:je)=ca1;
  cbeybcr(i,1:je)=cb1;
  caeybcf(i+iebc+ie,1:jebc)=ca1;
  cbeybcf(i+iebc+ie,1:jebc)=cb1;
  caeybcb(i+iebc+ie,1:jebc)=ca1;
  cbeybcb(i+iebc+ie,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
```

```
caey(ib,1:je)=ca1;
cbey(ib,1:je)=cb1;
caeybcf(iebc+ib,1:jebc)=ca1;
cbeybcf(iebc+ib,1:jebc)=cb1;
caeybcb(iebc+ib,1:jebc)=ca1;
cbeybcb(iebc+ib,1:jebc)=cb1;

for i=1:iebc
  x1=i*dx;
  x2=(i-1)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcr(i,1:je) = da1;
  dbhzxbcr(i,1:je) = db1;
  dahzxbcf(i+ie+iebc,1:jebc)=da1;
  dbhzxbcf(i+ie+iebc,1:jebc)=db1;
  dahzxbcb(i+ie+iebc,1:jebc)=da1;
  dbhzxbcb(i+ie+iebc,1:jebc)=db1;
  caexbcr(i,2:je)=ca(1);
  cbexbcr(i,2:je)=cb(1);
  dahzybcr(i,1:je)=da(1);
  dbhzybcr(i,1:je)=db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

subplot(3,1,1),pcolor(ex');
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = 0']);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = 0']);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = 0']);

rect=get(gcf,'Position');
rect(1:2)=[0 0];

M=moviein(nmax/8,gcf,rect);

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n=1:nmax

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

ex(:,2:je)=caex(:,2:je).*ex(:,2:je)+...
           cbex(:,2:je).*(hz(:,2:je)-hz(:,1:je-1));
```

```
ey(2:ie,:)=caey(2:ie,:).*ey(2:ie,:)+...
          cbey(2:ie,:).*(hz(1:ie-1,:)-hz(2:ie,:));

ex(80,2:20)=0; %(added)
ex(80,30:je)=0; %(added)
ey(80,1:20)=0; %(added)
ey(80,30:je)=0; %(added)
%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

exbcf(:,2:jebc)=caexbcf(:,2:jebc).*exbcf(:,2:jebc)-...
  cbexbcf(:,2:jebc).*(hzxbcf(:,1:jebc-1)+hzybcf(:,1:jebc-1)-...
                    hzxbcf(:,2:jebc)-hzybcf(:,2:jebc));
ex(1:ie,1)=caex(1:ie,1).*ex(1:ie,1)-...
  cbex(1:ie,1).*(hzxbcf(ibbc:iebc+ie,jebc)+...
              hzybcf(ibbc:iebc+ie,jebc)-hz(1:ie,1));

%     BACK

exbcb(:,2:jebc-1)=caexbcb(:,2:jebc-1).*exbcb(:,2:jebc-1)-...
  cbexbcb(:,2:jebc-1).*(hzxbcb(:,1:jebc-2)+hzybcb(:,1:jebc-2)-...
                    hzxbcb(:,2:jebc-1)-hzybcb(:,2:jebc-1));
ex(1:ie,jb)=caex(1:ie,jb).*ex(1:ie,jb)-...
  cbex(1:ie,jb).*(hz(1:ie,jb-1)-hzxbcb(ibbc:iebc+ie,1)-...
              hzybcb(ibbc:iebc+ie,1));

%     LEFT

exbcl(:,2:je)=caexbcl(:,2:je).*exbcl(:,2:je)-...
  cbexbcl(:,2:je).*(hzxbcl(:,1:je-1)+hzybcl(:,1:je-1)-...
                  hzxbcl(:,2:je)-hzybcl(:,2:je));
exbcl(:,1)=caexbcl(:,1).*exbcl(:,1)-...
  cbexbcl(:,1).*(hzxbcf(1:iebc,jebc)+hzybcf(1:iebc,jebc)-...
              hzxbcl(:,1)-hzybcl(:,1));
exbcl(:,jb)=caexbcl(:,jb).*exbcl(:,jb)-...
  cbexbcl(:,jb).*(hzxbcl(:,je)+hzybcl(:,je)-...
               hzxbcb(1:iebc,1)-hzybcb(1:iebc,1));

%     RIGHT

exbcr(:,2:je)=caexbcr(:,2:je).*exbcr(:,2:je)-...
  cbexbcr(:,2:je).*(hzxbcr(:,1:je-1)+hzybcr(:,1:je-1)-...
                  hzxbcr(:,2:je)-hzybcr(:,2:je));
exbcr(:,1)=caexbcr(:,1).*exbcr(:,1)-...
  cbexbcr(:,1).*(hzxbcf(1+iebc+ie:iefbc,jebc)+...
              hzybcf(1+iebc+ie:iefbc,jebc)-...
              hzxbcr(:,1)-hzybcr(:,1));
exbcr(:,jb)=caexbcr(:,jb).*exbcr(:,jb)-...
  cbexbcr(:,jb).*(hzxbcr(:,je)+hzybcr(:,je)-...
               hzxbcb(1+iebc+ie:iefbc,1)-...
               hzybcb(1+iebc+ie:iefbc,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

eybcf(2:iefbc,:)=caeybcf(2:iefbc,:).*eybcf(2:iefbc,:)-...
  cbeybcf(2:iefbc,:).*(hzxbcf(2:iefbc,:)+hzybcf(2:iefbc,:)-...
                    hzxbcf(1:iefbc-1,:)-hzybcf(1:iefbc-1,:));

%     BACK

eybcb(2:iefbc,:)=caeybcb(2:iefbc,:).*eybcb(2:iefbc,:)-...
  cbeybcb(2:iefbc,:).*(hzxbcb(2:iefbc,:)+hzybcb(2:iefbc,:)-...
                    hzxbcb(1:iefbc-1,:)-hzybcb(1:iefbc-1,:));

%     LEFT

eybcl(2:iebc,:)=caeybcl(2:iebc,:).*eybcl(2:iebc,:)-...
  cbeybcl(2:iebc,:).*(hzxbcl(2:iebc,:)+hzybcl(2:iebc,:)-...
                hzxbcl(1:iebc-1,:)-hzybcl(1:iebc-1,:));
ey(1,:)=caey(1,:).*ey(1,:)-...
  cbey(1,:).*(hz(1,:)-hzxbcl(iebc,:)-hzybcl(iebc,:));

%    RIGHT

eybcr(2:iebc,:)=caeybcr(2:iebc,:).*eybcr(2:iebc,:)-...
  cbeybcr(2:iebc,:).*(hzxbcr(2:iebc,:)+hzybcr(2:iebc,:)-...
                hzxbcr(1:iebc-1,:)-hzybcr(1:iebc-1,:));
ey(ib,:)=caey(ib,:).*ey(ib,:)-...
  cbey(ib,:).*(hzxbcr(1,:)+hzybcr(1,:)- hz(ie,:));


%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

hz(1:ie,1:je)=dahz(1:ie,1:je).*hz(1:ie,1:je)+...
          dbhz(1:ie,1:je).*(ex(1:ie,2:jb)-ex(1:ie,1:je)+...
                        ey(1:ie,1:je)-ey(2:ib,1:je));

hz(is,js)=source(n);

hz(80,1:20)=0; %(added)
hz(80,30:je)=0; %(added)

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

hzxbcf(1:iefbc,:)=dahzxbcf(1:iefbc,:).*hzxbcf(1:iefbc,:)-...
  dbhzxbcf(1:iefbc,:).*(eybcf(2:ibfbc,:)-eybcf(1:iefbc,:));

%    BACK

hzxbcb(1:iefbc,:)=dahzxbcb(1:iefbc,:).*hzxbcb(1:iefbc,:)-...
  dbhzxbcb(1:iefbc,:).*(eybcb(2:ibfbc,:)-eybcb(1:iefbc,:));

%    LEFT

hzxbcl(1:iebc-1,:)=dahzxbcl(1:iebc-1,:).*hzxbcl(1:iebc-1,:)-...
  dbhzxbcl(1:iebc-1,:).*(eybcl(2:iebc,:)-eybcl(1:iebc-1,:));
hzxbcl(iebc,:)=dahzxbcl(iebc,:).*hzxbcl(iebc,:)-...
  dbhzxbcl(iebc,:).*(ey(1,:)-eybcl(iebc,:));

%    RIGHT

hzxbcr(2:iebc,:)=dahzxbcr(2:iebc,:).*hzxbcr(2:iebc,:)-...
  dbhzxbcr(2:iebc,:).*(eybcr(3:ibbc,:)-eybcr(2:iebc,:));
hzxbcr(1,:)=dahzxbcr(1,:).*hzxbcr(1,:)-...
  dbhzxbcr(1,:).*(eybcr(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

hzybcf(:,1:jebc-1)=dahzybcf(:,1:jebc-1).*hzybcf(:,1:jebc-1)-...
  dbhzybcf(:,1:jebc-1).*(exbcf(:,1:jebc-1)-exbcf(:,2:jebc));
hzybcf(1:iebc,jebc)=dahzybcf(1:iebc,jebc).*hzybcf(1:iebc,jebc)-...
  dbhzybcf(1:iebc,jebc).*(exbcf(1:iebc,jebc)-exbcl(1:iebc,1));
hzybcf(iebc+1:iebc+ie,jebc)=...
  dahzybcf(iebc+1:iebc+ie,jebc).*hzybcf(iebc+1:iebc+ie,jebc)-...
  dbhzybcf(iebc+1:iebc+ie,jebc).*(exbcf(iebc+1:iebc+ie,jebc)-...
                         ex(1:ie,1));
hzybcf(iebc+ie+1:iefbc,jebc)=...
  dahzybcf(iebc+ie+1:iefbc,jebc).*hzybcf(iebc+ie+1:iefbc,jebc)-...
  dbhzybcf(iebc+ie+1:iefbc,jebc).*(exbcf(iebc+ie+1:iefbc,jebc)-...
                          exbcr(1:iebc,1));

%    BACK
```

```
hzybcb(1:iefbc,2:jebc)=dahzybcb(1:iefbc,2:jebc).*hzybcb(1:iefbc,2:jebc)-...
  dbhzybcb(1:iefbc,2:jebc).*(exbcb(1:iefbc,2:jebc)-exbcb(1:iefbc,3:jbbc));
hzybcb(1:iebc,1)=dahzybcb(1:iebc,1).*hzybcb(1:iebc,1)-...
  dbhzybcb(1:iebc,1).*(exbcl(1:iebc,jb)-exbcb(1:iebc,2));
hzybcb(iebc+1:iebc+ie,1)=...
  dahzybcb(iebc+1:iebc+ie,1).*hzybcb(iebc+1:iebc+ie,1)-...
  dbhzybcb(iebc+1:iebc+ie,1).*(ex(1:ie,jb)-exbcb(iebc+1:iebc+ie,2));
hzybcb(iebc+ie+1:iefbc,1)=...
  dahzybcb(iebc+ie+1:iefbc,1).*hzybcb(iebc+ie+1:iefbc,1)-...
  dbhzybcb(iebc+ie+1:iefbc,1).*(exbcr(1:iebc,jb)-...
                              exbcb(iebc+ie+1:iefbc,2));

%     LEFT

hzybcl(:,1:je)=dahzybcl(:,1:je).*hzybcl(:,1:je)-...
  dbhzybcl(:,1:je).*(exbcl(:,1:je)-exbcl(:,2:jb));

%     RIGHT

hzybcr(:,1:je)=dahzybcr(:,1:je).*hzybcr(:,1:je)-...
  dbhzybcr(:,1:je).*(exbcr(:,1:je)-exbcr(:,2:jb));

%***********************************************************************
%     Visualize fields
%***********************************************************************

if mod(n,8)==0;

timestep=int2str(n);

subplot(3,1,1),pcolor(ex');
colormap hot %(added)
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = ',timestep]);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = ',timestep]);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = ',timestep]);

nn=n/8;
M(:,nn)=getframe(gcf,rect);

end;

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(gcf,M,0,10,rect);
```

```
%Filename: onewl3.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all %(added)
close all %(added)
pack %(added)
clc %(added)

%***********************************************************************
%     Fundamental constants
%***********************************************************************

cc=2.99792458e8;           %speed of light in free space
muz=4.0*pi*1.0e-7;         %permeability of free space
epsz=1.0/(cc*cc*muz);      %permittivity of free space

freq=10.0e+9;              %center frequency of source excitation
lambda=cc/freq;            %center wavelength of source excitation
omega=2.0*pi*freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

ie=100;          %number of grid cells in x-direction
je=50;           %number of grid cells in y-direction

ib=ie+1;
jb=je+1;

is=15;           %location of z-directed hard source
js=je/2;         %location of z-directed hard source

dx=3.0e-3;       %space increment of square lattice
dt=dx/(2.0*cc);  %time step

nmax=160; %(changed)        %total number of time steps

iebc=8;          %thickness of left and right PML region
jebc=8;          %thickness of front and back PML region
rmax=0.00001;
orderbc=2;
ibbc=iebc+1;
jbbc=jebc+1;
iefbc=ie+2*iebc;
jefbc=je+2*jebc;
ibfbc=iefbc+1;
jbfbc=jefbc+1;

%***********************************************************************
%     Material parameters
%***********************************************************************

media=2;

eps=[1.0 1.0];
sig=[0.0 1.0e+7];
mur=[1.0 1.0];
sim=[0.0 0.0];

%***********************************************************************
%     Wave excitation
%***********************************************************************
```

```
rtau=160.0e-12;
tau=rtau/dt;
delay=3*tau;

source=zeros(1,nmax);
for n=1:1*tau %(added)
  source(n)=sin(omega*dt*n); %(added)
end %(added)

%***********************************************************************
%     Field arrays
%***********************************************************************

ex=zeros(ie,jb);           %fields in main grid
ey=zeros(ib,je);
hz=zeros(ie,je);

exbcf=zeros(iefbc,jebc);   %fields in front PML region
eybcf=zeros(ibfbc,jebc);
hzxbcf=zeros(iefbc,jebc);
hzybcf=zeros(iefbc,jebc);

exbcb=zeros(iefbc,jbbc);   %fields in back PML region
eybcb=zeros(ibfbc,jebc);
hzxbcb=zeros(iefbc,jebc);
hzybcb=zeros(iefbc,jebc);

exbcl=zeros(iebc,jb);      %fields in left PML region
eybcl=zeros(iebc,je);
hzxbcl=zeros(iebc,je);
hzybcl=zeros(iebc,je);

exbcr=zeros(iebc,jb);      %fields in right PML region
eybcr=zeros(ibbc,je);
hzxbcr=zeros(iebc,je);
hzybcr=zeros(iebc,je);

%***********************************************************************
%     Updating coefficients
%***********************************************************************

for i=1:media
  eaf  =dt*sig(i)/(2.0*epsz*eps(i));
  ca(i)=(1.0-eaf)/(1.0+eaf);
  cb(i)=dt/epsz/eps(i)/dx/(1.0+eaf);
  haf  =dt*sim(i)/(2.0*muz*mur(i));
  da(i)=(1.0-haf)/(1.0+haf);
  db(i)=dt/muz/mur(i)/dx/(1.0+haf);
end

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

caex(1:ie,1:jb)=ca(1);
cbex(1:ie,1:jb)=cb(1);

caey(1:ib,1:je)=ca(1);
cbey(1:ib,1:je)=cb(1);

dahz(1:ie,1:je)=da(1);
dbhz(1:ie,1:je)=db(1);

%     Add PEC with slit of width equals one lambda
caex(80,[1:20])=ca(2); %(added)
caex(80,[30:je])=ca(2); %(added)
cbex(80,[1:20])=cb(2); %(added)
cbex(80,[30:je])=cb(2); %(added)
caey(80,[1:20])=ca(2); %(added)
caey(80,[30:je])=ca(2); %(added)
cbey(80,[1:20])=cb(2); %(added)
cbey(80,[30:je])=cb(2); %(added)
```

```
%***********************************************************************
%     Fill the PML regions
%***********************************************************************

delbc=iebc*dx;
sigmam=-log(rmax/100.0)*epsz*cc*(orderbc+1)/(2*delbc);
bcfactor=eps(1)*sigmam/(dx*(delbc^orderbc)*(orderbc+1));

%     FRONT region

caexbcf(1:iefbc,1)=1.0;
cbexbcf(1:iefbc,1)=0.0;
for j=2:jebc
  y1=(jebc-j+1.5)*dx;
  y2=(jebc-j+0.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1.0-ca1)/(sigmay*dx);
  caexbcf(1:iefbc,j)=ca1;
  cbexbcf(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,1)=ca1;
cbex(1:ie,1)=cb1;
caexbcl(1:iebc,1)=ca1;
cbexbcl(1:iebc,1)=cb1;
caexbcr(1:iebc,1)=ca1;
cbexbcr(1:iebc,1)=cb1;

for j=1:jebc
  y1=(jebc-j+1)*dx;
  y2=(jebc-j)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcf(1:iefbc,j)=da1;
  dbhzybcf(1:iefbc,j)=db1;
  caeybcf(1:ibfbc,j)=ca(1);
  cbeybcf(1:ibfbc,j)=cb(1);
  dahzxbcf(1:iefbc,j)=da(1);
  dbhzxbcf(1:iefbc,j)=db(1);
end

%     BACK region

caexbcb(1:iefbc,jbbc)=1.0;
cbexbcb(1:iefbc,jbbc)=0.0;
for j=2:jebc
  y1=(j-0.5)*dx;
  y2=(j-1.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmay*dx);
  caexbcb(1:iefbc,j)=ca1;
  cbexbcb(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,jb)=ca1;
cbex(1:ie,jb)=cb1;
caexbcl(1:iebc,jb)=ca1;
cbexbcl(1:iebc,jb)=cb1;
caexbcr(1:iebc,jb)=ca1;
cbexbcr(1:iebc,jb)=cb1;

for j=1:jebc
  y1=j*dx;
  y2=(j-1)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcb(1:iefbc,j)=da1;
  dbhzybcb(1:iefbc,j)=db1;
  caeybcb(1:ibfbc,j)=ca(1);
  cbeybcb(1:ibfbc,j)=cb(1);
  dahzxbcb(1:iefbc,j)=da(1);
  dbhzxbcb(1:iefbc,j)=db(1);
end

%     LEFT region

caeybcl(1,1:je)=1.0;
cbeybcl(1,1:je)=0.0;
for i=2:iebc
  x1=(iebc-i+1.5)*dx;
  x2=(iebc-i+0.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcl(i,1:je)=ca1;
  cbeybcl(i,1:je)=cb1;
  caeybcf(i,1:jebc)=ca1;
  cbeybcf(i,1:jebc)=cb1;
  caeybcb(i,1:jebc)=ca1;
  cbeybcb(i,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(1,1:je)=ca1;
cbey(1,1:je)=cb1;
caeybcf(iebc+1,1:jebc)=ca1;
cbeybcf(iebc+1,1:jebc)=cb1;
caeybcb(iebc+1,1:jebc)=ca1;
cbeybcb(iebc+1,1:jebc)=cb1;

for i=1:iebc
  x1=(iebc-i+1)*dx;
  x2=(iebc-i)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcl(i,1:je)=da1;
  dbhzxbcl(i,1:je)=db1;
  dahzxbcf(i,1:jebc)=da1;
  dbhzxbcf(i,1:jebc)=db1;
  dahzxbcb(i,1:jebc)=da1;
  dbhzxbcb(i,1:jebc)=db1;
  caexbcl(i,2:je)=ca(1);
  cbexbcl(i,2:je)=cb(1);
  dahzybcl(i,1:je)=da(1);
  dbhzybcl(i,1:je)=db(1);
end

%     RIGHT region

caeybcr(ibbc,1:je)=1.0;
cbeybcr(ibbc,1:je)=0.0;
for i=2:iebc
  x1=(i-0.5)*dx;
  x2=(i-1.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcr(i,1:je)=ca1;
  cbeybcr(i,1:je)=cb1;
  caeybcf(i+iebc+ie,1:jebc)=ca1;
  cbeybcf(i+iebc+ie,1:jebc)=cb1;
  caeybcb(i+iebc+ie,1:jebc)=ca1;
  cbeybcb(i+iebc+ie,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
```

```
caey(ib,1:je)=ca1;
cbey(ib,1:je)=cb1;
caeybcf(iebc+ib,1:jebc)=ca1;
cbeybcf(iebc+ib,1:jebc)=cb1;
caeybcb(iebc+ib,1:jebc)=ca1;
cbeybcb(iebc+ib,1:jebc)=cb1;

for i=1:iebc
  x1=i*dx;
  x2=(i-1)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcr(i,1:je) = da1;
  dbhzxbcr(i,1:je) = db1;
  dahzxbcf(i+ie+iebc,1:jebc)=da1;
  dbhzxbcf(i+ie+iebc,1:jebc)=db1;
  dahzxbcb(i+ie+iebc,1:jebc)=da1;
  dbhzxbcb(i+ie+iebc,1:jebc)=db1;
  caexbcr(i,2:je)=ca(1);
  cbexbcr(i,2:je)=cb(1);
  dahzybcr(i,1:je)=da(1);
  dbhzybcr(i,1:je)=db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

subplot(3,1,1),pcolor(ex');
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = 0']);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = 0']);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = 0']);

rect=get(gcf,'Position');
rect(1:2)=[0 0];

M=moviein(nmax/8,gcf,rect);

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n=1:nmax

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

ex(:,2:je)=caex(:,2:je).*ex(:,2:je)+...
           cbex(:,2:je).*(hz(:,2:je)-hz(:,1:je-1));
```

```
ey(2:ie,:)=caey(2:ie,:).*ey(2:ie,:)+...
          cbey(2:ie,:).*(hz(1:ie-1,:)-hz(2:ie,:));

ex(80,2:20)=0; %(added)
ex(80,30:je)=0; %(added)
ey(80,1:20)=0; %(added)
ey(80,30:je)=0; %(added)
%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

exbcf(:,2:jebc)=caexbcf(:,2:jebc).*exbcf(:,2:jebc)-...
  cbexbcf(:,2:jebc).*(hzxbcf(:,1:jebc-1)+hzybcf(:,1:jebc-1)-...
                    hzxbcf(:,2:jebc)-hzybcf(:,2:jebc));
ex(1:ie,1)=caex(1:ie,1).*ex(1:ie,1)-...
  cbex(1:ie,1).*(hzxbcf(ibbc:iebc+ie,jebc)+...
              hzybcf(ibbc:iebc+ie,jebc)-hz(1:ie,1));

%     BACK

exbcb(:,2:jebc-1)=caexbcb(:,2:jebc-1).*exbcb(:,2:jebc-1)-...
  cbexbcb(:,2:jebc-1).*(hzxbcb(:,1:jebc-2)+hzybcb(:,1:jebc-2)-...
                      hzxbcb(:,2:jebc-1)-hzybcb(:,2:jebc-1));
ex(1:ie,jb)=caex(1:ie,jb).*ex(1:ie,jb)-...
  cbex(1:ie,jb).*(hz(1:ie,jb-1)-hzxbcb(ibbc:iebc+ie,1)-...
               hzybcb(ibbc:iebc+ie,1));

%     LEFT

exbcl(:,2:je)=caexbcl(:,2:je).*exbcl(:,2:je)-...
  cbexbcl(:,2:je).*(hzxbcl(:,1:je-1)+hzybcl(:,1:je-1)-...
                  hzxbcl(:,2:je)-hzybcl(:,2:je));
exbcl(:,1)=caexbcl(:,1).*exbcl(:,1)-...
  cbexbcl(:,1).*(hzxbcf(1:iebc,jebc)+hzybcf(1:iebc,jebc)-...
               hzxbcl(:,1)-hzybcl(:,1));
exbcl(:,jb)=caexbcl(:,jb).*exbcl(:,jb)-...
  cbexbcl(:,jb).*(hzxbcl(:,je)+hzybcl(:,je)-...
                hzxbcb(1:iebc,1)-hzybcb(1:iebc,1));

%     RIGHT

exbcr(:,2:je)=caexbcr(:,2:je).*exbcr(:,2:je)-...
  cbexbcr(:,2:je).*(hzxbcr(:,1:je-1)+hzybcr(:,1:je-1)-...
                  hzxbcr(:,2:je)-hzybcr(:,2:je));
exbcr(:,1)=caexbcr(:,1).*exbcr(:,1)-...
  cbexbcr(:,1).*(hzxbcf(1+iebc+ie:iefbc,jebc)+...
               hzybcf(1+iebc+ie:iefbc,jebc)-...
               hzxbcr(:,1)-hzybcr(:,1));
exbcr(:,jb)=caexbcr(:,jb).*exbcr(:,jb)-...
  cbexbcr(:,jb).*(hzxbcr(:,je)+hzybcr(:,je)-...
                hzxbcb(1+iebc+ie:iefbc,1)-...
                hzybcb(1+iebc+ie:iefbc,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

eybcf(2:iefbc,:)=caeybcf(2:iefbc,:).*eybcf(2:iefbc,:)-...
  cbeybcf(2:iefbc,:).*(hzxbcf(2:iefbc,:)+hzybcf(2:iefbc,:)-...
                     hzxbcf(1:iefbc-1,:)-hzybcf(1:iefbc-1,:));

%     BACK

eybcb(2:iefbc,:)=caeybcb(2:iefbc,:).*eybcb(2:iefbc,:)-...
  cbeybcb(2:iefbc,:).*(hzxbcb(2:iefbc,:)+hzybcb(2:iefbc,:)-...
                     hzxbcb(1:iefbc-1,:)-hzybcb(1:iefbc-1,:));

%     LEFT

eybcl(2:iebc,:)=caeybcl(2:iebc,:).*eybcl(2:iebc,:)-...
  cbeybcl(2:iebc,:).*(hzxbcl(2:iebc,:)+hzybcl(2:iebc,:)-...
                    hzxbcl(1:iebc-1,:)-hzybcl(1:iebc-1,:));
ey(1,:)=caey(1,:).*ey(1,:)-...
  cbey(1,:).*(hz(1,:)-hzxbcl(iebc,:)-hzybcl(iebc,:));

%    RIGHT

eybcr(2:iebc,:)=caeybcr(2:iebc,:).*eybcr(2:iebc,:)-...
  cbeybcr(2:iebc,:).*(hzxbcr(2:iebc,:)+hzybcr(2:iebc,:)-...
                    hzxbcr(1:iebc-1,:)-hzybcr(1:iebc-1,:));
ey(ib,:)=caey(ib,:).*ey(ib,:)-...
  cbey(ib,:).*(hzxbcr(1,:)+hzybcr(1,:)- hz(ie,:));


%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

hz(1:ie,1:je)=dahz(1:ie,1:je).*hz(1:ie,1:je)+...
             dbhz(1:ie,1:je).*(ex(1:ie,2:jb)-ex(1:ie,1:je)+...
                             ey(1:ie,1:je)-ey(2:ib,1:je));

hz(15,1:10:je)=source(n); %(changed)

hz(80,1:20)=0; %(added)
hz(80,30:je)=0; %(added)

