UNIVERSITY OF BIRMINGHAM

Finite-Difference Time-Domain Simulation

James Pallister

Issue: 1
Date: January 20, 2019

School of Physics and Astronomy
University of Birmingham
Birmingham, B15 2TT

Contents

1 Introduction

The Finite-Difference Time-Domain (FDTD) method is a popular technique for solving Maxwell's equations computationally and demonstrating how they evolve in time. This investigation looked into creating an FDTD simulation using the Yee algorithm. A 1D solver for Maxwell's equations was produced in Python, which took an input signal and plotted its propagation through space and time. This FDTD method was then further studied by looking at the propagation of signals through a material with a finite conductivity, otherwise known as a lossy material and then improved upon by introducing absorbing boundary conditions (ABC's) and a soft-wired source.

2 Background on Maxwell's equations

First of all, it is necessary to look at and understand Maxwell's equations in a vacuum, as they will prove crucial to the entire investigation:

$$\nabla \times \vec{E} = -\mu \frac{d\vec{H}}{dt} \tag{1}$$

$$\nabla \times \vec{H} = \epsilon \frac{d\vec{E}}{dt} \tag{2}$$

These equations relate the spatial propagation of Electric (E) and Magnetic (H) fields to their temporal movement. The cross product and nabla effectively are differentiating with respect to space and the dt on the right hand side the of the equations is a differential with respect to time. (The μ is just a constant and is the permeability of free space whilst the ϵ is another constant and is the permittivity of free space). Of note is how E and H appear in both equations (1) and (2). This indicates that the change in one field has an effect on the other.

Already it is clear that given a set of initial conditions (or an input signal, say), these equations when solved can give the Electric and Magnetic field values of a wave at any point in time or space. This is the essence of the FDTD Method.

Equations (1) and (2) are a representation of Maxwell's equations in 3D which can be a very complicated thing to solve. Therefore it is useful to reduce these equations to 1D, which is easier to solve. This involves choosing just one out of three of the x,y and z components of the magnetic and electric fields and the chosen components are described in Figure 1.

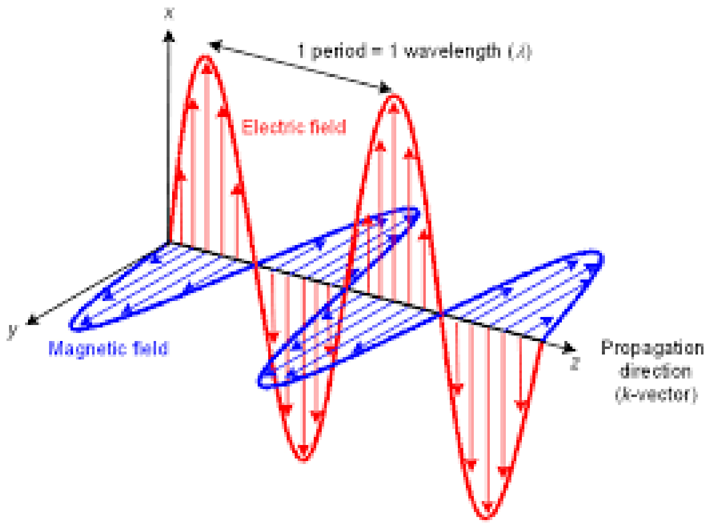


Figure 1: A diagram showing the propagation of electric and magnetic fields. The two fields are always at right angles to one another and the direction of propagation. In this investigation the orientation of fields in this picture will be used, i.e the x component of the electric field and y component of the magnetic field. Note this choice is arbitrary and it makes no difference physically [1].

With this established equations (1) and (2) can be reduced to 1D by neglecting the E_y, E_z and H_x, H_z components (here i, j and k are the usual unit vectors in the x,y and z directions respectively):

$$\begin{aligned}
-\mu\frac{d\vec{H}}{dt} &= \nabla\times\vec{E} \\
\Rightarrow -\mu j\frac{\partial H_y}{\partial t} &= \begin{vmatrix} i & j & k \\ 0 & 0 & \frac{\partial}{\partial z} \\ E_x & 0 & 0 \end{vmatrix} = j\frac{\partial E_x}{\partial z} \\
\Rightarrow -\mu\frac{\partial H_y}{\partial t} &= \frac{\partial E_x}{\partial z} \qquad (3)
\end{aligned}$$

$$\begin{aligned}
\epsilon\frac{d\vec{E}}{dt} &= \nabla\times\vec{H} \\
\Rightarrow i\epsilon\frac{\partial E_x}{\partial t} &= \begin{vmatrix} i & j & k \\ 0 & 0 & \frac{\partial}{\partial z} \\ 0 & H_y & 0 \end{vmatrix} = -i\frac{\partial H_y}{\partial z} \\
\Rightarrow \epsilon\frac{\partial E_x}{\partial t} &= -\frac{\partial H_y}{\partial z} \qquad (4)
\end{aligned}$$

3 Understanding the Yee algorithm

3.1 Central finite-difference (cfd) approximation

As stated in the previous section, a change in one of the E or H fields causes a change in the other. More specifically the change in the E-field with time is dependent on the change in the H-field in space. Thus a slight increment in time of the E-field at a given point in space, depends on the current values of the E and H fields at that point. This is often talked of as ”Leap-Frogging” as one ’hops’ from the E-field to the H-field via equation (4) and then from the H to the E by (3) and then from E to H once more as the process repeats itself, going on through time [2].

To implement this formally, Yee's algorithm uses a central finite-difference approximation.

