The FDTD Method: Computation and Analysis

May 13, 2005

1 Introduction

In Dennis M. Sullivan's *Electromagnetic Simulation: Using the FDTD Method* [2], the object is to quickly familiarize oneself with the computational aspects of simulation of electromagnetic waves using the Finite-Difference Time-Domain (FDTD) scheme[1]. By the author's own account the book does not

> ...explain the theory of FDTD simulation in great detail. It is not a survey of all possible approaches to the FDTD method nor is it a "cookbook" of applications.

This paper intends to fill in some of the gaps left by the author in Chapter 1, especially for those interested in a more mathematically oriented approach to FDTD.

We will be begin with a few comments on why FDTD is a widely used scheme in approximating electromagnetic waves. Then we will provide some background information about the scheme: both the people behind it and its applications.

FDTD is an important and widely used scheme for many reasons: the time required to learn the FDTD scheme is minimal; it does not use matrices in its implementation; and its simulations can be used to study such phenomena as wave interaction and coupling.

[1]Finite difference refers the method used to obtain discrete approximations of derivatives. Time-domain refers to the fact that the simulations are done in time, rather than in the frequency domain.

James T. Maxwell studied the interaction of the magnetic and electric components of electricity. Around 1870 he developed the famous equations that describe electromagnetic behavior. In 1966 Kane S. Yee found an algorithm to approximate these equations. His paper "Numerical solutions of initial boundary value problems involving Maxwell's equations in isotropic media" introduced the FDTD scheme and the Yee cell. Due to the deficient computational capacity of early computers, his paper was overlooked. However, by the mid-seventies the computational capabilities of computers had been developed enough to make the implementation of Yee's algorithm feasible on a broad range of important problems. In 1975, Taflove and Browdin wrote a paper demonstrating the value of the FDTD method in electrodynamics and gave the first absorbing boundary conditions [3]. This was the beginning of computational electrodynamics as we know it today.

The historic motivation for the development of computational electrodynamic is as follows. During World War II, electromagnetism was studied because of its connection to radar. The atomic era brought the threat of high-level electromagnetic pulses capable of burning out electric devices for miles around, leaving a country's infrastructure devastated. Numerous developments in electromagnetic wave theory were made during the cold war in the name of national defense. Currently, the field of electromagnetism is becoming important in fields such as high-speed computing and biomedicine, as well as in national defense. Now that we have some background about FDTD let us begin our exploration and implementation of the scheme itself.[3]

2 Electromagnetic Simulation in Free Space

2.1 Mathematical Development

In Sullivan's book, the simulation of electromagnetic waves generated by a pulse is discussed. The equations for these waves are given by the time-dependent Maxwell's equations in free space[2]. Maxwell's curl equations are

[2]Free space means far removed from objects that could influence propagation of electromagnetic waves.

given by

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\epsilon_0}\nabla \times \mathbf{H}, \tag{1}$$

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0}\nabla \times \mathbf{E}. \tag{2}$$

Both **E** and **H** are vectors in three dimensions which represent the electric field and magnetic field respectively.

To gain insight into the algorithm, we start by reducing Maxwell's equations to one dimension. First, we write equations (1) and (2) in vector form. The vector differential operator, "del" or "nabla", is defined

$$\nabla = \mathbf{i}\frac{\partial}{\partial \mathbf{x}} + \mathbf{j}\frac{\partial}{\partial \mathbf{y}} + \mathbf{k}\frac{\partial}{\partial \mathbf{z}}.$$

Let $\mathbf{E} = (E_x, E_y, E_z)$. The cross product of ∇ and **E** is given by

$$\nabla \times \mathbf{E} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \\ E_x & E_y & E_z \end{vmatrix}.$$

Taking $\nabla \times \mathbf{H}$ gives a similar result.

Using the above expressions we rewrite (1) and (2) as

$$\begin{bmatrix} \dfrac{\partial E_x}{\partial t} \\ \dfrac{\partial E_y}{\partial t} \\ \dfrac{\partial E_z}{\partial t} \end{bmatrix} = \frac{1}{\varepsilon_0}\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \\ H_x & H_y & H_z \end{vmatrix} = \frac{1}{\varepsilon_0}\begin{bmatrix} \dfrac{\partial H_z}{\partial y} - \dfrac{\partial H_y}{\partial z} \\ \dfrac{\partial H_x}{\partial z} - \dfrac{\partial H_z}{\partial x} \\ \dfrac{\partial H_y}{\partial x} - \dfrac{\partial H_x}{\partial y} \end{bmatrix}, \tag{3}$$

$$\begin{bmatrix} \dfrac{\partial H_x}{\partial t} \\ \dfrac{\partial H_y}{\partial t} \\ \dfrac{\partial H_z}{\partial t} \end{bmatrix} = -\frac{1}{\mu_0}\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \\ E_x & E_y & E_z \end{vmatrix} = -\frac{1}{\mu_0}\begin{bmatrix} \dfrac{\partial E_z}{\partial y} - \dfrac{\partial E_y}{\partial z} \\ \dfrac{\partial E_x}{\partial z} - \dfrac{\partial E_z}{\partial x} \\ \dfrac{\partial E_y}{\partial x} - \dfrac{\partial E_x}{\partial y} \end{bmatrix}. \tag{4}$$

To continue our simplification to the one-dimensional problem we must make the assumption that the electric field oscillates only in the xz plane and that it travels in the z direction. Then $E_y = 0$. Since electromagnetic waves are transverse waves, they oscillate perpendicularly to the direction of propagation z. Thus $E_z = H_z = 0$, and hence $\mathbf{E} = (E_x, 0, 0)$.

We also know that the electric field and magnetic field travel perpendicular to one another. Thus their dot product must be zero, or

$$\mathbf{E} \cdot \mathbf{H} = (E_x, 0, 0) \cdot (H_x, H_y, 0) = E_x H_x = 0.$$

Knowing that $E_x \neq 0$ and using the zero factor property we see that $H_x = 0$. Thus, since $H_z = 0$, we see that $\mathbf{H} = (0, H_y, 0)$. Hence equations (3) and (4) become

$$\begin{bmatrix} \dfrac{\partial E_x}{\partial t} \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\varepsilon_0} \begin{bmatrix} \dfrac{\partial H_y}{\partial z} \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ \dfrac{\partial H_y}{\partial t} \\ 0 \end{bmatrix} = -\frac{1}{\mu_0} \begin{bmatrix} 0 \\ \dfrac{\partial E_x}{\partial z} \\ 0 \end{bmatrix}.$$

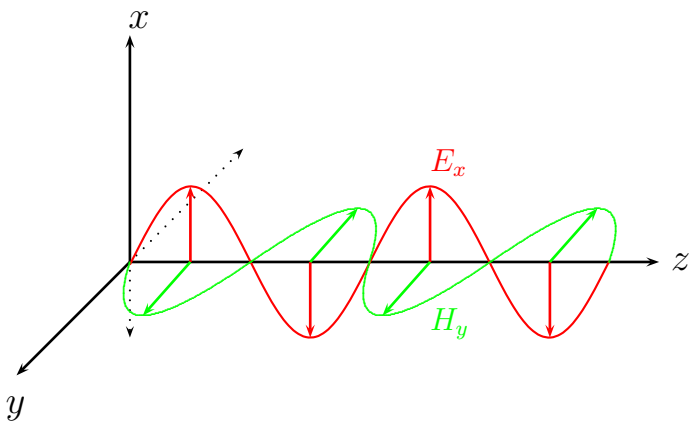
Or equivalently,

$$\frac{\partial E_x}{\partial t} = -\frac{1}{\varepsilon_0} \frac{\partial H_y}{\partial z}, \tag{5}$$

$$\frac{\partial H_y}{\partial t} = -\frac{1}{\mu_0} \frac{\partial E_x}{\partial z}. \tag{6}$$