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

hzxbcf(1:iefbc,:)=dahzxbcf(1:iefbc,:).*hzxbcf(1:iefbc,:)-...
  dbhzxbcf(1:iefbc,:).*(eybcf(2:ibfbc,:)-eybcf(1:iefbc,:));

%    BACK

hzxbcb(1:iefbc,:)=dahzxbcb(1:iefbc,:).*hzxbcb(1:iefbc,:)-...
  dbhzxbcb(1:iefbc,:).*(eybcb(2:ibfbc,:)-eybcb(1:iefbc,:));

%    LEFT

hzxbcl(1:iebc-1,:)=dahzxbcl(1:iebc-1,:).*hzxbcl(1:iebc-1,:)-...
  dbhzxbcl(1:iebc-1,:).*(eybcl(2:iebc,:)-eybcl(1:iebc-1,:));
hzxbcl(iebc,:)=dahzxbcl(iebc,:).*hzxbcl(iebc,:)-...
  dbhzxbcl(iebc,:).*(ey(1,:)-eybcl(iebc,:));

%    RIGHT

hzxbcr(2:iebc,:)=dahzxbcr(2:iebc,:).*hzxbcr(2:iebc,:)-...
  dbhzxbcr(2:iebc,:).*(eybcr(3:ibbc,:)-eybcr(2:iebc,:));
hzxbcr(1,:)=dahzxbcr(1,:).*hzxbcr(1,:)-...
  dbhzxbcr(1,:).*(eybcr(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

hzybcf(:,1:jebc-1)=dahzybcf(:,1:jebc-1).*hzybcf(:,1:jebc-1)-...
  dbhzybcf(:,1:jebc-1).*(exbcf(:,1:jebc-1)-exbcf(:,2:jebc));
hzybcf(1:iebc,jebc)=dahzybcf(1:iebc,jebc).*hzybcf(1:iebc,jebc)-...
  dbhzybcf(1:iebc,jebc).*(exbcf(1:iebc,jebc)-exbcl(1:iebc,1));
hzybcf(iebc+1:iebc+ie,jebc)=...
  dahzybcf(iebc+1:iebc+ie,jebc).*hzybcf(iebc+1:iebc+ie,jebc)-...
  dbhzybcf(iebc+1:iebc+ie,jebc).*(exbcf(iebc+1:iebc+ie,jebc)-...
                                 ex(1:ie,1));
hzybcf(iebc+ie+1:iefbc,jebc)=...
  dahzybcf(iebc+ie+1:iefbc,jebc).*hzybcf(iebc+ie+1:iefbc,jebc)-...
  dbhzybcf(iebc+ie+1:iefbc,jebc).*(exbcf(iebc+ie+1:iefbc,jebc)-...
                                  exbcr(1:iebc,1));

%    BACK
```

```
hzybcb(1:iefbc,2:jebc)=dahzybcb(1:iefbc,2:jebc).*hzybcb(1:iefbc,2:jebc)-...
  dbhzybcb(1:iefbc,2:jebc).*(exbcb(1:iefbc,2:jebc)-exbcb(1:iefbc,3:jbbc));
hzybcb(1:iebc,1)=dahzybcb(1:iebc,1).*hzybcb(1:iebc,1)-...
  dbhzybcb(1:iebc,1).*(exbcl(1:iebc,jb)-exbcb(1:iebc,2));
hzybcb(iebc+1:iebc+ie,1)=...
  dahzybcb(iebc+1:iebc+ie,1).*hzybcb(iebc+1:iebc+ie,1)-...
  dbhzybcb(iebc+1:iebc+ie,1).*(ex(1:ie,jb)-exbcb(iebc+1:iebc+ie,2));
hzybcb(iebc+ie+1:iefbc,1)=...
  dahzybcb(iebc+ie+1:iefbc,1).*hzybcb(iebc+ie+1:iefbc,1)-...
  dbhzybcb(iebc+ie+1:iefbc,1).*(exbcr(1:iebc,jb)-...
                              exbcb(iebc+ie+1:iefbc,2));

%     LEFT

hzybcl(:,1:je)=dahzybcl(:,1:je).*hzybcl(:,1:je)-...
  dbhzybcl(:,1:je).*(exbcl(:,1:je)-exbcl(:,2:jb));

%     RIGHT

hzybcr(:,1:je)=dahzybcr(:,1:je).*hzybcr(:,1:je)-...
  dbhzybcr(:,1:je).*(exbcr(:,1:je)-exbcr(:,2:jb));

%***********************************************************************
%     Visualize fields
%***********************************************************************

if mod(n,8)==0;

timestep=int2str(n);

subplot(3,1,1),pcolor(ex');
colormap hot %(added)
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = ',timestep]);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = ',timestep]);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = ',timestep]);

nn=n/8;
M(:,nn)=getframe(gcf,rect);

end;

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(gcf,M,0,10,rect);
```

```
%Filename: onewl4.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%            Department of Electrical and Computer Engineering
%            University of Wisconsin-Madison
%            1415 Engineering Drive
%            Madison, WI 53706-1691
%            608-265-5739
%            hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all %(added)
close all %(added)
pack %(added)
clc %(added)

%***********************************************************************
%     Fundamental constants
%***********************************************************************

cc=2.99792458e8;            %speed of light in free space
muz=4.0*pi*1.0e-7;          %permeability of free space
epsz=1.0/(cc*cc*muz);       %permittivity of free space

freq=5.0e+9;                %center frequency of source excitation
lambda=cc/freq;             %center wavelength of source excitation
omega=2.0*pi*freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

ie=100;           %number of grid cells in x-direction
je=50;            %number of grid cells in y-direction

ib=ie+1;
jb=je+1;

is=15;            %location of z-directed hard source
js=je/2;          %location of z-directed hard source

dx=3.0e-3;        %space increment of square lattice
dt=dx/(2.0*cc);   %time step

nmax=160; %(changed)        %total number of time steps

iebc=8;           %thickness of left and right PML region
jebc=8;           %thickness of front and back PML region
rmax=0.00001;
orderbc=2;
ibbc=iebc+1;
jbbc=jebc+1;
iefbc=ie+2*iebc;
jefbc=je+2*jebc;
ibfbc=iefbc+1;
jbfbc=jefbc+1;

%***********************************************************************
%     Material parameters
%***********************************************************************

media=2;

eps=[1.0 1.0];
sig=[0.0 1.0e+7];
mur=[1.0 1.0];
sim=[0.0 0.0];

%***********************************************************************
%     Wave excitation
%***********************************************************************
```

```
rtau=160.0e-12;
tau=rtau/dt;
delay=3*tau;

source=zeros(1,nmax);
spacing=10;%set wavelength to be 10-cell  %(added)
sign_bit=logical(1);%set the initial sign of a number %(added)
for n=1:spacing:1*tau %(added)
   source(n)=(-1)^(n+sign_bit); %(added)
   sign_bit = ~sign_bit; %(added)
end

%***********************************************************************
%     Field arrays
%***********************************************************************

ex=zeros(ie,jb);           %fields in main grid
ey=zeros(ib,je);
hz=zeros(ie,je);

exbcf=zeros(iefbc,jebc);   %fields in front PML region
eybcf=zeros(ibfbc,jebc);
hzxbcf=zeros(iefbc,jebc);
hzybcf=zeros(iefbc,jebc);

exbcb=zeros(iefbc,jbbc);   %fields in back PML region
eybcb=zeros(ibfbc,jebc);
hzxbcb=zeros(iefbc,jebc);
hzybcb=zeros(iefbc,jebc);

exbcl=zeros(iebc,jb);      %fields in left PML region
eybcl=zeros(iebc,je);
hzxbcl=zeros(iebc,je);
hzybcl=zeros(iebc,je);

exbcr=zeros(iebc,jb);      %fields in right PML region
eybcr=zeros(ibbc,je);
hzxbcr=zeros(iebc,je);
hzybcr=zeros(iebc,je);

%***********************************************************************
%     Updating coefficients
%***********************************************************************

for i=1:media
  eaf  =dt*sig(i)/(2.0*epsz*eps(i));
  ca(i)=(1.0-eaf)/(1.0+eaf);
  cb(i)=dt/epsz/eps(i)/dx/(1.0+eaf);
  haf  =dt*sim(i)/(2.0*muz*mur(i));
  da(i)=(1.0-haf)/(1.0+haf);
  db(i)=dt/muz/mur(i)/dx/(1.0+haf);
end

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

caex(1:ie,1:jb)=ca(1);
cbex(1:ie,1:jb)=cb(1);

caey(1:ib,1:je)=ca(1);
cbey(1:ib,1:je)=cb(1);

dahz(1:ie,1:je)=da(1);
dbhz(1:ie,1:je)=db(1);

%     Add PEC with slit of width equals one lambda
caex(80,[1:20])=ca(2); %(added)
caex(80,[30:je])=ca(2); %(added)
cbex(80,[1:20])=cb(2); %(added)
cbex(80,[30:je])=cb(2); %(added)
caey(80,[1:20])=ca(2); %(added)
caey(80,[30:je])=ca(2); %(added)
```

```
cbey(80,[1:20])=cb(2); %(added)
cbey(80,[30:je])=cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
%***********************************************************************

delbc=iebc*dx;
sigmam=-log(rmax/100.0)*epsz*cc*(orderbc+1)/(2*delbc);
bcfactor=eps(1)*sigmam/(dx*(delbc^orderbc)*(orderbc+1));

%     FRONT region

caexbcf(1:iefbc,1)=1.0;
cbexbcf(1:iefbc,1)=0.0;
for j=2:jebc
  y1=(jebc-j+1.5)*dx;
  y2=(jebc-j+0.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1.0-ca1)/(sigmay*dx);
  caexbcf(1:iefbc,j)=ca1;
  cbexbcf(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,1)=ca1;
cbex(1:ie,1)=cb1;
caexbcl(1:iebc,1)=ca1;
cbexbcl(1:iebc,1)=cb1;
caexbcr(1:iebc,1)=ca1;
cbexbcr(1:iebc,1)=cb1;

for j=1:jebc
  y1=(jebc-j+1)*dx;
  y2=(jebc-j)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcf(1:iefbc,j)=da1;
  dbhzybcf(1:iefbc,j)=db1;
  caeybcf(1:ibfbc,j)=ca(1);
  cbeybcf(1:ibfbc,j)=cb(1);
  dahzxbcf(1:iefbc,j)=da(1);
  dbhzxbcf(1:iefbc,j)=db(1);
end

%     BACK region

caexbcb(1:iefbc,jbbc)=1.0;
cbexbcb(1:iefbc,jbbc)=0.0;
for j=2:jebc
  y1=(j-0.5)*dx;
  y2=(j-1.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmay*dx);
  caexbcb(1:iefbc,j)=ca1;
  cbexbcb(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,jb)=ca1;
cbex(1:ie,jb)=cb1;
caexbcl(1:iebc,jb)=ca1;
cbexbcl(1:iebc,jb)=cb1;
caexbcr(1:iebc,jb)=ca1;
cbexbcr(1:iebc,jb)=cb1;

for j=1:jebc
  y1=j*dx;
  y2=(j-1)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcb(1:iefbc,j)=da1;
  dbhzybcb(1:iefbc,j)=db1;
  caeybcb(1:ibfbc,j)=ca(1);
  cbeybcb(1:ibfbc,j)=cb(1);
  dahzxbcb(1:iefbc,j)=da(1);
  dbhzxbcb(1:iefbc,j)=db(1);
end

%     LEFT region

caeybcl(1,1:je)=1.0;
cbeybcl(1,1:je)=0.0;
for i=2:iebc
  x1=(iebc-i+1.5)*dx;
  x2=(iebc-i+0.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcl(i,1:je)=ca1;
  cbeybcl(i,1:je)=cb1;
  caeybcf(i,1:jebc)=ca1;
  cbeybcf(i,1:jebc)=cb1;
  caeybcb(i,1:jebc)=ca1;
  cbeybcb(i,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(1,1:je)=ca1;
cbey(1,1:je)=cb1;
caeybcf(iebc+1,1:jebc)=ca1;
cbeybcf(iebc+1,1:jebc)=cb1;
caeybcb(iebc+1,1:jebc)=ca1;
cbeybcb(iebc+1,1:jebc)=cb1;

for i=1:iebc
  x1=(iebc-i+1)*dx;
  x2=(iebc-i)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcl(i,1:je)=da1;
  dbhzxbcl(i,1:je)=db1;
  dahzxbcf(i,1:jebc)=da1;
  dbhzxbcf(i,1:jebc)=db1;
  dahzxbcb(i,1:jebc)=da1;
  dbhzxbcb(i,1:jebc)=db1;
  caexbcl(i,2:je)=ca(1);
  cbexbcl(i,2:je)=cb(1);
  dahzybcl(i,1:je)=da(1);
  dbhzybcl(i,1:je)=db(1);
end

%     RIGHT region

caeybcr(ibbc,1:je)=1.0;
cbeybcr(ibbc,1:je)=0.0;
for i=2:iebc
  x1=(i-0.5)*dx;
  x2=(i-1.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcr(i,1:je)=ca1;
  cbeybcr(i,1:je)=cb1;
  caeybcf(i+iebc+ie,1:jebc)=ca1;
  cbeybcf(i+iebc+ie,1:jebc)=cb1;
  caeybcb(i+iebc+ie,1:jebc)=ca1;
  cbeybcb(i+iebc+ie,1:jebc)=cb1;
end
```

```
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(ib,1:je)=ca1;
cbey(ib,1:je)=cb1;
caeybcf(iebc+ib,1:jebc)=ca1;
cbeybcf(iebc+ib,1:jebc)=cb1;
caeybcb(iebc+ib,1:jebc)=ca1;
cbeybcb(iebc+ib,1:jebc)=cb1;

for i=1:iebc
  x1=i*dx;
  x2=(i-1)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcr(i,1:je) = da1;
  dbhzxbcr(i,1:je) = db1;
  dahzxbcf(i+ie+iebc,1:jebc)=da1;
  dbhzxbcf(i+ie+iebc,1:jebc)=db1;
  dahzxbcb(i+ie+iebc,1:jebc)=da1;
  dbhzxbcb(i+ie+iebc,1:jebc)=db1;
  caexbcr(i,2:je)=ca(1);
  cbexbcr(i,2:je)=cb(1);
  dahzybcr(i,1:je)=da(1);
  dbhzybcr(i,1:je)=db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

subplot(3,1,1),pcolor(ex');
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = 0']);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = 0']);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = 0']);

rect=get(gcf,'Position');
rect(1:2)=[0 0];

M=moviein(nmax/8,gcf,rect);

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n=1:nmax

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************
```

```
ex(:,2:je)=caex(:,2:je).*ex(:,2:je)+...
          cbex(:,2:je).*(hz(:,2:je)-hz(:,1:je-1));

ey(2:ie,:)=caey(2:ie,:).*ey(2:ie,:)+...
          cbey(2:ie,:).*(hz(1:ie-1,:)-hz(2:ie,:));

ex(80,2:20)=0; %(added)
ex(80,30:je)=0; %(added)
ey(80,1:20)=0; %(added)
ey(80,30:je)=0; %(added)
%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

exbcf(:,2:jebc)=caexbcf(:,2:jebc).*exbcf(:,2:jebc)-...
  cbexbcf(:,2:jebc).*(hzxbcf(:,1:jebc-1)+hzybcf(:,1:jebc-1)-...
                    hzxbcf(:,2:jebc)-hzybcf(:,2:jebc));
ex(1:ie,1)=caex(1:ie,1).*ex(1:ie,1)-...
  cbex(1:ie,1).*(hzxbcf(ibbc:iebc+ie,jebc)+...
              hzybcf(ibbc:iebc+ie,jebc)-hz(1:ie,1));

%     BACK

exbcb(:,2:jebc-1)=caexbcb(:,2:jebc-1).*exbcb(:,2:jebc-1)-...
  cbexbcb(:,2:jebc-1).*(hzxbcb(:,1:jebc-2)+hzybcb(:,1:jebc-2)-...
                    hzxbcb(:,2:jebc-1)-hzybcb(:,2:jebc-1));
ex(1:ie,jb)=caex(1:ie,jb).*ex(1:ie,jb)-...
  cbex(1:ie,jb).*(hz(1:ie,jb-1)-hzxbcb(ibbc:iebc+ie,1)-...
              hzybcb(ibbc:iebc+ie,1));

%     LEFT

exbcl(:,2:je)=caexbcl(:,2:je).*exbcl(:,2:je)-...
  cbexbcl(:,2:je).*(hzxbcl(:,1:je-1)+hzybcl(:,1:je-1)-...
                hzxbcl(:,2:je)-hzybcl(:,2:je));
exbcl(:,1)=caexbcl(:,1).*exbcl(:,1)-...
  cbexbcl(:,1).*(hzxbcf(1:iebc,jebc)+hzybcf(1:iebc,jebc)-...
              hzxbcl(:,1)-hzybcl(:,1));
exbcl(:,jb)=caexbcl(:,jb).*exbcl(:,jb)-...
  cbexbcl(:,jb).*(hzxbcl(:,je)+hzybcl(:,je)-...
               hzxbcb(1:iebc,1)-hzybcb(1:iebc,1));

%     RIGHT

exbcr(:,2:je)=caexbcr(:,2:je).*exbcr(:,2:je)-...
  cbexbcr(:,2:je).*(hzxbcr(:,1:je-1)+hzybcr(:,1:je-1)-...
                hzxbcr(:,2:je)-hzybcr(:,2:je));
exbcr(:,1)=caexbcr(:,1).*exbcr(:,1)-...
  cbexbcr(:,1).*(hzxbcf(1+iebc+ie:iefbc,jebc)+...
              hzybcf(1+iebc+ie:iefbc,jebc)-...
              hzxbcr(:,1)-hzybcr(:,1));
exbcr(:,jb)=caexbcr(:,jb).*exbcr(:,jb)-...
  cbexbcr(:,jb).*(hzxbcr(:,je)+hzybcr(:,je)-...
               hzxbcb(1+iebc+ie:iefbc,1)-...
               hzybcb(1+iebc+ie:iefbc,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

eybcf(2:iefbc,:)=caeybcf(2:iefbc,:).*eybcf(2:iefbc,:)-...
  cbeybcf(2:iefbc,:).*(hzxbcf(2:iefbc,:)+hzybcf(2:iefbc,:)-...
                    hzxbcf(1:iefbc-1,:)-hzybcf(1:iefbc-1,:));

%     BACK

eybcb(2:iefbc,:)=caeybcb(2:iefbc,:).*eybcb(2:iefbc,:)-...
  cbeybcb(2:iefbc,:).*(hzxbcb(2:iefbc,:)+hzybcb(2:iefbc,:)-...
                    hzxbcb(1:iefbc-1,:)-hzybcb(1:iefbc-1,:));

%     LEFT
```

```
eybcl(2:iebc,:)=caeybcl(2:iebc,:).*eybcl(2:iebc,:)-...
  cbeybcl(2:iebc,:).*(hzxbcl(2:iebc,:)+hzybcl(2:iebc,:)-...
                  hzxbcl(1:iebc-1,:)-hzybcl(1:iebc-1,:));
ey(1,:)=caey(1,:).*ey(1,:)-...
  cbey(1,:).*(hz(1,:)-hzxbcl(iebc,:)-hzybcl(iebc,:));

%    RIGHT

eybcr(2:iebc,:)=caeybcr(2:iebc,:).*eybcr(2:iebc,:)-...
  cbeybcr(2:iebc,:).*(hzxbcr(2:iebc,:)+hzybcr(2:iebc,:)-...
                  hzxbcr(1:iebc-1,:)-hzybcr(1:iebc-1,:));
ey(ib,:)=caey(ib,:).*ey(ib,:)-...
  cbey(ib,:).*(hzxbcr(1,:)+hzybcr(1,:)- hz(ie,:));


%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

hz(1:ie,1:je)=dahz(1:ie,1:je).*hz(1:ie,1:je)+...
            dbhz(1:ie,1:je).*(ex(1:ie,2:jb)-ex(1:ie,1:je)+...
                            ey(1:ie,1:je)-ey(2:ib,1:je));

hz(15,1:je)=source(n); %(changed)

hz(80,1:20)=0; %(added)
hz(80,30:je)=0; %(added)

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

hzxbcf(1:iefbc,:)=dahzxbcf(1:iefbc,:).*hzxbcf(1:iefbc,:)-...
  dbhzxbcf(1:iefbc,:).*(eybcf(2:ibfbc,:)-eybcf(1:iefbc,:));

%    BACK

hzxbcb(1:iefbc,:)=dahzxbcb(1:iefbc,:).*hzxbcb(1:iefbc,:)-...
  dbhzxbcb(1:iefbc,:).*(eybcb(2:ibfbc,:)-eybcb(1:iefbc,:));

%    LEFT

hzxbcl(1:iebc-1,:)=dahzxbcl(1:iebc-1,:).*hzxbcl(1:iebc-1,:)-...
  dbhzxbcl(1:iebc-1,:).*(eybcl(2:iebc,:)-eybcl(1:iebc-1,:));
hzxbcl(iebc,:)=dahzxbcl(iebc,:).*hzxbcl(iebc,:)-...
  dbhzxbcl(iebc,:).*(ey(1,:)-eybcl(iebc,:));

%    RIGHT

hzxbcr(2:iebc,:)=dahzxbcr(2:iebc,:).*hzxbcr(2:iebc,:)-...
  dbhzxbcr(2:iebc,:).*(eybcr(3:ibbc,:)-eybcr(2:iebc,:));
hzxbcr(1,:)=dahzxbcr(1,:).*hzxbcr(1,:)-...
  dbhzxbcr(1,:).*(eybcr(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

hzybcf(:,1:jebc-1)=dahzybcf(:,1:jebc-1).*hzybcf(:,1:jebc-1)-...
  dbhzybcf(:,1:jebc-1).*(exbcf(:,1:jebc-1)-exbcf(:,2:jebc));
hzybcf(1:iebc,jebc)=dahzybcf(1:iebc,jebc).*hzybcf(1:iebc,jebc)-...
  dbhzybcf(1:iebc,jebc).*(exbcf(1:iebc,jebc)-exbcl(1:iebc,1));
hzybcf(iebc+1:iebc+ie,jebc)=...
  dahzybcf(iebc+1:iebc+ie,jebc).*hzybcf(iebc+1:iebc+ie,jebc)-...
  dbhzybcf(iebc+1:iebc+ie,jebc).*(exbcf(iebc+1:iebc+ie,jebc)-...
                              ex(1:ie,1));
hzybcf(iebc+ie+1:iefbc,jebc)=...
  dahzybcf(iebc+ie+1:iefbc,jebc).*hzybcf(iebc+ie+1:iefbc,jebc)-...
  dbhzybcf(iebc+ie+1:iefbc,jebc).*(exbcf(iebc+ie+1:iefbc,jebc)-...
                              exbcr(1:iebc,1));
```

```
%    BACK

hzybcb(1:iefbc,2:jebc)=dahzybcb(1:iefbc,2:jebc).*hzybcb(1:iefbc,2:jebc)-...
  dbhzybcb(1:iefbc,2:jebc).*(exbcb(1:iefbc,2:jebc)-exbcb(1:iefbc,3:jbbc));
hzybcb(1:iebc,1)=dahzybcb(1:iebc,1).*hzybcb(1:iebc,1)-...
  dbhzybcb(1:iebc,1).*(exbcl(1:iebc,jb)-exbcb(1:iebc,2));
hzybcb(iebc+1:iebc+ie,1)=...
  dahzybcb(iebc+1:iebc+ie,1).*hzybcb(iebc+1:iebc+ie,1)-...
  dbhzybcb(iebc+1:iebc+ie,1).*(ex(1:ie,jb)-exbcb(iebc+1:iebc+ie,2));
hzybcb(iebc+ie+1:iefbc,1)=...
  dahzybcb(iebc+ie+1:iefbc,1).*hzybcb(iebc+ie+1:iefbc,1)-...
  dbhzybcb(iebc+ie+1:iefbc,1).*(exbcr(1:iebc,jb)-...
                              exbcb(iebc+ie+1:iefbc,2));

%    LEFT

hzybcl(:,1:je)=dahzybcl(:,1:je).*hzybcl(:,1:je)-...
  dbhzybcl(:,1:je).*(exbcl(:,1:je)-exbcl(:,2:jb));

%    RIGHT

hzybcr(:,1:je)=dahzybcr(:,1:je).*hzybcr(:,1:je)-...
  dbhzybcr(:,1:je).*(exbcr(:,1:je)-exbcr(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************

if mod(n,8)==0;

timestep=int2str(n);

subplot(3,1,1),pcolor(ex');
colormap hot %(added)
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = ',timestep]);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = ',timestep]);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = ',timestep]);

nn=n/8;
M(:,nn)=getframe(gcf,rect);

end;

%***********************************************************************
%    END TIME-STEPPING LOOP
%***********************************************************************

end

movie(gcf,M,0,10,rect);
```

```
%Filename: twowl2.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%            Department of Electrical and Computer Engineering
%            University of Wisconsin-Madison
%            1415 Engineering Drive
%            Madison, WI 53706-1691
%            608-265-5739
%            hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all %(added)
close all %(added)
pack %(added)
clc %(added)

%***********************************************************************
%    Fundamental constants
%***********************************************************************

cc=2.99792458e8;           %speed of light in free space
muz=4.0*pi*1.0e-7;         %permeability of free space
epsz=1.0/(cc*cc*muz);       %permittivity of free space

freq=5.0e+9;               %center frequency of source excitation
lambda=cc/freq;            %center wavelength of source excitation
omega=2.0*pi*freq;

%***********************************************************************
%    Grid parameters
%***********************************************************************

ie=100;          %number of grid cells in x-direction
je=50;           %number of grid cells in y-direction

ib=ie+1;
jb=je+1;

is=15;           %location of z-directed hard source
js=je/2;         %location of z-directed hard source

dx=3.0e-3;        %space increment of square lattice
dt=dx/(2.0*cc);   %time step

nmax=240; %(changed)        %total number of time steps

iebc=8;          %thickness of left and right PML region
jebc=8;          %thickness of front and back PML region
rmax=0.00001;
orderbc=2;
ibbc=iebc+1;
jbbc=jebc+1;
iefbc=ie+2*iebc;
jefbc=je+2*jebc;
ibfbc=iefbc+1;
jbfbc=jefbc+1;

%***********************************************************************
%    Material parameters
%***********************************************************************

media=2;

eps=[1.0 1.0];
sig=[0.0 1.0e+7];
mur=[1.0 1.0];
sim=[0.0 0.0];

%***********************************************************************
%    Wave excitation
%***********************************************************************
```

```
rtau=160.0e-12;
tau=rtau/dt;
delay=3*tau;

source=zeros(1,nmax);
for n=1:4*tau
  source(n)=sin(omega*(n-delay)*dt)*exp(-((n-delay)^2/tau^2));
end

