Finite Difference Time–Domain Modelling of Metamaterials: GPU Implementation of Cylindrical Cloak

Attique Dawood

Department of Electrical Engineering,
National University of Computer and Emerging Sciences (FAST), Islamabad, Pakistan
E-mail: attique.dawood@nu.edu.pk

Abstract

Finite difference time–domain (FDTD) technique can be used to model metamaterials by treating them as dispersive material. Drude or Lorentz model can be incorporated into the standard FDTD algorithm for modelling negative permittivity and permeability. FDTD algorithm is readily parallelisable and can take advantage of GPU acceleration to achieve speed–ups of 5x–50x depending on hardware setup. Metamaterial scattering problems are implemented using dispersive FDTD technique on GPU resulting in performance gain of 10x–15x compared to conventional CPU implementation.

1. Introduction

Standard FDTD algorithm cannot cater for negative values of permittivity or permeability. This is because of the Courant stability criterion. As soon as the permeability or permittivity becomes less than unity the algorithm will not be stable. A metamaterial object can be modelled as a dispersive substance using either the Lorentz or Drude dispersive models. These models can yield negative values of permittivity (or permeability) for certain frequency ranges [1]. Using these dispersive models, FDTD update equations are modified and permittivity and permeabilities are replaced with terms dependent on frequency of operation.

Two problems are chosen for GPU implementation. First is the electromagnetic wave scattering by a slab with negative permittivity and permeability; also known as DNG (double negative) medium. Second problem is the simulation of cylindrical cloak. An incident Gaussian pulse on DNG slab will undergo dispersion resulting in different frequency components being separated. Refractive index and transmission coefficient are calculated numerically to ascertain the validity of implementation. The cylindrical cloak was first proposed and tested by Pendry et. al. [2]. The first FDTD implementation was by Zhao et. al [3] and implemented on Comsol, a commercial electromagnetic simulation software. Simulations are implemented on Matlab, C++ and GPU. Performance comparison reveals a 10–15 times increase in performance with GPU implementations. Performance gain is greater for larger problem sizes and greater simulation times.

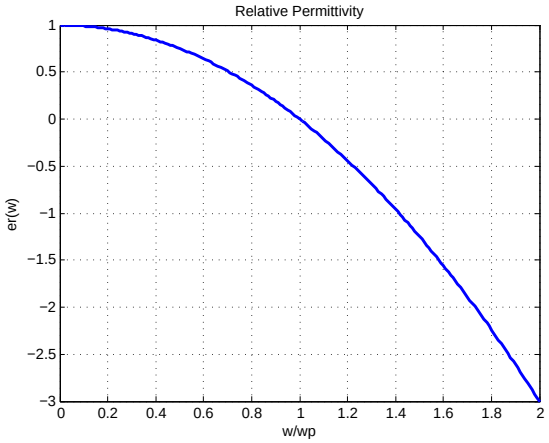


Figure 1: ϵ_r plotted against ω/ω_p for $\epsilon_\infty = 1$ and $\gamma = 0$

2. Drude Dispersion Model

In ideal conditions the permittivity (and permeability) of a material remain constant for any frequency and throughout the structure of that material. Speed of electromagnetic waves in such a medium remain constant if frequency changes. Additionally, there is no loss in energy as the waves pass through the medium.

In reality, such a material does not exist. Speed of EM waves varies with frequency of operation. Also, there is a loss associated with the material. A material is dispersive if its permittivity or permeability is dependent on frequency [4, Ch. 10].

The relative permittivity in Drude model is given by

$$\hat{\epsilon_r}(\omega) = \epsilon_\infty - \frac{\omega_p^2}{\omega^2 - j\gamma\omega}. \quad (1)$$

Where, ω_p is plasma frequency and γ is collision frequency. Setting $\gamma = 0$ and $\epsilon_\infty = 1$, relative permittivity comes out to be negative for $\omega/\omega_p > 1$ (figure 1). Thus, Drude model can be effectively used to model metamaterials with permittivity or permeability less than one by incorporating it into FDTD update equations.

3. FDTD Update Equations

Faraday's Law and Ampere's Law in differential form are given by

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{2}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t}. \tag{3}$$

From [5], the update equations for a wave propagating in z direction are given by

$$H_y^{n+\frac{1}{2}}\left[k+\frac{1}{2}\right] = H_y^{n-\frac{1}{2}}\left[k+\frac{1}{2}\right] + \frac{\Delta t}{\mu \Delta x}\left(E_x^n[k] - E_x^n[k+1]\right), \tag{4}$$

$$E_x^{n+1}[k] = E_x^n[k] + \frac{\Delta t}{\epsilon \Delta x}\left(H_y^{n+\frac{1}{2}}\left[k-\frac{1}{2}\right] - H_y^{n+\frac{1}{2}}\left[k+\frac{1}{2}\right]\right). \tag{5}$$

In the conventional FDTD algorithm future magnetic field components are first computed from past electric field components (eq. 4). Using the updated magnetic field components, future electric field components are then calculated (eq. 5) and the simulation proceeds in a leap–frog manner [5].

