John Hodge

ECE 5106: FDTD Class Project

05/01/16

Part 1: 1-D FDTD

Constants:

Parameter	Value
ε	1.0
μ	1.0
$c = 1/\sqrt{\varepsilon\mu}$	1.0
β	0 (for fundamental modes)
L	30
N	31
Δt	0.5
Δz	1.0
n	6
M_t	10^4 = 10000

Analytical (PEC Boundaries):

*Note: Assume beta = 0 for fundamental resonant cavity modes

General Solution: $E_x = [A\cos(k_z\, z) + B\sin(k_z\, z)]e^{-j\beta x}e^{+j\omega t}$

Boundary Conditions: E_x(z = 0; t = 0) = 0; E_x(z = 30; t = 0) = 0;

At z = 0: $E_x(z = 0; t = 0) = [A\cos(k_z\, 0) + B\sin(k_z\, 0)] = A * 1 + B * 0 = A = 0$ -> A = 0;

At z = 30: $E_x(z = 30; t = 0) = [0 * \cos(k_z\,(30)) + B\sin(k_z\,(30))] = B\sin(k_z\,(30)) = 0$

$$k_z\, L = m\,\pi\,; m = 1, 2, 3, \ldots$$

$$k_z = \frac{m\,\pi}{L} = \frac{m\,\pi}{30}$$

$$\omega^2\varepsilon\mu = k_z^2 + \beta^2 = (\frac{m\pi}{L})^2 + \beta^2$$

At beta = 0; eps = 1.0; mu = 1.0: $\omega = k_z$

$$\omega^2(1.0)(1.0) = (\frac{m\pi}{30})^2$$

$$\omega = 2\pi f \rightarrow f = \omega/(2\pi)$$

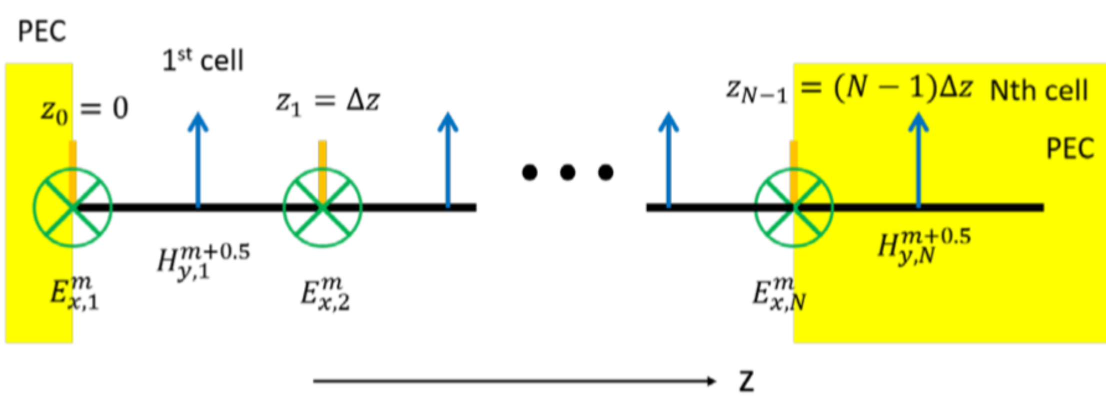
$$(2\pi f)^2(1.0)(1.0) = (\frac{m\pi}{30})^2$$

$$f_{c\,m} = \frac{1}{2\pi\sqrt{\epsilon\mu}}\left(\frac{m\pi}{30}\right) = \frac{1}{2\pi\sqrt{(1.0)(1.0)}}\left(\frac{m\pi}{30}\right) = \frac{m}{60}$$

Table 1: 1-D Cavity Resonant Modes w/ L=30 and PEC Boundaries

m	***f*** **(Hz)**	**ω (rad/s)**
1	0.0167	0.1049
2	0.0333	0.2092
3	0.0500	0.3142
4	0.0667	0.4191
5	0.0833	0.5234
6	0.1000	0.6283

1-D FDTD Simulation Set-Up:



1-D FDTD Assumptions:

Finite Difference Time Domain (FDTD) Simulations

$$\nabla \times \vec{E} = -\mu(z)\frac{\partial \vec{H}}{\partial t} - \vec{M}(z)$$

$$\nabla \times \vec{H} = \epsilon(z)\frac{\partial \vec{E}}{\partial t} + \vec{J}(z)$$

$$\vec{E} = E_x \vec{e}_x$$

$$\vec{H} = H_y \vec{e}_y$$

Specific assumptions (for this 1D problem):

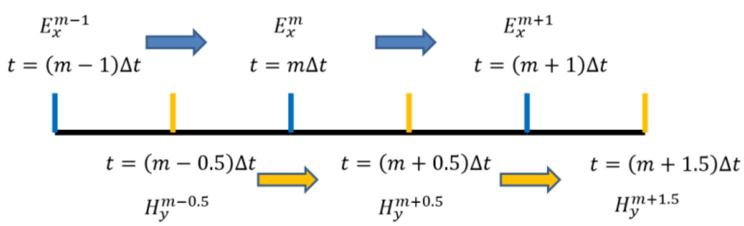
- The EM structure only contains z dependence.
- The EM field only depends on z.
- The electric field is along the x direction.
- The magnetic field is along the y direction.

$$\mu(z)\frac{\partial H_y}{\partial t} + M_y = -\frac{\partial E_x}{\partial z} \qquad \frac{\partial H_y}{\partial t} = \frac{1}{\mu(z)}\left\{-\frac{\partial E_x}{\partial z} - M_y\right\}$$

$$\epsilon(z)\frac{\partial E_x}{\partial t} + J_x = -\frac{\partial H_y}{\partial z} \qquad \frac{\partial E_x}{\partial t} = \frac{1}{\epsilon(z)}\left\{-\frac{\partial H_y}{\partial z} - J_x\right\}$$

Algorithm Overview:

1D FDTD Algorithm



Algorithm:

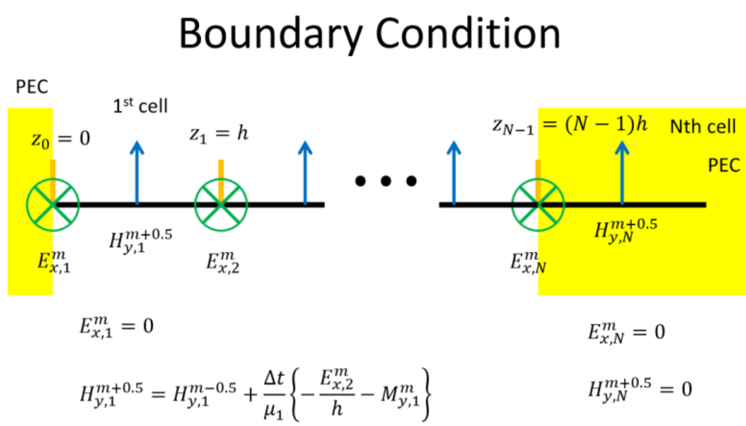
Set E and H field at t=0

Update E $\quad H_{y,n}^{m+0.5} = H_{y,n}^{m-0.5} + \frac{\Delta t}{\mu_n}\left\{-\frac{E_{x,n+1}^{m} - E_{x,n}^{m}}{h} - M_{y,n}^{m}\right\}$

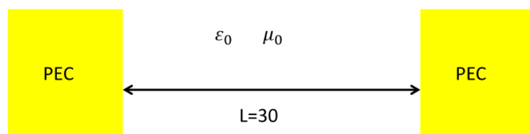
Update H $\quad E_{x,n}^{m+1} = E_{x,n}^{m} + \frac{\Delta t}{\epsilon_n}\left\{-\frac{H_{y,n}^{m+0.5} - H_{y,n-1}^{m+0.5}}{h} - J_{x,n}^{m+0.5}\right\}$

Repeat this process to obtain E and H as a function of time

Boundary Conditions:



Simulation Assignment Overview:

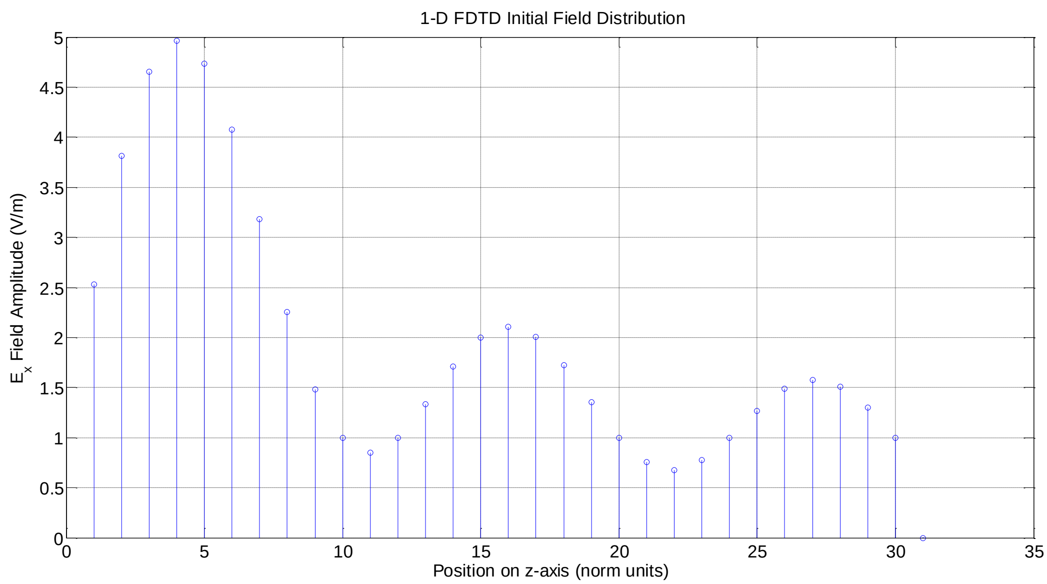


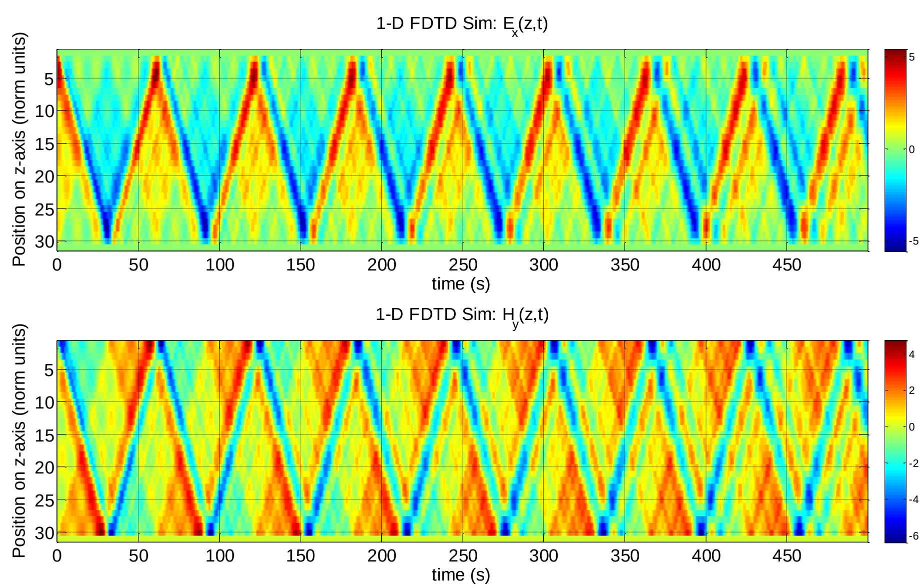
Step 1: Set up initial values for E_x and H_y, i.e., $E_x(z; t = 0)$, and $H_y(z;\ t = 0.5\Delta t)$.
Step 2: Evolve the EM fields according to the algorithm given in the previous slide.
Step 3: Perform a Fourier transform of the EM fields and obtain the EM field as a function of frequency, i.e., $E_x(z = nh; \omega)$ and $H_y(z = (n+0.5)h; \omega)$.
Step 4: The absolute value of Ex and Hy will have peaks in the frequency domain. Each of these peaks. Verify this for two lowest frequency modes.

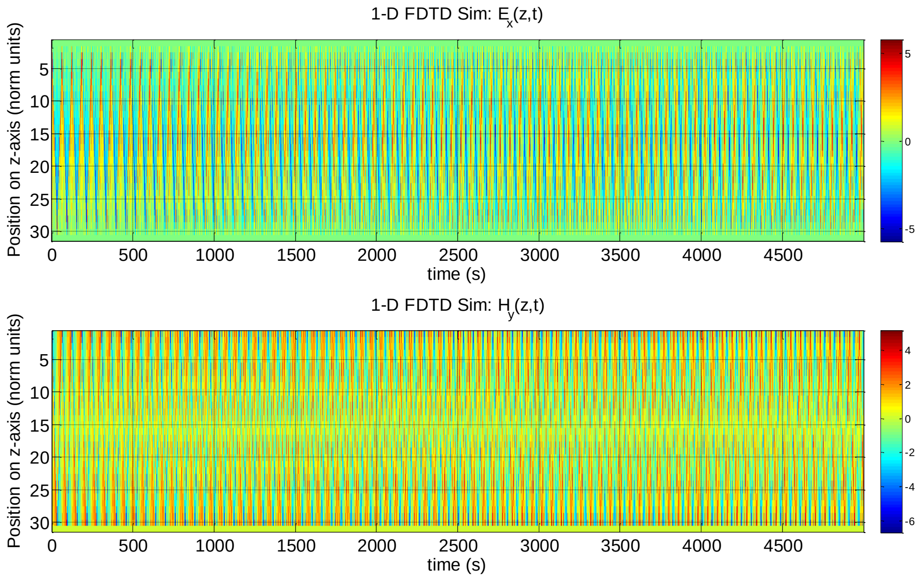
1-D FDTD Simulation Results: Source A

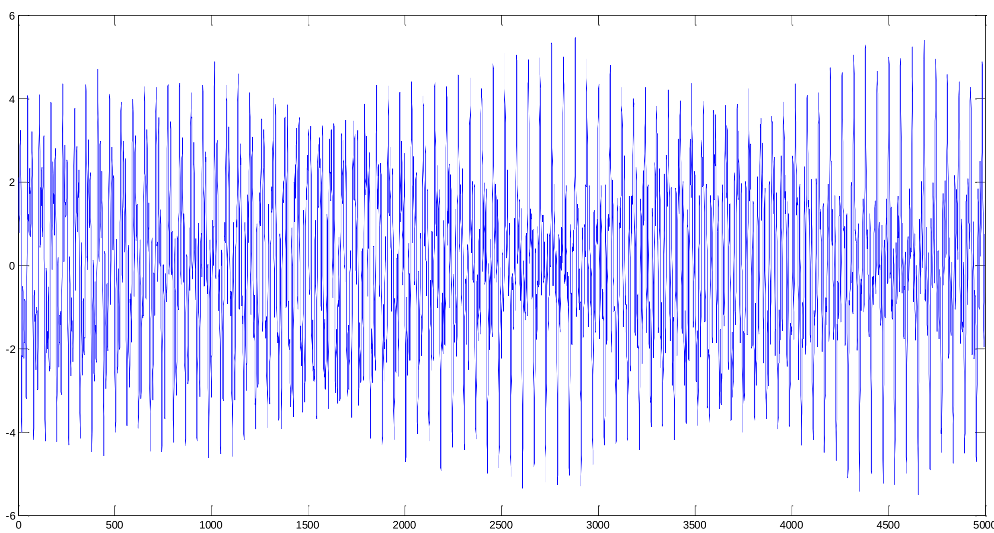
Initial Field Distribution: Summation of First Five Analytically Calculated Modes

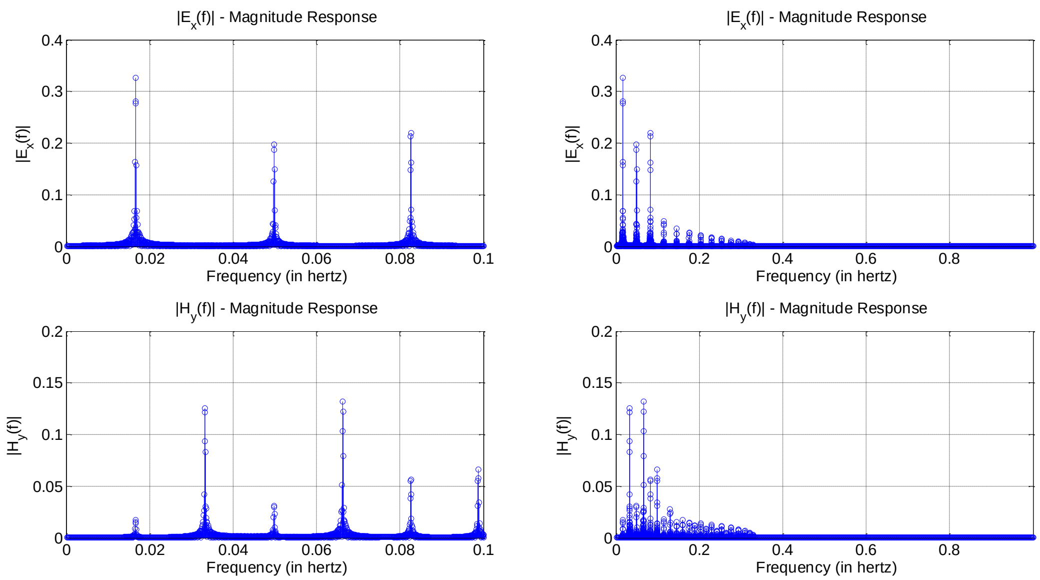
```
sig_terms = 5;
sig = ones(1, N-1);
for cc = 1:sig_terms
    sig = sig + sin((cc.*pi)./30.*(1:N-1));
end
Ex(1:end-1,2) = sig;
```

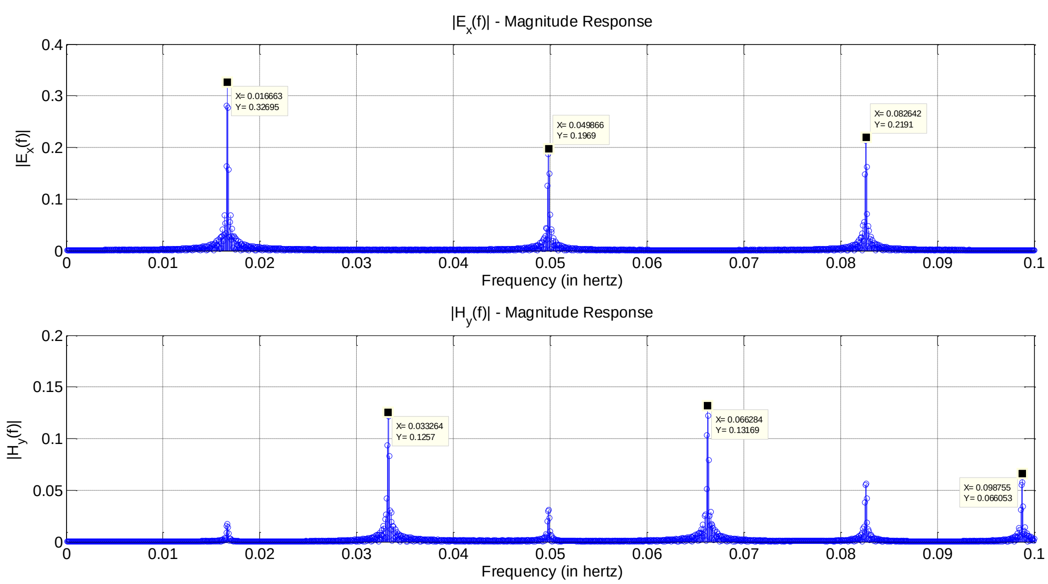

1-D FDTD Initial Field Distribution
E_x Field Amplitude (V/m)
Position on z-axis (norm units)


