Jurnal Politeknik Caltex Riau
Terbit Online pada laman https://jurnal.pcr.ac.id/index.php/jkt/
| e- ISSN : 2460-5255 (Online) | p- ISSN : 2443-4159 (Print) |

Implementation of Parallel Programming in One-Dimensional Electromagnetic Simulation with the FDTD Method

Edmond Febrinicko Armay[*†1], **Heni Rachmawati**[2]

[1*]UIN Sultan Syarif Kasim Riau, email: nicko@uin-suska.ac.id
[†]Lund University
[2]Polytechnic of Caltex Riau, email: henni@pcr.ac.id

Abstract

Simulation of electromagnetic in 1D has been done using FDTD method. The simulation is conducted on some scenario, starting from one-dimensional free space, absorbing boundary condition in one dimension, propagation in a dielectric medium, simulating different sources, and propagation in a lossy dielectric medium. To accelerate calculation process, parallel programming technique is used on looping parts of simulation. We get significant increase for speedup between 10 – 50 times.

Keywords: *parallel; 1D; electromagnetic; FDTD; speedup*

1. Introduction

Computational electromagnetics (CEM) is the process of modeling in the interaction of electromagnetic fields with physical objects and the environment. It typically involves using computationally efficient approximations to Maxwell's equation and is used to calculate antenna performance, electromagnetic compatibility, radar cross section, and, in this paper, electromagnetic wave propagation [1]. In modern electromagnetic solution, one approach is to discretize the space in terms of grids (both orthogonal and non-orthogonal) and solving Maxwell's equations at each point in the grid, as shown in Fig. 1.

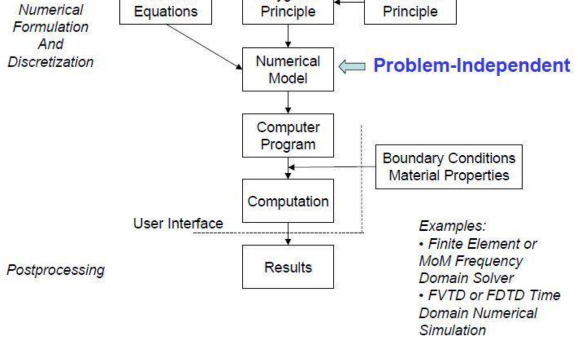

Numerical Formulation And Discretization
Maxwell's Equations
Huygens' Principle
Variational Principle
Numerical Model
Problem-Independent
Computer Program
Boundary Conditions Material Properties
Computation
User Interface
Examples:
• Finite Element or MoM Frequency Domain Solver
• FVTD or FDTD Time Domain Numerical Simulation
Postprocessing
Results

Figure 1. Modern electromagnetic solution [2]

Dokumen diterima pada 24 April, 2022
Dipublikasikan pada 31 Mei, 2022

Discretization consumes computer memory, and solving the equations takes significant time. Large scale CEM problems face memory and CPU limitations. As of 2007, CEM problems require supercomputers, high performance clusters, vector processors and/or parallel computer [2]. Choosing the right technique for solving a problem in CEM is important, as choosing the wrong one can either result in incorrect results, or results which take excessively long to compute. The comparison of three popular methods as shown in Table I.

Table 1. Comparison of cem methods [2]

	Method of Moments	**Finite Element Method**	**Finite Difference Time Domain**
Discretization	Only wires or surfaces	Entire domain (tetrahedron)	Entire domain (cube)
Solution method	Frequency domain, linear equations, full matrix	Frequency domain, linear equations, sparse matrix	Time domain, iterations
Boundary conditions	No need for special boundary conditions	Absorbing boundary conditions	Absorbing boundary conditions
Numerical effort	~N^3	~N^2	~N
Well suited for:	Wire antennas, metal surfaces, coupling between distant elements, arbitrary shapes of surfaces, single or few frequencies	Arbitrary shapes and materials, single or few frequencies	Preferably orthogonal planar boundaries, arbitrary materials, broadband investigations
Not so well suited for:	Various materials, electrically very large structures, broadband investigations	Electrically large structures, coupling between distant elements, broadband investigations	Coupling between distant elements, high-Q structures

Finite difference time domain (FDTD) is a popular CEM technique. It is easy to understand. It has an exceptionally simple implementation for a full wave solver. It is at least an order of magnitude less work to implement a basic FDTD solver than either Finite Element Method (FEM) or Method of Moments (MoM) solver. FDTD is the only technique where one person can realistically implement oneself in a reasonable time frame, but even then, this will be for a quite specific problem [2].