4. FDTD Update Equations Based on Drude Model

Electric flux density and electric field are related by

$$\mathbf{D} = \epsilon \mathbf{E}. \tag{6}$$

Where $\epsilon = \epsilon_r \epsilon_0$ and ϵ_r for Drude model is given by equation 1. Substituting Drude model ϵ_r, equation 6 can be written as

$$\omega^2 \mathbf{D} - \gamma(j\omega)\mathbf{D} = \epsilon_\infty \omega^2 \mathbf{E} - \omega_p^2 \mathbf{E} - \epsilon_\infty \gamma(j\omega)\mathbf{E}. \tag{7}$$

Following the treatment of [3] and [4], frequency domain quantities can be converted to time–domain using the relationships $j\omega \rightarrow \partial/\partial t$ and $\omega^2 \rightarrow -\partial^2/\partial t^2$. Moreover, fields multiplying with ω_p^2 are averaged in time. Second–order difference scheme is used, both for single and double derivatives to keep all the terms in accordance with the second–order nature of whole expression. This will result in easier implementation. The final form is given by

$$E_x^{n+1} = a_e\left(D_x^{n+1} - 2D_x^n + D_x^{n-1}\right) + b_e\left(D_x^{n+1} - D_x^{n-1}\right) + c_e\left(2E_x^n - E_x^{n-1}\right) + d_e\left(2E_x^n + E_x^{n-1}\right) + e_e E_x^{n-1}. \tag{8}$$

This is also known as the auxiliary update equation for electric field where a_e–e_e are scalars given by

$$a_e = 4/g,\ b_e = \gamma(2\Delta t)/g,\ c_e = 4\epsilon_0\epsilon_\infty/g,$$
$$d_e = -\epsilon_0\omega_p^2(\Delta t)^2/g,\ e_e = \epsilon_0\epsilon_\infty\gamma_e(2\Delta t)/g,$$
$$g = 4\epsilon_0\epsilon_\infty + \epsilon_0\omega_p^2(\Delta t)^2 + \epsilon_0\epsilon_\infty\gamma(2\Delta t)$$

A similar procedure can be carried out to obtain auxiliary update equation for magnetic field.

$$H_y^{n+1} = a_m\left(B_y^{n+1} - 2B_y^n + B_y^{n-1}\right) + b_m\left(B_y^{n+1} - B_y^{n-1}\right) + c_m\left(2H_y^n - H_y^{n-1}\right) + d_m\left(2H_y^n + H_y^{n-1}\right) + e_m H_y^{n-1}. \tag{9}$$

For a wave propagating in z direction, FDTD update equations are

$$B_y^{n+1}(k) = B_y^n(k) + \frac{\Delta t}{\Delta z}\left(E_x^n(k) - E_x^n(k+1)\right) \tag{10}$$

and

$$D_x^{n+1}(k) = D_x^n(k) + \frac{\Delta t}{\Delta z}\left(H_y^{n+1}(k-1) - H_y^{n+1}(k)\right). \tag{11}$$

Equations 10 and 11 drive the FDTD algorithm which give future values of By and Dx from past fields. Equations 8 and 9 are auxiliary equations which give future fields H_y and E_x at $n+1$. A dry run without any scatterer is carried out before actual simulation to record incident fields. After simulation any post–processing is done to calculate required parameters like refractive index. The whole algorithm is depicted in figure 2.

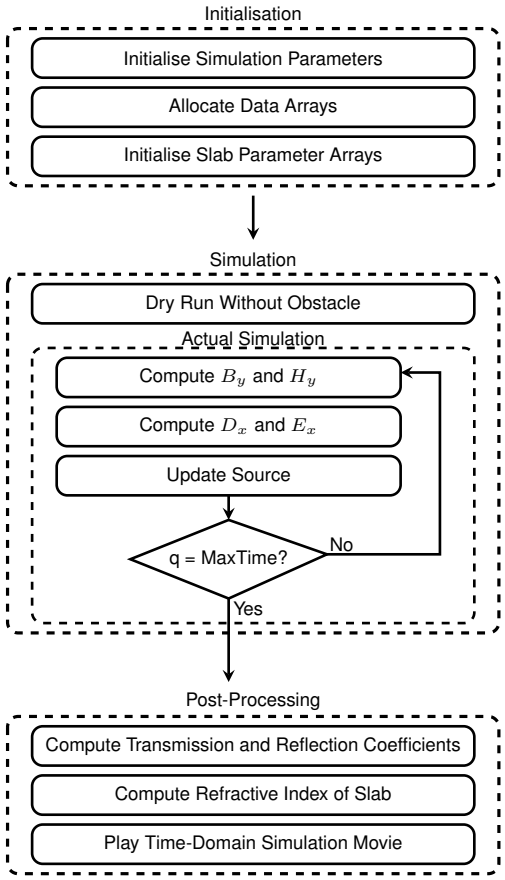


Figure 2: FDTD algorithm

5. GPU Considerations

With the evolution of 3D graphics arose a need for faster computing means to handle real–time graphics processing. The graphics processing unit or GPU was originally meant to act as a separate processor to handle graphics computations. A modern GPU has several hundred small processors that can work in parallel. Generally, these processors are referred to as shaders. The idea is to divide a problem into smaller sub–problems meant to execute in parallel. To take advantage of GPU acceleration a problem must have either task–parallel or data–parallel nature.