1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
time (s)
1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
time (s)


1-D FDTD Sim: E_x(z,t)
Position on z-axis (norm units)
time (s)
1-D FDTD Sim: H_y(z,t)
Position on z-axis (norm units)
time (s)



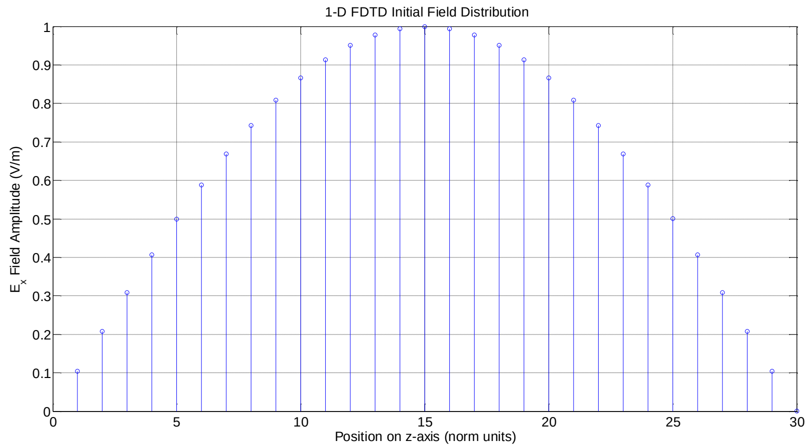


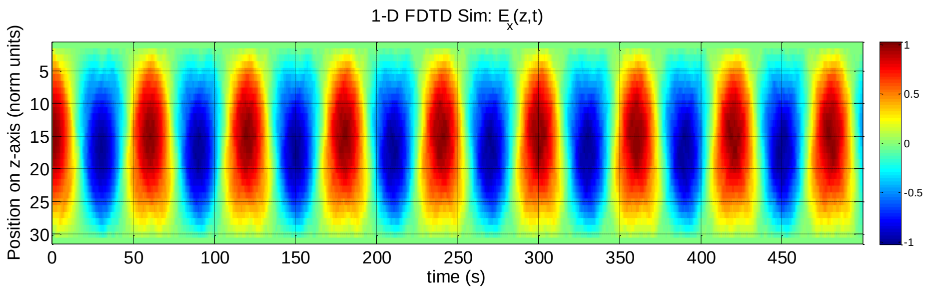


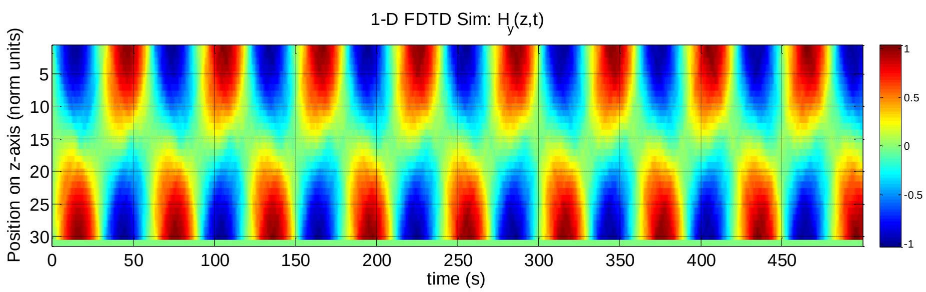
Analysis: Successfully extracts fundamental modes by taking FFT of E_x and H_y. Odd number modes are much stronger in the FFT of Ex and even number modes are much strong in FFT of Hy. Numerically calculated FFT modes are very close to analytically calculated values.

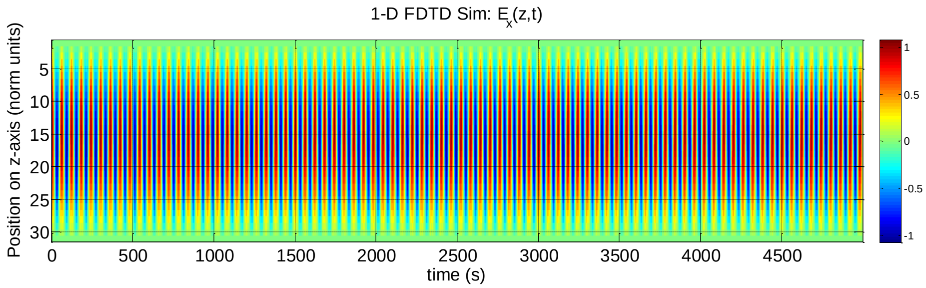
1-D FDTD Simulation Results: Source B

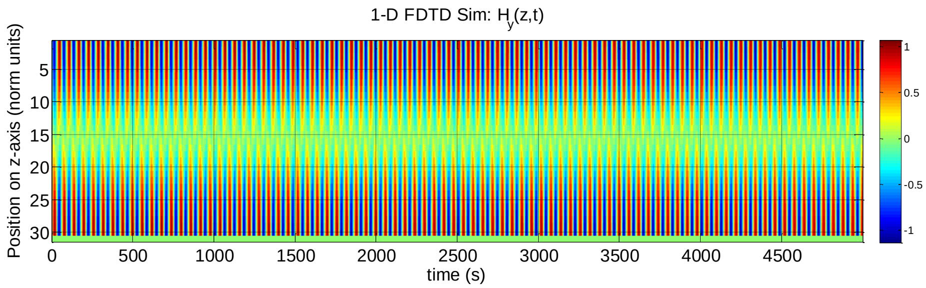
Initial Field Distribution: Ex Field of Fundamental Cavity Mode of First Order

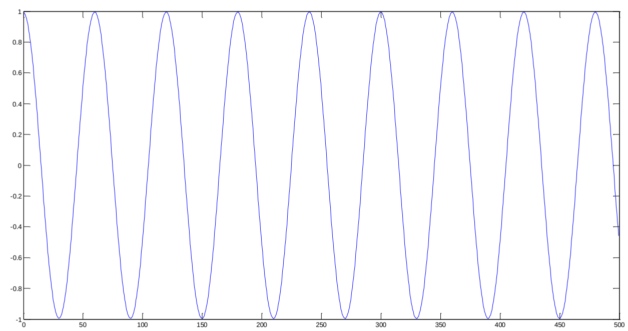

1-D FDTD Initial Field Distribution
E_x Field Amplitude (V/m)
Position on z-axis (norm units)

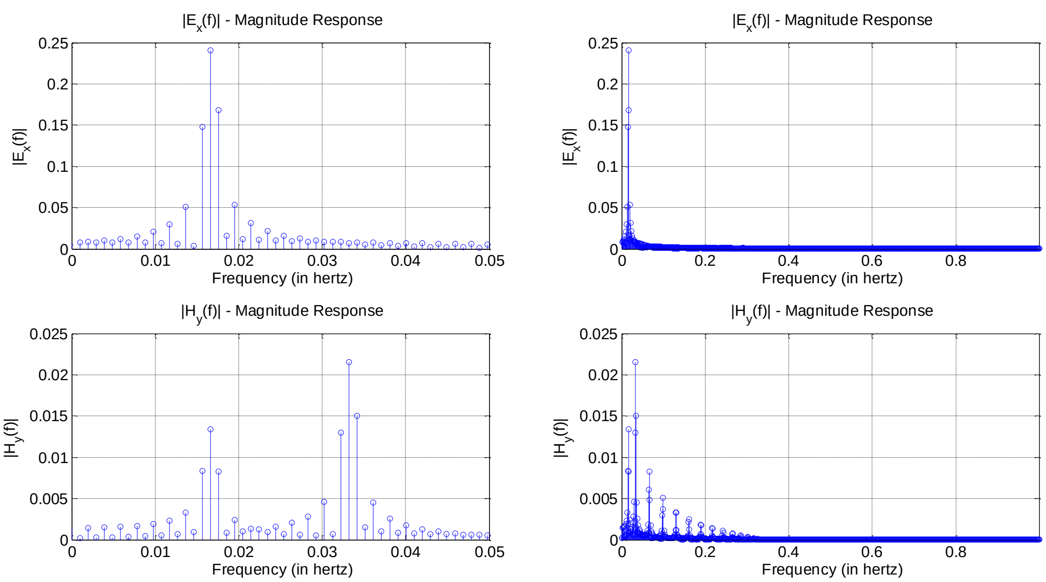

1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
time (s)

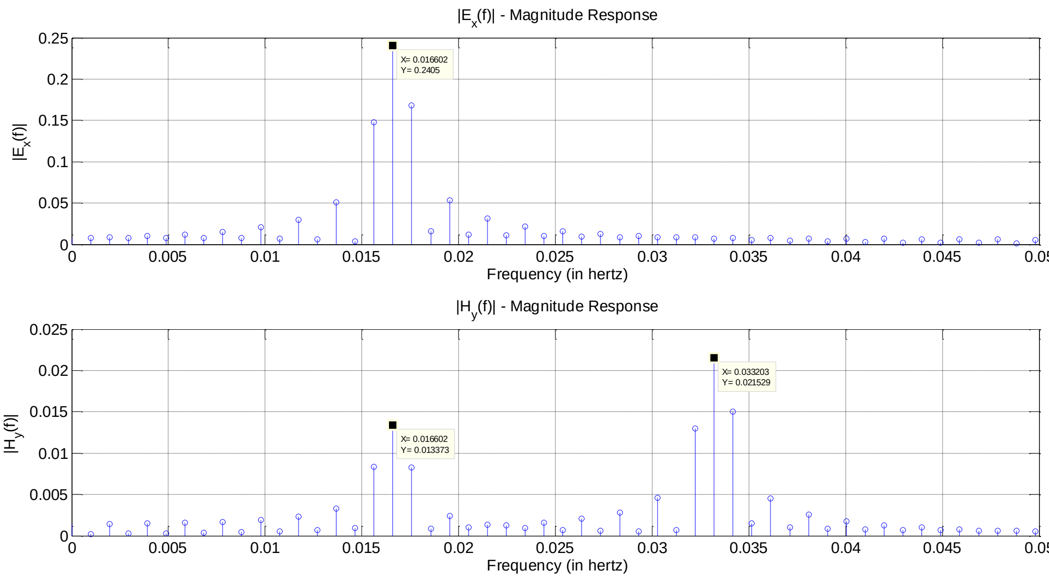

1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
time (s)


1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
time (s)


1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
time (s)

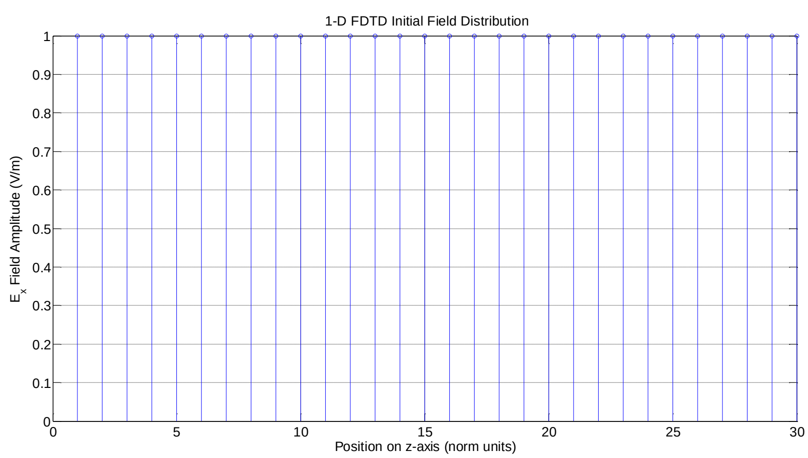


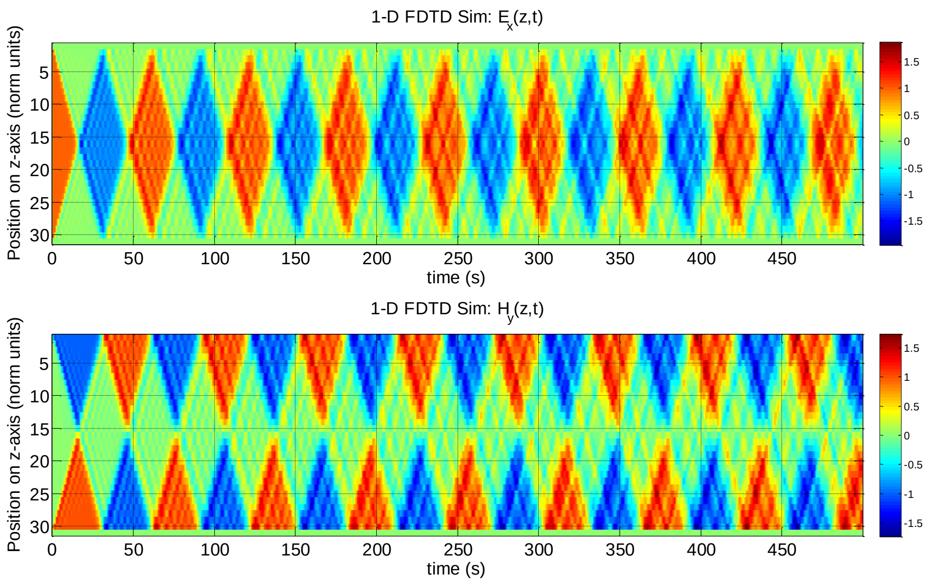

$|E_x(f)|$ - Magnitude Response
$|E_x(f)|$
Frequency (in hertz)
$|E_x(f)|$ - Magnitude Response
$|E_x(f)|$
Frequency (in hertz)
$|H_y(f)|$ - Magnitude Response
$|H_y(f)|$
Frequency (in hertz)
$|H_y(f)|$ - Magnitude Response
$|H_y(f)|$
Frequency (in hertz)

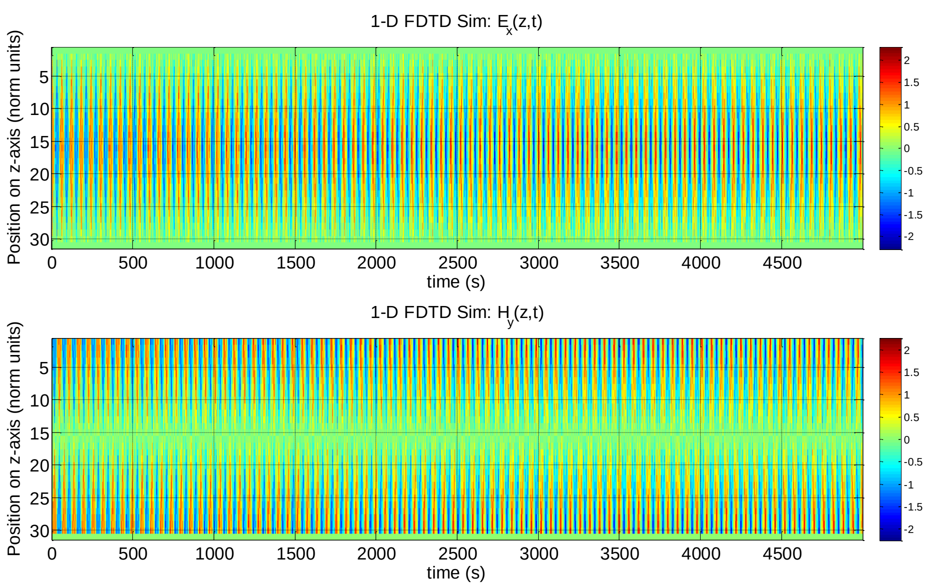


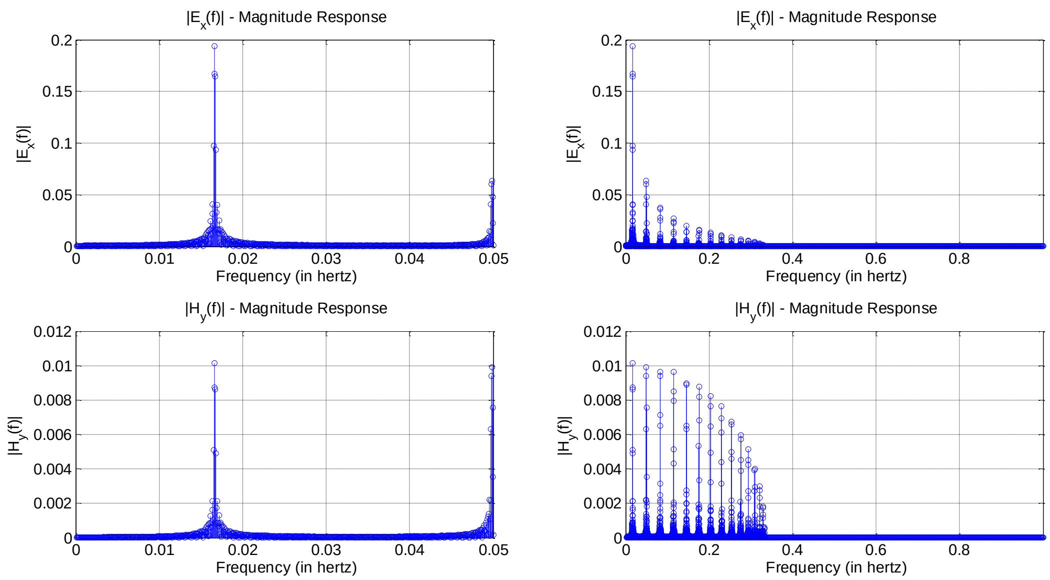
1-D FDTD Simulation Results: Source C

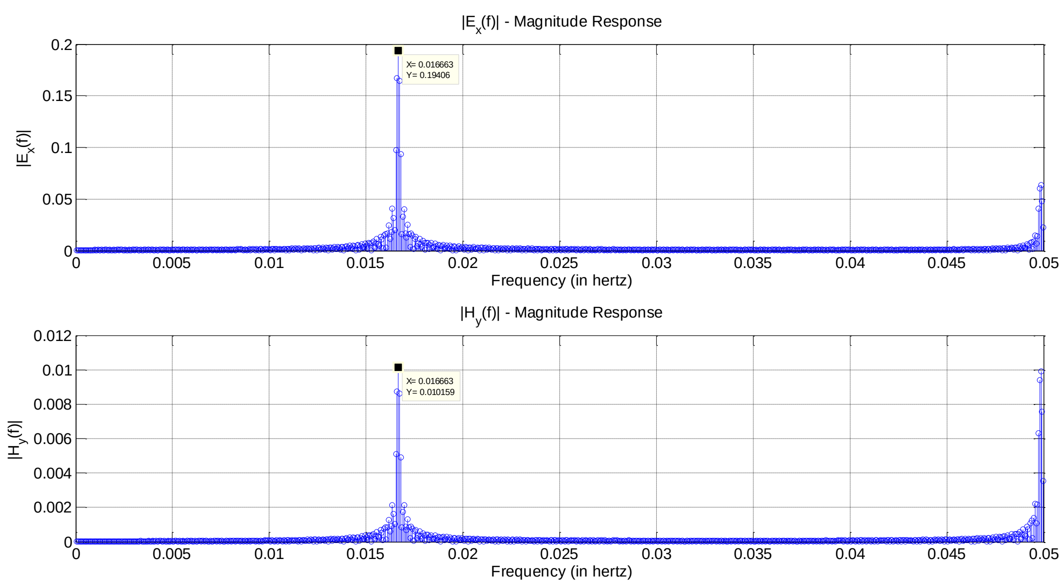
Initial Field Distribution: Uniform Ex Field




