Report

Finite Difference Time Domain

Martin Aleksiev, Felix Wechsler, Mei Yunhao, Mingxuan Zhang

Group 3

June 24, 2020

Abbe School of Photonics

Friedrich-Schiller-Universität Jena

Contents

1 Introduction

In this report we present the basics of the Finite Difference Time Domain numerical method. This method allows to fully solve Maxwell's equation without any approximations. After providing the physical background, we show how to implement the solution with a Yee-grid and show results for the simulation of a pulsed beam in a 1D and 3D case.

2 Physical Background

The physical basics are Maxwell's equation (MWEQ):

$$\nabla \times \mathbf{E}(\mathbf{r},\omega) = -\frac{\partial \mathbf{B}(\mathbf{r},\omega)}{\partial t} \tag{1}$$

$$\nabla \times \mathbf{H}(\mathbf{r},\omega) = \frac{\partial \mathbf{D}(\mathbf{r},\omega)}{\partial t} + \mathbf{j}(\mathbf{r},\omega) \tag{2}$$

$$\nabla \cdot \mathbf{D}(\mathbf{r},\omega) = \rho(\mathbf{r},t) \tag{3}$$

$$\nabla \cdot \mathbf{B}(\mathbf{r},\omega) = 0 \tag{4}$$

with $\mathbf{E}(\mathbf{r},\omega)$ being the electric field, $\mathbf{H}(\mathbf{r},\omega)$ the magnetic field, $\mathbf{D}(\mathbf{r},\omega)$ the dielectric flux density, $\mathbf{B}(\mathbf{r},\omega)$ the magnetic flux density, $\mathbf{P}(\mathbf{r},\omega)$ the dielectric polarization, $\rho(\mathbf{r},t)$ the external charge density and $\mathbf{j}(\mathbf{r},\omega)$ the macroscopic current density. In an isotropic, dispersionless and non-magnetic media we furthermore obtain the following material equations:

$$\mathbf{D}(\mathbf{r},\omega) = \epsilon_0 \epsilon(\mathbf{r}) \mathbf{E}(\mathbf{r},\omega) \tag{5}$$

$$\mathbf{B}(\mathbf{r},\omega) = \mu_0 \mathbf{H}(\mathbf{r},\omega) \tag{6}$$

In this case, MWEQ can be expressed as:

$$\nabla \times \mathbf{E}(\mathbf{r},\omega) = -\mu_0 \frac{\partial \mathbf{H}(\mathbf{r},\omega)}{\partial t} \tag{7}$$

$$\nabla \times \mathbf{H}(\mathbf{r},\omega) = \epsilon_0 \epsilon(\mathbf{r}) \frac{\partial \mathbf{E}(\mathbf{r},\omega)}{\partial t} + \mathbf{j}(\mathbf{r},\omega) \tag{8}$$

7 and 8 are the final equations we are going to solve in the next sections.

3 Numerical Implementation

To solve the equations, we can explicitly express one of the cross products:

$$\frac{\partial E_x}{\partial t} = \frac{1}{\epsilon_0 \epsilon(\mathbf{r})} \left(\frac{\partial H_z}{\partial y} - \frac{\partial H_z}{\partial y} - j_x \right) \tag{9}$$

Using a second order central-difference scheme for both the time and spatial derivatives we obtain the resulting equations for a 1D case:

$$E_x^{n+1} = E_x^{n-1} + \frac{1}{\epsilon_0 \epsilon} \frac{\Delta t}{\Delta x} (H_{x+\Delta x}^n - H_{x-\Delta x}^n - j_x) \tag{10}$$

$$H_y^{n+1} = H_y^{n-1} + \frac{1}{\mu_0} \frac{\Delta t}{\Delta x} (E_{x+\Delta x}^n - E_{x-\Delta x}^n) \tag{11}$$

It can be observed that E and H are updated in an alternating way. Therefore, one can introduce the Yee grid with a Leapfrog time stepping scheme. A visual representation of the scheme can be see in Figure 1.

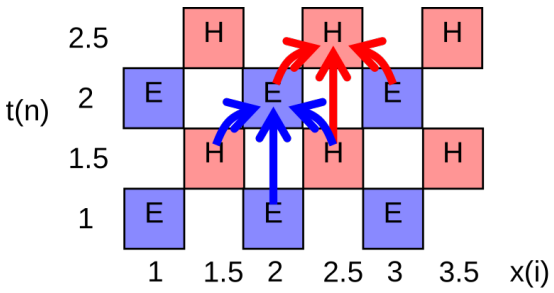


Figure 1: Yee grid with a Leapfrog time stepping. Figure taken from [PRK20].

