Christopher Stricklan
03/09/2011
EEL 5390 – Special Topics (FDTD)
HW #6

Notes:

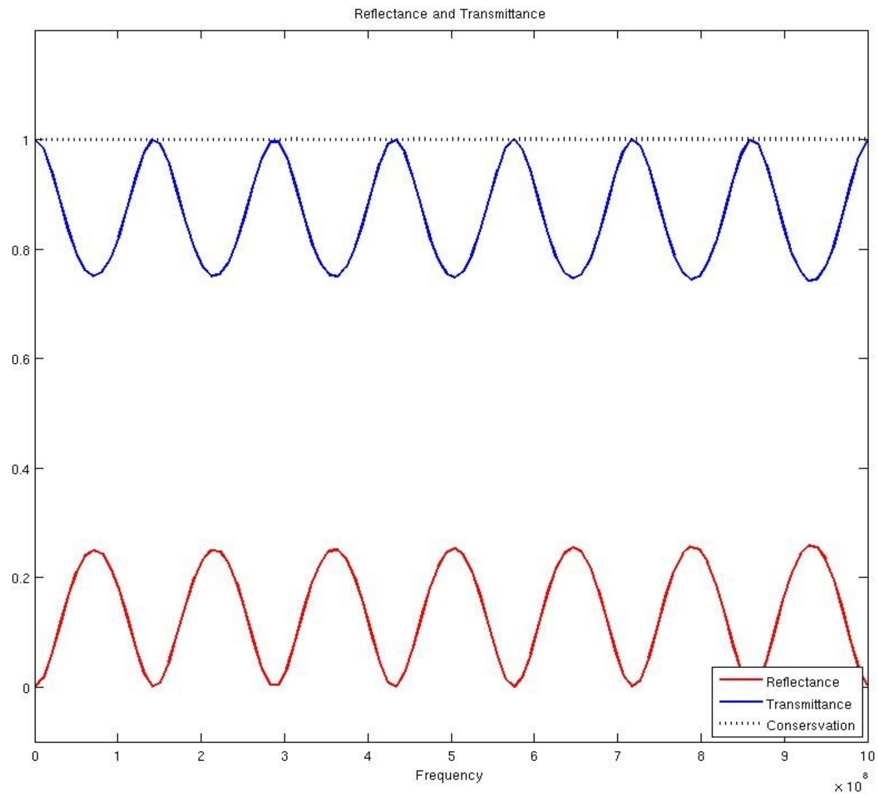
I have broken up the code in multiple methods. The goal here would be to have one FDTD1D method to handle all cases. The issue though is the conversion between freq and wavelength. I have a elegant way to handle that in mind, but didn't have a chance to finish it. So my Dielectric Slab and Invisible Slab both used the FDTD1D.m method and the Blind Missile was it's own file.

P1 – Draw1D

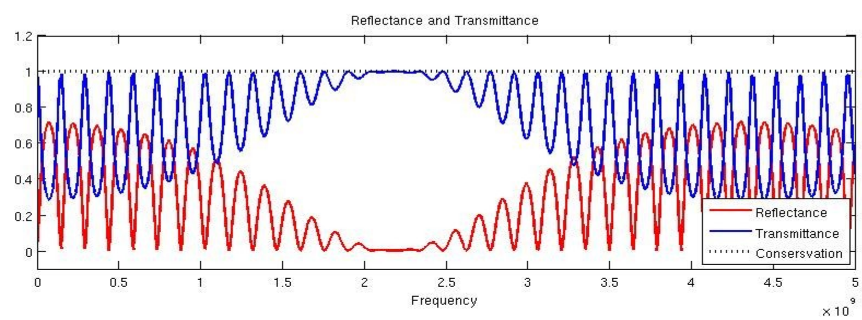
```
function Draw1D( ER, E, H, dz )

   %Initialize
   za=[0:length(E)-1]*dz;

   % Just inverse our Permitivity to get grayscale value
   Color = 1./ER;

   % Need to do an initial draw so we can start the hold for plotting.
   fill(0,0,'-w');   hold on;
   i = 1;
   count = 0;
   prev = 0;
   while i < length(ER)
    i = i + 1;

    if(prev == 1)
      prev = ER(i);
      continue;
    end

    if(prev == ER(i))
      count = count + 1;

    else
      xstart = (i-count)*dz;
      xend = xstart + count*dz;

      x = [ xstart xend xend xstart xstart ];
      y = [ -1.5 -1.5 1.5 1.5 -1.5 ];
      fill(x,y,[Color(i-1) Color(i-1) Color(i-1)]);

      count = 1;
      prev = ER(i);
    end
   end

   %Plot Fields
   plot(za, E, '-b');
   plot(za, H, '-r');

   hold off;
end
```

