“In the Name of Who Remains”

“The most complete gift of God is a life based on knowledge” <***Imam Ali***>

Numerical Methods in Electromagnetics

<Computational Electro Magnetics>

Associated Professor:

Prof. **Reza Faraji Dana**

HW7 – **MohammadReza Arani** – 810100511

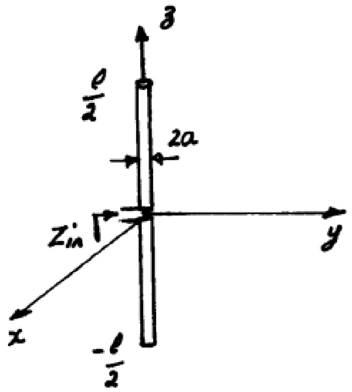
University of Tehran

2/01/2023

Table of Contents

Table of Contents

Q1:



۱- آنتن استوانه‌ای به طول l و شعاع a از وسط تغذیه می‌کنیم.

الف) با استفاده از روش ممان توزیع جریان روی آنتن را به ازای مقادیر مختلف $l = \frac{\lambda}{4}$ ، $\frac{\lambda}{2}$ ، $\frac{3\lambda}{4}$ و λ بدست آورید و امپدانس ورودی آنتن را محاسبه نمائید. a/λ را برابر 0.001 درنظر بگیرید.

ب) پترن آنتن در صفحات $\varphi=0$ (θ متغیر) و $\theta=\frac{\pi}{2}$ (φ متغیر) را در هر حالت محاسبه و رسم نمائید

ج) یافته‌های خود را با آنچه از تحلیل کلاسیک آنتن‌های دی‌پل انتظار دارید مقایسه کنید و در جنبه‌های مختلف روش ممان که در عمل به آن برمی‌خورید بحث نمائید.

Part -1)

We are using Method of Moments to achieve the simulated answer of this solution:

- Expansion functions (Basis functions) are chosen to be Triangular pulses.

$$f = \sum_{n=1}^{N} \alpha_n f_n$$

- Weighting functions are chosen from the **Galerkin** method, the same as our basis functions.

$$W_m = f_m;\ f_{m=T_m}$$

The Main point in this question, is to find the current over the wire! -- >> The geometry of this problem is defined as a simple dipole on a thined wire with diameter equal to 2a

Thined wire is defined as a wire with diameter which is small in comparison to wave length!

The MoM is applied to and results in Current over the wire! -- >> This Current is then used to find the Scattered Field and then the Total Field!

Incident Wave can be modeled as both:

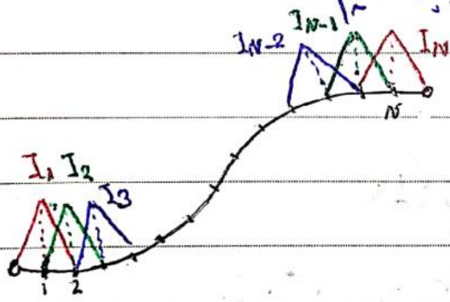
1) Gap Generator
2) A plane wave propagating in the area

is defined as:

$$E^i = \mathrm{jw}\mu_0 \hat{\ell} . \int I(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' - \frac{1}{\mathrm{jw}\epsilon_0} \hat{\ell} . \nabla \int \frac{\partial}{\partial \ell'} I(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' ;$$

where E^i equals to the rigth side of the $L\{f\} = g$ equation!

To apply Method of moments, first we must write our f in terms of corresponding expansion functions!



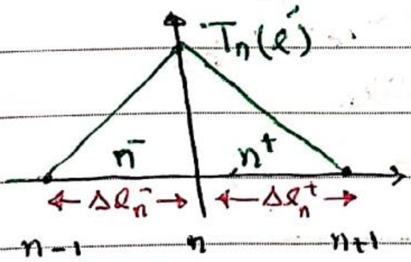
سعی می‌کنیم با روش ممان و توابع پایه مثلثی به تحلیل مسئله بپردازیم:
اگر N+1 قطعه روی سیم در نظر بگیریم؛ تقریبی تکه‌ای -
خطی از توزیع جریان به ما خواهد داد و می‌دانیم در ابتدا و انتها
با جریان صفر انتخاب بسیار خوبی است:

$$I(\ell') = \sum_{n=1}^{N} I_n T_n(\ell')$$

Figure 1

After choosing Expansion Function, the Second Step is considered Done!

