Table of Contents


```
function Abbas_HW_9_Radpat_final()

% Hasan Tahir Abbas
% ECEN 637
% Homework 9: Far-field Calculation of a PEC Box
% 11/17/2015
%
%
%  2D FDTD far-field plot of a PEC box using near-field to far-field
%  transformation based on Umashankar and Taflove's paper:
%  METHOD TO ANALYZE EM SCATTERING
%
% Soft Gaussian pulse excitation
% Equivalent Current Source modeling
% Calculation of DFT of the scattered field
% Far-field transformation of the near-field
%
% PARAMETERS AND VARIABLES LIST
% -----------------------------
%   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
%   finc  =  Frequency of the source signal
%   lambda_0 = Free-space wavelegngth of source signal
%   k = Propagation constant of the signal
```

```
%   eta_0 = free-space characteristic impedance
%   nx  = x-spatial intervals in space
%   nx2 = x-Center of the computational space
%   ny  = y-spatial intervals in space
%   ny2 = = x-Center of the computational space
%   nda = Number of angles to be computed
%   nt = Number of time steps
%   mxst = X-Start of box
%   mxnd = X-End of box
%   myst = Y-Start of box
%   mynd = Y-End of box
%   mxcl = X-Start of contour path
%   mxcr = X-End of contour path
%   mycb = Y-Start of contour path
%   myct = Y-End of contour path
%   mxcite = Location of excitation
%   dt  = time step size
%   dx  = spatial step size
%   n  = time incrementing variable
%   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
%   Ezinc  = incident Electric Field
%  Ca Cb Da Db  = Coefficient terms as defined in Taflove's book (Sec.
 3.6.4)
%   ABC_Order = Order of the Liao Absrobing Boundary conditions
%   flag_medium = Flag to specify media
%         1. Free-space
%         2. PEC
%  Source_signal = Time-domain source signal
%  beta  = Variance of the Gaussian source
%  mxcite  = Location of source excitation
%  source_type  = Type of Source
%         1. Sinusoidal
%         2. Gaussian
%         3. Pulse
%
% FUNCTIONS LIST
% -----------------------------
%   initialize() = Set arrays to be used to zero
%   define_media() = Create the structure between mxst,myst and
 mxnd,mynd depending on flag_medium
%   define_coefficients() = Generate coefficients in different media
%   define_Liao() = Computes the co-efficints for the given order to
 be
%                    used in Liao ABC
%   adv_Ez() = E-field computation part of the FDTD method. Calculates
 both
%              total and incident fields
%   Liao_ABC() = Implements Liao ABC of the given order (ABC_Order)
%   adv_H() = H-field computation part of the FDTD method. Calculates
 both
%              Hx,Hy total and incident fields
```

```
%   ft_field() = DFT calculation of the scattered field and the
 incident
%                wave
%   Source() = Creaates the excitation signal based on source_type
%           1. Sinusoidal
%           2. Gaussian
%           3. Pulse
%   far_field() = Models the Contour integration of the equivalent
 sources
%                 as a sum and computes it
%   plot_far_field() = Plots the far-field in both polar and cartesian
%                     co-ordinates
% -----------------------------
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 k lambda_0 eta_0
global nx ny nt nx2 ny2
global mxst mxnd myst mynd
global mxcl mxcr mycb myct
global mxcite
global dt dx n ds
global ABC_order % Order of the Liao ABC
global beta source_type
global finc phi
global d b nda
global flag_medium

% *************************
% *************************
% Data
% *************************
% *************************
c = 2.99792458e8; % Speed of light
xmu = 4*pi*1e-7;  % Permeability of free space
eps0 = 8.854187817e-12; % Permittivity of free space
epsR = 0;
sigma = 0;
rho_prime = 0;
finc = 300e6; % 300 MHz
lambda_0 = c/finc;
k = 2*pi/lambda_0;
eta_0 = sqrt(xmu/eps0);
nx = 150;      % Number of cells in x-direction
nx2 = nx/2;
ny = 150;
```

```
ny2 = ny/2;
nda = 361;
nt = 2000;     % Number of time steps
mxst = nx2 - 10;    % X-Start of box
mxnd = nx2 + 10;    % X-End of box
myst = ny2 - 10;    % X-Start of box
mynd = ny2 + 10;    % X-End of box
mxcl = mxst - 10;
mxcr = mxnd + 10;
mycb = myst - 10;
myct = mynd + 10;
mxcite = mxcl - 15;
beta = 10;
phi = linspace(0,2*pi,nda);
%###################################
dx = lambda_0/(20*pi); %% Length Increment
ds = dx;
d = 20*dx;
b = 20*dx;
```

