Milestone 3

J. Evans, M. Khumalo, L. Liu

March 15, 2022

3.1:

Courant number: $S = c * dt/dz$

Then $dt = s * dz/c$

Due to the Yee's leapfrog time-stepping, using the magic step $\Delta t = \Delta z/c$ in one-dimensional FDTD implies there is a $0.5\Delta z$ (which means S=1/2) free-space wave motion between insertion of the initial E and H. If the dielectric filling of the grid is dispersive, the calculation of the wave positional shift is not straightforward, and the grid is filled with a nondispersive dielectric medium which can reduce the speed of wave motion below c for the positional shifts when S=1/2, so the S=1/2 corresponds to the magic step [Taflove & Hagness, 2000], pg 185.

3.2:

Gaussian pulse: $E(s(z_s, t)) = E_0 \exp\{(-(t - t_s)^2/(\sigma_s^2))\}, z_s = L/4, t_s = 20\Delta t, \sigma_s = 8\Delta t$

The effects of parameters can be shown by changing the Courant number. The pulse propagation for S=0.99 shows no observable difference relative to the perfect propagation case (S=1). For S=0.5, the pulse propagation shows only a slight retardation relative to the S=1 case. There is no superluminal precursor observed.

The end of the simulation time is assumed to be $t_f = 6 * \sigma$ so that over 99% of the Gaussian pulse can be realized during the simulation.

3.3:

The disadvantage of a Gaussian pulse as an representation of a photon is that it is not the true representation like a sinusoidal wave. An advantage is that the Gaussian is a mathematical construct that is the same shape in both the time domain and the frequency domain. A continuous sine wave in the time domain has a spectrum of a single line. Conversely, a single narrow pulse in the time domain has a broad spectrum. Although a true Gaussian is infinite, truncating the tails makes it finite [Daqarta, 2006].

3.4:

Please see the attached code (Yee_v14). The code was run with S=0.5 (magic time step), S=0.2, and S=0.8. The "wave packet" is formed by a superposition of waves that develop over time as the Gaussian pulse is introduced at $\frac{L}{4}$. The wave propagates in the same manner for each run with the exception of speed. When S=0.2, the wave propagates much slower. Conversely, when S=0.8, the wave propagates much faster. Figure 1 shows in detail how the wave packet forms and propagates over time. The time-induced Gaussian pulse grows at $\frac{L}{4}$ until it reaches it's normalized peak of amplitude = 1, then it reduces. The "wave packet" then splits in two and propagates in opposite directions via the Yee algorithm (the equations from Part 2.6).

3.5:

See Figure 2. The left boundary gives a bounce back, with positive amplitude. The wave on the left side then propagates forward and reflects (with negative amplitude) off of the Gaussian pulse boundary at $\frac{L}{4}$. The right boundary gives a reflection (negative amplitude) at the same time that the left boundary gives another bounce back. We believe the difference between the two sides has to do with the fact that we have different boundaries on the left and right. On the left, we hold E(0) = 0 & H(0) = 0. However, on the right, we sometimes