Equations (5) and (6) correspond to transverse plane-polarized waves travelling in the z direction, which can be visualized as follows:

Propagation of Transverse Plane-Polarized EM Waves



2.2 The Algorithm

Now that we have the system we want to work with, we need some way to numerically approximate E_x and H_y. For our approximation we will use Δx for change along the z axis and Δt for change in time. Using the central difference approximation in both the spacial and time derivatives, we can express (5) and (6) as

$$\begin{aligned} \frac{(E_x)_k^{n+\frac{1}{2}} - (E_x)_k^{n-\frac{1}{2}}}{\Delta t} &= -\frac{1}{\varepsilon_0}\frac{(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n}{\Delta x} \\ \frac{(H_y)_k^{n+\frac{1}{2}} - (H_y)_k^{n-\frac{1}{2}}}{\Delta t} &= -\frac{1}{\mu_0}\frac{(E_x)_{k+\frac{1}{2}}^n - (E_x)_{k-\frac{1}{2}}^n}{\Delta x}. \end{aligned}$$

In this approximation, each time step is represented by n and each spacial step by k. Solving for E_x and H_y at time step $n+\frac{1}{2}$ and $n+1$ respectively yields

$$(E_x)_k^{n+\frac{1}{2}} = \nu[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] + (E_x)_k^{n-\frac{1}{2}}, \tag{7}$$

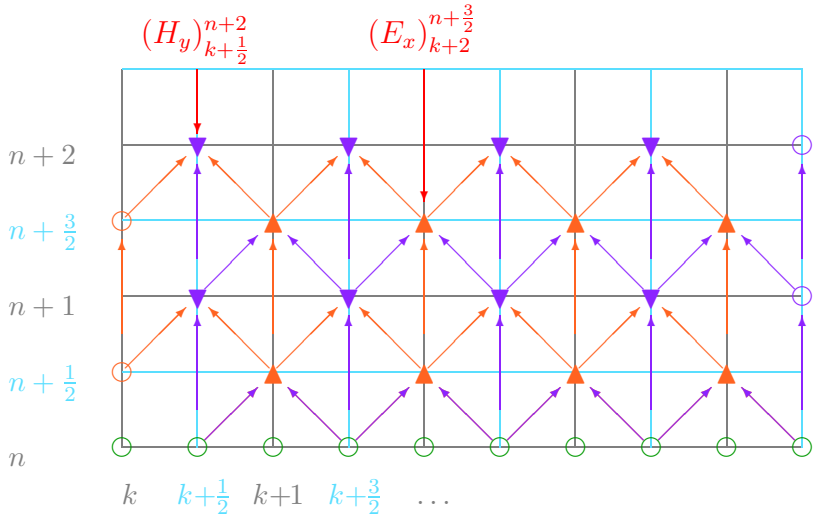
$$(H_y)_{k+\frac{1}{2}}^{n+1} = \upsilon[(E_x)_{k+1}^{n+\frac{1}{2}} - (E_x)_k^{n+\frac{1}{2}}] + (H_y)_{k+\frac{1}{2}}^n, \tag{8}$$

where $\nu = -\frac{1}{\varepsilon_0}\frac{\Delta t}{\Delta x}$ and $\upsilon = -\frac{1}{\mu_0}\frac{\Delta t}{\Delta x}$. There are some physical concerns we must address, but before we do that let us take a look at how this system of

equations is interlocked, or coupled.

The figure below represents the FDTD method. The orange triangles represent approximations of E_x while the purple triangles represent H_x approximations. Each row corresponds to a specific instant in time, at half time steps, whereas each column represents a single spacial grid point through time. The gray and blue grid line represent whole and half steps respectively, in both time and space. The initial values that must be given are the green circles, and the boundaries are the orange and purple circles.

The FDTD Approximation Grid



We see that for every triangle there are three arrows pointing towards it. Of these arrows, one comes from the previous time step. The other two come from a half-step down and a half-step to the right or left. From this we see that our algorithm will be interwoven in space and time just as the partial differential equations are coupled in time and space.

The FDTD method is also referred to as the leap-frog method. It is easy to imagine a frog starting on any triangle, for instance $(E_x)^{n+\frac{1}{2}}_{k+2}$, and spreading his legs to jump up to the triangle, $(E_x)^{n+\frac{3}{2}}_{k+2}$. The arrows moving first outward and then inward form a triangle and trace the path of his legs as he jumps. In the example starting at triangle $(E_x)^{n+\frac{1}{2}}_{k+2}$, the frog would spread his legs outward to $(H_y)^{n+1}_{k+\frac{3}{2}}$ on the left and $(H_y)^{n+1}_{k+\frac{5}{2}}$ on the right, finally landing one time step ahead on $(E_x)^{n+\frac{3}{2}}_{k+2}$.

As mentioned above, before we get to the algorithm, we must address some issues. The magnitudes of (5) and (6) are not the same, but we want them to be for our computations. The permittivity of free space and the permeability of free space, ε_0 and μ_0, respectively, cause the problem. Their values are

$$\begin{aligned} \varepsilon_0 &= 8.8542 \times 10^{-12}\ F/m, \\ \mu_0 &= 4\pi \times 10^{-7}\ H/m, \end{aligned}$$

where F/m is Farad's per meter and H/m is Henries per meter. A change of variables is necessary to fix this discrepancy in our equations. Taking

$$\tilde{\mathbf{E}} = \sqrt{\frac{\epsilon_0}{\mu_0}}\mathbf{E}$$

and substituting $\mathbf{E} = \sqrt{\frac{\mu_0}{\epsilon_0}}\tilde{\mathbf{E}}$ in equations (7) and (8) we have

$$(\tilde{E}_x)_k^{n+\frac{1}{2}} = \eta[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] + (\tilde{E}_x)_k^{n-\frac{1}{2}} \tag{9}$$

$$(H_y)_{k+\frac{1}{2}}^{n+1} = \eta[(\tilde{E}_x)_{k+1}^{n+\frac{1}{2}} - (\tilde{E}_x)_k^{n+\frac{1}{2}}] + (H_y)_{k+\frac{1}{2}}^n. \tag{10}$$

where $\eta = \dfrac{1}{\sqrt{\epsilon_0\mu_0}}\dfrac{\Delta t}{\Delta x}$.

