Task 1: Absorbing Boundary Conditions

Ang, Angeleene S.

May 13, 2016

Background

Some calculations would require the background material to be extended infinitely in at least one direction. However, given the limitations on computer storage, we need to be able to create boundaries such that the domain appears to be infinite.

Here, we consider the local boundary conditions designed by Engquist and Majda in 1977. These were developed such that the amplitude of the reflection coefficients were small, and that the BCs, along with the associated differential equation, guarantee a well-posed initial boundary value problem.

Derivation

For the 1-dimensional wave equation given by

$$\frac{\partial^2 U}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 U}{\partial t^2} \tag{1}$$

we can define an operator G where

$$G = \frac{\partial^2}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2}{\partial t^2} \tag{2}$$

so that the wave equation can be written easily as

$$GU = 0 \tag{3}$$

We can separate G such that

$$G = G^+ G^- \tag{4}$$

$$G^+ = \frac{\partial}{\partial x} + \frac{1}{c}\frac{\partial}{\partial t} \tag{5}$$

$$G^- = \frac{\partial}{\partial x} - \frac{1}{c}\frac{\partial}{\partial t} \tag{6}$$

At the left side of the boundary, where $x = 0$, the application of the operator G^- to any wavefunction U exactly absorbs a plane wave propagating toward the left boundary at any incident angle. Simularly, the operator G^+ acting on U absorbs a plane wave propagating toward the right boundary. Hence we have

$$G^- U = 0 \qquad \longrightarrow \qquad \frac{\partial U}{\partial x} - \frac{1}{c}\frac{\partial U}{\partial t} = 0 \tag{7}$$

and at the right side, at $x = d$,

$$G^+ U = 0 \qquad \longrightarrow \qquad \frac{\partial U}{\partial x} + \frac{1}{c}\frac{\partial U}{\partial t} = 0 \tag{8}$$

We take the time derivative of both of the equations above

$$\frac{\partial^2 U}{\partial x \partial t} - \frac{1}{c}\frac{\partial^2 U}{\partial t^2} = 0 \qquad \text{at } x = 0 \tag{9}$$

$$\frac{\partial^2 U}{\partial x \partial t} + \frac{1}{c}\frac{\partial^2 U}{\partial t^2} = 0 \qquad \text{at } x = d \tag{10}$$

in order to be able to use the following expressions for the partial derivatives

$$\left.\frac{\partial^2 W}{\partial x \partial t}\right|_{1/2}^{n} = \frac{1}{2}\left[\left(\frac{W|_1^{n+1} - W|_0^{n+1}}{\Delta t \Delta x}\right) - \left(\frac{W|_1^{n-1} - W|_0^{n-1}}{\Delta t \Delta x}\right)\right] \tag{11}$$

$$= \frac{1}{2}\left[\left(\frac{W|_i^{n+1} - W|_{i-1}^{n+1}}{\Delta t \Delta x}\right) - \left(\frac{W|_i^{n-1} - W|_{i-1}^{n-1}}{\Delta t \Delta x}\right)\right] \tag{12}$$

$$\left.\frac{\partial^2 W}{\partial t^2}\right|_{1/2}^{n} = \frac{1}{2}\left[\left(\frac{W|_0^{n+1} - 2W|_0^{n} + W|_0^{n-1}}{(\Delta t)^2}\right) + \left(\frac{W|_1^{n+1} - 2W|_1^{n} + W|_1^{n-1}}{(\Delta t)^2}\right)\right] \tag{13}$$

$$= \frac{1}{2}\left[\left(\frac{W|_{i-1}^{n+1} - 2W|_{i-1}^{n} + W|_{i-1}^{n-1}}{(\Delta t)^2}\right) + \left(\frac{W|_i^{n+1} - 2W|_i^{n} + W|_i^{n-1}}{(\Delta t)^2}\right)\right] \tag{14}$$

At $x = 0$, we want to obtain the function at the next time step given the information from the previous cycle. Here, we are looking for $W\Big|_0^{n+1}$. Substituting the expressions for the partial derivatives into the wave equation yields

$$W|_0^{n+1} = -W|_1^{n-1} + \left(\frac{c\Delta t - \Delta x}{c\Delta t + \Delta x}\right)\left(W|_1^{n+1} + W|_0^{n-1}\right) + \frac{2\Delta x}{c\Delta t + \Delta x}\left(W|_0^{n} + W|_1^{n}\right) \tag{15}$$

In the code, we will initialize all values of W to be zero.