Consider the Taylor's expansions for some function $f(x)$ about a point x_0 with offsets $\pm\frac{k}{2}$ [3]:

$$\begin{aligned} f(x_0+\frac{k}{2}) &= f(x_0)+\frac{k}{2}f'(x_0)+\frac{1}{2!}(\frac{k}{2})^2f''(x_0)+\frac{1}{3!}(\frac{k}{2})^3f'''(x_0)+\dots \\ f(x_0-\frac{k}{2}) &= f(x_0)-\frac{k}{2}f'(x_0)+\frac{1}{2!}(\frac{k}{2})^2f''(x_0)-\frac{1}{3!}(\frac{k}{2})^3f'''(x_0)+\dots \end{aligned}$$

Taking (5) - (6) :

$$f(x_0+\frac{k}{2})-f(x_0-\frac{k}{2}) = kf'(x_0)+\frac{2}{3!}(\frac{k}{2})^3f'''(x_0)+\dots$$

Divide by k:

$$\frac{f(x_0+\frac{k}{2})-f(x_0-\frac{k}{2})}{k} = f'(x_0)+\frac{2}{3!2^3}(k)^2f'''(x_0)+\dots$$

Finally rearrange for an expression of $f'(x_0)$:

$$f'(x_0) = \frac{f(x_0+\frac{k}{2})-f(x_0-\frac{k}{2})}{k}+O(k^2) \tag{5}$$

Taking 'k' to be very small one arrives at:

$$f'(x_0) = \frac{f(x_0+\frac{k}{2})-f(x_0-\frac{k}{2})}{k} \tag{6}$$

This is the central finite-difference approximation wherein the slope of a function is evaluated at some central point, by first working out the value of the function at some finite difference away from it. This is most clearly seen in Figure 2.

Figure 2 also shows two other finite-difference approximations, forward and backward. It is worth showing why these two methods are not as desirable a choice to use for Yee's algorithm as the central one.

From figure 2, the forward difference is given by:

$$\begin{aligned} f(x_0+k) &= f(x_0)+kf'(x_0)+\frac{1}{2!}(k)^2f''(x_0)+\frac{1}{3!}(k)^3f'''(x_0)+\dots \\ \Rightarrow f'(x_0) &= \frac{f(x_0+k)-f(x_0)}{k}+O(k) \end{aligned} \tag{7}$$

And the backward difference is given by:

$$\begin{aligned} f(x_0-k) &= f(x_0) - kf'(x_0) + \frac{1}{2!}(k)^2 f''(x_0) - \frac{1}{3!}(k)^3 f'''(x_0) + \ldots \\ \Rightarrow f'(x_0) &= \frac{f(x_0) - f(x_0+k)}{k} + O(k) \end{aligned} \tag{8}$$

By comparing (7) and (8) to (5) it can be seen that the order of the error approximated with (5) is proportional to k^2 whereas for (7) and (8) its proportional to k. If k is taken to be smaller, the error associated with (5) reduces quadratically and not linearly like with (7) and (8). Thus the central finite-difference is the more desirable approximation to undertake as there is less error associated with it at smaller 'k'.

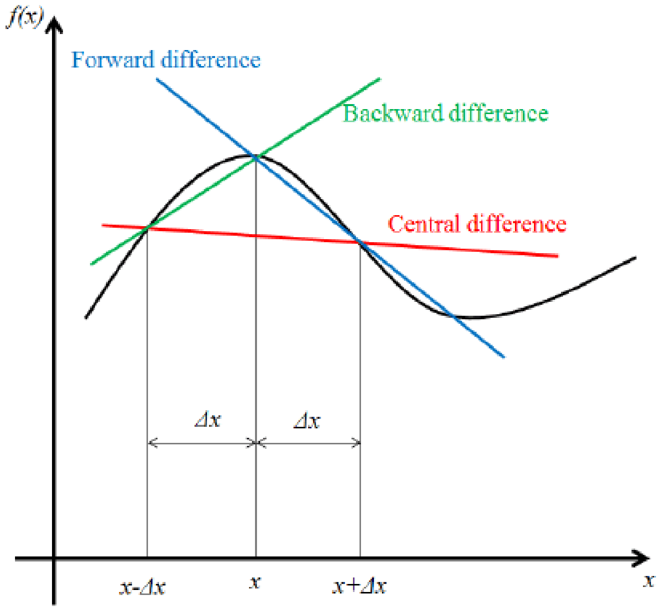


Figure 2: Depiction of the central finite-difference method. Also shown is the forward difference and backward difference both of which can be seen to be less accurate an approximation to the slope of the curve at 'x' [4].

3.2 Yee's algorithm and the cfd approximation

Returning to equations (3) and (4) it is apparent that the cfd approximation can be applied to both the space and time derivatives of the H and E field components. As to how exactly to do this, this is where Yee's algorithm comes in.

Starting with the LHS of (4), the time derivative can be approximated at some $t = n\Delta t$ and location, k using the cfd approximation. Here $n\Delta t$ is just the nth time-step in an EM wave's propagation.

$$\epsilon\frac{\partial E_x}{\partial t} \approx \epsilon\frac{E_x^{n+\frac{1}{2}}(k) - E_x^{n-\frac{1}{2}}(k)}{\Delta t} \tag{9}$$

(Note the superscripts in this equation which are using the cfd approximation by evaluating the E-field at a time step of $(n-\frac{1}{2})\Delta t$ and $(n+\frac{1}{2})\Delta t$).

In a similar vein the RHS of (4) can be approximated as follows:

$$-\frac{\partial H_y}{\partial z} \approx -\frac{H_y^n(k+\frac{1}{2}) - H_y^n(k-\frac{1}{2})}{\Delta z} \tag{10}$$

In this instance, as the differential is in space, the time stays fixed at $n\Delta t$ and it's the value of the H-field in space that is approximated via the cfd approximation.