Stability Condition

```
dt = dx/(c*(2)); % Stability Condition
%###################################
```

Main Program

*********************** *********************** *********************** First Time sweep for Incident field ***********************

```
initialize();
flag_medium = 1; % Free-space computational domain
define_media();  % Create the structure between mxst,myst and
 mxnd,mynd depending on flag_medium
define_coefficients(); % Generate coefficients in different media
source_type = 2; % 1 is sinusoidal source,2 is Gaussian, 3 is unit-
step
ABC_order = 4;
define_Liao(); % 4th order LIAO ABC (other options 3 or 5)
% First Time sweep for Total field
for n = 1 : nt
    adv_Ez(); % Compute E-field
    Liao_ABC(); % Invoke ABC algorithm
    adv_H(); % Compute H-field
%         if rem(n,5) == 0 && n < 500 % Plot at every 5th time step
%          figure(1);
%          my_surface_plot(Ezi);
%         end
    ft_field(); % Compute fourier transform of E-field
end
% ************************
% Second Time sweep for Total field
% ************************
```

```
flag_medium = 2; % PEC box in the computational domain
define_media();  % Create the structure between mxst,myst and
 mxnd,mynd depending on flag_medium
define_coefficients(); % Generate coefficients in different media
define_Liao(); % 4th order LIAO ABC (other options 3 or 5)
for n = 1 : nt
    adv_Ez(); % Compute E-field
    Liao_ABC(); % Invoke ABC algorithm
    adv_H(); % Compute H-field
%          if rem(n,5) == 0 && n < 500 % Plot at every 5th time step
%           figure(2);
%           my_surface_plot(Ez);
%          end
    ft_field(); % Compute fourier transform of E-field
end
far_field(); % Sums the FT-fields to compute far-field (Eq. 24)
plot_far_field(); % Plots the far-field in polar and rectangular plots

%##########################################################################

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

Initialize

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

function initialize()
% Set all the variables to zero

global nx ny
global mxcl mxcr mycb myct
global Ez Hx Hy; % Create E and H field components.
global mediaEz mediaHx mediaHy %
global Hxi Hyi Ezi
global Ca Cb Da Db % Define material based coefficients
global Ez1 Ez2 Ez3 Ez4 Ez5 % For Bubbling of total E-fields in Liao
 ABC
global Ez1i Ez2i Ez3i Ez4i Ez5i % For Bubbling of incident E-fields in
 Liao ABC
global ftEz_right ftEz_top ftEz_left ftEz_bottom
global ftHx_top ftHx_bottom
global ftHy_right ftHy_left
global Ez_norm Ezi_norm ftEinc

% FDTD Fields
% ************************
% Total Fields
Ez = zeros(nx,ny); %% z-component of total E-field
Hx = zeros(nx,ny); %% x-component of total H-field
Hy = zeros(nx,ny); %% y-component of total H-field
```

```
% Incident Fields
Ezi = zeros(nx,ny); %% z-component of incident E-field
Hxi = zeros(nx,ny); %% x-component of incident H-field
Hyi = zeros(nx,ny); %% y-component of incident H-field

% ************************
% Medium Arrays
% ************************
mediaEz = ones(nx,ny); %% z-component of E-field