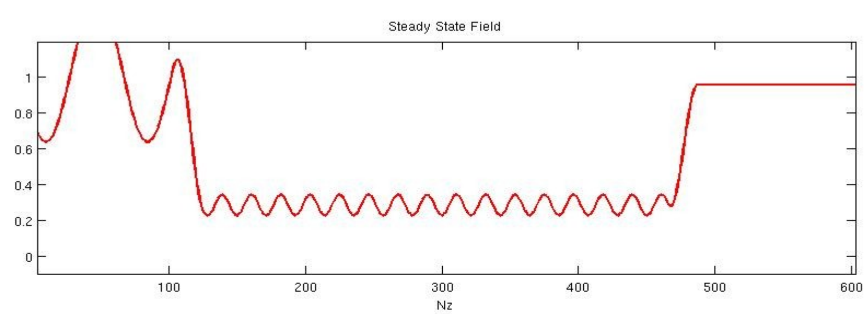
P2 – Perform Simulations of Examples

Dielectric Slab

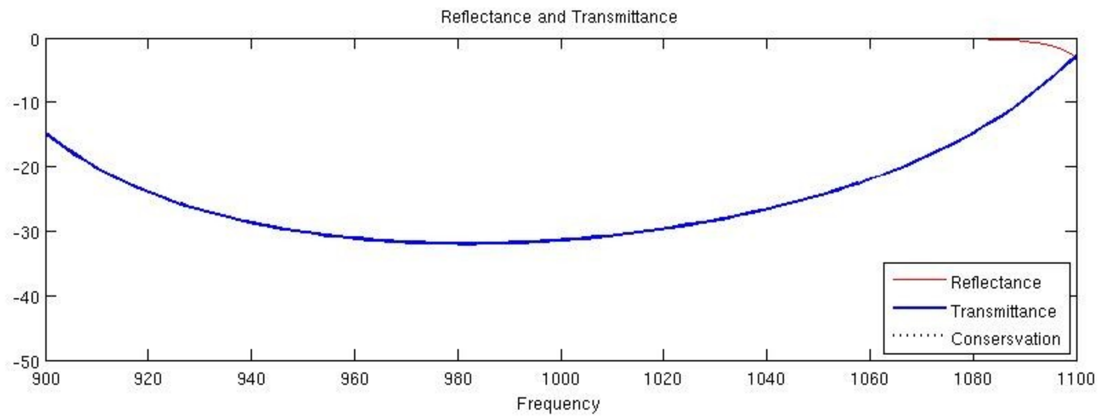


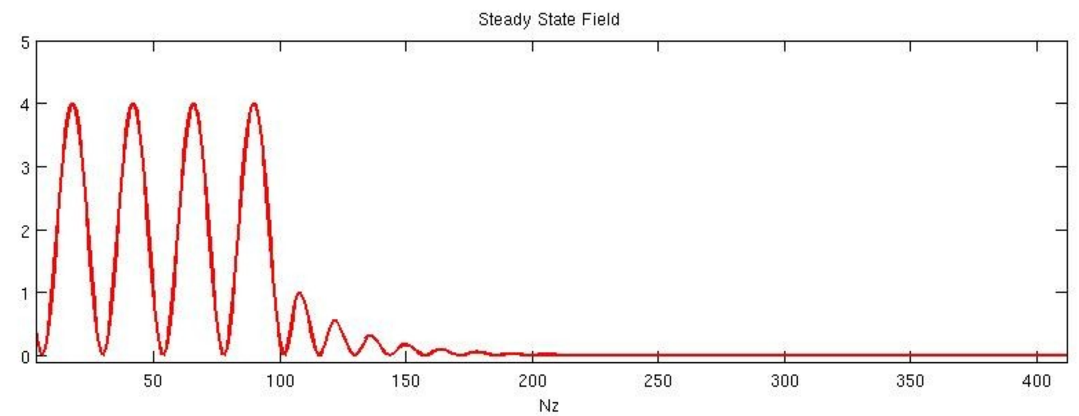
Invisible Slab





Blinded Missle





Appendix

GetNlambda.m

```
function [ N_lambda ] = GetNlambda( ER, UR )
%GETNLAMBDA Summary of this function goes here
%   Detailed explanation goes here

nmax = Getnmax(ER, UR);

if(nmax < 10)
  N_lambda = 20;
end

if((nmax > 10) && (nmax<=40))
  N_lambda = 30;
end

if((nmax > 40) && (nmax <= 60))
  N_lambda = 60;
end


if(nmax > 60)
  N_lambda = 200;
end


end
```

Getnmax.m

```
function [ nmax ] = Getnmax( ER, UR )
%GETNMAX Method calculates the nmax for a materials array for 1D
%   Detailed explanation goes here

  if(length(ER) ~= length(UR))
    MException('ArraySize', 'Materials arrays are not the same size');
  end

  n = zeros([1 size(ER)]);
  n = sqrt(ER.*UR);  % Calculate refractive index of each material
  nmax = max(n);
end
```

DielectricSlab.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Dielectric Slab Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Initialize MATLAB
close all; clc;
clear all;

% Dimensions
% Slab is 12 inches thick surrounded by air on each side
d = 12 * 2.54; %cm thick
dc = d/100; %meters Our critical dimension in this case is the whole slab
rNz = ceil(d); %This Nz represents real world size
rNz = rNz + 2;  % We are going to add air on each side of the problem.

%Material Vectors Initialized at Air
rER = ones([1 rNz]);
rUR = ones([1 rNz]);

% Add our Slab materials to the model
rER(1:rNz) = 6;
rUR(1:rNz) = 2;

% Frequency

freq_start = 0; %DC
freq_end = 1e9; %1Ghz

NFREQ = freq_end / 10e6; %Frequencies every 100Mhz upto 10Ghz
FREQ = linspace(freq_start, freq_end, NFREQ); %FREQ List

FDTD1D( dc, dc, rER, rUR, -1, -1, FREQ, NFREQ, 1000, -1, 'HW#6-P2-Dielectric Slab'