Turning our sights to (3), the LHS can be depicted as:

$$-\mu\frac{\partial H_y}{\partial t} \approx -\mu\frac{H_y^{n+1}(k+\frac{1}{2}) - H_y^n(k+\frac{1}{2})}{\Delta t} \tag{11}$$

Just like in (9), space is kept fixed at $k+\frac{1}{2}$ whilst the time is evaluated using a cfd approximation. In this instance though the time has advanced a half interval ($\frac{n}{2}$) along since (9) and so is being evaluated at $n+\frac{1}{2}$ as opposed to at n.

Finally the RHS of (3) can be represented as:

$$\frac{\partial E_x}{\partial z} \approx \frac{E_x^{n+\frac{1}{2}}(k+1) - E_x^{n+\frac{1}{2}}(k)}{\Delta z} \tag{12}$$

Where time is fixed and space is being evolved.

Putting (9), (10), (11) and (12) together in the same form as (3) and (4) one arrives at:

$$\epsilon\frac{E_x^{n+\frac{1}{2}}(k) - E_x^{n-\frac{1}{2}}(k)}{\Delta t} = -\frac{H_y^n(k+\frac{1}{2}) - H_y^n(k-\frac{1}{2})}{\Delta z} \tag{13}$$

$$-\mu\frac{H_y^{n+1}(k+\frac{1}{2}) - H_y^n(k+\frac{1}{2})}{\Delta t} = \frac{E_x^{n+\frac{1}{2}}(k+1) - E_x^{n+\frac{1}{2}}(k)}{\Delta z} \tag{14}$$

This can now be rearranged into an 'update' form which is more conducive to computer coding:

$$\begin{aligned} E_x^{n+\frac{1}{2}}(k) &= E_x^{n-\frac{1}{2}}(k) + \frac{\Delta t}{\epsilon\Delta z}(H_y^n(k-\frac{1}{2}) - H_y^n(k+\frac{1}{2})) \\ H_y^{n+1}(k+\frac{1}{2}) &= H_y^n(k+\frac{1}{2}) + \frac{\Delta t}{\mu\Delta z}(E_x^{n+\frac{1}{2}}(k) - E_x^{n+\frac{1}{2}}(k+1)) \end{aligned}$$

For reasons discussed in section 3.3, it is easier and more convenient to introduce a scaling to the E-field like:

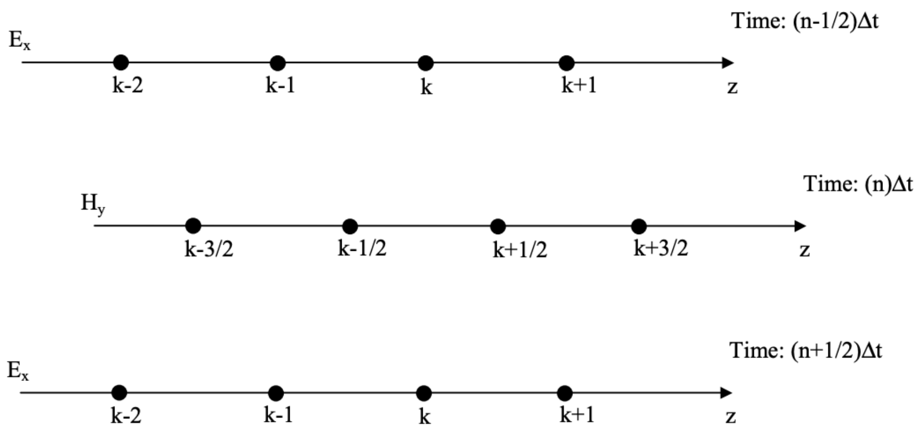
$$E_{new} = \sqrt{\frac{\epsilon}{\mu}}E$$
$$\frac{1}{\sqrt{\epsilon\mu}} = c$$

Producing:

$$E_x^{n+\frac{1}{2}}(k) = E_x^{n-\frac{1}{2}}(k) + \frac{c\Delta t}{\Delta z}(H_y^n(k-\frac{1}{2}) - H_y^n(k+\frac{1}{2})) \quad (15)$$
$$H_y^{n+1}(k+\frac{1}{2}) = H_y^n(k+\frac{1}{2}) + \frac{c\Delta t}{\Delta z}(E_x^{n+\frac{1}{2}}(k) - E_x^{n+\frac{1}{2}}(k+1)) \quad (16)$$

This is what the Yee algorithm is then:
-Replace all the derivatives in Maxwell's equations with central finite difference approximations to discretize space and time.
-Formalize 'update' equations from what results, wherein the value of 'future' fields are calculated from those in the 'past'.
-Use the update equations to calculate the 'future' electric and magnetic fields now making them 'past' fields.
-Use the 'past' fields to calculate 'future' fields at some later time and keep repeating this process for the desired duration.

Figure 3 depicts the Yee algorithm more clearly.



depiction.png

Figure 3: Depiction of the Yee algorithm. Every dot on the picture represents a value of the field that has to be calculated in order to work out equations (13) and (14). [5]

3.3 Stability of 1D FDTD

Most numerical techniques such as FDTD have stability criteria which must be fulfilled to avoid nonphysical results.

The stability of a 1D FDTD method is determined by the Courant–Friedrichs–Lewy condition [6]. In essence this condition states that the bounds of the numerical domain, i.e. what time-step (Δt) and space-step (Δz) are being used, have to bound the physical domain (the wave's speed, which for EM waves is the speed of light, c). Formally this can be stated as:

$$\frac{c\Delta t}{\Delta z} \leq 1 \tag{17}$$

If the wave is moving faster than the steps its being evaluated at, it can't be evaluated accurately and thus nonphysical results are obtained.

If (17), known as the Courant number, is taken to be exactly equal to 1, this greatly simplifies the coefficients on the RHS of (15) and (16). This is why the scaling in 3.2. was choosen so that after imposing this condition what results would be in its simplest form.