Now that we have a working discrete representation of (1) and (2) we can construct an algorithm. For the time being we will choose $\eta = \frac{1}{2}$. This choice of η will be explained when we discuss boundary conditions. Using this assumption the main loop of our algorithm[3] is

```
for n = 2:max_time-1

    %Inner Loop E - Increments electric wave in space
    for k = 2:max_space
        E(k) =  E(k)+eta*(H(k-1)-H(k));
    end
```

[3]My algorithm is written for use in MATLAB. The code given by the author is written in **C**.

```
    %Hard Source- imposes a value on the grid
    pulse = exp(-.5*((t0-n)/spread)^2);
    E(center_problem_space) = pulse;

    %Inner Loop H - Increments magnetic wave in space
    for j = 1:max_space-1
        H(j) = H(j) +eta*(E(j)-E(j+1));
    end

end
```

The preceding loop may seem strange. One question that arises is where the half time, $n \pm \frac{1}{2}$ and half spacial steps, $k \pm \frac{1}{2}$ are taken into account. First, let us look at some initializations in the beginning of the program. We define two important arrays

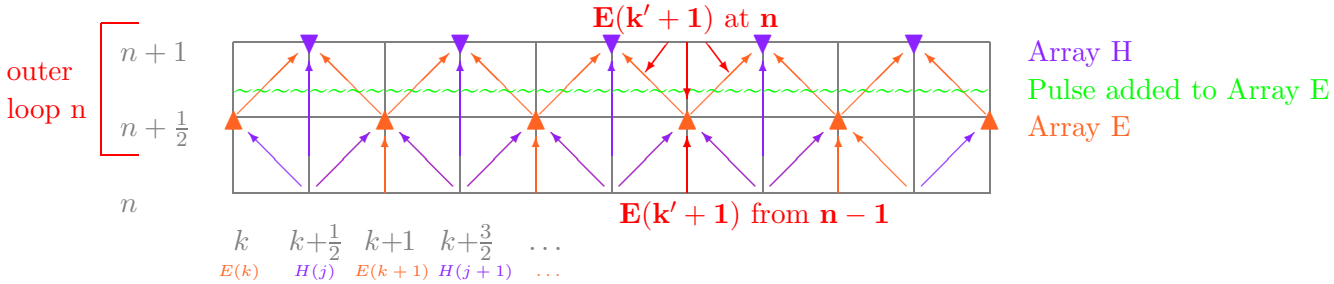
```
E = zeros(max_space,1);              %Initialize Electric array
H = E;                               %Initialize Magnetic array.
```

Each of these one-dimensional arrays is `max_space`, 200 in our simulation, in length. Each value in array E represents $(E_x)_k$ for all k, and like-wise each value in array H represents $(H_y)_j$ for all j. The spacial half steps are represented by the two different inner loops each updating the values in arrays E and H. We can think of the index of array E starting at zero and progressing by whole spacial grid steps, tying $(\tilde{E}_x)_k^{n+\frac{1}{2}}$ to array E(k) . Likewise, we must think of the index of array H as starting at $\frac{1}{2}$ and progressing by whole spacial grid steps, tying $(H_y)_{j+\frac{1}{2}}^{n+1}$ to array H(j). This seems fairly intuitive, however, the difference in time may seem less apparent.

Two things are important in understanding the algorithm compared to the equations. The first is that the values of array E are updated at time step $n + \frac{1}{2}$ whereas the values of array H are updated at time $n + 1$.

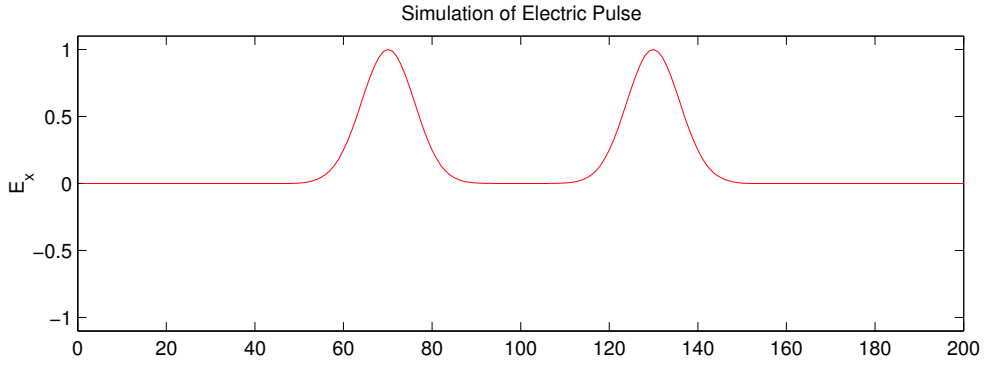
The other factor in the time equation is the placement of the pulse. It happens between the updates of array E and array H. So the pulse is factored into the arrays H and E at time step n and $n+\frac{1}{2}$ respectively. A visualization of this process is given in the figure below.

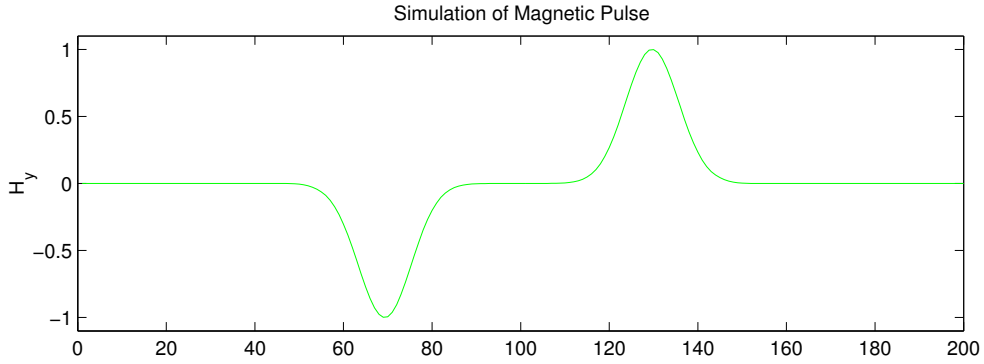
Pulse Placement in the FDTD Grid



Lastly, let's take a look at the simulation after 100 time steps. The time step Δt must be chosen carefully. We will discuss this in the next section.

Simulation of Electric and Magnetic Pulse at Time n = 100





2.3 Stability, Convergence, and the CFL Condition

In our introduction to the algorithm, we skipped an important concept. We must now take a look at the choice of cell size and how it relates to the algorithm. To do this we return to the equations.

$$(\tilde{E}_x)_k^{n+\frac{1}{2}} = \frac{1}{\sqrt{\epsilon_0\mu_0}}\frac{\Delta t}{\Delta x}[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] + (\tilde{E}_x)_k^{n-\frac{1}{2}} \tag{11}$$

$$(H_y)_{k+\frac{1}{2}}^{n+1} = \frac{1}{\sqrt{\epsilon_0\mu_0}}\frac{\Delta t}{\Delta x}[(\tilde{E}_x)_{k+1}^{n+\frac{1}{2}} - (\tilde{E}_x)_k^{n+\frac{1}{2}}] + (H_y)_{k+\frac{1}{2}}^n. \tag{12}$$

As an added feature, once we show stability, the Lax Equivalence Theorem guarantees convergence of the scheme [1]. We now perform a Fourier Stability Analysis to determine the conditions under which the FDTD scheme is stable. We follow the discussion found in [1] closely.