At one iteration, we have the value for $W\Big|_1^{n+1}$. After we use the above equation to obtain $W\Big|_0^{n+1}$, the program will then transfer the information in the variable $W\Big|^{n-1}$ to $W\Big|^{n}$, and $W\Big|^{n}$ to $W\Big|^{n+1}$, so that the following iteration can use these values.

Similarly at the other side of the boundary, we want to obtain $W\Big|_i^{n+1}$.

$$W|_i^{n+1} = -W|_{i-1}^{n-1} + \left(W|_{i-1}^{n+1} + W|_i^{n-1}\right)\left(\frac{c\Delta t - \Delta x}{c\Delta t + \Delta x}\right) + \frac{2\Delta x}{c\Delta t + \Delta x}\left(W|_i^{n} + W|_{i-1}^{n}\right) \tag{16}$$

Similar to the left side of the boundary, we initialize all values of W at zero. At a given iteration, we have $W\Big|_{i-1}^{n+1}$. After solving for $W\Big|_i^{n+1}$, the program will cycle through the variables in the same process as described in the left boundary.

1 Python Implementation

First, we start with importing several libraries for plotting, mathematical functions, and numpy for the arrays. The variable fignum is also defined; this is used to create several plots of the field.

```
In [1]: %matplotlib inline
        from matplotlib import pyplot as plt

        import numpy as np
```

```
import math as m

fignum = 0
```

We then define constants for the size of the spatial domain nx, the wave impedance $Z = \sqrt{\frac{\mu}{\epsilon}}$, and the speed of light $c = \frac{1}{\sqrt{\mu\epsilon}}$. We assume that the field is propagating in vacuum, hence $\epsilon = 1$ and $\mu = 1$. The spatial domain size is defined arbitrarily.

In [2]:
```
nx = 300
imp0 = 337.0
epsilon = 1
c = 1/np.sqrt(epsilon)
```

For the Gaussian source, we introduce variables for its location, width, and delay. The location of the source is placed at the center of the domain, the width is defined arbitrarily, and the ratio of the delay with respect to the source is taken from the previous task.

We also define the temporal domain at this point. The term nx+srcdel is taken from the previous task, with the factor 1.5 added in order to view the field as it interacts with the boundary conditions.

In [3]:
```
srcori = int(nx/2)              #source origin
srcwid = 30.0*np.sqrt(epsilon)  #source width
srcdel = 10*srcwid              #source delay
nt = int(1.5*nx+srcdel)
```

Now, we create arrays for the E_z and H_y components of the field, along with an array x for the spatial coordinates. Both fields are initialized at zero.

In [4]:
```
ez = np.zeros(nx)
hy = np.zeros(nx)
x = np.arange(0,nx-1,1)
```

Note that we defined the domain as integer steps, not as a real spatial interval divided into smaller intervals. Hence, the step dx and dt are already predefined as unity. We define the constants a and b such that

$$a = \frac{c\Delta t - \Delta x}{c\Delta t + \Delta x} \tag{17}$$

$$b = \frac{2\Delta x}{c\Delta t + \Delta x} \tag{18}$$

and we assign these to variables

In [5]:
```
a = (c-1)/(c+1)
b = 2/(c + 1)
```

We also define variables for the boundary conditions, for both sides of the E-field and for both sides of the H-field. At the left, we have

$$hwnp10 \rightarrow H\Big|_0^{n+1} \qquad ewnp10 \rightarrow E|_0^{n+1} \tag{19}$$

$$hwnm11 \rightarrow H\Big|_0^{n-1} \qquad ewnm11 \rightarrow E|_0^{n-1} \tag{20}$$