3.4 Implementation of 1D FDTD in Python script

Trying to implement the Yee algorithm straight into Python raised a number of issues which had to be addressed before any useful results could be obtained.

Firstly, the Yee algorithm relies upon a series of adjacent nodes (figure 3) in the update equations but at some point these nodes have to stop, otherwise the simulation would run forever to calculate an infinite amount of data points. But if the nodes are terminated this means that nodes on the boundaries won't have two neighbours meaning the update equations can't be used. Consider for example a domain of width 200 space-steps. If the update equations start with the electric field at space-step zero then the program will immediately fail as E[0] relies upon H[$-\frac{1}{2}$] and this is not defined - it is outside of the available domain. Likewise at the other end the H field at space-step 199 has no E field at space-step 200 to update from. (The index of the last value in any array in Python is always one less than its size)

This problem was solved by just fixing values of the boundary nodes and not updating them. For example, the node at E[0], i.e. the value of the E-field at the very left edge of the domain was set to a Gaussian pulse, which was allowed to propagate through the simulation. Likewise the H field node at space-step 199 was just set to 0.

Secondly, arrays in Python use integer indexes not half integer's. Inputting $H_y^n(k+\frac{1}{2})$ for example would be nonsensical. The way around this problem is to subtract $\frac{1}{2}$ of an index away from any non integer indexes in equations (13) and (14), effectively rounding them. Remembering that the Courant number was set to equal 1 this gives:

$$\begin{aligned} E_x^n(k) &= E_x^{n-1}(k) + (H_y^n(k-1) - H_y^n(k+1)) \quad (18)\\ H_y^{n+1}(k) &= H_y^n(k) + (E_x^{n+1}(k) - E_x^{n+1}(k+1)) \quad (19)\end{aligned}$$

Finally to test whether the program gives the correct results, it has to be checked against some simple physical phenomenon, such as the incidence of a wave upon a dielectric medium boundary. The only alteration this makes to the equations derived so far is a division by an impedance term in (18) [7]:

$$\begin{aligned} E_x^n(k) &= E_x^{n-1}(k) + (H_y^n(k-1) - H_y^n(k+1))/imp && (20) \\ H_y^{n+1}(k) &= H_y^n(k) + (E_x^{n+1}(k) - E_x^{n+1}(k+1)) && (21) \end{aligned}$$

This brings the equations to their final form, which will be used for coding.

At this stage it is also useful to introduce Fresnel's equations to double check the behaviour of the system at the dielectric medium boundary [8] (Fresnel's equations differ depending on the polarization of the incident field, but in 1D with a zero degree angle of incidence this doesn't matter):

Ratio of the reflected and incident electric fields -

$$\frac{E_r}{E_i} = \frac{n_1 cos(\theta_i) - n_2 cos(\theta_t)}{n_1 cos(\theta_i) + n_2 cos(\theta_t)}$$

Ratio of the transmitted and incident electric fields -

$$\frac{E_t}{E_i} = \frac{2n_1 cos(\theta_i)}{n_1 cos(\theta_i) + n_2 cos(\theta_t)}$$

with n_1 and n_2 being the refractive indices of the mediums on either side of the boundary and θ_i and θ_t being the angles of incidence and transmission. As the first medium is free space, n_1 is equal to 1 and for the purposes of simplifying the maths, the second medium has n_2 set equal to 3. Taking the angles of incidence and reflection as zero degrees the equations simplify to:

$$\begin{aligned} \frac{E_r}{E_i} &= \frac{n_1 - n_2}{n_1 + n_2} = -\frac{1}{2} && (22) \\ \frac{E_t}{E_i} &= \frac{2n_1}{n_1 + n_2} = \frac{1}{2} && (23) \end{aligned}$$

i.e. the reflected and transmitted waves should have amplitudes that are half of that incident on the boundary, and the reflected wave should change sign. If the program does not produce these results we know it must be wrong.

4 Discussion of Results

4.1 Dielectric Medium results

Initially the program was tested in free space with a simple Gaussian pulse initializing the E-field of form:

$$E[0] = \exp\left[-(t-30)*(t-30)/100\right] \tag{24}$$

Figure 4 shows the results of this. The pulse has propagated from its initial position until it reached the 50th space-step at time-step 50 (remember, the wave's speed was set to 1 space-step per time-step). The output for E[50] then subsequently follows the form of the Gaussian pulse which was introduced at E[0]. This is a good indicator that the program is working.

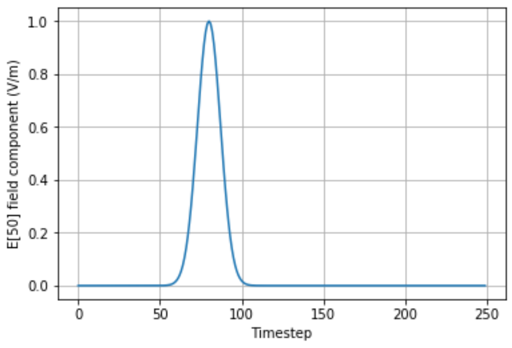
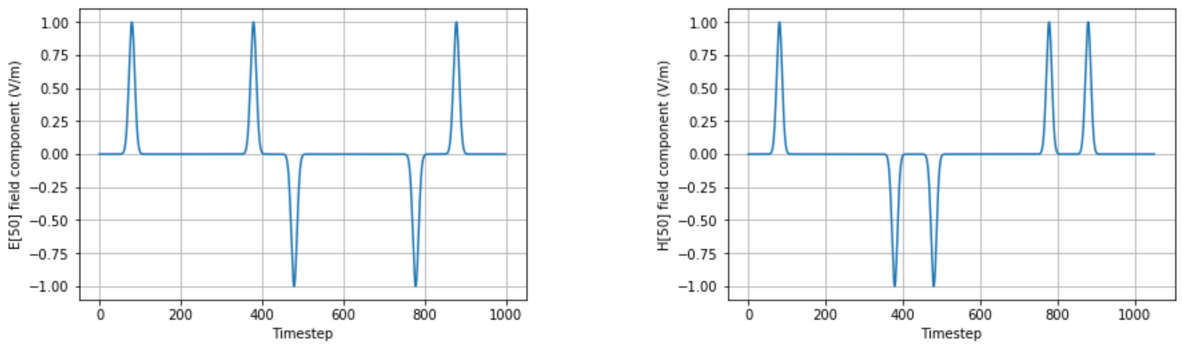