One can finally express the equations in the Yee grid as:

$$E_i^{n+1} = E_i^n + \frac{1}{\epsilon_0 \epsilon} \frac{\Delta t}{\Delta x} (H_{i+0.5}^{n+0.5} - H_{i-0.5}^{n+0.5} - \Delta x \cdot j_{i+0.5}^{n+0.5}) \tag{12}$$

$$H_{i+0.5}^{n+1.5} = H_{i+0.5}^{n+0.5} + \frac{1}{\mu_0} \frac{\Delta t}{\Delta x} (E_{i+1}^{n+1} - E_i^{n+1}) \tag{13}$$

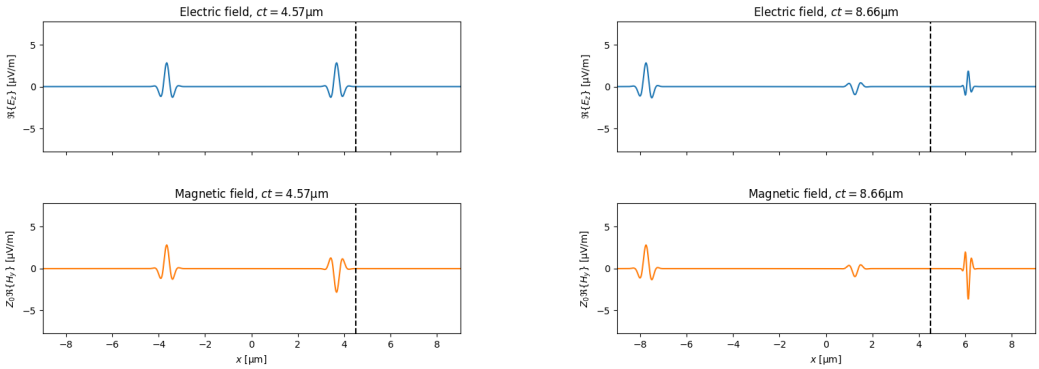
The current source j_z is defined using a delta distribution, meaning it is zero everywhere, except for one spatial position. It has a gaussian temporal envelope and is described by the following equation:

$$j_z = \exp(-2\pi i f t) \cdot \exp\left(-\frac{(t - t_0)^2}{\tau^2} \right) \tag{14}$$

The fields are calculated for different spatial positions using indexing and for different times via a for loop. The E-field is assumed to be a perfect conductor at the boundaries and as a result, those values are set to zero.
For the 3D case, there are a fer aditional considerations taken. The tangential E-fields and normal H-fields are stored in arrays of size N, whereas the tangential H-fields and normal E-fields are stored in $N-1$ size arrays. This is due to the Yee grid being in 3 dimentions and having integer and fractional indices. Due to these indices, the permittivity must be aslo interpolated. A for loop is used to calculate the fields for different times, analogously to the 1D case with the addition of saving the calculated fields every output step, which we choose. These fields are interpolated so that we have all fields on a common grid in space and time.

4 Results

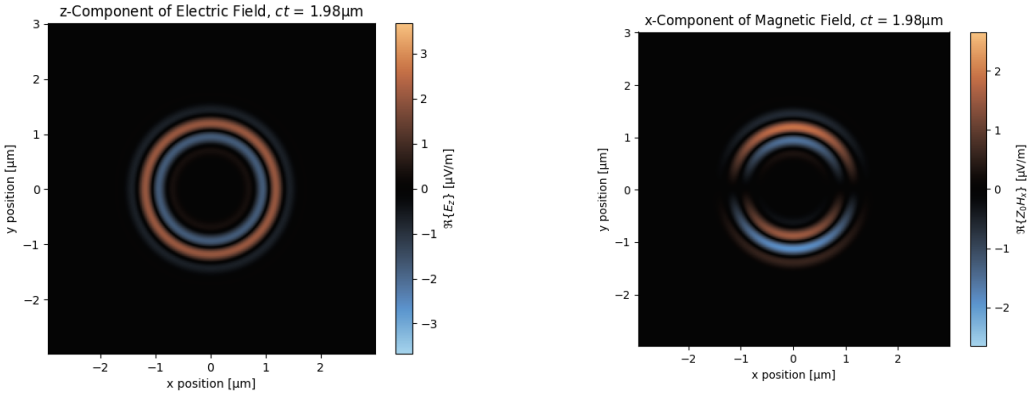
In this section we present some results of the simulations. Figure 2 shows the electric and magnetic field for a current being injected into the space. We observe that two pulses are propagating in different direction.