2. Research Methodology

A. Deriving Formulations for One-Dimensional Electromagnetic Simulation

In one-dimensional free space, the equations of a plane wave with the electric field oriented in the x direction, and the magnetic field oriented in the y direction are [3]

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

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

Where,

$$\frac{1}{\sqrt{\varepsilon_0\mu_0}}\frac{\Delta t}{\Delta x} = \frac{1}{2} \quad (3)$$

In the first two equations, time is specified by the superscripts, i.e., “n” actually means a time t=Δt.n. The terms in parentheses represent distance, i.e., “k” actually means the distance z=Δx.k. Rewriting (1) and (2) in C computer code gives the following:

ex[k]=ex[k]+0.5*(hy[k-1]-hy[k]) (4)

hy[k]=hy[k]+0.5*(ex[k]-ex[k+1]) (5)

note that the n or n+1/2 or n-1/2 in the superscripts is gone. Time is implicit in the FDTD method. Position, however, is explicit.

If absorbing boundary condition is added, the fields at the edge must be propagating outward. Therefore, an acceptable boundary condition might be [3]

$$E_x^n(0) = E_x^{n-2}(1) \qquad (6)$$

If relative dielectric constant ε_r is added, a portion of the electric field propagates into the medium and a portion is reflected, in keeping with basic electromagnetic theory. Therefore, (1) becomes [3]

$$\tilde{E}_x^{n+\frac{1}{2}}(k) = \tilde{E}_x^{n-\frac{1}{2}}(k) - \frac{1/2}{\varepsilon_r}\left[H_y^n\left(k+\frac{1}{2}\right) - H_y^n\left(k-\frac{1}{2}\right)\right] \qquad (7)$$

From this, the computer code becomes

```
Ex[k]=ex[k]+cb[k]*(hy[k-1]-hy[k])                    (8)
```

Where

```
cb[k]=0.5/epsilon          (9)
```

over those values of k that specify the dielectric material.

If a Gaussian pulse as the source switches to a sinusoidal source, the parameter pulse=exp(-0.5*(pow((t0-T)/spread, 2.0))) becomes

```
Pulse=sin(2*pi*freq_in*dt*T)(10)
```

The parameter freq_in determines the frequency of the wave. The time step dt is specified by [3]

$$\Delta t = \frac{\Delta x}{2c_0} \qquad (11)$$

Where c_0 is the speed of light in free space, and the cell size Δx is determined by [3]

$$\Delta x = \frac{\lambda_m}{10} \qquad (12)$$

$$\lambda_m = \frac{c_0/\sqrt{\varepsilon_r}}{f} \qquad (13)$$

Where f is the frequency of the wave in MHz.

If EM propagates in a media that also have a loss term specified by the conductivity, this loss term results in the attenuation of the propagating energy. Therefore, (7) becomes [3]

$$\tilde{E}_x^{n+\frac{1}{2}}(k) = \frac{\left(1-\frac{\Delta t.\sigma}{2\varepsilon_r\varepsilon_0}\right)}{\left(1+\frac{\Delta t.\sigma}{2\varepsilon_r\varepsilon_0}\right)}\tilde{E}_x^{n-\frac{1}{2}}(k) - \frac{1/2}{\varepsilon_r\left(1+\frac{\Delta t.\sigma}{2\varepsilon_r\varepsilon_0}\right)}\left[H_y^n\left(k+\frac{1}{2}\right) - H_y^n\left(k-\frac{1}{2}\right)\right] \quad (14)$$

From this, the computer code becomes

```
Ex[k]=ca[k]*ex[k]+cb[k]*(hy[k-1]-hy[k])              (15)
```

Where

```
ca[k]=(1-eaf)/(1+eaf)          (16)
```

cb[k]=0.5/(epsilon*(1+eaf)) (17)

eaf=dt*sigma/(2*epsilon*epsz)(18)

From equations (4), (5), (8), and (15), it can be seen that distance variable k will be used repeatedly in simulation of electric field $E_x(k)$ and magnetic field $H_y(k)$. The variable is called loop index. In computing, if we used sequential computing to access loop index, then solving the equations will be time consuming. To reduce the processing time, we introduce another approach to access loop index using parallel computing. Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel") [4]. Parallel programming for simulation in parallel computer is more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common [5]. However, throwing more resources at a task will shorten its time to completion. Main memory in a parallel computer is either shared memory, or distributed memory [6].

