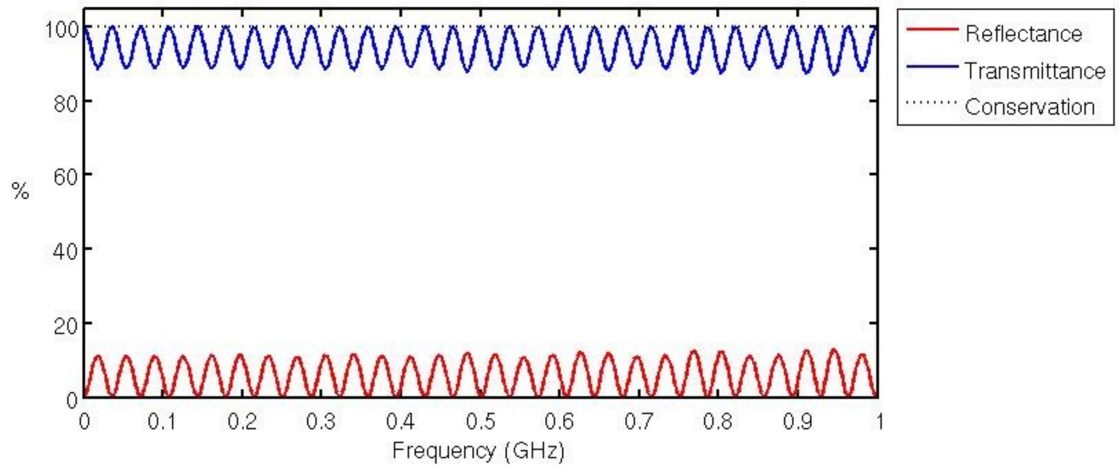
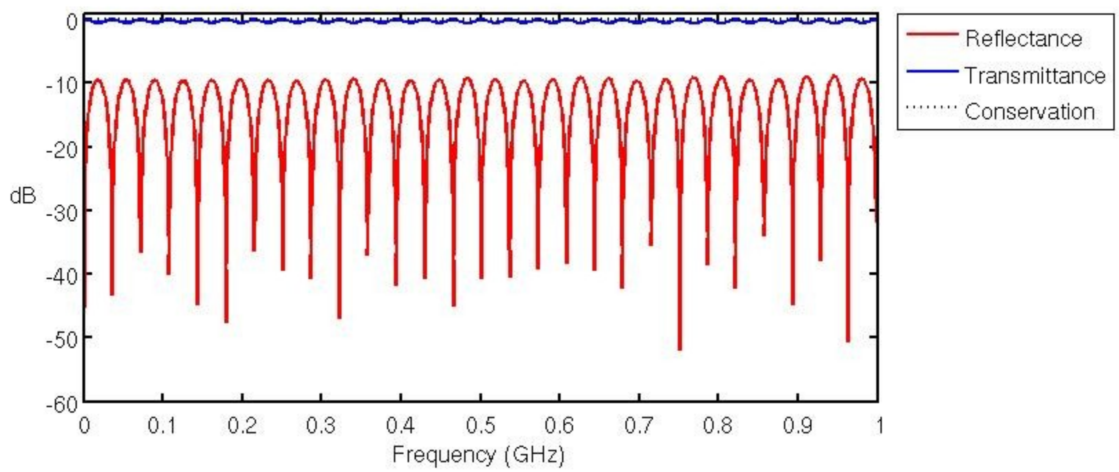
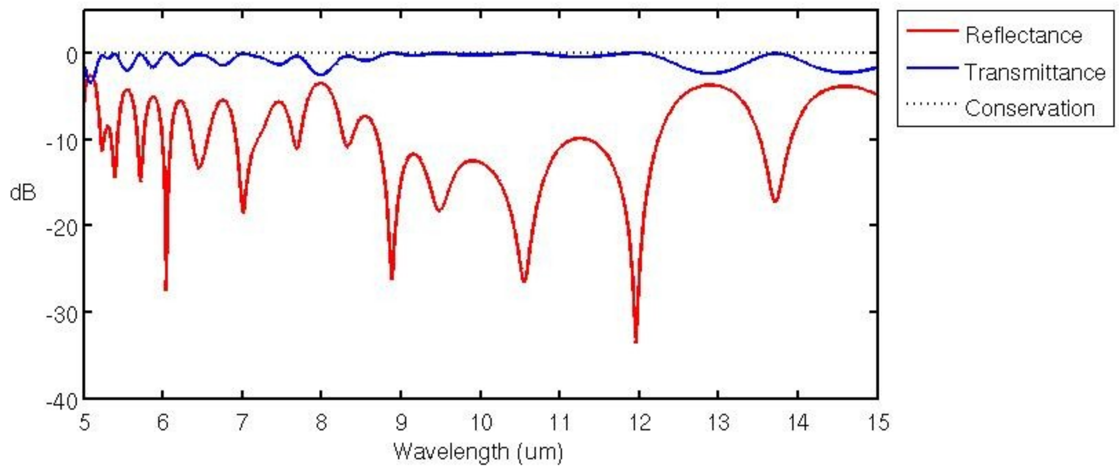
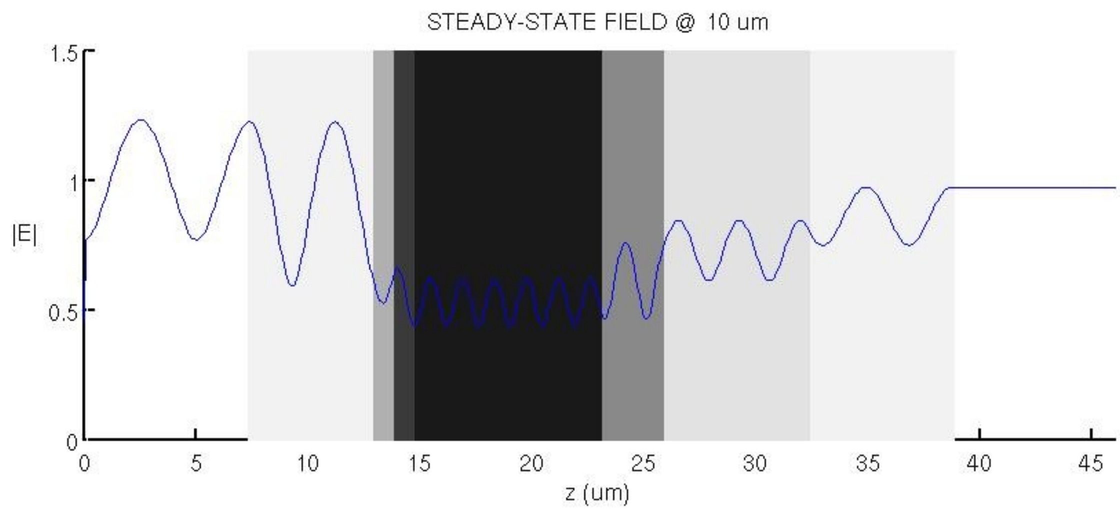
P1 – Magnetodielectric Device





