T.C.

Gazi University

Engineering Faculty

EEE456 Electromagnetic Modelling

Group 8 Members

1-Nevzat TANVER 151110064

2-Bahadır Ömer GÜNGÖR 151110025

Instructor

Prof. Dr. ERKAN AFACAN

May 2020

Contents

1.Theoretical Information.....3

1.1 Free Space.....3

1.2 Stability 6

1.3 Absorbing Boundary Conditions 6

1.4 Dielectric Medium.....6

1.5 Simulating Different Sources.....7

1.6 Lossy Dielectric 7

2.Project Explanation.....9

2.1 Problem Statement 9

2.2 Code Explanation.....9

2.3 Full Code.....10

1.Theoretical Information

This report provides a clear illustration of one dimensional pulse propagation simulation by using Finite Difference Time Domain(FDTD) method step-by-step. Beginning with the simplest medium, several subsequent medium will be explained in this report. At the end, it will be given a quick example of this subject with "Python" script.

1.1 Free Space

The time-dependent Maxwell's curl equations for free space are

$$\frac{\partial \boldsymbol{E}}{\partial t} = \frac{1}{\varepsilon_0} \nabla \times \boldsymbol{H},$$
$$\frac{\partial \boldsymbol{H}}{\partial t} = -\frac{1}{\mu_0} \nabla \times \boldsymbol{E}.$$

Figure 1.1.1 Maxwell's Equations

E and H vectors are three-dimensional so these equations represents three equations for each coordinates x,y and z. We have a one dimensional problem so we can simplify these equations just using Ex and Hy.

$$\frac{\partial E_x}{\partial t} = -\frac{1}{\varepsilon_0} \frac{\partial H_y}{\partial z},$$
$$\frac{\partial H_y}{\partial t} = -\frac{1}{\mu_0} \frac{\partial E_x}{\partial z}.$$

Figure 1.1.2 Maxwell's Equations

These are the equations of a plane wave travelling in the z-direction with the electric field oriented in the x direction (Ex) and the magnetic field oriented in the y direction (Hy).

Taking the central difference approximations for both the temporal and spatial derivatives gives

$$\frac{E_x^{n+1/2}(k) - E_x^{n-1/2}(k)}{\Delta t} = -\frac{1}{\varepsilon_0} \frac{H_y^n(k+\frac{1}{2}) - H_y^n(k-\frac{1}{2})}{\Delta x},$$
$$\frac{H_y^{n+1}(k+\frac{1}{2}) - H_y^n(k+\frac{1}{2})}{\Delta t} = -\frac{1}{\mu_0} \frac{E_x^{n+1/2}(k+1) - E_x^{n+1/2}(k)}{\Delta x}.$$

Figure 1.1.3 Central Difference Approximations of Maxwell's Equations

In these two equations, time is specified by the superscripts that is n represents a time step, and the time is t=Δt.n . In order to execute this simulation on the computer, we have to discretize everything into the computer. The term n+1 refers to one time step later and the terms in the parantheses next to the Ex and Hy(for example k+1/2) represent distance that is k is used to calculate the distance z=Δx.k **(We know that propagation is in z-direction but we write Δx instead of Δz because these formulas are like this in the literature so don't worry about it.)** Moreover, the exponents of Ex and Hy seems complicated. The term (n+1/2) illustrates us that the H field values are assumed to be located between E field values. We can infer that the E and H fields are interleaved in both space and time. Consequently, in the computer program step-by-step we will compute E and H in order. This situation is described in the table below.