1. Implementing OpenMP on 1D FDTD

Open Multi-Processing (OpenMP) is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran, on most processor architectures and operating systems [7]. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior [8]. A method of parallelizing one-dimensional electromagnetic is a master thread forks a specified number of slave threads and a task is divided among them, as shown in Fig. 2.

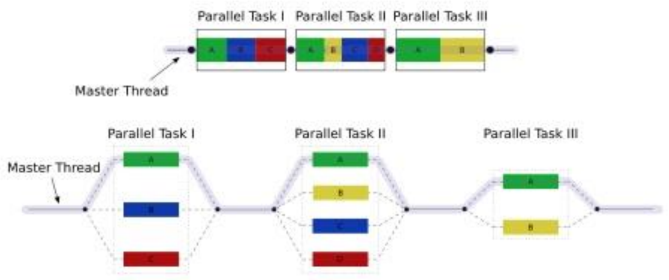


Figure 2. An illustration of multithreading

Parallel Task I is assigned to calculate the initialization of electric and magnetic fields, respectively. The parallel code for the initialization as it follows:

```
#pragma omp parallel for
        for (k=0; k<KE; k++)
        {
                ex[k]=0.;
                hy[k]=0.;
        }
```

Parallel Task II is assigned to calculate the magnitude of electric and magnetic fields at a given distance k. For 1D FDTD simulation in free space, the parallel code as it follows:

```
#pragma omp parallel for
        for (k=1; k<KE; k++)
```

```
                    {
                    ex[k]=ex[k]+.5*(hy[k-1]-hy[k]);
                    }
#pragma omp parallel for
          for (k=0; k<KE-1; k++)
                    {
                    hy[k]=hy[k]+.5*(ex[k]-ex[k+1]);
                    }
```

The code above is also used if absorbing boundary condition is added. If we simulate a pulse and/or a sinusoidal wave hitting a dielectric medium, the parallel code above becomes:

```
#pragma omp parallel for
          for (k = 1; k < KE; k++)
          {
                    ex[k] = ex[k] + cb[k] * (hy[k - 1] - hy[k]);
          }
#pragma omp parallel for
          for (k = 0; k < KE - 1; k++)
          {
                    hy[k] = hy[k] + .5*(ex[k] - ex[k + 1]);
          }
```

where cb[k] is defined by equation (9). If we simulate a sinusoidal wave hitting a lossy dielectric medium, the parallel code above becomes:

```
#pragma omp parallel for
for (k = 1; k < KE; k++)
{
          ex[k] = ca[k] * ex[k] + cb[k] * (hy[k - 1] - hy[k]);
}
#pragma omp parallel for
          for (k = 0; k < KE - 1; k++)
          {
                    hy[k] = hy[k] + .5*(ex[k] - ex[k + 1]);
          }
```

where ca[k] is defined by equation (16).

Parallel Task III is assigned to write the result of the magnitudes into a file called Ex.xls and Hy.xls.

```
#pragma omp for
          for (k=1; k<=KE; k++)
          {
                    fprintf(fp, "%3d\t %f\n", k, ex[k]);
          }
#pragma omp for
          for (k=1; k<=KE; k++)
          {
                    fprintf(fp, "%3d\t %f\n", k, hy[k]);
          }
```

To measure for how much relative performance improvement when executing a task, both in sequential and parallel programming, we used an equation as it follows [6]:

$$S = \frac{T_{sequential}}{T_{parallel}} \qquad (19)$$

where S is the resultant speedup, $T_{sequential}$ is the sequential execution time, and $T_{parallel}$ is the parallel execution time. Each program was run ten times. The timing was recorded using the C *time* header. The programs were run on Visual Studio Code under Windows Subsystem for Linux in Windows 11, with 2.40 GHz Intel Core i5 (8 cores) processor and 6 GB of RAM. The raw results were recorded by the script in files called Sequential.doc and Parallel.doc. These raw results were tabulated, averaged, and graphed.