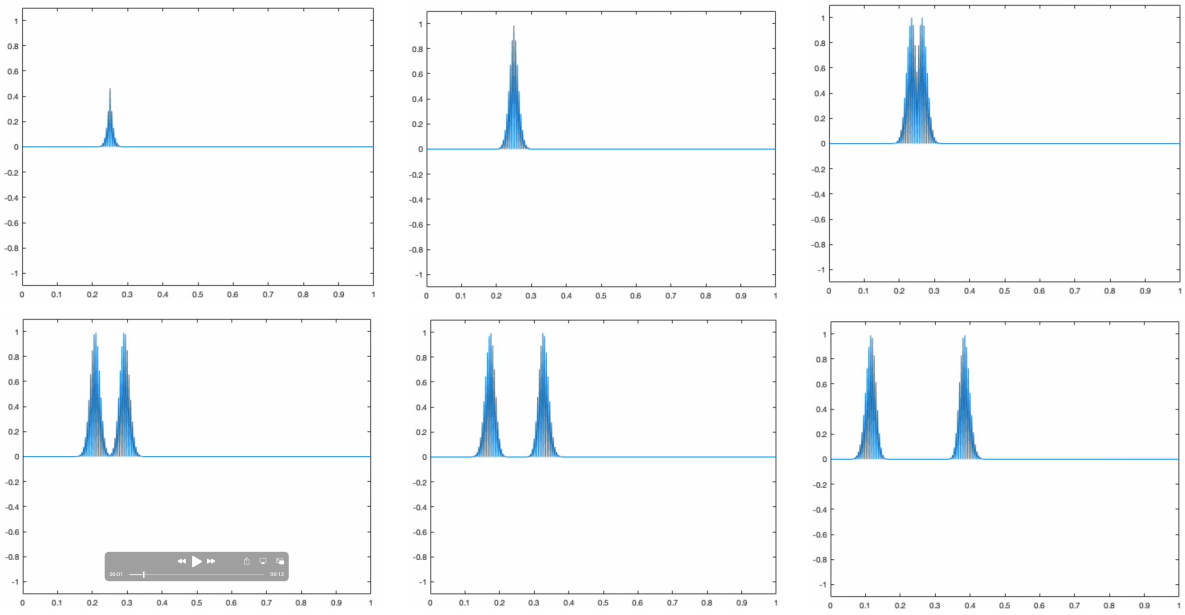


Figure 1: Propagation via Yee algorithm of a time-induced Gaussian pulse at $\frac{L}{4}$. Read from left to right. The pictures are taken at unequal time intervals. Produced by Matlab [Mathworks, 2021].

hold two zeros for the last two values of E and H, but not always. This was for utility to make the code work. However, the calculations obviously get messed up in exotic ways at these boundaries. Finally, there is a secondary reflection at $\frac{L}{4}$ because of the Gaussian function that overwrites the value at that point.

3.6:

Our boundary issues can be solved by setting appropriate boundary conditions. Part 4 of this project uses Absorbing Boundary Conditions to overcome these challenges. The magnetic field and the electric field require defined boundary conditions.

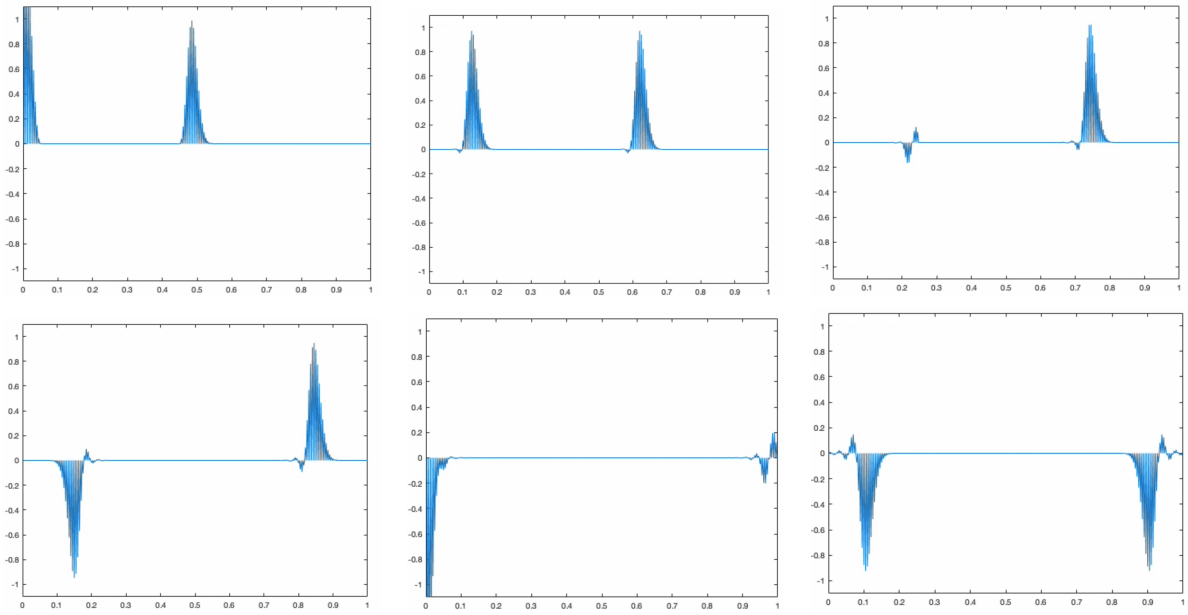


Figure 2: Boundary bounce and reflection due to lack of Absorbing Boundary Conditions. Also, notice the reflection off of the boundary at $\frac{L}{4}$ due to the continued driver of the time-induced Gaussian pulse at that location. Read from left to right. The pictures are taken at unequal time intervals. Produced by Matlab [Mathworks, 2021].