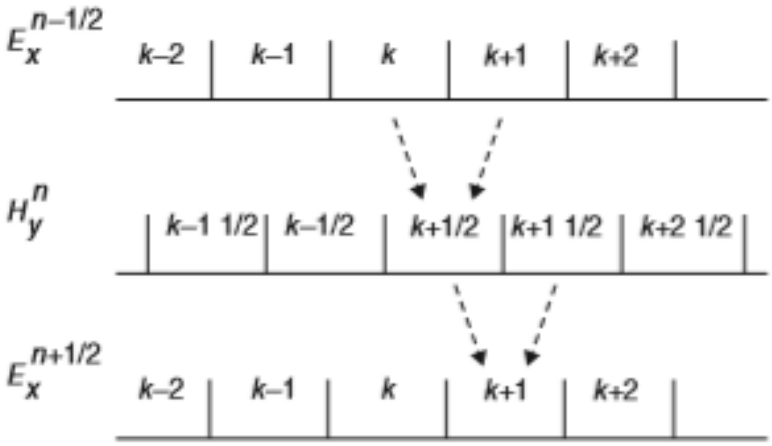


Figure 1.1.4 Step-By-Step Calculation

This is the fundamental paradigm of the FDTD method.

We can rewrite equations like below:

$$E_x^{n+1/2}(k) = E_x^{n-1/2}(k) - \frac{\Delta t}{\varepsilon_0 \cdot \Delta x}\left[H_y^n\left(k+\frac{1}{2}\right) - H_y^n\left(k-\frac{1}{2}\right)\right]$$

$$H_y^{n+1}\left(k+\frac{1}{2}\right) = H_y^n\left(k+\frac{1}{2}\right) - \frac{\Delta t}{\mu_0 \cdot \Delta x}\left[E_x^{n+1/2}(k+1) - E_x^{n+1/2}(k)\right]$$

Figure 1.1.5 Rewrited Maxwell's Equations

New value of E is calculated from the previous Ex and most recent values of Hy.

Two equations above are very similar, but because ε0 and μ0 differ by several orders of magnitude, Ex and Hy will differ by several orders of magnitude. This is circumvented by making the following change of variables:

$$\tilde{E} = \sqrt{\frac{\varepsilon_0}{\mu_0}} E.$$

We can rewrite the equations again owing to this substition.

$$\tilde{E}_x^{n+1/2}(k) = \tilde{E}_x^{n-1/2}(k) - \frac{\Delta t}{\sqrt{\varepsilon_0 \mu_0} \cdot \Delta x}\left[H_y^n\left(k + \frac{1}{2}\right) - H_y^n\left(k - \frac{1}{2}\right)\right]$$

$$H_y^{n+1}\left(k + \frac{1}{2}\right) = H_y^n\left(k + \frac{1}{2}\right) - \frac{\Delta t}{\sqrt{\varepsilon_0 \mu_0} \cdot \Delta x}\left[\tilde{E}_x^{n+1/2}(k+1) - \tilde{E}_x^{n+1/2}(k)\right]$$

Figure 1.1.6 Rewrited Maxwell's Equations

Once the cell size Δx is chosen, then the time step Δt is determined by

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

where c0 is the speed of light in free space. Therefore, remembering that ε0μ0 = 1/(c0*c0)

$$\frac{\Delta t}{\sqrt{\varepsilon_0 \mu_0} \cdot \Delta x} = \frac{\Delta x}{2 \cdot c_0} \cdot \frac{1}{\sqrt{\varepsilon_0 \mu_0} \cdot \Delta x} = \frac{1}{2}.$$

This point will be explained in the following sections of this report.

The following points are important about the program:

1. The Ex and Hy values are calculated by separate loops, and they employ the interleaving described above.

2. After the Ex values are calculated, the source is calculated. This is done by simply specifying a value of Ex at the point k = kc and overriding what was previously calculated.

1.2 Stability

We need to discuss how to determine time step. An EM wave cannot go faster than light speed even if it is in the free space. To propagate a distance of one cell requires a minimum time of $\Delta t = \Delta x/c0$. With a two dimensional simulation, we must allow for the propagation in the diagonal direction, which brings the requirement to $\Delta t = \Delta x/((\sqrt{2})c0)$. Obviously, a three-dimensional simulation requires $\Delta t = \Delta x/((\sqrt{3})c0)$. This is summarized by the well-known Courant Condition. In this report, we will use the formula below in order to avoid square roots.