$$I(\ell') = \sum_{n=1} I_n T_n(\ell')$$



گام اول: جایگذاری عبارت بالا در معادلات به دست آمده:

$$E_\ell^i(\ell) = \sum_{n=1}^{N} I_n \left\{ j\omega\mu_0 \, \hat{\ell} \cdot \underset{\hat{\ell}'}{\hat{\ell}_n} \int_{n-1}^{n+1} T_n(\ell') \frac{e^{-jkR}}{4\pi R} d\ell' - \frac{1}{j\omega\epsilon_0} \frac{\partial}{\partial \ell} \int_{n-1}^{n+1} \frac{\partial T_n(\ell')}{\partial \ell'} \cdot \frac{e^{-jkR}}{4\pi R} d\ell' \right\}$$

$$\Rightarrow E^i = \mathrm{j}w\mu_0 \hat{\ell}' . \int \sum_{n=1} I_n T_n(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' - \frac{1}{\mathrm{j}w\epsilon_0} \hat{\ell} . \nabla \int \frac{\partial}{\partial \ell'} \sum_{n=1} I_n T_n(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' \Rightarrow$$

due to linearity of operators, we have:

$$\Rightarrow E^i = \mathrm{j}w\mu_0 \hat{\ell}' . \sum_n \int I_n T_n(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' - \frac{1}{\mathrm{j}w\epsilon_0} \hat{\ell} . \sum_n \nabla \int \frac{\partial}{\partial \ell'} I_n T_n(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' \Rightarrow \;-->>$$

Now, it is time to choose for the weighting functions at:

$$< W_m, g > \; = \; < W_m \, , \, L\{f\} >$$

=> For Galerkin we have:

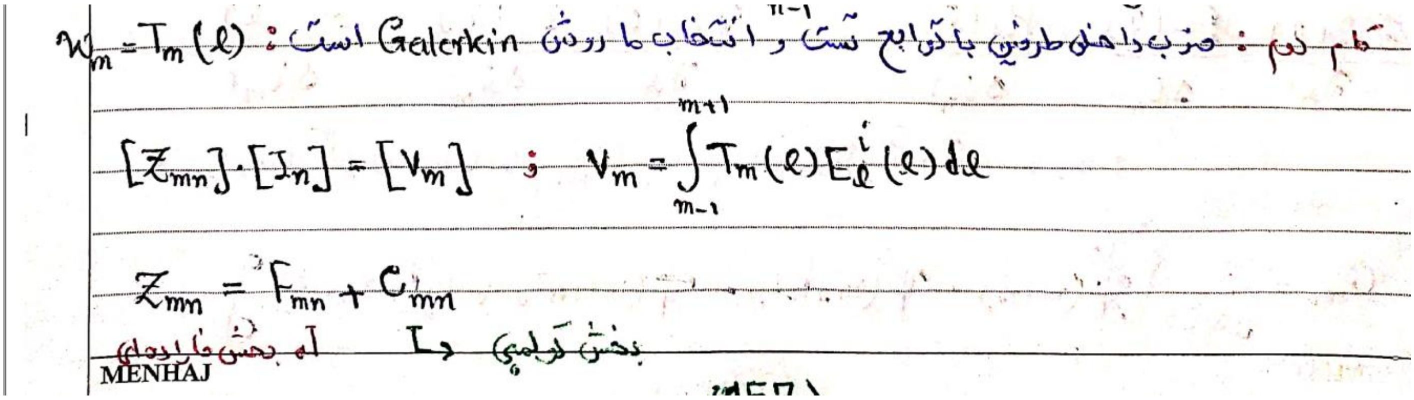


Figure 2

$$< W_m \, , E^i > \; = V_m \; = \mathrm{jw}\mu_0 \hat{\ell}.\sum_n \int W_m \int I_n T_n(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' d\ell - \frac{1}{\mathrm{jw}\epsilon_0} \hat{\ell}.\sum_n \int W_m \nabla \int \frac{\partial}{\partial \ell'} I_n T_n(\ell') \frac{e^{-\mathrm{jkR}}}{4\pi R} d\ell' d\ell$$

Due to the fact that we are facing an antenna problem, it is desired to have our value equal to 0 at both ends of the antenna!