$$hwnp11 \rightarrow H\Big|_1^{n+1} \qquad ewnp11 \rightarrow E|_1^{n+1} \tag{21}$$

$$hwnm10 \rightarrow H\Big|_0^{n-1} \qquad ewnm10 \rightarrow E|_0^{n-1} \tag{22}$$

$$hwn0 \rightarrow H\Big|_0^{n} \qquad ewn0 \rightarrow E|_0^{n} \tag{23}$$

$$hwn1 \rightarrow H\Big|_1^{n} \qquad ewn1 \rightarrow E|_1^{n} \tag{24}$$

And at the right,

$$hwnp1im1 \to H\Big|_{i-1}^{n+1} \qquad ewnp1im1 \to E|_{i-1}^{n+1} \tag{25}$$

$$hwnm1i \to H\Big|_{i-1}^{n-1} \qquad ewnm1i \to E|_{i-1}^{n-1} \tag{26}$$

$$hwnp1i \to H\Big|_{i}^{n+1} \qquad ewnp1i \to E|_{i}^{n+1} \tag{27}$$

$$hwnm1im1 \to H\Big|_{i-1}^{n-1} \qquad ewnm1im1 \to E|_{i-1}^{n-1} \tag{28}$$

$$hwnim1 \to H\Big|_{i-1}^{n} \qquad ewnim1 \to E|_{i-1}^{n} \tag{29}$$

$$hwni \to H\Big|_{i}^{n} \qquad ewni \to E|_{i}^{n} \tag{30}$$

All of these variables are initialized to zero as the field is initially zero.

```
In [6]: hwnp10, ewnp10 = 0,0 # W / ^{n+1} _{0}
        hwnm11, ewnm11 = 0,0 # W / ^{n-1} _{1}
        hwnp11, ewnp11 = 0,0 # W / ^{n+1} _{1}
        hwnm10, ewnm10 = 0,0 # W / ^{n-1} _{0}
        hwn0  , ewn0   = 0,0 # W / ^{n  } _{0}
        hwn1  , ewn1   = 0,0 # W / ^{n  } _{1}

        hwnp1im1, ewnp1im1 = 0,0 # W / ^{n+1} _{i-1}
        hwnm1i  , ewnm1i   = 0,0 # W / ^{n-1} _{i  }
        hwnp1i  , ewnp1i   = 0,0 # W / ^{n+1} _{i  }
        hwnm1im1, ewnm1im1 = 0,0 # W / ^{n-1} _{i-1}
        hwnim1  , ewnim1   = 0,0 # W / ^{n  } _{i-1}
        hwni    , ewni     = 0,0 # W / ^{n  } _{i  }
```

For the actual loop, we begin with modifying the loop of the field with an additive source from the previous task. The source is a Gaussian field at the origin, defined here in line 24. The electric and magnetic fields propagate in lines 5 and 24.

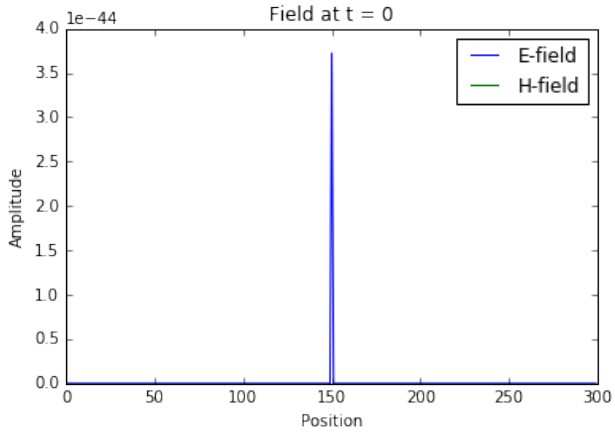
In applying the boundary conditions at the left, we begin with setting the variable $H\Big|_{1}^{n+1}$ ($E\Big|_{1}^{n+1}$) using the field that was obtained in line 5(24). then we use the equation that we derived for $H\Big|_{0}^{n+1}$ ($ E j^{n+1}_{0}) in line 9(29). This then becomes the initial value of the field at $x = 0$ for the next iteration. We then cycle through the values of $H(E)$ which will be used for the next run of the loop.

