Table of Contents


```
function Abbas_HW8_Luebbers()

% Hasan Tahir Abbas
% ECEN 637
% Homework 8: Reflection coefficient of a Dielectric Slab
% Conditions
% 11/03/2015
%
%
%  1D FDTD dielectric slab reflection coefficient code
%  recreating Leubber's results in Fig 1 and 6b
%
% Soft Gaussian pulse excitation
% observation of reflected field
% Calculation of DFT and reflection coefficient
% Total absorption at boundaries
%
%   c  =   speed of light
%   xmu  =  free-space permeability
%   eps0  = free-space permittivity
%   epsR  = dielectric constant of the slab
%   sigma  = conductivity of the slab
%   rho_prime  = magnetic conductivity of the slab
%   nx  = spatial intervals in space
%   nt  = total number of time intervals
%   nxst  = starting position of the slab
%   nxnd  = ending position of the slab
%   slabwidth  = width of the slab in spatial intervals
%   dt  = time step size
%   dx  = spatial step size
%   n  = time incrementing variable
%   num_freq  = number of frequencies needed to compute the fourier
 transform
%   num_samples  = power of 2 number to obtain frequency resolution
```

```
%   Ez  = z-component of Electric Field
%   Hy  = y-component of Magnetic Field
%  mediaEz  = array to define the structure in terms of E-field points
%  mediaHy  = array to define the structure in terms of H-field points
%   Einc  = incident Electric Field
%   Hinc  = incident Magnetic Field
%  Ca Cb Da Db  = Coefficient terms as defined in Taflove's book (Sec.
 3.6.4)
%  coll_pt = time domain field data collection point
%  ftEinc ftEref = Fourier Transform of inc. and ref. fields
%  gamma
%  d_gamma = slab electric conductivity
%  d_num_freq
%   fmax  = maximum frequency to be plotted
%  Source_signal = Time-domain source signal
%  beta  = Variance of the Gaussian source
%  xp  = Location of source excitation
%  source_type  = Type of Source 1. Sinusoidal 2. Gaussian
%  nsnap  = Plot after every nsnap intervals
%  ymin ymax  = min/max limits in the field plots
%  plot_record  = Time instants that replicate plots in Luebber's
 paper
%   max_plot  = maximum time step upto which field is displayed
clear;close all
% ************************
```

Parameters

****

Global variables are used to span across all the functions in this code According to MATLAB's documentation, a better and safer option will be persistent type variables

```
global c xmu eps0 epsR sigma rho_prime
global nx nt nxst nxnd slabwidth
global dt dx n
global num_freq num_samples
global Ez  Hy; % Create E and H field components.
global mediaEz mediaHy %
global Hyinc Ezinc
global Ca Cb Da Db % Define material based coefficients
global coll_pt % Collection point for reflection coefficient
 calculation
global ftEinc ftEref
global gamma d_gamma d_num_freq
global Source_signal beta xp source_type
global nsnap ymin ymax plot_record max_plot fmax

c = 2.99792458e8; % Speed of light
xmu = 4*pi*1e-7;  % Permeability of free space
eps0 = 8.854187817e-12; % Permittivity of free space
epsR = 4;
sigma = 0;
rho_prime = 0;
```

```
slabwidth = 0.09;
nx = 600;      % Number of cells in x-direction
nt = 4096;     % Number of time steps
nxst = 250;    % Start of slab
nxnd = 309;    % End of slab
num_freq = 500;
num_samples = 16384; % 2^14
xp = 50;
nsnap = 5;
coll_pt = 240;
beta = 10;
ymin = -1;
ymax = 1;
plot_record = [ 190, 245, 315, 370, 385, 485];
max_plot = 2000;
fmax = 5e9;
% ************************
```

Initialize

```
************************

Ca = zeros(2,1);
Cb = zeros(2,1);
Da = zeros(2,1);
Db = zeros(2,1);

mediaEz = ones(1,nx);
mediaHy = ones(1,nx); %%!Field medium index
gamma = zeros(1,num_freq);
d_gamma = zeros(1,num_freq);
d_num_freq = zeros(1,num_freq);
Source_signal = zeros(1,nt);

Ez = zeros(1,nx);    %%%!zero the incident and total fields
Hy = zeros(1,nx);
Ezinc = zeros(1,nx);
Hyinc = zeros(1,nx);

ftEinc = zeros(1,num_freq);  %%%!zero the DFTs
ftEref = zeros(1,num_freq);


dx = slabwidth/(nxnd - nxst + 1); %% Length Increment
% ************************
```

Stability Condition

```
************************

dt = dx/(c); % Stability Condition
% ************************
% ***********************
```

```
%#########################################################################
% ************************
% ************************
```