5.1. Task–Parallelism

In task–parallelism computation consist of several independent tasks that run concurrently. These tasks may be completely unrelated but the end result is dependent on their outputs. Consider summation of a series, where we want to compute the value of e^x from Taylor series. The mathematical expression is given by

$$e^x = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + ... + \frac{x^n}{n!} \quad (12)$$

A single–threaded conventional implementation would have to calculate all the terms one–by–one and then sum up the result in the end. However, in a multi–threaded implementation, each thread would calculate only one term. All the treads will perform their computation in parallel and the end result is then summed up. The computation–intensive task of calculating factorials and higher powers of x is parallelised and results in significant reduction of computation time.

5.2. Data–Parallelism

In data–parallelism same operation is performed on individual elements of data. A simple example is that of scalar matrix multiplication. Consider an array of size n being multiplied with a scalar constant c. The result of each multiplication can be calculated independently by assigning a separate thread for the task. This is illustrated in figure 3 where input array is `A[]` and resultant array is `B[]`. Data–parallelism applies to any scenario where values in resultant data array only depends on values from input array. FDTD is a good example of data–parallelism and a GPU implementation can take advantage of accelerated computing.

6. Simulation of 1D DNG Slab

6.1. Problem Specification

An electromagnetic wave travelling in z direction is incident on a slab with negative values of permittivity and permeability (DNG) at the frequency of operation [6]. Sinusoidal wave, Gaussian pulse and Ricker wavelet are used as sources. Transmission and reflection coefficients are calculated at the air–slab interface. Refractive index of slab for a range of frequencies is also computed. By varying parameters, the simulation can be scaled to any desired frequency or wavelength.

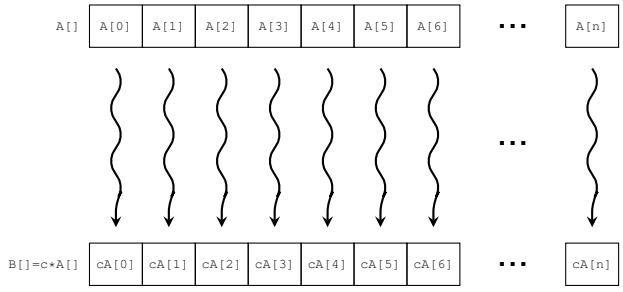


Figure 3: Data–parallelism

6.2. Simulation Results

The simulation is run for both lossless and lossy cases with sinusoidal, Gaussian and Ricker wavelet sources. The slab parameters are set such that at frequency of operation, f_0, the permittivity and permeability of slab are both negative and result in a refractive index $n = -1$.

6.3. Simulation Parameters

The number of spatial steps is set as 4096 and simulation is run for 4×4096 time steps. The slab is located between steps 1365 and 2731. Δz or spatial step is set as 3 mm and time step, Δt, is set as 50 ps. Frequency of operation is $f_0 = 0.1953125$ GHz and Courant number for this configuration comes out to be $S_c = 0.5$. In order to obtain relative permittivity and permeability of -1 at required f_0, plasma frequencies are set as $\omega_{pm}^2 = \omega_{pe}^2 = 2 \times (2\pi f_0)^2$ with $\epsilon_\infty = \mu_\infty = 1$. First order absorbing boundary condition (ABC) are applied on fields at end points.

6.4. Incident and Transmitted Fields

Simulation with Gaussian pulse reveals that low frequency components are reflected at the interface which is confirmed from the transmission and reflection coefficients obtained for the air–slab interface. At f_0, the transmission coefficient is 1 and there are no reflections when a sinusoidal source with f_0 is incident on the slab. Under steady–state conditions, transmitted wave inside the slab has negative phase velocity while energy is propagating in $+\hat{z}$ direction as expected.

6.5. Refractive Index

Following [6], the refractive index was calculated from

$$n_{FDTD} = \frac{1}{jk_0(z_1 - z_2)} log \left| \frac{E_x(\omega, z_2)}{E_x(\omega, z_1)} \right|. \quad (13)$$

Where, k_0 was the wave number set as ω_0/c and the fields were recorded at locations $z_1 = 1415\Delta z$ and $z_2 = 1424\Delta z$. For both, Gaussian pulse and Ricker wavelet, $Re(n)$ was -1 at f_0 while $Im(n)$ was sufficiently close to 0.

