
Table of Contents

.....	1
Parameters	2
Main Program	3
Input Admittance	3
Far-fields	3
Currents	3
Plot Solutions	3
End Main Program	3
Initialize	3
Pocklington Equation Solver	4
Numerical Integration Using Gauss_Quadratures	4
Calculate the current	5
Input impedance and admittance	5
Pocklington Equation Solver only for Far-fields	6
Numerical Integration Using Gauss_Quadratures	6
Calculate the current	7
Far-field	7
Pocklington Equation Current Solver	8
Numerical Integration Using Gauss_Quadratures	8
Calculate the current	9
Plot Solutions	9

```
function Abbas_HW_10_Wire_MoM()  
  
% Hasan Tahir Abbas  
% ECEN 637  
% Homework 10: Computation of Input Admittance and Radiation Pattern  
% of a  
% Wire Antenna  
% 11/27/2015  
%  
%  
% Pocklington Integral Equation is used to compute the Admittance and  
% Radiation Pattern as explained in:  
% Gibson, Walton C. "The Method of Moments in Electromagnetics,"  
% Taylor  
% and Francis/CRC,  
% 2008.  
%  
%  
% Gauss_Quadrature based Numerical Integration (Built-in Function)  
% Input Admittance  
% Far-field pattern calculation  
% Utilization of Toeplitz nature of the impedance matrix (Only one row  
% needs to be computed)  
%  
% PARAMETERS AND VARIABLES LIST  
% -----  
% M = Total Wire Segments  
% f = Frequency
```

```

% c = Speed of light
% xmu = Permeability of free space
% eps0 = Permittivity of free space
% omega = Angular Frequency
% K = Propagation Constant
% lambda = Wavelength
% a = Wire Radius;
% L = Wire Length
% E_rad = Radiated Far-field
% I = Current on the wire
% Y = Input Admittance of the wire
% num_theta = number of angle values for polar plot
% num_length = number of lengths of wire
%
% FUNCTIONS LIST
% -----
% Pocklington() = Calculates the Pocklington EFIE solution
% Pocklington_Pattern() = Calculates the Pocklington EFIE Far-field
  pattern
% Pocklington_Current() = Calculates the Current on the wire
% -----
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
global f omega lambda K
global Lengths M a
global num_theta num_length theta
global Yin
global E_rad_4by5 E_rad
global I_wire
% *****
% *****
% Data
% *****
% *****
f = 300e6;
c = 2.99792458e8; % Speed of light
xmu = 4*pi*1e-7; % Permeability of free space
eps0 = 8.854187817e-12; % Permittivity of free space
omega = 2.0*pi*f;
K = omega*sqrt(xmu*eps0);
lambda = c/f;
a = 7.022e-3*lambda;
num_length = 101;
num_theta = 361;
M = 51;

```

```
%#####
```

Main Program

```
*****
```

```
initialize();
```

Input Admittance

```
for i = 1 : num_length
    Y = Pocklington(Lengths(i));
    Yin(i) = Y;
end
```

Far-fields

```
First h = 4/5*lambda
E_rad_4by5 = Pocklington_pattern(4*lambda/5);

% Second h = lambda
E_rad = Pocklington_pattern(lambda);
```

Currents

```
I_wire = Pocklington_Current(.7*lambda, .001588*lambda); % to match
the given figure
```

Plot Solutions

```
plot_solutions();

end
%#####
```

End Main Program

```
*****
```

Initialize

```
*****

function initialize()
% Set all the variables to zero

global lambda
global Lengths M
global num_theta num_length
global E_rad_4by5 E_rad I_wire Yin theta
```

```

% Pocklington EFIE outputs
% *****
E_rad_4by5 = zeros(1,num_theta);
E_rad = zeros(1,num_theta);
I_wire = zeros(1,M);
Yin = zeros(1,num_length);
theta = linspace(0,2.0*pi,num_theta);
Lengths = linspace(.5*lambda/pi,3.5*lambda/pi,num_length);

end
% *****
% *****
%

```

Pocklington Equation Solver

```

*****

function Yin = Pocklington(L)

global eps0
global omega K
global M a

deltaz = L / M;

z = linspace(-0.5*L , 0.5*L - deltaz, M);
Z = zeros(M);

for j = 1 : 1 % Only one iteration is needed as impedance matrix can
    be constructed through its Toeplitz property
    for k = 1 : M

        fun = @(zp) exp(-1i*K*sqrt(( z(j)+.5*deltaz - zp).^2 +
a^2))...
        ./sqrt(( z(j)+.5*deltaz - zp).^2 + a^2); % Make a symbolic
Function of z_prime
        % This calculation is based on the reference cited in the code
        % introduction.
        % Mathematical Modeling of Eq. 4.63
        zp_upper = z(k) + deltaz; % Upper limit of integration
        dz_upper = z(j)+.5*deltaz - zp_upper; % Represents (z(m) -
z(n)) for the upper limit
        zp_lower = z(k); % Lower limit of integration
        dz_lower = z(j)+.5*deltaz - zp_lower; % Represents (z(m) -
z(n)) for the limit limit
    end
end