(a) $ct = 4.57\,\mu m$ (b) $ct = 8.66\,\mu m$

Figure 2: Electric and magnetic field for the 1D problem.

In Figure 2b we can observe that in the medium on the right hand side of the dashed line (higher ϵ) the pulse is compressed. This is due to the slower group velocity. Furthermore the amplitude is reduced. We notice that a third pulse appeared due to reflection according to Fresnel reflection laws.
In Figure 3 we can see the electric field propagating in radial manner starting at the center. In the center we injected a current guided in z direction.



(a) Electric field of the z-component

(b) Magnetic field of the x-component

Figure 3: Cross section at of the 3D field with a current pulse originating from the center.

The electrical field is radial symmetric, the magnetic field not. This is due to the reason that we displaying the x component of the H field. And due to the divergence condition the x component must be 0 in the x direction.

4.1 Computational Performance

In this section we provide some data for the computational performance. Our setup for all measurements was Python 3.8.2, SciPy 1.3.2, NumPy 1.17.3 and a Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz. The example scripts are attached to this report. In Table 1 we can see the computational time for different datatypes and for 1D and 3D. Obviously 1D is much faster than the full 3D approach. However,

	Total time needed in s
1D, single precision	0.136 ± 0.001
1D, double precision	0.187 ± 0.001
3D, single precision	5.11 ± 0.05
3D, double precision	6.413 ± 0.02

Table 1: Results for the computing time

we can see that the datatype does not have a significant impact on the time needed. This is due to the reason that Python is an interpreted language and therefore source code relying on for loops is usually slow. In Python this cannot circumvented since

the core functions within the for loop are rather trivial and do not rely on external functions call taking most of the time. For a better performance once should use a compiled language or modern approaches like Julia.

4.2 Convergence Rate

In this part we show results of the convergence rate for different spatial and time resolutions. For a quick estimation we compare the fields to higher sampled numerical versions instead of an analytical one. In our source code the spatial and temporal sampling are connected:

$$\Delta t = \frac{\Delta r}{2c} \tag{15}$$

Therefore by varying the spatial sampling we also vary the temporal one.

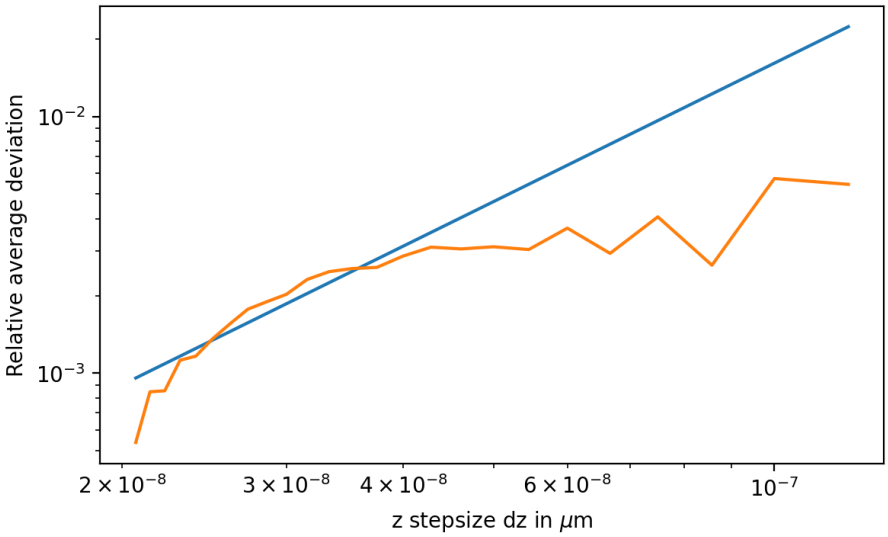


Figure 4: Convergence behaviour with different Δr sampling.

In Figure 4 we can see the convergence behaviour of the FDTD. Note that this plot is double logarithmic. The slope of the blue curve is $(1.79 \pm 0.16)\,\frac{1}{\mu m}$. The theoretical convergence of the algorithm is 2, so our measured values comes close to that one. The reason for the deviation could be the boundary handling and artificats due to the fact that we comparing against a numerical solution and not an analytical one.

5 Conclusion

In this report we explained the physical and numerical basics behind the finite difference time domain method to solve Maxwell's equation. We could observe several physical effects like different group velocities or the validation of divergence condition. At the end we presented several computational results like time and convergence.

References

[PRK20] T. Pertsch, C. Rockstuhl, and T. Kaiser. *Computational Photonics*. Lecture Notes on Computational Photonics. 2020.

[Vir+20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. 1. 0. Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.