mediaHx = ones(nx,ny); %% x-component of H-field
mediaHy = ones(nx,ny); %% x-component of H-field

% ************************
% FDTD Equation Coefficients
% ************************
Ca = zeros(2,1); %% x-component of H-field
Cb = zeros(2,1); %% x-component of H-field
Da = zeros(2,1); %% x-component of H-field
Db = zeros(2,1); %% x-component of H-field

% ************************
% Liao Bouncing terms
% ************************
% Total field
Ez1 = zeros(nx,ny);
Ez2 = zeros(nx,ny);
Ez3 = zeros(nx,ny);
Ez4 = zeros(nx,ny);
Ez5 = zeros(nx,ny);

% Incident field
Ez1i = zeros(nx,ny);
Ez2i = zeros(nx,ny);
Ez3i = zeros(nx,ny);
Ez4i = zeros(nx,ny);
Ez5i = zeros(nx,ny);

% ************************
% Fourier terms
% ************************
% Total Ez-field
ftEz_right = zeros(1,myct - mycb);  % !zero the DFTs
ftEz_top = zeros(1,mxcr - mxcl);
ftEz_left = zeros(1,myct - mycb);
ftEz_bottom = zeros(1,mxcr - mxcl);

% Total Hx-field
ftHx_top = zeros(1,mxcr - mxcl);
ftHx_bottom = zeros(1,mxcr - mxcl);

% Total Hy-field

ftHy_right = zeros(1,myct - mycb);
ftHy_left = zeros(1,myct - mycb);
```

```
Ez_norm = zeros(1, length(0 : pi/180 : 2*pi));
Ezi_norm = zeros(1, length(0 : pi/180 : 2*pi));
ftEinc = 0;

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

Create Coefficients for the equations

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

function define_coefficients()

global Ca Cb Da Db ; % Define material based coefficients
global xmu eps0
global dt ds
% % % % % % % % Field Coefficients

dte = dt/(ds*eps0);
dtm = dt/(ds*xmu);

Da(1) = 1;
Db(1) = dtm;
Ca(1) = 1;
Cb(1) = dte;
% % !  PEC Box coefficients

Da(2) = 0;
Db(2) = 0;
Ca(2) = 0;
Cb(2) = 0;

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

Create Structure in the computational space

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

function define_media()

global nx  ny mxst mxnd myst mynd
global mediaEz mediaHx mediaHy
global flag_medium
if (flag_medium == 2)
```

```
    for  i = 1:nx
        for j = 1:ny
            if (i >= mxst && i <= mxnd)
                if ( j >= myst && j <= mynd)
                    mediaEz(i,j) = 2;
                end
            end
        end
    end

    for  i = 1:nx
        for j = 1:ny
            if (i >= mxst && i <= mxnd)
                if ( j >= myst && j <= mynd-1)
                    mediaHx(i,j) = 2;
                end
            end
        end
    end

    for  i = 1:nx
        for j = 1:ny
            if (i >= mxst && i <= mxnd-1)
                if ( j >= myst && j <= mynd)
                    mediaHy(i,j) = 2;
                end
            end
        end
    end
end

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

Create Coefficients for LIAO ABC

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

function define_Liao()

global c1 c2 c3 c4 c5 ; % Define material based coefficients
global ABC_order % Order of the Liao ABC
switch ABC_order
    case 5 %% 5th order LIAO ABC Coefficients
        c1=5;
        c2=10;
        c3=10;
        c4=5;
        c5=1;

    case 4 %% 4th order LIAO ABC Coefficients