```

Numerical Integration Using Gauss_Quadratures

```

int_part =
quadgk(fun,zp_lower,zp_upper,'RelTol',1e-8,'AbsTol',1e-12);

```

```

        R_upper = sqrt(dz_upper^2 + a^2); % Represents R for the upper
limit
        R_lower = sqrt(dz_lower^2 + a^2); % Represents R for the lower
limit
        sum_part_upper = (dz_upper)*(1 + 1i*K*R_upper)...
        ./R_upper^3*exp(-1i*K*R_upper); % Exact Evaluation of the
second term (upper limit) in Eq. 4.63 of the reference
        sum_part_lower = (dz_lower)*(1 + 1i*K*R_lower)...
        ./R_lower^3*exp(-1i*K*R_lower); % Exact Evaluation of the
second term (lower limit) in Eq. 4.63 of the reference
        Z(j,k) = K^2*int_part + sum_part_upper - sum_part_lower; %
Pocklington Integral for j not equal to k

        if j == k
            % This is done to avoid very small numbers in the denominator
            % For all the diagonal elements of Z, terms only need to
be calculated once
            R_upper = sqrt((-deltaz/2)^2 + a^2); % Here z(m) = z(n)
            R_lower = sqrt((deltaz/2)^2 + a^2);
            sum_part_upper = (-deltaz/2)*(1 + 1i*K*R_upper)...
            ./R_upper^3*exp(-1i*K*R_upper);
            sum_part_lower = (deltaz/2)*(1 + 1i*K*R_lower)...
            ./R_lower^3*exp(-1i*K*R_lower);
            num = sqrt(1 + 4*a^2/deltaz^2) + 1;
            denom = sqrt(1 + 4*a^2/deltaz^2) - 1;
            self = (log(num/denom) - 1i*K*deltaz); % Approximation of
the integral term as described in reference
            Z(j,k) = K^2*self + sum_part_upper - sum_part_lower; %
Diagonal Elements of the impedance matrix
        end

    end
end

Z = toeplitz(real(Z(1,:))) + 1i*toeplitz(imag(Z(1,:))); % Make a
Toeplitz matrix out of a row vector
V = zeros(M,1); % Initialize Source (RHS)
V(floor(M/2)+1) = -1i*4*pi*omega*eps0*(1.0/deltaz); % Delta Source

```

Calculate the current

```
I = Z\V;
```

Input impedance and admittance

```

Zin = 1.0 / I(floor(M/2)+1);
Yin = 1./Zin;

end
% *****
% *****
%

```

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

Pocklington Equation Solver only for Far-fields

```
*****
```

```
function ERad = Pocklington_pattern(L)

global eps0 xmu
global omega K
global M a num_theta theta

deltaz = L / M;
ERad = zeros(1,num_theta);
z = linspace(-0.5*L , 0.5*L - deltaz, M);
Z = zeros(M);

for j = 1 : 1
    for k = 1 : M

        fun = @(zp) exp(-1i*K*sqrt(( z(j)+.5*deltaz - zp).^2 +
a^2))...
            ./sqrt(( z(j)+.5*deltaz - zp).^2 + a^2); % Make a symbolic
Function of z_prime
        % This calculation is based on the reference cited in the code
        % introduction.
        % Mathematical Modeling of Eq. 4.63
        zp_upper = z(k) + deltaz; % Upper limit of integration
        dz_upper = z(j)+.5*deltaz - zp_upper; % Represents (z(m) -
z(n)) for the upper limit
        zp_lower = z(k); % Lower limit of integration
        dz_lower = z(j)+.5*deltaz - zp_lower; % Represents (z(m) -
z(n)) for the limit limit
```