2. Result

In this section, we discuss briefly all of simulation results which are categorized into simulation in one-dimensional free space, absorbing boundary condition in one dimension, propagation in a dielectric medium, simulating different sources, and propagation in a lossy dielectric medium.

A. One-Dimensional Free Space

The C computer code in equation (4)-(5) is a one-dimensional FDTD program. It generates a Gaussian pulse in the center of the problem space, and the pulse propagates away in both directions as seen in Fig. 3.

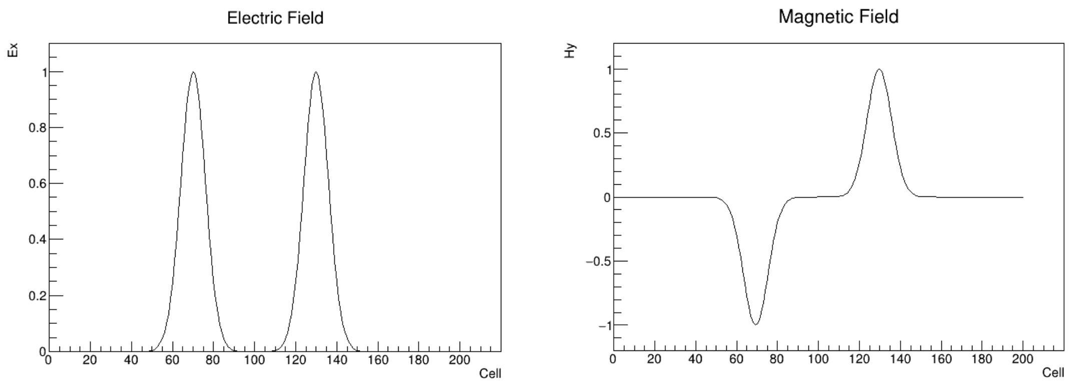
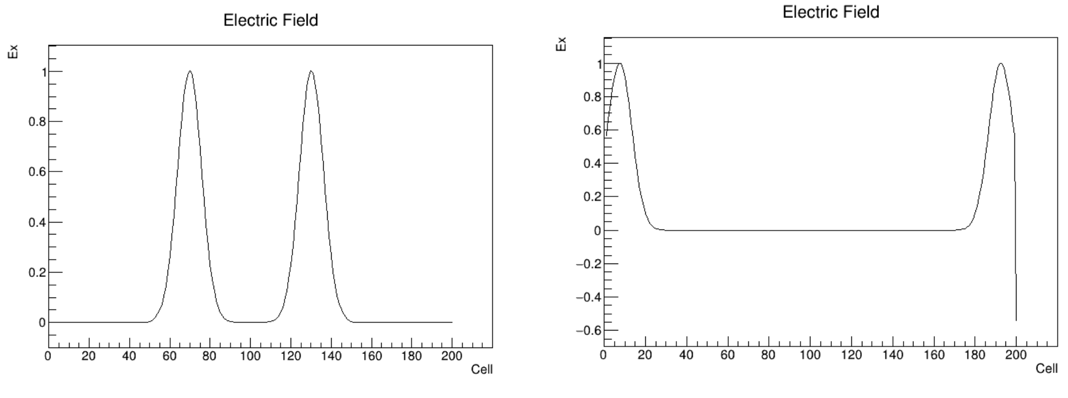


Figure 3. FDTD simulation of a pulse in free space after 100 timesteps. The pulse originated in the center and travels outward.

The E_x field is positive in both directions, but the H_y field is negative direction. This is because the E_x and H_y values are calculated by separate loops, and they employ the interleaving. After the E_x values are calculated, the source is calculated. This is done by specifying a value of E_x at the point k=kc, and overriding what was previously calculated. This is referred to as a "hard source," because a specific value is imposed on the FDTD grid.

B. The Absorbing Boundary Condition in One Dimension

To implement equation (6), we store a value of $E_x(1)$ for two timesteps, and we put it in $E_x(0)$. Boundary conditions such as these are implemented at both ends of the E_x array in C computer code. Fig. 4 and Fig. 5 show the results of a simulation using the code. A pulse that originates in the center propagates outward and is absorbed without reflecting anything back into the problem space.