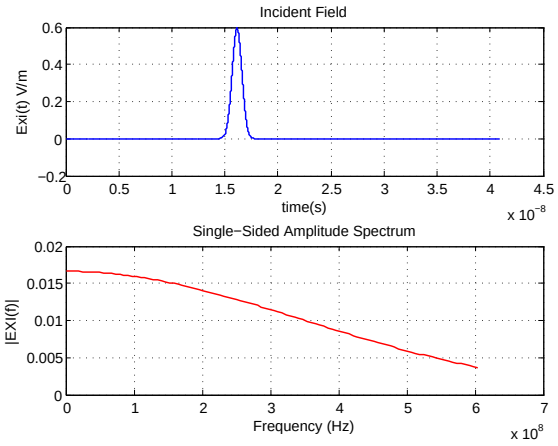


Figure 4: Incident Gaussian pulse

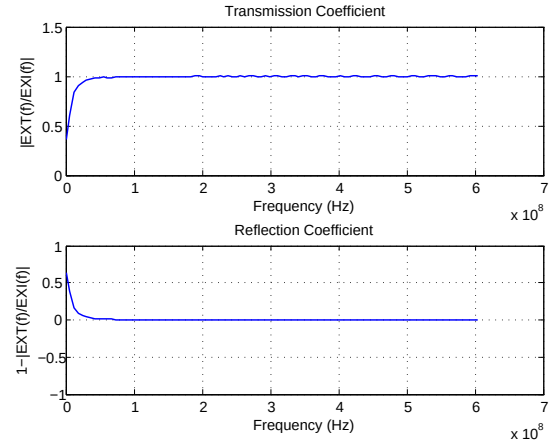


Figure 5: Transmission and reflection coefficients

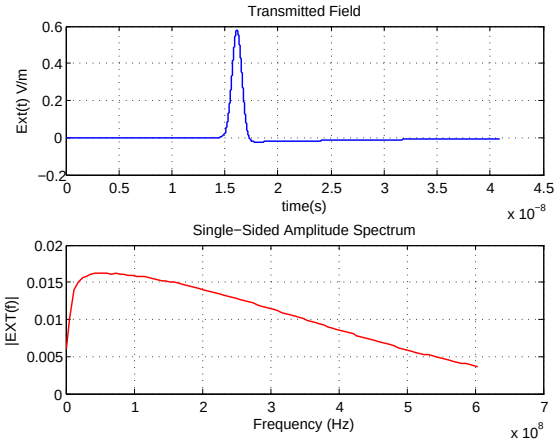


Figure 6: Transmitted Gaussian pulse

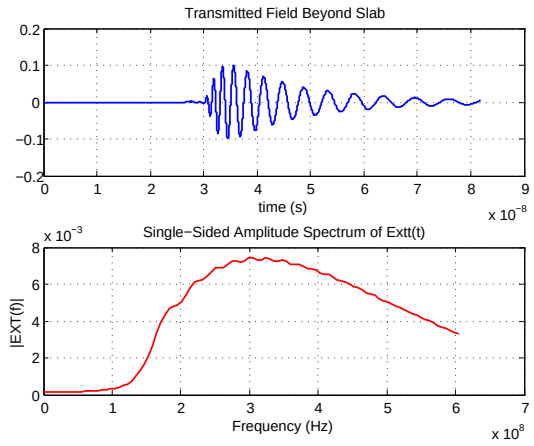


Figure 7: Transmitted Gaussian pulse beyond slab

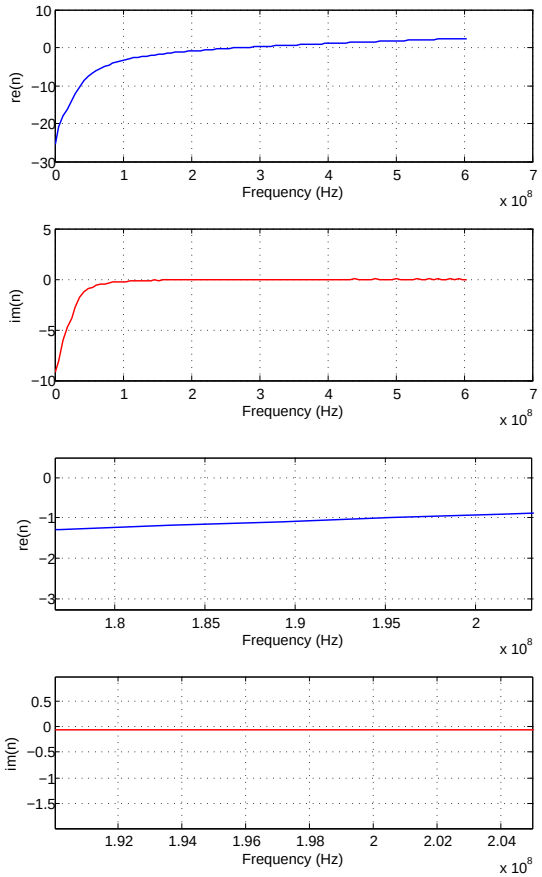


Figure 8: Refractive index

7. Simulation of 2D DNG Slab

7.1. Update Equations

The most common 2D configurations are TE^z or TM^z polarisation where the problem space is confined to xy–plane. In TE^z, electric field components are transverse to z–axis and vice versa. For 2D simulation of DNG slab, TM^z polarisation is assumed. The field components of interest would be E_z, H_x and H_y. Following the approach for 1D