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

Figure 1.2.1 Time Step

1.3 Absorbing Boundary Conditions

Absorbing boundary conditions are necessary to keep outgoing E and H fields from being reflected back into the problem space. In literature, there are a few explanation about this situation but we can easily apply boundary conditions by resetting the values of the E and H on the edges. Simple python script is given below.

```
# Absorbing Boundary Conditions
ex[0] = boundary_low.pop(0)
boundary_low.append(ex[1])

ex[ke - 1] = boundary_high.pop(0)
boundary_high.append(ex[ke - 2])
```

1.4 Dielectric Medium

In order to simulate a medium with a dielectric constant other than 1, which corresponds to free space, we have to add the relative dielectric constant εr to Maxwell's equations:

$$\frac{\partial \boldsymbol{E}}{\partial t} = \frac{1}{\varepsilon_r \varepsilon_0} \nabla \times \boldsymbol{H},$$
$$\frac{\partial \boldsymbol{H}}{\partial t} = -\frac{1}{\mu_0} \nabla \times \boldsymbol{E}.$$

Figure 1.4.1 Maxwell's Equations in Dielectric Medium

Procedures for the free space is same with dielectric medium. If we can apply same procedures for dielectric medium, we can obtain the formula below.

$$\widetilde{E}_x^{n+1/2}(k) = \widetilde{E}_x^{n-1/2}(k) - \frac{1}{2 \cdot \varepsilon_r}\left[H_y^n\left(k+\frac{1}{2}\right) - H_y^n\left(k-\frac{1}{2}\right)\right],$$

$$H_y^{n+1}\left(k+\frac{1}{2}\right) = H_y^n\left(k+\frac{1}{2}\right) - \frac{1}{2}\left[\widetilde{E}_x^{n+1/2}(k+1) - \widetilde{E}_x^{n+1/2}(k)\right].$$

Figure 1.4.2 Rewrited Maxwell's Equations

1.5 Simulating Different Sources

If we need to simulate different source wave propagations, we need to change some parameters in python scripts given in the last pages. Instead of gaussian pulse, we can prefer sine wave. The code is given below.

```
# Put a wave at the low end
    #pulse = exp(-0.5 * ((t0 - time_step) / spread) ** 2)
    pulse = sin(2 * pi * freq_in * dt * time_step)
    ex[5] = pulse + ex[5]
```

1.6 Lossy Dielectric

There are many media that also have a loss term specified by the conductivity. This loss term results in the attenuation of the propagating energy. Once more we will start with the time-dependent Maxwell's curl equations, but we will write them in a more general form, which allows us to simulate propagation in media that have conductivity:

$$\varepsilon_r \varepsilon_0 \frac{\partial \boldsymbol{E}}{\partial t} = \nabla \times \boldsymbol{H} - \boldsymbol{J},$$

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

Figure 1.6.1 Maxwell's Equations

J, the current density, can also be written as

$$\boldsymbol{J} = \sigma \boldsymbol{E},$$

If we apply same processes before applied for free space, we can obtain the Ex formula like below.

$$E_x^{n+1/2}(k) = \frac{\left(1 - \frac{\Delta t \cdot \sigma}{2\varepsilon_r \varepsilon_0}\right)}{\left(1 + \frac{\Delta t \cdot \sigma}{2\varepsilon_r \varepsilon_0}\right)} E_x^{n-1/2}(k) - \frac{\left(\frac{1}{2}\right)}{\varepsilon_r \left(1 + \frac{\Delta t \cdot \sigma}{2\varepsilon_r \varepsilon_0}\right)} \left[H_y^n\left(k + \frac{1}{2}\right) - H_y^n\left(k - \frac{1}{2}\right)\right].$$

Figure 1.6.2 Maxwell's Equations in Lossy Dielectric Medium

The formula of Hy is same because in our project the value "magnetic permeability (μr)" does not change at all.