);
```

InvisibleSlab.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Invisible Slab Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize MATLAB
close all; clc;
clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Problem
%    A radome is being deigned to protect an antenna.
%    Antenna operates at 2.4Ghz
%    radome is 1ft thick with a dielectric constant = 12
%    We want  to maximize transmission through dome
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Constants
c0 = 299792458;

% Frequency we want to transmit
f_trans = 2.4e9; %2.4Ghz
lambda_trans = c0/f_trans;


% Dimensions
% Radome is 12 inches thick
d_radome = 12 * 2.54/100; %cm thick

% We need to place a Anti-Reflective Layer on each side of the Radome
e_radome = 12;
e_air = 1;
e_nonreflective = sqrt(e_radome*e_air);

n_nonreflective = sqrt(e_nonreflective);
d_nonreflective = lambda_trans/(4*n_nonreflective);

dc = d_nonreflective; %meters Our critical dimension in this case the anti-
reflectivelayer
rNz = ceil((round(d_radome*100) + 2*round(d_nonreflective*100)))+2; %This Nz
represents real world size

% Material Vectors Initialized at Air
rER = ones([1 rNz]);
rUR = ones([1 rNz]);

% Add our Materials to the model
zstart = 1;
zend = ceil(d_nonreflective*100);
rER(zstart: zend) = e_nonreflective;

zstart = zend + 1;
zend = zstart + floor(d_radome*100);
rER(zstart:zend) = e_radome;

zstart=zend+1;
zend = zstart + floor(d_nonreflective*100);
```

```
rER(zstart:zend) = e_nonreflective;


% Frequency

freq_start = 0; %DC
freq_end = 5e9;%f_trans*2; %1Ghz

NFREQ = freq_end / 10e6; %Frequencies every 100Mhz upto 5Gz
FREQ = linspace(freq_start, freq_end, NFREQ); %FREQ List

FDTD1D( dc, (d_radome+2*d_nonreflective), rER, rUR, 35000, 100, FREQ, NFREQ, 1000,
2.4e9, 'HW#6-P2-Invisible Slab');
```