The Fourier modes of

$$(\tilde{E}_x)_k^{n+\frac{1}{2}} - (\tilde{E}_x)_k^{n-\frac{1}{2}} - \eta[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] = 0, \tag{13}$$

$$(H_y)_{k+\frac{1}{2}}^{n+1} - (H_y)_{k+\frac{1}{2}}^n - \eta[(\tilde{E}_x)_{k+1}^{n+\frac{1}{2}} - (\tilde{E}_x)_k^{n+\frac{1}{2}}] = 0, \tag{14}$$

which are just equations (9) and (10) rewritten are given by

$$(\tilde{E}_x)^{n-\frac{1}{2}} = \lambda^n e^{i(k\Delta x)}\hat{E} \quad \text{and} \quad (H_y)^n = \lambda^n e^{i(k\Delta x)}\hat{H}.$$

The parameter λ is known as the *amplification factor* of the mode. If $|\lambda| > 1$ then the Fourier modes will grow unboundedly as n increases, and the solution will be unstable.

Substituting into equation (13) yields

$$\lambda^{n+1}e^{i(k\Delta x)}\hat{E} - \lambda^n e^{i(k\Delta x)}\hat{E} - \eta[\lambda^n e^{i(k\Delta x+\frac{1}{2}\Delta x)}\hat{H} - \lambda^n e^{i(k\Delta x-\frac{1}{2}\Delta x)}\hat{H}] = 0.$$

Next we cancel all terms $\lambda^n e^{ik\Delta x}$ leaving

$$(\lambda - 1)\hat{E} - \eta[e^{i(\frac{1}{2}\Delta x)} - e^{i(-\frac{1}{2}\Delta x)}]\hat{H} = 0. \tag{15}$$

Using the fact that

$$e^{i\theta} = \cos\theta + i\sin\theta \quad \text{and} \quad e^{-i\theta} = \cos\theta - i\sin\theta,$$

we will reduce $e^{i(\frac{1}{2}\Delta x)} - e^{i(-\frac{1}{2}\Delta x)}$. Taking $\theta = (\frac{1}{2}\Delta x)$ we have

$$\begin{aligned} e^{i(\frac{1}{2}\Delta x)} &= \cos(\frac{1}{2}\Delta x) + i\sin(\frac{1}{2}\Delta x), \\ e^{-i(\frac{1}{2}\Delta x)} &= \cos(\frac{1}{2}\Delta x) - i\sin(\frac{1}{2}\Delta x). \end{aligned}$$

Subtracting these equations generates the equation

$$e^{i(\frac{1}{2}\Delta x)} - e^{-i(\frac{1}{2}\Delta x)} = 2i\sin(\frac{1}{2}\Delta x).$$

Returning to (15), we see that it becomes

$$(\lambda - 1)\hat{E} + 2i\eta\sin(\frac{1}{2}\Delta x)\hat{H} = 0.$$

After a similar transformation of (14) we obtain the system of equations

$$\begin{aligned} (\lambda - 1)\hat{E} + 2i\eta\sin(\frac{1}{2}\Delta x)\hat{H} &= 0, \\ 2i\eta\sin(\frac{1}{2}\Delta x)\hat{E} + (\lambda - 1)\hat{H} &= 0, \end{aligned}$$

or

$$\begin{pmatrix} \lambda - 1 & 2i\eta\sin(\frac{1}{2}\Delta x) \\ 2i\eta\sin(\frac{1}{2}\Delta x) & \lambda - 1 \end{pmatrix} \begin{pmatrix} \hat{E} \\ \hat{H} \end{pmatrix} = 0.$$

This system has non-trivial solutions only when

$$\lambda^2 - 2(1 - 2\eta^2\sin^2\frac{1}{2}\Delta x))\lambda + 1 = 0.$$

The roots of the last equation can be found using the quadratic formula. Thus we have

$$\lambda = 1 - 2\eta^2\sigma^2 \pm 2i\eta\sigma[1 - \eta^2\sigma^2]^{\frac{1}{2}},$$

with $\sigma = \sin\frac{1}{2}k\Delta x$.

Now, in order for the solution to be bounded, we must have $|\lambda| \leq 1$, or equivalently $|\lambda|^2 \leq 1$. Taking the modulus squared of λ gives

$$\begin{aligned} |\lambda|^2 &= 1 - 4(\eta\sigma)^2 + 4(\eta\sigma)^4 - 4(\eta\sigma)^2(1 - (\eta\sigma)^2), \\ &= 1 - 8(\eta\sigma)^2 + 8(\eta\sigma)^4. \end{aligned}$$

Leading to the inequality

$$1 - 8(\eta\sigma)^2 + 8(\eta\sigma)^4 \leq 1.$$

Subtracting $1 - (8\eta\sigma)^2$ from both sides and then dividing by the positive quantity $8(\eta\sigma)^2$ results in

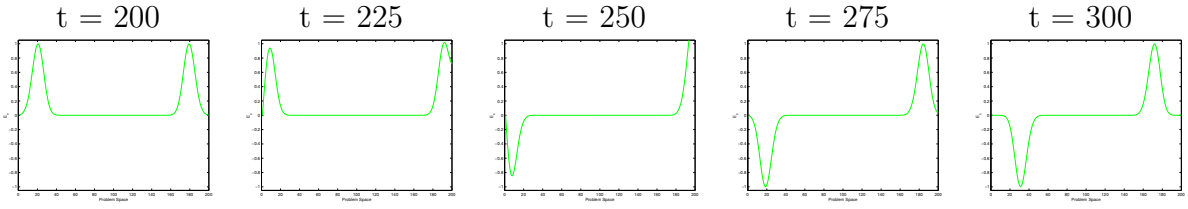
$$(\eta\sigma)^2 \quad \leq \quad 1.$$

Thus, since $0 < |sin(\frac{1}{2}k\Delta x)| \leq 1$ for all k, we have $|\sigma| \geq 1$ and hence stability is guaranteed provided $|\eta| \leq 1$. Finally, we note that in this case $|\eta| \leq 1$ is also the Courant, Friedrichs, and Lewy (CLF) condition for our scheme [1].

2.4 Absorbing Boundary Conditions

What happens in our model as time increases? Eventually, we get to the boundary of the problem space. In this case, the problem space is between the values 0 and 200. Consider the following simulation:

Graphs of E_x at Various Time Steps



We see that when the pulse hits the boundary of the problem space it is reflected. We would like our problem space to act as if the boundaries were at infinity. Then the pulse would move through the boundary without being reflected. This happens because the boundaries, E(1) and E(200) in the MATLAB arrays, act like solid walls.

In order to have the boundary act as if the wave passes through it, we must find a way to approximate values for $E(1)$ and $E(\texttt{max\_space})$. A simple approach which works in free space is to find the distance the wave moves during one time step. Then we can use the value of the array from the previous time step for the boundary value at the current time.

The distance formula, $distance = rate \times time$, will tell us how many iterations it takes the wave front to travel over one cell. The pulse moves at

the speed of light, c_0, during the time step, Δt. Recall, that Δt was defined as $\Delta t = \frac{\Delta x}{2c_0}$. Implying that the distance is given by

$$distance = c_0 \Delta t = c_0 \frac{\Delta x}{2c_0} = \frac{1}{2}\Delta x = \eta \Delta x.$$

In Sullivan's book, $\eta = \frac{1}{2}$ is the convention, in which case in two time steps the wave travels a distance Δx. We note that $\eta = \frac{1}{2}$ satisfies both the stability and the CFL conditions.