From these we can get the latest computer equations:

```
ex[k] = ca[k] * ex[k] + cb[k] * (hy[k - 1] - hy[k])
hy[k] = hy[k] + 0.5 * (ex[k] - ex[k + 1]),
```

where:

```
 eaf = dt * sigma/(2 * epsz * epsilon),
ca[k] = (1 - eaf)/(1 + eaf),
cb[k] = 0.5/(epsilon * (1 + eaf)).
```

The terms in the program above are:

epsilon : εr

dt : time step

epsz : ε0

sigma : conductivity

If we need to summarize the subject :

1.In free space, sigma is equal to zero because there is no loss and epsilon is one in air so ca[k] becomes 1 and cb[k] becomes 0.5

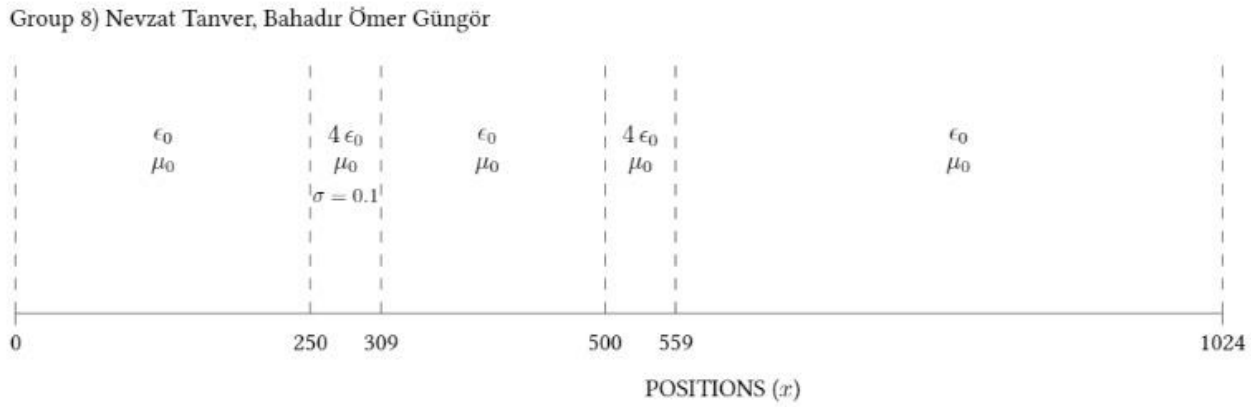
2.In dielectric medium, sigma is still zero because there is still no loss and epsilon is equal to the dielectric permittivity of medium.

3.In lossy dielectric medium, sigma is different from zero because there is a loss of magnitude, epsilon is equal to the dielectric permittivity of medium.

2.Project Explanation

2.1 Problem Statement

In our project, we are asked to send the signal from the low end to the high end through two different medium. The first medium(250-309) is lossy dielectric medium and the second one(500-559) is a dielectric medium without loss, other mediums are free space. The surface is given below.



Because we have two different medium, we used two arrays in our code in order to explain two different medium. Indeed, the formula is same for everywhere except its magnitude so two arrays are defined with different magnitude. They are shown below.

```
ca1[cb_start:309] = (1 - eaf1) / (1 + eaf1)
cb1[cb_start:309] = 0.5 / (epsilon1 * (1 + eaf1))
cb2[500:559] = 0.5 / (epsilon1)
```

Where

```
eaf1 = dt * sigma1 / (2 * epsz * epsilon1)
```

In the following section, there is a full explanation about code. At the last section of report, there is the python script of this project.

2.2 Code Explanation

In the first section, we define constants such as ke (array size), sigma, epsilon. In addition, we create the ex, hy arrays. For this, we use zeros from the Numpy module. This gives us a array full of zeros. In the same way, we define the coefficients such as ca1, cb1. Then we calculate their values in accordance with the assignment.