1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
5 10 15 20 25 30
0 50 100 150 200 250 300 350 400 450
time (s)
1.5 1 0.5 0 -0.5 -1 -1.5
1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
5 10 15 20 25 30
0 50 100 150 200 250 300 350 400 450
time (s)
1.5 1 0.5 0 -0.5 -1 -1.5


1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
5 10 15 20 25 30
0 500 1000 1500 2000 2500 3000 3500 4000 4500
time (s)
2 1.5 1 0.5 0 -0.5 -1 -1.5 -2
1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
5 10 15 20 25 30
0 500 1000 1500 2000 2500 3000 3500 4000 4500
time (s)
2 1.5 1 0.5 0 -0.5 -1 -1.5 -2

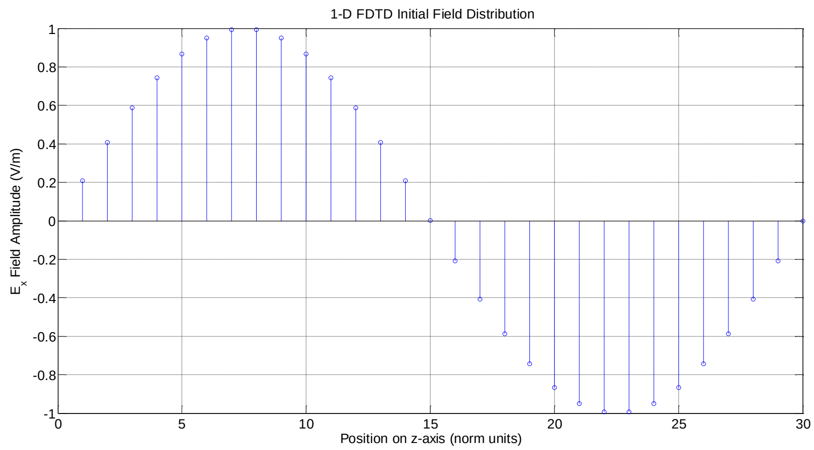


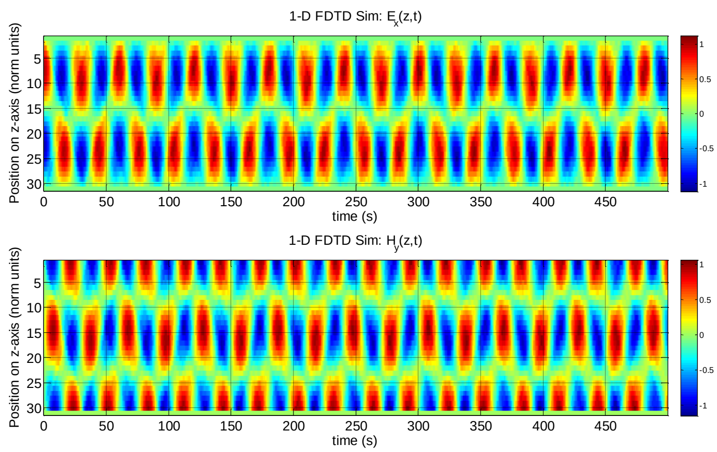


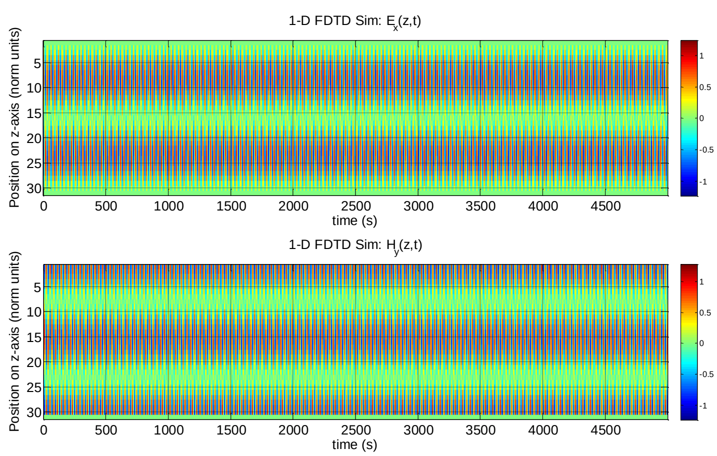
Note: Not seeing the second order (m=2) mode in this initial field distribution.

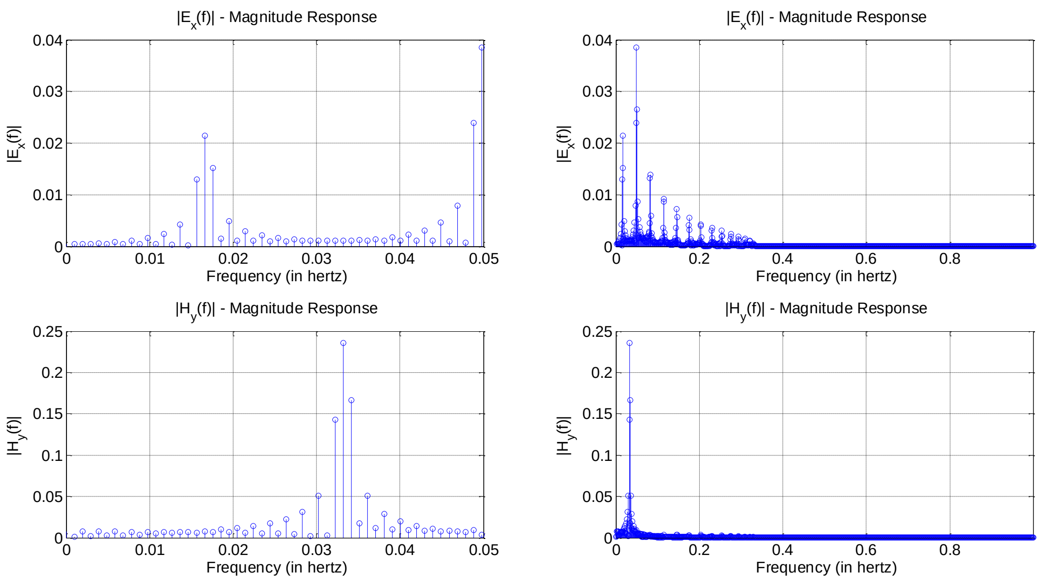
1-D FDTD Simulation Results: Source D

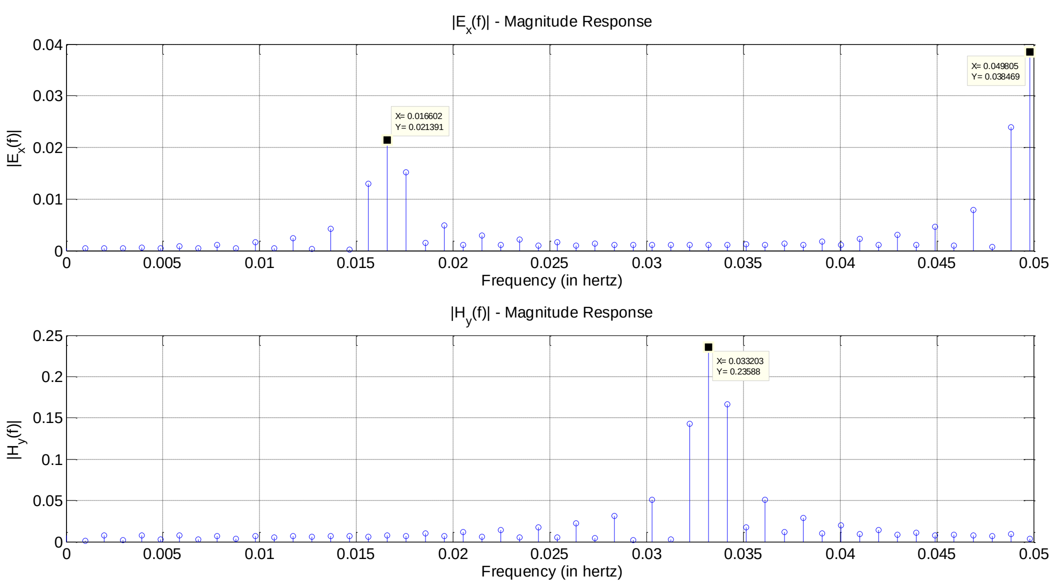
Initial Field Distribution: Ex Field of Second-Order (m=2) Cavity Mode


1-D FDTD Initial Field Distribution
E_x Field Amplitude (V/m)
Position on z-axis (norm units)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25
30


1-D FDTD Sim: E_x(z,t)
Position on z-axis (norm units)
time (s)
0
50
100
150
200
250
300
350
400
450
1-D FDTD Sim: H_y(z,t)
Position on z-axis (norm units)
time (s)


1-D FDTD Sim: E_x(z,t)
Position on z-axis (norm units)
time (s)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
1-D FDTD Sim: H_y(z,t)
Position on z-axis (norm units)
time (s)

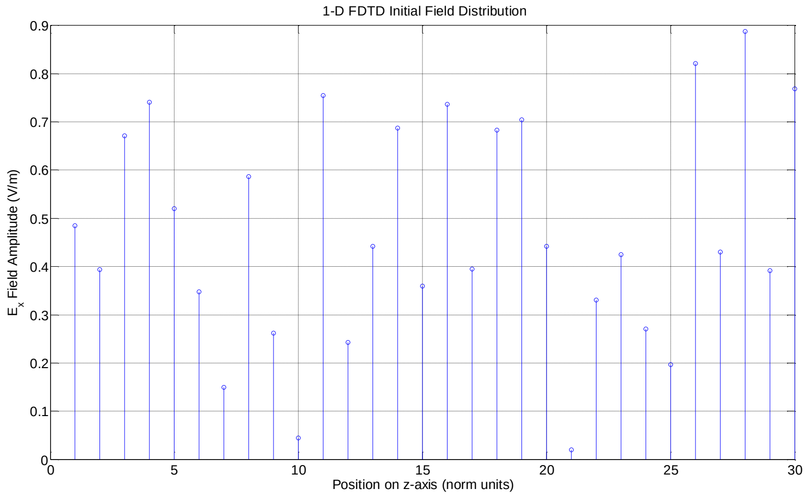




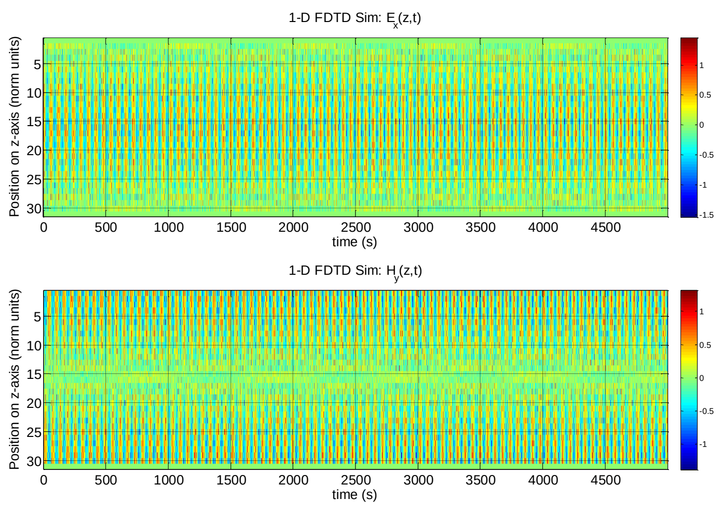
Analysis: Correct second order mode shows up in the FFT of Hy.

1-D FDTD Simulation Results: Source E

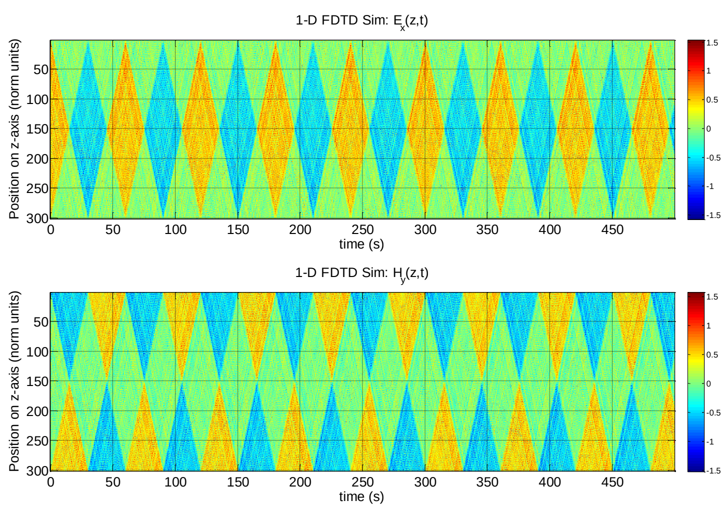
Initial Field Distribution: Randomized Ex Field



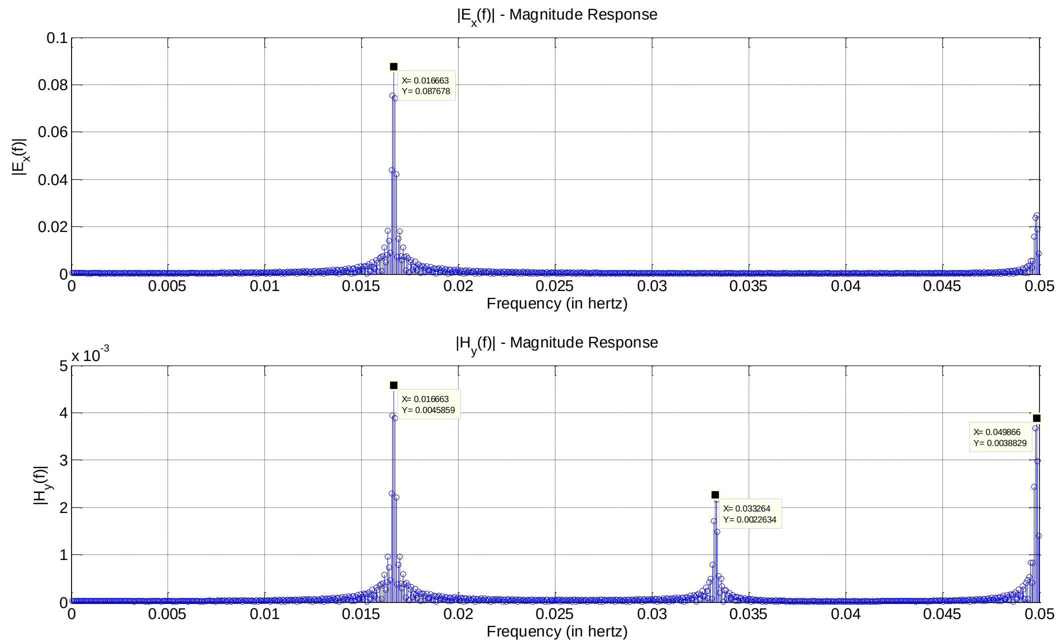
1-D FDTD Initial Field Distribution
E_x Field Amplitude (V/m)
Position on z-axis (norm units)



1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
time (s)
1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
time (s)



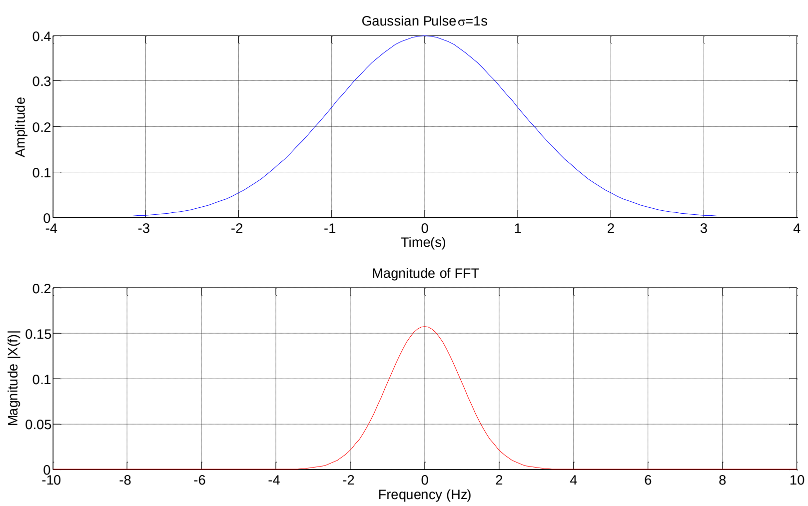
1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
time (s)
1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
time (s)



Analysis: Successfully able to extract lowest order modes in the 1-D cavity from random initial Ex field distribution.

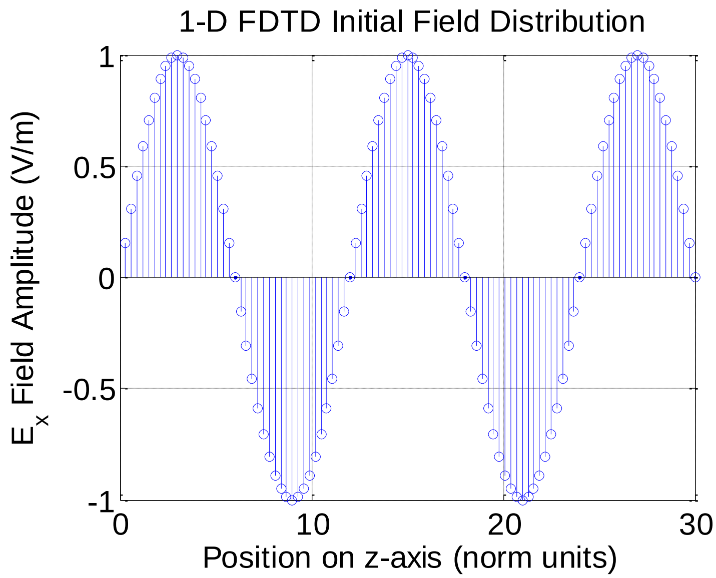
<u>Note: Gaussian Pulse Time-Varying Excitations as an Alternative Source</u>

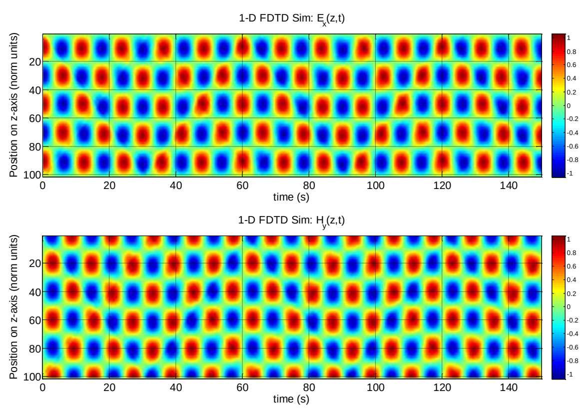
I also experimented with time-varying Gaussian pulses as wideband excitations. This is a popular and valid method as a source for arbitrary FDTD simulations. However, given the domain knowledge of the rectangular waveguide problem and the knowledge of analytical mode equations, I decided to use initial E-field and H-field distributions at t=0 as my excitation.