Figure 4: Output from 1D FDTD solver for a very simple Gaussian pulse. The pulse begins at time-step 50 and peaks at time-step 80 which is in accordance with the form of (24). The program was terminated at time-step 250.

The next step was to run the program over a longer time domain in order to see its behaviour upon hitting the ends of the space domain (figure 5). The first peak in both graphs is the pulse from figure 4. The second peak is then the pulse having been reflected from the right most boundary. It has been stated that the right most boundary has been set as $H[199] = 0$. To ensure the magnetic field goes to zero at this point the reflected magnetic field must have equal magnitude but opposite sign to the incident one. Hence the form of the second peak in fig 5)b) (this is known as a PMC or Perfect Magnetic Conductor boundary [3]). The electric field's only constraint is that the value of the field at the boundary must be equal to what is reflected, so it doesn't change sign in fig 5)a), it just reflects. Moving our attention to the third peak, this is the peak produced when the pulse reflects off of the left side of the domain, i.e. the point where it started. Although the value of E[0] was set to the Gaussian pulse in question, by this time its amplitude has basically gone to zero. This means there is a boundary condition imposed on the electric field that means the total field here must be zero (just like with the magnetic field at the right boundary). This produces the first inverted peak in figure 5)a) (this is known as a PEC or Perfect Electric Conductor boundary). The magnetic field at this point only has to satisfy the total magnetic field coming in equalling that going out and so is reflected but not inverted. By continuing this logic further it is possible to account for all the peaks in figures 5)a) and b)

With the program running as expected it's time to test it against an actual physical phenomenon such as incidence on a dielectric boundary (figure 6).

The pulse strikes the boundary at space-step 100 and is partially reflected and transmitted. As is clear the amplitudes of the reflected and transmitted parts of the wave are both equal to a half with the reflected wave having flipped sign. This passes the criteria stated in equations (22) and (23) and so it can safely be concluded that the program has worked correctly.



(a) E[50] field value output

(b) H[50] field value output

Figure 5: Output from 1D FDTD solver for the E and H field components of a Gaussian pulse, this time allowed to run for 1000 time steps meaning the pulse has encountered the edge of the domain and has been reflected.

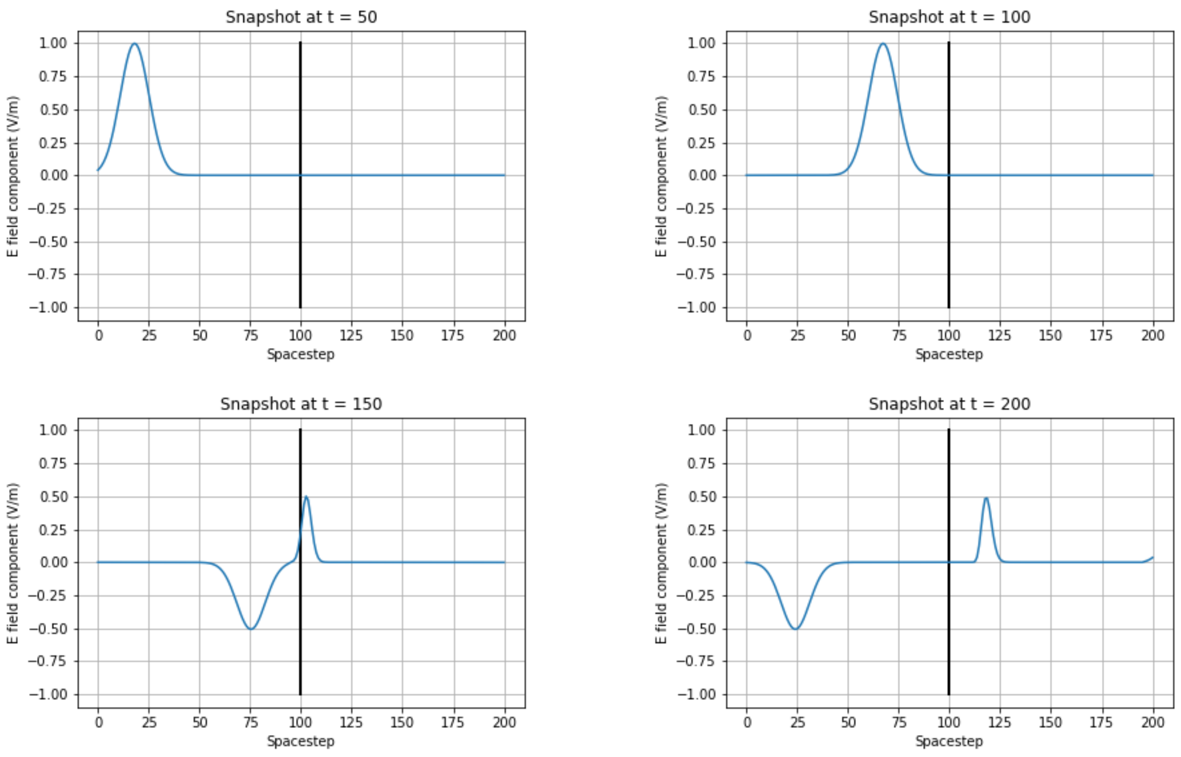


Figure 6: Multiple snapshots taken at different time-steps that show the wave propagating and then hitting a dielectric medium at space-step 100. The wave is then partially reflected and transmitted and the reflected and transmitted parts satisfy Fresnel's equations, which is what was to be expected.