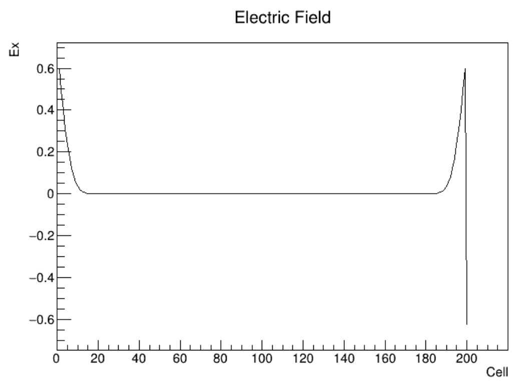


Figure 4. FDTD simulation of an electric field in free space after 100, 225, and 250 timesteps., respectively, with absorbing boundary conditions. The field is absorbed at the edges without reflecting anything back.

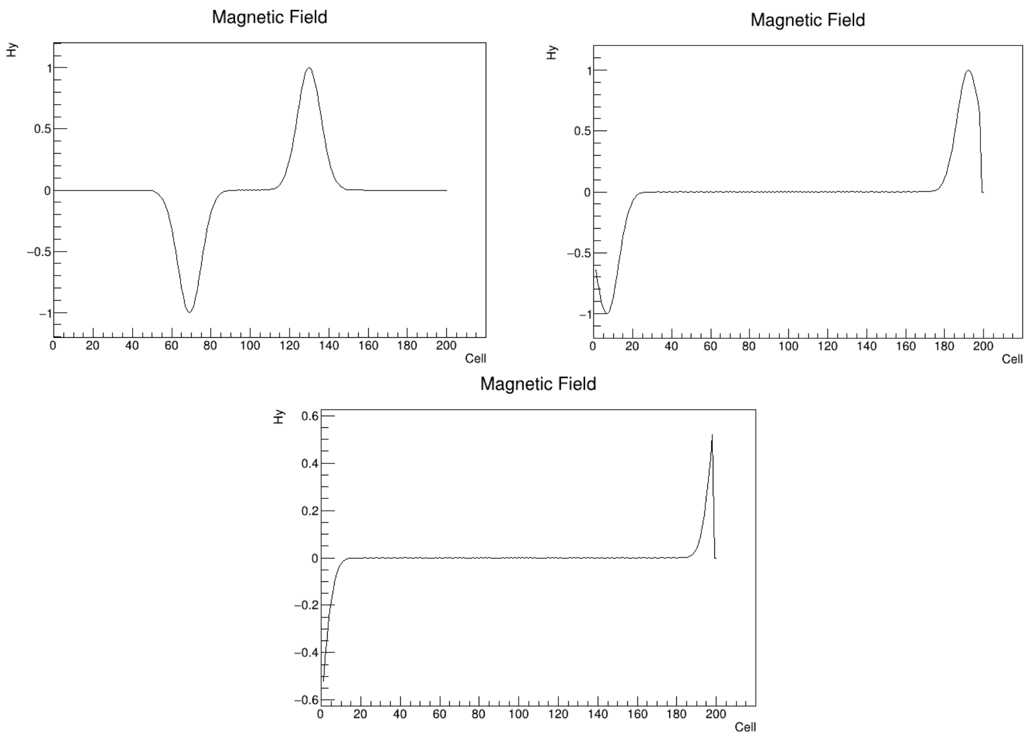


Figure 5. FDTD simulation of a magnetic field in free space after 100, 225, and 250 timesteps., respectively, with absorbing boundary conditions. Like electric part, the field is also absorbed at the edges without reflecting anything back.

C. Propagation in a Dielectric Medium

The C computer code in equation (8) is simulating the interaction of a pulse traveling in free space until it strikes a dielectric medium. The medium is specified by the parameter cb in equation (9). Fig. 6 and Fig. 7 show the result of a simulation with a dielectric medium having a relative dielectric constant of 4. We can see from the figures that a portion of the pulse propagates into the medium and a portion is reflected