This value of η motivates Sullivan's choice of boundary conditions at the left boundary given by

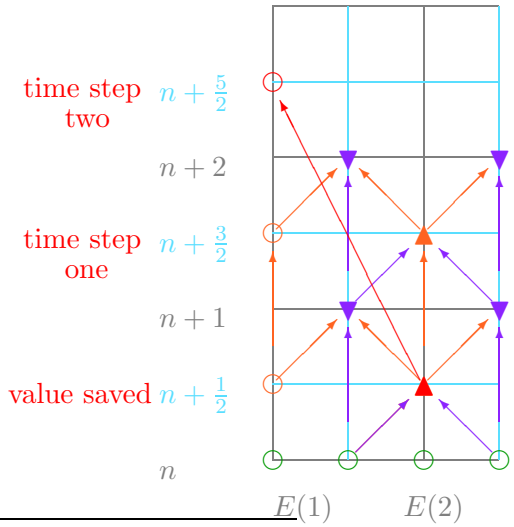
$$E(1)^n = E(2)^{n-2}.$$

Similarly, for the right boundary conditions we use

$$E(\texttt{max\_space})^n = E(\texttt{max\_space} - 1)^{n-2}.$$

The algorithm is augmented by saving the value $E(2)$ for two time steps[4] and replacing the current boundary condition, $E(1)$, with this value and like-wise for the right boundary. We illustrate in the following figure.

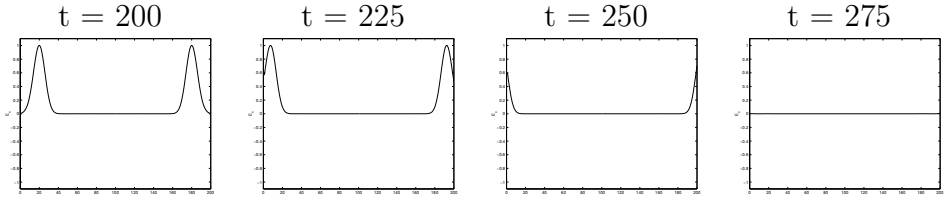
Left Absorbing Boundary Conditions Grid



[4]This is one step off of the values given by Sullivan because the arrays in MATLAB are indexed differently that arrays in C.

The figure below is similar to the one given above when there were no boundary conditions. First we will show the progression of the wave from $t = 200$ to $t = 275$ in 25 time step increments.

Graphs of E_x with Absorbing Boundary Conditions



3 Beyond Free Space

3.1 Dielectric Medium

Now that we have absorbing boundary conditions in free space we will simulate a pulse moving through a dielectric medium. Returning to Maxwell's equations, we add the dielectric constant ε_r[5],

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon_r \varepsilon_0} \nabla \times \mathbf{H} \tag{16}$$

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0} \nabla \times \mathbf{E}. \tag{17}$$

Clearly, when $\varepsilon_r = 1$ we have Maxwell's curl equations in free space; that is the dielectric constant of free space is one.

Two changes to our code are necessary with the addition of the dielectric constant. The first is merely cosmetic. We change the pulse from the center of the problem space to the left hand side. This way we will only be watching one pulse cross the problem space.

The other has to do with plotting the dielectric medium. In order to watch as the dielectric constant changes we use the following loop.

```
dielectric_begin = 100;
```

[5]The definition of a dielectric constant is:

> A measure of the ability of a material to resist the formation of an electric field within it.

```
dielectric_end = max_space;
eta = 0.5;
epsilon = 4;  %Dielectric constant - resistance of material to electric field
etaeps = eta/epsilon;
dielectic_space = ones(max_space,1);


%Dielectric Medium - wave velocity slows as it travels through the medium
for i = 1:max_space
    if ( i >= dielectric_begin & i <= dielectric_end )
        dielectric_space(i) = etaeps;
    else
        dielectric_space(i) = eta;
    end
end
```

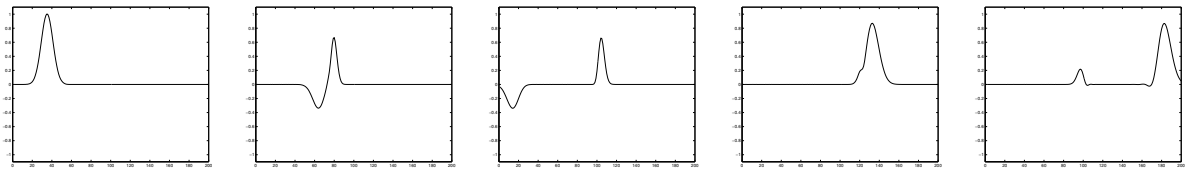
Then the loop which generates values for array E becomes:

```
 for k = 2:max_space
        E(k) =  E(k)+dielectric_space(k)*(H(k-1)-H(k));
 end
```

In the following set of figures, we see the pulse move in and out of the dielectric medium which starts at 75. We see the pulse crossing into the dielectric medium at $t = 200$ in the figure. The dielectric medium ends at 125, in the figure at $t = 400$. Notice that as the wave moves between mediums, part is reflected and part transmitted. This causes the lump on the left hand side of the pulse at $t = 400$.

Graphs of E_x at Moving Between Free Space and A Medium with Dielectric Constant 4

t = 100 t = 200 t = 300 t = 400 t = 500



Before we look at the amplitudes of the reflected and transmitted waves, notice that the absorbing boundary conditions in the last problem were

for free space. If we want absorbing boundary conditions in the dielectric medium we need to see how the medium effects the velocity of the wave and compensate in the code, which is left to the industrious reader.

3.2 Measuring our Success

In the appendix for Chapter 1, the author gives a method for verifying our simulation. We are given the equations for the reflection and transmission coefficients which give the fractional values of the amplitude reflected and transmitted waves. In our case, when $\mu = \mu_0$, Γ gives the reflection coefficient and τ gives the transmission coefficient. These equations are

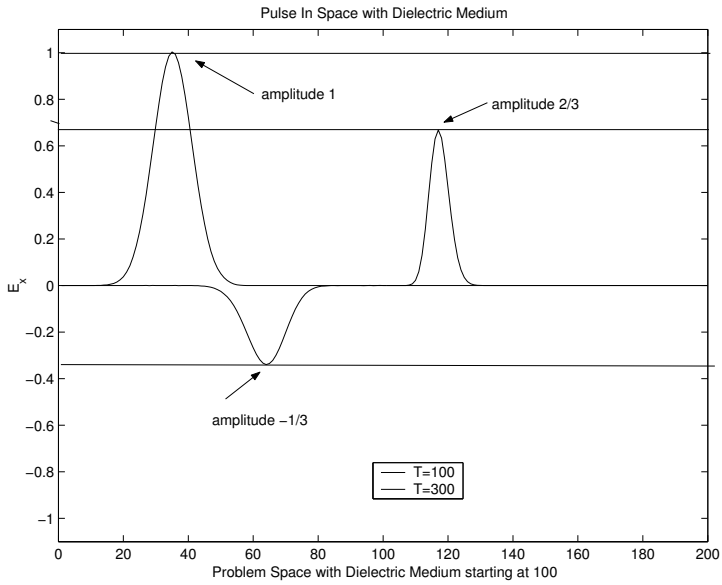
$$\begin{aligned} \Gamma &= \frac{E_{ref}}{E_{inc}} = \frac{\sqrt{\epsilon_1^*} - \sqrt{\epsilon_2^*}}{\sqrt{\epsilon_1^*} + \sqrt{\epsilon_2^*}}, \\ \tau &= \frac{E_{trans}}{E_{inc}} = \frac{2 \cdot \sqrt{\epsilon_1^*}}{\sqrt{\epsilon_1^*} + \sqrt{\epsilon_2^*}}, \end{aligned}$$

where E_{ref}, E_{inc}, and E_{trans} denote the amplitude of the reflected, transmitted, and incident electromagnetic waves respectively.