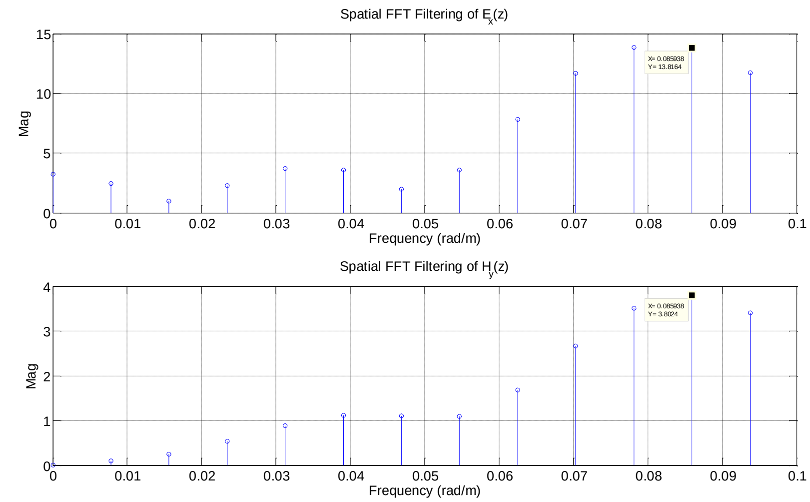


Bonus:

Bonus Case A:

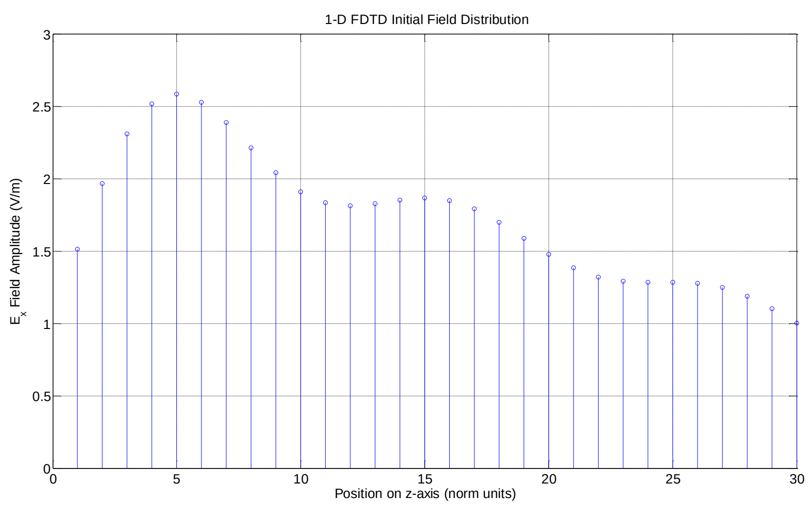


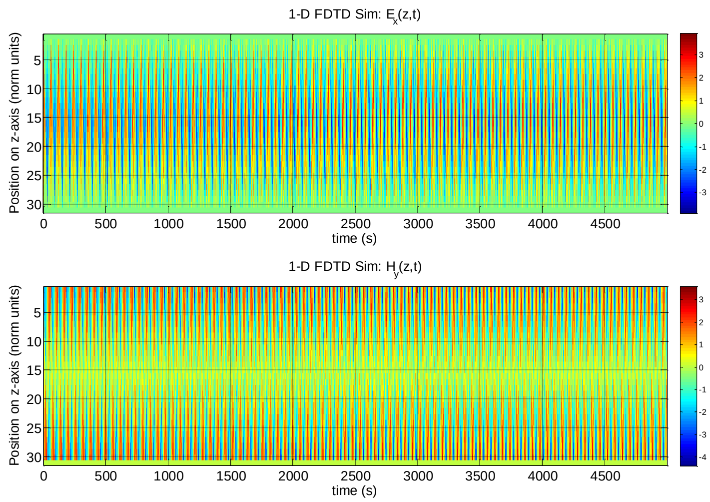


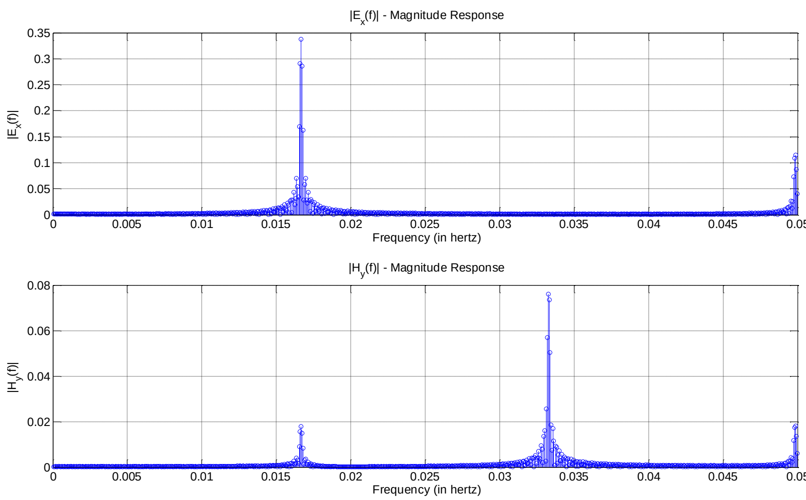


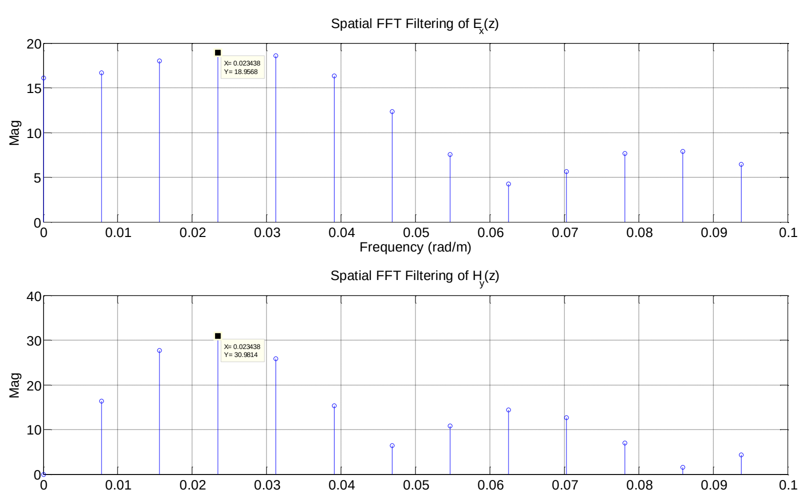
Analysis: For this well-defined mode of m=5, the spatial FFT filter comes to close to extracting to analytical value of k_z = 0.0833 rad/m. With greater spatial resolution, better results may be achievable.

Bonus Case B:



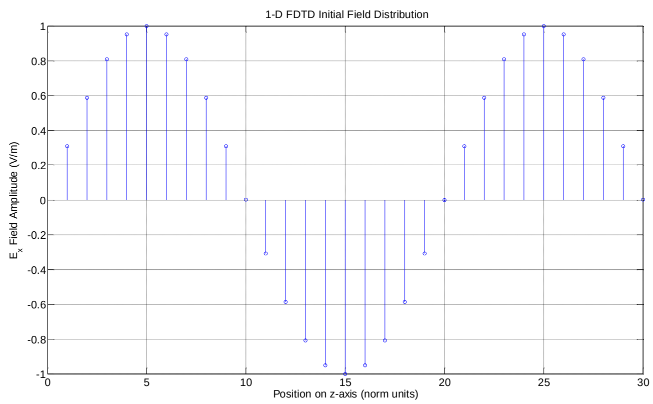

1-D FDTD Sim: $E_x(z,t)$
Position on z-axis (norm units)
time (s)
1-D FDTD Sim: $H_y(z,t)$
Position on z-axis (norm units)
time (s)

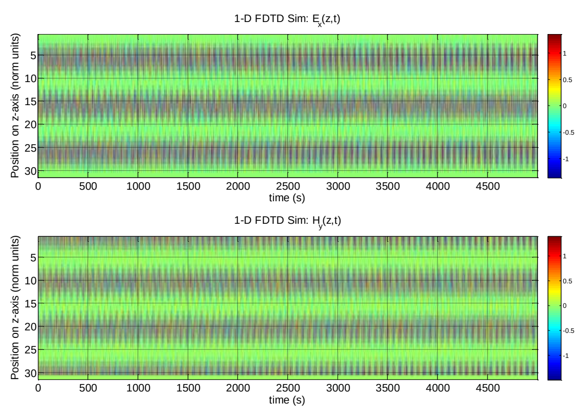

$|E_x(f)|$ - Magnitude Response
$|E_x(f)|$
Frequency (in hertz)
$|H_y(f)|$ - Magnitude Response
$|H_y(f)|$
Frequency (in hertz)

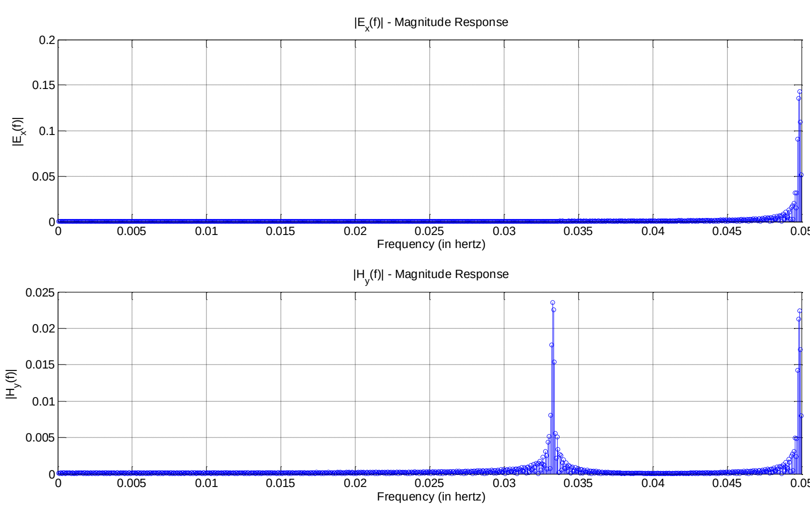

Spatial FFT Filtering of $E_x(z)$
Mag
Frequency (rad/m)
Spatial FFT Filtering of $H_y(z)$
Mag
Frequency (rad/m)

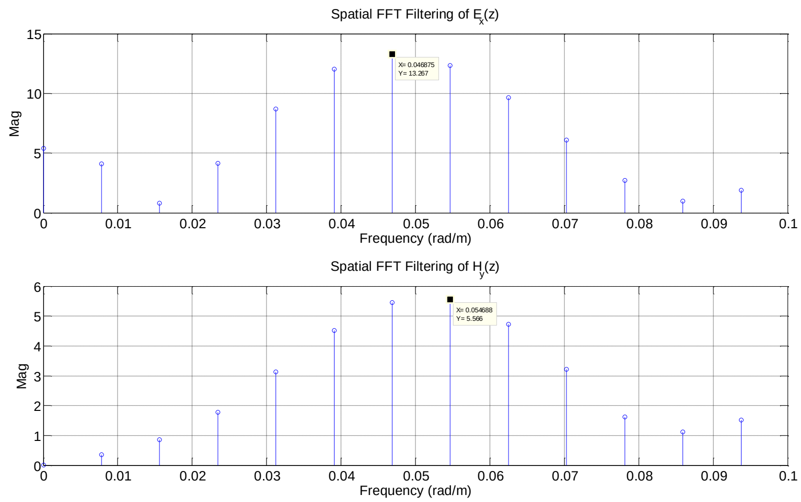
Analysis: For this case, it does not seem like there is sufficient frequency resolution w/ N=30 to precisely extract the lowest order frequency modes.

<u>Bonus Case C:</u>





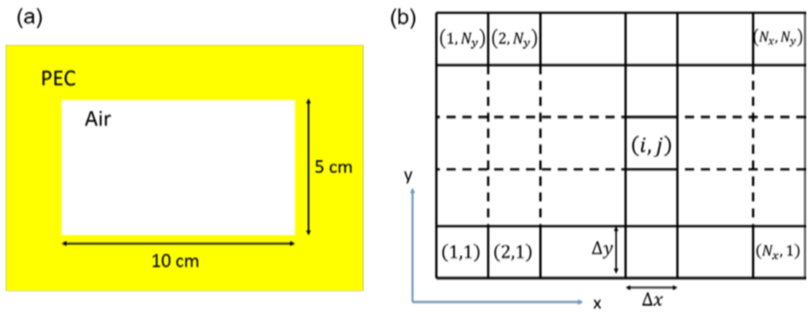




Analysis: For this well-defined mode of m=3, the spatial FFT filter comes to close to extracting to analytical value of k_z = 0.05 rad/m.

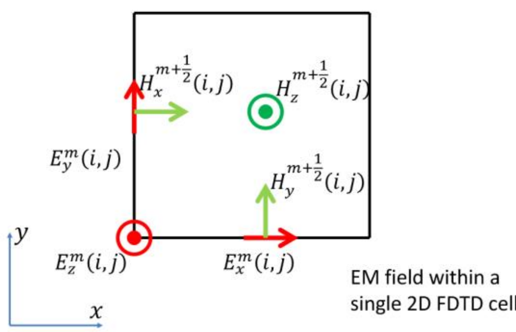
Part 2: 2-D FDTD

2-D Waveguide Diagram:



FDTD 2-D Cell Definition Using Yee's Method:

Two Dimensional (2D) FDTD



Maxwell's Equations Reduced to Two Dimensions:

Two Dimensional (2D) FDTD Algorithm

Assumptions: All EM fields are uniform (i.e., constant) along the z direction.

$$-\mu\frac{\partial \vec{H}}{\partial t} = \nabla \times \vec{E}$$

$$\varepsilon\frac{\partial \vec{E}}{\partial t} = \nabla \times \vec{H}$$

We can again divide EM field into two classes of polarizations: TE and TM, which were discussed last semester. Notice that these designations may not be the same for different sub-areas in EM.

TE: $E_z \quad H_x \quad H_y$

TM: $H_z \quad E_x \quad E_y$

$$\frac{\partial \vec{H}}{\partial t} = -\frac{1}{\mu}\left[\frac{\partial E_z}{\partial y}\vec{e}_x - \frac{\partial E_z}{\partial x}\vec{e}_y + \left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}\right)\vec{e}_z\right]$$

$$\frac{\partial \vec{E}}{\partial t} = \frac{1}{\varepsilon}\left[\frac{\partial H_z}{\partial y}\vec{e}_x - \frac{\partial H_z}{\partial x}\vec{e}_y + \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right)\vec{e}_z\right]$$

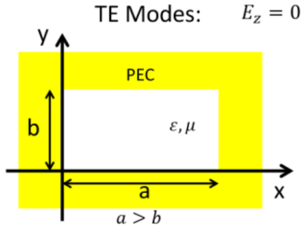
Analytical Derivation of TE-PEC Rectangular Waveguide Resonant Frequency Modes:

$$\nabla_t^2 H_z + k_t^2 H_z = 0 \qquad \frac{\partial H_z}{\partial n} = 0, on\ \Gamma$$

$$\nabla_t^2 E_z + k_t^2 E_z = 0 \qquad E_z = 0, on\ \Gamma$$

Uniform Rectangular Waveguide: TE Modes

TE Modes: $E_z = 0 \qquad H_z \neq 0 \qquad \frac{\partial H_z}{\partial n} = 0, on\ \Gamma$



$$\frac{\partial^2 H_z}{\partial x^2} + \frac{\partial^2 H_z}{\partial y^2} + k_t^2 H_z = 0$$

Boundary Conditions:

@ x=0 and x=a: $\frac{\partial H_z}{\partial x} = 0$

@ y=0 and y=b: $\frac{\partial H_z}{\partial y} = 0$

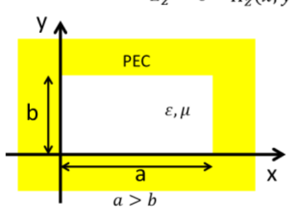
$$H_z = [A\cos(k_x x) + B\sin(k_x x)][C\cos(k_y y) + D\sin(k_y y)]e^{j\omega t}e^{-j\beta z}$$

$\frac{\partial H_z}{\partial x} = 0 \Rightarrow B = 0$, $k_x a = m\pi$, $m = 0,1,2,\cdots$

$\frac{\partial H_z}{\partial y} = 0 \Rightarrow D = 0$, $k_y b = n\pi$, $n = 0,1,2,\cdots$

Uniform Rectangular Waveguide: TE Modes

TE Modes: $E_z = 0 \quad H_z(x,y,z,t) = H_0\cos(k_x x)\cos(k_y y)\,e^{j\omega t}e^{-j\beta_{mn} z}$



$$k_x = \frac{m\pi}{a} \qquad k_y = \frac{n\pi}{b}$$

$m = 0,1,2,\cdots \qquad n = 0,1,2,\cdots$

<u>m and n cannot both be zero.</u>

$$\beta_{mn} = \sqrt{\omega^2\varepsilon\mu - \left(\frac{m\pi}{a}\right)^2 - \left(\frac{n\pi}{b}\right)^2}$$

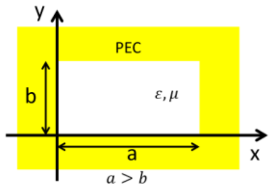
Now let us consider the transverse components.

$$\vec{E}_t = \frac{-j}{\omega^2\varepsilon\mu - \beta^2}[\beta\nabla_t E_z - \omega\mu\vec{e}_z \times \nabla_t H_z] \qquad \vec{H}_t = \frac{-j}{\omega^2\varepsilon\mu - \beta^2}[\beta\nabla_t H_z + \omega\varepsilon\vec{e}_z \times (\nabla_t E_z)]$$

$$\Rightarrow \quad \vec{E}_t = \frac{j\omega\mu}{\omega^2\varepsilon\mu - \beta_{mn}^2}\vec{e}_z \times \nabla_t H_z \qquad \vec{H}_t = \frac{-j\beta_{mn}}{\omega^2\varepsilon\mu - \beta_{mn}^2}\nabla_t H_z$$