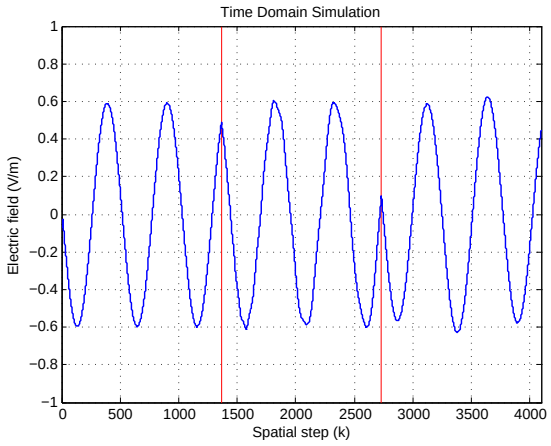


Figure 9: Steady-state under lossless conditions

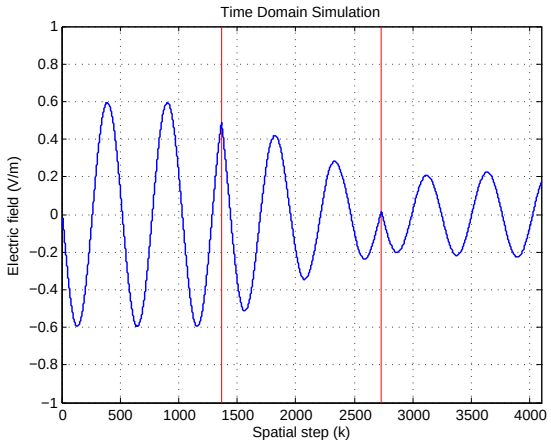


Figure 10: Steady-state under lossy conditions

case FDTD update equations are

$$D_z^{n+1}[i,j] = D_z^n[i,j] + \frac{\Delta t}{\Delta}\left(H_y^{n+\frac{1}{2}}\left[i+\frac{1}{2},j\right] - H_y^{n+\frac{1}{2}}\left[i-\frac{1}{2},j\right] - H_x^{n+\frac{1}{2}}\left[i,j+\frac{1}{2}\right] + H_x^{n+\frac{1}{2}}\left[i,j-\frac{1}{2}\right]\right), \tag{14}$$

$$B_x^{n+\frac{1}{2}}\left[i,j+\frac{1}{2}\right] = B_x^{n-\frac{1}{2}}\left[i,j+\frac{1}{2}\right] + \frac{\Delta t}{\Delta}\left(-E_z^n[i,j+1] + E_z^n[i,j]\right) \tag{15}$$

and

$$B_y^{n+\frac{1}{2}}\left[i+\frac{1}{2},j\right] = B_y^{n-\frac{1}{2}}\left[i+\frac{1}{2},j\right] + \frac{\Delta t}{\Delta}\left(E_z^n[i+1,j] - E_z^n[i,j]\right). \tag{16}$$

It is important to note here that DNG slab medium is assumed anisotropic. The auxiliary update equations (9 and 8) to obtain electric and magnetic fields, E_v and H_v ($v \in x, y, z$), from corresponding flux densities, D_v and B_v, remain unchanged.

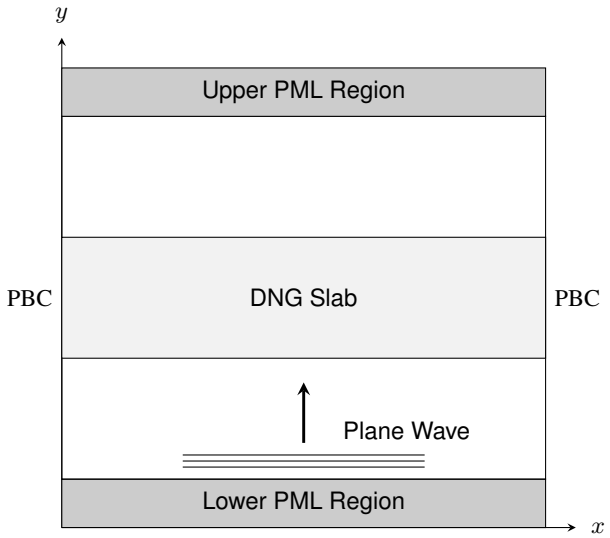


Figure 11: Simulation geometry

7.2. Simulation Geometry

The DNG slab interface is perpendicular to incident plane wave propagating in $+y$ direction. Periodic boundary conditions (PBC) are applied at $x = 0$ and $x = x_{max}$. In y direction the grid is terminated at both ends by perfectly matched layer (PML) to absorb any incoming waves. The solution geometry is depicted in figure 11. The arrangement of field nodes is shown in figure 12. Magnetic field components are updated first and then used to update electric field.