The author asks us to look at the relative amplitudes of the reflected and transmitted pulses and compare with the values of Γ and τ. Since free space has dielectric constant of 1 and the medium has dielectric constant 4, Γ and τ are

$$\begin{aligned} \Gamma &= \frac{\sqrt{\epsilon_1^*} - \sqrt{\epsilon_2^*}}{\sqrt{\epsilon_1^*} + \sqrt{\epsilon_2^*}} = \frac{1-2}{1+2} = \frac{-1}{3}, \\ \tau &= \frac{2 \cdot \sqrt{\epsilon_1^*}}{\sqrt{\epsilon_1^*} + \sqrt{\epsilon_2^*}} = \frac{2 \cdot 1}{1+2} = \frac{2}{3}. \end{aligned}$$

The figure shows how these constants are related to the waves.



Thus we see that the transmission and reflection coefficients provide an accurate measure of the amplitudes of the reflected and transmitted fields.

3.3 Lossy Dielectric Medium and a Sinusoidal Source

3.3.1 Simulating a Sinusoidal Source

To simulate a sinusoidal pulse we change our code from

```
pulse = exp( -.5*( (t0-n)/spread )^2 );

%Hard Source- imposes a value on the grid
E(center_problem_space) = pulse;
```

to

```
pulse = sin(2*pi*frequency*delta_t*n);

%Soft Source- value is added to E
E(place_pulse) = E(place_pulse) + pulse;
```

Notice, that the pulse here is placed to our specification within the problem space. Also we add the previous value of the array to the new pulse value. Finally, for the sinusoidal pulse we need the frequency of the wave as well

as the time step, consequently we will need to add some parameters to our program.

Recall, that

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

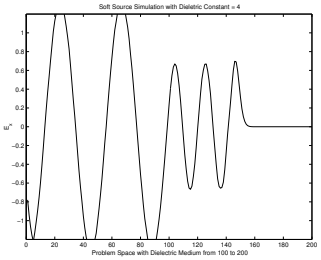
This equation corresponds to

```
delta_x = .01;
delta_t = delta_x/(2*3e8);
```

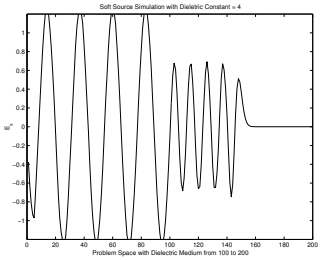
in our code. We now simulate a sinusoidal pulse at frequencies 700Mhz-increasing by 600Mhz, at time step 425 and with dielectric constant 4.

Increasing Frequencies of a Sinusoidal Pulse in a Dielectric Space

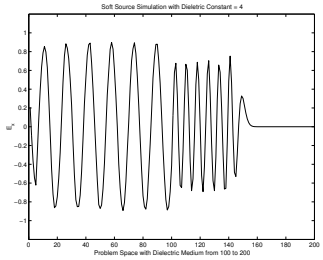
freq = 700



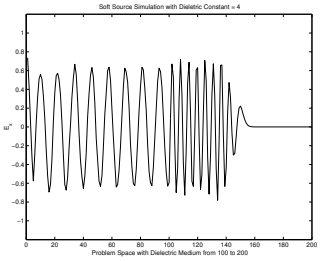
freq = 1300



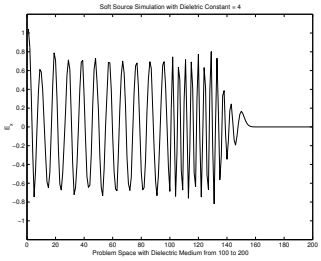
freq = 1900



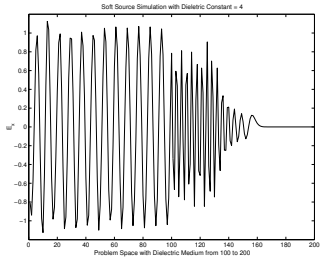
freq = 2500



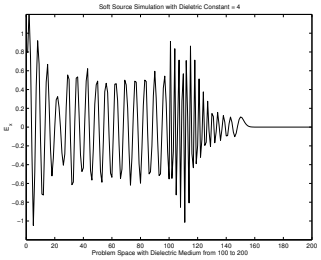
freq = 3100



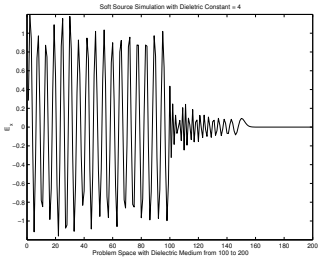
freq = 3700



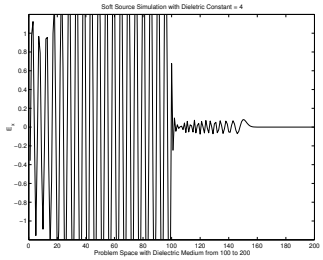
freq = 4300



freq = 4900



freq = 5500



It looks as if as the frequency increases we get a steady state between the frequency and amplitude of the pulse within and without the dielectric medium. As the frequency continues to get higher we get chaos.

3.3.2 A Lossy Medium

In building our model, we started from a simulation of Maxwell's equation in free space. Then we added a dielectric constant to make the model more viable. Now we add a loss term to the dielectric medium for an even more general model.

The measure of how well a material carries electric charge is called conductivity. Maxwell's equations in one dimension for media with conductivity are

$$\begin{aligned} \frac{\partial E_x}{\partial t} &= -\frac{1}{\varepsilon_0\varepsilon_r}\frac{\partial H_y}{\partial z} - \frac{1}{\varepsilon_0\varepsilon_r}J_x \\ \frac{\partial H_y}{\partial t} &= -\frac{1}{\mu_0}\frac{\partial E_x}{\partial z} \end{aligned} \tag{18}$$

where $J_x = \sigma E_x$. We now must rederive the difference equation for the electric field.

We will only work with the top equation, but we make the change of variable in both. After the change of variable and using σE_x in place of J_x, equation (18) becomes

$$\frac{\partial \tilde{E}_x}{\partial t} = -\frac{1}{\varepsilon_r\sqrt{\varepsilon_0\mu_0}}\frac{\partial H_y}{\partial z} - \frac{\sigma}{\varepsilon_0\varepsilon_r}\tilde{E}_x.$$

Again we take the central difference approximation in space and time. However, for the last term we take the average of the values at the different times and we get

$$\frac{(\tilde{E}_x)_k^{n+\frac{1}{2}} - (\tilde{E}_x)_k^{n-\frac{1}{2}}}{\Delta t} = -\frac{1}{\varepsilon_0\varepsilon_r}\frac{(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n}{\Delta x} - \frac{\sigma}{\varepsilon_0\varepsilon_r}\frac{(\tilde{E}_x)_k^{n+\frac{1}{2}} + (\tilde{E}_x)_k^{n-\frac{1}{2}}}{2}.$$