FDTD1D.m

```
function FDTD1D( dc, Length, rER, rUR, Steps, Buffer, FREQ, NFREQ, Update, SSFREQ,
Title )
%FDTD1D Method executes a FDTD1D Model
%   Detailed explanation goes here

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Pre-Program Work
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Constants
c0 = 299792458; %m/s
e0 = 8.854187817*10^-12; %F/m
u0 = 1.256637061*10^-6; %H/m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Initialization of Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f_max = FREQ(length(FREQ));
nmax = Getnmax(rER, rUR);

%Compute Grid Resolution
% Wave Length Rsolution
N_lambda = GetNlambda(rER, rUR);
lambda_min = c0 / (f_max);
d_lambda = lambda_min/N_lambda/nmax;

% Structure Resolution
N_d = 4;
d_d = dc/4;

% Calculate grid resolution dz
dz = min(d_lambda, d_d);
N_prime = ceil(dc/dz);
dz = dc/N_prime;

% Calculate Grid Size
Nz = ceil(Length/dz);

% Add free space buffer and TF/SF
if(Buffer == -1)
  buffer = ceil(d_lambda/dz) * 5;
  buffert = buffer*2 + 3;
else
  buffer = Buffer;
  buffert = buffer*2;
end

Nz = Nz + buffert;


%Compute Time Steps
dt = dz/(2*c0); %secs

% Source Parameters
```

```
nzc = 2;  %Position of Sources at our TF/SF boundary
tau = 0.5/f_max;        % tau parameter
t0 = 6*tau;              % Delay/Pulse Position

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cf = floor((Nz - buffert)/length(rER)); % Conversion factor to convert our real
grid to our numerical grid

%Material Vectors
ER = zeros([1 Nz]);
UR = zeros([1 Nz]);

% We Need to lay our real materials vectors over our numerical material
% grid

% Lets place our real grid in proper location on numerical grid
for i = 0 : length(rER)-1
  index = buffer+2 + i*cf+1;
%  disp(['i: ' num2str(i) ' i2: ' num2str(index)]);
  ER(index) = rER(i+1);
  UR(index) = rUR(i+1);
end



% Need to backfill in our values
ER(1:buffer+2) = 1;
ER(length(ER)-buffer-1:length(UR)) = 1;
UR(1:buffer+2) = 1;
UR(length(UR)-buffer-1:length(UR)) = 1;

for i=buffer+2 : length(ER-buffer-1)
  if(ER(i) == 0)
    ER(i) = ER(i-1);
  end

  if(UR(i) == 0)
    UR(i) = UR(i-1);
  end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Calculate STEPS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

STEPS = Steps;
if(STEPS == -1)
  tprop = (nmax*Nz*dz)/c0; % Wave Propagation time;
  T = 12*tau + 5*tprop;
  STEPS = ceil(T/dt);
end

ta = [0:STEPS-1]*dt;     % Time Axis;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Source
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s = dz/(2*c0) + dt/2;     % Delay between E and H
Esrc = exp(-((ta-t0)/tau).^2); % E Source
A = -sqrt(ER(nzc)/UR(nzc));     % H Amplitude
Hsrc = A*exp(-((ta-t0+s)/tau).^2); % H Source


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FDTD Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Grid Axis
za=[0:Nz-1]*dz;


% Compute Update Coefficients
mER = (c0*dt/dz)./ER;
mHR = (c0*dt/dz)./UR;

% Initialize Feilds
Ey = zeros([1 Nz]);
Hx = zeros([1 Nz]);


%PAB Parameters
h1 = 0; h2 = 0; h3 = 0;
e1 = 0; e2 = 0; e3 = 0;

%Power Measurements
REF = zeros(1, NFREQ);
TRN = zeros(1, NFREQ);
SRC = zeros(1, NFREQ);
K = exp(-1i*2*pi*dt*FREQ);

SSFK = exp(-1i*2*pi*dt*SSFREQ);
SSFPOWER = zeros(1, Nz);
SSFSRC = zeros(1, Nz);