Main Program

```
*********************** ***********************

define_media();  % Create the dielectric slab between nxst and nxnd
define_coefficients(); % Generate coefficients in different media
source_type = 2; % 1 is sinusoidal source,2 is Gaussian, 3 is unit-
step

for n = 1 : nt
    adv_Ez(); % Compute E-field
    adv_H(); % Compute H-field
    field_plot(); % Plot E-field only upto 2000 points but calculate
 till nt
    ft_field(); % Compute fourier transform of E-field
end
reflection(); % Compute field reflection coefficient in frequency
 domain
spectrum_plot(); % Plot the reflection and transmission coefficents
%#########################################################################

end
% ************************
% ************************
%
```

Create Coefficients for the equations

```
*********************** ***********************

function define_coefficients()

global Ca Cb Da Db ; % Define material based coefficients
global xmu eps0 epsR rho_prime sigma
global dt dx
% % % % % % % % Field Coefficients

Ca(1) = (1 - sigma * dt / (2 * eps0) ) / ( 1 + sigma * dt / (2 *
 eps0) );
Cb(1) = (dt / (eps0 * dx) )/( 1 + sigma * dt /(2 * eps0) );
Da(1) = (1 - rho_prime * dt/(2*xmu) )/(1 + rho_prime *dt/(2*xmu) );
Db(1) = (dt/(xmu * dx) )/(1 + rho_prime * dt/(2 * xmu) );

% % !  slab coefficients

Ca(2) = (1 - sigma * dt / (2 * epsR * eps0) )/(1 + sigma * dt /(2 *
 epsR * eps0) );
Cb(2) = (dt /(epsR * eps0 * dx) ) / (1 + sigma * dt/(2 * epsR *
 eps0) );
```

```
Da(2) = Da(1);
Db(2) = Db(1);

end
% ************************
% ************************
%
```

Create Structure in the computational space

```
*********************** ***********************

function define_media()

global nxst nxnd;
global mediaEz mediaHy;

for i = nxst:nxnd  % insert the slab into free space

    mediaEz(i) = 2;  % Ez exist at ever index point in the slab

end

for i = nxst:nxnd-1  % Hy does not extend to the right index of the
 slab

    mediaHy(i) = 2;

end

end
% ************************
% ************************
%
```

Create the excitation signal

```
*********************** ***********************

function Ezs = Source()

global beta xp
global n source_type
% Creates a half-sinusoidal source between the time increments
% 1 and 10.%
% When source = 1 : Sinusoid
%               2 : Gaussian
%               3 : Unit-Step
%
% For Sinusoidal Source
if source_type == 1
    if ( (n-xp) >=1 && (n-xp) <= xp)
        Ezs = sin((n-xp)*pi/xp);
```

```
    else
        Ezs = 0;
    end
% For Gaussian Source
elseif source_type == 2
    xn0 = 4*beta;
    Ezs = exp(-((n-xn0)/(beta))^2);
% For Pulse Source
elseif source_type == 3
    if ( (n-xp) >=1 && (n-xp) <= xp)
        Ezs = 1;
    else
        Ezs = 0;
    end
end
end


% ************************
% ************************
```

Algorithm for Computing E-field

```
*********************** **********************

function adv_Ez()
% Compute z-component of E-field
global Ez Hy Hyinc Ezinc
global mediaEz
global Source_signal
global Ca Cb
global nx xp
global n

Ez(1) = Ez(2);     % left side total field perfect ABC
Ez(nx) = Ez(nx-1);   % right side total field perfect ABC

Ezinc(1) = Ezinc(2);  % left side incident field perfect ABC
Ezinc(nx) = Ezinc(nx-1); % right side incident field perfect ABC

for i = 2 : nx-1   % Ez Total (field with slab)

    if (i == xp)

        Ezs = Source(); % Incident field source excitation
        Source_signal(n) = Ezs;

    else

        Ezs = 0;

    end
    m = mediaEz(i);
```

```
    Ez(i) = Ez(i) * Ca(m) + ( Hy(i) - Hy(i-1) ) * Cb(m)...
        + Ezs;      % soft source



end

for i = 2 : nx-1    % Ez Incident (no slab)

    if (i == xp)

        Ezs = Ez_inc(n); % Incident field source excitation

    else

        Ezs = 0;

    end
    m = 1;
    Ezinc(i) = Ezinc(i) * Ca(m)...
        + ( Hyinc(i) - Hyinc(i-1) ) * Cb(m)...
        + Ezs;              % soft source
end
end


% ************************
% ************************
```

Algorithm for Computing H-field

```
*********************** ***********************

function adv_H()
% Compute z-component of E-field
global Ez  Hy Hyinc Ezinc
global mediaHy
global Da Db
global nx

% % %     Compute x-component of total H-field

for i = 1 : nx - 1

    m = mediaHy(i);
    Hy(i) = Hy(i) * Da(m)...
        + ( Ez(i+1) - Ez(i) ) * Db(m);
end

% % %     Compute x-component of incident H-field
for i = 1 : nx - 1

    m = 1;
```

```
    Hyinc(i) = Hyinc(i) * Da(m)...
        + ( Ezinc(i+1) - Ezinc(i) ) * Db(m);

end
end


% ************************
% ************************
```