Numerical Integration Using Gauss_Quadratures

```
int_part =
quadgk(fun,zp_lower,zp_upper,'RelTol',1e-8,'AbsTol',1e-12);
R_upper = sqrt(dz_upper^2 + a^2); % Represents R for the upper
limit
R_lower = sqrt(dz_lower^2 + a^2); % Represents R for the lower
limit
sum_part_upper = (dz_upper)*(1 + 1i*K*R_upper)...
    ./R_upper^3*exp(-1i*K*R_upper); % Exact Evaluation of the
second term (upper limit) in Eq. 4.63 of the reference
sum_part_lower = (dz_lower)*(1 + 1i*K*R_lower)...
    ./R_lower^3*exp(-1i*K*R_lower); % Exact Evaluation of the
second term (lower limit) in Eq. 4.63 of the reference
```

```

        Z(j,k) = K^2*int_part + sum_part_upper - sum_part_lower; %
        Pocklington Integral for j not equal to k

        if j == k
            % This is done to avoid very small numbers in the denominator
            % For all the diagonal elements of Z, terms only need to
            be calculated once
            R_upper = sqrt( (-deltaz/2)^2 + a^2); % Here z(m) = z(n)
            R_lower = sqrt((deltaz/2)^2 + a^2);
            sum_part_upper = (-deltaz/2)*(1 + 1i*K*R_upper)...
                ./R_upper^3*exp(-1i*K*R_upper);
            sum_part_lower = (deltaz/2)*(1 + 1i*K*R_lower)...
                ./R_lower^3*exp(-1i*K*R_lower);
            num = sqrt(1 + 4*a^2/deltaz^2) + 1;
            denom = sqrt(1 + 4*a^2/deltaz^2) - 1;
            self = (log(num/denom) - 1i*K*deltaz); % Approximation of
            the integral term as described in reference
            Z(j,k) = K^2*self + sum_part_upper - sum_part_lower; %
            Diagonal Elements of the impedance matrix
        end

    end
end

Z = toeplitz(real(Z(1,:))) + 1i*toeplitz(imag(Z(1,:))); % Make a
Toeplitz matrix out of a row vector
V = zeros(M-2,1); % Initialize Source (RHS)
V(floor((M-2)/2)+1) = -1i*4*pi*omega*eps0*(1.0/deltaz); % Delta
Source
I = zeros(M,1);

```

Calculate the current

```

I(2:M-1) =Z(2:M-1,2:M-1)\V;

```

Far-field

```

for i = 1 : num_theta
    cosTheta = cos(theta(i));
    sinTheta = sin(theta(i));
    ERad(i) = 0;
    for m = 1:M
        z_m = z(m) + 0.5*deltaz;
        ERad(i) = ERad(i) +
        deltaz*I(m)*sinTheta*exp(1i*K*z_m*cosTheta); % Summation
        representation of the integral
    end
    ERad(i) = -(1i*omega*xmu/(4.0*pi))*ERad(i);
end

end

% *****
% *****

```

%

Pocklington Equation Current Solver

```
function I = Pocklington_Current(L,a)

global eps0
global omega K
global M

deltaz = L / M;

z = linspace(-0.5*L , 0.5*L - deltaz, M);
Z = zeros(M);

for j = 1 : 1 % Only one iteration is needed as impedance matrix can
    be constructed through its Toeplitz property
    for k = 1 : M

        fun = @(zp) exp(-1i*K*sqrt(( z(j)+.5*deltaz - zp).^2 +
a^2))...
            ./sqrt(( z(j)+.5*deltaz - zp).^2 + a^2); % Make a symbolic
Function of z_prime
        % This calculation is based on the reference cited in the code
        % introduction.
        % Mathematical Modeling of Eq. 4.63
        zp_upper = z(k) + deltaz; % Upper limit of integration
        dz_upper = z(j)+.5*deltaz - zp_upper; % Represents (z(m) -
z(n)) for the upper limit
        zp_lower = z(k); % Lower limit of integration
        dz_lower = z(j)+.5*deltaz - zp_lower; % Represents (z(m) -
z(n)) for the limit limit
```