To calculate the distance, one shall use R_{eff} where:

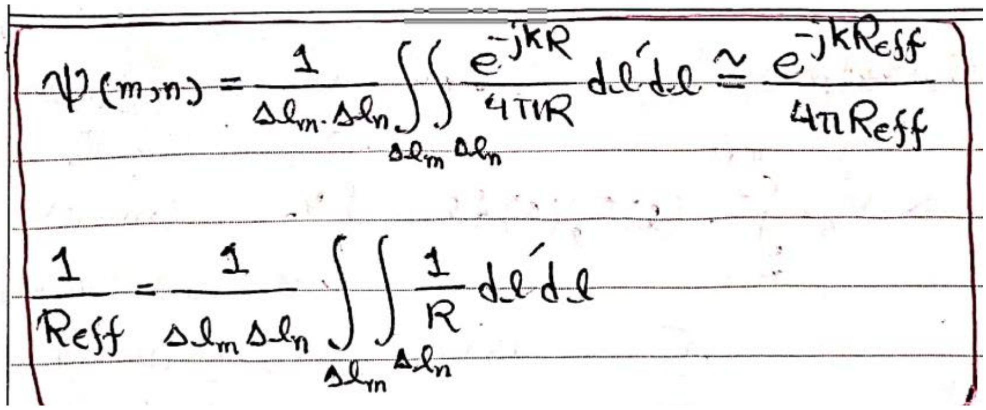


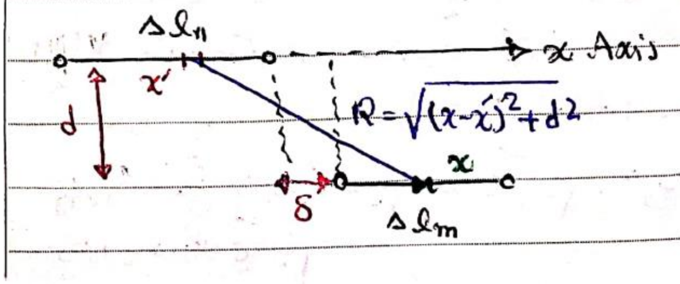
Figure 3

$$\Psi\,(m,n) = \frac{e^{-\mathrm{jkR_{eff}}}}{4\pi R_{\mathrm{eff}}} \text{ where } R_{\mathrm{eff}} = \frac{\mathrm{delta}_{\ell_m} * \mathrm{delta}_{\ell_n}}{M};\,|$$

For a single wire, M equals to:

حالت های مختلف سطحات جریان :

(الف) دو قطعه (طول) موازی



$$\int_{0}^{\Delta\ell_n}\int_{\Delta\ell_n+\delta}^{\Delta\ell_n+\delta+\Delta\ell_m}\frac{dx'\,dx}{\sqrt{(x-x')^2+d^2}}$$

Figure 4

From Fredrick W. Grover, Inductance Calculations-Working Formulas and Tables, we get the closed form formulation for R_{eff}.

Assuming:

$$\int\frac{dx}{\sqrt{x^2+a^2}}=\sinh^{-1}\left(\frac{x}{a}\right)+C \quad ; \quad \int\sinh^{-1}\left(\frac{x}{a}\right)dx=x\sinh^{-1}\left(\frac{x}{a}\right)-\sqrt{x^2+a^2}+C$$

پس تعریف کنیم :

$$\alpha=\Delta\ell_n+\Delta\ell_m+\delta \ ، \ \beta=\Delta\ell_n+\delta \ ، \ \gamma=\Delta\ell_m+\delta$$

Figure 5

$\alpha=\mathrm{delta}_{\ell_n}+\mathrm{delta}_{\ell_m}+\delta$; $\beta=\mathrm{delta}_{\ell_n}+\delta$; $\gamma=\mathrm{delta}_{\ell_m}+\delta$;

$$M=\alpha.\sinh^{-1}\left(\frac{\alpha}{d}\right)-\beta.\sinh^{-1}\left(\frac{\beta}{d}\right)-\gamma.\sinh^{-1}\left(\frac{\gamma}{d}\right)+\delta.\sinh^{-1}\left(\frac{\delta}{d}\right)-\sqrt{\alpha^2+d^2}+\sqrt{\beta^2+d^2}+\sqrt{\gamma^2+d^2}-\sqrt{\delta^2+d^2}$$

For self-terms we get:

۱۳

$$\delta = -\Delta l_n \,,\ \Delta l_m = \Delta l_n \,,\ d = a \quad \text{for Self Term}$$

$$\Rightarrow \beta = \gamma = 0 \quad , \quad \alpha = \Delta l$$

$$\boxed{M = 2\Delta l . \sinh^{-1}\left(\frac{\Delta l}{a}\right) - 2\sqrt{\Delta l^2 + a^2} + 2a \ \text{(for Self Term)}}$$

Figure 6

$$\delta = \delta_{\ell_n} \;, \mathrm{delta}_{\ell_m} = \mathrm{delta}_{\ell_n}; \quad d = a;$$

$$\Rightarrow \beta = \gamma = 0; \ \alpha = \mathrm{delta}_{\ell};$$

$$M = 2\mathrm{delta}_{\ell} . \sinh^{-1}\left(\frac{\mathrm{delta}_{\ell}}{a}\right) - 2\sqrt{\mathrm{delta}_{\ell}^2 + a^2} + 2a$$

Based on these given formulas, we can now implement our functions:

Functions:

M_calc:

```
function  M = M_calc( m , n , delta_l , d )

delta = delta_calc(m,n,delta_l);

alpha = 2*delta_l + delta;
Beta  =  delta_l  + delta;
Gamma =  delta_l  + delta;

if(m==n)% self Term:
         M = 2*delta_l*asinh(delta_l/d) - 2*sqrt(delta_l^2+ d^2) + 2*d;
else
                     M = alpha*asinh(alpha/d) - Beta*asinh(Beta/d)  - Gamma*asinh(Gamma/d) +
                     delta*asinh(delta/d) ...
              - sqrt(alpha^2 +d^2) + sqrt(Beta^2 +d^2) + sqrt(Gamma^2 +d^2) - sqrt(delta^2 +d^2)
;
end

end
```

Z_calc:

```
function Z2 = Z_calc(N,w,mu0,delta_l,eps0,PSAI )


Z2 = zeros(N,N);
```

```
Z2(1,1) = 1j*w*mu0*(delta_l^2) *PSAI(1,1) + 1/(1j*w*eps0) * ( 2*PSAI(1,1)-2*PSAI(1,2) ) ; %
Self Term

m=1;
for n=2:N
        Z2(m,n) = 1j*w*mu0*(delta_l^2)*PSAI(1,abs(m-n)+1) + 1/(1j*w*eps0) * ( 2*PSAI(1,abs(m-
n)+1) - PSAI(1,abs(m-n)+2) - PSAI(1,abs(m-n)) )  ;
end


for m=2:N
    for n=1:N
        % Self Terms:
        if(m==n)
            Z2(m,n) = Z2(1,1);
        else
            Z2(m,n) = Z2(1,abs(m-n)+1) ;
        end


    end
end


end
```

delta_calc:

```
function delta = delta_calc(m,n,delta_l)
    delta = (abs(m-n)-1)*delta_l;
end
```

Dipole_Antenna_exact_Z:

```
function  Z_in = Dipole_Antenna_exact_Z(l,a,Lambda)
    eta = 120*pi;
    k = 2*pi/Lambda;
    C = 0.5772;

    X  = eta/(4*pi) * ( 2*fresnels(k*l) + cos(k*l)*(2*fresnels(k*l) - fresnels(2*k*l) ) ...
                  -sin(k*l)*( 2*fresnelc(k*l)-fresnelc(2*k*l)-fresnelc(2*k*a^2/l)  ) ) ;

    Rr  = eta/(2*pi) * (  ...
         C + log(k*l) - fresnelc(k*l) ...
         + 1/2*sin(k*l)* (fresnels(2*k*l)-2*fresnels(k*l)) ...
         + 1/2*cos(k*l)* (C + log(k*l/2)+ fresnelc(2*k*l) - 2*fresnelc(k*l) ) ...
                      )  ; %  C = 0.5772 (Euler s constant)
    R_in = Rr/(1e-3+sin(k*l/2))^2;
    X_in = X/(sin(k*l/2))^2;

    Z_in = R_in + 1j*X_in;



end
```