References

Daqarta (2006). Interstellar research: Data acquisition and real-time analysis.

Mathworks (2021). Matlab r2021b (9.11.0.1769968).

Taflove, A. & Hagness, S. (2000). *Computational Electrodynamics: The Finite-Difference Time-Domain Method, Second Ed.* Artech House, Inc.

Team Remix Project

Joe Evans, Maxine Khumalo, Lang Liu

Yee Algorithm

This code is built-on and modified from research work that J. Evans did in the Fall of 2020.

Version Comments

Propagation:

This is the same propagation code as Yee_v10, except that the gaussian is fed in as a pulse over time as defined in part 3 of the project. This version propagates the wave in the same manner as v10 did, but the wave is made up of a superpostion of waves caused by the pulse.

Reflection:

The left boundary gives a bounce back, with positive amplitude. The wave on the left side then propagates forward and reflects (with negative amplitude) off of the Gaussian pulse boundary at L/4. The right boundary gives a reflection (negative amplitude) at the same time that the left boundary gives another bounce back. We believe the difference between the two sides has to do with the fact that we have different boundaries on the left and right. On the left, we hold E(0) = 0 \& H(0) = 0. However, on the right, we sometimes hold two zeros for the last two values of E and H, but not always. This was for utility to make the code work. However, the calculations obviously get messed up in exotic ways at these boundaries. Finally, there is a secondary reflection at L/4 because of the Gaussian function that overwrites the value at that point.

Admin

```
clearvars
sympref('FloatingPointOutput',true);
```

Section 1: Define Constants and Courant Number

Given Constants

```
e0 = 8.86e-12; %permittivity of free space (F/m)

mu0 = 4e-7 .* pi; % permeability of free space (H/m)

er = 1; %permittivity in medium (set to 1 for free space)

mur = 1; %permeability in medium (set to 1 for free space)

e = er .* e0; % e = epsilon (Units F/m)

mu = mur .* mu0; % (Units: H/m)

c = 1./(sqrt(e0*mu0));
```

Set Courant number (S)

Set $S = \frac{c\Delta t}{\Delta z}$

Then $\Delta t = S\frac{\Delta z}{c}$

```
S = 0.5; %Courant Number of 1/2 corresponds to the magic time step

c = 3e8; % speed of light in (meters / second)

dz = 5e-3; %space increment in meters as defined in the project part 3 parameters

dt = S .* (dz./c) ;
```

Section 2: Set up the Wave

Gaussian Pulse in Time "Wave Packet"

```
sigma = 8 .* dt;  %as defined in Project Part 3

ts = 20 .* dt;

E0 = 1;

Es = @(t) E0 .* exp(-((t-ts)^2)./(sigma^2));

Z=(dz/2):(dz/2):1;    %for plotting and video purposes
```

Create Matrices to hold wave values

```


L = 1./dz;               %Number of Space Positions so that z(L) = 1

Tmax = 6 .* sigma;
T = Tmax ./ dt;           %Number of Time Steps so that over 99% of the Guassian pulse is complete

%Matrices to hold E and H values will be row vectors of length 2L, so that
%each position represents a half step.

E = zeros(1,2.*L);          %Matrix of electric field values in space (@ t)
Ein = zeros(1,2.*L);        %Temporary Placeholder for the E value input for each space step
Eout = zeros(1,2.*L);       %Temproary placeholder for the E value output for each space step

H = zeros (1,2.*L);         %Matrix of magnetic field values in space (@ t)
Hin = zeros(1,2.*L);        %Temporary placeholder for the H value input for each space step
Hout = zeros (1,2.*L);      %Temporary placeholder for the H value output for each space step

zs = L./4;                  %Define the space point of the Gaussian pulse
```