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('% Parameters');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');

disp(['f_max' num2str(f_max)]);
disp(['lamda_min: ' num2str(lambda_min)]);
disp(['d_lambda: ' num2str(d_lambda)]);
disp(['nmax: ' num2str(nmax)]);
disp(['dc: ' num2str(dc)]);
disp(['d_d: ' num2str(d_d)]);
disp(['Nz: ' num2str(Nz)]);
disp(['buffer: ' num2str(buffer)]);
disp(['dz: ' num2str(dz)]);
disp(['Length: ' num2str(Nz*dz)]);
disp(['dt: ' num2str(dt)]);
disp(['tau: ' num2str(tau)]);
disp(['t0: ' num2str(t0)]);
disp(['STEPS: ' num2str(STEPS)]);
```

```
disp(['s: ' num2str(s)]);
disp(['A: ' num2str(A)]);
% disp(['ER: ' num2str(length(ER))]);
% disp(ER);
% disp(['UR: ' num2str(length(UR))]);;
% disp(UR);
% return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Execute Simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for t = 1:STEPS

  % Calculate H
  for nz = 1:Nz-1
    Hx(nz) = Hx(nz) + mHR(nz)*(Ey(nz+1)-Ey(nz));
  end

  Hx(Nz) = Hx(Nz) + mHR(Nz)*(e3 - Ey(Nz));

  %H Sources
  Hx(nzc-1) = Hx(nzc-1) - mHR(nzc-1)*Esrc(t);

  h3 = h2; h2 = h1; h1 = Hx(1); % Boundary Params;

  % Calculate E
  Ey(1) = Ey(1) + mER(1)*(Hx(1) - h3);
  for nz = 2:Nz
    Ey(nz) = Ey(nz) + mER(nz)*(Hx(nz)-Hx(nz-1));
  end

  %Inject Source
  Ey(nzc) = Ey(nzc) - mER(nzc)*Hsrc(t);

  e3=e2; e2=e1; e1=Ey(Nz); % Boundary Params;

 %Update Fourier Transforms
 for nf = 1: NFREQ
   REF(nf) = REF(nf) + (K(nf)^t)*Ey(1)*dt;
   TRN(nf) = TRN(nf) + (K(nf)^t)*Ey(Nz)*dt;
   SRC(nf) = SRC(nf) + (K(nf)^t)*Esrc(t)*dt;
 end


 if(SSFREQ ~= -1)
   for n = 3 : Nz-1
     SSFPOWER(n) = SSFPOWER(n) + (SSFK^t)*Ey(n)*dt;
     SSFSRC(n) = SSFSRC(n) + (SSFK^t)*Esrc(t)*dt;
   end
 end

 if(mod(t,Update) == 0 || t == 1)
   h = subplot(11,1,1:4);
   Draw1D(ER, Ey, Hx, dz);
   axis([za(1) za(Nz) -1.5 1.5]);
   xlabel('z');
   title(['Field at Step ' num2str(t) ' of ' num2str(STEPS)]);
```

```
   R = abs(REF./SRC).^2;
   T = abs(TRN./SRC).^2;

   subplot(11,1,8:11)
   plot(FREQ, R, '-r'); hold on;
   plot(FREQ, T, '-b');
   plot(FREQ, R+T, ':k', 'LineWidth', 2); hold off;
   axis([FREQ(1) FREQ(NFREQ) -0.1 1.5]);
   xlabel('Frequency');
   title('Reflectance and Transmittance');
   drawnow();
 end



  %if(mod(t,50) == 0)
  %  saveas(h, ['images/' num2str(t) '.jpg'], 'jpg');
  %end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute Values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

REF = abs(REF./SRC).^2;
TRN = abs(TRN./SRC).^2;
CON = REF+TRN;

if(SSFREQ ~= -1)
  SSFPOWER = abs(SSFPOWER./SSFSRC).^2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Fields
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fig = figure;
SetFigure(fig, Title, [500 274 965 826]);

if(SSFREQ ~= -1)
  subplot(11,1,1:4);
end;

plot(FREQ, REF, '-r', 'LineWidth', 2); hold on;
plot(FREQ, TRN, '-b', 'LineWidth', 2);
plot(FREQ, CON, ':k', 'LineWidth', 3); hold off;
axis([FREQ(1) FREQ(NFREQ) -0.1 1.2]);
xlabel('Frequency');
title('Reflectance and Transmittance');
legend('Reflectance','Transmittance','Consersvation','Location','SouthEast');


if SSFREQ ~= -1
  subplot(11,1,8:11)
  plot(SSFPOWER, '-r', 'LineWidth', 2);
  axis([3 Nz-1  -0.1  1.2]);
  xlabel('Nz');
  title('Steady State Field');
```

```
end

end
```