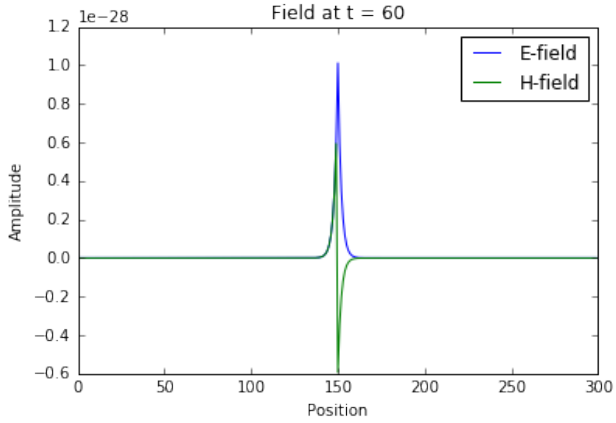
Similarly to the right, we begin with $H\Big|_{i-1}^{n+1}$ ($E\Big|_{i-1}^{n+1}$), and assign to this the value of the field at at $nx - 2$ (note that the field goes only from 0 to $nx - 1$). In order to obtain the field at $nx - 1$, we use the previously derived equation and assign this to the array at $x = nx - 1$. Then we also cycle through the values of $H(E)$ for the next iteration.

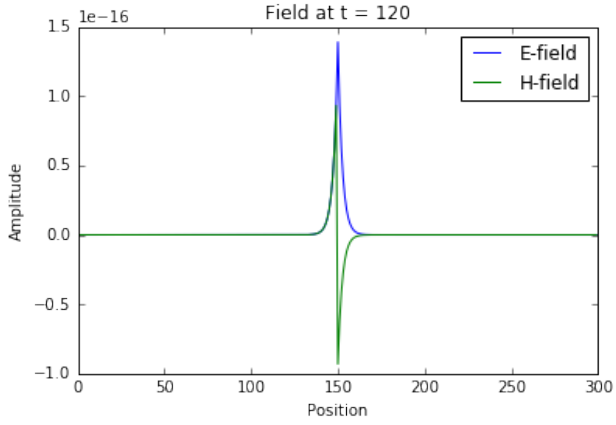
The code starting from line 41 is used to plot the fields.

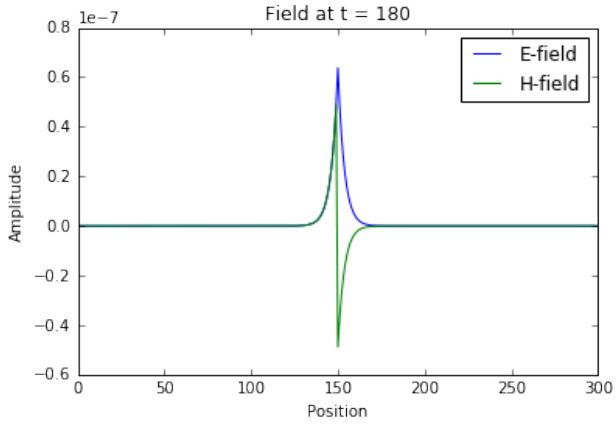
```
In [7]: for dt in range(0,nt):
            ######################
            #Magnetic field
            ######################
            hy[x] = hy[x] + (ez[x+1] - ez[x])/imp0

            #abc at left
            hwnp11 = hy[1]
```