P2 – Multi-Layer Device





Appendix A

Magnetodielectric.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Magnetodielectric Device Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize MATLAB
close all; clc;
clear all;

% UNITS
meters = 1;
decimeters = 1e-1 * meters;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
inches = 2.54 * centimeters;
feet = 12 * inches;
seconds = 1;
hertz = 1/seconds;
kilohertz = 1e3 * hertz;
megahertz = 1e6 * hertz;
gigahertz = 1e9 * hertz;


% Dimensions
% SLAB1
d1 = 1/decimeters; %cm thick
ur1 = 3;
er1 = 6;

% SLAB2
d2 = 0.5/decimeters; %cm thick
ur2 = 2;
er2 = 2;

% Set our critical dimension to SLAB2
dc = d2;

% Create our Material Vectors

rNz = ceil(d1+d2); %This Nz represents real world size

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

% Add our Slab materials to the model
rER(1:d1) = er1;
rUR(1:d1) = ur1;

rER(d1+1:d1+1+d2-1)= er2;
rUR(d1+1:d1+1+d2-1)= ur2;

% Frequency
```

```
freq_start = 0; %DC
freq_end = 1*gigahertz; %1Ghz

NFREQ = freq_end / (1*megahertz); %Frequencies every 100Mhz upto 10Ghz
FREQ = linspace(freq_start, freq_end, NFREQ); %FREQ List

Title = 'Exam #1 - Magnetodielectric Device';
[REF TRN CON] = FDTD1D( (dc)*decimeters, (d1+d2)*decimeters, rER, rUR, 15000, 100,
FREQ, NFREQ, 50, -1, Title );

PlotMag;
```

FDTD1D.m

```
function [oREF oTRN oCON] = 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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% UNITS
meters = 1;
decimeters = 1e-1 * meters;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
inches = 2.54 * centimeters;
feet = 12 * inches;
seconds = 1;
hertz = 1/seconds;
kilohertz = 1e3 * hertz;
megahertz = 1e6 * hertz;
gigahertz = 1e9 * hertz;

%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
nsrc = 1;  %Source is injected in air
dt = nsrc * 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 = 0;