The solution space is bounded in y direction at both ends by H_x which acts as the boundary for PML. H_x at these end points is set to 0 so that any out–going fields are reflected back into PML. Essentially, the j size of H_x arrays is one greater than H_y and E_z arrays.

7.3. Simulation Parameters and Results

The solution space is 512 cells in both x and y directions without taking into account the width of PML, which is 50 cells wide. The plane wave source is located 10 cells from the lower PML layer. The spatial and temporal steps are set as $\Delta x = \Delta y = \Delta = 3$ mm and $\Delta t = 50$ ps, respectively; with a Courant number of 0.5. Frequency of operation is $f_0 = 1.5625$ GHz. The DNG slab parameters are same as in the case of 1D DNG simulation.

Figure 13 shows the refractive index obtained for a range of frequencies with a Gaussian pulse excitation. Again, real part of refractive index at f_0 is close to -1, whereas, imaginary part is close to 0.

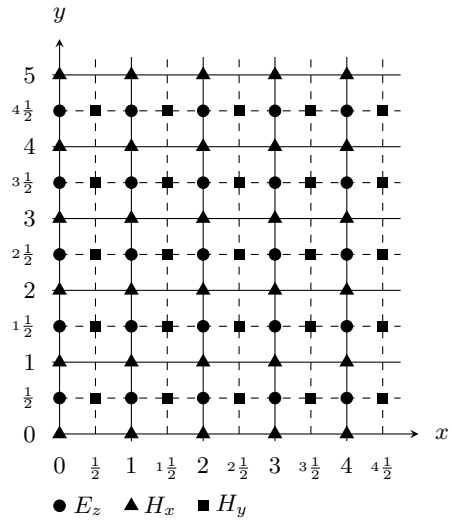


Figure 12: Arrangement of field nodes

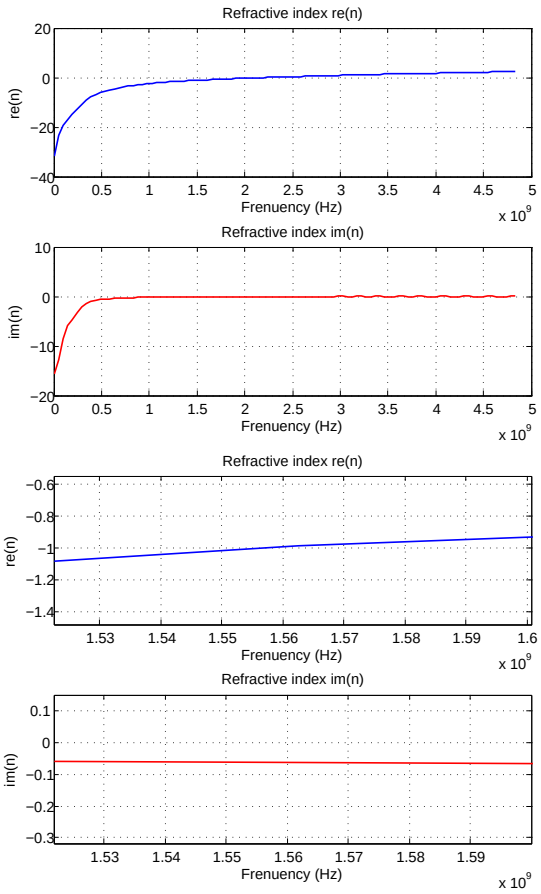


Figure 13: Refractive index of 2D DNG slab

8. FDTD Modelling of Lossless Cylindrical Cloak

8.1. The Electromagnetic Invisibility Cloak

It is well known that we can only see things when light bounces off them and reaches our eyes. This is the reason why transparent objects can sometimes be difficult to spot,