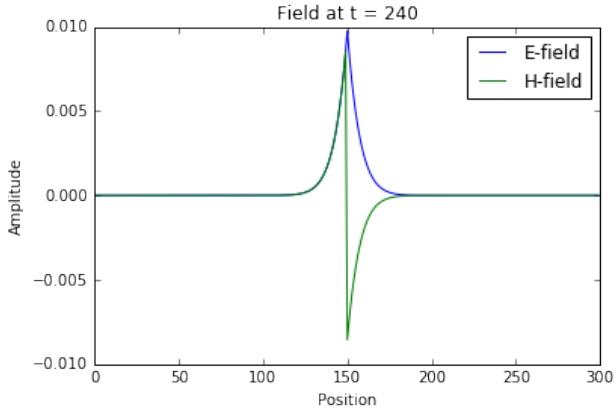
```
hwnp10 = -hwnm11 + a*(hwnp11 + hwnm10) + b*(hwn0 + hwn1)
hy[0] = hwnp10
hwnm11, hwnm10 = hwn1, hwn0
hwn1, hwn0  = hwnp11, hwnp10

#abc at right
hwnp1im1 = hy[-2]
hwnp1i = - hwnm1im1 + a*(hwnp1im1 + hwnm1i) + b*(hwnp1i + hwnim1)
hy[-1] = hwnp1i
hwnm1i, hwnm1im1 = hwni, hwnim1
hwni, hwnim1  = hwnp1i, hwnp1im1

######################
#Electric field
######################
ez[x+1] = ez[x+1] + (hy[x+1]-hy[x])*imp0/epsilon
ez[srcori] += m.exp(-((dt-srcdel)*(dt-srcdel))/(srcwid*srcwid))

#abc at left
ewnp11 = ez[1]
ewnp10 = -ewnm11 + a*(ewnp11 + ewnm10) + b*(ewn0 + ewn1)
ez[0] = ewnp10
ewnm11, ewnm10 = ewn1, ewn0
ewn1, ewn0  = ewnp11, ewnp10

#abc at right
ewnp1im1 = ez[-2]
ewnp1i = - ewnm1im1 + a*(ewnp1im1 + ewnm1i) + b*(ewnp1i + ewnim1)
ez[-1] = ewnp1i
ewnm1i, ewnm1im1 = ewni, ewnim1
ewni, ewnim1  = ewnp1i, ewnp1im1

plt.hold(True)
if (dt % 60 == 0 or dt == nt-2):
    fignum = fignum + 1
    plt.figure(fignum)
    plt.xlabel("Position")
    plt.ylabel("Amplitude")
    plt.title("Field at t = "+ str(dt))
    plt.plot(ez, label="E-field")
    plt.plot(hy*imp0, label="H-field")
    plt.legend()
```

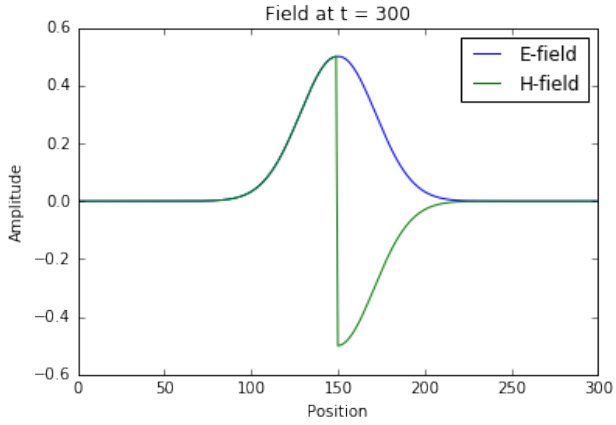

Field at t = 0
1e−44
E-field
H-field
Amplitude
Position
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300

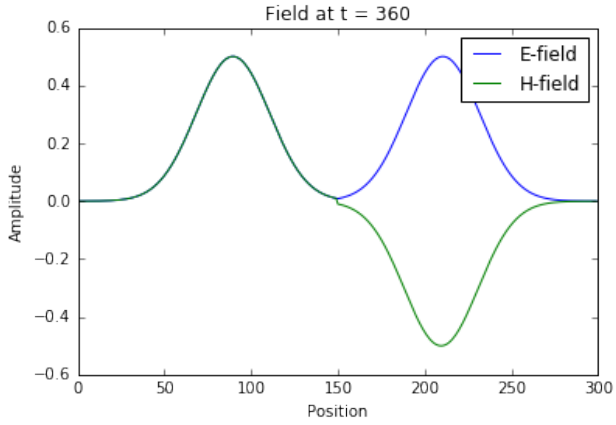

Field at t = 60
1e−28
E-field
H-field
Amplitude
Position
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
50
100
150
200
250
300