W_calc:

```
function [W,W2] = W_calc(N,delta_l_index,l_vec)
```

```
W = zeros(1,length(l_vec));

for i=1 : N+1
    if(mod(i,2)==1)
        W(1 + (i-1)*delta_l_index:i*delta_l_index) = 0.5*linspace(0,1,delta_l_index);
    else
        W(1 + (i-1)*delta_l_index:(i)*delta_l_index) =  -0.5*( linspace(1,2,delta_l_index) )+1;
    end
end
W2 =  circshift([W(1:end-delta_l_index),zeros(1,delta_l_index) ],delta_l_index);
if(mod(N,2)==0)
 W(end-delta_l_index:end) = 0;
else
 W2(end-delta_l_index:end) = 0;
end

end
```

G_m_calc:

```
function V_m  = G_m_calc(W , E_i , m ,l_vec , delta_l_index)

    Axis   = zeros(1,length(l_vec));
    Axis(1:2*delta_l_index) = W(1:2*delta_l_index);

    T_m    = circshift( Axis , (m-1)*delta_l_index  ) ;

    V_m    = sum( T_m.*E_i , 'all') ;

end
```

Pattern_draw:

```
function Object_Antenna= Pattern_draw(Object_Antenna)

L =Object_Antenna.L ;
I = Object_Antenna.I2;
Lambda = Object_Antenna.Lambda;
delta_l =Object_Antenna.delta_l;
k =Object_Antenna.k;

theta = -180: 0.1 :180 ;
zn = linspace(-L/2,L/2,length(I))';
Pattern = sind(theta).*sum( delta_l*I.*exp(1j*k*zn*cosd(theta)) ) ;


Object_Antenna.theta = theta;
Object_Antenna.Pattern_theta = Pattern;

figure()
polarplot(pi*theta/180, abs(Pattern))
% plot( abs(Pattern) , theta )
% hold on
% plot( -abs(Pattern) , theta )
title("Pattern of Antenna for \lambda  = "+Lambda+" and L = "+L/Lambda+"*Lambda")
grid on
```

```
end
```

PSAI_calc:

```
function  PSAI = PSAI_calc(N,delta_l , d,k)
    PSAI =zeros(N+1,N+1);
    M = PSAI;

    for m=1:N+1
        for n=1:N+1
            M(m,n)    = M_calc( m , n , delta_l , d );
            Reff      =  (delta_l*delta_l)/M(m,n);
            PSAI(m,n) = exp(-1j*k*Reff)/(4*pi*Reff);
        end
    end

end
```

Total_Worker:

```
function Total_Object = Total_Worker(N,L,a,f,c,draw)



Lambda = c/f ;
delta_l = L/(N+1);
delta_l_index = 100; % Each Triangle Delta_l is equal to 100 indexes in l_vec
l_vec = linspace(0,L,(N+1)*delta_l_index);

Total_Object = struct();
Total_Object.Lambda        = Lambda;
Total_Object.delta_l       = delta_l;
Total_Object.delta_l_index = delta_l_index ;
Total_Object.l_vec = l_vec;
Total_Object.draw = draw;
Total_Object.N = N;
Total_Object.L = L;
Total_Object.f = f;