Uniform Rectangular Waveguide: TE Modes

TE Modes: $E_z = 0 \quad H_z(x,y,z,t) = H_0 \cos(k_x x)\cos(k_y y)\, e^{j\omega t} e^{-j\beta_{mn} z}$



$$k_x = \frac{m\pi}{a} \qquad k_y = \frac{n\pi}{b}$$

$$m = 0,1,2,\cdots \qquad n = 0,1,2,\cdots$$

$$\vec{E}_t = \frac{j\omega\mu}{\omega^2\varepsilon\mu - \beta_{mn}^2}\vec{e}_z \times \nabla_t H_z$$

$$\vec{H}_t = \frac{-j\beta_{mn}}{\omega^2\varepsilon\mu - \beta_{mn}^2}\nabla_t H_z$$

The forms of the transverse components mean that m=0 and n=0 is no longer a viable choice. This is because if m=0 and n=0, we have

$$\vec{H}_t = 0 \quad \vec{E}_t = 0$$

Therefore, among TE and TM modes, the one with the lowest cutoff frequency has m=1, n=0, and is a TE mode. For TE10 mode, its cutoff frequency is given by:

$$f_{10}^{cutoff} = \frac{1}{2a\sqrt{\varepsilon\mu}}$$

$$k_x\, a = m\,\pi\,; m = 0, 1, 2, 3, \ldots;\ k_y\, b = n\,\pi\,; n = 0, 1, 2, 3, \ldots$$

*Note: m and n cannot both be zero

$$k_x = \frac{m\,\pi}{a} = \frac{m\,\pi}{0.10\,meters}\ ;\ k_x = \frac{n\,\pi}{b} = \frac{n\,\pi}{0.05\,meters}$$

$$\omega^2\varepsilon\mu = k_x^2 + k_y^2 + \beta^2 = (\frac{m\pi}{a})^2 + (\frac{n\pi}{b})^2 + \beta^2$$

At beta = 0; eps = 1.0; mu = 1.0:

$$\omega^2(1.0)(1.0) = (\frac{m\pi}{0.10})^2 + (\frac{n\pi}{0.05})^2$$

$$\omega = 2\pi f \rightarrow f = \omega/(2\pi)$$

$$(2\pi f)^2(1.0)(1.0) = (\frac{m\pi}{0.10})^2 + (\frac{n\pi}{0.05})^2$$

$$f_{c\,m,n} = \frac{1}{2\pi\sqrt{\epsilon\mu}}\sqrt{(\frac{m\pi}{a})^2 + (\frac{n\pi}{b})^2} = \frac{1}{2\pi\sqrt{\epsilon\mu}}\sqrt{(\frac{m\pi}{0.10})^2 + (\frac{n\pi}{0.05})^2}$$

Lowest order mode -> TE_{10}

$$f_{c\,1,0} = \frac{1}{2\pi\sqrt{(1.0)(1.0)}}\sqrt{(\frac{1\pi}{0.10})^2 + (\frac{0\pi}{0.05})^2} = \frac{1}{2}\left(\frac{1}{0.10}\right) = \frac{1}{0.20} = 5\,Hz$$

Fundamental TE Modes:

m	n	f (Hz)	ω (rad/s)
1	0	5.0000	31.4159
0	1	10.0000	62.8319
2	0	10.0000	62.8319
1	1	11.1803	70.2482
2	1	14.1421	88.8577
0	2	20.0000	125.664
1	2	20.6155	129.531
2	2	22.3607	140.496

2-D FDTD Simulation Parameters:

```
boundary = 'PEC';
eps_0 = 1;
mu_0 = 1;
c = 1/sqrt(eps_0*mu_0);
L = 30;
N = 31;
dz = 1;
h = dz;
Nx = 51; % Number of Steps in X
Ny = 51; % Number of Steps in Y
Lx = 0.1; % Length of Waveguide in X
Ly = 0.05; % Length of Waveguide in Y
```

2-D FDTD Field Update Equations:

H-Field:

$$H_x^{m+\frac{1}{2}}(i,j) = H_x^{m-\frac{1}{2}}(i,j) - \frac{\Delta t}{\mu}\left\{\frac{E_z^m(i,j+1) - E_z^m(i,j)}{\Delta y} + j\beta E_y^m(i,j)\right\}$$

$$H_y^{m+\frac{1}{2}}(i,j) = H_y^{m-\frac{1}{2}}(i,j) + \frac{\Delta t}{\mu}\left\{\frac{E_z^m(i+1,j) - E_z^m(i,j)}{\Delta x} + j\beta E_x^m(i,j)\right\}$$

$$H_z^{m+\frac{1}{2}}(i,j) = H_z^{m-\frac{1}{2}}(i,j) + \frac{\Delta t}{\mu}\left\{\frac{E_x^m(i,j+1) - E_x^m(i,j)}{\Delta y} - \frac{E_y^m(i+1,j) - E_y^m(i,j)}{\Delta x}\right\}$$

E-Field:

$$E_x^{m+1}(i,j) = E_x^m(i,j) + \frac{\Delta t}{\varepsilon}\left\{\frac{H_z^{m+\frac{1}{2}}(i,j) - H_z^{m+\frac{1}{2}}(i,j-1)}{\Delta y} + j\beta H_y^{m+\frac{1}{2}}(i,j)\right\}$$

$$E_y^{m+1}(i,j) = E_y^m(i,j) - \frac{\Delta t}{\varepsilon}\left\{\frac{H_z^{m+\frac{1}{2}}(i,j) - H_z^{m+\frac{1}{2}}(i-1,j)}{\Delta x} + j\beta H_x^{m+\frac{1}{2}}(i,j)\right\}$$

$$E_z^{m+1}(i,j) = E_z^m(i,j) + \frac{\Delta t}{\varepsilon}\left\{\frac{H_y^{m+\frac{1}{2}}(i,j) - H_y^{m+\frac{1}{2}}(i-1,j)}{\Delta x} - \frac{H_x^{m+\frac{1}{2}}(i,j) - H_x^{m+\frac{1}{2}}(i,j-1)}{\Delta y}\right\}$$

Boundary Conditions PEC:

```
% Upper X
Hx(2:end-1,Ny,m) = 0; Hy(2:end-1,Ny,m) = 0; Hz(2:end-1,Ny,m) = 0;
Ex(2:end-1,Ny,m) = 0; Ey(2:end-1,Ny,m) = 0; Ez(2:end-1,Ny,m) = 0;

% Lower X
Hy(2:end-1,1,m) = 0;
Ex(2:end-1,1,m) = 0; Ez(2:end-1,1,m) = 0;

% Left Y
Hx(1,2:end-1,m) = 0;
Ey(1,2:end-1,m) = 0; Ez(1,2:end-1,m) = 0;

% Right Y
Hx(Nx,2:end-1) = 0; Hy(Nx,2:end-1) = 0; Hz(Nx,2:end-1) = 0;
Ex(Nx,2:end-1,m) = 0; Ey(Nx,2:end-1,m) = 0;  Ez(Nx,2:end-1,m) = 0;

% (1,1)
Hx(1,1,m) = 0; Hy(1,1,m) = 0;
Ex(1,1,m) = 0; Ey(1,1,m) = 0; Ez(1,1,m) = 0;

% (Nx,1)
Hx(Nx,1,m) = 0; Hy(Nx,1,m) = 0; Hz(Nx,1,m) = 0;
Ex(Nx,1,m) = 0; Ey(Nx,1,m) = 0; Ez(Nx,1,m) = 0;

% (1,Ny)
Hx(1,Ny,m) = 0; Hy(1,Ny,m) = 0; Hz(1,Ny,m) = 0;
Ex(1,Ny,m) = 0; Ey(1,Ny,m) = 0; Ez(1,Ny,m) = 0;
```

```
% (Nx,Ny)
Hx(Nx,Ny,m) = 0; Hy(Nx,Ny,m) = 0; Hz(Nx,Ny,m) = 0;
Ex(Nx,Ny,m) = 0; Ey(Nx,Ny,m) = 0; Ez(Nx,Ny,m) = 0;
```

Stability Criterion:

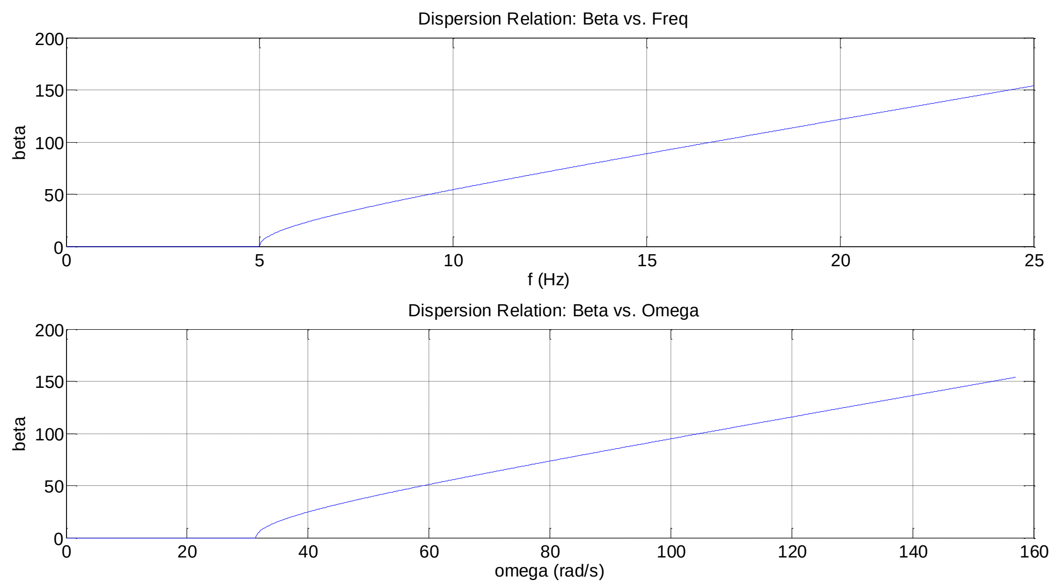
```
%% Calculate Stability for FDTD
t_stab = 1 ./ (c .* sqrt(1/(dx).^2 + 1/(dy).^2));
```

$$\Delta t \leq \frac{1}{c\sqrt{(\frac{1}{\Delta x})^2 + (\frac{1}{\Delta y})^2}}$$

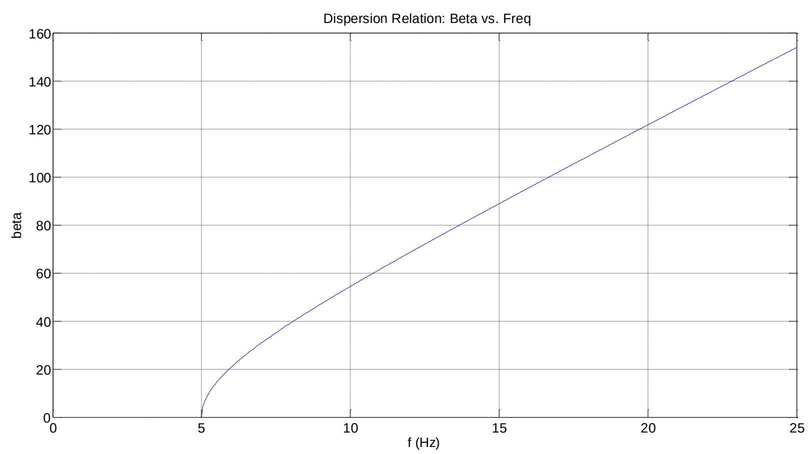
Note: *For this project, I have kept Δt less than or equal to 0.5 of the criterion to improve simulation accuracy

Dispersion Relation:

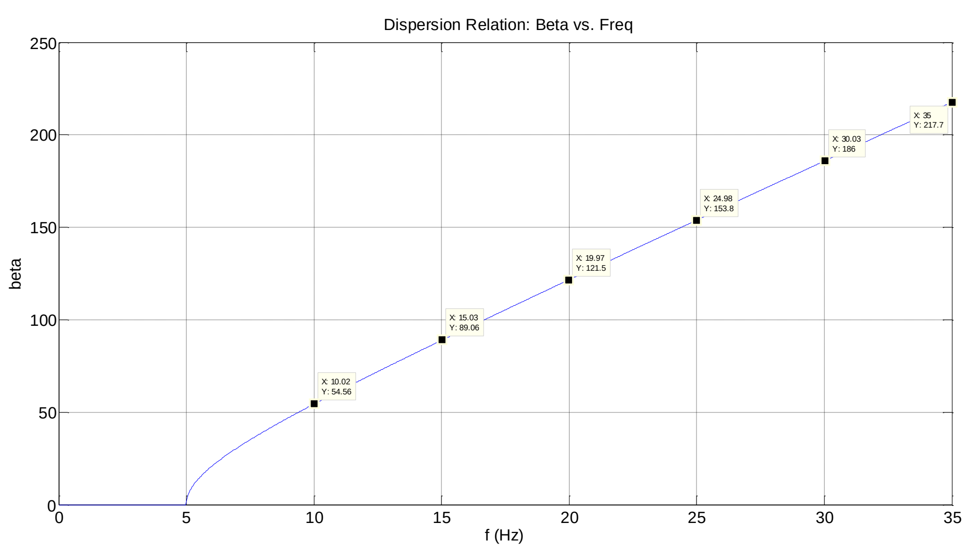
```
eps = 1;
mu = 1;
a = 0.1;
b = 0.05;
m = 1;
n = 0;
kx = m*pi/a;
ky = n*pi/b;
f = linspace(0,25, 1000);
w = 2*pi*f;
beta = sqrt(w.^2.*eps.*mu - kx.*kx - ky.*ky);
```



Dispersion Relation: Beta vs. Freq
beta
f (Hz)
Dispersion Relation: Beta vs. Omega
beta
omega (rad/s)



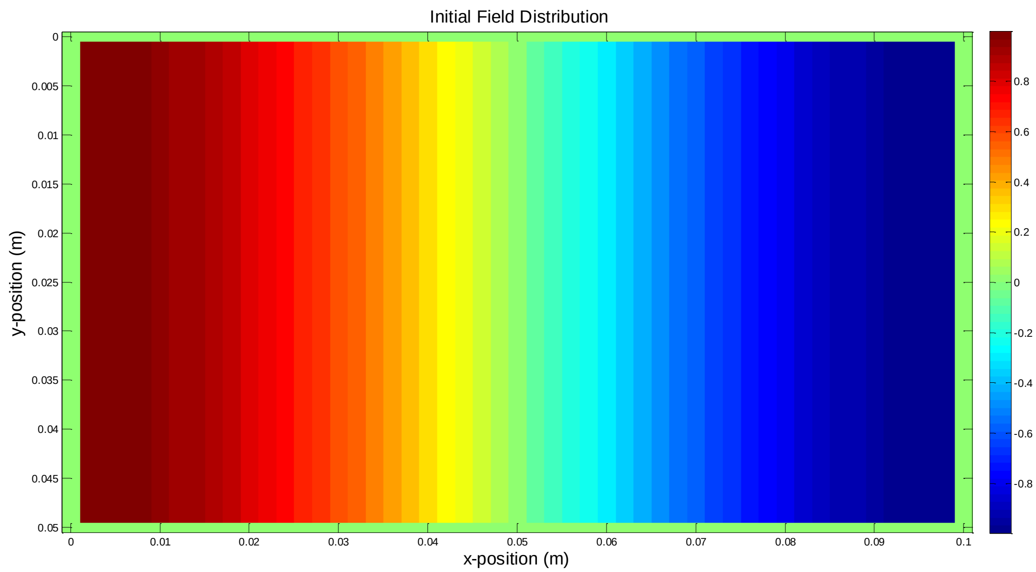
Dispersion Relation: Beta vs. Freq
beta
f (Hz)

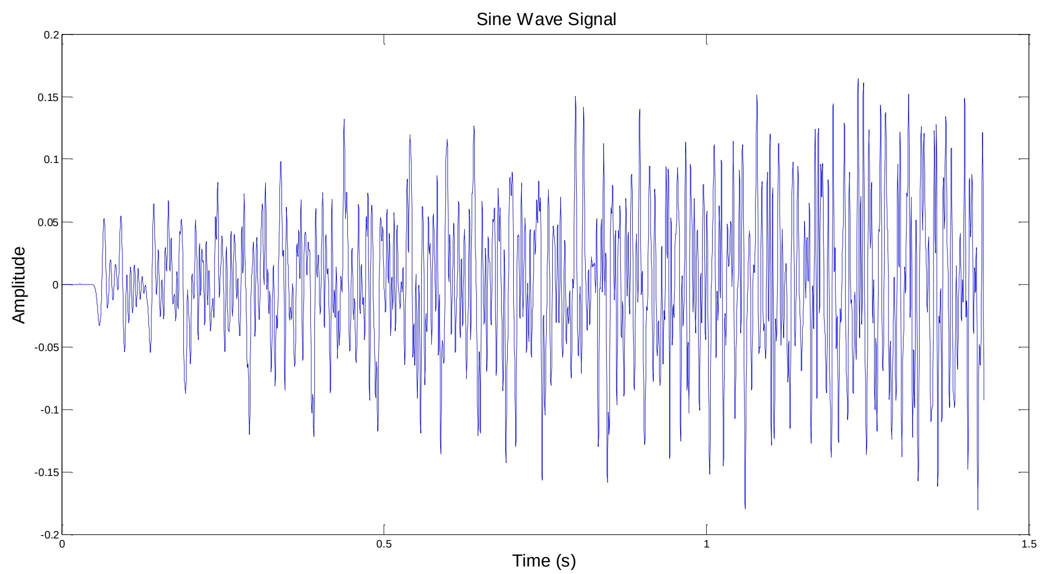


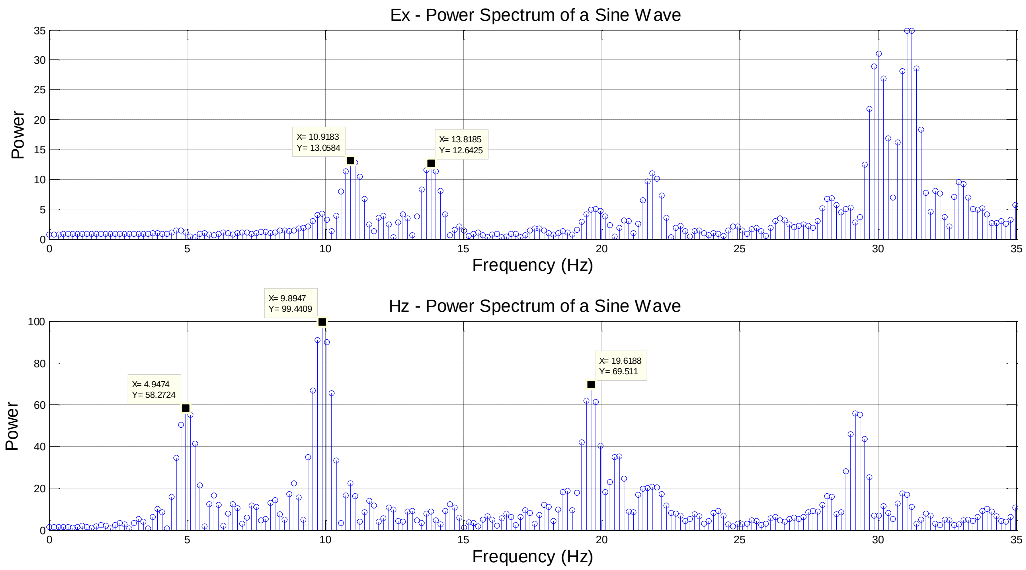
Dispersion Relation: Beta vs. Freq
beta
f (Hz)
X: 10.02
Y: 54.56
X: 15.03
Y: 89.06
X: 19.97
Y: 121.5
X: 24.98
Y: 153.8
X: 30.03
Y: 186
X: 35
Y: 217.7