%***********************************************************************
%     Field arrays
%***********************************************************************

ex=zeros(ie,jb);           %fields in main grid
ey=zeros(ib,je);
hz=zeros(ie,je);

exbcf=zeros(iefbc,jebc);   %fields in front PML region
eybcf=zeros(ibfbc,jebc);
hzxbcf=zeros(iefbc,jebc);
hzybcf=zeros(iefbc,jebc);

exbcb=zeros(iefbc,jbbc);   %fields in back PML region
eybcb=zeros(ibfbc,jebc);
hzxbcb=zeros(iefbc,jebc);
hzybcb=zeros(iefbc,jebc);

exbcl=zeros(iebc,jb);      %fields in left PML region
eybcl=zeros(iebc,je);
hzxbcl=zeros(iebc,je);
hzybcl=zeros(iebc,je);

exbcr=zeros(iebc,jb);      %fields in right PML region
eybcr=zeros(ibbc,je);
hzxbcr=zeros(iebc,je);
hzybcr=zeros(iebc,je);

%***********************************************************************
%     Updating coefficients
%***********************************************************************

for i=1:media
  eaf  =dt*sig(i)/(2.0*epsz*eps(i));
  ca(i)=(1.0-eaf)/(1.0+eaf);
  cb(i)=dt/epsz/eps(i)/dx/(1.0+eaf);
  haf  =dt*sim(i)/(2.0*muz*mur(i));
  da(i)=(1.0-haf)/(1.0+haf);
  db(i)=dt/muz/mur(i)/dx/(1.0+haf);
end

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

caex(1:ie,1:jb)=ca(1);
cbex(1:ie,1:jb)=cb(1);

caey(1:ib,1:je)=ca(1);
cbey(1:ib,1:je)=cb(1);

dahz(1:ie,1:je)=da(1);
dbhz(1:ie,1:je)=db(1);

%     Add PEC with slit of width equals two lambda
caex(80,[1:15])=ca(2); %(added)
caex(80,[35:je])=ca(2); %(added)
cbex(80,[1:15])=cb(2); %(added)
cbex(80,[35:je])=cb(2); %(added)
caey(80,[1:15])=ca(2); %(added)
caey(80,[35:je])=ca(2); %(added)
cbey(80,[1:15])=cb(2); %(added)
cbey(80,[35:je])=cb(2); %(added)
```

```
%***********************************************************************
%     Fill the PML regions
%***********************************************************************

delbc=iebc*dx;
sigmam=-log(rmax/100.0)*epsz*cc*(orderbc+1)/(2*delbc);
bcfactor=eps(1)*sigmam/(dx*(delbc^orderbc)*(orderbc+1));

%     FRONT region

caexbcf(1:iefbc,1)=1.0;
cbexbcf(1:iefbc,1)=0.0;
for j=2:jebc
  y1=(jebc-j+1.5)*dx;
  y2=(jebc-j+0.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1.0-ca1)/(sigmay*dx);
  caexbcf(1:iefbc,j)=ca1;
  cbexbcf(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,1)=ca1;
cbex(1:ie,1)=cb1;
caexbcl(1:iebc,1)=ca1;
cbexbcl(1:iebc,1)=cb1;
caexbcr(1:iebc,1)=ca1;
cbexbcr(1:iebc,1)=cb1;

for j=1:jebc
  y1=(jebc-j+1)*dx;
  y2=(jebc-j)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcf(1:iefbc,j)=da1;
  dbhzybcf(1:iefbc,j)=db1;
  caeybcf(1:ibfbc,j)=ca(1);
  cbeybcf(1:ibfbc,j)=cb(1);
  dahzxbcf(1:iefbc,j)=da(1);
  dbhzxbcf(1:iefbc,j)=db(1);
end

%     BACK region

caexbcb(1:iefbc,jbbc)=1.0;
cbexbcb(1:iefbc,jbbc)=0.0;
for j=2:jebc
  y1=(j-0.5)*dx;
  y2=(j-1.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmay*dx);
  caexbcb(1:iefbc,j)=ca1;
  cbexbcb(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,jb)=ca1;
cbex(1:ie,jb)=cb1;
caexbcl(1:iebc,jb)=ca1;
cbexbcl(1:iebc,jb)=cb1;
caexbcr(1:iebc,jb)=ca1;
cbexbcr(1:iebc,jb)=cb1;

for j=1:jebc
  y1=j*dx;
  y2=(j-1)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcb(1:iefbc,j)=da1;
  dbhzybcb(1:iefbc,j)=db1;
  caeybcb(1:ibfbc,j)=ca(1);
  cbeybcb(1:ibfbc,j)=cb(1);
  dahzxbcb(1:iefbc,j)=da(1);
  dbhzxbcb(1:iefbc,j)=db(1);
end

%     LEFT region

caeybcl(1,1:je)=1.0;
cbeybcl(1,1:je)=0.0;
for i=2:iebc
  x1=(iebc-i+1.5)*dx;
  x2=(iebc-i+0.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcl(i,1:je)=ca1;
  cbeybcl(i,1:je)=cb1;
  caeybcf(i,1:jebc)=ca1;
  cbeybcf(i,1:jebc)=cb1;
  caeybcb(i,1:jebc)=ca1;
  cbeybcb(i,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(1,1:je)=ca1;
cbey(1,1:je)=cb1;
caeybcf(iebc+1,1:jebc)=ca1;
cbeybcf(iebc+1,1:jebc)=cb1;
caeybcb(iebc+1,1:jebc)=ca1;
cbeybcb(iebc+1,1:jebc)=cb1;

for i=1:iebc
  x1=(iebc-i+1)*dx;
  x2=(iebc-i)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcl(i,1:je)=da1;
  dbhzxbcl(i,1:je)=db1;
  dahzxbcf(i,1:jebc)=da1;
  dbhzxbcf(i,1:jebc)=db1;
  dahzxbcb(i,1:jebc)=da1;
  dbhzxbcb(i,1:jebc)=db1;
  caexbcl(i,2:je)=ca(1);
  cbexbcl(i,2:je)=cb(1);
  dahzybcl(i,1:je)=da(1);
  dbhzybcl(i,1:je)=db(1);
end

%     RIGHT region

caeybcr(ibbc,1:je)=1.0;
cbeybcr(ibbc,1:je)=0.0;
for i=2:iebc
  x1=(i-0.5)*dx;
  x2=(i-1.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcr(i,1:je)=ca1;
  cbeybcr(i,1:je)=cb1;
  caeybcf(i+iebc+ie,1:jebc)=ca1;
  cbeybcf(i+iebc+ie,1:jebc)=cb1;
  caeybcb(i+iebc+ie,1:jebc)=ca1;
  cbeybcb(i+iebc+ie,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
```

```
caey(ib,1:je)=ca1;
cbey(ib,1:je)=cb1;
caeybcf(iebc+ib,1:jebc)=ca1;
cbeybcf(iebc+ib,1:jebc)=cb1;
caeybcb(iebc+ib,1:jebc)=ca1;
cbeybcb(iebc+ib,1:jebc)=cb1;

for i=1:iebc
  x1=i*dx;
  x2=(i-1)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcr(i,1:je) = da1;
  dbhzxbcr(i,1:je) = db1;
  dahzxbcf(i+ie+iebc,1:jebc)=da1;
  dbhzxbcf(i+ie+iebc,1:jebc)=db1;
  dahzxbcb(i+ie+iebc,1:jebc)=da1;
  dbhzxbcb(i+ie+iebc,1:jebc)=db1;
  caexbcr(i,2:je)=ca(1);
  cbexbcr(i,2:je)=cb(1);
  dahzybcr(i,1:je)=da(1);
  dbhzybcr(i,1:je)=db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

subplot(3,1,1),pcolor(ex');
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = 0']);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = 0']);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = 0']);

rect=get(gcf,'Position');
rect(1:2)=[0 0];

M=moviein(nmax/8,gcf,rect);

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n=1:nmax

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

ex(:,2:je)=caex(:,2:je).*ex(:,2:je)+...
           cbex(:,2:je).*(hz(:,2:je)-hz(:,1:je-1));
```

```
ey(2:ie,:)=caey(2:ie,:).*ey(2:ie,:)+...
          cbey(2:ie,:).*(hz(1:ie-1,:)-hz(2:ie,:));

ex(80,2:15)=0; %(added)
ex(80,35:je)=0; %(added)
ey(80,1:15)=0; %(added)
ey(80,35:je)=0; %(added)
%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

exbcf(:,2:jebc)=caexbcf(:,2:jebc).*exbcf(:,2:jebc)-...
  cbexbcf(:,2:jebc).*(hzxbcf(:,1:jebc-1)+hzybcf(:,1:jebc-1)-...
                    hzxbcf(:,2:jebc)-hzybcf(:,2:jebc));
ex(1:ie,1)=caex(1:ie,1).*ex(1:ie,1)-...
  cbex(1:ie,1).*(hzxbcf(ibbc:iebc+ie,jebc)+...
               hzybcf(ibbc:iebc+ie,jebc)-hz(1:ie,1));

%     BACK

exbcb(:,2:jebc-1)=caexbcb(:,2:jebc-1).*exbcb(:,2:jebc-1)-...
  cbexbcb(:,2:jebc-1).*(hzxbcb(:,1:jebc-2)+hzybcb(:,1:jebc-2)-...
                      hzxbcb(:,2:jebc-1)-hzybcb(:,2:jebc-1));
ex(1:ie,jb)=caex(1:ie,jb).*ex(1:ie,jb)-...
  cbex(1:ie,jb).*(hz(1:ie,jb-1)-hzxbcb(ibbc:iebc+ie,1)-...
               hzybcb(ibbc:iebc+ie,1));

%     LEFT

exbcl(:,2:je)=caexbcl(:,2:je).*exbcl(:,2:je)-...
  cbexbcl(:,2:je).*(hzxbcl(:,1:je-1)+hzybcl(:,1:je-1)-...
                  hzxbcl(:,2:je)-hzybcl(:,2:je));
exbcl(:,1)=caexbcl(:,1).*exbcl(:,1)-...
  cbexbcl(:,1).*(hzxbcf(1:iebc,jebc)+hzybcf(1:iebc,jebc)-...
               hzxbcl(:,1)-hzybcl(:,1));
exbcl(:,jb)=caexbcl(:,jb).*exbcl(:,jb)-...
  cbexbcl(:,jb).*(hzxbcl(:,je)+hzybcl(:,je)-...
                hzxbcb(1:iebc,1)-hzybcb(1:iebc,1));

%     RIGHT

exbcr(:,2:je)=caexbcr(:,2:je).*exbcr(:,2:je)-...
  cbexbcr(:,2:je).*(hzxbcr(:,1:je-1)+hzybcr(:,1:je-1)-...
                  hzxbcr(:,2:je)-hzybcr(:,2:je));
exbcr(:,1)=caexbcr(:,1).*exbcr(:,1)-...
  cbexbcr(:,1).*(hzxbcf(1+iebc+ie:iefbc,jebc)+...
               hzybcf(1+iebc+ie:iefbc,jebc)-...
               hzxbcr(:,1)-hzybcr(:,1));
exbcr(:,jb)=caexbcr(:,jb).*exbcr(:,jb)-...
  cbexbcr(:,jb).*(hzxbcr(:,je)+hzybcr(:,je)-...
                hzxbcb(1+iebc+ie:iefbc,1)-...
                hzybcb(1+iebc+ie:iefbc,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

eybcf(2:iefbc,:)=caeybcf(2:iefbc,:).*eybcf(2:iefbc,:)-...
  cbeybcf(2:iefbc,:).*(hzxbcf(2:iefbc,:)+hzybcf(2:iefbc,:)-...
                     hzxbcf(1:iefbc-1,:)-hzybcf(1:iefbc-1,:));

%     BACK

eybcb(2:iefbc,:)=caeybcb(2:iefbc,:).*eybcb(2:iefbc,:)-...
  cbeybcb(2:iefbc,:).*(hzxbcb(2:iefbc,:)+hzybcb(2:iefbc,:)-...
                     hzxbcb(1:iefbc-1,:)-hzybcb(1:iefbc-1,:));

%     LEFT

eybcl(2:iebc,:)=caeybcl(2:iebc,:).*eybcl(2:iebc,:)-...
  cbeybcl(2:iebc,:).*(hzxbcl(2:iebc,:)+hzybcl(2:iebc,:)-...
                hzxbcl(1:iebc-1,:)-hzybcl(1:iebc-1,:));
ey(1,:)=caey(1,:).*ey(1,:)-...
  cbey(1,:).*(hz(1,:)-hzxbcl(iebc,:)-hzybcl(iebc,:));

%   RIGHT

eybcr(2:iebc,:)=caeybcr(2:iebc,:).*eybcr(2:iebc,:)-...
  cbeybcr(2:iebc,:).*(hzxbcr(2:iebc,:)+hzybcr(2:iebc,:)-...
                hzxbcr(1:iebc-1,:)-hzybcr(1:iebc-1,:));
ey(ib,:)=caey(ib,:).*ey(ib,:)-...
  cbey(ib,:).*(hzxbcr(1,:)+hzybcr(1,:)- hz(ie,:));


%***********************************************************************
%   Update magnetic fields (HZ) in main grid
%***********************************************************************

hz(1:ie,1:je)=dahz(1:ie,1:je).*hz(1:ie,1:je)+...
          dbhz(1:ie,1:je).*(ex(1:ie,2:jb)-ex(1:ie,1:je)+...
                        ey(1:ie,1:je)-ey(2:ib,1:je));

hz(is,js)=source(n);

hz(80,1:15)=0; %(added)
hz(80,35:je)=0; %(added)

%***********************************************************************
%   Update HZX in PML regions
%***********************************************************************

%   FRONT

hzxbcf(1:iefbc,:)=dahzxbcf(1:iefbc,:).*hzxbcf(1:iefbc,:)-...
  dbhzxbcf(1:iefbc,:).*(eybcf(2:ibfbc,:)-eybcf(1:iefbc,:));

%   BACK

hzxbcb(1:iefbc,:)=dahzxbcb(1:iefbc,:).*hzxbcb(1:iefbc,:)-...
  dbhzxbcb(1:iefbc,:).*(eybcb(2:ibfbc,:)-eybcb(1:iefbc,:));

%   LEFT

hzxbcl(1:iebc-1,:)=dahzxbcl(1:iebc-1,:).*hzxbcl(1:iebc-1,:)-...
  dbhzxbcl(1:iebc-1,:).*(eybcl(2:iebc,:)-eybcl(1:iebc-1,:));
hzxbcl(iebc,:)=dahzxbcl(iebc,:).*hzxbcl(iebc,:)-...
  dbhzxbcl(iebc,:).*(ey(1,:)-eybcl(iebc,:));

%   RIGHT

hzxbcr(2:iebc,:)=dahzxbcr(2:iebc,:).*hzxbcr(2:iebc,:)-...
  dbhzxbcr(2:iebc,:).*(eybcr(3:ibbc,:)-eybcr(2:iebc,:));
hzxbcr(1,:)=dahzxbcr(1,:).*hzxbcr(1,:)-...
  dbhzxbcr(1,:).*(eybcr(2,:)-ey(ib,:));

%***********************************************************************
%   Update HZY in PML regions
%***********************************************************************

%   FRONT

hzybcf(:,1:jebc-1)=dahzybcf(:,1:jebc-1).*hzybcf(:,1:jebc-1)-...
  dbhzybcf(:,1:jebc-1).*(exbcf(:,1:jebc-1)-exbcf(:,2:jebc));
hzybcf(1:iebc,jebc)=dahzybcf(1:iebc,jebc).*hzybcf(1:iebc,jebc)-...
  dbhzybcf(1:iebc,jebc).*(exbcf(1:iebc,jebc)-exbcl(1:iebc,1));
hzybcf(iebc+1:iebc+ie,jebc)=...
  dahzybcf(iebc+1:iebc+ie,jebc).*hzybcf(iebc+1:iebc+ie,jebc)-...
  dbhzybcf(iebc+1:iebc+ie,jebc).*(exbcf(iebc+1:iebc+ie,jebc)-...
                          ex(1:ie,1));
hzybcf(iebc+ie+1:iefbc,jebc)=...
  dahzybcf(iebc+ie+1:iefbc,jebc).*hzybcf(iebc+ie+1:iefbc,jebc)-...
  dbhzybcf(iebc+ie+1:iefbc,jebc).*(exbcf(iebc+ie+1:iefbc,jebc)-...
                          exbcr(1:iebc,1));

%   BACK
```

```
hzybcb(1:iefbc,2:jebc)=dahzybcb(1:iefbc,2:jebc).*hzybcb(1:iefbc,2:jebc)-...
  dbhzybcb(1:iefbc,2:jebc).*(exbcb(1:iefbc,2:jebc)-exbcb(1:iefbc,3:jbbc));
hzybcb(1:iebc,1)=dahzybcb(1:iebc,1).*hzybcb(1:iebc,1)-...
  dbhzybcb(1:iebc,1).*(exbcl(1:iebc,jb)-exbcb(1:iebc,2));
hzybcb(iebc+1:iebc+ie,1)=...
  dahzybcb(iebc+1:iebc+ie,1).*hzybcb(iebc+1:iebc+ie,1)-...
  dbhzybcb(iebc+1:iebc+ie,1).*(ex(1:ie,jb)-exbcb(iebc+1:iebc+ie,2));
hzybcb(iebc+ie+1:iefbc,1)=...
  dahzybcb(iebc+ie+1:iefbc,1).*hzybcb(iebc+ie+1:iefbc,1)-...
  dbhzybcb(iebc+ie+1:iefbc,1).*(exbcr(1:iebc,jb)-...
                              exbcb(iebc+ie+1:iefbc,2));

%    LEFT

hzybcl(:,1:je)=dahzybcl(:,1:je).*hzybcl(:,1:je)-...
  dbhzybcl(:,1:je).*(exbcl(:,1:je)-exbcl(:,2:jb));

%    RIGHT

hzybcr(:,1:je)=dahzybcr(:,1:je).*hzybcr(:,1:je)-...
  dbhzybcr(:,1:je).*(exbcr(:,1:je)-exbcr(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************

if mod(n,8)==0;

timestep=int2str(n);

subplot(3,1,1),pcolor(ex');
colormap hot %(added)
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = ',timestep]);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = ',timestep]);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = ',timestep]);

nn=n/8;
M(:,nn)=getframe(gcf,rect);

end;

%***********************************************************************
%    END TIME-STEPPING LOOP
%***********************************************************************

end

movie(gcf,M,0,10,rect);
```

```
%Filename: threewl2.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all %(added)
close all %(added)
pack %(added)
clc %(added)

%***********************************************************************
%   Fundamental constants
%***********************************************************************

cc=2.99792458e8;          %speed of light in free space
muz=4.0*pi*1.0e-7;        %permeability of free space
epsz=1.0/(cc*cc*muz);     %permittivity of free space

freq=5.0e+9;              %center frequency of source excitation
lambda=cc/freq;           %center wavelength of source excitation
omega=2.0*pi*freq;

%***********************************************************************
%   Grid parameters
%***********************************************************************

ie=100;         %number of grid cells in x-direction
je=50;          %number of grid cells in y-direction

ib=ie+1;
jb=je+1;

is=15;          %location of z-directed hard source
js=je/2;        %location of z-directed hard source

dx=3.0e-3;      %space increment of square lattice
dt=dx/(2.0*cc);  %time step

nmax=240; %(changed)       %total number of time steps

iebc=8;         %thickness of left and right PML region
jebc=8;         %thickness of front and back PML region
rmax=0.00001;
orderbc=2;
ibbc=iebc+1;
jbbc=jebc+1;
iefbc=ie+2*iebc;
jefbc=je+2*jebc;
ibfbc=iefbc+1;
jbfbc=jefbc+1;

%***********************************************************************
%   Material parameters
%***********************************************************************

media=2;

eps=[1.0 1.0];
sig=[0.0 1.0e+7];
mur=[1.0 1.0];
sim=[0.0 0.0];

%***********************************************************************
%   Wave excitation
%***********************************************************************
```

```
rtau=160.0e-12;
tau=rtau/dt;
delay=3*tau;

source=zeros(1,nmax);
for n=1:4*tau
  source(n)=sin(omega*(n-delay)*dt)*exp(-((n-delay)^2/tau^2));
end

%***********************************************************************
%     Field arrays
%***********************************************************************

ex=zeros(ie,jb);          %fields in main grid
ey=zeros(ib,je);
hz=zeros(ie,je);

exbcf=zeros(iefbc,jebc);   %fields in front PML region
eybcf=zeros(ibfbc,jebc);
hzxbcf=zeros(iefbc,jebc);
hzybcf=zeros(iefbc,jebc);

exbcb=zeros(iefbc,jbbc);   %fields in back PML region
eybcb=zeros(ibfbc,jebc);
hzxbcb=zeros(iefbc,jebc);
hzybcb=zeros(iefbc,jebc);

exbcl=zeros(iebc,jb);      %fields in left PML region
eybcl=zeros(iebc,je);
hzxbcl=zeros(iebc,je);
hzybcl=zeros(iebc,je);

exbcr=zeros(iebc,jb);      %fields in right PML region
eybcr=zeros(ibbc,je);
hzxbcr=zeros(iebc,je);
hzybcr=zeros(iebc,je);

%***********************************************************************
%     Updating coefficients
%***********************************************************************

for i=1:media
  eaf  =dt*sig(i)/(2.0*epsz*eps(i));
  ca(i)=(1.0-eaf)/(1.0+eaf);
  cb(i)=dt/epsz/eps(i)/dx/(1.0+eaf);
  haf  =dt*sim(i)/(2.0*muz*mur(i));
  da(i)=(1.0-haf)/(1.0+haf);
  db(i)=dt/muz/mur(i)/dx/(1.0+haf);
end

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

caex(1:ie,1:jb)=ca(1);
cbex(1:ie,1:jb)=cb(1);

caey(1:ib,1:je)=ca(1);
cbey(1:ib,1:je)=cb(1);

dahz(1:ie,1:je)=da(1);
dbhz(1:ie,1:je)=db(1);

%     Add PEC with slit of width equals three lambda
caex(80,[1:10])=ca(2); %(added)
caex(80,[40:je])=ca(2); %(added)
cbex(80,[1:10])=cb(2); %(added)
cbex(80,[40:je])=cb(2); %(added)
caey(80,[1:10])=ca(2); %(added)
caey(80,[40:je])=ca(2); %(added)
cbey(80,[1:10])=cb(2); %(added)
cbey(80,[40:je])=cb(2); %(added)
```

```
%***********************************************************************
%     Fill the PML regions
%***********************************************************************

delbc=iebc*dx;
sigmam=-log(rmax/100.0)*epsz*cc*(orderbc+1)/(2*delbc);
bcfactor=eps(1)*sigmam/(dx*(delbc^orderbc)*(orderbc+1));

%     FRONT region

caexbcf(1:iefbc,1)=1.0;
cbexbcf(1:iefbc,1)=0.0;
for j=2:jebc
  y1=(jebc-j+1.5)*dx;
  y2=(jebc-j+0.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1.0-ca1)/(sigmay*dx);
  caexbcf(1:iefbc,j)=ca1;
  cbexbcf(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,1)=ca1;
cbex(1:ie,1)=cb1;
caexbcl(1:iebc,1)=ca1;
cbexbcl(1:iebc,1)=cb1;
caexbcr(1:iebc,1)=ca1;
cbexbcr(1:iebc,1)=cb1;

for j=1:jebc
  y1=(jebc-j+1)*dx;
  y2=(jebc-j)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcf(1:iefbc,j)=da1;
  dbhzybcf(1:iefbc,j)=db1;
  caeybcf(1:ibfbc,j)=ca(1);
  cbeybcf(1:ibfbc,j)=cb(1);
  dahzxbcf(1:iefbc,j)=da(1);
  dbhzxbcf(1:iefbc,j)=db(1);
end

%     BACK region

caexbcb(1:iefbc,jbbc)=1.0;
cbexbcb(1:iefbc,jbbc)=0.0;
for j=2:jebc
  y1=(j-0.5)*dx;
  y2=(j-1.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmay*dx);
  caexbcb(1:iefbc,j)=ca1;
  cbexbcb(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,jb)=ca1;
cbex(1:ie,jb)=cb1;
caexbcl(1:iebc,jb)=ca1;
cbexbcl(1:iebc,jb)=cb1;
caexbcr(1:iebc,jb)=ca1;
cbexbcr(1:iebc,jb)=cb1;

for j=1:jebc
  y1=j*dx;
  y2=(j-1)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcb(1:iefbc,j)=da1;
  dbhzybcb(1:iefbc,j)=db1;
  caeybcb(1:ibfbc,j)=ca(1);
  cbeybcb(1:ibfbc,j)=cb(1);
  dahzxbcb(1:iefbc,j)=da(1);
  dbhzxbcb(1:iefbc,j)=db(1);
end

%    LEFT region

caeybcl(1,1:je)=1.0;
cbeybcl(1,1:je)=0.0;
for i=2:iebc
  x1=(iebc-i+1.5)*dx;
  x2=(iebc-i+0.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcl(i,1:je)=ca1;
  cbeybcl(i,1:je)=cb1;
  caeybcf(i,1:jebc)=ca1;
  cbeybcf(i,1:jebc)=cb1;
  caeybcb(i,1:jebc)=ca1;
  cbeybcb(i,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(1,1:je)=ca1;
cbey(1,1:je)=cb1;
caeybcf(iebc+1,1:jebc)=ca1;
cbeybcf(iebc+1,1:jebc)=cb1;
caeybcb(iebc+1,1:jebc)=ca1;
cbeybcb(iebc+1,1:jebc)=cb1;

for i=1:iebc
  x1=(iebc-i+1)*dx;
  x2=(iebc-i)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcl(i,1:je)=da1;
  dbhzxbcl(i,1:je)=db1;
  dahzxbcf(i,1:jebc)=da1;
  dbhzxbcf(i,1:jebc)=db1;
  dahzxbcb(i,1:jebc)=da1;
  dbhzxbcb(i,1:jebc)=db1;
  caexbcl(i,2:je)=ca(1);
  cbexbcl(i,2:je)=cb(1);
  dahzybcl(i,1:je)=da(1);
  dbhzybcl(i,1:je)=db(1);
end

%    RIGHT region

caeybcr(ibbc,1:je)=1.0;
cbeybcr(ibbc,1:je)=0.0;
for i=2:iebc
  x1=(i-0.5)*dx;
  x2=(i-1.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcr(i,1:je)=ca1;
  cbeybcr(i,1:je)=cb1;
  caeybcf(i+iebc+ie,1:jebc)=ca1;
  cbeybcf(i+iebc+ie,1:jebc)=cb1;
  caeybcb(i+iebc+ie,1:jebc)=ca1;
  cbeybcb(i+iebc+ie,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
```