[W,W2] = W_calc(N,delta_l_index,l_vec);

if(draw==1)
    figure()
    plot(l_vec,W);

    grid on
    hold on
    plot(l_vec , W2)
    for i=1:N+1
        plot( i*delta_l*ones(1,10) , linspace(0,max(W),10),'r--');
    end
    legend("W","W2")
end


Total_Object.W = W;
Total_Object.W2 = W2;
```

```
V1 = zeros(N+1,1);
E_i = zeros(1,(N+1)*delta_l_index)  ;

mid_point = floor(length(E_i)/2) ; % Tahrik az vasat



E_i(mid_point) = 1  ;
for m = 1:N+1
    V1(m)  = G_m_calc(W , E_i , m ,l_vec , delta_l_index);
end

V2 = zeros(N+1,1);
mid_point2 = floor(length(V2)/2);
V2(mid_point2) = 1;

Total_Object.mid_point = mid_point;
Total_Object.E_i = E_i;
Total_Object.V1 = V1;
Total_Object.V2 = V2;


M = zeros(N+1,N+1);
Z1 = M;
PSAI1 = M;
% PSAI_f = PSAI;

d = a;
k = 2*pi/Lambda;  % wave number

w = 2*pi*f; % Rad/m

mu0  = 4*pi*1e-07; % H/m
eps0 = 8.85*1e-12; % F/m

Total_Object.eps0 = eps0;
Total_Object.mu0 = mu0;
Total_Object.w = w;
Total_Object.d = a;
Total_Object.k = k;



for m = 1:N+1
    for n=1:N+1
        M(m,n)    = M_calc( m , n , delta_l , d );
        Reff      =  (delta_l*delta_l)/M(m,n);
        PSAI1(m,n) = exp(-1j*k*Reff)/(4*pi*Reff);
        % PSAI_f(m,n) =   ;

        if( (m==N+1) || (n==N+1)  )
            Z1(m,n) = 1j*w*mu0*delta_l*delta_l*PSAI1(m,n) + ...
                (1/(1j*w*eps0)*( 0+PSAI1(m,n)- 0 - 0 ) ) ;
        else
            Z1(m,n)    = 1j*w*mu0*delta_l*delta_l*PSAI1(m,n) +...
                (1/(1j*w*eps0)*( PSAI1(m+1,n+1)+PSAI1(m,n)- PSAI1(m+1,n) - PSAI1(m,n+1) ) ) ;
        end

    end
end
```

```
PSAI2 = PSAI_calc(N,delta_l , d,k);
Z2 = Z_calc(N,w,mu0,delta_l,eps0,PSAI2 );

% First Check whether the Z2 is illconditioned:
dZ2 = decomposition(Z2);
is_ILL_Cond = isIllConditioned(dZ2);

if(is_ILL_Cond)
    disp("Z2 is ill Conditioned!!!")
end

I2 = inv(Z2)*V2(1:end-1);


Total_Object.PSAI2 = PSAI2;
Total_Object.Z2 = Z2;



Total_Object.M =M;
Total_Object.PSAI1 = PSAI1;
Total_Object.Z1 = Z1;


I1 = inv(Z1)*V1;

Z_in1 = V1(floor(mid_point/delta_l_index))/I1(floor(mid_point/delta_l_index));

Z_in2 = V2((mid_point2))/I2((mid_point2));

% disp("Impedance for Antenna with L = "+L/Lambda+"*Lambda: ")
% disp((Z_in))

Total_Object.Z_in1 = Z_in1;
Total_Object.Z_in2 = Z_in2;
Total_Object.I1 = I1;
Total_Object.I2 = I2;

end
```