Algorithm for Computing Fourier Transform E-field

```
*********************** ***********************

function ft_field()

global coll_pt num_freq num_samples
global Ez Ezinc
global n dt ftEinc ftEref

Einc = Ezinc(coll_pt);
Eref = Ez(coll_pt)-Einc;

for k = 1 : num_freq

    dft_exp = -2 * 1i * pi * k * n / num_samples;
    ftEinc(k) = ftEinc(k) + dt * Einc * exp(dft_exp); %%%!incident
 field dft
    ftEref(k) = ftEref(k) + dt * Eref * exp(dft_exp); %%%!reflected
 field dft

end

end

% ************************
% ************************
```

Algorithm for Computing Reflection Coefficient

```
*********************** ***********************

function reflection()
global gamma num_freq num_samples
global d_gamma d_num_freq
global dt ftEinc ftEref

    for i = 1:num_freq

        if ( abs( ftEinc(i) ) > 0.0 )
```

```
            gamma(i) = ( abs(ftEref(i) ) ) / ( abs( ftEinc(i) ) );

        else

            gamma(i) = 0;

        end

        d_num_freq(i) = i * (1  / (num_samples * dt) );
        d_gamma(i) = gamma(i);

    end
end


% ************************
% ************************
```

Plot routine for the Line Plot of Ez

```
*********************** ***********************

function field_plot()
% Plots the line plots for E-field
global n nx nsnap nxst nxnd
global Ez
global ymin ymax plot_record max_plot

if (rem(n,nsnap) == 0 && n <= max_plot) % Plot at every 5th time step
    figure(1);
    set(gcf,'Color','white');
    plot(Ez,'Color','black','LineWidth',1.4)
    set (gca,'FontName','times new roman')
    hold on
    title(['E-field at Time Step
 ',int2str(n)],'FontSize',11,'Interpreter','latex','FontName','times
 new roman')
    h = rectangle('Position',[nxst ymin nxnd-nxst ymax-ymin]);
    h.FaceColor = 'none';
    h.EdgeColor = [.4 .4 .4];
    axis([0 nx ymin ymax])
    xlabel('Position (1.5 cells)')
    ylabel('Electric Field')
    hold off
    pause(.001)
    if (ismember(n,plot_record)) % Save Plot at every 20th time step
        %          cleanfigure();
        %          matlab2tikz('filename',sprintf('ECEN637_HW8_Ez_
%d.tex',n));
        saveas(gcf,[sprintf('ECEN637_HW7_Ez_surf_PEC_space_
%d',n),'.png']) % Save visualizations
    end
end
```

```
end

% ************************
% ************************
```

Plot routine for the Spectrum Plot of the field

```
*********************** ***********************

function spectrum_plot()
% Plots the line plots for E-field
global d_num_freq d_gamma
global fmax

figure(2)
plot(d_num_freq/1e9,d_gamma,'Color','black','LineWidth',1.4);
set(gcf,'Color','white');
set (gca,'FontName','times new roman')
hold on
title('Fourier Transform of the reflected
 field','FontSize',11,'Interpreter','latex','FontName','times new
 roman')
h = line([0 5],[.6 .6]);
h.Color = [.4 .4 .4];
h.LineStyle = '--';
axis([0 fmax/1e9 0 1])
xlabel('Frequency (GHz)')
ylabel('Magnitude $\Gamma$','Interpreter','latex')
hold off
%     cleanfigure();
%     matlab2tikz('filename',sprintf('ECEN637_HW8_Spectrum.tex'));
saveas(gcf,[sprintf('ECEN637_HW8_Spectrum'),'.png']) % Save
 visualizations


figure(3)
plot(d_num_freq/1e9,sqrt(1-
(d_gamma.^2)),'Color','black','LineWidth',1.4);
set(gcf,'Color','white');
set (gca,'FontName','times new roman')
hold on
title('Fourier Transform of the transmitted
 field','FontSize',11,'Interpreter','latex','FontName','times new
 roman')
% h = line([0 5],[.6 .6]);
% h.Color = [.4 .4 .4];
% h.LineStyle = '--';
axis([0 fmax/1e9 0.5 1.5])
xlabel('Frequency (GHz)')
ylabel('Magnitude $\textit{T}$','Interpreter','latex')
hold off
% cleanfigure();
```

```
%
 matlab2tikz('filename',sprintf('ECEN637_HW8_Transmission_Spectrum.tex'));
saveas(gcf,[sprintf('ECEN637_HW8_Transmission_Spectrum'),'.png']) %
 Save visualizations

end
```

Published with MATLAB® R2015b