```
caey(ib,1:je)=ca1;
cbey(ib,1:je)=cb1;
caeybcf(iebc+ib,1:jebc)=ca1;
cbeybcf(iebc+ib,1:jebc)=cb1;
caeybcb(iebc+ib,1:jebc)=ca1;
cbeybcb(iebc+ib,1:jebc)=cb1;

for i=1:iebc
  x1=i*dx;
  x2=(i-1)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcr(i,1:je) = da1;
  dbhzxbcr(i,1:je) = db1;
  dahzxbcf(i+ie+iebc,1:jebc)=da1;
  dbhzxbcf(i+ie+iebc,1:jebc)=db1;
  dahzxbcb(i+ie+iebc,1:jebc)=da1;
  dbhzxbcb(i+ie+iebc,1:jebc)=db1;
  caexbcr(i,2:je)=ca(1);
  cbexbcr(i,2:je)=cb(1);
  dahzybcr(i,1:je)=da(1);
  dbhzybcr(i,1:je)=db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

subplot(3,1,1),pcolor(ex');
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = 0']);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = 0']);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = 0']);

rect=get(gcf,'Position');
rect(1:2)=[0 0];

M=moviein(nmax/8,gcf,rect);

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n=1:nmax

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

ex(:,2:je)=caex(:,2:je).*ex(:,2:je)+...
           cbex(:,2:je).*(hz(:,2:je)-hz(:,1:je-1));
```

```
ey(2:ie,:)=caey(2:ie,:).*ey(2:ie,:)+...
           cbey(2:ie,:).*(hz(1:ie-1,:)-hz(2:ie,:));

ex(80,2:10)=0; %(added)
ex(80,40:je)=0; %(added)
ey(80,1:10)=0; %(added)
ey(80,40:je)=0; %(added)
%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

exbcf(:,2:jebc)=caexbcf(:,2:jebc).*exbcf(:,2:jebc)-...
  cbexbcf(:,2:jebc).*(hzxbcf(:,1:jebc-1)+hzybcf(:,1:jebc-1)-...
                    hzxbcf(:,2:jebc)-hzybcf(:,2:jebc));
ex(1:ie,1)=caex(1:ie,1).*ex(1:ie,1)-...
  cbex(1:ie,1).*(hzxbcf(ibbc:iebc+ie,jebc)+...
              hzybcf(ibbc:iebc+ie,jebc)-hz(1:ie,1));

%     BACK

exbcb(:,2:jebc-1)=caexbcb(:,2:jebc-1).*exbcb(:,2:jebc-1)-...
  cbexbcb(:,2:jebc-1).*(hzxbcb(:,1:jebc-2)+hzybcb(:,1:jebc-2)-...
                      hzxbcb(:,2:jebc-1)-hzybcb(:,2:jebc-1));
ex(1:ie,jb)=caex(1:ie,jb).*ex(1:ie,jb)-...
  cbex(1:ie,jb).*(hz(1:ie,jb-1)-hzxbcb(ibbc:iebc+ie,1)-...
               hzybcb(ibbc:iebc+ie,1));

%     LEFT

exbcl(:,2:je)=caexbcl(:,2:je).*exbcl(:,2:je)-...
  cbexbcl(:,2:je).*(hzxbcl(:,1:je-1)+hzybcl(:,1:je-1)-...
                  hzxbcl(:,2:je)-hzybcl(:,2:je));
exbcl(:,1)=caexbcl(:,1).*exbcl(:,1)-...
  cbexbcl(:,1).*(hzxbcf(1:iebc,jebc)+hzybcf(1:iebc,jebc)-...
               hzxbcl(:,1)-hzybcl(:,1));
exbcl(:,jb)=caexbcl(:,jb).*exbcl(:,jb)-...
  cbexbcl(:,jb).*(hzxbcl(:,je)+hzybcl(:,je)-...
                hzxbcb(1:iebc,1)-hzybcb(1:iebc,1));

%     RIGHT

exbcr(:,2:je)=caexbcr(:,2:je).*exbcr(:,2:je)-...
  cbexbcr(:,2:je).*(hzxbcr(:,1:je-1)+hzybcr(:,1:je-1)-...
                  hzxbcr(:,2:je)-hzybcr(:,2:je));
exbcr(:,1)=caexbcr(:,1).*exbcr(:,1)-...
  cbexbcr(:,1).*(hzxbcf(1+iebc+ie:iefbc,jebc)+...
               hzybcf(1+iebc+ie:iefbc,jebc)-...
               hzxbcr(:,1)-hzybcr(:,1));
exbcr(:,jb)=caexbcr(:,jb).*exbcr(:,jb)-...
  cbexbcr(:,jb).*(hzxbcr(:,je)+hzybcr(:,je)-...
                hzxbcb(1+iebc+ie:iefbc,1)-...
                hzybcb(1+iebc+ie:iefbc,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

eybcf(2:iefbc,:)=caeybcf(2:iefbc,:).*eybcf(2:iefbc,:)-...
  cbeybcf(2:iefbc,:).*(hzxbcf(2:iefbc,:)+hzybcf(2:iefbc,:)-...
                     hzxbcf(1:iefbc-1,:)-hzybcf(1:iefbc-1,:));

%     BACK

eybcb(2:iefbc,:)=caeybcb(2:iefbc,:).*eybcb(2:iefbc,:)-...
  cbeybcb(2:iefbc,:).*(hzxbcb(2:iefbc,:)+hzybcb(2:iefbc,:)-...
                     hzxbcb(1:iefbc-1,:)-hzybcb(1:iefbc-1,:));

%     LEFT

eybcl(2:iebc,:)=caeybcl(2:iebc,:).*eybcl(2:iebc,:)-...
  cbeybcl(2:iebc,:).*(hzxbcl(2:iebc,:)+hzybcl(2:iebc,:)-...
                hzxbcl(1:iebc-1,:)-hzybcl(1:iebc-1,:));
ey(1,:)=caey(1,:).*ey(1,:)-...
  cbey(1,:).*(hz(1,:)-hzxbcl(iebc,:)-hzybcl(iebc,:));

%    RIGHT

eybcr(2:iebc,:)=caeybcr(2:iebc,:).*eybcr(2:iebc,:)-...
  cbeybcr(2:iebc,:).*(hzxbcr(2:iebc,:)+hzybcr(2:iebc,:)-...
                hzxbcr(1:iebc-1,:)-hzybcr(1:iebc-1,:));
ey(ib,:)=caey(ib,:).*ey(ib,:)-...
  cbey(ib,:).*(hzxbcr(1,:)+hzybcr(1,:)- hz(ie,:));


%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

hz(1:ie,1:je)=dahz(1:ie,1:je).*hz(1:ie,1:je)+...
          dbhz(1:ie,1:je).*(ex(1:ie,2:jb)-ex(1:ie,1:je)+...
                          ey(1:ie,1:je)-ey(2:ib,1:je));

hz(is,js)=source(n);

hz(80,1:10)=0; %(added)
hz(80,40:je)=0; %(added)

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

hzxbcf(1:iefbc,:)=dahzxbcf(1:iefbc,:).*hzxbcf(1:iefbc,:)-...
  dbhzxbcf(1:iefbc,:).*(eybcf(2:ibfbc,:)-eybcf(1:iefbc,:));

%    BACK

hzxbcb(1:iefbc,:)=dahzxbcb(1:iefbc,:).*hzxbcb(1:iefbc,:)-...
  dbhzxbcb(1:iefbc,:).*(eybcb(2:ibfbc,:)-eybcb(1:iefbc,:));

%    LEFT

hzxbcl(1:iebc-1,:)=dahzxbcl(1:iebc-1,:).*hzxbcl(1:iebc-1,:)-...
  dbhzxbcl(1:iebc-1,:).*(eybcl(2:iebc,:)-eybcl(1:iebc-1,:));
hzxbcl(iebc,:)=dahzxbcl(iebc,:).*hzxbcl(iebc,:)-...
  dbhzxbcl(iebc,:).*(ey(1,:)-eybcl(iebc,:));

%    RIGHT

hzxbcr(2:iebc,:)=dahzxbcr(2:iebc,:).*hzxbcr(2:iebc,:)-...
  dbhzxbcr(2:iebc,:).*(eybcr(3:ibbc,:)-eybcr(2:iebc,:));
hzxbcr(1,:)=dahzxbcr(1,:).*hzxbcr(1,:)-...
  dbhzxbcr(1,:).*(eybcr(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

hzybcf(:,1:jebc-1)=dahzybcf(:,1:jebc-1).*hzybcf(:,1:jebc-1)-...
  dbhzybcf(:,1:jebc-1).*(exbcf(:,1:jebc-1)-exbcf(:,2:jebc));
hzybcf(1:iebc,jebc)=dahzybcf(1:iebc,jebc).*hzybcf(1:iebc,jebc)-...
  dbhzybcf(1:iebc,jebc).*(exbcf(1:iebc,jebc)-exbcl(1:iebc,1));
hzybcf(iebc+1:iebc+ie,jebc)=...
  dahzybcf(iebc+1:iebc+ie,jebc).*hzybcf(iebc+1:iebc+ie,jebc)-...
  dbhzybcf(iebc+1:iebc+ie,jebc).*(exbcf(iebc+1:iebc+ie,jebc)-...
                           ex(1:ie,1));
hzybcf(iebc+ie+1:iefbc,jebc)=...
  dahzybcf(iebc+ie+1:iefbc,jebc).*hzybcf(iebc+ie+1:iefbc,jebc)-...
  dbhzybcf(iebc+ie+1:iefbc,jebc).*(exbcf(iebc+ie+1:iefbc,jebc)-...
                            exbcr(1:iebc,1));

%    BACK
```

```
hzybcb(1:iefbc,2:jebc)=dahzybcb(1:iefbc,2:jebc).*hzybcb(1:iefbc,2:jebc)-...
  dbhzybcb(1:iefbc,2:jebc).*(exbcb(1:iefbc,2:jebc)-exbcb(1:iefbc,3:jbbc));
hzybcb(1:iebc,1)=dahzybcb(1:iebc,1).*hzybcb(1:iebc,1)-...
  dbhzybcb(1:iebc,1).*(exbcl(1:iebc,jb)-exbcb(1:iebc,2));
hzybcb(iebc+1:iebc+ie,1)=...
  dahzybcb(iebc+1:iebc+ie,1).*hzybcb(iebc+1:iebc+ie,1)-...
  dbhzybcb(iebc+1:iebc+ie,1).*(ex(1:ie,jb)-exbcb(iebc+1:iebc+ie,2));
hzybcb(iebc+ie+1:iefbc,1)=...
  dahzybcb(iebc+ie+1:iefbc,1).*hzybcb(iebc+ie+1:iefbc,1)-...
  dbhzybcb(iebc+ie+1:iefbc,1).*(exbcr(1:iebc,jb)-...
                              exbcb(iebc+ie+1:iefbc,2));

%     LEFT

hzybcl(:,1:je)=dahzybcl(:,1:je).*hzybcl(:,1:je)-...
  dbhzybcl(:,1:je).*(exbcl(:,1:je)-exbcl(:,2:jb));

%     RIGHT

hzybcr(:,1:je)=dahzybcr(:,1:je).*hzybcr(:,1:je)-...
  dbhzybcr(:,1:je).*(exbcr(:,1:je)-exbcr(:,2:jb));

%***********************************************************************
%     Visualize fields
%***********************************************************************

if mod(n,8)==0;

timestep=int2str(n);

subplot(3,1,1),pcolor(ex');
colormap hot %(added)
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = ',timestep]);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = ',timestep]);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = ',timestep]);

nn=n/8;
M(:,nn)=getframe(gcf,rect);

end;

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(gcf,M,0,10,rect);
```

```
%Filename: PEC2.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%             Department of Electrical and Computer Engineering
%             University of Wisconsin-Madison
%             1415 Engineering Drive
%             Madison, WI 53706-1691
%             608-265-5739
%             hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all %(added)
close all %(added)
pack %(added)
clc %(added)

%***********************************************************************
%     Fundamental constants
%***********************************************************************

cc=2.99792458e8;            %speed of light in free space
muz=4.0*pi*1.0e-7;          %permeability of free space
epsz=1.0/(cc*cc*muz);       %permittivity of free space

freq=5.0e+9;                %center frequency of source excitation
lambda=cc/freq;             %center wavelength of source excitation
omega=2.0*pi*freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

ie=100;          %number of grid cells in x-direction
je=50;           %number of grid cells in y-direction

ib=ie+1;
jb=je+1;

is=15;           %location of z-directed hard source
js=je/2;         %location of z-directed hard source

dx=3.0e-3;       %space increment of square lattice
dt=dx/(2.0*cc);   %time step

nmax=256; %(changed)        %total number of time steps

iebc=8;          %thickness of left and right PML region
jebc=8;          %thickness of front and back PML region
rmax=0.00001;
orderbc=2;
ibbc=iebc+1;
jbbc=jebc+1;
iefbc=ie+2*iebc;
jefbc=je+2*jebc;
ibfbc=iefbc+1;
jbfbc=jefbc+1;

%***********************************************************************
%     Material parameters
%***********************************************************************

media=2;

eps=[1.0 1.0];
sig=[0.0 1.0e+7];
mur=[1.0 1.0];
sim=[0.0 0.0];

%***********************************************************************
%     Wave excitation
%***********************************************************************
```

```
rtau=160.0e-12;
tau=rtau/dt;
delay=3*tau;

source=zeros(1,nmax);
for n=1:4*tau
  source(n)=sin(omega*(n-delay)*dt)*exp(-((n-delay)^2/tau^2));
end

%***********************************************************************
%     Field arrays
%***********************************************************************

ex=zeros(ie,jb);           %fields in main grid
ey=zeros(ib,je);
hz=zeros(ie,je);

exbcf=zeros(iefbc,jebc);   %fields in front PML region
eybcf=zeros(ibfbc,jebc);
hzxbcf=zeros(iefbc,jebc);
hzybcf=zeros(iefbc,jebc);

exbcb=zeros(iefbc,jbbc);   %fields in back PML region
eybcb=zeros(ibfbc,jebc);
hzxbcb=zeros(iefbc,jebc);
hzybcb=zeros(iefbc,jebc);

exbcl=zeros(iebc,jb);      %fields in left PML region
eybcl=zeros(iebc,je);
hzxbcl=zeros(iebc,je);
hzybcl=zeros(iebc,je);

exbcr=zeros(iebc,jb);      %fields in right PML region
eybcr=zeros(ibbc,je);
hzxbcr=zeros(iebc,je);
hzybcr=zeros(iebc,je);

%***********************************************************************
%     Updating coefficients
%***********************************************************************

for i=1:media
  eaf  =dt*sig(i)/(2.0*epsz*eps(i));
  ca(i)=(1.0-eaf)/(1.0+eaf);
  cb(i)=dt/epsz/eps(i)/dx/(1.0+eaf);
  haf  =dt*sim(i)/(2.0*muz*mur(i));
  da(i)=(1.0-haf)/(1.0+haf);
  db(i)=dt/muz/mur(i)/dx/(1.0+haf);
end

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

caex(1:ie,1:jb)=ca(1);
cbex(1:ie,1:jb)=cb(1);

caey(1:ib,1:je)=ca(1);
cbey(1:ib,1:je)=cb(1);

dahz(1:ie,1:je)=da(1);
dbhz(1:ie,1:je)=db(1);

%     Add PEC without slit
caex(80,[1:je])=ca(2); %(added)
cbex(80,[1:je])=cb(2); %(added)
caey(80,[1:je])=ca(2); %(added)
cbey(80,[1:je])=cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
%***********************************************************************
```

```
delbc=iebc*dx;
sigmam=-log(rmax/100.0)*epsz*cc*(orderbc+1)/(2*delbc);
bcfactor=eps(1)*sigmam/(dx*(delbc^orderbc)*(orderbc+1));

%     FRONT region

caexbcf(1:iefbc,1)=1.0;
cbexbcf(1:iefbc,1)=0.0;
for j=2:jebc
  y1=(jebc-j+1.5)*dx;
  y2=(jebc-j+0.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1.0-ca1)/(sigmay*dx);
  caexbcf(1:iefbc,j)=ca1;
  cbexbcf(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,1)=ca1;
cbex(1:ie,1)=cb1;
caexbcl(1:iebc,1)=ca1;
cbexbcl(1:iebc,1)=cb1;
caexbcr(1:iebc,1)=ca1;
cbexbcr(1:iebc,1)=cb1;

for j=1:jebc
  y1=(jebc-j+1)*dx;
  y2=(jebc-j)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcf(1:iefbc,j)=da1;
  dbhzybcf(1:iefbc,j)=db1;
  caeybcf(1:ibfbc,j)=ca(1);
  cbeybcf(1:ibfbc,j)=cb(1);
  dahzxbcf(1:iefbc,j)=da(1);
  dbhzxbcf(1:iefbc,j)=db(1);
end

%     BACK region

caexbcb(1:iefbc,jbbc)=1.0;
cbexbcb(1:iefbc,jbbc)=0.0;
for j=2:jebc
  y1=(j-0.5)*dx;
  y2=(j-1.5)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  ca1=exp(-sigmay*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmay*dx);
  caexbcb(1:iefbc,j)=ca1;
  cbexbcb(1:iefbc,j)=cb1;
end
sigmay = bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmay*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmay*dx);
caex(1:ie,jb)=ca1;
cbex(1:ie,jb)=cb1;
caexbcl(1:iebc,jb)=ca1;
cbexbcl(1:iebc,jb)=cb1;
caexbcr(1:iebc,jb)=ca1;
cbexbcr(1:iebc,jb)=cb1;

for j=1:jebc
  y1=j*dx;
  y2=(j-1)*dx;
  sigmay=bcfactor*(y1^(orderbc+1)-y2^(orderbc+1));
  sigmays=sigmay*(muz/(epsz*eps(1)));
  da1=exp(-sigmays*dt/muz);
  db1=(1-da1)/(sigmays*dx);
  dahzybcb(1:iefbc,j)=da1;
  dbhzybcb(1:iefbc,j)=db1;
  caeybcb(1:ibfbc,j)=ca(1);
  cbeybcb(1:ibfbc,j)=cb(1);
  dahzxbcb(1:iefbc,j)=da(1);
  dbhzxbcb(1:iefbc,j)=db(1);
end

%     LEFT region

caeybcl(1,1:je)=1.0;
cbeybcl(1,1:je)=0.0;
for i=2:iebc
  x1=(iebc-i+1.5)*dx;
  x2=(iebc-i+0.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcl(i,1:je)=ca1;
  cbeybcl(i,1:je)=cb1;
  caeybcf(i,1:jebc)=ca1;
  cbeybcf(i,1:jebc)=cb1;
  caeybcb(i,1:jebc)=ca1;
  cbeybcb(i,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(1,1:je)=ca1;
cbey(1,1:je)=cb1;
caeybcf(iebc+1,1:jebc)=ca1;
cbeybcf(iebc+1,1:jebc)=cb1;
caeybcb(iebc+1,1:jebc)=ca1;
cbeybcb(iebc+1,1:jebc)=cb1;

for i=1:iebc
  x1=(iebc-i+1)*dx;
  x2=(iebc-i)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcl(i,1:je)=da1;
  dbhzxbcl(i,1:je)=db1;
  dahzxbcf(i,1:jebc)=da1;
  dbhzxbcf(i,1:jebc)=db1;
  dahzxbcb(i,1:jebc)=da1;
  dbhzxbcb(i,1:jebc)=db1;
  caexbcl(i,2:je)=ca(1);
  cbexbcl(i,2:je)=cb(1);
  dahzybcl(i,1:je)=da(1);
  dbhzybcl(i,1:je)=db(1);
end

%     RIGHT region

caeybcr(ibbc,1:je)=1.0;
cbeybcr(ibbc,1:je)=0.0;
for i=2:iebc
  x1=(i-0.5)*dx;
  x2=(i-1.5)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  ca1=exp(-sigmax*dt/(epsz*eps(1)));
  cb1=(1-ca1)/(sigmax*dx);
  caeybcr(i,1:je)=ca1;
  cbeybcr(i,1:je)=cb1;
  caeybcf(i+iebc+ie,1:jebc)=ca1;
  cbeybcf(i+iebc+ie,1:jebc)=cb1;
  caeybcb(i+iebc+ie,1:jebc)=ca1;
  cbeybcb(i+iebc+ie,1:jebc)=cb1;
end
sigmax=bcfactor*(0.5*dx)^(orderbc+1);
ca1=exp(-sigmax*dt/(epsz*eps(1)));
cb1=(1-ca1)/(sigmax*dx);
caey(ib,1:je)=ca1;
cbey(ib,1:je)=cb1;
caeybcf(iebc+ib,1:jebc)=ca1;
cbeybcf(iebc+ib,1:jebc)=cb1;
```

```
caeybcb(iebc+ib,1:jebc)=ca1;
cbeybcb(iebc+ib,1:jebc)=cb1;

for i=1:iebc
  x1=i*dx;
  x2=(i-1)*dx;
  sigmax=bcfactor*(x1^(orderbc+1)-x2^(orderbc+1));
  sigmaxs=sigmax*(muz/(epsz*eps(1)));
  da1=exp(-sigmaxs*dt/muz);
  db1=(1-da1)/(sigmaxs*dx);
  dahzxbcr(i,1:je) = da1;
  dbhzxbcr(i,1:je) = db1;
  dahzxbcf(i+ie+iebc,1:jebc)=da1;
  dbhzxbcf(i+ie+iebc,1:jebc)=db1;
  dahzxbcb(i+ie+iebc,1:jebc)=da1;
  dbhzxbcb(i+ie+iebc,1:jebc)=db1;
  caexbcr(i,2:je)=ca(1);
  cbexbcr(i,2:je)=cb(1);
  dahzybcr(i,1:je)=da(1);
  dbhzybcr(i,1:je)=db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

subplot(3,1,1),pcolor(ex');
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = 0']);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = 0']);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = 0']);

rect=get(gcf,'Position');
rect(1:2)=[0 0];

M=moviein(nmax/8,gcf,rect);

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n=1:nmax

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

ex(:,2:je)=caex(:,2:je).*ex(:,2:je)+...
           cbex(:,2:je).*(hz(:,2:je)-hz(:,1:je-1));

ey(2:ie,:)=caey(2:ie,:).*ey(2:ie,:)+...
           cbey(2:ie,:).*(hz(1:ie-1,:)-hz(2:ie,:));

ex(80,2:25)=0; %(added)
ex(80,26:je)=0; %(added)
ey(80,1:25)=0; %(added)
ey(80,26:je)=0; %(added)
%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

exbcf(:,2:jebc)=caexbcf(:,2:jebc).*exbcf(:,2:jebc)-...
  cbexbcf(:,2:jebc).*(hzxbcf(:,1:jebc-1)+hzybcf(:,1:jebc-1)-...
                  hzxbcf(:,2:jebc)-hzybcf(:,2:jebc));
ex(1:ie,1)=caex(1:ie,1).*ex(1:ie,1)-...
  cbex(1:ie,1).*(hzxbcf(ibbc:iebc+ie,jebc)+...
             hzybcf(ibbc:iebc+ie,jebc)-hz(1:ie,1));

%     BACK

exbcb(:,2:jebc-1)=caexbcb(:,2:jebc-1).*exbcb(:,2:jebc-1)-...
  cbexbcb(:,2:jebc-1).*(hzxbcb(:,1:jebc-2)+hzybcb(:,1:jebc-2)-...
                   hzxbcb(:,2:jebc-1)-hzybcb(:,2:jebc-1));
ex(1:ie,jb)=caex(1:ie,jb).*ex(1:ie,jb)-...
  cbex(1:ie,jb).*(hz(1:ie,jb-1)-hzxbcb(ibbc:iebc+ie,1)-...
             hzybcb(ibbc:iebc+ie,1));

%     LEFT

exbcl(:,2:je)=caexbcl(:,2:je).*exbcl(:,2:je)-...
  cbexbcl(:,2:je).*(hzxbcl(:,1:je-1)+hzybcl(:,1:je-1)-...
                hzxbcl(:,2:je)-hzybcl(:,2:je));
exbcl(:,1)=caexbcl(:,1).*exbcl(:,1)-...
  cbexbcl(:,1).*(hzxbcf(1:iebc,jebc)+hzybcf(1:iebc,jebc)-...
             hzxbcl(:,1)-hzybcl(:,1));
exbcl(:,jb)=caexbcl(:,jb).*exbcl(:,jb)-...
  cbexbcl(:,jb).*(hzxbcl(:,je)+hzybcl(:,je)-...
              hzxbcb(1:iebc,1)-hzybcb(1:iebc,1));

%     RIGHT

exbcr(:,2:je)=caexbcr(:,2:je).*exbcr(:,2:je)-...
  cbexbcr(:,2:je).*(hzxbcr(:,1:je-1)+hzybcr(:,1:je-1)-...
                hzxbcr(:,2:je)-hzybcr(:,2:je));
exbcr(:,1)=caexbcr(:,1).*exbcr(:,1)-...
  cbexbcr(:,1).*(hzxbcf(1+iebc+ie:iefbc,jebc)+...
             hzybcf(1+iebc+ie:iefbc,jebc)-...
             hzxbcr(:,1)-hzybcr(:,1));
exbcr(:,jb)=caexbcr(:,jb).*exbcr(:,jb)-...
  cbexbcr(:,jb).*(hzxbcr(:,je)+hzybcr(:,je)-...
              hzxbcb(1+iebc+ie:iefbc,1)-...
              hzybcb(1+iebc+ie:iefbc,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

eybcf(2:iefbc,:)=caeybcf(2:iefbc,:).*eybcf(2:iefbc,:)-...
  cbeybcf(2:iefbc,:).*(hzxbcf(2:iefbc,:)+hzybcf(2:iefbc,:)-...
                  hzxbcf(1:iefbc-1,:)-hzybcf(1:iefbc-1,:));

%     BACK

eybcb(2:iefbc,:)=caeybcb(2:iefbc,:).*eybcb(2:iefbc,:)-...
  cbeybcb(2:iefbc,:).*(hzxbcb(2:iefbc,:)+hzybcb(2:iefbc,:)-...
                  hzxbcb(1:iefbc-1,:)-hzybcb(1:iefbc-1,:));

%     LEFT

eybcl(2:iebc,:)=caeybcl(2:iebc,:).*eybcl(2:iebc,:)-...
  cbeybcl(2:iebc,:).*(hzxbcl(2:iebc,:)+hzybcl(2:iebc,:)-...
                  hzxbcl(1:iebc-1,:)-hzybcl(1:iebc-1,:));
ey(1,:)=caey(1,:).*ey(1,:)-...
  cbey(1,:).*(hz(1,:)-hzxbcl(iebc,:)-hzybcl(iebc,:));
```

```
%    RIGHT

eybcr(2:iebc,:)=caeybcr(2:iebc,:).*eybcr(2:iebc,:)-...
  cbeybcr(2:iebc,:).*(hzxbcr(2:iebc,:)+hzybcr(2:iebc,:)-...
                    hzxbcr(1:iebc-1,:)-hzybcr(1:iebc-1,:));
ey(ib,:)=caey(ib,:).*ey(ib,:)-...
  cbey(ib,:).*(hzxbcr(1,:)+hzybcr(1,:)- hz(ie,:));