BlindedMissile.m

```
%Blinded Missle

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Pre-Program Work
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize MATLAB
close all; clc;
clear all;

%Constants
c0 = 299792458; %m/s
e0 = 8.854187817*10^-12; %F/m
u0 = 1.256637061*10^-6; %H/m

nanometers = 1e-9;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Simulated Environment Settings
NLAM = 5e9;  % 5Ghz
lambda_0 = 980*nanometers; % Anti-reflective frequency;
lambda_min = 900*nanometers;
PERIODS = 15;

nSiN = 2.0;
erSiN = nSiN^2;

nSiO2 = 1.5;
erSiO2 = nSiO2^2;
nmax = nSiN;

%Calculate the Length of our layers.
LSiN = lambda_0/(4*nSiN);
LSiO2 = lambda_0/(4*nSiO2);

dc = LSiN;%meters Our critical dimension in this case is the width of one period.


%Compute Grid Resolution
N_lambda = 20;
d_wl = lambda_min/N_lambda/nmax;
N_d = 4;
d_d = dc/4; % since we are only working with freespace we will set d to 1;
dz = min(d_wl, d_d);
Nprime = ceil(dc/dz);
dz = dc/Nprime;

Nz = PERIODS*ceil((LSiN+LSiO2)/dz);


Nz = Nz + 2*(100) + 3;
```

```
%Grid Axis
za=[0:Nz-1]*dz;

%Compute Time Steps
dt = dz/(2*c0); %secs

% Source Parameters
nzc = 2;  %Position of Sources
NLAM = 100;
LAMBDA = linspace(900, 1100, NLAM)*nanometers; %FREQ List
tau = 0.5/(c0/lambda_min);        % tau parameter
t0 = 6*tau;              % Delay/Pulse Position

T = 12*tau + 5*(nmax*Nz*dz/c0);
STEPS = ceil(T/dt);
STEPS=20000;

ta = [0:STEPS-1]*dt;     % Time Axis;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Material Vectors
ER = ones([1 Nz]);
UR = ones([1 Nz]);

nstart = 100+2;
nend = nstart;


for p = 1:PERIODS
  nend = nstart + round(LSiN/dz)-1;
  ER(nstart:nend) = erSiN;

  nstart = nend+1;
  nend = nstart+ round(LSiO2/dz) -1;
  ER(nstart:nend) = erSiO2;

  nstart = nend+1;
end;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Source
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s = dz/(2*c0) + dt/2;    % Delay between E and H
Esrc = exp(-((ta-t0)/tau).^2); % E Source
A = -sqrt(ER(nzc)/UR(nzc));    % H Amplitude
Hsrc = A*exp(-((ta-t0+s)/tau).^2); % H Source



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FDTD Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Compute Update Coefficients
```

```
mER = (c0*dt/dz)./ER;
mHR = (c0*dt/dz)./UR;

% Initialize Feilds
Ey = zeros([1 Nz]);
Hx = zeros([1 Nz]);


%PAB Parameters
h1 = 0; h2 = 0; h3 = 0;
e1 = 0; e2 = 0; e3 = 0;

%Power Measurements
REF = zeros(1, NLAM);
TRN = zeros(1, NLAM);
SRC = zeros(1, NLAM);
K = exp(-1i*2*pi*dt*(c0./LAMBDA));

SSFK = exp(-1i*2*pi*dt*c0./lambda_0);
SSFPOWER = zeros(1, Nz);
SSFSRC = zeros(1, Nz);

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('% Parameters');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');

disp(['lamda_min: ' num2str(lambda_min)]);
disp(['d_lambda: ' num2str(d_wl)]);
disp(['nmax: ' num2str(nmax)]);
disp(['dc: ' num2str(dc)]);
disp(['d_d: ' num2str(d_d)]);