Beta Test #0: beta = 0 -> Expected Lowest Resonant Mode: f = 5 Hz

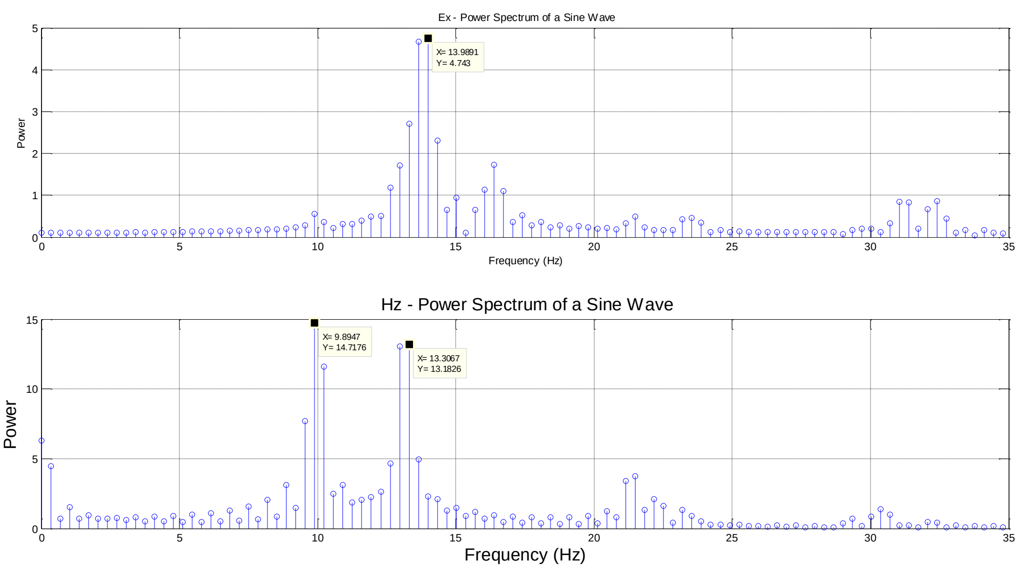
Initial H_y-Field Distribution



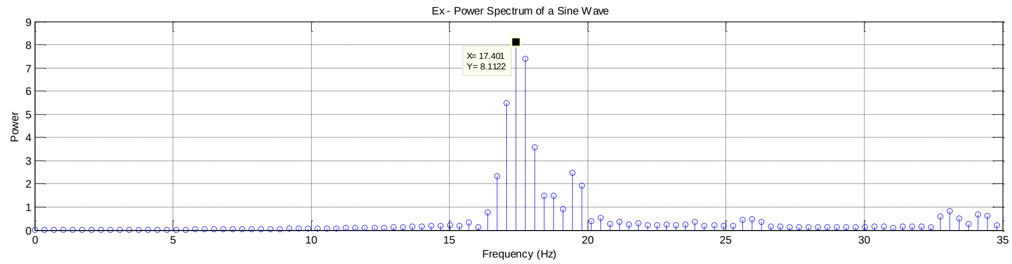


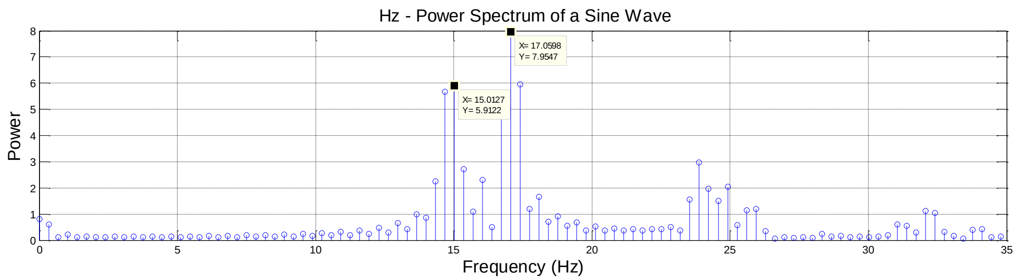


Beta Test #1: beta = 55 -> Expected Lowest Resonant Mode: f = 10 Hz

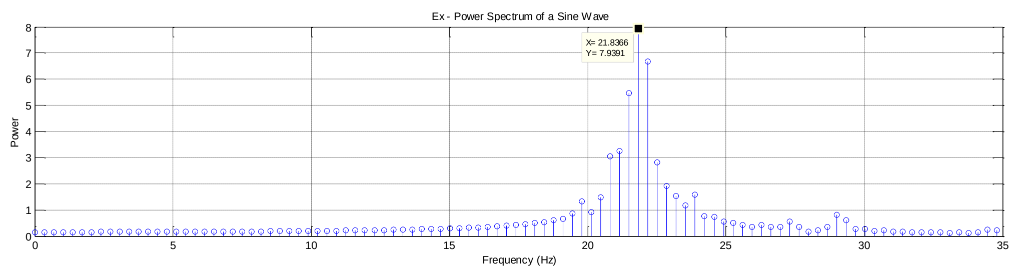


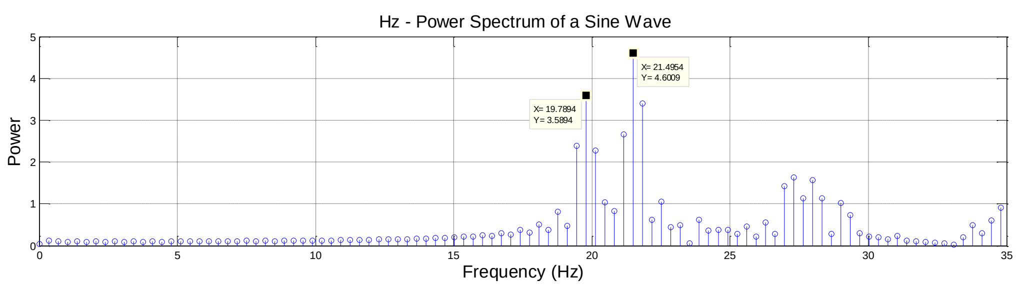
Beta Test #2: beta = 88.9 -> Expected Lowest Resonant Mode: f = 15 Hz



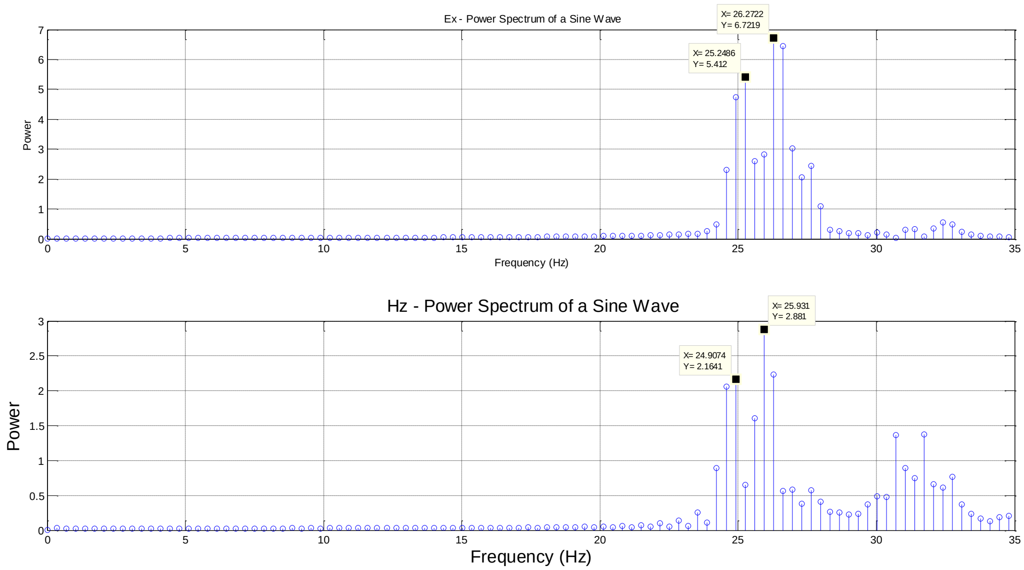


Beta Test #3: beta = 122 -> Expected Lowest Resonant Mode: f = 20 Hz

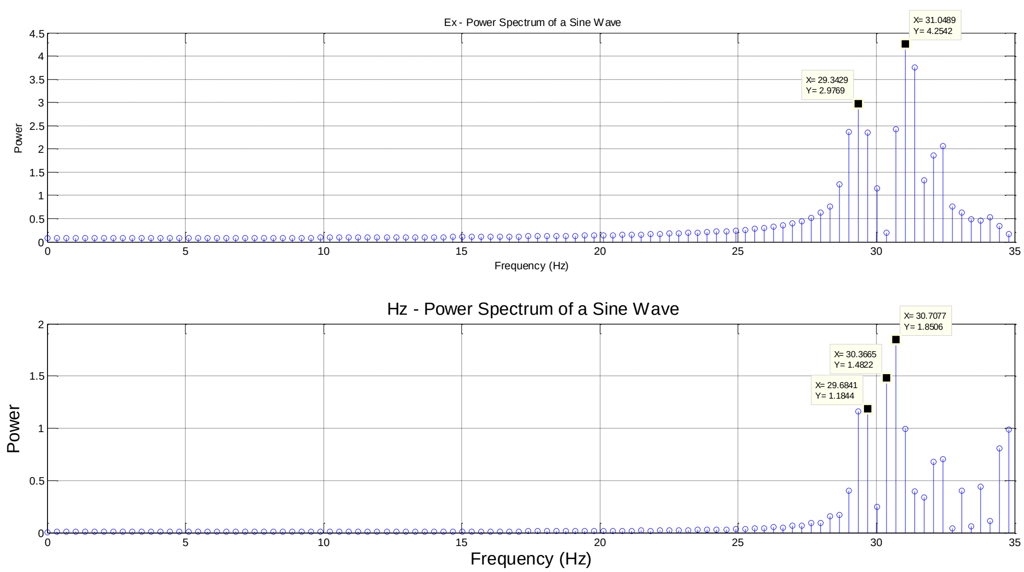




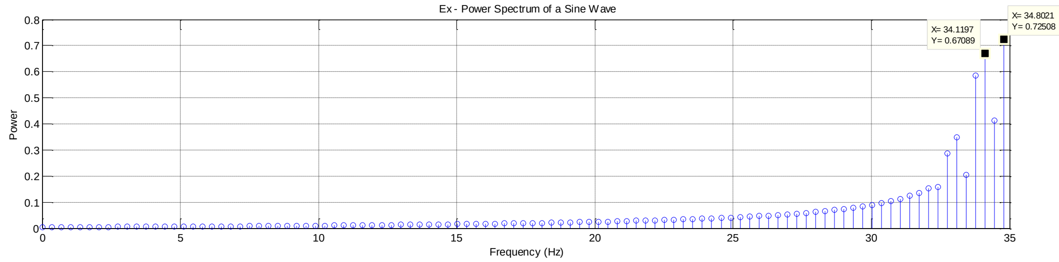
Beta Test #4: beta = 154 -> Expected Lowest Resonant Mode: f = 25 Hz

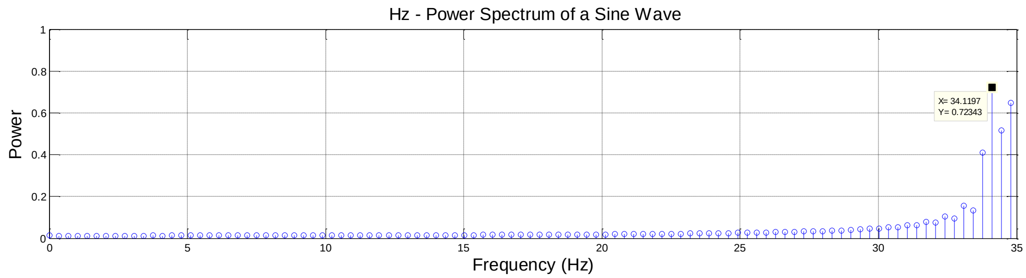


Beta Test #5: beta = 186 -> Expected Lowest Resonant Mode: f = 30 Hz



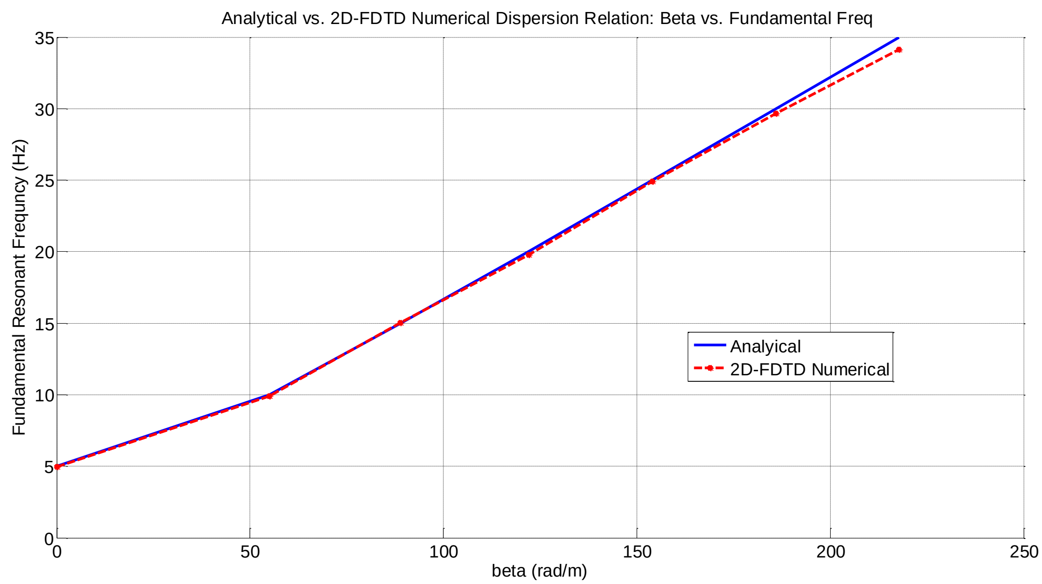
Beta Test #6: beta = 217.7 -> Expected Lowest Resonant Mode: f = 35 Hz





Summary Table:

β (rad/m)	F (analytical) [Hz]	F (numerical 2D-FDTD) [Hz]	Percent Error [%]
0.0	5.0	4.95	-1.0
55.0	10.0	9.89	-1.1
88.9	15.0	15.01	< 0.1
122.0	20.0	19.79	-1.1
154.0	25.0	24.91	-0.4
186.0	30.0	29.68	-1.1
217.7	35.0	34.11	-2.5



Analysis: Very close agreement between 2D-FDTD numerical simulation results and analytical calculation for fundamental resonant mode of the 2-D rectangular waveguide under consideration.

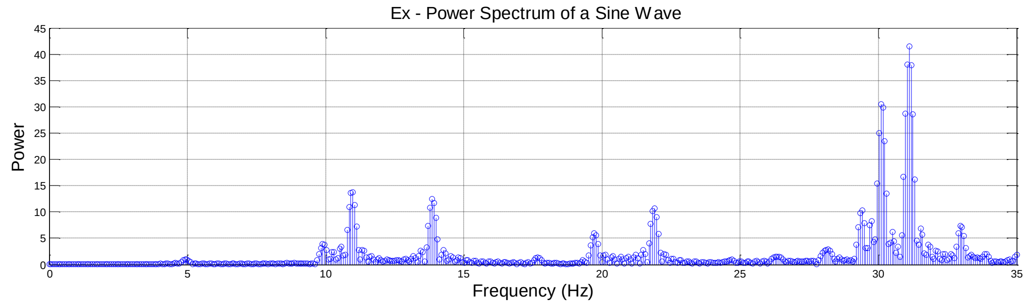
<u>2-D Bonus</u>:

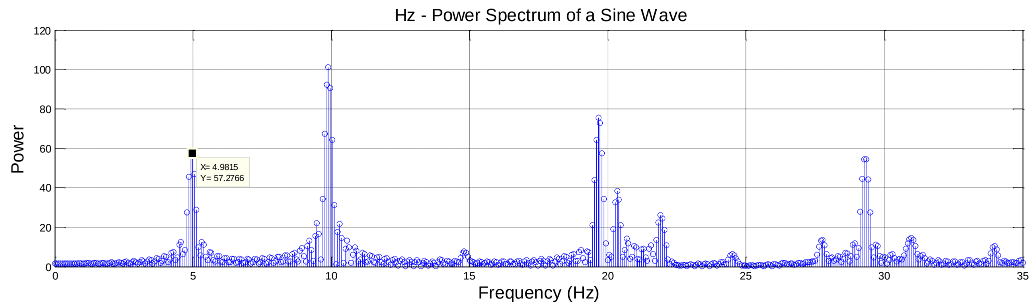
Vary Time Step:

Δt (s)	Δt Stability Criterion	**Δt to Stability Criterion Ratio**	Δx (m)	Δy (m)	M_t	Analytical Lowest Resonant Freq [Hz]	**Lowest Resonant Freq [Hz]**	**Percent Error (%)**
3.5777e-04	8.9443e-04	**0.4**	0.002	0.001	4000	5.0	**4.94**	**-1.2**
4.4721e-04	8.9443e-04	**0.5**	0.002	0.001	4000	5.0	**4.91**	**-1.8**
8.9443e-05	8.9443e-04	**0.1**	0.002	0.001	4000	5.0	**5.45**	**9.0**
6.2610e-04	8.9443e-04	**0.7**	0.002	0.001	4000	5.0	**4.97**	**-0.6**
8.9443e-04	8.9443e-04	**1.0**	0.002	0.001	4000	5.0	**4.98**	**-0.4**
0.0018	8.9443e-04	**2.0**	0.002	0.001	4000	5.0	**Blows up**	**N/A**
8.9443e-06	8.9443e-04	**0.01**	0.002	0.001	4000	5.0	**0.0**	**100%**

Analysis: There is a sweet spot for the Δt time step value for the 2-D FDTD simulations between roughly 0.4 to 1.0 ratio of the calculated stability criteria. I expected the 0.4 and 0.5 values to be more accurate than the 1.0 ratio value so this result was a bit surprising. It shows that the $\Delta t, \Delta x, and\ \Delta y$ should be within similar orders of magnitude and governed by the stability criteria for accurate results.