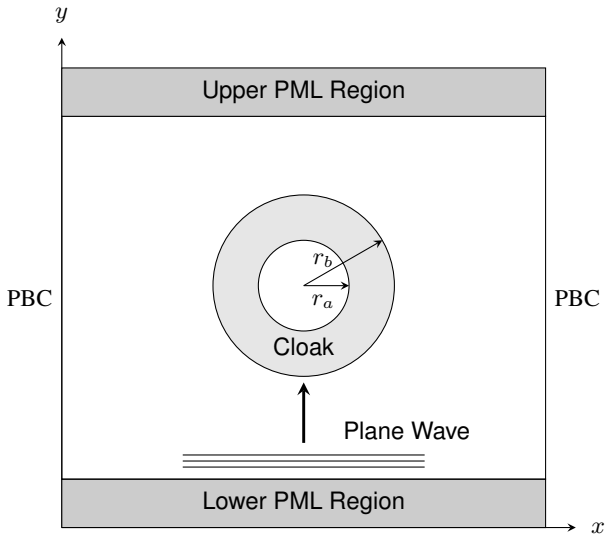


Figure 14: Simulation geometry

like window panes. If we can somehow make light bend around the corners of an object so that it continues on its trajectory then that object will appear invisible. Invisibility has, for ages, remained confined to only fiction books but in modern day world of optical transformations and artificially engineered materials, this no longer is a mystery.

An electromagnetic cloak at microwave frequencies was proposed and tested for the first time by Pendry et. al. [2]. The cloak is cylindrical in shape meant to hide a circular object. This is probably the simplest implementation sufficient for proof of concept. To bend light around the circular object the cloaking material must exhibit negative values of permittivity and permeability. The cloaking medium itself is anisotropic. The first attempt at modelling this electromagnetic cloak using the FDTD algorithm was by Zhao et. al. [3]. Their FDTD implementation was tested using simulation software Comsol.

8.2. Problem Specification

The implementation in [3] is for TE^z while in this paper TM^z approach is followed as in the case of DNG slab problems. The cloaking parameters are same and given by

$$\mu_r(r) = \frac{r - r_a}{r}, \tag{17}$$

$$\mu_\phi(r) = \frac{r}{r - r_a}, \tag{18}$$

$$\epsilon_z(r) = \left(\frac{r_b}{r_b - r_a}\right)^2 \frac{r - r_a}{r}. \tag{19}$$

Here, r_a and r_b are the inner and outer radii of cloaking shell. r_a and r_b are chosen as 0.1 m and 0.2 m respectively. The FDTD implementation is first tested on Matlab, then implemented using C++ and finally on GPU. The problem geometry is depicted in figure 14. The FDTD update equations for TM^z are completely analogous to those

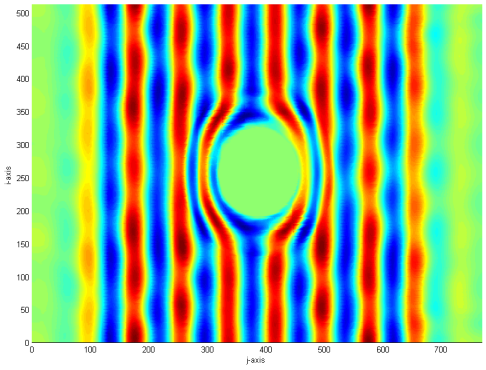


Figure 15: Ez under steady state for lossless cloak

	Platform/ Configuration	Compiler/ Toolchain
Matlab	x64	NA
C++	x64/O3	VC++, gcc/g++
OpenCL	x64/O3	VC++, gcc/g++
CUDA	x64/O3	nvcc for win/linux

Table 1: Operating system, platform/configuration and compiler/toolchain used for performance testing

for TE^z presented in [3]. The update equations for D_z, H_x and H_y remain the same as in the case of 2D DNG problem. Figure 15 shows E_z under steady–state. All simulation codes are available at `http://code.google.com/p/computational-electromagnetics/`

9. Performance Analysis

9.1. Hardware and Software Set–up

Matlab, C++, CUDA and OpenCL implementations are tested with respect to space and time. Operating system, platform/configuration and compiler/toolchain for these implementations are listed in table 1. Simulations use `double` data type for arrays on 64 bit optimised platform. The CPU is an Intel Core 2 Duo E8400 @3.00 GHz with 4 GB RAM. For CUDA simulations, GPU is GTX 500 Ti with 192 shaders and 1 GB of memory. Visual C++ 2010 Express is the IDE used on 64 bit Windows 7. The flavour of linux is Fedora 14 64 bit. Hardware and software configuration is listed in table 2.

10. Conclusions

The goal of this paper was to analyse GPU performance gain of dispersive FDTD and GPU implementation of lossless cylindrical cloak. It was shown that materials with negative permeability and permittivity can be effectively modelled using Drude dispersive model. The numerical results showed good agreement with theoretical values in the case of DNG slab problem

CPU	Intel Core 2 Duo E8400 @3.00 GHz
RAM	4.00 GB DDR2
GPU	nVidia Geforce GTX 550 Ti 1 GB
Matlab	2010a 64 bit
Linux	Fedora 14 64 bit
Windows	Win7 64 bit
Cygwin	64 bit on Win7