%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

hz(1:ie,1:je)=dahz(1:ie,1:je).*hz(1:ie,1:je)+...
            dbhz(1:ie,1:je).*(ex(1:ie,2:jb)-ex(1:ie,1:je)+...
                            ey(1:ie,1:je)-ey(2:ib,1:je));

hz(is,js)=source(n);

hz(80,1:25)=0; %(added)
hz(80,26:je)=0; %(added)

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

hzxbcf(1:iefbc,:)=dahzxbcf(1:iefbc,:).*hzxbcf(1:iefbc,:)-...
  dbhzxbcf(1:iefbc,:).*(eybcf(2:ibfbc,:)-eybcf(1:iefbc,:));

%    BACK

hzxbcb(1:iefbc,:)=dahzxbcb(1:iefbc,:).*hzxbcb(1:iefbc,:)-...
  dbhzxbcb(1:iefbc,:).*(eybcb(2:ibfbc,:)-eybcb(1:iefbc,:));

%    LEFT

hzxbcl(1:iebc-1,:)=dahzxbcl(1:iebc-1,:).*hzxbcl(1:iebc-1,:)-...
  dbhzxbcl(1:iebc-1,:).*(eybcl(2:iebc,:)-eybcl(1:iebc-1,:));
hzxbcl(iebc,:)=dahzxbcl(iebc,:).*hzxbcl(iebc,:)-...
  dbhzxbcl(iebc,:).*(ey(1,:)-eybcl(iebc,:));

%    RIGHT

hzxbcr(2:iebc,:)=dahzxbcr(2:iebc,:).*hzxbcr(2:iebc,:)-...
  dbhzxbcr(2:iebc,:).*(eybcr(3:ibbc,:)-eybcr(2:iebc,:));
hzxbcr(1,:)=dahzxbcr(1,:).*hzxbcr(1,:)-...
  dbhzxbcr(1,:).*(eybcr(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

hzybcf(:,1:jebc-1)=dahzybcf(:,1:jebc-1).*hzybcf(:,1:jebc-1)-...
  dbhzybcf(:,1:jebc-1).*(exbcf(:,1:jebc-1)-exbcf(:,2:jebc));
hzybcf(1:iebc,jebc)=dahzybcf(1:iebc,jebc).*hzybcf(1:iebc,jebc)-...
  dbhzybcf(1:iebc,jebc).*(exbcf(1:iebc,jebc)-exbcl(1:iebc,1));
hzybcf(iebc+1:iebc+ie,jebc)=...
  dahzybcf(iebc+1:iebc+ie,jebc).*hzybcf(iebc+1:iebc+ie,jebc)-...
  dbhzybcf(iebc+1:iebc+ie,jebc).*(exbcf(iebc+1:iebc+ie,jebc)-...
                              ex(1:ie,1));
hzybcf(iebc+ie+1:iefbc,jebc)=...
  dahzybcf(iebc+ie+1:iefbc,jebc).*hzybcf(iebc+ie+1:iefbc,jebc)-...
  dbhzybcf(iebc+ie+1:iefbc,jebc).*(exbcf(iebc+ie+1:iefbc,jebc)-...
                              exbcr(1:iebc,1));

%    BACK

hzybcb(1:iefbc,2:jebc)=dahzybcb(1:iefbc,2:jebc).*hzybcb(1:iefbc,2:jebc)-...
  dbhzybcb(1:iefbc,2:jebc).*(exbcb(1:iefbc,2:jebc)-exbcb(1:iefbc,3:jbbc));
hzybcb(1:iebc,1)=dahzybcb(1:iebc,1).*hzybcb(1:iebc,1)-...
  dbhzybcb(1:iebc,1).*(exbcl(1:iebc,jb)-exbcb(1:iebc,2));
```

```
hzybcb(iebc+1:iebc+ie,1)=...
  dahzybcb(iebc+1:iebc+ie,1).*hzybcb(iebc+1:iebc+ie,1)-...
  dbhzybcb(iebc+1:iebc+ie,1).*(ex(1:ie,jb)-exbcb(iebc+1:iebc+ie,2));
hzybcb(iebc+ie+1:iefbc,1)=...
  dahzybcb(iebc+ie+1:iefbc,1).*hzybcb(iebc+ie+1:iefbc,1)-...
  dbhzybcb(iebc+ie+1:iefbc,1).*(exbcr(1:iebc,jb)-...
                              exbcb(iebc+ie+1:iefbc,2));

%    LEFT

hzybcl(:,1:je)=dahzybcl(:,1:je).*hzybcl(:,1:je)-...
  dbhzybcl(:,1:je).*(exbcl(:,1:je)-exbcl(:,2:jb));

%    RIGHT

hzybcr(:,1:je)=dahzybcr(:,1:je).*hzybcr(:,1:je)-...
  dbhzybcr(:,1:je).*(exbcr(:,1:je)-exbcr(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************

if mod(n,8)==0;

timestep=int2str(n);

subplot(3,1,1),pcolor(ex');
colormap hot %(added)
shading flat;
caxis([-80.0 80.0]);
axis([1 ie 1 jb]);
colorbar;
axis image;
axis off;
title(['Ex at time step = ',timestep]);

subplot(3,1,2),pcolor(ey');
shading flat;
caxis([-80.0 80.0]);
axis([1 ib 1 je]);
colorbar;
axis image;
axis off;
title(['Ey at time step = ',timestep]);

subplot(3,1,3),pcolor(hz');
shading flat;
caxis([-0.2 0.2]);
axis([1 ie 1 je]);
colorbar;
axis image;
axis off;
title(['Hz at time step = ',timestep]);

nn=n/8;
M(:,nn)=getframe(gcf,rect);

end;

%***********************************************************************
%    END TIME-STEPPING LOOP
%***********************************************************************

end

movie(gcf,M,0,10,rect);
```

```
%Filename: onewl1_new.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%         Department of Electrical and Computer Engineering
%         University of Wisconsin-Madison
%         1415 Engineering Drive
%         Madison, WI 53706-1691
%         608-265-5739
%         hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%   Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 10e9;%center frequency of source excitation
lambda = light_velocity_freespace/source_freq;%center wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%   Grid parameters
%***********************************************************************

i_end = 100;%number of grid cells in x-direction
j_end = 50;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 15;%location of z-directed hard source
j_source_location = j_end/2;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 176;%total number of time steps

PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%   Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%   Wave excitation
%***********************************************************************
```

```
rtau = 160e-12;%the 1/e point of a bandpass Gaussian pulse in second
tau = rtau/dt;

source = zeros(1,time_step_max);
n_index = [1:1*tau];
source(n_index) = sin(source_omega.*(n_index).*dt);

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add PEC with slit of width equals one lambda

wall_at_x_axis = 80;
ca_ex(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ex(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)
ca_ey(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ey(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
%***********************************************************************

del_bc = PML_LR*dx;
```

```
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%     LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%     RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
```

```
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

%screen_size = [195 190 420 190];
%screen_size = [1 29 800 553];
screen_size = [152 32 478 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- Metal Sheet, Sine wave'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
            cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
            cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex(wall_at_x_axis,[2:20,30:j_end]) = 0;%added
   ey(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

   ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
     cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
     hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

   ex_bc_f(wall_at_x_axis+PML_FB,2:PML_FB) = 0;%added

   ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
     cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
     hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%
%

%    BACK

   ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
     cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
     hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

   ex_bc_b(wall_at_x_axis+PML_FB,2:PML_FB-1) = 0;%added

   ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
     cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
     hzy_bc_b(ib_bc:PML_LR+i_end,1));

%
%

%    LEFT

   ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
     cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
     hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

   ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
     cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
     hzx_bc_l(:,1)-hzy_bc_l(:,1));

   ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
     cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
     hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

%    RIGHT

   ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
     cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
     hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

   ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
     cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
     hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

   ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
     cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
     hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
     hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%    Update EY in PML regions
%***********************************************************************

%    FRONT

   ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
     ey_bc_f(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
     hzy_bc_f(2:i_main_grid_plus_PML,:)-...
     hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

   ey_bc_f(wall_at_x_axis+PML_FB,:) = 0;%added

%    BACK

   ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
     ey_bc_b(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
     hzy_bc_b(2:i_main_grid_plus_PML,:)-...
```

```
     hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));

   ey_bc_b(wall_at_x_axis+PML_FB,:) = 0;%added

%    LEFT

   ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
     cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
     hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

   ey(1,:) = ca_ey(1,:).*ey(1,:)-...
     cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

%    RIGHT

   ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
     cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
     hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

   ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
     cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

   hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
     db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
     ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

   hz(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

   hz(i_source_location,j_source_location) = source(n);

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

   hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
     hzx_bc_f(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
     ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

   hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
     hzx_bc_b(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
     ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

   hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
     db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

   hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
     db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

   hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
     db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

   hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
     db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

   hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
```

```
    db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));

  hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
    db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));

  hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
    db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

  hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
    da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
    db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

  hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
    da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

  hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
    db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

  hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
    db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

  hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
    da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
    db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

  hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
    db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

  hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
    db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
  number_of_skip_frame = 4;

  if mod(n,number_of_skip_frame) == 0,

     timestep = int2str(n);
     colormap hot %(added)

     subplot(3,1,1),pcolor(ex')
     shading flat
     caxis([-80.0 80.0])
     axis([1 i_end 1 jb])
     colorbar
     axis image
     axis off
     title(['E_x at time step = ',timestep])

     subplot(3,1,2),pcolor(ey')
     shading flat
     caxis([-80.0 80.0])
     axis([1 ib 1 j_end])
     colorbar
     axis image
     axis off
     title(['E_y at time step = ',timestep])

     subplot(3,1,3),pcolor(hz')
     shading flat
     caxis([-0.2 0.2])
     axis([1 i_end 1 j_end])
     colorbar
     axis image
     axis off
     title(['H_z at time step = ',timestep])

     rect = get(h1,'Position');
     rect(1:2) = [0 0];%change the position of the frame
                 %with respect to the current figure window

     M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%***********************************************************************
%    END TIME-STEPPING LOOP
%***********************************************************************

end

movie(h1,M,0,10,rect)
```

```
%Filename: onewl2_new.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%     Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 10e9;%center frequency of source excitation
lambda = light_velocity_freespace/source_freq;%center wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

i_end = 100;%number of grid cells in x-direction
j_end = 50;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 15;%location of z-directed hard source
j_source_location = j_end/2;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 248;%total number of time steps

PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%     Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%     Wave excitation
%***********************************************************************
```

```
rtau = 160e-12;%the 1/e point of a bandpass Gaussian pulse in second
tau = rtau/dt;
%this delay is needed for a smooth transition from zero into the pulse
delay = 3*tau;

source = zeros(1,time_step_max);
n_index = [1:4*tau];
source(n_index) = sin(source_omega.*(n_index-delay).*dt).*...
   exp(-(((n_index-delay)./tau).^2));

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add PEC with slit of width equals one lambda

wall_at_x_axis = 80;
ca_ex(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ex(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)
ca_ey(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ey(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
```

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

del_bc = PML_LR*dx;
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%     LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%     RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
```

```
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

%screen_size = [195 190 420 190];
%screen_size = [1 29 800 553];
screen_size = [152 32 478 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- Metal Sheet, Bandpass Gaussian Pulse'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
            cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
            cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex(wall_at_x_axis,[2:20,30:j_end]) = 0;%added
   ey(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT
```

```
  ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
    cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
    hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

  ex_bc_f(wall_at_x_axis+PML_FB,2:PML_FB) = 0;%added

  ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
    cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
    hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%
%

%    BACK

  ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
    cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
    hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

  ex_bc_b(wall_at_x_axis+PML_FB,2:PML_FB-1) = 0;%added

  ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
    cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
    hzy_bc_b(ib_bc:PML_LR+i_end,1));

%
%

%    LEFT

  ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
    cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
    hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

  ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
    cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
    hzx_bc_l(:,1)-hzy_bc_l(:,1));

  ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
    cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
    hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

%    RIGHT

  ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
    cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
    hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

  ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
    cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
    hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

  ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
    cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
    hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
    hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%    Update EY in PML regions
%***********************************************************************

%    FRONT

  ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
    ey_bc_f(2:i_main_grid_plus_PML,:)-...
    cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
    hzy_bc_f(2:i_main_grid_plus_PML,:)-...
    hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

  ey_bc_f(wall_at_x_axis+PML_FB,:) = 0;%added

%    BACK

  ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
```

```
    ey_bc_b(2:i_main_grid_plus_PML,:)-...
    cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
    hzy_bc_b(2:i_main_grid_plus_PML,:)-...
    hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));

  ey_bc_b(wall_at_x_axis+PML_FB,:) = 0;%added

%    LEFT

  ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
    cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
    hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

  ey(1,:) = ca_ey(1,:).*ey(1,:)-...
    cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

%    RIGHT

  ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
    cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
    hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

  ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
    cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

  hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
    db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
    ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

  hz(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

  hz(i_source_location,j_source_location) = source(n);

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

  hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
    hzx_bc_f(1:i_main_grid_plus_PML,:)-...
    db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
    ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

  hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
    hzx_bc_b(1:i_main_grid_plus_PML,:)-...
    db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
    ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

  hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
    db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

  hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
    db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

  hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
    db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

  hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
    db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************
```

```
%    FRONT

   hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
     db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));

   hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
     db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));

   hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
     hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
     db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
     (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

   hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
     da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
     hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
     db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
     (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

   hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
     da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
     hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
     db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
     (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
     ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

   hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
     db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

   hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
     hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
     db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
     (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

   hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
     da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
     hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
     db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
     (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

   hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
     db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

   hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
     db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
   number_of_skip_frame = 4;

   if mod(n,number_of_skip_frame) == 0,

      timestep = int2str(n);
      colormap hot %(added)

      subplot(3,1,1),pcolor(ex')
      shading flat
      caxis([-80.0 80.0])
      axis([1 i_end 1 jb])
      colorbar
      axis image
      axis off
      title(['E_x at time step = ',timestep])

      subplot(3,1,2),pcolor(ey')
      shading flat
      caxis([-80.0 80.0])
      axis([1 ib 1 j_end])
```

```
    colorbar
    axis image
    axis off
    title(['E_y at time step = ',timestep])

    subplot(3,1,3),pcolor(hz')
    shading flat
    caxis([-0.2 0.2])
    axis([1 i_end 1 j_end])
    colorbar
    axis image
    axis off
    title(['H_z at time step = ',timestep])

    rect = get(h1,'Position');
    rect(1:2) = [0 0];%change the position of the frame
                  %with respect to the current figure window

    M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%***********************************************************************
%    END TIME-STEPPING LOOP
%***********************************************************************

end

movie(h1,M,0,10,rect)
```

```
%Filename: onewl1_wall.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%            Department of Electrical and Computer Engineering
%            University of Wisconsin-Madison
%            1415 Engineering Drive
%            Madison, WI 53706-1691
%            608-265-5739
%            hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%    Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 10e9;%center frequency of source excitation
lambda = light_velocity_freespace/source_freq;%center wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%    Grid parameters
%***********************************************************************

i_end = 100;%number of grid cells in x-direction
j_end = 50;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 15;%location of z-directed hard source
j_source_location = j_end/2;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 176;%total number of time steps

PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%    Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%    Wave excitation
%***********************************************************************
```

```
rtau = 160e-12;%the 1/e point of a bandpass Gaussian pulse in second
tau = rtau/dt;

source = zeros(1,time_step_max);
n_index = [1:1*tau];
source(n_index) = sin(source_omega.*(n_index).*dt);

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add PEC with slit of width equals one lambda

wall_at_x_axis = [70:80];
ca_ex(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ex(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)
ca_ey(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ey(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
%***********************************************************************

del_bc = PML_LR*dx;
```

```
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%     LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%     RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
```

```
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%    Movie initialization
%***********************************************************************

%screen_size = [195 190 420 190];
%screen_size = [1 29 800 553];
screen_size = [152 32 478 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- Metal Wall, Sine wave'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%    BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%    Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
           cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
           cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex(wall_at_x_axis,[2:20,30:j_end]) = 0;%added
   ey(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

%***********************************************************************
%    Update EX in PML regions
%***********************************************************************

%    FRONT

   ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
     cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
     hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

   ex_bc_f(wall_at_x_axis+PML_FB,2:PML_FB) = 0;%added

   ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
     cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
     hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%
%

%    BACK

   ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
     cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
     hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

   ex_bc_b(wall_at_x_axis+PML_FB,2:PML_FB-1) = 0;%added

   ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
     cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
     hzy_bc_b(ib_bc:PML_LR+i_end,1));

%
%

%    LEFT

   ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
     cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
     hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

   ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
     cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
     hzx_bc_l(:,1)-hzy_bc_l(:,1));

   ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
     cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
     hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

%    RIGHT

   ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
     cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
     hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

   ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
     cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
     hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

   ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
     cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
     hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
     hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%    Update EY in PML regions
%***********************************************************************

%    FRONT

   ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
     ey_bc_f(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
     hzy_bc_f(2:i_main_grid_plus_PML,:)-...
     hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

   ey_bc_f(wall_at_x_axis+PML_FB,:) = 0;%added

%    BACK

   ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
     ey_bc_b(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
     hzy_bc_b(2:i_main_grid_plus_PML,:)-...
     hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));

   ey_bc_b(wall_at_x_axis+PML_FB,:) = 0;%added

%    LEFT

   ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
     cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
     hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

   ey(1,:) = ca_ey(1,:).*ey(1,:)-...
     cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

%    RIGHT

   ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
     cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
     hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

   ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
     cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

   hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
     db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
     ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

   hz(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

   hz(i_source_location,j_source_location) = source(n);

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

   hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
     hzx_bc_f(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
     ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

   hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
     hzx_bc_b(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
     ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

   hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
     db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

   hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
     db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

   hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
     db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

   hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
     db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

   hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
```

```
    db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));

  hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
    db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));

  hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
    db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

  hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
    da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
    db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

  hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
    da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

  hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
    db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

  hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
    db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

  hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
    da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
    db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

  hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
    db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

  hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
    db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
  number_of_skip_frame = 4;

  if mod(n,number_of_skip_frame) == 0,

     timestep = int2str(n);
     colormap hot %(added)

     subplot(3,1,1),pcolor(ex')
     shading flat
     caxis([-80.0 80.0])
     axis([1 i_end 1 jb])
     colorbar
     axis image
     axis off
     title(['E_x at time step = ',timestep])

     subplot(3,1,2),pcolor(ey')
     shading flat
     caxis([-80.0 80.0])
     axis([1 ib 1 j_end])
     colorbar
     axis image
     axis off
     title(['E_y at time step = ',timestep])

     subplot(3,1,3),pcolor(hz')
     shading flat
     caxis([-0.2 0.2])
     axis([1 i_end 1 j_end])
     colorbar
     axis image
     axis off
     title(['H_z at time step = ',timestep])

     rect = get(h1,'Position');
     rect(1:2) = [0 0];%change the position of the frame
                 %with respect to the current figure window

     M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(h1,M,0,10,rect)
```

```
%Filename: onewl2_wall.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%             Department of Electrical and Computer Engineering
%             University of Wisconsin-Madison
%             1415 Engineering Drive
%             Madison, WI 53706-1691
%             608-265-5739
%             hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%     Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 10e9;%center frequency of source excitation
lambda = light_velocity_freespace/source_freq;%center wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

i_end = 100;%number of grid cells in x-direction
j_end = 50;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 15;%location of z-directed hard source
j_source_location = j_end/2;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 240;%total number of time steps

PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%     Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%     Wave excitation
%***********************************************************************
```

```
rtau = 160e-12;%the 1/e point of a bandpass Gaussian pulse in second
tau = rtau/dt;
%this delay is needed for a smooth transition from zero into the pulse
delay = 3*tau;

source = zeros(1,time_step_max);
n_index = [1:4*tau];
source(n_index) = sin(source_omega.*(n_index-delay).*dt).*...
   exp(-(((n_index-delay)./tau).^2));

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add PEC with slit of width equals one lambda

wall_at_x_axis = [70:80];
ca_ex(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ex(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)
ca_ey(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ey(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
```

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

del_bc = PML_LR*dx;
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%     LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%     RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
```

```
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

%screen_size = [195 190 420 190];
%screen_size = [1 29 800 553];
screen_size = [152 32 478 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- Metal Wall, Bandpass Gaussian Pulse'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
            cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
            cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex(wall_at_x_axis,[2:20,30:j_end]) = 0;%added
   ey(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT
```

```
   ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
     cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
     hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

   ex_bc_f(wall_at_x_axis+PML_FB,2:PML_FB) = 0;%added

   ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
     cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
     hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%
%

%    BACK

   ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
     cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
     hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

   ex_bc_b(wall_at_x_axis+PML_FB,2:PML_FB-1) = 0;%added

   ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
     cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
     hzy_bc_b(ib_bc:PML_LR+i_end,1));

%
%

%    LEFT

   ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
     cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
     hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

   ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
     cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
     hzx_bc_l(:,1)-hzy_bc_l(:,1));

   ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
     cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
     hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

%    RIGHT

   ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
     cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
     hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

   ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
     cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
     hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

   ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
     cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
     hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
     hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%    Update EY in PML regions
%***********************************************************************

%    FRONT

   ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
     ey_bc_f(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
     hzy_bc_f(2:i_main_grid_plus_PML,:)-...
     hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

   ey_bc_f(wall_at_x_axis+PML_FB,:) = 0;%added

%    BACK

   ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
```

```
    ey_bc_b(2:i_main_grid_plus_PML,:)-...
    cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
    hzy_bc_b(2:i_main_grid_plus_PML,:)-...
    hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));

  ey_bc_b(wall_at_x_axis+PML_FB,:) = 0;%added

%    LEFT

  ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
    cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
    hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

  ey(1,:) = ca_ey(1,:).*ey(1,:)-...
    cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

%    RIGHT

  ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
    cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
    hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

  ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
    cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

  hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
    db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
    ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

  hz(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

  hz(i_source_location,j_source_location) = source(n);

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

  hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
    hzx_bc_f(1:i_main_grid_plus_PML,:)-...
    db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
    ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

  hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
    hzx_bc_b(1:i_main_grid_plus_PML,:)-...
    db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
    ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

  hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
    db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

  hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
    db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

  hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
    db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

  hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
    db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************
```

```
%    FRONT

  hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
    db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));

  hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
    db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));

  hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
    db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

  hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
    da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
    db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

  hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
    da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

  hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
    db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

  hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
    db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

  hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
    da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
    db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

  hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
    db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

  hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
    db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
  number_of_skip_frame = 4;

  if mod(n,number_of_skip_frame) == 0,

     timestep = int2str(n);
     colormap hot %(added)

     subplot(3,1,1),pcolor(ex')
     shading flat
     caxis([-80.0 80.0])
     axis([1 i_end 1 jb])
     colorbar
     axis image
     axis off
     title(['E_x at time step = ',timestep])

     subplot(3,1,2),pcolor(ey')
     shading flat
     caxis([-80.0 80.0])
     axis([1 ib 1 j_end])
     colorbar
     axis image
     axis off
     title(['E_y at time step = ',timestep])

     subplot(3,1,3),pcolor(hz')
     shading flat
     caxis([-0.2 0.2])
     axis([1 i_end 1 j_end])
     colorbar
     axis image
     axis off
     title(['H_z at time step = ',timestep])

     rect = get(h1,'Position');
     rect(1:2) = [0 0];%change the position of the frame
                  %with respect to the current figure window

     M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(h1,M,0,10,rect)
```

```
%Filename: fdtd2d_gaussian_planewave.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%             Department of Electrical and Computer Engineering
%             University of Wisconsin-Madison
%             1415 Engineering Drive
%             Madison, WI 53706-1691
%             608-265-5739
%             hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%     Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

source_freq = 10e9;%center frequency of source excitation
lambda = light_velocity_freespace/source_freq;%center wavelength of source excitation
source_omega = 2*pi*source_freq;

%***********************************************************************
%     Grid parameters
%***********************************************************************

i_end = 100;%number of grid cells in x-direction
j_end = 50;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 15;%location of z-directed hard source
j_source_location = j_end/2;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 240;%total number of time steps

PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%     Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%     Wave excitation
%***********************************************************************
```

```
rtau = 160e-12;%the 1/e point of a bandpass Gaussian pulse in second
tau = rtau/dt;
%this delay is needed for a smooth transition from zero into the pulse
delay = 3*tau;

source = zeros(1,time_step_max);
n_index = [1:4*tau];
source(n_index) = sin(source_omega.*(n_index-delay).*dt).*...
   exp(-(((n_index-delay)./tau).^2));

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add PEC with slit of width equals one lambda

wall_at_x_axis = [70:80];
ca_ex(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ex(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)
ca_ey(wall_at_x_axis,[1:20,30:j_end]) = ca(2); %(added)
cb_ey(wall_at_x_axis,[1:20,30:j_end]) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
```

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

del_bc = PML_LR*dx;
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%    LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%    RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
```

```
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

screen_size = [152 32 610 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- Metal Wall, Plane wave using bandpass Gaussian pulse'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
           cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
           cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex(wall_at_x_axis,[2:20,30:j_end]) = 0;%added
   ey(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

   ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
```

```
     cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
     hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

   ex_bc_f(wall_at_x_axis+PML_FB,2:PML_FB) = 0;%added

   ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
     cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
     hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%    BACK

   ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
     cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
     hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

   ex_bc_b(wall_at_x_axis+PML_FB,2:PML_FB-1) = 0;%added

   ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
     cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
     hzy_bc_b(ib_bc:PML_LR+i_end,1));

%    LEFT

   ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
     cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
     hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

   ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
     cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
     hzx_bc_l(:,1)-hzy_bc_l(:,1));

   ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
     cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
     hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

%    RIGHT

   ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
     cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
     hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

   ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
     cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
     hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

   ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
     cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
     hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
     hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%    Update EY in PML regions
%***********************************************************************

%    FRONT

   ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
     ey_bc_f(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
     hzy_bc_f(2:i_main_grid_plus_PML,:)-...
     hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

   ey_bc_f(wall_at_x_axis+PML_FB,:) = 0;%added

%    BACK

   ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
     ey_bc_b(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
     hzy_bc_b(2:i_main_grid_plus_PML,:)-...
     hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));

   ey_bc_b(wall_at_x_axis+PML_FB,:) = 0;%added

%    LEFT
```

```
  ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
    cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
    hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

  ey(1,:) = ca_ey(1,:).*ey(1,:)-...
    cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

%    RIGHT

  ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
    cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
    hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

  ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
    cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

  hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
    db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
    ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

  hz(wall_at_x_axis,[1:20,30:j_end]) = 0;%added

  hz(15,1:10:j_end) = source(n);


%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

  hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
    hzx_bc_f(1:i_main_grid_plus_PML,:)-...
    db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
    ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

  hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
    hzx_bc_b(1:i_main_grid_plus_PML,:)-...
    db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
    ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

  hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
    db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

  hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
    db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

  hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
    db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

  hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
    db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

  hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
    db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));

  hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
    db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));
  hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
    db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

  hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
    da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
    db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

  hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
    da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

  hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
    db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

  hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
    db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

  hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
    da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
    db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

  hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
    db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

  hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
    db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
  number_of_skip_frame = 4;

  if mod(n,number_of_skip_frame) == 0,

    timestep = int2str(n);
    colormap hot %(added)

    subplot(3,1,1),pcolor(ex')
    shading flat
    caxis([-80.0 80.0])
    axis([1 i_end 1 jb])
    colorbar
    axis image
    axis off
    title(['E_x at time step = ',timestep])

    subplot(3,1,2),pcolor(ey')
    shading flat
    caxis([-80.0 80.0])
    axis([1 ib 1 j_end])
    colorbar
    axis image
    axis off
    title(['E_y at time step = ',timestep])

    subplot(3,1,3),pcolor(hz')
    shading flat
    caxis([-0.2 0.2])
    axis([1 i_end 1 j_end])
    colorbar
    axis image
    axis off
    title(['H_z at time step = ',timestep])

    rect = get(h1,'Position');
    rect(1:2) = [0 0];%change the position of the frame
                %with respect to the current figure window

    M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(h1,M,0,10,rect)
```

```
%Filename: fdtd2d_fix_screen.m
%Date of creation: November 2006

clear all
close all
pack
clc

x = -(4*pi):0.1:(4*pi);
y = x;

[x,y] = meshgrid(x,y);

hz = sin(((x.^2)+(y.^2)).^(1/2));

ex = y.*cos(((x.^2)+(y.^2)).^(1/2));

ey = (-x).*cos(((x.^2)+(y.^2)).^(1/2));

title_text = ['- Sine wave'];

h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',[1 29 800 504],...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

meshc(x,y,hz)
title(['H_z'])
xlabel('x-axis'),ylabel('y-axis'),zlabel('z-axis')

h2 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',[50 29 800 504],...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

meshc(x,y,ex)
title(['E_x'])
xlabel('x-axis'),ylabel('y-axis'),zlabel('z-axis')

h3 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',[100 29 800 504],...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

meshc(x,y,ey)
title(['E_y'])
xlabel('x-axis'),ylabel('y-axis'),zlabel('z-axis')
```