```

```
            c1 = 4;
            c2 = 6;
            c3 = 4;
            c4 = 1;
            c5 = 0;

        case 3 %% 3rd order LIAO ABC Coefficients
            c1 = 3;
            c2 = 3;
            c3 = 1;
            c4 = 0;
            c5 = 0;

        otherwise
            disp('Error: Wrong Value');
    end
    end

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

Implement LIAO ABC

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

    function Liao_ABC()

    global c1 c2 c3 c4 c5 ; % Define material based coefficients
    global Ez Ezi; % E field component.
    global Ez1 Ez2 Ez3 Ez4 Ez5 % For Bubbling of total E-fields in Liao
     ABC
    global Ez1i Ez2i Ez3i Ez4i Ez5i % For Bubbling of incident E-fields in
     Liao ABC
    global nx  ny
    global flag_medium ABC_order


    % General BC for any order LIAO ABC
    if (flag_medium == 1)
        for  j = 1:ny
            Ezi(1,j) = c1*Ez1i(2,j)-c2*Ez2i(3,j)+c3*Ez3i(4,j)...
                -c4*Ez4i(5,j)+c5*Ez5i(6,j);  %%%left side
        end
        for j = 1:ny
            Ezi(nx,j) = c1*Ez1i(nx-1,j)-
    c2*Ez2i(nx-2,j)+c3*Ez3i(nx-3,j) ...
                -c4*Ez4i(nx-4,j)+c5*Ez5i(nx-5,j);  %%%right side
        end
        for i = 2:nx-1
            Ezi(i,1) = c1*Ez1i(i,2)-c2*Ez2i(i,3)+c3*Ez3i(i,4) ...
                -c4*Ez4i(i,5)+c5*Ez5i(i,6); %%%bottom
        end
```

```
    for i = 2:nx-1
        Ezi(i,ny) = c1*Ez1i(i,ny-1)-
c2*Ez2i(i,ny-2)+c3*Ez3i(i,ny-3) ...
            -c4*Ez4i(i,ny-4)+c5*Ez5i(i,ny-5); %%%top
    end
    switch ABC_order

        case 5

            Ez5i = Ez4i;
            Ez4i = Ez3i;
            Ez3i = Ez2i;
            Ez2i = Ez1i;
            Ez1i = Ezi;
        case 4

            Ez4i = Ez3i;
            Ez3i = Ez2i;
            Ez2i = Ez1i;
            Ez1i = Ezi;
        case 3

            Ez3i = Ez2i;
            Ez2i = Ez1i;
            Ez1i = Ezi;
        otherwise
            disp('Error: Wrong Value');
    end


elseif (flag_medium == 2)

    for  j = 1:ny
        Ez(1,j) = c1*Ez1(2,j)-c2*Ez2(3,j)+c3*Ez3(4,j)...
            -c4*Ez4(5,j)+c5*Ez5(6,j);  %%%left side
    end
    for j = 1:ny
        Ez(nx,j) = c1*Ez1(nx-1,j)-c2*Ez2(nx-2,j)+c3*Ez3(nx-3,j) ...
            -c4*Ez4(nx-4,j)+c5*Ez5(nx-5,j);  %%%right side
    end
    for i = 2:nx-1
        Ez(i,1) = c1*Ez1(i,2)-c2*Ez2(i,3)+c3*Ez3(i,4) ...
            -c4*Ez4(i,5)+c5*Ez5(i,6); %%%bottom
    end
    for i = 2:nx-1
        Ez(i,ny) = c1*Ez1(i,ny-1)-c2*Ez2(i,ny-2)+c3*Ez3(i,ny-3) ...
            -c4*Ez4(i,ny-4)+c5*Ez5(i,ny-5); %%%top
    end


    switch ABC_order

        case 5
            Ez5 = Ez4;
```

```
            Ez4 = Ez3;
            Ez3 = Ez2;
            Ez2 = Ez1;
            Ez1 = Ez;

        case 4
            Ez4 = Ez3;
            Ez3 = Ez2;
            Ez2 = Ez1;
            Ez1 = Ez;

        case 3
            Ez3 = Ez2;
            Ez2 = Ez1;
            Ez1 = Ez;

        otherwise
            disp('Error: Wrong Value');
    end
end
end
% ************************
% ************************
%
%
```