Numerical Integration Using Gauss_Quadratures

```
int_part =
quadgk(fun,zp_lower,zp_upper,'RelTol',1e-8,'AbsTol',1e-12);

R_upper = sqrt(dz_upper^2 + a^2); % Represents R for the upper
limit
R_lower = sqrt(dz_lower^2 + a^2); % Represents R for the lower
limit
sum_part_upper = (dz_upper)*(1 + 1i*K*R_upper)...
    ./R_upper^3*exp(-1i*K*R_upper); % Exact Evaluation of the
second term (upper limit) in Eq. 4.63 of the reference
sum_part_lower = (dz_lower)*(1 + 1i*K*R_lower)...
    ./R_lower^3*exp(-1i*K*R_lower); % Exact Evaluation of the
second term (lower limit) in Eq. 4.63 of the reference
```

```

        Z(j,k) = K^2*int_part + sum_part_upper - sum_part_lower; %
        Pocklington Integral for j not equal to k

        if j == k
            % This is done to avoid very small numbers in the denominator
            % For all the diagonal elements of Z, terms only need to
            be calculated once
            R_upper = sqrt((-deltaz/2)^2 + a^2); % Here z(m) = z(n)
            R_lower = sqrt((deltaz/2)^2 + a^2);
            sum_part_upper = (-deltaz/2)*(1 + 1i*K*R_upper)...
                ./R_upper^3*exp(-1i*K*R_upper);
            sum_part_lower = (deltaz/2)*(1 + 1i*K*R_lower)...
                ./R_lower^3*exp(-1i*K*R_lower);
            num = sqrt(1 + 4*a^2/deltaz^2) + 1;
            denom = sqrt(1 + 4*a^2/deltaz^2) - 1;
            self = (log(num/denom) - 1i*K*deltaz); % Approximation of
            the integral term as described in reference
            Z(j,k) = K^2*self + sum_part_upper - sum_part_lower; %
            Diagonal Elements of the impedance matrix
        end

    end
end

Z = toeplitz(real(Z(1,:))) + 1i*toeplitz(imag(Z(1,:))); % Make a
Toeplitz matrix out of a row vector
V = zeros(M-2,1); % Initialize Source (RHS)
V(floor((M-2)/2)+1) = -1i*4*pi*omega*eps0*(1.0/deltaz); % Delta
Source
I = zeros(M,1);

```

Calculate the current

```

I(2:M-1) = Z(2:M-1,2:M-1)\V;

end
% *****
% *****
%

```

Plot Solutions

```

*****

function plot_solutions()

global f lambda
global Lengths M
global theta
global Yin
global E_rad_4by5 E_rad
global I_wire

```

```

% Plot Input Admittance
figure(1);
plot(pi*Lengths/lambda, real(Yin)*1e3);
hold on;
H = plot(pi*Lengths/lambda, real(Yin)*1e3, pi*Lengths/lambda,
    imag(Yin)*1e3);
xlim([1 3.5])
H(1).Color = 'black';
H(1).LineWidth = 1.4;
H(2).Color = 'black';
H(2).LineWidth = 1.4;
H(2).LineStyle = '--';
title(['Input Admittance versus length at f = ',int2str(f/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 = [0.5 1 2 3];
xlabel('$\beta L/2$', 'Interpreter','latex'); % X-axis label
ylabel('G and B in mmhos ', 'Interpreter','latex'); % y-axis label
grid on
cleanfigure();
matlab2tikz('filename',sprintf('ECEN637_HW10_Admittance_plot.tex'));

% First Polar Plot
figure(2)
h1 = polar(theta, abs(E_rad_4by5));
h1.Color = 'black';
h1.LineWidth = 1.4;
title(['Radiation Pattern of a Wire of length $4\lambda/5$ at f
    = ',int2str(f/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_HW10_Polar_plot_h_4by5lambda.tex'));

% First Polar Plot
figure(3)
h3 = polar(theta, abs(E_rad));
h3.Color = 'black';
h3.LineWidth = 1.4;
title(['Radiation Pattern of a Wire of length $\lambda$ at f =
    ',int2str(f/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_HW10_Polar_plot_h_lambda.tex'));

% Current Plot
figure(4)

```

```

x = linspace(-.7*lambda/2, .7*lambda/2, M) ;
H = plot(x, real(I_wire),x, imag(I_wire));
ax = gca;
H(1).Color = 'black';
H(1).LineWidth = 1.4;
H(2).Color = 'black';
H(2).LineWidth = 1.4;
H(2).LineStyle = '--';
title(['Current on the wire of half-length $ h = .35\lambda$ at f = ',int2str(f/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.XTick = [-.3498 -0.2625 -0.1750 -0.0875 0 0.0875 0.1750 0.2625 0.3498];
ax.XTickLabel = { '-h', '-.75h', '-.5h', '-.25h' , '0' , '.25h', '5h', '.75h', 'h'};
ax.YTick = [-2e-3 -1e-3 0 1e-3 2e-3];
ax.YTickLabel = { '-.002', '-.001', '0' , '.001', '.002'};
axis([ -.3498 .3498 -2.5e-3 2.5e-3]);
hold on
xlabel('z')
ylabel('A')
grid on
cleanfigure();
matlab2tikz('filename',sprintf('ECEN637_HW10_Current_on_wire_plot.tex'));
end

```

Published with MATLAB® R2015b