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

disp(['f_max' num2str(f_max)]);
```

```matlab
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(['SSFREQ: ' num2str(SSFREQ)]);
% 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)
     SSFPOWER = SSFPOWER + (SSFK^t)*Ey*dt;
     SSFSRC = SSFSRC + (SSFK^t)*Esrc(t)*dt;
 end

 if(mod(t,Update) == 0 || t == 1)
   % draw field on top of materials
   subplot(311);
   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(312);
   plot(FREQ/gigahertz,R,'-r'); hold on;
   plot(FREQ/gigahertz,T,'-b');
   plot(FREQ/gigahertz,R+T,':k'); hold off;
   axis([FREQ(1)/gigahertz FREQ(NFREQ)/gigahertz -0.1 1.5]);
   xlabel('Frequency (GHz)');
   title('REFLECTANCE AND TRANSMITTANCE');

   % plot the steady-state field
   subplot(313);
   plot(za,abs(SSFPOWER/SSFSRC),'-b');
   axis([za(1) za(Nz) -0.1 1.5]);
   xlabel('z (meters)');
   title('STEADY-STATE FIELD');


   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);
end

oREF = REF;
oTRN = TRN;
```

```
oCON = CON;

end
```

PlotMag.m

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

subplot(211);
h = plot(FREQ/gigahertz,100*REF,'-r','LineWidth',2);
hold on;
plot(FREQ/gigahertz,100*TRN,'-b','LineWidth',2);
plot(FREQ/gigahertz,100*CON,':k','LineWidth',2);
hold off;
axis([FREQ(1)/gigahertz FREQ(NFREQ)/gigahertz 0 105 ]);
h2 = get(h,'Parent');
set(h2,'FontSize',14,'LineWidth',2);
h = legend('Reflectance','Transmittance','Conservation');
set(h,'Location','NorthEastOutside');
xlabel('Frequency (GHz)');
ylabel('%','Rotation',0,'HorizontalAlignment','right');


subplot(212);
h = plot(FREQ/gigahertz,10*log10(REF),'-r','LineWidth',2);
hold on;
plot(FREQ/gigahertz,10*log10(TRN),'-b','LineWidth',2);
plot(FREQ/gigahertz,10*log10(CON),':k','LineWidth',2);
hold off;
axis([FREQ(1)/gigahertz FREQ(NFREQ)/gigahertz -60 1 ]);
h2 = get(h,'Parent');
set(h2,'FontSize',14,'LineWidth',2);
h = legend('Reflectance','Transmittance','Conservation');
set(h,'Location','NorthEastOutside');
xlabel('Frequency (GHz)');
ylabel('dB','Rotation',0,'HorizontalAlignment','right');
```

MultiLayer.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Multi-Layer Device
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize MATLAB
close all; clc;
clear all;

% UNITS
meters = 1;
decimeters = 1e-1 * meters;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
micrometers = 1e-6 * meters;
nanometers = 1e-9 * meters;
picometers = 1e-12 * meters;
inches = 2.54 * centimeters;
feet = 12 * inches;
seconds = 1;
hertz = 1/seconds;
kilohertz = 1e3 * hertz;
megahertz = 1e6 * hertz;
gigahertz = 1e9 * hertz;

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

lambda_0 = 10*micrometers;

%Dimensions
d = [0.55504 0.1 0.1 0.9 0.27293 0.67907 0.56973]*lambda_0/micrometers;
n = [1.3045 2.3640 3.2847 3.5 2.7029 1.8344 1.3];
er = n.^2;

% Set our critical dimension to SLAB2
dc = min(d);

% Create our Material Vectors

rNz = ceil(sum(d))+1; %This Nz represents real world size

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

zstart = 1;
zend = zstart;

for i = 1 : length(d)
  zend = zstart + round(d(i))-1;
  rER(zstart:zend) = er(i);
  zstart = zend + 1;
end
```

```
NLAM = 15/0.01;
LAMBDA = linspace(5, 15, NLAM)*micrometers; %WL List

Title = 'Exam #1 - Multi-Layer Device';
[REF TRN CON ssEy ER dz za] = FDTD1DWL( (dc*micrometers), (sum(d)*micrometers),
rER, rUR, -1, 100, LAMBDA, NLAM, 50, lambda_0 , Title );

PlotMulti;
```

FDTDWL.m

```
function [oREF oTRN oCON oSSEy oER odz oza] = FDTD1DWL( dc, Length, rER, rUR,
Steps, Buffer, LAMBDA, NLAM, Update, SSFLAM, Title )
%FDTD1DWL Method executes a FDTD1D Model
%   Detailed explanation goes here

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

% UNITS
meters = 1;
decimeters = 1e-1 * meters;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
nanometers = 1e-9 * meters;
inches = 2.54 * centimeters;
feet = 12 * inches;
seconds = 1;
hertz = 1/seconds;
kilohertz = 1e3 * hertz;
megahertz = 1e6 * hertz;
gigahertz = 1e9 * hertz;

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

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

nmax = Getnmax(rER, rUR);

%Compute Grid Resolution
% Wave Length Rsolution
N_lambda = GetNlambda(rER, rUR);
lambda_min = LAMBDA(1);
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
nsrc = 1;  %Source is injected in air
dt = nsrc * dz/(2*c0); %secs