Appendix D

Two-dimensional FDTD simulation of TDR excited by a raised cosine time waveform Source Code

```
%Filename: fdtd2d_tdr_ver1.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%    Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

%***********************************************************************
%    Grid parameters
%***********************************************************************

i_end = 75;%number of grid cells in x-direction
j_end = 27;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 1;%location of z-directed hard source
j_source_location = 14;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 160;%total number of time steps


PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%    Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%    Wave excitation
%***********************************************************************

T = 40e-12;
dT = T/dt;
```

```
n_index = [1:time_step_max];
source = zeros(size(n_index));

source = 0.5.*(1 - cos(pi.*n_index.*dt./T));

source(find(n_index >= dT)) = 1;%fill those value behind T with 1
source(find(n_index >= (dT+1))) = 0;%fill those value behind that grid with 0

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add PEC with slit of width equals one lambda

wall_at_y_axis = [9:10,18:19];
ca_ex([1:67],wall_at_y_axis) = ca(2); %(added)
cb_ex([1:67],wall_at_y_axis) = cb(2); %(added)
ca_ey([1:67],wall_at_y_axis) = ca(2); %(added)
cb_ey([1:67],wall_at_y_axis) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
%***********************************************************************
```

```
del_bc = PML_LR*dx;
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%     LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%     RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
```

```
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

%screen_size = [195 190 420 190];
%screen_size = [1 29 800 553];
screen_size = [152 32 478 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- TDR, Step function using raised cosine'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
             cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
             cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex([2:67],wall_at_y_axis) = 0;%added
   ey([1:67],wall_at_y_axis) = 0;%added

%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

   ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
```

```
    cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
    hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

%   ex_bc_f(wall_at_y_axis+PML_FB,2:PML_FB) = 0;%added

  ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
    cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
    hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%    BACK

  ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
    cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
    hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

%   ex_bc_b(wall_at_y_axis+PML_FB,2:PML_FB-1) = 0;%added

  ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
    cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
    hzy_bc_b(ib_bc:PML_LR+i_end,1));

%    LEFT

  ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
    cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
    hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

  ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
    cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
    hzx_bc_l(:,1)-hzy_bc_l(:,1));

  ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
    cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
    hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

%    RIGHT

  ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
    cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
    hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

  ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
    cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
    hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

  ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
    cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
    hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
    hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%    Update EY in PML regions
%***********************************************************************

%    FRONT

  ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
    ey_bc_f(2:i_main_grid_plus_PML,:)-...
    cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
    hzy_bc_f(2:i_main_grid_plus_PML,:)-...
    hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

%   ey_bc_f(wall_at_y_axis+PML_FB,:) = 0;%added

%    BACK

  ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
    ey_bc_b(2:i_main_grid_plus_PML,:)-...
    cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
    hzy_bc_b(2:i_main_grid_plus_PML,:)-...
    hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));

%   ey_bc_b(wall_at_y_axis+PML_FB,:) = 0;%added

%    LEFT
```

```
   ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
     cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
     hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

   ey(1,:) = ca_ey(1,:).*ey(1,:)-...
     cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

%    RIGHT

   ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
     cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
     hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

   ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
     cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

   hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
     db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
     ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

   hz([1:67],wall_at_y_axis) = 0;%added

   hz(i_source_location,j_source_location) = source(n);%changed

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

   hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
     hzx_bc_f(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
     ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

   hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
     hzx_bc_b(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
     ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

   hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
     db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

   hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
     db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

   hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
     db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

   hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
     db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

   hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
     db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));

   hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
     db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));
```

```
  hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
    db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

  hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
    da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
    db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

  hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
    da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

  hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
    db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

  hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
    db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

  hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
    da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
    db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

  hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
    db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

  hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
    db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
  number_of_skip_frame = 4;

  if mod(n,number_of_skip_frame) == 0,

    timestep = int2str(n);
    colormap hot %(added)

    subplot(3,1,1),pcolor(ex')
    shading flat
    caxis([-80.0 80.0])
    axis([1 i_end 1 jb])
    colorbar
    axis image
    axis off
    title(['E_x at time step = ',timestep])

    subplot(3,1,2),pcolor(ey')
    shading flat
    caxis([-80.0 80.0])
    axis([1 ib 1 j_end])
    colorbar
    axis image
    axis off
    title(['E_y at time step = ',timestep])

    subplot(3,1,3),pcolor(hz')
    shading flat
    caxis([-0.2 0.2])
    axis([1 i_end 1 j_end])
    colorbar
    axis image
    axis off
    title(['H_z at time step = ',timestep])

    rect = get(h1,'Position');
    rect(1:2) = [0 0];%change the position of the frame
                %with respect to the current figure window

    M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(h1,M,0,10,rect)
```

```
%Filename: fdtd2d_tdr_ver2.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%    Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

%***********************************************************************
%    Grid parameters
%***********************************************************************

i_end = 75;%number of grid cells in x-direction
j_end = 27;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 1;%location of z-directed hard source
j_source_location = 14;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 160;%total number of time steps


PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%    Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%    Wave excitation
%***********************************************************************

T = 40e-12;
dT = T/dt;
```

```
n_index = [1:time_step_max];
source = zeros(size(n_index));

source = 0.5.*(1 - cos(pi.*n_index.*dt./T));

source(find(n_index >= dT)) = 1;%fill those value behind T with 1
source(find(n_index >= (dT+1))) = 0;%fill those value behind that grid with 0

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add PEC with slit of width equals one lambda

wall_at_y_axis = [9:10,18:19];
ca_ex([1:67],wall_at_y_axis) = ca(2); %(added)
cb_ex([1:67],wall_at_y_axis) = cb(2); %(added)
ca_ey([1:67],wall_at_y_axis) = ca(2); %(added)
cb_ey([1:67],wall_at_y_axis) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
%***********************************************************************
```

```
del_bc = PML_LR*dx;
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%     LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%     RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
```

```
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

%screen_size = [195 190 420 190];
%screen_size = [1 29 800 553];
screen_size = [152 32 478 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- TDR, Step function using raised cosine'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
            cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
            cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex([2:67],wall_at_y_axis) = 0;%added
   ey([1:67],wall_at_y_axis) = 0;%added

%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT

   ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
```

```
     cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
     hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

%    ex_bc_f(wall_at_y_axis+PML_FB,2:PML_FB) = 0;%added

   ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
     cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
     hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%     BACK

   ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
     cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
     hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

%    ex_bc_b(wall_at_y_axis+PML_FB,2:PML_FB-1) = 0;%added

   ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
     cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
     hzy_bc_b(ib_bc:PML_LR+i_end,1));

%     LEFT

   ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
     cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
     hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

   ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
     cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
     hzx_bc_l(:,1)-hzy_bc_l(:,1));

   ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
     cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
     hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

   ex_bc_l(:,:) = 0;%%%added

%     RIGHT

   ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
     cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
     hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

   ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
     cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
     hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

   ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
     cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
     hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
     hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%     Update EY in PML regions
%***********************************************************************

%     FRONT

   ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
     ey_bc_f(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
     hzy_bc_f(2:i_main_grid_plus_PML,:)-...
     hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

%    ey_bc_f(wall_at_y_axis+PML_FB,:) = 0;%added

%     BACK

   ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
     ey_bc_b(2:i_main_grid_plus_PML,:)-...
     cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
     hzy_bc_b(2:i_main_grid_plus_PML,:)-...
     hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));

%    ey_bc_b(wall_at_y_axis+PML_FB,:) = 0;%added
%    LEFT

   ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
     cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
     hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

   ey(1,:) = ca_ey(1,:).*ey(1,:)-...
     cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

   ey_bc_l(2:PML_LR,:) = 0;%%%added

%    RIGHT

   ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
     cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
     hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

   ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
     cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

   hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
     db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
     ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

   hz([1:67],wall_at_y_axis) = 0;%added

   hz(i_source_location,j_source_location) = source(n);%changed

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

   hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
     hzx_bc_f(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
     ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

   hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
     hzx_bc_b(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
     ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

   hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
     db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

   hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
     db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

   hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
     db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

   hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
     db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT

   hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
     db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));
```

```
  hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
    db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));

  hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
    db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

  hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
    da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
    db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

  hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
    da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

  hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
    db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

  hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
    db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

  hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
    da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
    db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

  hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
    db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

  hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
    db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
  number_of_skip_frame = 4;

  if mod(n,number_of_skip_frame) == 0,

     timestep = int2str(n);
     colormap hot %(added)

     subplot(3,1,1),pcolor(ex')
     shading flat
     caxis([-80.0 80.0])
     axis([1 i_end 1 jb])
     colorbar
     axis image
     axis off
     title(['E_x at time step = ',timestep])

     subplot(3,1,2),pcolor(ey')
     shading flat
     caxis([-80.0 80.0])
     axis([1 ib 1 j_end])
     colorbar
     axis image
     axis off
     title(['E_y at time step = ',timestep])
     subplot(3,1,3),pcolor(hz')
     shading flat
     caxis([-0.2 0.2])
     axis([1 i_end 1 j_end])
     colorbar
     axis image
     axis off
     title(['H_z at time step = ',timestep])

     rect = get(h1,'Position');
     rect(1:2) = [0 0];%change the position of the frame
                 %with respect to the current figure window

     M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%***********************************************************************
%     END TIME-STEPPING LOOP
%***********************************************************************

end

movie(h1,M,0,10,rect)
```

```
%Filename: fdtd2d_tdr_ver3.m
%Date of creation: November 2006
%
%Adapted from fdtd2D.m written by Susan C. Hagness
% Program author: Susan C. Hagness
%           Department of Electrical and Computer Engineering
%           University of Wisconsin-Madison
%           1415 Engineering Drive
%           Madison, WI 53706-1691
%           608-265-5739
%           hagness@engr.wisc.edu
%
% Date of this version:  February 2000

clear all
close all
pack
clc

%***********************************************************************
%    Fundamental constants
%***********************************************************************

light_velocity_freespace = 2.99792458e8;%speed of light in free space
mu_0 = 4*pi*1e-7;%permeability of free space
epsilon_0 = 8.854187818e-12;%permittivity of free space

%***********************************************************************
%    Grid parameters
%***********************************************************************

i_end = 75;%number of grid cells in x-direction
j_end = 27;%number of grid cells in y-direction

ib = i_end+1;
jb = j_end+1;

i_source_location = 1;%location of z-directed hard source
j_source_location = 14;%location of z-directed hard source

dx = 3e-3;%space increment of square lattice in metre
dt = dx/(2*light_velocity_freespace);%time step in second

time_step_max = 348;%total number of time steps


PML_LR = 8;%thickness of left and right PML region
PML_FB = 8;%thickness of front and back PML region
r_max = 1e-5;
order_bc = 2;
ib_bc = PML_LR+1;
jb_bc = PML_FB+1;
i_main_grid_plus_PML = i_end + 2*PML_LR;%total grids along x-axis
j_main_grid_plus_PML = j_end + 2*PML_FB;%total grids along y-axis
ib_fbc = i_main_grid_plus_PML + 1;
jb_fbc = j_main_grid_plus_PML + 1;

%***********************************************************************
%    Material parameters
%***********************************************************************

number_of_media = 2;%1 = vaccum; 2 = metal

epsilon_r = [1 1];
sigma = [0 1e7];
mu_r = [1 1];
sigma_magnetic = [0 0];

%***********************************************************************
%    Wave excitation
%***********************************************************************

T = 40e-12;
dT = T/dt;
```

```
n_index = [1:time_step_max];
source = zeros(size(n_index));

source = 0.5.*(1 - cos(pi.*n_index.*dt./T));

source(find(n_index >= dT)) = 1;%fill those value behind T with 1
source(find(n_index >= (dT+1))) = 0;%fill those value behind that grid with 0

%***********************************************************************
%     Field arrays
%***********************************************************************

ex = zeros(i_end,jb);%fields in main grid
ey = zeros(ib,j_end);
hz = zeros(i_end,j_end);

ex_bc_f = zeros(i_main_grid_plus_PML,PML_FB);%fields in front PML region
ey_bc_f = zeros(ib_fbc,PML_FB);
hzx_bc_f = ex_bc_f;
hzy_bc_f = hzx_bc_f;

ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);%fields in back PML region
ey_bc_b = ey_bc_f;
hzx_bc_b = hzx_bc_f;
hzy_bc_b = hzy_bc_f;

ex_bc_l = zeros(PML_LR,jb);%fields in left PML region
ey_bc_l = zeros(PML_LR,j_end);
hzx_bc_l = ey_bc_l;
hzy_bc_l = hzx_bc_l;

ex_bc_r = ex_bc_l;%fields in right PML region
ey_bc_r = zeros(ib_bc,j_end);
hzx_bc_r = hzx_bc_l;
hzy_bc_r = hzy_bc_l;

%***********************************************************************
%     Updating coefficients
%***********************************************************************

epsilon = epsilon_0*epsilon_r;
mu = mu_0*mu_r;

ca = (1-(dt.*sigma)./(2.*epsilon))./(1+(dt.*sigma)./(2.*epsilon));
cb = (dt./(epsilon.*dx))./(1+(dt.*sigma)./(2.*epsilon));
da = (1-(dt.*sigma_magnetic)./(2.*mu))./(1+(dt.*sigma_magnetic)./(2.*mu));
db = (dt./(mu.*dx))./(1+(dt.*sigma_magnetic)./(2.*mu));

%***********************************************************************
%     Geometry specification (main grid)
%***********************************************************************

%     Initialize entire main grid to free space

ca_ex(1:i_end,1:jb) = ca(1);
cb_ex(1:i_end,1:jb) = cb(1);

ca_ey(1:ib,1:j_end) = ca(1);
cb_ey(1:ib,1:j_end) = cb(1);

da_hz(1:i_end,1:j_end) = da(1);
db_hz(1:i_end,1:j_end) = db(1);

%     Add two PEC rods

wall_at_y_axis = [9:10,18:19];
rod_start = 20;
rod_end = 67;
ca_ex([rod_start:rod_end],wall_at_y_axis) = ca(2); %(added)
cb_ex([rod_start:rod_end],wall_at_y_axis) = cb(2); %(added)
ca_ey([rod_start:rod_end],wall_at_y_axis) = ca(2); %(added)
cb_ey([rod_start:rod_end],wall_at_y_axis) = cb(2); %(added)

%***********************************************************************
%     Fill the PML regions
```

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

del_bc = PML_LR*dx;
sigmam = -log(r_max/100)*epsilon_0*light_velocity_freespace*(order_bc+1)/(2*del_bc);
bc_factor = epsilon_r(1)*sigmam/(dx*(del_bc^order_bc)*(order_bc+1));

%     FRONT region

ca_ex_bc_f = ones(i_main_grid_plus_PML,1);
cb_ex_bc_f = zeros(i_main_grid_plus_PML,1);
for j_index = 2:PML_FB,
  y1 = (PML_FB-j_index+1.5)*dx;
  y2 = (PML_FB-j_index+0.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_f(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_f(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,1) = ca1;
cb_ex(1:i_end,1) = cb1;
ca_ex_bc_l(1:PML_LR,1) = ca1;
cb_ex_bc_l(1:PML_LR,1) = cb1;
ca_ex_bc_r(1:PML_LR,1) = ca1;
cb_ex_bc_r(1:PML_LR,1) = cb1;

for j_index = 1:PML_FB,
  y1 = (PML_FB-j_index+1)*dx;
  y2 = (PML_FB-j_index)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_f(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_f(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_f(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_f(1:i_main_grid_plus_PML,j_index) = db(1);
end

%     BACK region

ca_ex_bc_b = ones(i_main_grid_plus_PML,jb_bc);
cb_ex_bc_b = zeros(i_main_grid_plus_PML,jb_bc);
for j_index=2:PML_FB,
  y1 = (j_index-0.5)*dx;
  y2 = (j_index-1.5)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmay*dx);
  ca_ex_bc_b(1:i_main_grid_plus_PML,j_index) = ca1;
  cb_ex_bc_b(1:i_main_grid_plus_PML,j_index) = cb1;
end
sigmay = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmay*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmay*dx);
ca_ex(1:i_end,jb) = ca1;
cb_ex(1:i_end,jb) = cb1;
ca_ex_bc_l(1:PML_LR,jb) = ca1;
cb_ex_bc_l(1:PML_LR,jb) = cb1;
ca_ex_bc_r(1:PML_LR,jb) = ca1;
cb_ex_bc_r(1:PML_LR,jb) = cb1;

for j_index = 1:PML_FB,
  y1 = j_index*dx;
  y2 = (j_index-1)*dx;
  sigmay = bc_factor*(y1^(order_bc+1)-y2^(order_bc+1));
  sigmays = sigmay*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmays*dt/mu_0);
  db1 = (1-da1)/(sigmays*dx);
  da_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = da1;
  db_hzy_bc_b(1:i_main_grid_plus_PML,j_index) = db1;
  ca_ey_bc_b(1:ib_fbc,j_index) = ca(1);
  cb_ey_bc_b(1:ib_fbc,j_index) = cb(1);
  da_hzx_bc_b(1:i_main_grid_plus_PML,j_index) = da(1);
  db_hzx_bc_b(1:i_main_grid_plus_PML,j_index)=db(1);
end

%    LEFT region

ca_ey_bc_l = ones(1,j_end);
cb_ey_bc_l = zeros(1,j_end);
for i = 2:PML_LR,
  x1 = (PML_LR-i+1.5)*dx;
  x2 = (PML_LR-i+0.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_l(i,1:j_end) = ca1;
  cb_ey_bc_l(i,1:j_end) = cb1;
  ca_ey_bc_f(i,1:PML_FB) = ca1;
  cb_ey_bc_f(i,1:PML_FB) = cb1;
  ca_ey_bc_b(i,1:PML_FB) = ca1;
  cb_ey_bc_b(i,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(1,1:j_end) = ca1;
cb_ey(1,1:j_end) = cb1;
ca_ey_bc_f(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+1,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+1,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+1,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = (PML_LR-i+1)*dx;
  x2 = (PML_LR-i)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_l(i,1:j_end) = da1;
  db_hzx_bc_l(i,1:j_end) = db1;
  da_hzx_bc_f(i,1:PML_FB) = da1;
  db_hzx_bc_f(i,1:PML_FB) = db1;
  da_hzx_bc_b(i,1:PML_FB) = da1;
  db_hzx_bc_b(i,1:PML_FB) = db1;
  ca_ex_bc_l(i,2:j_end) = ca(1);
  cb_ex_bc_l(i,2:j_end) = cb(1);
  da_hzy_bc_l(i,1:j_end) = da(1);
  db_hzy_bc_l(i,1:j_end) = db(1);
end

%    RIGHT region

ca_ey_bc_r = ones(ib_bc,j_end);
cb_ey_bc_r = zeros(ib_bc,j_end);
for i = 2:PML_LR,
  x1 = (i-0.5)*dx;
  x2 = (i-1.5)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
  cb1 = (1-ca1)/(sigmax*dx);
  ca_ey_bc_r(i,1:j_end) = ca1;
  cb_ey_bc_r(i,1:j_end) = cb1;
  ca_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_f(i+PML_LR+i_end,1:PML_FB) = cb1;
  ca_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = ca1;
  cb_ey_bc_b(i+PML_LR+i_end,1:PML_FB) = cb1;
end
sigmax = bc_factor*(0.5*dx)^(order_bc+1);
ca1 = exp(-sigmax*dt/(epsilon_0*epsilon_r(1)));
cb1 = (1-ca1)/(sigmax*dx);
ca_ey(ib,1:j_end) = ca1;
cb_ey(ib,1:j_end) = cb1;
```

```
ca_ey_bc_f(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_f(PML_LR+ib,1:PML_FB) = cb1;
ca_ey_bc_b(PML_LR+ib,1:PML_FB) = ca1;
cb_ey_bc_b(PML_LR+ib,1:PML_FB) = cb1;

for i = 1:PML_LR,
  x1 = i*dx;
  x2 = (i-1)*dx;
  sigmax = bc_factor*(x1^(order_bc+1)-x2^(order_bc+1));
  sigmaxs = sigmax*(mu_0/(epsilon_0*epsilon_r(1)));
  da1 = exp(-sigmaxs*dt/mu_0);
  db1 = (1-da1)/(sigmaxs*dx);
  da_hzx_bc_r(i,1:j_end) = da1;
  db_hzx_bc_r(i,1:j_end) = db1;
  da_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_f(i+i_end+PML_LR,1:PML_FB) = db1;
  da_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = da1;
  db_hzx_bc_b(i+i_end+PML_LR,1:PML_FB) = db1;
  ca_ex_bc_r(i,2:j_end) = ca(1);
  cb_ex_bc_r(i,2:j_end) = cb(1);
  da_hzy_bc_r(i,1:j_end) = da(1);
  db_hzy_bc_r(i,1:j_end) = db(1);
end

%***********************************************************************
%     Movie initialization
%***********************************************************************

%screen_size = [195 190 420 190];
%screen_size = [1 29 800 553];
screen_size = [152 32 478 546];

%sigma_text = num2str(sigma/(1e-3));
title_text = ['- TDR, Step function using raised cosine'];

%Property 'CloseRequestFcn', is set to 'disp("")'.
%This will disable the 'X' button on the top right-hand
%corner and the close command in MATLAB. Only delete command
%can close the figure
h1 = figure('Units','pixels',...
   'Name',['FDTD 2D demonstration ',title_text],...
   'NumberTitle','off',...
   'Color',[1 1 1],...
   'Position',screen_size,...
   'MenuBar','none',...
   'ToolBar','none',...
   'Resize','off',...
   'WindowStyle','normal');

%'CloseRequestFcn','disp("")'

%***********************************************************************
%     BEGIN TIME-STEPPING LOOP
%***********************************************************************

for n = 1:time_step_max,

%***********************************************************************
%     Update electric fields (EX and EY) in main grid
%***********************************************************************

   ex(:,2:j_end) = ca_ex(:,2:j_end).*ex(:,2:j_end)+...
           cb_ex(:,2:j_end).*(hz(:,2:j_end)-hz(:,1:j_end-1));

   ey(2:i_end,:) = ca_ey(2:i_end,:).*ey(2:i_end,:)+...
           cb_ey(2:i_end,:).*(hz(1:i_end-1,:)-hz(2:i_end,:));

   ex([rod_start+1:rod_end],wall_at_y_axis) = 0;%added
   ey([rod_start:rod_end],wall_at_y_axis) = 0;%added

%***********************************************************************
%     Update EX in PML regions
%***********************************************************************

%     FRONT
```

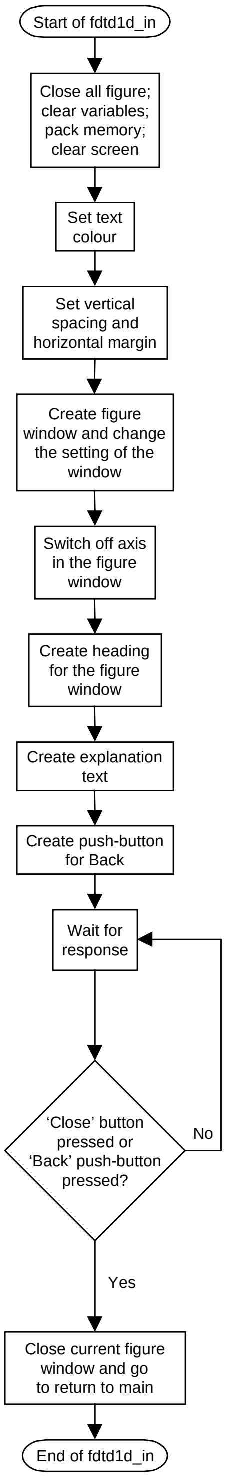
```
  ex_bc_f(:,2:PML_FB) = ca_ex_bc_f(:,2:PML_FB).*ex_bc_f(:,2:PML_FB)-...
    cb_ex_bc_f(:,2:PML_FB).*(hzx_bc_f(:,1:PML_FB-1)+hzy_bc_f(:,1:PML_FB-1)-...
    hzx_bc_f(:,2:PML_FB)-hzy_bc_f(:,2:PML_FB));

%   ex_bc_f(wall_at_y_axis+PML_FB,2:PML_FB) = 0;%added

  ex(1:i_end,1) = ca_ex(1:i_end,1).*ex(1:i_end,1)-...
    cb_ex(1:i_end,1).*(hzx_bc_f(ib_bc:PML_LR+i_end,PML_FB)+...
    hzy_bc_f(ib_bc:PML_LR+i_end,PML_FB)-hz(1:i_end,1));

%    BACK

  ex_bc_b(:,2:PML_FB-1) = ca_ex_bc_b(:,2:PML_FB-1).*ex_bc_b(:,2:PML_FB-1)-...
    cb_ex_bc_b(:,2:PML_FB-1).*(hzx_bc_b(:,1:PML_FB-2)+hzy_bc_b(:,1:PML_FB-2)-...
    hzx_bc_b(:,2:PML_FB-1)-hzy_bc_b(:,2:PML_FB-1));