All of this can be represented on a waterfall plot which is a plot of all the snapshots at different times on one graph with a slight offset (in this case vertical) from one another (figure 7). The pulse travels to the right and then hits the dielectric boundary at space-step 100. It is reflected and transmitted causing two diagonal 'lines' to form akin to a wavefront. The pulses continue to propagate though the clarity of the image decreases as multiple 'wavefronts' are formed and interfere with one another.

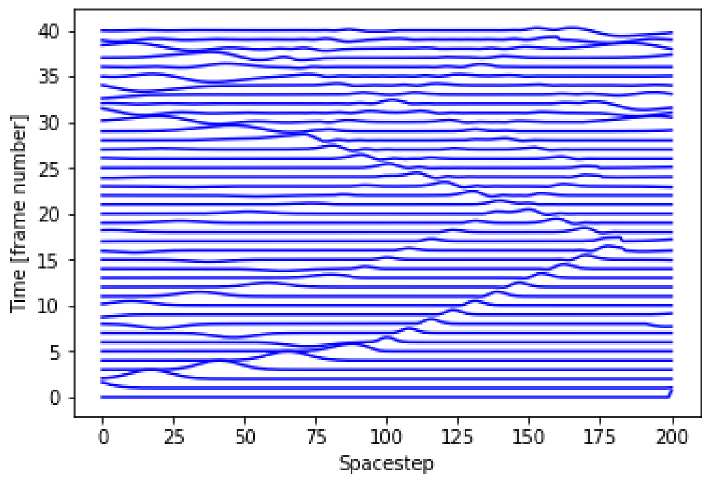


Figure 7: Waterfall plot for a Gaussian pulse encountering a dielectric boundary at space-step 100.

4.2 Lossy Dielectric Medium Results

Having looked at a dielectric medium it's time to consider a lossy dielectric medium. A lossy dielectric medium is one with a finite conductivity that attenuates any wave in the medium.

The difference in Maxwell's equations is an additional term on the LHS of equation (4) [3]:

$$\sigma E_x + \epsilon \frac{\partial E_x}{\partial t} = -\frac{\partial H_y}{\partial z}$$

The non-derived E_x term on the LHS can be approximated as an average in time:

$$E_x^n[k] \approx \frac{E_x^{n+\frac{1}{2}}[k] + E_x^{n-\frac{1}{2}}[k]}{2}$$

Combining this with the CFD approximations already provided gives:

$$\sigma \frac{E_x^{n+\frac{1}{2}}[k] + E_x^{n-\frac{1}{2}}[k]}{2} + \epsilon \frac{E_x^{n+\frac{1}{2}}(k) - E_x^{n-\frac{1}{2}}(k)}{\Delta t} = -\frac{H_y^n(k+\frac{1}{2}) - H_y^n(k-\frac{1}{2})}{\Delta z}$$

and then rearranging this and taking the E-scaling previously stated, with the Courant number set to 1 and the rounding of the k values we arrive at:

$$E_x^n(k) = \frac{1 - \frac{\sigma \Delta t}{2\epsilon}}{1 + \frac{\sigma \Delta t}{2\epsilon}} E_x^{n-1}(k) + \frac{1}{1 + \frac{\sigma \Delta t}{2\epsilon}} (H_y^n(k-1) - H_y^n(k+1))/imp \tag{25}$$

The expression $\frac{\sigma \Delta t}{2\epsilon}$ is known as the loss and for the purposes of creating a clear simulation (i.e. not one in which the wave attenuates too fast to see) this was set at the rather small value of 0.01. Also note that if σ is zero then (25) reduces to (20)

Figure 8 shows the attenuation of a Gaussian pulse after it hits the boundary between free space and a lossy dielectric. Unlike with the standard dielectric the pulse's amplitude gradually attenuates with time, exactly as expected.

As well as this, just like before, a waterfall plot can be produced that more clearly shows the attenuation of the pulse. The 'wavefronts' to the right of space-step 100 attenuate and fall off in magnitude, which was expected.