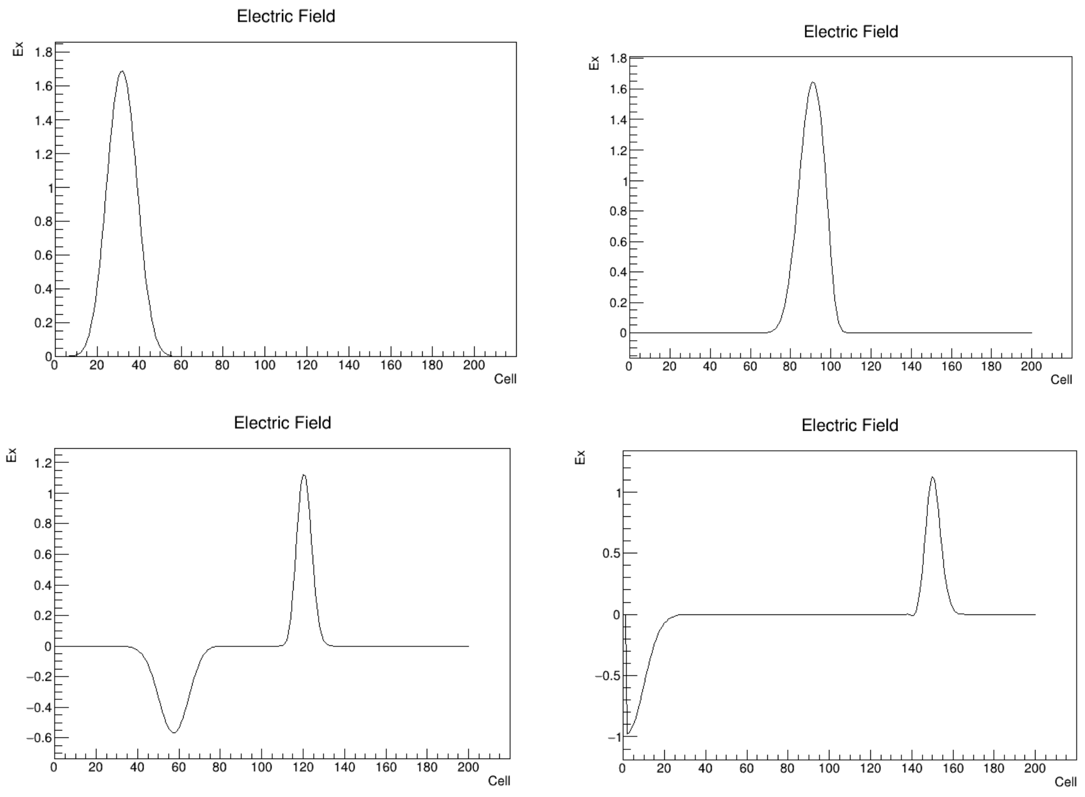


Figure 6. FDTD simulation of an electric field in free space after 100, 220, 320, and 440 timesteps., respectively, striking a dielectric material with a dielectric constant of 4.