%   ex_bc_b(wall_at_y_axis+PML_FB,2:PML_FB-1) = 0;%added

  ex(1:i_end,jb) = ca_ex(1:i_end,jb).*ex(1:i_end,jb)-...
    cb_ex(1:i_end,jb).*(hz(1:i_end,jb-1)-hzx_bc_b(ib_bc:PML_LR+i_end,1)-...
    hzy_bc_b(ib_bc:PML_LR+i_end,1));

%    LEFT

  ex_bc_l(:,2:j_end) = ca_ex_bc_l(:,2:j_end).*ex_bc_l(:,2:j_end)-...
    cb_ex_bc_l(:,2:j_end).*(hzx_bc_l(:,1:j_end-1)+hzy_bc_l(:,1:j_end-1)-...
    hzx_bc_l(:,2:j_end)-hzy_bc_l(:,2:j_end));

  ex_bc_l(:,1) = ca_ex_bc_l(:,1).*ex_bc_l(:,1)-...
    cb_ex_bc_l(:,1).*(hzx_bc_f(1:PML_LR,PML_FB)+hzy_bc_f(1:PML_LR,PML_FB)-...
    hzx_bc_l(:,1)-hzy_bc_l(:,1));

  ex_bc_l(:,jb) = ca_ex_bc_l(:,jb).*ex_bc_l(:,jb)-...
    cb_ex_bc_l(:,jb).*(hzx_bc_l(:,j_end)+hzy_bc_l(:,j_end)-...
    hzx_bc_b(1:PML_LR,1)-hzy_bc_b(1:PML_LR,1));

  ex_bc_l(:,:) = 0;%%%added

%    RIGHT

  ex_bc_r(:,2:j_end) = ca_ex_bc_r(:,2:j_end).*ex_bc_r(:,2:j_end)-...
    cb_ex_bc_r(:,2:j_end).*(hzx_bc_r(:,1:j_end-1)+hzy_bc_r(:,1:j_end-1)-...
    hzx_bc_r(:,2:j_end)-hzy_bc_r(:,2:j_end));

  ex_bc_r(:,1) = ca_ex_bc_r(:,1).*ex_bc_r(:,1)-...
    cb_ex_bc_r(:,1).*(hzx_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)+...
    hzy_bc_f(1+PML_LR+i_end:i_main_grid_plus_PML,PML_FB)-hzx_bc_r(:,1)-hzy_bc_r(:,1));

  ex_bc_r(:,jb) = ca_ex_bc_r(:,jb).*ex_bc_r(:,jb)-...
    cb_ex_bc_r(:,jb).*(hzx_bc_r(:,j_end)+hzy_bc_r(:,j_end)-...
    hzx_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1)-...
    hzy_bc_b(1+PML_LR+i_end:i_main_grid_plus_PML,1));

%***********************************************************************
%    Update EY in PML regions
%***********************************************************************

%    FRONT

  ey_bc_f(2:i_main_grid_plus_PML,:) = ca_ey_bc_f(2:i_main_grid_plus_PML,:).*...
    ey_bc_f(2:i_main_grid_plus_PML,:)-...
    cb_ey_bc_f(2:i_main_grid_plus_PML,:).*(hzx_bc_f(2:i_main_grid_plus_PML,:)+...
    hzy_bc_f(2:i_main_grid_plus_PML,:)-...
    hzx_bc_f(1:i_main_grid_plus_PML-1,:)-hzy_bc_f(1:i_main_grid_plus_PML-1,:));

%   ey_bc_f(wall_at_y_axis+PML_FB,:) = 0;%added

%    BACK

  ey_bc_b(2:i_main_grid_plus_PML,:) = ca_ey_bc_b(2:i_main_grid_plus_PML,:).*...
    ey_bc_b(2:i_main_grid_plus_PML,:)-...
    cb_ey_bc_b(2:i_main_grid_plus_PML,:).*(hzx_bc_b(2:i_main_grid_plus_PML,:)+...
    hzy_bc_b(2:i_main_grid_plus_PML,:)-...
    hzx_bc_b(1:i_main_grid_plus_PML-1,:)-hzy_bc_b(1:i_main_grid_plus_PML-1,:));
```

```
%   ey_bc_b(wall_at_y_axis+PML_FB,:) = 0;%added

%    LEFT

   ey_bc_l(2:PML_LR,:) = ca_ey_bc_l(2:PML_LR,:).*ey_bc_l(2:PML_LR,:)-...
     cb_ey_bc_l(2:PML_LR,:).*(hzx_bc_l(2:PML_LR,:)+hzy_bc_l(2:PML_LR,:)-...
     hzx_bc_l(1:PML_LR-1,:)-hzy_bc_l(1:PML_LR-1,:));

   ey(1,:) = ca_ey(1,:).*ey(1,:)-...
     cb_ey(1,:).*(hz(1,:)-hzx_bc_l(PML_LR,:)-hzy_bc_l(PML_LR,:));

   ey_bc_l(2:PML_LR,:) = 0;%%%added

%    RIGHT

   ey_bc_r(2:PML_LR,:) = ca_ey_bc_r(2:PML_LR,:).*ey_bc_r(2:PML_LR,:)-...
     cb_ey_bc_r(2:PML_LR,:).*(hzx_bc_r(2:PML_LR,:)+hzy_bc_r(2:PML_LR,:)-...
     hzx_bc_r(1:PML_LR-1,:)-hzy_bc_r(1:PML_LR-1,:));

   ey(ib,:) = ca_ey(ib,:).*ey(ib,:)-...
     cb_ey(ib,:).*(hzx_bc_r(1,:)+hzy_bc_r(1,:)- hz(i_end,:));

%***********************************************************************
%    Update magnetic fields (HZ) in main grid
%***********************************************************************

   hz(1:i_end,1:j_end) = da_hz(1:i_end,1:j_end).*hz(1:i_end,1:j_end)+...
     db_hz(1:i_end,1:j_end).*(ex(1:i_end,2:jb)-ex(1:i_end,1:j_end)+...
     ey(1:i_end,1:j_end)-ey(2:ib,1:j_end));

   hz([rod_start:rod_end],wall_at_y_axis) = 0;%added

   hz(i_source_location,j_source_location) = source(n);%changed

%***********************************************************************
%    Update HZX in PML regions
%***********************************************************************

%    FRONT

   hzx_bc_f(1:i_main_grid_plus_PML,:) = da_hzx_bc_f(1:i_main_grid_plus_PML,:).*...
     hzx_bc_f(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_f(1:i_main_grid_plus_PML,:).*(ey_bc_f(2:ib_fbc,:)-...
     ey_bc_f(1:i_main_grid_plus_PML,:));

%    BACK

   hzx_bc_b(1:i_main_grid_plus_PML,:) = da_hzx_bc_b(1:i_main_grid_plus_PML,:).*...
     hzx_bc_b(1:i_main_grid_plus_PML,:)-...
     db_hzx_bc_b(1:i_main_grid_plus_PML,:).*(ey_bc_b(2:ib_fbc,:)-...
     ey_bc_b(1:i_main_grid_plus_PML,:));

%    LEFT

   hzx_bc_l(1:PML_LR-1,:) = da_hzx_bc_l(1:PML_LR-1,:).*hzx_bc_l(1:PML_LR-1,:)-...
     db_hzx_bc_l(1:PML_LR-1,:).*(ey_bc_l(2:PML_LR,:)-ey_bc_l(1:PML_LR-1,:));

   hzx_bc_l(PML_LR,:) = da_hzx_bc_l(PML_LR,:).*hzx_bc_l(PML_LR,:)-...
     db_hzx_bc_l(PML_LR,:).*(ey(1,:)-ey_bc_l(PML_LR,:));

%    RIGHT

   hzx_bc_r(2:PML_LR,:) = da_hzx_bc_r(2:PML_LR,:).*hzx_bc_r(2:PML_LR,:)-...
     db_hzx_bc_r(2:PML_LR,:).*(ey_bc_r(3:ib_bc,:)-ey_bc_r(2:PML_LR,:));

   hzx_bc_r(1,:) = da_hzx_bc_r(1,:).*hzx_bc_r(1,:)-...
     db_hzx_bc_r(1,:).*(ey_bc_r(2,:)-ey(ib,:));

%***********************************************************************
%    Update HZY in PML regions
%***********************************************************************

%    FRONT
```

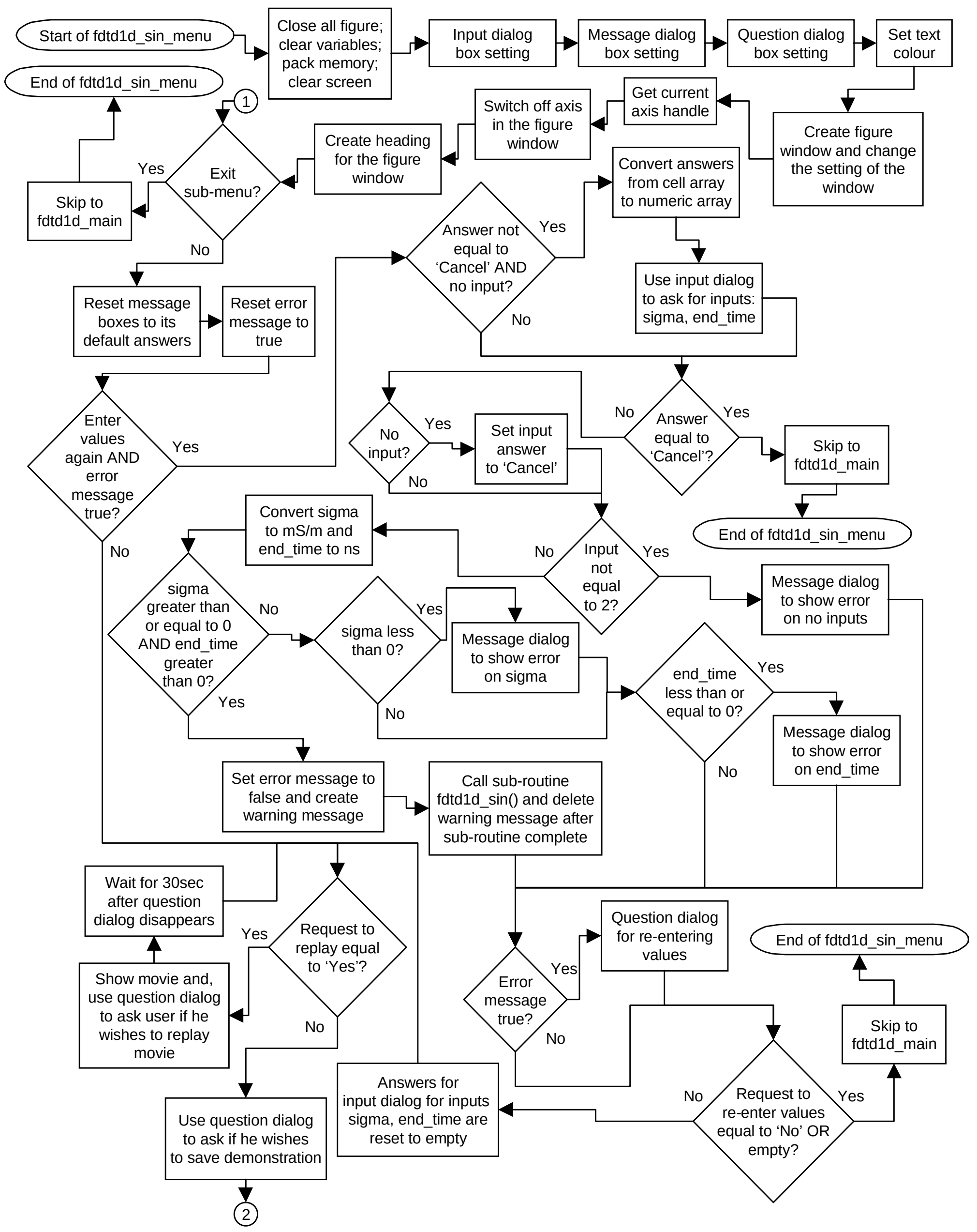
```
  hzy_bc_f(:,1:PML_FB-1) = da_hzy_bc_f(:,1:PML_FB-1).*hzy_bc_f(:,1:PML_FB-1)-...
    db_hzy_bc_f(:,1:PML_FB-1).*(ex_bc_f(:,1:PML_FB-1)-ex_bc_f(:,2:PML_FB));

  hzy_bc_f(1:PML_LR,PML_FB) = da_hzy_bc_f(1:PML_LR,PML_FB).*hzy_bc_f(1:PML_LR,PML_FB)-...
    db_hzy_bc_f(1:PML_LR,PML_FB).*(ex_bc_f(1:PML_LR,PML_FB)-ex_bc_l(1:PML_LR,1));

  hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB) = da_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-...
    db_hzy_bc_f(PML_LR+1:PML_LR+i_end,PML_FB).*...
    (ex_bc_f(PML_LR+1:PML_LR+i_end,PML_FB)-ex(1:i_end,1));

  hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB) =...
    da_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-...
    db_hzy_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB).*...
    (ex_bc_f(PML_LR+i_end+1:i_main_grid_plus_PML,PML_FB)-ex_bc_r(1:PML_LR,1));

%    BACK

  hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB) =...
    da_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    db_hzy_bc_b(1:i_main_grid_plus_PML,2:PML_FB).*...
    (ex_bc_b(1:i_main_grid_plus_PML,2:PML_FB)-...
    ex_bc_b(1:i_main_grid_plus_PML,3:jb_bc));

  hzy_bc_b(1:PML_LR,1) = da_hzy_bc_b(1:PML_LR,1).*hzy_bc_b(1:PML_LR,1)-...
    db_hzy_bc_b(1:PML_LR,1).*(ex_bc_l(1:PML_LR,jb)-ex_bc_b(1:PML_LR,2));

  hzy_bc_b(PML_LR+1:PML_LR+i_end,1) = da_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    hzy_bc_b(PML_LR+1:PML_LR+i_end,1)-...
    db_hzy_bc_b(PML_LR+1:PML_LR+i_end,1).*...
    (ex(1:i_end,jb)-ex_bc_b(PML_LR+1:PML_LR+i_end,2));

  hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1) =...
    da_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1)-...
    db_hzy_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,1).*...
    (ex_bc_r(1:PML_LR,jb)-ex_bc_b(PML_LR+i_end+1:i_main_grid_plus_PML,2));

%    LEFT

  hzy_bc_l(:,1:j_end) = da_hzy_bc_l(:,1:j_end).*hzy_bc_l(:,1:j_end)-...
    db_hzy_bc_l(:,1:j_end).*(ex_bc_l(:,1:j_end)-ex_bc_l(:,2:jb));

%    RIGHT

  hzy_bc_r(:,1:j_end) = da_hzy_bc_r(:,1:j_end).*hzy_bc_r(:,1:j_end)-...
    db_hzy_bc_r(:,1:j_end).*(ex_bc_r(:,1:j_end)-ex_bc_r(:,2:jb));

%***********************************************************************
%    Visualize fields
%***********************************************************************
  number_of_skip_frame = 4;

  if mod(n,number_of_skip_frame) == 0,

    timestep = int2str(n);
    colormap hot %(added)

    subplot(3,1,1),pcolor(ex')
    shading flat
    caxis([-80.0 80.0])
    axis([1 i_end 1 jb])
    colorbar
    axis image
    axis off
    title(['E_x at time step = ',timestep])

    subplot(3,1,2),pcolor(ey')
    shading flat
    caxis([-80.0 80.0])
    axis([1 ib 1 j_end])
    colorbar
    axis image
    axis off
    title(['E_y at time step = ',timestep])

    subplot(3,1,3),pcolor(hz')
    shading flat
    caxis([-0.2 0.2])
    axis([1 i_end 1 j_end])
    colorbar
    axis image
    axis off
    title(['H_z at time step = ',timestep])

    rect = get(h1,'Position');
    rect(1:2) = [0 0];%change the position of the frame
                %with respect to the current figure window

    M(:,n/number_of_skip_frame) = getframe(h1,rect);

  end

%**********************************************************************
%     END TIME-STEPPING LOOP
%**********************************************************************