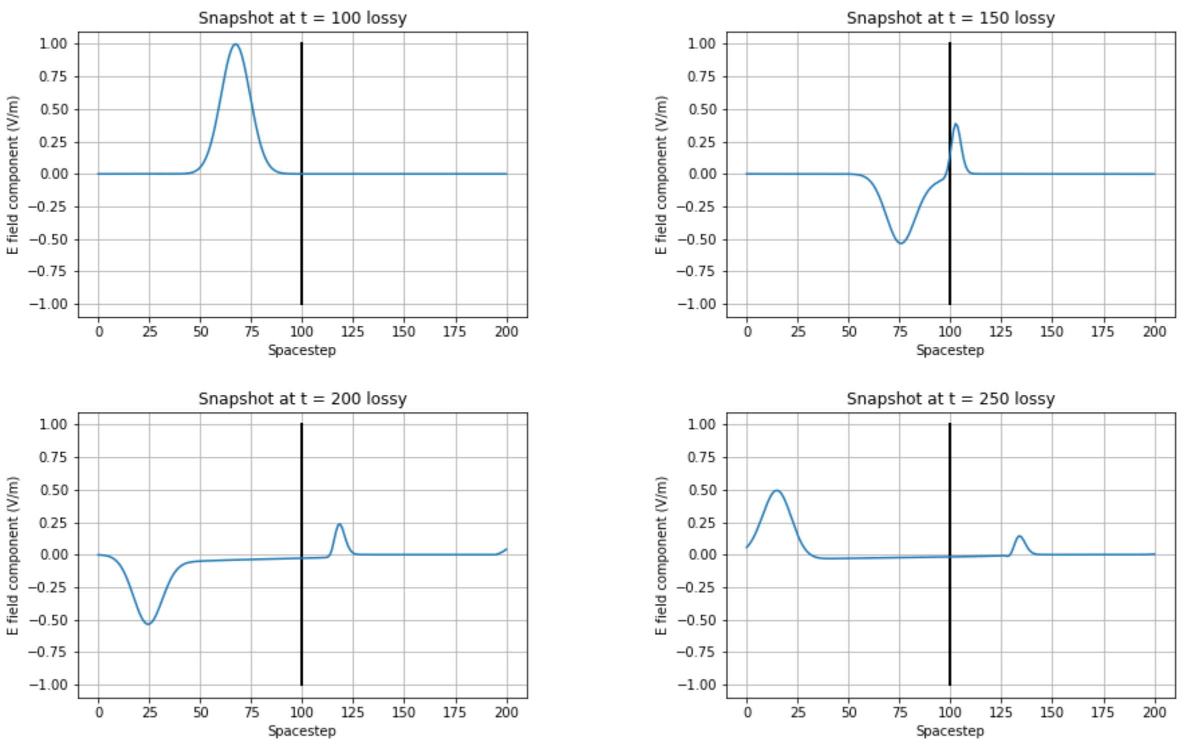


Figure 8: Multiple snapshots taken at different time steps that show the pulse propagating and then hitting a lossy dielectric medium at space-step 100. The wave is then partially reflected and transmitted and the transmitted wave is slowly attenuated over time.

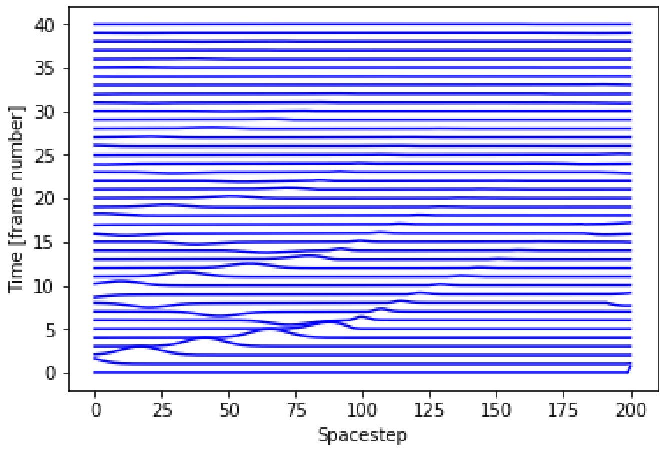


Figure 9: Waterfall plot for a Gaussian pulse encountering a lossy dielectric medium at space-step 100.

5 Absorbing boundary conditions implementation and improvements

A couple of issues not discussed so far have been the hard-wiring of the source and the breakdown of clarity as time progresses.

In all the programs ran so far the source has been hardwired, this means it has been introduced at E[0] and then E[0] has been set to effectively zero from then on. This is an issue as no energy can pass through this node. To rectify this, an additive (soft-wired) source can be used. This is merely the same Gaussian pulse but added to the current field value at an arbitrary point in space, say E[50]. This will produce two pulses, one left moving and one right moving.

The issue now encountered is that these two pulses are going to continue reflecting off of the edges of the domain and the dielectric boundary for the duration of the program and this is not a very physical process. The program being written is to solve Maxwell's equations in 1D, not solve them in 1D in a confined space. Therefore there also needs to be someway to allow the fields to 'extend' to infinity space-steps and not interfere with the rest of the system. This is implemented using absorbing boundary conditions (ABC) [3]. Effectively the pulse is absorbed into the boundary and so no longer continues to reflect. A simple way to do this is to set E[0] = E[1] and H[199] = H[198], i.e. the very edges of the domain are updated by the value of the field just before it reaches the edge and not by its reflection. This means the pulse effectively 'carries on' through the end of the domain and so disappears from the system.

Both these aspects are illustrated below, along with the 'messiness' encountered by not having ABC's. Even with ABC's there is still a small pulse fluctuation left when there shouldn't be due to minute imprecisions introduced by the CFD approximations made.