Making the substitution $\dfrac{1}{\sqrt{\epsilon_0\mu}}\dfrac{\Delta t}{\Delta x} = \frac{1}{2}$ and rewriting the last term we have

$$(\tilde{E}_x)_k^{n+\frac{1}{2}} - (\tilde{E}_x)_k^{n-\frac{1}{2}} = -\frac{1/2}{\varepsilon_r}[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] - \frac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}(\tilde{E}_x)_k^{n+\frac{1}{2}} - \frac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}(\tilde{E}_x)_k^{n-\frac{1}{2}}.$$

Combining the $(\tilde{E}_x)_k^{n+\frac{1}{2}}$ and the $(\tilde{E}_x)_k^{n-\frac{1}{2}}$ terms results in

$$(\tilde{E}_x)_k^{n+\frac{1}{2}} + \frac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}(\tilde{E}_x)_k^{n+\frac{1}{2}} = -\frac{1/2}{\varepsilon_r}[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] + (\tilde{E}_x)_k^{n-\frac{1}{2}} - \frac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}(\tilde{E}_x)_k^{n-\frac{1}{2}},$$

and factoring gives

$$(\tilde{E}_x)_k^{n+\frac{1}{2}}\left(1+\frac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}\right) = -\frac{1/2}{\varepsilon_r}[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] + (\tilde{E}_x)_k^{n-\frac{1}{2}}\left(1-\frac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}\right).$$

Dividing through by $1+\dfrac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}$ leaves the equation we will use in our algorithm

$$(\tilde{E}_x)_k^{n+\frac{1}{2}} = -\frac{1/2}{\varepsilon_r\left(1+\dfrac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}\right)}[(H_y)_{k+\frac{1}{2}}^n - (H_y)_{k-\frac{1}{2}}^n] + \frac{\left(1-\dfrac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}\right)}{\left(1+\dfrac{\sigma\Delta t}{2\varepsilon_0\varepsilon_r}\right)}(\tilde{E}_x)_k^{n-\frac{1}{2}}.$$

It will be necessary to give a value to ε_0 in the code. Remember that ε_0 is permittivity of free space given by

$$\varepsilon_0 = 8.8542 \times 10^{-12}\ F/m.$$

In the code we add

```
epszero = 8.85419e-12;.
```

This will allow us to define what we call the `loss_term`. The loss term affects the dielectric medium as follows:

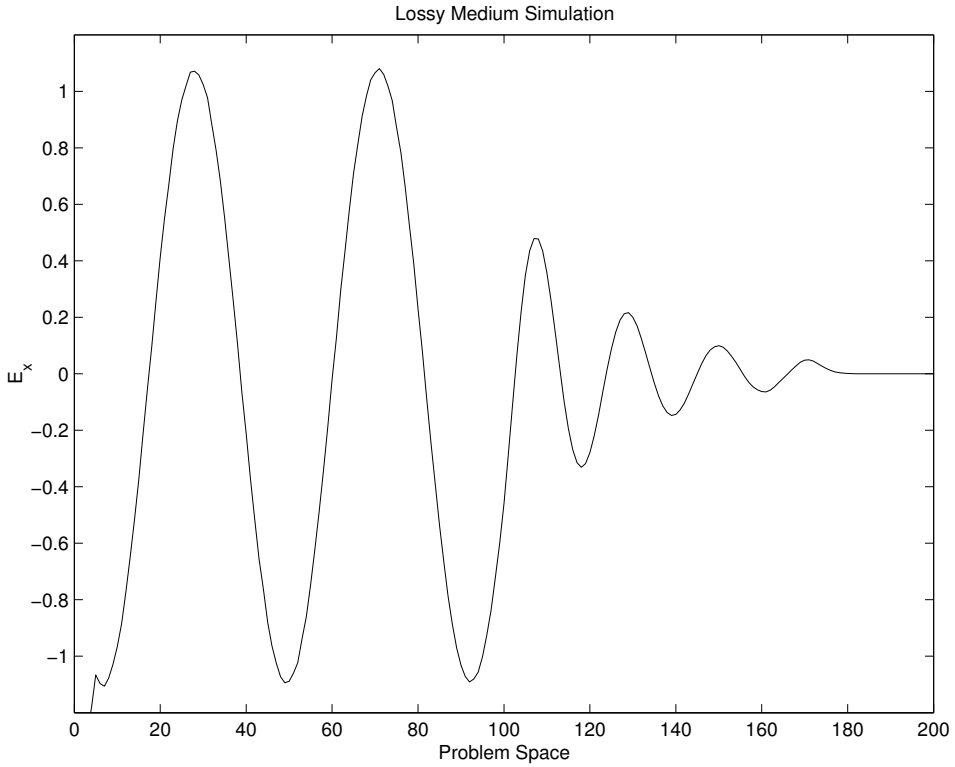
```
%Lossy Medium variables
loss_term = (delta_t*sigma) / (2*epszero*dielectric_constant);

%Set up dielectric space within the problem space
for i = 1:max_space
    if (i>=dielectric_begin & i <= dielectric_end)
        dielectric_space_a(i)=(1-loss_term)/(1+loss_term);
        dielectric_space_b(i)=0.5/(dielectric_constant *(1+loss_term));
    else
        dielectric_space_a(i)=1;
        dielectric_space_b(i)=eta;
    end
end
```

Within the time and space loops the code for the E and H values is

```
E(k) =  dielectric_space_a(k)*E(k) + dielectric_space_b(k)*(H(k-1)-H(k));
 H(j) = H(j) +eta*(E(j)-E(j+1));
```

The following figure is the simulation of the sinusoidal pulse with frequency 700Mhz, at time step 500, and dielectric constant 4. We see that the dielectric medium does not conduct well and the pulse losses energy as it moves toward the end of the problem space.



4 Conclusion

This is just the beginning of the many simulations that can be done using FDTD scheme. Dennis Sullivan's book leads the reader through simulations in two and three-dimensions and for various other media using FDTD. For interested programmers, engineers, and mathematicians there are many avenues to explore in Sullivan's book and in the current research using these techniques.

References

[1] Morton and Mayers *Numerical Solution of Partial Differential Equations*, Cambridge University Press, 1994.

[2] Sullivan *Electromagnetic Simulation Using the FDTD Method*, IEEE Press, 2000.

[3] Taflove and Hagness *Computational Electrodynamics- the finite-difference time-domain method, second addition*, Artech House, 2000.