Create the excitation signal

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

function Ezs = Source()

global beta mxcite
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-mxcite) >=1 && (n-mxcite) <= mxcite)
        Ezs = sin((n-mxcite)*pi/mxcite);
    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-mxcite) >=1 && (n-mxcite) <= mxcite)
        Ezs = 1;
    else
        Ezs = 0;
    end
end
end
%
%
```

Algorithm for Computing E-field

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

function adv_Ez()
% Compute z-component of E-field
global Ez  Hx  Hy
global Ezi  Hxi  Hyi
global mediaEz flag_medium
global Ca Cb
global nx ny n
global mxcite Source_signal

% Free-space E-field computation
if (flag_medium == 1)
    for i = 2 : nx - 1
        for j = 2 : ny - 1
            m  = 1; % Enforce free-space everywhere
            if (i == mxcite)   %% Incident Field Source Excitation
                Es = Source(); % Es is a soft source
                Source_signal(n) = Es;
            else
                Es = 0;
            end
            Ezi(i,j) = Ezi(i,j)*Ca(m) + Cb(m)*(Hyi(i,j) -
 Hyi(i-1,j)...
                - (Hxi(i,j) - Hxi(i,j-1))) + Es;

        end
    end
    % Space with box computation
elseif (flag_medium == 2)
    for i = 2 : nx - 1
        for j = 2 : ny - 1
            m  = mediaEz(i,j);
            if (i == mxcite)   %% Incident Field Source Excitation
                Es = Source(); % Es is a soft source
            else
                Es = 0;
            end
            Ez(i,j) = Ez(i,j)*Ca(m) + Cb(m)*(Hy(i,j) - Hy(i-1,j)...
                - (Hx(i,j) - Hx(i,j-1))) + Es;
        end
```

```
    end
end
end


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

Algorithm for Computing H-field

********************** **********************

```
function adv_H()
% Compute z-component of E-field
global Ez  Hx  Hy
global Ezi  Hxi  Hyi
global mediaHx mediaHy flag_medium
global Da Db
global nx ny

% % %     Compute x-component of H-field

% Free-space Hx-field computation
if (flag_medium == 1)
    for i = 1 : nx
        for j = 1 : ny - 1
            m = 1;
            Hxi(i,j) = Hxi(i,j)*Da(m) - Db(m)*(Ezi(i,j+1) - Ezi(i,j));
        end
    end

    % Space with box Hx-field computation
elseif (flag_medium == 2)
    for i = 1 : nx
        for j = 1 : ny - 1
            m = mediaHx(i,j);
            Hx(i,j) = Hx(i,j)*Da(m) - Db(m)*(Ez(i,j+1) - Ez(i,j));
        end
    end
end


% % %     Compute y-component of H-field

% Free-space Hy-field computation
if (flag_medium == 1)
    for i = 1 : nx - 1
        for j = 1 : ny
            m = 1;
            Hyi(i,j) = Hyi(i,j)*Da(m) + Db(m)*(Ezi(i+1,j) - Ezi(i,j));
        end
    end
```

```
    % Space with box Hx-field computation
elseif (flag_medium == 2)
    for i = 1 : nx - 1
        for j = 1 : ny
            m = mediaHy(i,j);
            Hy(i,j) = Hy(i,j)*Da(m) + Db(m)*(Ez(i+1,j) - Ez(i,j));
        end
    end
end
end

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

Algorithm for Computing Fourier Transform E-field

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

function ft_field()

global Ez Hx Hy
global Ezi
global mxcl mxcr mycb myct
global n dt finc
global flag_medium
global ftEz_right ftEz_top ftEz_left ftEz_bottom
global ftHx_top ftHx_bottom
global ftHy_right ftHy_left
global ftEinc


dft_exp = -2 * 1i * pi * finc * n * dt; % Exponential in the DFT

% Incident Field Transform
if (flag_medium == 1)

    X = mxcr; Y = myct; % Recording position for incident field DFT
    ftEinc = ftEinc +  dt * Ezi(X,Y) * exp(dft_exp);


elseif (flag_medium == 2)
```

Right side terms ---------------- X = d

```
    X = mxcr; Y = mycb; % Starting position
    for i = 1 : (myct - mycb)

        ftEz_right(i) = ftEz_right(i) + dt * Ez(X,Y+i-1) *
 exp(dft_exp); % Ez DFT at the right
        ftHy_right(i) = ftHy_right(i) + dt * (Hy(X,Y+i-1) + Hy(X-1,Y
+i-1)) * exp(dft_exp)/2; % Hy DFT at the right
```