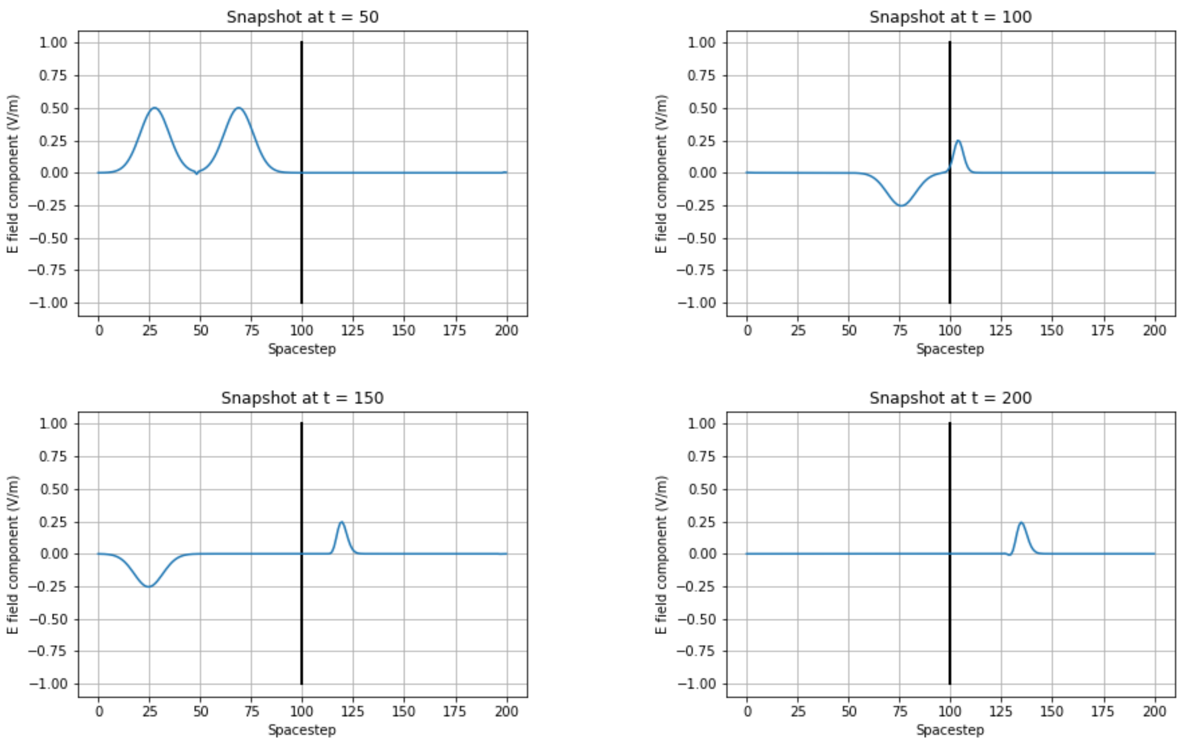
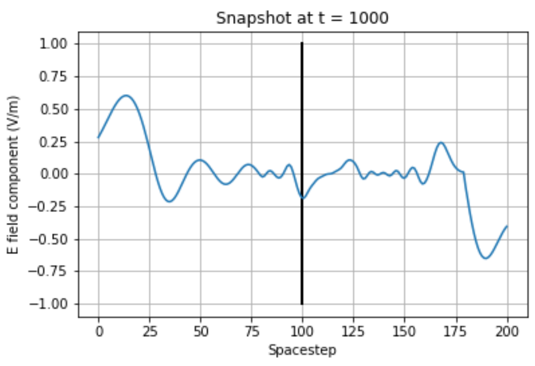
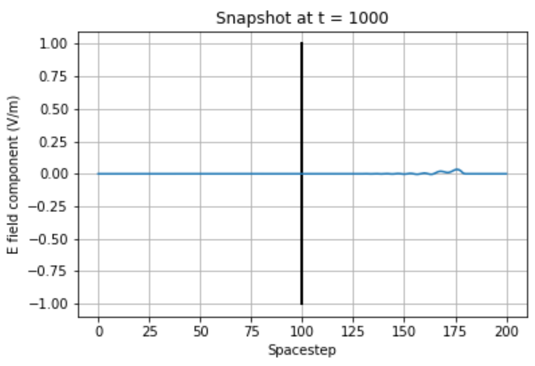


Figure 10: Multiple snapshots of an additive source, the Gaussian pulse, propagating to the left and right and then being absorbed at the edges. The snapshot at t = 100 only has two peaks from the reflected and transmitted parts of the right travelling pulse in t = 50, there are no peaks from the left travelling pulse as this has 'disappeared' from the system and allowed to continue to infinity. Likewise the left pulse at t = 150 was absorbed and so does not appear in graph t = 200.



(a) Before ABC introduction.



(b) After ABC introduction.

Figure 11: Graph showing the 'messiness' that the program produced before the introduction of ABC's and what it looks like after their introduction. There is a small fluctuation in the right-most part of b).

6 Conclusion

In corollary the FDTD method has been successfully programmed in Python. The program used the Yee algorithm and produced results that obeyed physical phenomena such as a wave's incidence on a dielectric boundary. Investigations were then taken further by looking at the propagation of a wave pulse in a lossy dielectric medium and this was once again produced successfully. Finally, issues such as the hard-wiring of the wave source and the wave not being allowed to propagate to infinity were addressed successfully and an even more proficient program was constructed than before.

References

[1] Electric and Magnetic Field propagation picture
`http://iopscience.iop.org/chapter/978-1-6817-4613-5/bk978-1-6817-4613-5ch1.pdf`

[2] 'Leap-frogging' description
`https://en.wikipedia.org/wiki/Finite-difference_time-domain_method`

[3] Understanding the Finite-Difference Time-Domain Method, John B. Schneider, 2010. chap3 - Taylor's expansion, PMC and PEC boundary, lossy dielectric medium formation and ABC boundary
`https://www.eecs.wsu.edu/~schneidj/ufdtd/chap3.pdf`

[4] Central finite-difference approximation illustation
`https://www.researchgate.net/figure/Different-geometric-interpretations-of-the-first-order-finite-difference-approximation_fig3_280700552`

[5] Yee's Algorithm Depiction
`http://www.ece.utah.edu/~ece6340/LECTURES/lecture%2014/FDTD.pdf`

[6] Courant–Friedrichs–Lewy condition
R. Courant, K. Friedrichs and H. Lewy, "On the Partial Difference Equations of Mathematical Physics," in IBM Journal of Research and Development, vol. 11, no. 2, pp. 215-234, March 1967. doi: 10.1147/rd.112.0215
`https://ieeexplore.ieee.org/document/5391985`

[7] Propagation in a dielectric medium
`http://farside.ph.utexas.edu/teaching/em/lectures/node98.html`

[8] Fresnel's equations
`http://epweb2.ph.bham.ac.uk/user/lazzeroni/EM2_2018/Lecture14_EM2.pdf`