5 Code Appendix

5.1 FDTD

FDTD code in Free Space with no absorbing boundary conditions.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FDTD  Finite-Difference Time-domain in free space no ABC
% FDTD(t) runs FDTD for time t.
%
% FDTD implements the FDTD/staggered leapfrog scheme
% over time 0-max_time and space 0-max_space.  This program
% simulates a gaussian pulse in the grid specified with
% max_time and max_space.  The pulse is sent from the center of
% the problem space and disperses in both directions
%-Kira Heater
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function myFDTD01 = myFDTD01(time)

max_time = time;
max_space = 200;                    %must be divisible by two
eta = 1/2;

E = zeros(max_space,1);             %Initialize Electric array
H = E;                              %Initialize Magnetic array
```

```
spread = 12;                          %width of pulse
center_problem_space = max_space/2;
t0 = 40;                              %center of pulse


%Outer loop steps through time -
%here E is 1/2 step behind since the pulse
%enters the problem space between the
%E and H loops.
for n = 1:max_time


    %Inner Loop E - Increments electric wave in space
    for k = 2:max_space
        E(k) =  E(k) + eta*( H(k-1)-H(k) );
    end

    %Hard Source- imposes a value on the grid
    pulse = exp( -.5*( (t0-n)/spread )^2 );
    E(center_problem_space) = pulse;


    %Inner Loop H - Increments magnetic wave in space
    for j = 1:max_space-1
        H(j) = H(j) + eta*( E(j)-E(j+1) );
    end


     %Plots progression of the electric wave
       figure(1)
       plot(E)
       axis([1 max_space -1.1 1.1])
       title('FDTD Simulation of and Electric Pulse in Free Space')
       xlabel('Problem Space')
       ylabel('E_x')
       pause(.05)

  end
```

```
    %Plots electric and magnetic fields at max_time
    figure(2)
    subplot(2,1,2)
    plot(E,'r')
    title('Simulation of Electric Pulse')
    ylabel('E_x')
    axis([0 max_space -1.1 1.1])
    subplot(2,1,1)
    plot(H,'g')
    ylabel('H_y')
    title('Simulation of Magnetic Pulse')
    axis([0 max_space -1.1 1.1])
```

5.2 Absorbing Boundary Conditions in Free Space

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ABC_Free_Space FDTD with absorbing boundary conditions
% ABC_Free_Space(T) Implements a gaussian pulse through the
% problem space for T time steps.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ABC_Free_Space = ABC_Free_Space(time)

max_time = time;
max_space = 200;            %must be divisible by two
eta = 0.5;

%Initialize arrays
E = zeros(max_space,1);             %Electric array
H = E;                              %Magnetic array


%Initialize values for ABC
E_low_m2 = 0;
E_low_m1 = 0;
E_high_m2 = 0;
```

```
E_high_m1 = 0;

%Initialize values for pulse
spread = 12;
center_problem_space = max_space/2;
t0 = 40;
place_pulse = 5;


%Main FDTD Loop
for n = 1:max_time
    %Loop for E_x
    for k = 2:max_space
        E(k) =  E(k) + eta*( H(k-1)-H(k) );
    end

    pulse = exp(-.5*((t0-n)/spread)^2);
    E(center_problem_space) = E(center_problem_space) + pulse;

    %ABC E
    E(1) = E_low_m2;
    E_low_m2 = E_low_m1;
    E_low_m1 = E(2);
    E(max_space) = E_high_m2;
    E_high_m2 = E_high_m1;
    E_high_m1 = E(max_space-1);

    %Loop for H_y
    for j = 1:max_space-1
        H(j) = H(j) + eta*( E(j)-E(j+1) );
    end
end

    figure(1)
    plot(E,'r')
    ylabel('E_x')
    axis([0 max_space -1.1 1.1])
```

5.3 Dielectric Constant

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ABC_FDTD_Die(T) Implements simulation of a Gaussian Pulse
% over T time steps.  ABC are for free space.  If boundaries are in
% the Dielectric medium then the ABC fail.  Dielectric medium begin and
% end can be specified with the code.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ABC_FDTD_Die = ABC_FDTD_Die(time)

max_time = time;
max_space = 200;
E = zeros(max_space,1);             %Electric array
H = E;                              %Magnetic array

dielectric_begin = 75;
dielectric_end = 125;
dielectic_space = ones(max_space,1);

eta = 0.5;
epsilon = 4;       %Dielectric constant
                    - resistance of material to electric field
etaeps = eta/epsilon;

%Initalize ABC conditions
E_low_m2 = 0;
E_low_m1 = 0;
E_high_m2 = 0;
E_high_m1 = 0;

%Initialize pulse variables
spread = 12;
place_pulse = 5;
center_problem_space = max_space/2;
t0 = 40.0;

%Dielectric Medium - wave velocity slows as it travels through the medium
```

```
for i = 1:max_space
    if ( i >= dielectric_begin & i <= dielectric_end )
        dielectric_space(i) = etaeps;
    else
        dielectric_space(i) = eta;
    end
end

%Main Loop
for n = 1:max_time

    for k = 2:max_space
        E(k) =  E(k)+dielectric_space(k)*(H(k-1)-H(k));
    end

    pulse = exp(-.5*((t0-n)/spread)^2);
    E(place_pulse) = E(place_pulse) + pulse;

    E(1) = E_low_m2;
    E_low_m2 = E_low_m1;
    E_low_m1 = E(2);
    E(max_space) = E_high_m2;
    E_high_m2 = E_high_m1;
    E_high_m1 = E(max_space-1);

    for j = 1:max_space-1
        H(j) = H(j) +eta*(E(j)-E(j+1));
    end

       figure(1)
       plot(E)
       axis([0 200 -1.1 1.1])
       pause(0.05)
end
```

5.4 Lossy FDTD and Sinusoidal Pulse

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lossy_FDTD FDTD in a Lossy Medium
% Lossy_FDTD(T,F) Implements a sinusoidal pulse with frequency F
% through the% problem space for T time steps.
% For a non-Lossy medium set sigma = 0.
% For Free space do the above and set dielectric_constant =1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function Lossy_FDTD = Lossy_FDTD(time,freq)

delta_x = .01;
delta_t = delta_x/(2*3e8);
max_time = time;
max_space = 200;
E = zeros(max_space,1);
H = E;
dielectric_begin = 100;
dielectric_end = max_space;
dielectic_space_a = ones(max_space,1);
dielectic_space_b = ones(max_space,1);

eta = 0.5;
dielectric_constant = 4;
etaeps = eta/dielectric_constant;

%Lossy Medium variables - for non-Lossy medium set sigma = 0;
sigma = 0.04;
epszero = 8.85419e-12;
loss_term = (delta_t*sigma) / (2*epszero*dielectric_constant);


%ABS needed for LHS of problem space do not work with Lossy medium
E_low_m2 = 0;
E_low_m1 = 0;
E_high_m2 = 0;
```

```
E_high_m1 = 0;

%Initialize Pulse Variables
frequency = freq*1e6;
place_pulse = 5;
spread = 12;
center_problem_space = max_space/2;
t0 = 40.0;

%Set up dielectric medium within the problem space
for i = 1:max_space
    if (i>=dielectric_begin & i <= dielectric_end)
        dielectric_space_a(i)=(1-loss_term)/(1+loss_term);
        dielectric_space_b(i)=0.5/(dielectric_constant *(1+loss_term));
    else
        dielectric_space_a(i)=1;
        dielectric_space_b(i)=eta;
    end
end


for n = 1:max_time

    for k = 2:max_space
        E(k) =  dielectric_space_a(k)*E(k) + dielectric_space_b(k)*(H(k-1)-H(k)
    end

    pulse = sin(2*pi*frequency*delta_t*n);
    E(place_pulse) = E(place_pulse) + pulse;

    E(1) = E_low_m2;
    E_low_m2 = E_low_m1;
    E_low_m1 = E(2);
    E(max_space) = E_high_m2;
    E_high_m2 = E_high_m1;
    E_high_m1 = E(max_space-1);

    for j = 1:max_space-1
```

```
        H(j) = H(j) +eta*(E(j)-E(j+1));
    end

       plot(E)
       title('Soft Source Simulation with Dielectric Constant = 4')
       axis([0 max_space -1.2 1.2])
       drawnow
end
```