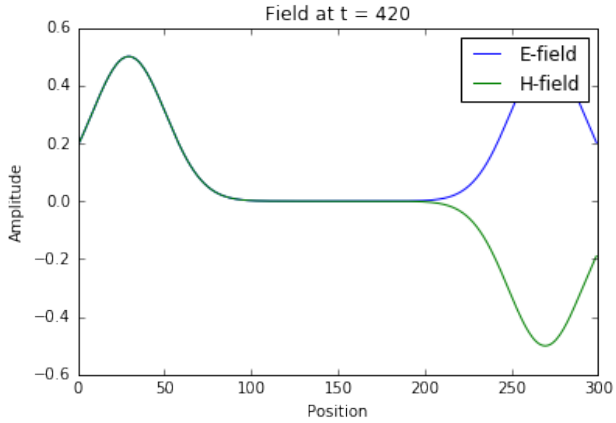

Field at t = 120
1e−16
E-field
H-field
Amplitude
Position
1.5
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250
300

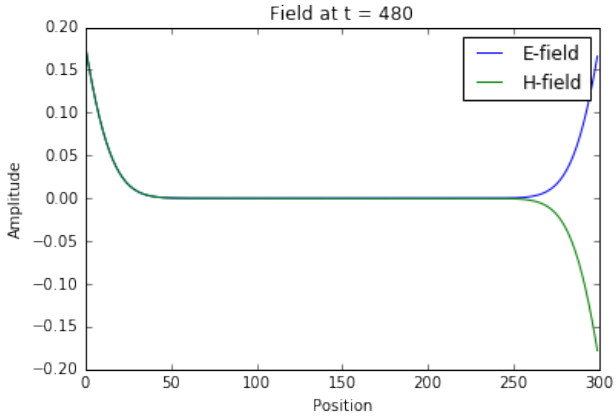

Field at t = 180
1e−7
E-field
H-field
Amplitude
Position
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
50
100
150
200
250
300

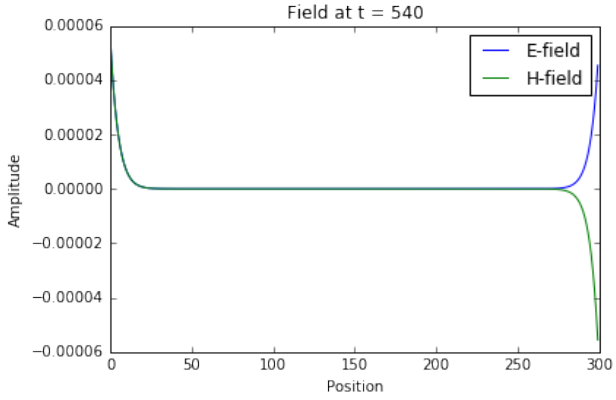

Field at t = 240
E-field
H-field
Amplitude
Position
0.010
0.005
0.000
−0.005
−0.010
0
50
100
150
200
250
300

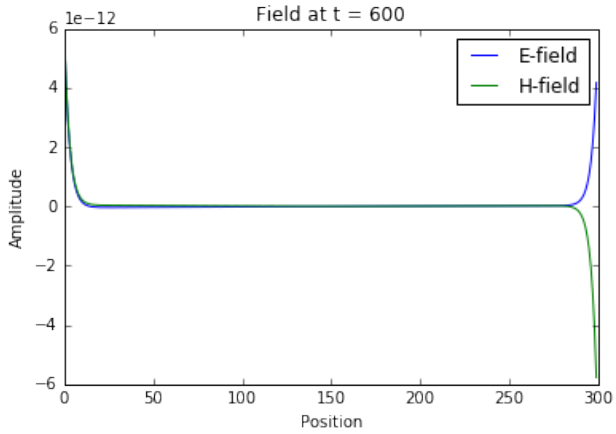

Field at t = 300
E-field
H-field
Amplitude
Position
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
50
100
150
200
250
300