Δt = 8.9443e-04s = 1.0* Stability Criterion





Vary Δx Step:

*Note: Holding Δt to Stability Criterion Ratio Constant

Δt (s)	Δt Stability Criterion	Δt to Stability Criterion Ratio	Δx (m)	Δy (m)	M_t	Analytical Lowest Resonant Freq [Hz]	Lowest Resonant Freq [Hz]	Percent Error (%)
3.5777e-04	8.9443e-04	**0.4**	**0.002**	0.001	4000	5.0	**4.95**	**-1.0**
3.9223e-04	8.9443e-04	**0.4**	**0.005**	0.001	4000	5.0	**4.98**	**-0.4**
3.9801e-04	8.9443e-04	**0.4**	**0.010**	0.001	4000	5.0	**4.91**	**-1.8**
3.9950e-04	8.9443e-04	**0.4**	**0.020**	0.001	4000	5.0	**9.31**	**86.2**
1.7889e-04	8.9443e-04	**0.4**	**5.0e-4**	0.001	4000	5.0	**5.11**	**2.2**
2.8284e-04	8.9443e-04	**0.4**	**0.001**	0.001	4000	5.0	**4.96**	**-0.8**

Analysis: There is also a sweet spot for the choice of Δx in terms of simulation accuracy and simulation run-time. For this experiment, I also varied the time to keep the delta t to stability criterion constant. I could have also kept delta t constant, however, that would also show that I could make the code blow up beyond a given threshold. For my constant ratio analysis, too large of delta Δx spatial steps eventually degrades simulation performance as shown with the Δx = 0.020 case with 86.2% error. However, more important Δx also impact computation time linear, which is significant. Decreasing the time step by a factor of two in each dimension, doubles the number computations required in that dimension. If the spatial resolution is double in by the Δx and Δy cases, that causes that number of required computations to square, significantly increasing the simulation run-time and computational cost. Also, since we are holding the stability criterion constant, a smaller dx spatial step results in a smaller Δx time step, meaning that M_t (# of time steps) must increase to simulation the same amount of absolute time. In terms of calculating the lower order resonant waveguide modes, the table above shows that the code still performs well at Δx = 0.010 and Δx = 0.005, while requiring significantly less computations than the smaller dx time step experiments without a significant degradation in performance. *Note: This analysis should also apply for varying Δy step since they are both spatial dimensions of a rectangle being evaluated using identical techniques. In absolute terms, Δy requires finer spatial steps since the 2-D waveguide is twice as long in the x-dimension as the y-dimension.

Vary M_t (# of time steps):

*Note: Holding Δt to Stability Criterion Ratio Constant

Δt (s)	Δt Stability Criterion	Δt to Stability Criterion Ratio	Δx (m)	Δy (m)	$\boldsymbol{M_t}$	Analytical Lowest Resonant Freq [Hz]	**Lowest Resonant Freq [Hz]**	**Percent Error (%)**
3.5777e-04	8.9443e-04	0.4	0.002	0.001	**4000**	5.0	**4.95**	**-1.0**
3.5777e-04	8.9443e-04	0.4	0.002	0.001	**1000**	5.0	**5.46**	**9.2**
3.5777e-04	8.9443e-04	0.4	0.002	0.001	**400**	5.0	**6.82**	**36.4**
3.5777e-04	8.9443e-04	0.4	0.002	0.001	**2000**	5.0	**5.11**	**2.2**
3.5777e-04	8.9443e-04	0.4	0.002	0.001	**5000**	5.0	**4.95**	**-1.0**
3.5777e-04	8.9443e-04	0.4	0.002	0.001	**10000**	5.0	**4.95**	**-1.0**

Analysis: Increasing the number of M_t (time steps) increases the amount of time simulated, however, it also increase the number of computations required in simulation and the run-time linearly (O[n]). For the purposes of calculating the lowest resonant modes of the 2-D waveguide, the data in the table above shows that too few time steps does not give the FFT enough data points to converge on the correct resonant frequency accurately, however, after a sufficient number of time steps, the accuracy of the simulation converges. Beyond this point, extra time steps buy a negligible amount of improved accuracy while increasing the simulation time linearly. For the purpose of the 2-D FDTD simulations in this project, M_t = 4000 time steps seemed to give the best trade-off between simulation time and accuracy.

End Matter: In addition to the project assignment, I was able to generate a number of cool field animations by experimenting with different type of sources. The code can handle customized spatial and time varying field excitations.

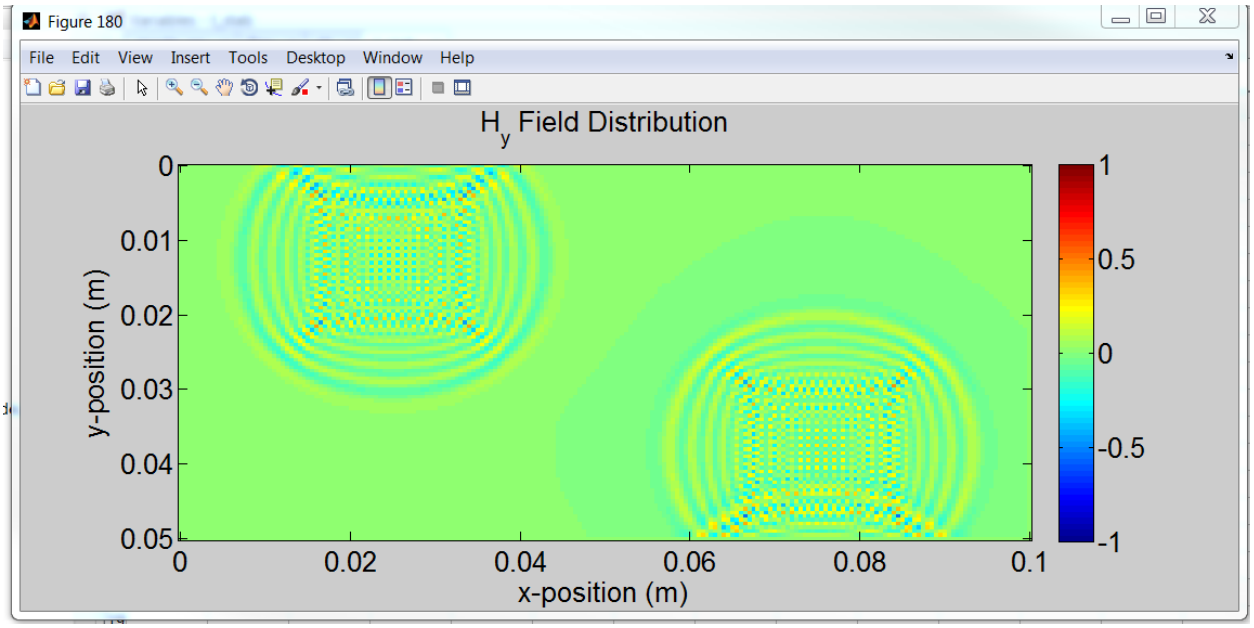


Figure 180
File Edit View Insert Tools Desktop Window Help
H_y Field Distribution
y-position (m)
x-position (m)
0
0.01
0.02
0.03
0.04
0.05
0.02
0.04
0.06
0.08
0.1
1
0.5
0
-0.5
-1

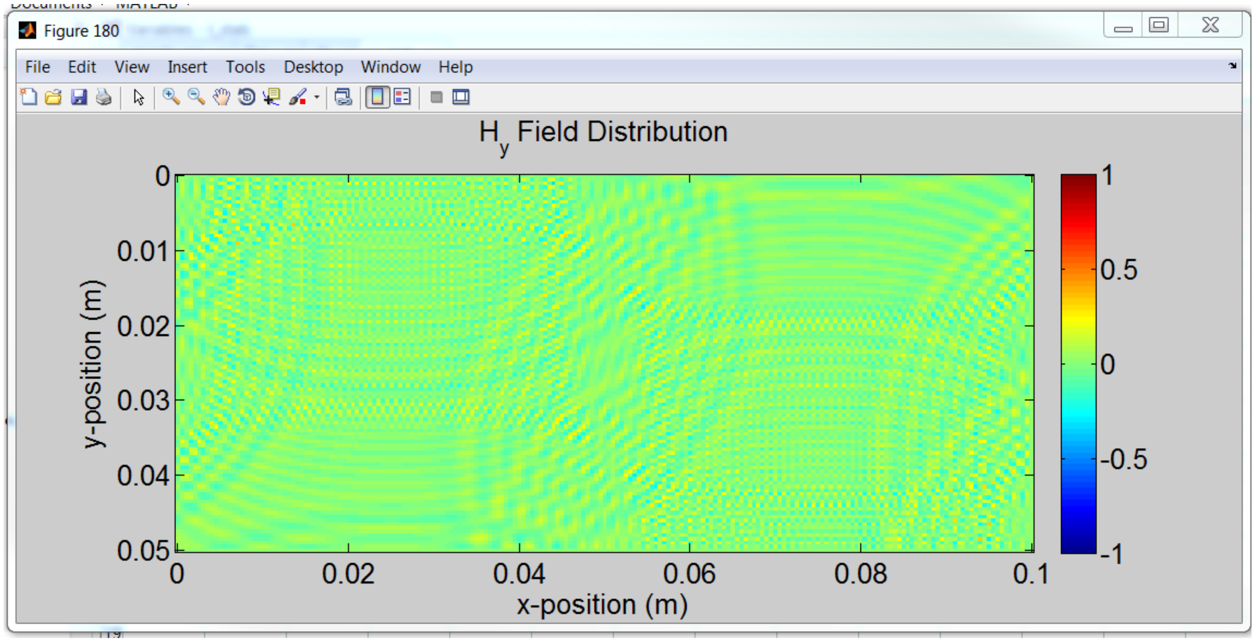
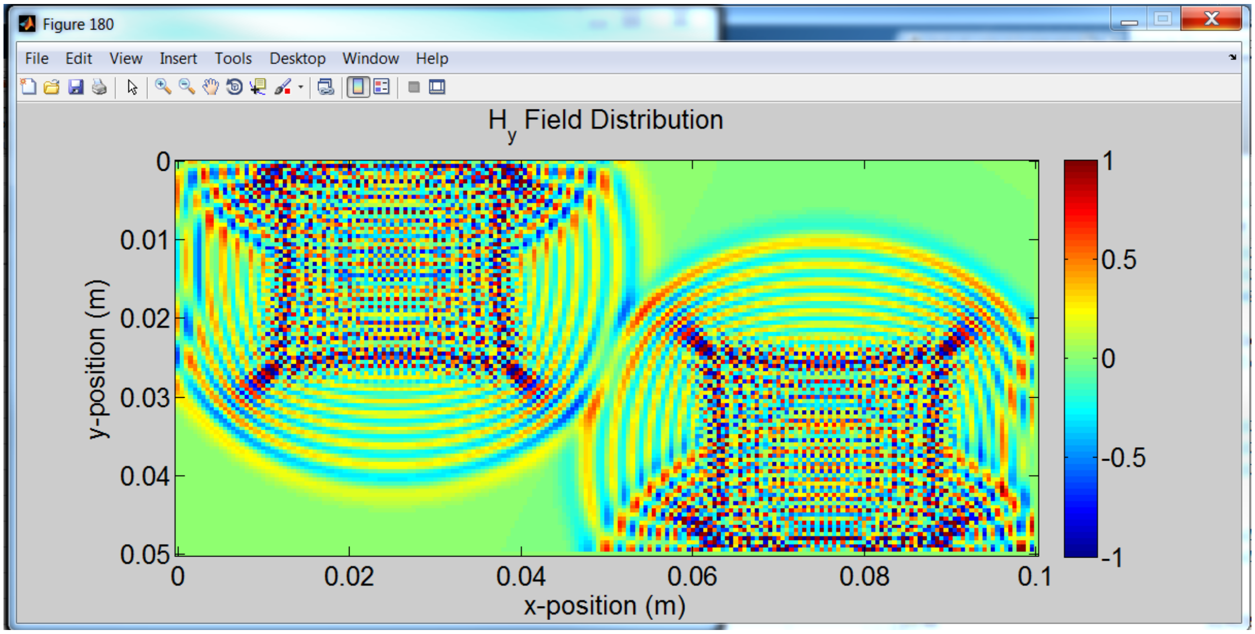


Figure 180
File Edit View Insert Tools Desktop Window Help
H_y Field Distribution
y-position (m)
x-position (m)
0
0.01
0.02
0.03
0.04
0.05
0.02
0.04
0.06
0.08
0.1
1
0.5
0
-0.5
-1



Appendix A: 1-D FDTD MATLAB Code

```
% John Hodge
% 05/01/16
% ECE 5106 1-D FDTD Code

clear all; clc; close all;

%% Set Constants
eps = 1;
mu = 1;

c = 1/sqrt(eps*mu);

L = 30;
%dz = 1;
N = 31;
dz = L./(N-1);
dt = 0.5*dz;

Mt  = 1000;

% t_plot =

%% Step 1: Set Initial Values for Ex and Hy
Ex = zeros(N, Mt);
Hy = zeros(N, Mt);
```

```matlab
% Ex(:,2) = sin(linspace(0, pi, N));
% Ex(:,2) = ones(1,N);

sig_terms = 5;
sig = ones(1, N-1);
for cc = 1:sig_terms
sig = sig + sin((cc.*pi)./30.*(1:N-1));
end

figure;
plot(sig)

Ex(1:end,2) = sin((2*pi)./30.*(1:N));
% Ex(1:end-1,2) = sig;
%Hy(1:end-1,2) = sig;
%Ex(:,2) = ones(1,N);
%Hy(:,2) = ones(1,N);

figure;
stem(Ex(:,2));
xlabel('Position on z-axis (norm units)', 'FontSize', 16)
ylabel('E_x Field Amplitude (V/m)', 'FontSize', 16)
title('1-D FDTD Initial Field Distribution', 'FontSize', 16)
grid on;
set(gca,'Fontsize',16);
xlim([0 30])

for m = 2:1:Mt

  % Update BCs
  Ex(1,m) = 0;
  Ex(N,m) = 0;
  Hy(N,m) = 0;

  % Update E
  for n = 1:N-1
    Hy(n,m) = Hy(n,m-1) + dt./mu .* ( -(Ex(n+1,m) - Ex(n,m))./dz );
  end

    % Update BCs
  Ex(1,m) = 0;
  Ex(N,m) = 0;
  Hy(N,m) = 0;

  % Update H
  for n = 2:N-1
   Ex(n,m+1) = Ex(n,m) + dt./eps .* ( -(Hy(n,m) - Hy(n-1,m))./dz );
  end
```

```
end

N_mid = round(N/2);

t_plot = 2*dt:dt:(Mt+1)*dt;
t_plot = t_plot - 2.*dt;

figure(702);
subplot(2,1,1)
imagesc(t_plot, 1:N, Ex);
title('1-D FDTD Sim: E_x(z,t)', 'FontSize', 16)
xlabel('time (s)', 'FontSize', 16)
ylabel('Position on z-axis (norm units)', 'FontSize', 16)
set(gca,'Fontsize',16);
colorbar;
grid on;

subplot(2,1,2)
imagesc(t_plot, 1:N, Hy);
title('1-D FDTD Sim: H_y(z,t)', 'FontSize', 16)
xlabel('time (s)', 'FontSize', 16)
ylabel('Position on z-axis (norm units)', 'FontSize', 16)
set(gca,'Fontsize',16);
colorbar;
grid on;

figure;
plot(t_plot, Ex(N_mid,2:end))

%% Determine Frequency Spectrum - Ex

x = Ex(N_mid,2:end);
P = nextpow2(length(x));

Nq = 2.^(P+1);
Fs = 1./dt;

%% Fourier Transform:
X = fftshift(fft(x,Nq));
%% Frequency specifications:
dF = Fs/Nq;                  % hertz
f = -Fs/2:dF:Fs/2-dF;        % hertz
%% Plot the spectrum:
figure(703);
subplot(2,2,2)
stem(f,abs(X)/Nq);
```

```
xlabel('Frequency (in hertz)','Fontsize',14);
ylabel('|E_x(f)|','Fontsize',14);
title('|E_x(f)| - Magnitude Response','Fontsize',14);
grid on;
set(gca,'Fontsize',14);
xlim([0 max(f)])

%figure;
subplot(2,2,1)
stem(f,abs(X)/Nq);
xlabel('Frequency (in hertz)','Fontsize',14);
ylabel('|E_x(f)|','Fontsize',14);
title('|E_x(f)| - Magnitude Response','Fontsize',14);
grid on;
set(gca,'Fontsize',14);
xlim([0 0.05])

figure(675);
subplot(2,1,1)
stem(f,abs(X)/Nq);
xlabel('Frequency (in hertz)','Fontsize',14);
ylabel('|E_x(f)|','Fontsize',14);
title('|E_x(f)| - Magnitude Response','Fontsize',14);
grid on;
set(gca,'Fontsize',14);
xlim([0 0.05])


%% Determine Frequency Spectrum - Hy

x = Hy(N_mid,2:end);
P = nextpow2(length(x));

Nq = 2.^(P+1);
Fs = 1./dt;

%% Fourier Transform:
X = fftshift(fft(x,Nq));
%% Frequency specifications:
dF = Fs/Nq;                  % hertz
f = -Fs/2:dF:Fs/2-dF;        % hertz
%% Plot the spectrum:
figure(703);
subplot(2,2,4);
stem(f,abs(X)/Nq);
xlabel('Frequency (in hertz)','Fontsize',14);
ylabel('|H_y(f)|','Fontsize',14);
title('|H_y(f)| - Magnitude Response','Fontsize',14);
```

```
grid on;
set(gca,'Fontsize',14);
xlim([0 max(f)])

% figure;
subplot(2,2,3);
stem(f,abs(X)/Nq);
xlabel('Frequency (in hertz)','Fontsize',14);
ylabel('|H_y(f)|','Fontsize',14);
title('|H_y(f)| - Magnitude Response','Fontsize',14);
grid on;
set(gca,'Fontsize',14);
xlim([0 0.05])

figure(675);
subplot(2,1,2);
stem(f,abs(X)/Nq);
xlabel('Frequency (in hertz)','Fontsize',14);
ylabel('|H_y(f)|','Fontsize',14);
title('|H_y(f)| - Magnitude Response','Fontsize',14);
grid on;
set(gca,'Fontsize',14);
xlim([0 0.05])
```