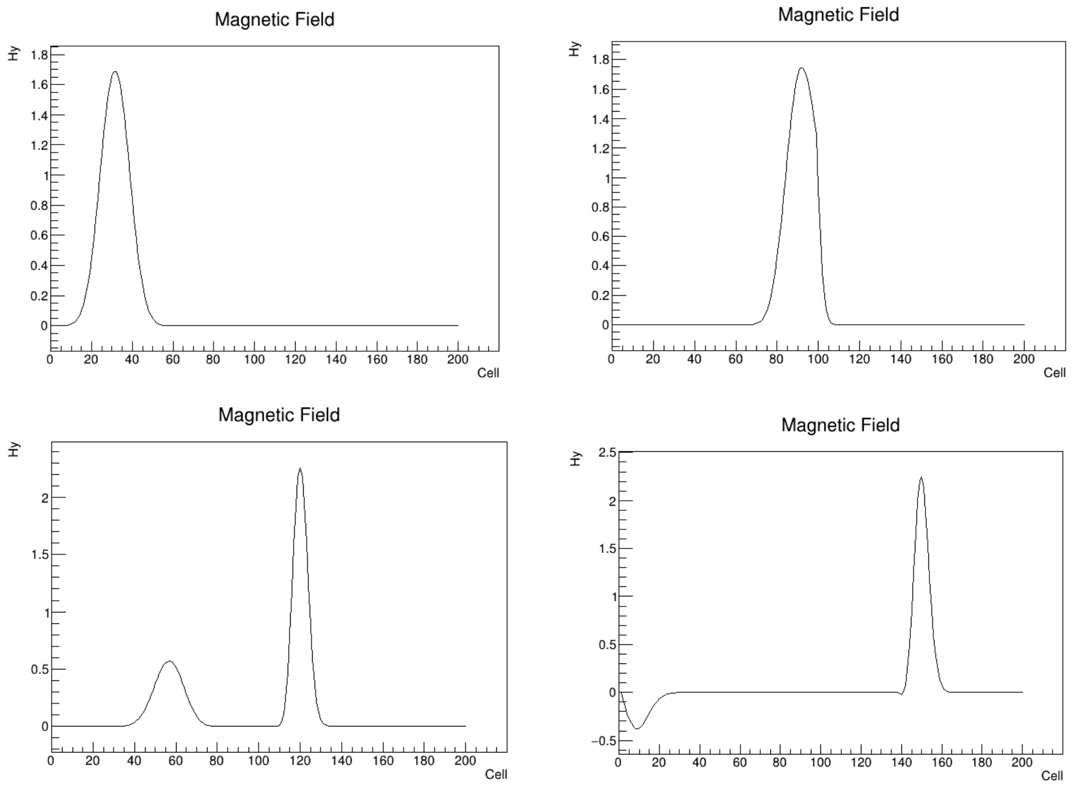


Figure 7. FDTD simulation of a magnetic field in free space after 100, 220, 320, and 440 timesteps., respectively, striking a dielectric material with a dielectric constant of 4.

D. *Simulating Different Sources*

The C computer code in equation (10) is a sinusoidal source which is striking the same dielectric medium problem. Fig. 8 and Fig. 9 show a sine wave with a frequency of 700 MHz is striking a dielectric material with a dielectric constant of 4. From the figures, we can see the simulation is stopped before the wave reached the far right side. It is because there is an absorbing boundary condition in free space. In the code, we specify the cell size and the time step explicitly (equation (11)-(12)). The cell size Δx is specified first because it is needed to calculate the time step Δt, and Δt is used in the calculation of pulse.

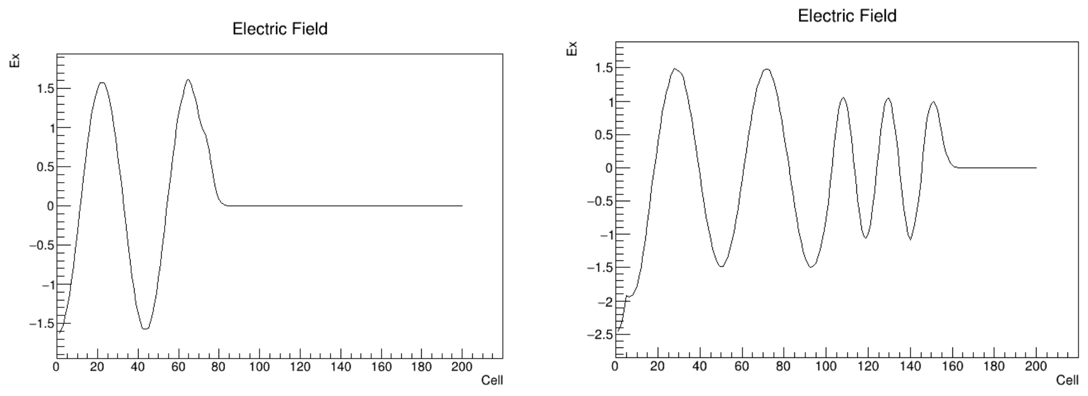


Figure 8. FDTD simulation of a 700 MHz sinusoidal electric field in free space after 150, and 425 timesteps., respectively, striking a dielectric material with a dielectric constant of 4.