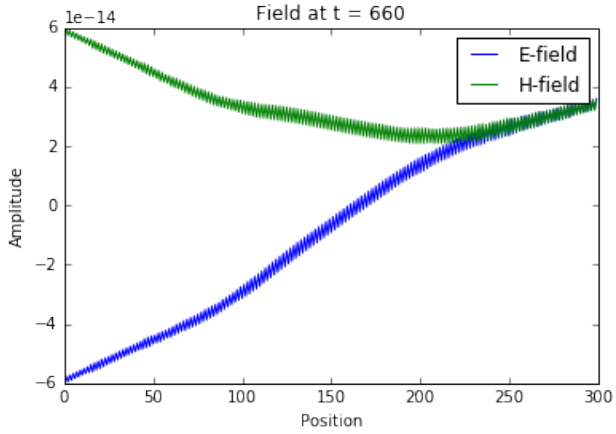

Field at t = 360
E-field
H-field
Amplitude
Position
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
50
100
150
200
250
300

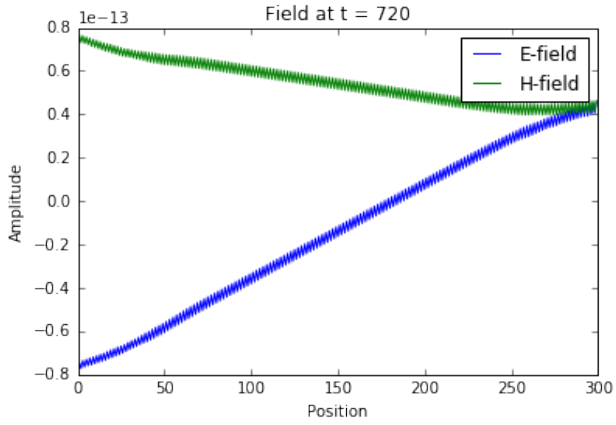

Field at t = 420
E-field
H-field
Amplitude
Position
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
50
100
150
200
250
300

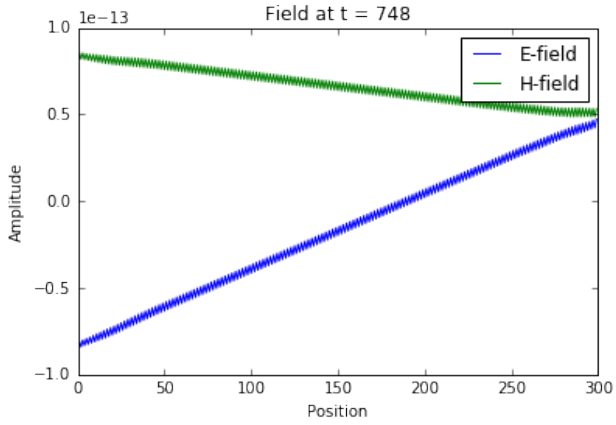

Field at t = 480
E-field
H-field
Amplitude
Position
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
0
50
100
150
200
250
300


Field at t = 540
E-field
H-field
Amplitude
Position
0.00006
0.00004
0.00002
0.00000
−0.00002
−0.00004
−0.00006
0
50
100
150
200
250
300


Field at t = 600
1e−12
E-field
H-field
Amplitude
Position
6
4
2
0
−2
−4
−6
0
50
100
150
200
250
300


Field at t = 660
1e−14
E-field
H-field
Amplitude
Position
6
4
2
0
−2
−4
−6
0
50
100
150
200
250
300





2 References

Taflove, A. & Hagness, S. C. Computational electrodynamics: the finite-difference time-domain method. (Artech House, 2005).

Engquist, B. & Majda, A. Absorbing boundary conditions for numerical simulation of waves. Proc Natl Acad Sci U S A 74, 1765–1766 (1977).