Guassian in Space

This is a Guassian in space for the E field, which can be introduced in lieu of a Guassian pulse in time for testing purposes

```
%   a = 1;     %height
%   b = 1/4;    %center
%   d = 8*dz;    %width
%   f=@(x)a*exp(-((x-b).^2)/(d^2));
%   E(:)=f(Z);
%
```

Square Wave in Space

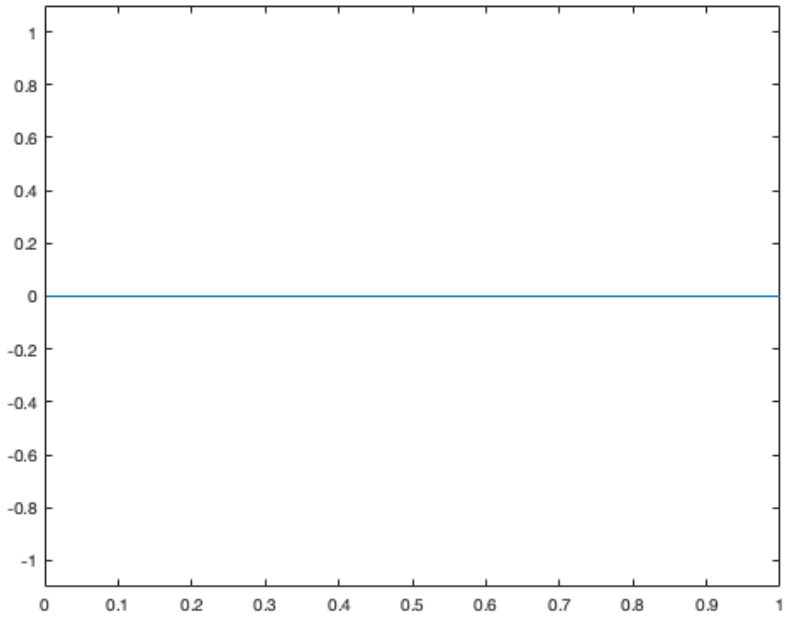
This is a Square wave in space for the E field, which can be introduced in lieu of a Gaussian pulse in time for testing purposes

```
% E(50:150)=1;
```

Section 3: Setup Video

```
v=VideoWriter("Wave","MPEG-4");
open(v)
plot(Z,E(:))
xlim([0 1]);
```

```
ylim([-1.1 1.1]);
Ylabel="Amplitude";
Xlabel="z";
axis manual
set(gca,"nextplot","replacechildren")
frame=getframe(gcf);
```




```
writeVideo(v,frame)
```

Section 4: Time-step the wave using the Yee Algorithm

This is an implementation of the equations in Part 2.6

```
for t=1:8*T                                %Iterate for T total time steps


%%%%%%%%%%%%%%%%%%%%% Calculate the E Field %%%%%%%%%%%%%%%%%%

%     E(1) = H(2);                           % ABCs
%     E(S) = H(S);

        for k=2:(2*L-1)
        Hout(k) = H(k+1);
        Hin(k)= H(k-1);
        end

        for k=2:(2*L-1)
        E(k) =  E(k) - ( S ./ er )  .* ( Hout(k) - Hin(k) );    %calculate E
        end

%%%%%%%%%%%%%%%%%%%% Introduce next E pulse Value %%%%%%%%%%%%%%%%%

    E(2*zs) = Es(t*dt); %Calculate the next value of the pulse at z = zs


%%%%%%%%%%%%%%%%%% Calculate H Field %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%     H(1) = E(1);                          % ABCs
%     H(S) = E(S-1);

    for k=2:(2*L-2)
        Eout(k) = E(k+2);
        Ein(k) = E(k);                % these values are populated with E @ t+(1/2)

    end

    for k=2:(2*L-1)

        H(k+1) = H(k+1) - (S ./ mur ) .* ( Eout(k) - Ein(k) );   %calculate H

    end

%%%%%%%%%%%%%%%%%% Plot and Write frame to video %%%%%%%%%%%%%%%%%%%%

    plot(Z,E(:));
    frame=getframe(gcf);
    pause(0.5)
    writeVideo(v,frame)

end
```

```
close(v)
```