disp(['Nz: ' num2str(Nz)]);
disp(['dz: ' num2str(dz)]);
disp(['Length: ' num2str(Nz*dz)]);
disp(['dt: ' num2str(dt)]);
disp(['tau: ' num2str(tau)]);
disp(['t0: ' num2str(t0)]);
disp(['STEPS: ' num2str(STEPS)]);
disp(['s: ' num2str(s)]);
disp(['A: ' num2str(A)]);
% disp(['ER: ' num2str(length(ER))]);
% disp(ER);
% disp(['UR: ' num2str(length(UR))]);;
% disp(UR);
% return;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Execute Simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for t = 1:STEPS

  % Calculate H
  for nz = 1:Nz-1
    Hx(nz) = Hx(nz) + mHR(nz)*(Ey(nz+1)-Ey(nz));
  end
```

```
  Hx(Nz) = Hx(Nz) + mHR(Nz)*(e3 - Ey(Nz));

  %H Sources
  Hx(nzc-1) = Hx(nzc-1) - mHR(nzc-1)*Esrc(t);

  h3 = h2; h2 = h1; h1 = Hx(1); % Boundary Params;

  % Calculate E
  Ey(1) = Ey(1) + mER(1)*(Hx(1) - h3);
  for nz = 2:Nz
    Ey(nz) = Ey(nz) + mER(nz)*(Hx(nz)-Hx(nz-1));
  end

  %Inject Source
  Ey(nzc) = Ey(nzc) - mER(nzc)*Hsrc(t);

  e3=e2; e2=e1; e1=Ey(Nz); % Boundary Params;

 %Update Fourier Transforms
 for nf = 1: NLAM
   REF(nf) = REF(nf) + (K(nf)^t)*Ey(1)*dt;
   TRN(nf) = TRN(nf) + (K(nf)^t)*Ey(Nz)*dt;
   SRC(nf) = SRC(nf) + (K(nf)^t)*Esrc(t)*dt;
 end

  for n = 3 : Nz-1
   SSFPOWER(n) = SSFPOWER(n) + (SSFK^t)*Ey(n)*dt;
   SSFSRC(n) = SSFSRC(n) + (SSFK^t)*Esrc(t)*dt;
  end

 if(mod(t,1000) == 0)
   h = subplot(11,1,1:4);
   Draw1D(ER, Ey, Hx, dz);
   axis([za(1) za(Nz) -1.5 1.5]);
   xlabel('z');
   title(['Field at Step ' num2str(t) ' of ' num2str(STEPS)]);

   R = abs(REF./SRC).^2;
   T = abs(TRN./SRC).^2;

   subplot(11,1,8:11)
   plot(LAMBDA/nanometers, 10*log10(R), '-r'); hold on;
   plot(LAMBDA/nanometers, 10*log10(T), '-b');
   plot(LAMBDA/nanometers, 10*log10(R+T), ':k', 'LineWidth', 2); hold off;
   axis([LAMBDA(1)/nanometers LAMBDA(NLAM)/nanometers -50 0]);
   xlabel('Frequency');
   title('Reflectance and Transmittance');
   drawnow();
 end



  %if(mod(t,50) == 0)
  %  saveas(h, ['images/' num2str(t) '.jpg'], 'jpg');
  %end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute Values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

REF = abs(REF./SRC).^2;
TRN = abs(TRN./SRC).^2;
CON = REF+TRN;

SSFPOWER = abs(SSFPOWER./SSFSRC).^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Fields
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fig = figure;
SetFigure(fig, 'HW#6-P2-Blinded Missle', [500 274 965 826]);

subplot(11,1,1:4);
plot(LAMBDA/nanometers, 10*log10(R), '-r'); hold on;
plot(LAMBDA/nanometers, 10*log10(T), '-b', 'LineWidth', 2);
plot(LAMBDA/nanometers, 10*log10(R+T), ':k', 'LineWidth', 2); hold off;
axis([LAMBDA(1)/nanometers LAMBDA(NLAM)/nanometers -50 0]);
xlabel('Frequency');
title('Reflectance and Transmittance');
legend('Reflectance','Transmittance','Consersvation','Location','SouthEast');


subplot(11,1,8:11)
plot(SSFPOWER, '-r', 'LineWidth', 2);
axis([3 Nz-1  -0.1  5]);
xlabel('Nz');
title('Steady State Field');
```