Derivation

May 15, 2020

1 Electromagnetic Waves in One Dimension by the Finite Difference Time Domain Method

by Anders Ward

In this project we will explore the propagation of electromagnetic waves in a vaccum. First, a travelling plane wave will be explored in detail, then, a section on the code implementation of the resulting coupled partial differential equations. We end with a note on stabilityThe algorithm used will be an FDTD (Finite Difference Time Domain).

1.1 Maxwell's Equations and EM Waves

Let's start with Maxwell's Equations...

E - The Electric field
B - The Magnetic field
∇ - The del operator, essentially the derivative in 3 dimensions.
ρ - The charge density function
J - The current density function
ϵ - A constant, the permittivity of the medium
μ - A constant, the permeability of the medium

$$\nabla \cdot E = \frac{\rho}{\epsilon}$$

$$\nabla \cdot B = 0$$

$$\nabla \times E = -\frac{\partial B}{\partial t}$$

$$\nabla \times B = \mu_0 J + \mu_0 \epsilon_0 \frac{\partial E}{\partial t}$$

Our simulation takes place in a vacuum, where there are no particles to carry charge ρ or current J. Additionally the permittivity and permeability of a vacuum are given by ϵ_0 and μ_0 respectively.

$$\nabla \cdot E = 0$$

$$\nabla \cdot B = 0$$

$$\nabla \times E = -\frac{\partial B}{\partial t}$$

$$\nabla \times B = \mu_0 \epsilon_0 \frac{\partial E}{\partial t}$$

To simplify things we consider a one dimensional solution ($\vec{k}$ represents the Electromagnetic wave vector)

$$E = E_x \hat{x}$$

$$B = B_y \hat{y}$$

$$\vec{k} = \vec{E} \times \vec{B} = E_x B_y \hat{z}$$

A plane wave (wave that is the same for all x and y values for any point $z = z_0$) travelling in the z-direction and polarized in the x-direction. We plug the last line above into Maxwell's Equations in a vaccum to obtain:

1.1.1 Final Formulae

The Electric field points in the x - direction, the magnetic field points in the y - direction, the EM wave propagates in the z - direction, and t represents time.

$$\frac{\partial E_x}{\partial x} = 0$$

$$\frac{\partial B_y}{\partial y} = 0$$

(1)

$$\frac{\partial B_y}{\partial t} = -\frac{\partial E_x}{\partial z}$$

$$\frac{\partial E_x}{\partial t} = -\frac{1}{\mu_0 \epsilon_0} \frac{\partial B_y}{\partial z}$$

The first two equations indicate that the E and B fields are constant along the x-y plane, proving we have a plane wave. We will not use those equations again. The second two equations indicate that the spatial evolution of one field determines the time evolution of the other. We will implement the second two equations into our stepping algorithm.

1.2 Computer Algorithm

In order to compute the derivatives $\frac{\partial}{\partial t}$, $\frac{\partial}{\partial x}$ we make approximations to arrive at a stepping algorithm:

The derivative of a function can be approximated as the linear slope of that function $m = \frac{y_2 - y_1}{x_2 - x_1}$. Another way to put it is the change in F with respect to time is equal to how much F changed over a period of time divided by that period of time.

(2)

$$\frac{\partial F(z,t)}{\partial t} \simeq \frac{F(z, t + \Delta t) - F(z,t)}{\Delta t}$$

$$\frac{\partial F(z,t)}{\partial z} \simeq \frac{F(z + \Delta z, t) - F(z - \Delta z, t)}{2\Delta z}$$

We take one step at a time in the time derivative, and two in the spatial derivative, as the algorithm will start with a solution in space and then step forward in time, thus it is necessary to calculate a value for each time step. With the spatial derivative however, we can afford to take two steps at a time for increased stability. This is because we start with an initial solution for all points in space but only one point in time.

We plug in the approximations of the derivatives (2) to the 1-D, vaccum case of Maxwell's Equations above (1), and obtain two stepping formulae (i is the spacial index, n is the time index).

1.2.1 Stepping Algorithm

The next Electric field at point i is the previous Electric field at point i minus the spatial deviation of the Magnetic field around that point i.

$$E_x[i, n+1] = E_x[i,n] - \frac{\Delta t}{2\mu_0 \epsilon_0 \Delta z}(B_y[i+1,n] - B_y[i-1,n])$$

$$B_y[i, n+1] = B_y[i,n] - \frac{\Delta t}{2\Delta z}(E_x[i+1,n] - E_x[i-1,n])$$

To implement this stepping formula we set the initial conditions of the Electric and Magnetic fields to be matching sine waves (for a plane wave the Electric and Magnetic fields are in phase).

The boundary conditions are periodic, $zmax + 1 = 0$, $0 - 1 = zmax$. The field at the right boundary is calculated using the left boundary and vice versa.

1.2.2 Interleaving Steps

For the purposes of stability we update the Electric field on even time steps, and the Magnetic field on odd time steps.

This may appear to violate the "instantaneousness" of derivatives, but it in fact reinforces it. If we update the Electric field based on the spatial change of the Magnetic field, and then incorrectly update the magnetic field based off how the Electric field *was*, then the magnetic field is in fact changing based off the past. Instead what we do is update the Electric field based on the Magnetic field, then update the Magnetic field based off the more current Electric field.

1.3 Stability

As the algorithm steps forward in time, small inaccuracies in the simulation are created through two main avenues:

1. The Algorithm takes finite time steps and space steps, while the mathematics asks for infinitely small steps.
2. Python's 64 bit float values are not exact, especially for extremely small and extremely large numbers.

These two problems do not work well together. We cannot take smaller steps because then the 64 bit representation would have more proportional error.

The Courant stability condition is commonly used in simulations such as these.

$$\frac{\Delta t}{\Delta z} \leq \frac{1}{2c}$$

In the implementation of the code we have the variable "fact", which sets this ratio, and the variables μ_0 and ϵ_0 which set the speed of light

$$fact = \frac{\Delta t}{2\Delta z} = 0.005$$

$$\frac{\Delta t}{\Delta z} = 0.01$$

$$\mu_0 = 1$$

$$\epsilon_0 = 0.5$$

$$c = \frac{1}{\sqrt{\mu_0 \epsilon_0}} = 1.41$$

$$\frac{1}{2c} = 0.35$$

$$0.01 \leq 0.35$$

So by the Courant condition our algorithm is stable.