After all these processes, we create the time loop. The wave will propagate through each time step of this loop. The first process of the loop is to create a pulse. We do this using the function

in the gauss distribution, which can also be remembered from statistics. The elements of the Gauss wave are constantly added to the array. It is convenient to add at every step, as it will already approach zero after a point. The next process of the loop is to calculate the electric and magnetic fields. It does this in accordance with the formula. The important point here is to dampen the boundary values after calculating the electric field. Because if it is not damped, wave will be reflected. After all these operations, a new version of the wave is drawn on the screen every 10 steps. Thus, we get the chance to observe the wave easily.

2.3 Full Code

```
import numpy as np
from math import pi, exp
from matplotlib import pyplot as plt

ke = 1024
ex = np.zeros(ke)
hy = np.zeros(ke)
ddx = 0.01  # Cell size
dt = ddx / 6e8  # Time step size

boundary_low = [0, 0]
boundary_high = [0, 0]

# Create Dielectric Profile
epsz = 8.854e-12
epsilon1 = 4
sigma1 = 0.04
sigma2 = 0

ca1 = np.ones(ke)
cb1 = np.ones(ke) * 0.5
ca2 = np.ones(ke)
cb2 = np.ones(ke) * 0.5
cb_start = 250

eaf1 = dt * sigma1 / (2 * epsz * epsilon1)
ca1[cb_start:309] = (1 - eaf1) / (1 + eaf1)
cb1[cb_start:309] = 0.5 / (epsilon1 * (1 + eaf1))
cb2[500:559] = 0.5 / (epsilon1)

nsteps = 5000
# Pulse constants
spread = 12
t0 = 40

# Create drawing area
fig = plt.figure(figsize=(15, 2.25))

# Main FDTD Loop
for time_step in range(1, nsteps + 1):

    # Put a pulse at the low end
    pulse = exp(-0.5 * ((t0 - time_step) / spread) ** 2)
    ex[5] = pulse + ex[5]
```

```
# Calculate the Ex field
for k in range(1, 309):
    ex[k] = ex[k] * ca1[k] + cb1[k] * (hy[k - 1] - hy[k])

for k in range(309, ke):
    ex[k] = ex[k] + cb2[k] * (hy[k - 1] - hy[k])

# Absorbing Boundary Conditions
ex[0] = boundary_low.pop(0)
boundary_low.append(ex[1])
ex[ke - 1] = boundary_high.pop(0)
boundary_high.append(ex[ke - 2])

# Calculate the Hy field
for k in range(ke - 1):
    hy[k] = hy[k] + 0.5 * (ex[k] - ex[k + 1])

# Draw Graph
if time_step % 10 == 0:
    plt.clf()
    plt.rcParams['font.size'] = 12
    # plt.figure(figsize=(15, 2.25))
    plt.plot(ex, color='r', linewidth=1)
    plt.ylabel('E$_x$', fontsize='14')
    plt.xticks(np.arange(0, 1024, step=100))
    plt.xlim(0, 1024)
    plt.yticks(np.arange(-1, 1.2, step=1))
    plt.ylim(-1.2, 1.2)
    plt.text(50, 0.5, 'T = {}'.format(time_step),
             horizontalalignment='center')
    plt.plot((0.5 / cb1 - 1) / 3, 'k--',
             linewidth=0.75)  # The math on cb is just for scaling
    plt.plot((0.5 / cb2 - 1) / 3, 'k--',
             linewidth=0.75)  # The math on cb is just for scaling
    plt.text(275, 0.7, 'εr = {}'.format(epsilon1),
             horizontalalignment='center')
    plt.text(275, 0.5, 'μ0 ',
             horizontalalignment='center')
    plt.text(280, 0.3, 'σ = {}'.format(sigma1),
             horizontalalignment='center')
    plt.text(530, 0.7, 'εr = {}'.format(epsilon1),
             horizontalalignment='center')
    plt.text(530, 0.5, 'μ0 ',
             horizontalalignment='center')
    plt.xlabel('FDTD cells')
    fig.canvas.draw()
    plt.pause(1e-5)
```