end

movie(h1,M,0,10,rect)
```

Appendix E

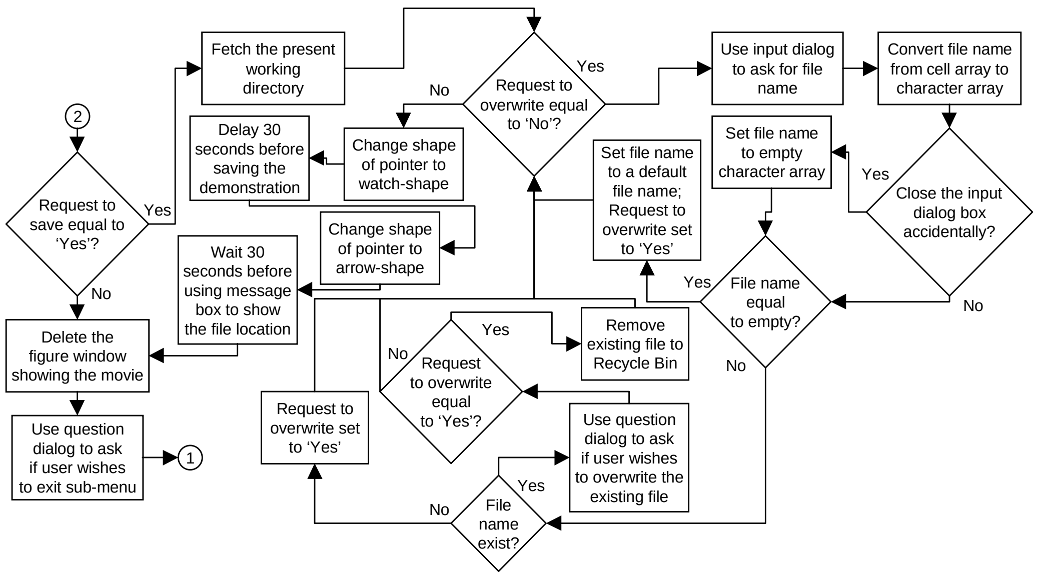
Flowchart of GUI


Start of fdtd1d_in
Close all figure; clear variables; pack memory; clear screen
Set text colour
Set vertical spacing and horizontal margin
Create figure window and change the setting of the window
Switch off axis in the figure window
Create heading for the figure window
Create explanation text
Create push-button for Back
Wait for response
'Close' button pressed or 'Back' push-button pressed?
No
Yes
Close current figure window and go to return to main
End of fdtd1d_in

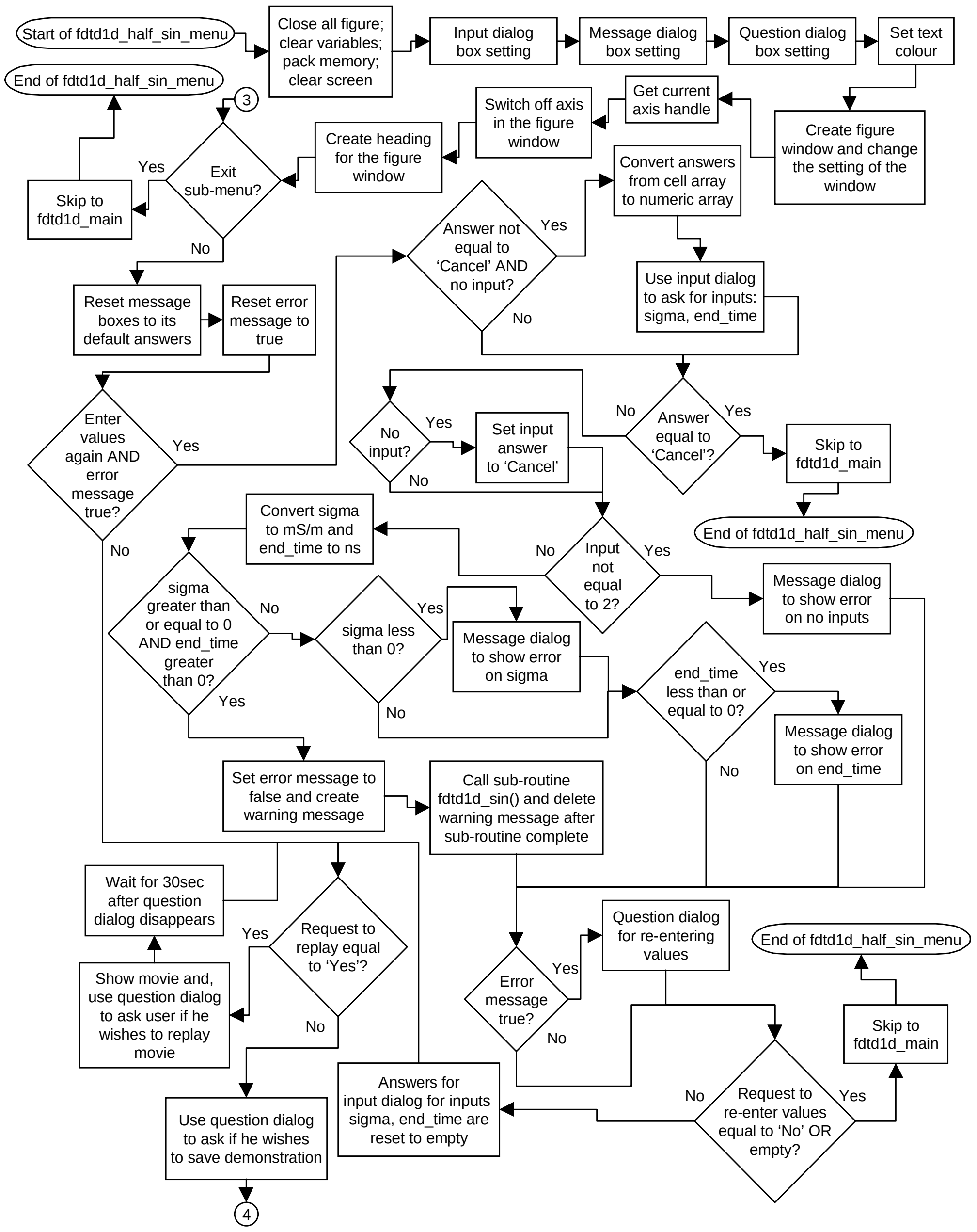
fdtd1d_in.m


Start of fdtd1d_sin_menu
Close all figure; clear variables; pack memory; clear screen
Input dialog box setting
Message dialog box setting
Question dialog box setting
Set text colour
Create figure window and change the setting of the window
Get current axis handle
Switch off axis in the figure window
Create heading for the figure window
Exit sub-menu?
Yes
Skip to fdtd1d_main
End of fdtd1d_sin_menu
No
Reset message boxes to its default answers
Reset error message to true
Enter values again AND error message true?
Yes
No
Answer not equal to 'Cancel' AND no input?
Yes
No
Convert answers from cell array to numeric array
Use input dialog to ask for inputs: sigma, end_time
Answer equal to 'Cancel'?
Yes
No
Skip to fdtd1d_main
End of fdtd1d_sin_menu
No input?
Yes
No
Set input answer to 'Cancel'
Input not equal to 2?
Yes
No
Message dialog to show error on no inputs
Convert sigma to mS/m and end_time to ns
sigma greater than or equal to 0 AND end_time greater than 0?
Yes
No
sigma less than 0?
Yes
No
Message dialog to show error on sigma
end_time less than or equal to 0?
Yes
No
Message dialog to show error on end_time
Set error message to false and create warning message
Call sub-routine fdtd1d_sin() and delete warning message after sub-routine complete
Wait for 30sec after question dialog disappears
Request to replay equal to 'Yes'?
Yes
No
Show movie and, use question dialog to ask user if he wishes to replay movie
Use question dialog to ask if he wishes to save demonstration
2
1
Answers for input dialog for inputs sigma, end_time are reset to empty
Error message true?
Yes
No
Question dialog for re-entering values
Request to re-enter values equal to 'No' OR empty?
Yes
No
Skip to fdtd1d_main
End of fdtd1d_sin_menu

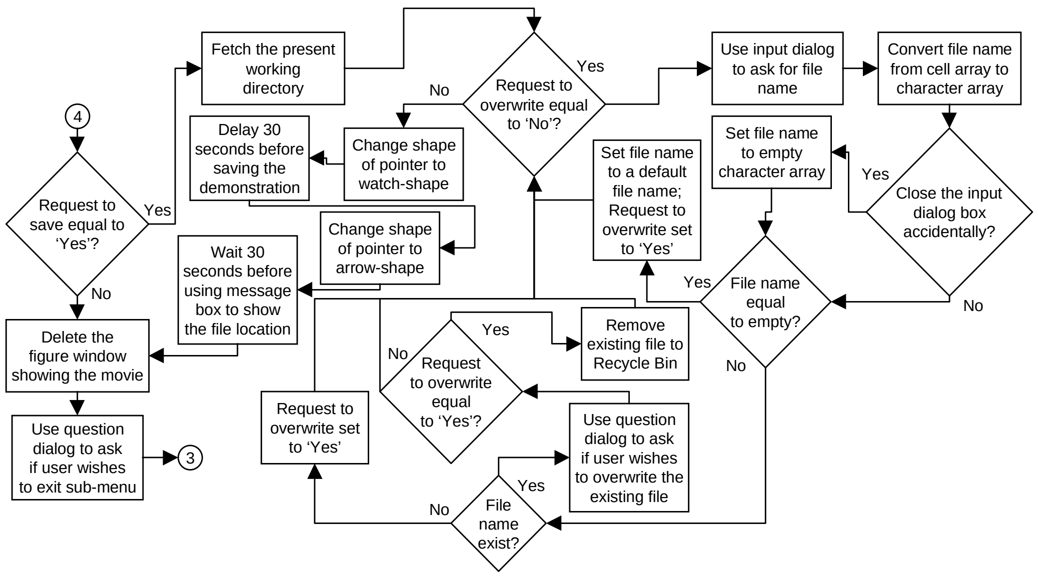
fdtd1d_sin_menu.m (Part 1)


2
Request to save equal to 'Yes'?
Yes
No
Fetch the present working directory
Request to overwrite equal to 'No'?
Yes
No
Use input dialog to ask for file name
Convert file name from cell array to character array
Close the input dialog box accidentally?
Yes
No
Set file name to empty character array
File name equal to empty?
Yes
No
Set file name to a default file name; Request to overwrite set to 'Yes'
Change shape of pointer to watch-shape
Delay 30 seconds before saving the demonstration
Change shape of pointer to arrow-shape
Wait 30 seconds before using message box to show the file location
Delete the figure window showing the movie
Use question dialog to ask if user wishes to exit sub-menu
1
Remove existing file to Recycle Bin
Request to overwrite equal to 'Yes'?
Yes
No
Request to overwrite set to 'Yes'
Use question dialog to ask if user wishes to overwrite the existing file
File name exist?
Yes
No

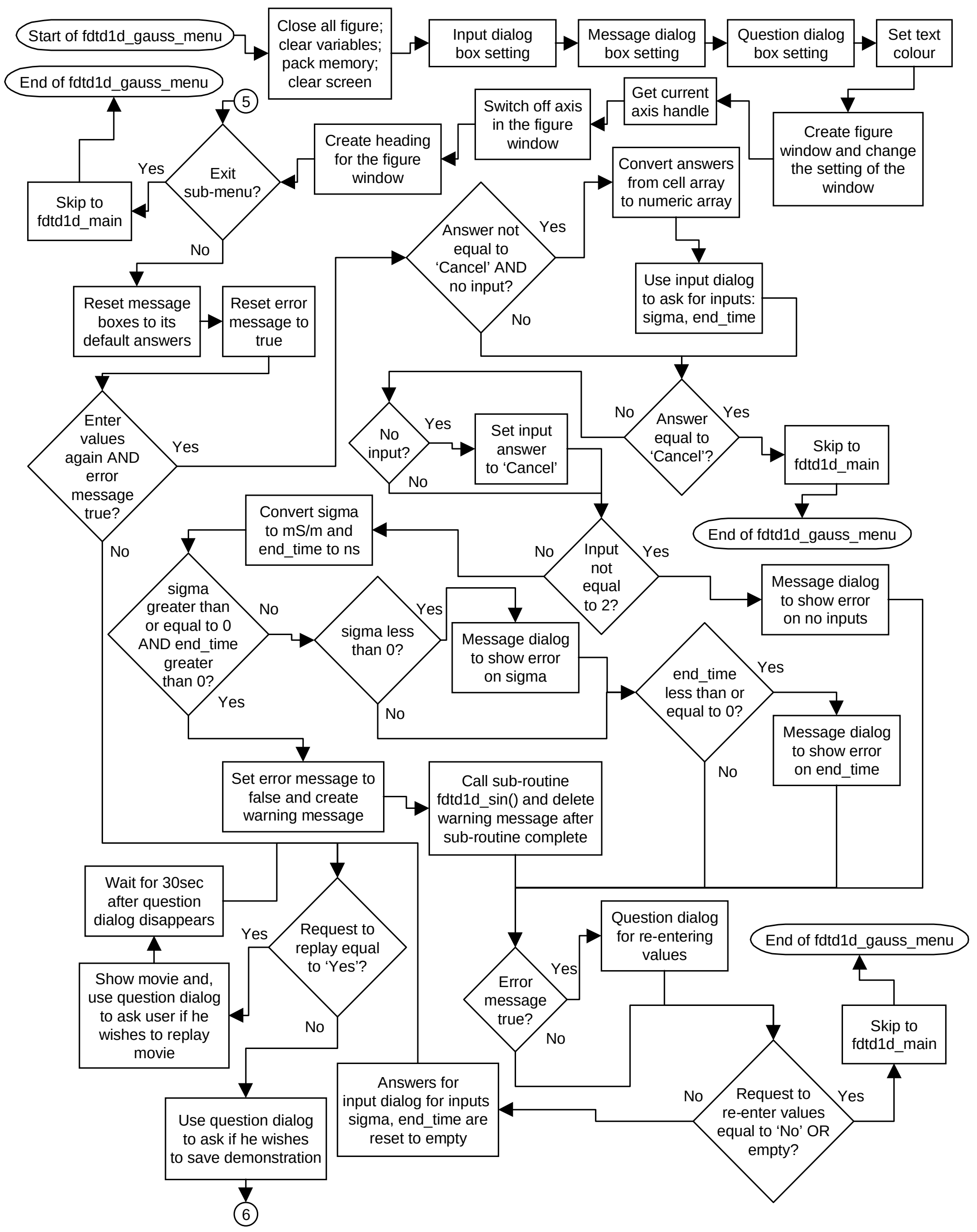
fdtd1d_sin_menu.m (Part 2)


Start of fdtd1d_half_sin_menu
Close all figure; clear variables; pack memory; clear screen
Input dialog box setting
Message dialog box setting
Question dialog box setting
Set text colour
End of fdtd1d_half_sin_menu
3
Get current axis handle
Switch off axis in the figure window
Create heading for the figure window
Create figure window and change the setting of the window
Exit sub-menu?
Yes
Skip to fdtd1d_main
No
Convert answers from cell array to numeric array
Answer not equal to 'Cancel' AND no input?
Yes
No
Use input dialog to ask for inputs: sigma, end_time
Reset message boxes to its default answers
Reset error message to true
Enter values again AND error message true?
Yes
No
No input?
Yes
No
Set input answer to 'Cancel'
Answer equal to 'Cancel'?
Yes
No
Skip to fdtd1d_main
End of fdtd1d_half_sin_menu
Convert sigma to mS/m and end_time to ns
Input not equal to 2?
Yes
No
Message dialog to show error on no inputs
sigma greater than or equal to 0 AND end_time greater than 0?
No
Yes
sigma less than 0?
Yes
No
Message dialog to show error on sigma
end_time less than or equal to 0?
Yes
No
Message dialog to show error on end_time
Set error message to false and create warning message
Call sub-routine fdtd1d_sin() and delete warning message after sub-routine complete
Wait for 30sec after question dialog disappears
Request to replay equal to 'Yes'?
Yes
No
Show movie and, use question dialog to ask user if he wishes to replay movie
Question dialog for re-entering values
Error message true?
Yes
No
End of fdtd1d_half_sin_menu
Skip to fdtd1d_main
Request to re-enter values equal to 'No' OR empty?
Yes
No
Answers for input dialog for inputs sigma, end_time are reset to empty
Use question dialog to ask if he wishes to save demonstration
4

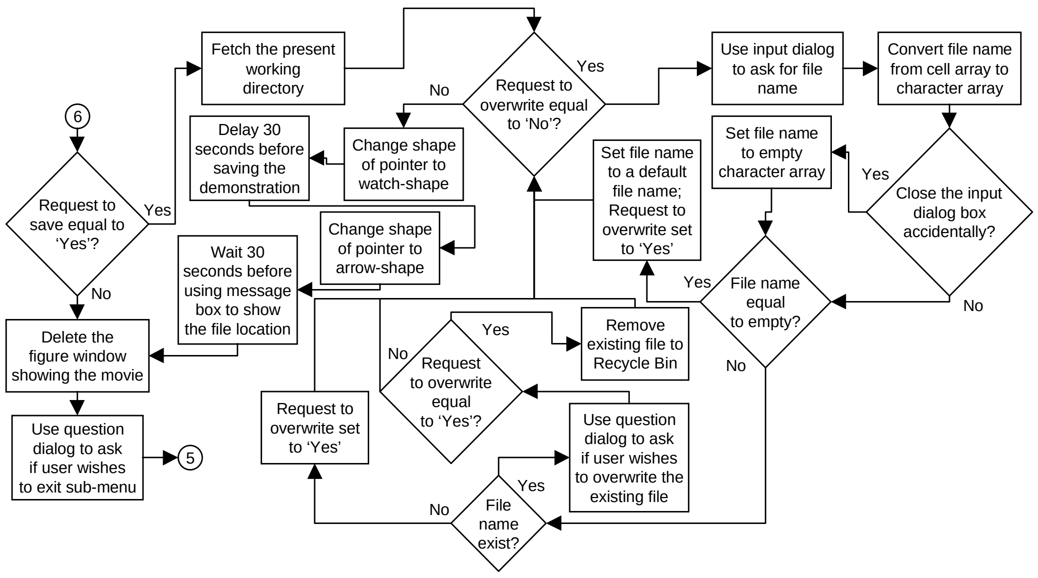
fdtd1d_half_sin_menu.m (Part 1)


4
Request to save equal to 'Yes'?
Yes
No
Fetch the present working directory
Request to overwrite equal to 'No'?
Yes
No
Use input dialog to ask for file name
Convert file name from cell array to character array
Close the input dialog box accidentally?
Yes
No
Set file name to empty character array
File name equal to empty?
Yes
No
Set file name to a default file name; Request to overwrite set to 'Yes'
File name exist?
Yes
No
Use question dialog to ask if user wishes to overwrite the existing file
Request to overwrite equal to 'Yes'?
Yes
No
Remove existing file to Recycle Bin
Request to overwrite set to 'Yes'
Change shape of pointer to watch-shape
Delay 30 seconds before saving the demonstration
Change shape of pointer to arrow-shape
Wait 30 seconds before using message box to show the file location
Delete the figure window showing the movie
Use question dialog to ask if user wishes to exit sub-menu
3

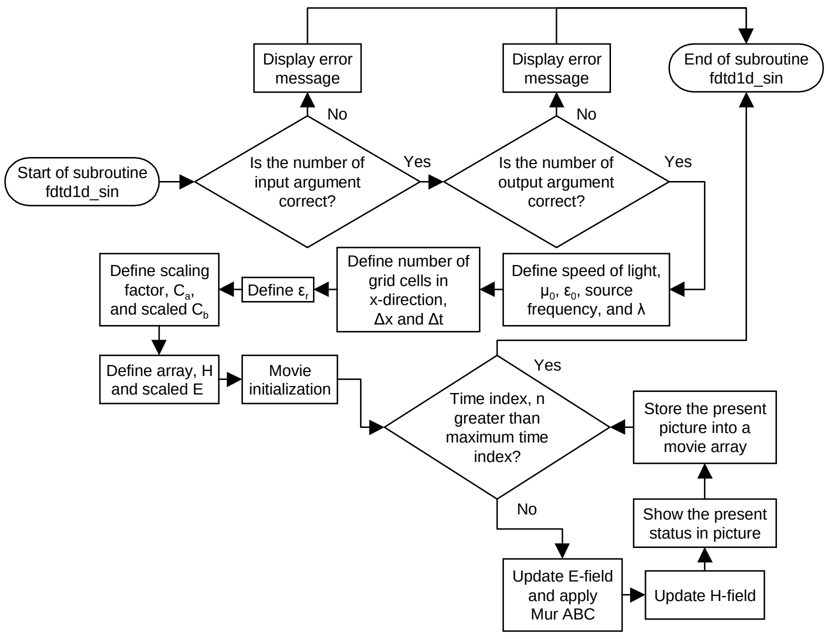
fdtd1d_half_sin_menu.m (Part 2)


Start of fdtd1d_gauss_menu
Close all figure; clear variables; pack memory; clear screen
Input dialog box setting
Message dialog box setting
Question dialog box setting
Set text colour
Create figure window and change the setting of the window
Get current axis handle
Switch off axis in the figure window
Create heading for the figure window
End of fdtd1d_gauss_menu
5
Exit sub-menu?
Yes
Skip to fdtd1d_main
No
Reset message boxes to its default answers
Reset error message to true
Enter values again AND error message true?
Yes
No
Answer not equal to 'Cancel' AND no input?
Yes
No
Convert answers from cell array to numeric array
Use input dialog to ask for inputs: sigma, end_time
Answer equal to 'Cancel'?
Yes
No
Skip to fdtd1d_main
End of fdtd1d_gauss_menu
No input?
Yes
No
Set input answer to 'Cancel'
Input not equal to 2?
Yes
No
Message dialog to show error on no inputs
Convert sigma to mS/m and end_time to ns
sigma greater than or equal to 0 AND end_time greater than 0?
Yes
No
sigma less than 0?
Yes
No
Message dialog to show error on sigma
end_time less than or equal to 0?
Yes
No
Message dialog to show error on end_time
Set error message to false and create warning message
Call sub-routine fdtd1d_sin() and delete warning message after sub-routine complete
Wait for 30sec after question dialog disappears
Request to replay equal to 'Yes'?
Yes
No
Show movie and, use question dialog to ask user if he wishes to replay movie
Use question dialog to ask if he wishes to save demonstration
6
Answers for input dialog for inputs sigma, end_time are reset to empty
Error message true?
Yes
No
Question dialog for re-entering values
Request to re-enter values equal to 'No' OR empty?
Yes
No
Skip to fdtd1d_main
End of fdtd1d_gauss_menu

fdtd1d_gauss_menu.m (Part 1)

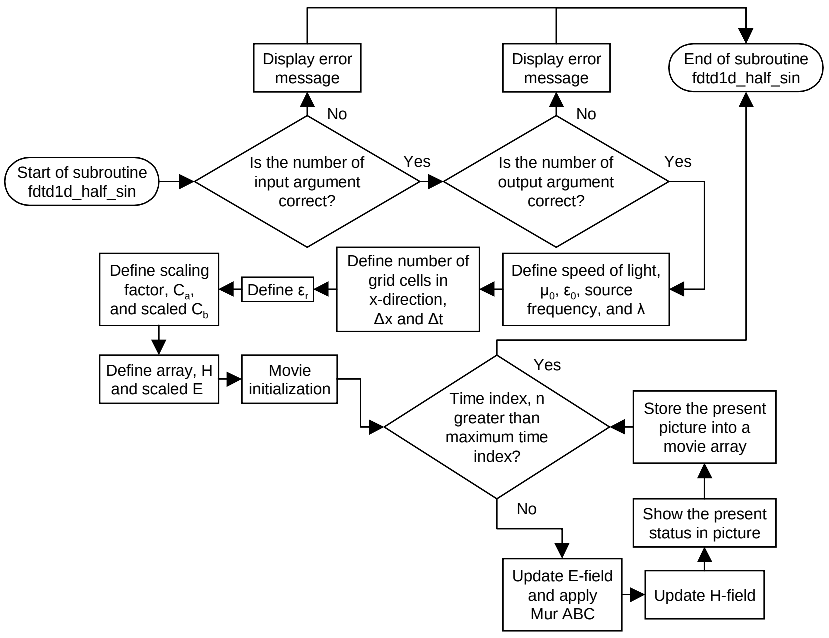

6
Request to save equal to 'Yes'?
Yes
No
Fetch the present working directory
Request to overwrite equal to 'No'?
Yes
No
Use input dialog to ask for file name
Convert file name from cell array to character array
Close the input dialog box accidentally?
Yes
No
Set file name to empty character array
File name equal to empty?
Yes
No
Set file name to a default file name; Request to overwrite set to 'Yes'
Delay 30 seconds before saving the demonstration
Change shape of pointer to watch-shape
Change shape of pointer to arrow-shape
Wait 30 seconds before using message box to show the file location
Delete the figure window showing the movie
Use question dialog to ask if user wishes to exit sub-menu
5
Remove existing file to Recycle Bin
Request to overwrite equal to 'Yes'?
Yes
No
Request to overwrite set to 'Yes'
Use question dialog to ask if user wishes to overwrite the existing file
File name exist?
Yes
No

fdtd1d_gauss_menu.m (Part 2)

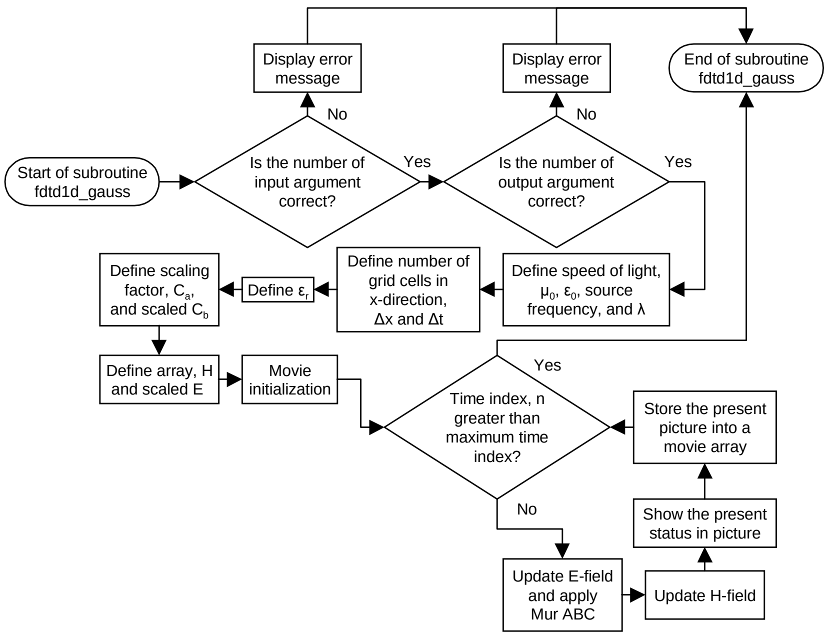


End of subroutine fdtd1d_sin
Display error message
Display error message
No
No
Yes
Yes
Start of subroutine fdtd1d_sin
Is the number of input argument correct?
Is the number of output argument correct?
Define scaling factor, C_a, and scaled C_b
Define ε_r
Define number of grid cells in x-direction, Δx and Δt
Define speed of light, μ_0, ε_0, source frequency, and λ
Define array, H and scaled E
Movie initialization
Yes
Time index, n greater than maximum time index?
Store the present picture into a movie array
No
Show the present status in picture
Update E-field and apply Mur ABC
Update H-field

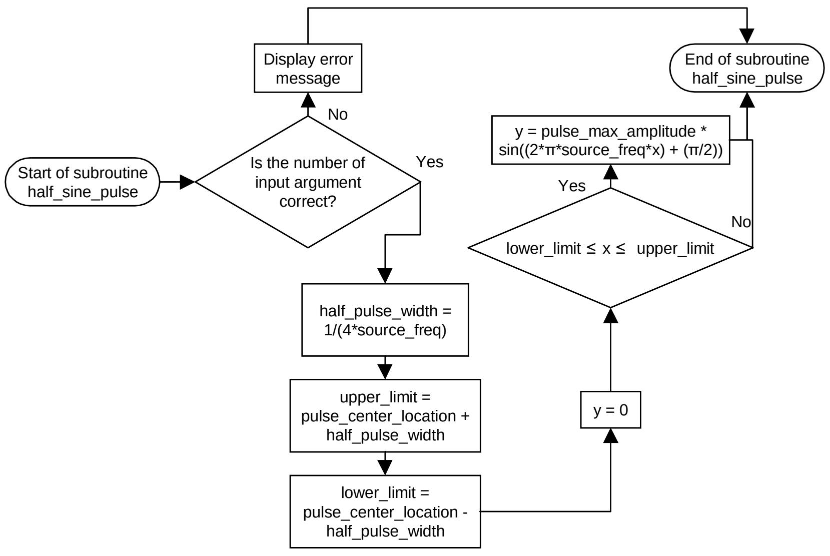
fdtd1d_sin.m


Start of subroutine fdtd1d_half_sin
Is the number of input argument correct?
No
Display error message
Yes
Is the number of output argument correct?
No
Display error message
Yes
End of subroutine fdtd1d_half_sin
Define speed of light, μ_0, ε_0, source frequency, and λ
Define number of grid cells in x-direction, Δx and Δt
Define ε_r
Define scaling factor, C_a, and scaled C_b
Define array, H and scaled E
Movie initialization
Time index, n greater than maximum time index?
Yes
No
Update E-field and apply Mur ABC
Update H-field
Show the present status in picture
Store the present picture into a movie array

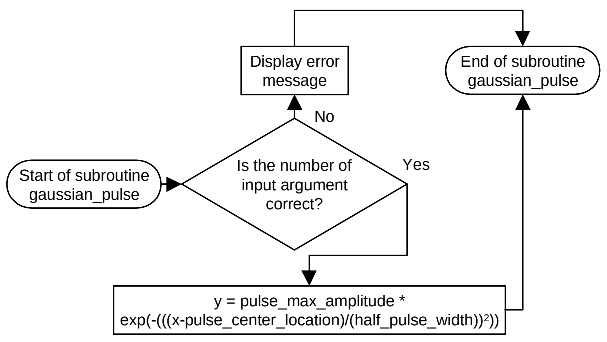
fdtd1d_half_sin.m


Display error message
Display error message
End of subroutine fdtd1d_gauss
No
No
Start of subroutine fdtd1d_gauss
Is the number of input argument correct?
Yes
Is the number of output argument correct?
Yes
Define scaling factor, C_a, and scaled C_b
Define ε_r
Define number of grid cells in x-direction, Δx and Δt
Define speed of light, μ_0, ε_0, source frequency, and λ
Define array, H and scaled E
Movie initialization
Yes
Time index, n greater than maximum time index?
Store the present picture into a movie array
No
Show the present status in picture
Update E-field and apply Mur ABC
Update H-field

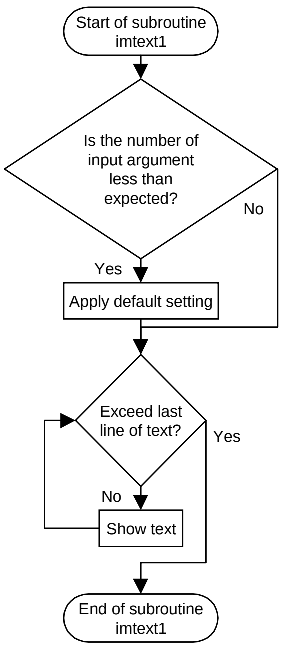
fdtd1d_gauss.m


Start of subroutine half_sine_pulse
Is the number of input argument correct?
No
Yes
Display error message
End of subroutine half_sine_pulse
half_pulse_width = 1/(4*source_freq)
upper_limit = pulse_center_location + half_pulse_width
lower_limit = pulse_center_location - half_pulse_width
y = 0
lower_limit ≤ x ≤ upper_limit
Yes
No
y = pulse_max_amplitude * sin((2*π*source_freq*x) + (π/2))

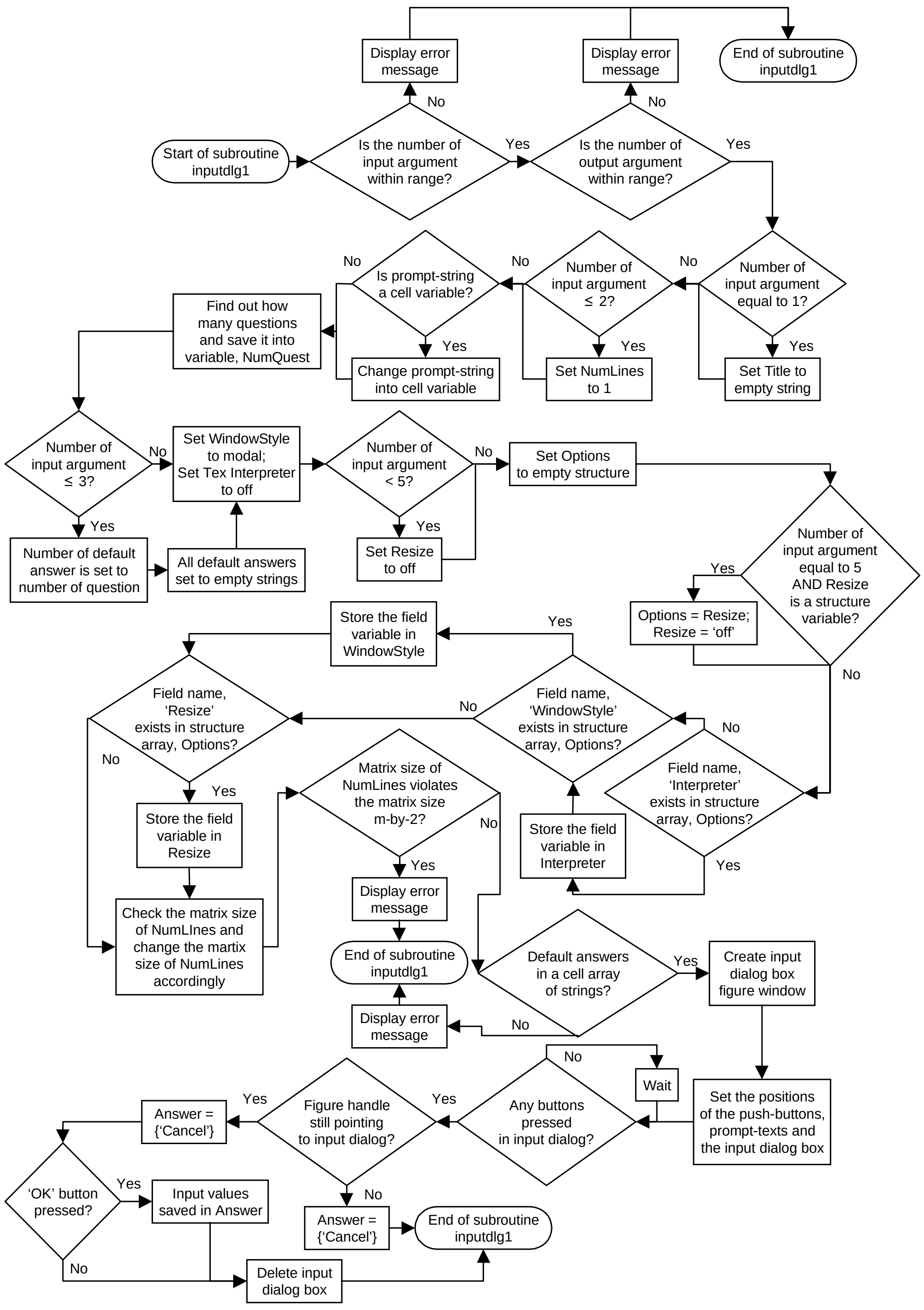
half_sine_pulse.m


Display error message
End of subroutine gaussian_pulse
No
Start of subroutine gaussian_pulse
Is the number of input argument correct?
Yes
y = pulse_max_amplitude * exp(-(((x-pulse_center_location)/(half_pulse_width))²))

gaussian_pulse.m


Start of subroutine imtext1
Is the number of input argument less than expected?
No
Yes
Apply default setting
Exceed last line of text?
Yes
No
Show text
End of subroutine imtext1

imtext1.m


Start of subroutine inputdlg1
Is the number of input argument within range?
No
Display error message
Yes
Is the number of output argument within range?
No
Display error message
Yes
End of subroutine inputdlg1
Number of input argument equal to 1?
Yes
Set Title to empty string
No
Number of input argument ≤ 2?
Yes
Set NumLines to 1
No
Is prompt-string a cell variable?
Yes
Change prompt-string into cell variable
No
Find out how many questions and save it into variable, NumQuest
Number of input argument ≤ 3?
Yes
Number of default answer is set to number of question
No
All default answers set to empty strings
Set WindowStyle to modal; Set Tex Interpreter to off
Number of input argument < 5?
Yes
Set Resize to off
No
Set Options to empty structure
Number of input argument equal to 5 AND Resize is a structure variable?
Yes
Options = Resize; Resize = 'off'
No
Field name, 'Interpreter' exists in structure array, Options?
Yes
Store the field variable in Interpreter
No
Field name, 'WindowStyle' exists in structure array, Options?
Yes
Store the field variable in WindowStyle
No
Field name, 'Resize' exists in structure array, Options?
Yes
Store the field variable in Resize
No
Check the matrix size of NumLInes and change the martix size of NumLines accordingly
Matrix size of NumLines violates the matrix size m-by-2?
Yes
Display error message
End of subroutine inputdlg1
No
Default answers in a cell array of strings?
Yes
Create input dialog box figure window
No
Display error message
Set the positions of the push-buttons, prompt-texts and the input dialog box
Wait
Any buttons pressed in input dialog?
No
Yes
Figure handle still pointing to input dialog?
No
Answer = {'Cancel'}
End of subroutine inputdlg1
Yes
Answer = {'Cancel'}
'OK' button pressed?
Yes
Input values saved in Answer
No
Delete input dialog box

inputdlg1.m