% Source Parameters
nzc = 2;  %Position of Sources at our TF/SF boundary
tau = 0.5/(c0/lambda_min);        % 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, NLAM);
TRN = zeros(1, NLAM);
SRC = zeros(1, NLAM);
K = exp(-1i*2*pi*dt*(c0./LAMBDA));

SSFK = exp(-1i*2*pi*dt*c0./SSFLAM);
SSFPOWER = zeros(1, Nz);
SSFSRC = 0;

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

%disp(['f_max' num2str(f_max)]);
```

```matlab
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(['SSFREQ: ' num2str(SSFLAM)]);
disp(['cf: ' num2str(cf)]);
% disp(['ER: ' num2str(length(ER))]);
% disp(ER);
% disp(['UR: ' num2str(length(UR))]);;
% disp(UR);
% oREF = -1;
% oTRN = -1;
% oCON = -1;
% oSSEy = -1;
% oER = -1;
% odz = -1;
% oza = -1;
% 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


 if(SSFLAM ~= -1)
     SSFPOWER = SSFPOWER + (SSFK^t)*Ey*dt;
     SSFSRC = SSFSRC + (SSFK^t)*Esrc(t)*dt;
 end

 if(mod(t,Update) == 0 || t == 1)
   % draw field on top of materials
   subplot(311);
   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(312);
   plot(LAMBDA/nanometers,R,'-r'); hold on;
   plot(LAMBDA/nanometers,T,'-b');
   plot(LAMBDA/nanometers,R+T,':k'); hold off;
   axis([LAMBDA(1)/nanometers LAMBDA(NLAM)/nanometers -0.1 1.5]);
   xlabel('Frequency (GHz)');
   title('REFLECTANCE AND TRANSMITTANCE');

   % plot the steady-state field
   subplot(313);
   h = Draw1D(ER,abs(SSFPOWER),-1, dz/nanometers);
   %axis([za(1)/nanometers za(Nz)/nanometers 0 (max(SSFLAM))]);
   xlabel('z (meters)');
   title('STEADY-STATE FIELD');


   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(SSFLAM ~= -1)
  SSFPOWER = abs(SSFPOWER/SSFSRC);
end

oREF = REF;
oTRN = TRN;
oCON = CON;

if(SSFLAM ~= -1)
  oSSEy = SSFPOWER;
else
  oSSEy = -1;
end

oER = ER;
odz = dz;
oza = za;
end
```

PlotMulti.m

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

subplot(211);
h = plot(LAMBDA/micrometers,10*log10(REF),'-r','LineWidth',2);
hold on;
plot(LAMBDA/micrometers,10*log10(TRN),'-b','LineWidth',2);
plot(LAMBDA/micrometers,10*log10(CON),':k','LineWidth',2);
hold off;
axis([LAMBDA(1)/micrometers LAMBDA(NLAM)/micrometers -40 5]);
h2 = get(h,'Parent');
set(h2,'FontSize',14,'LineWidth',2);
h = legend('Reflectance','Transmittance','Conservation');
set(h,'Location','NorthEastOutside');
xlabel('Wavelength (um)');
ylabel('dB','Rotation',0,'HorizontalAlignment','right');



subplot(212);
h = Draw1D(ER,abs(ssEy),-1, dz/micrometers);
axis([za(1)/micrometers za(length(za))/micrometers 0 1.5 ]);
xlabel('z (nm)');
h2 = get(h,'Parent');
set(h2,'FontSize',14,'LineWidth',2);
title(['STEADY-STATE FIELD @ ' num2str(lambda_0/micrometers) ' um']);
xlabel('z (um)');
ylabel('|E|','Rotation',0,'HorizontalAlignment','right');
```

Draw1D.m

```
function h = Draw1D( ER, E, H, dz )
   persistent Color;

   %Initialize
   ze=[0:length(E)-1]*dz;
   zh = ze + dz/2;

   if isempty(Color)
     % Just inverse our Permitivity to get grayscale value
     ER = ER - min(ER(:));

     Color = (ER/max(ER(:)));

     for i = 1 : length(Color)
       if Color(i) < .15 && Color(i) > 0
         Color(i) = 0.15;
       end
     end

     Color = abs(Color - 1.10);

     for i = 1 : length(Color)
       if Color(i) > 1
         Color(i) = 1;
       end
     end
   end

   % Need to do an initial draw so we can start the hold for plotting.
   cla;   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 = [ -2 -2 2 2 -2 ];
      fill(x,y,[Color(i-1) Color(i-1) Color(i-1)], 'LineStyle', 'none', 'Marker',
'none');

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

```
    end

    %Plot Fields
    h = plot(ze, E, '-b');

    if(H ~=-1)
      plot(zh, H, '-r');
    end;

    hold off;
end
```