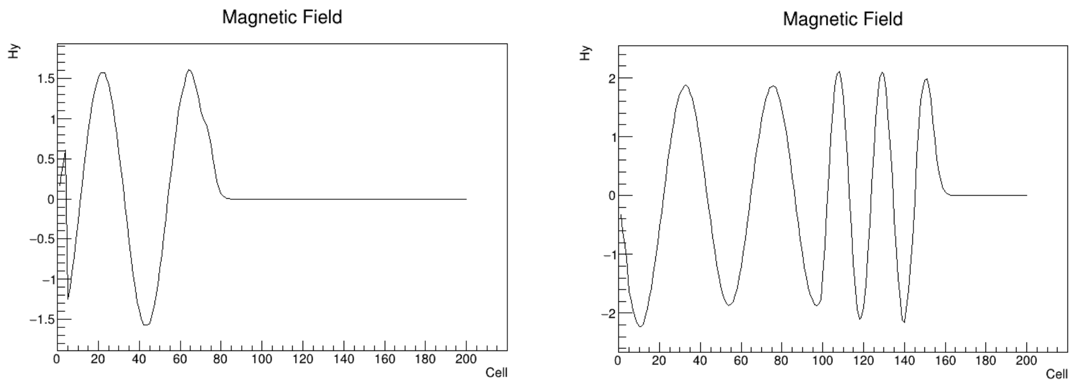


Figure 9. FDTD simulation of a 700 MHz sinusoidal magnetic field in free space after 150, and 425 timesteps., respectively, striking a dielectric material with a dielectric constant of 4.

E. *Propagation in a Lossy Dielectric Medium*

The C computer code in equation (15) is simulating a sine wave hitting a lossy medium that has a dielectric constant of 4 and a conductivity of 0.04. The pulse is generated at the far left side and propagates to the right. This is illustrated in Fig. 10. The waveform in the medium is absorbed before it hits the boundary, so we don't have to worry about absorbing boundary conditions.

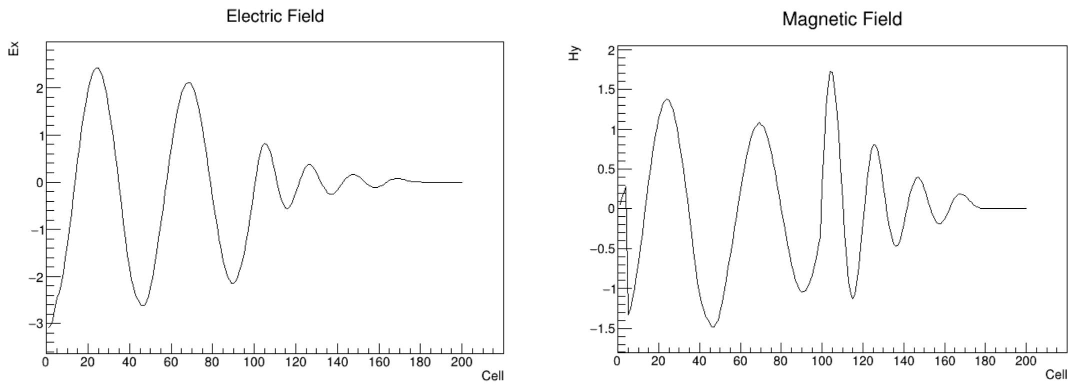


Figure 10. **FDTD simulation of a propagating sine wave striking a lossy dielectric material with a dielectric constant of 4 and a conductivity of 0.04 S/m after 500 timesteps. The source is 700 MHz.**

4. Conclusion

From simulations we get some graphs for different scenario on one-dimensional electromagnetic wave. The highest increase for speedup is obtained from one-dimensional free space around 50 times, while the lowest speedup is from propagation in a lossy dielectric medium around 10 times. We can conclude that more loops make speedup lower, so optimizing the simulation codes is necessary to decrease latency and to increase throughput.

Reference

[1] W. C. Chew, J. M. Jin, E. Michielssen, and J. Song, Fast and Efficient Algorithms in Computational Electromagnetics. Boston: Artech House, Inc., 2001.

[2] D. B. Davidson, Computational Electromagnetics for RF and Microwave Engineering. 2nd ed., New York: Cambridge University Press, 2011.

[3] D. M. Sullivan, Electromagnetic Simulation Using the FDTD Method. 2nd ed., New York: Wiley-IEEE Press, 2013.

[4] G. S. Almasi, and A. Gottlieb, Highly Parallel Computing. 2nd ed., Benjamin/Cummings Publishing, 1994.

[5] D. A. Patterson, and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface. 5th ed., Morgan Kaufmann, 2014.

[6] J. L. Hennessy, and D. A. Patterson, Computer Architecture: A Quantitative Approach. 5th ed., Morgan Kaufmann, 2011.

[7] A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts. 9th ed., USA: John Wiley & Sons, Inc., 2013.

[8] B. Chapman, G. Jost, R. V. D. Pas, Using OpenMP: Portable Shared Memory Parallel Programming. USA: The MIT Press, 2008.