Appendix B: 2-D FDTD MATLAB Code

```
%% John Hodge
%  03/13/16
% ECE 5106 FDTD Project: 1-D

clear all; clc; close all;

%% Initialize Constants

boundary = 'PEC';
eps_0 = 1;
mu_0 = 1;
c = 1/sqrt(eps_0*mu_0);
L = 30;
N = 31;
dz = 1;
h = dz;
Nx = 51; % Number of Steps in X
Ny = 51; % Number of Steps in Y
Lx = 0.1; % Length of Waveguide in X
Ly = 0.05; % Length of Waveguide in Y
```

```
Nx_mid = round(Nx/2);
Ny_mid = round(Ny/2);

dx = Lx ./ (Nx-1);
dy = Ly ./ (Ny-1);

%dt = 0.4;
n = 6;

%% Calculate Stability for FDTD
t_stab = 1 ./ (c .* sqrt(1/(dx).^2 + 1/(dy).^2));

dt = 0.4*t_stab
% dt = 3.5777e-04;

%dt = 0.3;

Mt = 10000;
t_steps = Mt;

t_plot = (0:dt:dt*(Mt-1));


%% Create Gaussian Pulse to Excite Waveguide

% clear all; clc; close all;

fs=100; %sampling frequency
sigma=1.0;
t=-0.5:1/fs:0.5; %time base
t = linspace(-pi, pi, 101);

variance=sigma^2;
x=1/(sqrt(2*pi*variance))*(exp(-t.^2/(2*variance)));
x_gauss_pulse = 10.*x;
subplot(2,1,1)
plot(t,x,'b');
title(['Gaussian Pulse \sigma=', num2str(sigma),'s'], 'FontSize', 16);
xlabel('Time(s)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;
set(gca,'Fontsize',16);

L=length(x);
NFFT = 1024;
X = fftshift(fft(x,NFFT));
Pxx=X.*conj(X)/(NFFT*NFFT); %computing power with proper scaling
```

```
f = fs*(-NFFT/2:NFFT/2-1)/NFFT; %Frequency Vector

gauss_pulse = x;

subplot(2,1,2)
plot(f,abs(X)/(L),'r');
title('Magnitude of FFT', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Magnitude |X(f)|', 'FontSize', 16);
xlim([-10 10])
grid on;
set(gca,'Fontsize',16);

%% Initialize Fields

Ex = zeros(Nx,Ny,t_steps); Ey = zeros(Nx,Ny,t_steps); Ez = zeros(Nx,Ny,t_steps); Hx =
zeros(Nx,Ny,t_steps); Hy = zeros(Nx,Ny,t_steps); Hz = zeros(Nx,Ny,t_steps);

% Hz(Nx_mid,Ny_mid,1) = 5;
%Hz(20,20,1:250) = 1.*ones(1,250);

%% Create Initial Mode Distribution

m = 1;
n = 0;

%Nx = 200;
%Ny = 100;

a = 0.1; % 10cm
b = 0.05; % 5cm

x_plot = linspace(0, a, Nx);
y_plot = linspace(0, b, Ny);

kx = (m*pi)/a;
ky = (n*pi)/b;
beta = 0;

w = 2*pi*4.5;
t = 0;

H0 = 1;

Hz(:,:,2) = zeros(Nx,Ny);

for xx = 2:length(x_plot)-1
  for yy = 2:length(y_plot)-1
```

```
        x = x_plot(xx);
        y = y_plot(yy);

        Hz(xx,yy,2) = H0.*cos(kx.*x).*cos(ky.*y);%.*exp(1i.*w.*t);

    end
end



figure;
imagesc(x_plot, y_plot, real(Hz(:,:,2))')
colorbar;
title('Initial Field Distribution', 'FontSize', 16);
xlabel('x-position (m)', 'FontSize', 16);
ylabel('y-position (m)', 'FontSize', 16);

%%


%% Step 1: Assign Initial Field Distributions for both Ex and Hy

% Define Source
% Source Location
source_x = Nx_mid;
source_y = Ny_mid;

% Main FDTD Loop
for m = 2:1:Mt
    m

%     if ( m < 50)
%        Hz(round(Nx_mid/2), round(Ny_mid/2)+m, 2) = 25;
%     end
%
%     if ( m < 50)
%        Hz(round(3*Nx_mid/2), round(3*Ny_mid/2)-m, 2) = -25;
%     end

    t = m*dt; % Current time

    %% Step 2: Evolve EM Fields Using Finite Difference Equation

    % Update H-field from E-field
    for ii = 1:Nx-1
        for jj = 1:Ny-1
```

```
        % Hx
        Hx(ii,jj,m+1) = Hx(ii,jj,m) - dt./mu_0 .* ( (Ez(ii,jj+1,m) - Ez(ii,jj,m) )./dy + 1i.*beta.* Ey(ii,jj,m) );

        % Hy
        Hy(ii,jj,m+1) = Hy(ii,jj,m) + dt./mu_0 .* ( (Ez(ii+1,jj,m) - Ez(ii,jj,m) )./dx + 1i.*beta.* Ex(ii,jj,m) );

        % Hz
        Hz(ii,jj,m+1) = Hz(ii,jj,m) + dt./mu_0 .* ( ( Ex(ii,jj+1,m) - Ex(ii,jj,m) )./dy - ( Ey(ii+1,jj,m) - Ey(ii,jj,m)
).dx );

      end
    end

    %% Update H-field Boundaries

    if (strcmp(boundary, 'PEC'))
      % Upper X
      Hx(2:end-1,Ny,m) = 0; Hy(2:end-1,Ny,m) = 0; Hz(2:end-1,Ny,m) = 0;
      Ex(2:end-1,Ny,m) = 0; Ey(2:end-1,Ny,m) = 0; Ez(2:end-1,Ny,m) = 0;

      % Lower X
      Hy(2:end-1,1,m) = 0;
      Ex(2:end-1,1,m) = 0; Ez(2:end-1,1,m) = 0;

      % Left Y
      Hx(1,2:end-1,m) = 0;
      Ey(1,2:end-1,m) = 0; Ez(1,2:end-1,m) = 0;

      % Right Y
      Hx(Nx,2:end-1) = 0; Hy(Nx,2:end-1) = 0; Hz(Nx,2:end-1) = 0;
      Ex(Nx,2:end-1,m) = 0; Ey(Nx,2:end-1,m) = 0;  Ez(Nx,2:end-1,m) = 0;

      % (1,1)
      Hx(1,1,m) = 0; Hy(1,1,m) = 0;
      Ex(1,1,m) = 0; Ey(1,1,m) = 0; Ez(1,1,m) = 0;

      % (Nx,1)
      Hx(Nx,1,m) = 0; Hy(Nx,1,m) = 0; Hz(Nx,1,m) = 0;
      Ex(Nx,1,m) = 0; Ey(Nx,1,m) = 0; Ez(Nx,1,m) = 0;

      % (1,Ny)
      Hx(1,Ny,m) = 0; Hy(1,Ny,m) = 0; Hz(1,Ny,m) = 0;
      Ex(1,Ny,m) = 0; Ey(1,Ny,m) = 0; Ez(1,Ny,m) = 0;

      % (Nx,Ny)
      Hx(Nx,Ny,m) = 0; Hy(Nx,Ny,m) = 0; Hz(Nx,Ny,m) = 0;
      Ex(Nx,Ny,m) = 0; Ey(Nx,Ny,m) = 0; Ez(Nx,Ny,m) = 0;
```

```
  end

  %% Update E-field from H-field
  for ii = 2:Nx-1
    for jj = 2:Ny-1

      % Ex
      Ex(ii,jj,m+1) = Ex(ii,jj,m) + dt./eps_0 .* ( ( Hz(ii,jj,m+1) - Hz(ii,jj-1,m+1) )./dy +
1i.*beta.*Hy(ii,jj,m+1) );

      % Ey
      Ey(ii,jj,m+1) = Ey(ii,jj,m) - dt./eps_0 .* ( ( Hz(ii,jj,m+1) - Hz(ii-1,jj,m+1) )./dx + 1i.*beta.*Hx(ii,jj,m+1)
);

      % Ez
      Ez(ii,jj,m+1) = Ez(ii,jj,m) + dt./eps_0 .* ( ( Hy(ii,jj,m+1) - Hy(ii-1,jj,m+1) )./dx - ( Hx(ii,jj,m+1) -
Hx(ii,jj-1,m+1) )./dy );

    end
  end

  %% Update Ex Boundary

  if (strcmp(boundary, 'PEC'))
    % Upper X
    Hx(2:end-1,Ny,m) = 0; Hy(2:end-1,Ny,m) = 0; Hz(2:end-1,Ny,m) = 0;
    Ex(2:end-1,Ny,m) = 0; Ey(2:end-1,Ny,m) = 0; Ez(2:end-1,Ny,m) = 0;

    % Lower X
    Hy(2:end-1,1,m) = 0;
    Ex(2:end-1,1,m) = 0; Ez(2:end-1,1,m) = 0;

    % Left Y
    Hx(1,2:end-1,m) = 0;
    Ey(1,2:end-1,m) = 0; Ez(1,2:end-1,m) = 0;

    % Right Y
    Hx(Nx,2:end-1) = 0; Hy(Nx,2:end-1) = 0; Hz(Nx,2:end-1) = 0;
    Ex(Nx,2:end-1,m) = 0; Ey(Nx,2:end-1,m) = 0;  Ez(Nx,2:end-1,m) = 0;

    % (1,1)
    Hx(1,1,m) = 0; Hy(1,1,m) = 0;
    Ex(1,1,m) = 0; Ey(1,1,m) = 0; Ez(1,1,m) = 0;

    % (Nx,1)
    Hx(Nx,1,m) = 0; Hy(Nx,1,m) = 0; Hz(Nx,1,m) = 0;
    Ex(Nx,1,m) = 0; Ey(Nx,1,m) = 0; Ez(Nx,1,m) = 0;
```

```
        % (1,Ny)
        Hx(1,Ny,m) = 0; Hy(1,Ny,m) = 0; Hz(1,Ny,m) = 0;
        Ex(1,Ny,m) = 0; Ey(1,Ny,m) = 0; Ez(1,Ny,m) = 0;

        % (Nx,Ny)
        Hx(Nx,Ny,m) = 0; Hy(Nx,Ny,m) = 0; Hz(Nx,Ny,m) = 0;
        Ex(Nx,Ny,m) = 0; Ey(Nx,Ny,m) = 0; Ez(Nx,Ny,m) = 0;

    end

end

%%

% Calc Min and Max Fields
Hz_max = max(max(max(Hz)));
Hz_min = min(min(min(Hz)));

figure;
stem(squeeze(Hz(20,20,:)))

figure;
stem(squeeze(Hz(Nx_mid,Ny_mid,:)))

pts_plot = 200;

figure;
subplot(1,2,1)
imagesc(squeeze(Hz(:,Ny_mid,1:pts_plot)))
colorbar;

subplot(1,2,2)
imagesc(squeeze(Hz(Nx_mid,:,1:pts_plot)))
colorbar;



figure;
subplot(2,3,1)
imagesc(squeeze(Hz(:,Ny_mid,1:pts_plot)))
colorbar;

subplot(2,3,4)
imagesc(squeeze(Hz(Nx_mid,:,1:pts_plot)))
colorbar;

subplot(2,3,2)
imagesc(squeeze(Ex(:,Ny_mid,1:pts_plot)))
```

```
colorbar;

subplot(2,3,5)
imagesc(squeeze(Ex(Nx_mid,:,1:pts_plot)))
colorbar;

subplot(2,3,3)
imagesc(squeeze(Ey(:,Ny_mid,1:pts_plot)))
colorbar;

subplot(2,3,6)
imagesc(squeeze(Ey(Nx_mid,:,1:pts_plot)))
colorbar;

figure;
stem(squeeze(Hz(source_x,source_y,:)))
title('Hz at Source Over Time')

Hz_source_FFT = abs(fftshift(fft(squeeze(Hz(source_x,source_y,:)))));
figure;
stem(Hz_source_FFT)
%%

%% Calc and Plot FFT - Hz

Fs = 1/dt;
t_plot = (0:dt:dt*(Mt-1));

Hz_source = squeeze(Hz(source_x,source_y,:));
%Hz_source = Hz_source(571:end);
%t_plot = t_plot(571:end);

% Hz_source = cos(2.*pi.*15.*t_plot);

nfft = 2.^(2+nextpow2(length(Hz_source)));
X = fft(Hz_source,nfft);
X = X(1:nfft/2);
% Take the mag of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;

% Generate the plot, title and labels.
figure;
plot(t_plot,Hz_source(2:end));
title('Sine Wave Signal', 'FontSize', 16);
xlabel('Time (s)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
```

```matlab
figure;
stem(f,mx);
title('Power Spectrum of a Sine Wave', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Power', 'FontSize', 16);

figure(832);
subplot(2,1,2)
stem(f,mx);
title('Hz - Power Spectrum of a Sine Wave', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Power', 'FontSize', 16);
grid on;
xlim([0 35]);

%% Calc and Plot FFT - Ex

Fs = 1/dt;
t_plot = (0:dt:dt*(Mt-1));

Ex_source = squeeze(Ex(source_x,source_y,:));
%Hz_source = Hz_source(571:end);
%t_plot = t_plot(571:end);

% Hz_source = cos(2.*pi.*15.*t_plot);

nfft = 2.^(2+nextpow2(length(Ex_source)));
X = fft(Ex_source,nfft);
X = X(1:nfft/2);
% Take the mag of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;

% Generate the plot, title and labels.
figure;
plot(t_plot,Ex_source(2:end));
title('Sine Wave Signal', 'FontSize', 16);
xlabel('Time (s)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);

figure;
stem(f,mx);
title('Power Spectrum of a Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Power');
```

```
figure(832);
subplot(2,1,1)
stem(f,mx);
title('Ex - Power Spectrum of a Sine Wave', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Power', 'FontSize', 16);
grid on;
xlim([0 35]);


%% Animate Fields

animate_fields = 0;
if (animate_fields == 1)
   for tt = 1:t_steps

       figure(800);
       subplot(1,3,1)
       imagesc(squeeze(Hz(:,:,tt)), [-1 1])
       title('Hz', 'FontSize', 16)
       colorbar;

       subplot(1,3,2)
       imagesc(squeeze(Ex(:,:,tt)), [-1 1])
       title('Ex', 'FontSize', 16)
       colorbar;

       subplot(1,3,3)
       imagesc(squeeze(Ey(:,:,tt)), [-1 1])
       title('Ey', 'FontSize', 16)
       colorbar;

   end
end

%%

% figure(180);
% colorbar;
% title('H_y Field Distribution', 'FontSize', 16);
% xlabel('x-position (m)', 'FontSize', 16);
% ylabel('y-position (m)', 'FontSize', 16);

animate_fields = 1;
if (animate_fields == 1)
   for tt = 1:t_steps

       figure(180);
```

```
    imagesc(x_plot, y_plot, squeeze(Hz(:,:,tt))', [-1 1])
    colorbar;
%       title('H_y Field Distribution', 'FontSize', 16);
%       xlabel('x-position (m)', 'FontSize', 16);
%       ylabel('y-position (m)', 'FontSize', 16);
%       set(gca,'Fontsize',16);

  end
end

% figure;
% subplot(2,1,1)
% imagesc(t_plot, 1:31, Ex)
% colorbar;
% title('Ex Field: Position vs. Time')
% ylabel('z-position')
% xlabel('time')
%
% subplot(2,1,2)
% imagesc(t_plot, 1:31, Hy)
% colorbar;
% ylabel('z-position')
% xlabel('time')
% title('Hy Field: Position vs. Time')
%
%
% figure;
% subplot(2,1,1)
% imagesc(t_plot, 1:31, Ex)
% colorbar;
% title('Ex Field: Position vs. Time')
% ylabel('z-position')
% xlabel('time')
% xlim([0 1000])
%
% subplot(2,1,2)
% imagesc(t_plot, 1:31, Hy)
% colorbar;
% ylabel('z-position')
% xlabel('time')
% title('Hy Field: Position vs. Time')
% xlim([0 1000])

% figure;
% subplot(2,1,1)
% imagesc(abs(Ex))
% colorbar;
% title('Ex Field: Position vs. Time')
```

```
% ylabel('z-position')
% xlabel('time')
%
% subplot(2,1,2)
% imagesc(abs(Hy))
% colorbar;
% ylabel('z-position')
% xlabel('time')
% title('Hy Field: Position vs. Time')
%
% figure;
% imagesc(abs(Ex).^2 + abs(Hy).^2)
% colorbar;
% title('Ex Field: Position vs. Time')
% ylabel('z-position')
% xlabel('time')


%% Animate Fields Over Time

animate_fields = 0;

if (animate_fields == 1)
  for tt = 1:t_steps,
    figure(101);
    subplot(2,1,1)
    stem(Ex(:,tt));
    axis([1 N -1 1])
    xlabel('z-position')
    ylabel('Ex Field Amplitude')
    grid on;
    drawnow

    subplot(2,1,2)
    stem(Hy(:,tt));
    axis([1 N -1 1])
    xlabel('z-position')
    ylabel('Hy Field Amplitude')
    grid on;
    drawnow
  end
end


% %% Step 3: Perform Fourier Transform of the EM Fields and obtain the EM field as function of
frequency
%
% % Ex FFT
```

```
% Ex_FFT = fft(Ex(16,:));
% Ex_FFT = Ex_FFT./max(Ex_FFT);
%
% x = Ex(16,:);
% fs = 1./dt;
% N = length(x);
% ws = 2*pi/N;
% wnorm = -pi:ws:pi;
% wnorm = wnorm(1:length(x));
% w = wnorm*fs;
% figure;
% plot(w,abs(fftshift(Ex_FFT)))
% % axis([-30,30,0,160])
%
% % Hy FFT
% Hy_FFT = fft(Hy(16,:));
% Hy_FFT = Hy_FFT./max(Hy_FFT);
%
% figure;
% subplot(2,1,1)
% stem(abs(fftshift(Ex_FFT)))
% title('Ex: FT')
%
% subplot(2,1,2)
% stem(abs(fftshift(Hy_FFT)))
% title('Hy: FT')
%
%
% %% Processing JH 04/14/16
%
% Ex_mid = Ex(6,:);
%
% Ex_mid_FFT = fft(Ex(16,:));
% Ex_mid_FFT_norm = Ex_FFT./max(Ex_FFT);
%
% Ex_mid_FFT = fftshift(Ex_mid_FFT);
% Ex_mid_FFT_norm = fftshift(Ex_mid_FFT_norm);
%
% figure;
% plot(abs(Ex_mid_FFT_norm))
% title('Ex_mid_FFT_norm')
%
% %%
% % find out signal's Fourier transform and plot its frequency spectrum
% omega_dn = -10;
% omega_up = 10;
% N_omega = t_steps;
%
```

```
% omega_array = linspace(omega_dn,omega_up,N_omega);
% E_omega_array = zeros(1,N_omega);
%
% % Massage Variable Names
% t_array = t_plot;
% E_array = Ex_mid;
% delta_t = dt;
%
% delta_omega = omega_array(2) - omega_array(1);
% for ii=1:N_omega
%     omega = omega_array(ii);
%     exp_function_temp = exp(-1j*omega*t_array);
%     temp_vec = E_array .* exp_function_temp;
%     E_omega_array(ii) = sum(temp_vec)*delta_t;
% end
%
%
% figure;
% set(gca,'Fontsize',20);
% semilogy(omega_array/2/pi,abs(E_omega_array).^2,'.'); grid on;
% xlabel('Frequency (Hz)');
% ylabel('|E_x(\nu)|^2');
%
% figure;
% set(gca,'Fontsize',20);
% stem(omega_array/2/pi,abs(E_omega_array).^2,'.'); grid on;
% xlabel('Frequency (Hz)');
% ylabel('|E_x(\nu)|^2');
```