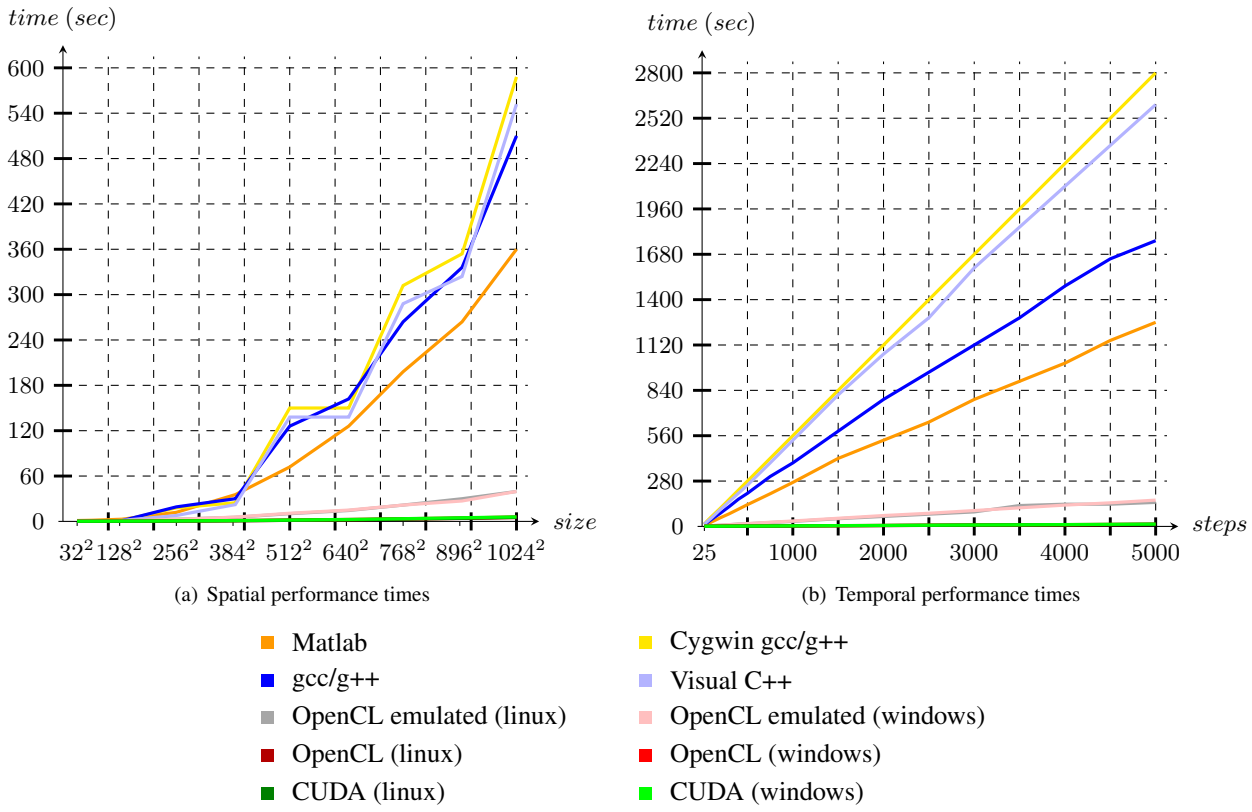
Table 2: Hardware and software used for performance testing

The 1D and 2D cases of slab problem was then implemented using C++ and GPU. A comprehensive performance analysis showed that GPU was able to perform much better both, as the problem domain and maximum time stepping are increased. For smaller problems, however, it is better to use Matlab or C++ due to data copying overhead associated with GPU implementation.

The lossless case of cylindrical cloak, as presented by Zhao et. al. in [3], was implemented using Matlab, C++ and finally on GPU with minor modifications. The simulations showed adequate similarity to accepted results.

Acknowledgement

I would like to thank Dr. Rashad Ramzan for his guidance.



(a) Spatial performance times

(b) Temporal performance times

Figure 16: Performance comparison

References

[1] S. G. et al, "Metamaterials and fdtd based numerical modeling studies," *ELECO*, Sep 2007.

[2] D. Schurig, J. J. Mock, B. J. Justice, S. A. Cummer, J. B. Pendry, A. F. Starr, and D. R. Smith, "Metamaterial Electromagnetic Cloak at Microwave Frequencies," *Science*, vol. 314, no. 5801, pp. 977–980, 2006.

[3] Y. H. Argyropoulos C., Yan Zhao, "A radially-dependent dispersive finite-difference time-domain method for the evaluation of electromagnetic cloaks," *IEEE Transactions on Antennas and Propagation*, vol. 57, pp. 1432–1441, May 2009.

[4] J. B. Schneider, *Understanding the Finite Difference Time-Domain Method*. www.eecs.wsu.edu/~schneidj/ufdtd, 2010.

[5] K. Yee, "Numerical solution of inital boundary value problems involving maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. 14, pp. 302–307, May 1966.

[6] R. W. Ziolkowski and E. Heyman, "Wave propagation in media having negative permittivity and permeability," *Physical Review*, vol. 64, October 2001.