```
        end
```

Top side terms ------------------ Y = b

```
        X = mxcr; Y = myct; % Starting position
        for i = 1 : (mxcr - mxcl)

            ftEz_top(i) = ftEz_top(i) + dt * Ez(X-i+1,Y) * exp(dft_exp); %
     Ez DFT at the top
            ftHx_top(i) = ftHx_top(i) + dt * (Hx(X-i+1,Y) + Hx(X-i+1,Y-1))
     * exp(dft_exp)/2; % Hx DFT at the top

        end
```

Left side terms ----------------- X = -d

```
        X = mxcl; Y = myct; % Starting position
        for i = 1 : (myct - mycb)

            ftEz_left(i) = ftEz_left(i) + dt * Ez(X,Y-i+1) *
     exp(dft_exp); % Ez DFT at the left
            ftHy_left(i) = ftHy_left(i) + dt * (Hy(X,Y-i+1) + Hy(X-1,Y-i
    +1)) * exp(dft_exp)/2; % Hy DFT at the left

        end
```

Bottom side terms --------------- Y = -b

```
        X = mxcl; Y = mycb; % Starting position
        for i = 1 : (mxcr - mxcl)

            ftEz_bottom(i) = ftEz_bottom(i) + dt * Ez(X+i-1,Y) *
     exp(dft_exp); % Ez DFT at the bottom
            ftHx_bottom(i) = ftHx_bottom(i) + dt * (Hx(X+i-1,Y) + Hx(X
    +i-1,Y-1)) * exp(dft_exp)/2; % Hx DFT at the bottom

        end
    end
    end

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

Summing Fields to calculate far-field

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

    function far_field()

    global ftEz_right ftEz_top ftEz_left ftEz_bottom
    global ftHx_top ftHx_bottom
```

```
global ftHy_right ftHy_left
global eta_0 k flag_medium
global d b phi nda
global mxcl mxcr mycb myct
global Ez_norm

Ez_right = zeros(1, length(0 : pi/180 : 2*pi));
Ez_top = zeros(1, length(0 : pi/180 : 2*pi));
Ez_left = zeros(1, length(0 : pi/180 : 2*pi));
Ez_bottom = zeros(1, length(0 : pi/180 : 2*pi));
Ez_norm = zeros(1, length(0 : pi/180 : 2*pi));

x = linspace(-d,d,length(1:(mxcr-mxcl))); % Define x used in the
 exponential
y = linspace(-b,b,length(1:(myct-mycb))); % Define x used in the
 exponential
x_flip = flip(x); % Reverse x for use on the top side
y_flip = flip(y); % Reverse y for use on the left side
di = x(2) - x(1); % delta_i interval
dj = y(2) - y(1); % delta_j interval

% Total Field Sum
```

Implementing Eq(24) from class notes

```
if (flag_medium == 2)
    for phi_it = 1 : nda

        for i = 1 : (myct - mycb)
            if i == 1 || i == (myct - mycb) % treating corners by
 halving the delta intervals
                Ez_right(phi_it) = Ez_right(phi_it) +
 (eta_0*ftHy_right(i) - ftEz_right(i)*cos(phi(phi_it)))*...
                    exp(1i*k*( d*cos(phi(phi_it)) +
 y(i)*sin(phi(phi_it))))*dj/2;
                Ez_top(phi_it) = Ez_top(phi_it) + (-eta_0*ftHx_top(i)
 - ftEz_top(i)*sin(phi(phi_it)))...
                    *exp(1i*k*( x_flip(i)*cos(phi(phi_it)) +
 b*sin(phi(phi_it))))*di/2;
                Ez_left(phi_it) = Ez_left(phi_it) + (-
eta_0*ftHy_left(i) + ftEz_left(i)*cos(phi(phi_it)))...
                    *exp(1i*k*( -d*cos(phi(phi_it)) +
 y_flip(i)*sin(phi(phi_it))))*dj/2;
                Ez_bottom(phi_it) = Ez_bottom(phi_it) +
 (eta_0*ftHx_bottom(i) + ftEz_bottom(i)*sin(phi(phi_it)))...
                    *exp(1i*k*( x(i)*cos(phi(phi_it)) -
 b*sin(phi(phi_it))))*di/2;
            else

                Ez_right(phi_it) = Ez_right(phi_it) +
 (eta_0*ftHy_right(i) - ftEz_right(i)*cos(phi(phi_it)))*...
                    exp(1i*k*( d*cos(phi(phi_it)) +
 y(i)*sin(phi(phi_it))))*dj;
```

```
                Ez_top(phi_it) = Ez_top(phi_it) + (-eta_0*ftHx_top(i)
 - ftEz_top(i)*sin(phi(phi_it)))...
                    *exp(1i*k*( x_flip(i)*cos(phi(phi_it)) +
 b*sin(phi(phi_it))))*di;
                Ez_left(phi_it) = Ez_left(phi_it) + (-
eta_0*ftHy_left(i) + ftEz_left(i)*cos(phi(phi_it)))...
                    *exp(1i*k*( -d*cos(phi(phi_it)) +
 y_flip(i)*sin(phi(phi_it))))*dj;
                Ez_bottom(phi_it) = Ez_bottom(phi_it) +
 (eta_0*ftHx_bottom(i) + ftEz_bottom(i)*sin(phi(phi_it)))...
                    *exp(1i*k*( x(i)*cos(phi(phi_it)) -
 b*sin(phi(phi_it))))*di;
            end
        end

        Ez_norm(phi_it) = Ez_right(phi_it) + Ez_top(phi_it) +
 Ez_left(phi_it) + Ez_bottom(phi_it); % Ez_norm is the contour
 integral

    end
end

end
```

Routine to Plot Far Field Plot

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

function plot_far_field()
% This function generates the 2D polar and rectangular plots of the
% scattered far-field

global phi Ez_norm ftEinc lambda_0 finc

%
figure(1) % Polar Plot
%
h1 = polar(phi,sqrt(pi/2)*abs(Ez_norm)/abs(ftEinc)/lambda_0);
h1.Color = 'black';
h1.LineWidth = 1.4;
title(['Radiation Pattern of a PEC Box at f =  ',int2str(finc/1e6), '
 MHz'],'Interpreter','latex')
set(gcf,'Color','white'); % Set background color to white
set (gca,'FontName','times new roman') % Set axes fonts to Times New
 Roman
% cleanfigure();
% matlab2tikz('filename',sprintf('ECEN637_HW9_Polar_plot.tex'));
%
%
figure(2)
%
%
h2 = plot(phi,sqrt(pi/2)*abs(Ez_norm)/abs(ftEinc)/lambda_0);
h2.Color = 'black';
```

```
h2.LineWidth = 1.4;
title(['Cartesian Radiation Pattern of a PEC Box at f =
 ',int2str(finc/1e6), ' MHz'],'Interpreter','latex')
set(gcf,'Color','white'); % Set background color to white
set(gca,'FontName','times new roman') % Set axes fonts to Times New
 Roman
ax = gca;
ax.XTick = [0 pi/2 pi 3*pi/2 2*pi];
ax.XTickLabel ={'0','\pi/2','\pi','3\pi/2','\pi'};
xlabel('\phi (rad)','Interpreter','latex'); % X-axis label
ylabel('\sqrt(\frac{RCS}{\lambda_0}) ','Interpreter','latex'); % y-
axis label
axis([ 0 2*pi 0 2.6]) % Set the x- and y- limits according to thee
 paper
Ez_scattered = sqrt(pi/2)*abs(Ez_norm)/abs(ftEinc)/lambda_0;
% save('radpatfile.mat','phi','Ez_scattered')
% cleanfigure();
% matlab2tikz('filename',sprintf('ECEN637_HW9_Rect_plot.tex'));
end
```

Published with MATLAB® R2015b