This function calculates the M value which is essential in calculation of R_{eff}.

M can be obtained based on the geometry of the Wire

<the problem>.

An example:

For N=8 we get:

Voltage based on Incident wave integration over T_m:

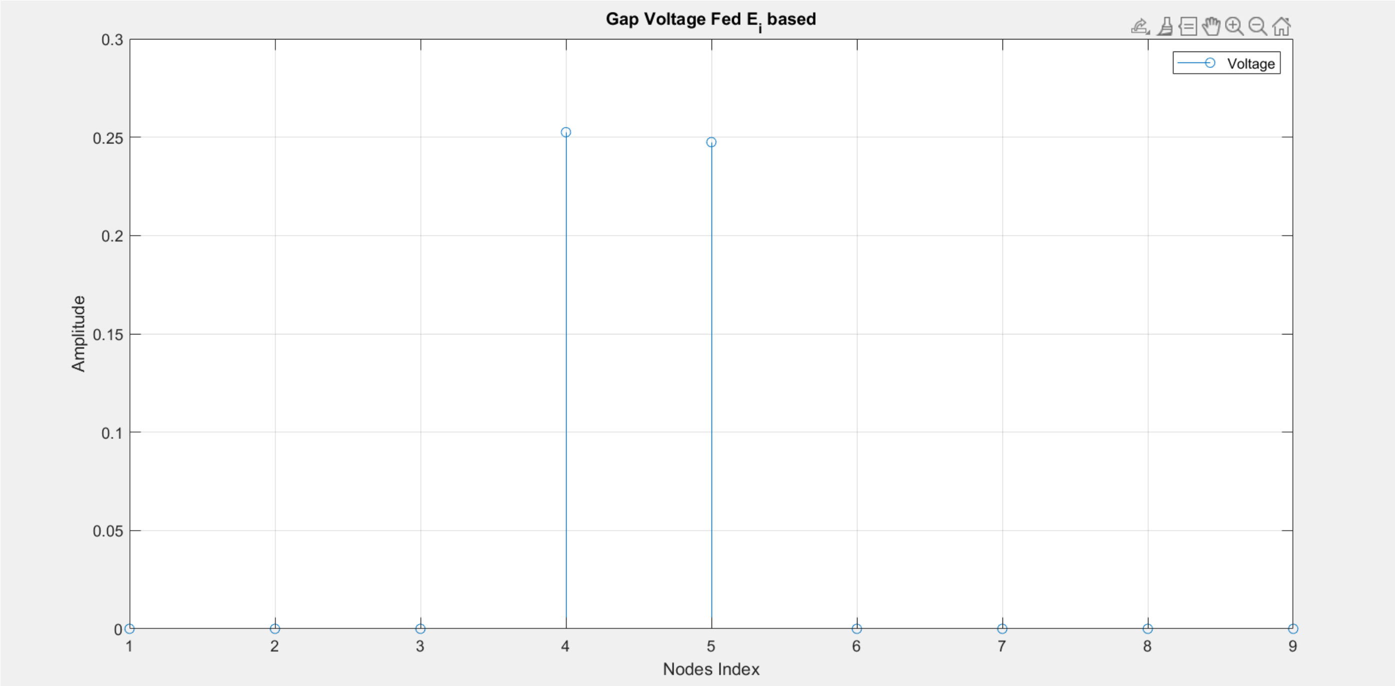


Figure 7

The V itself generated so that we see a delta form in V not in E^i:

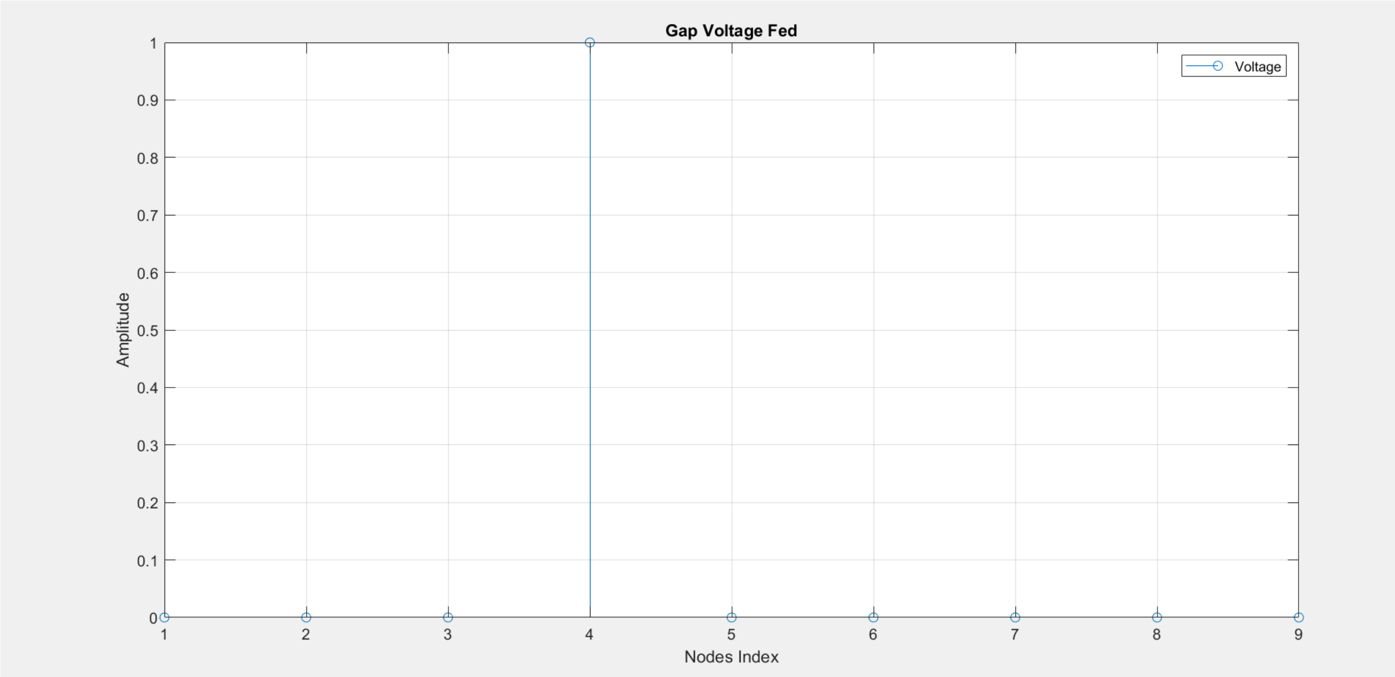


Figure 8

Current obtained based on given delta-form E^i:

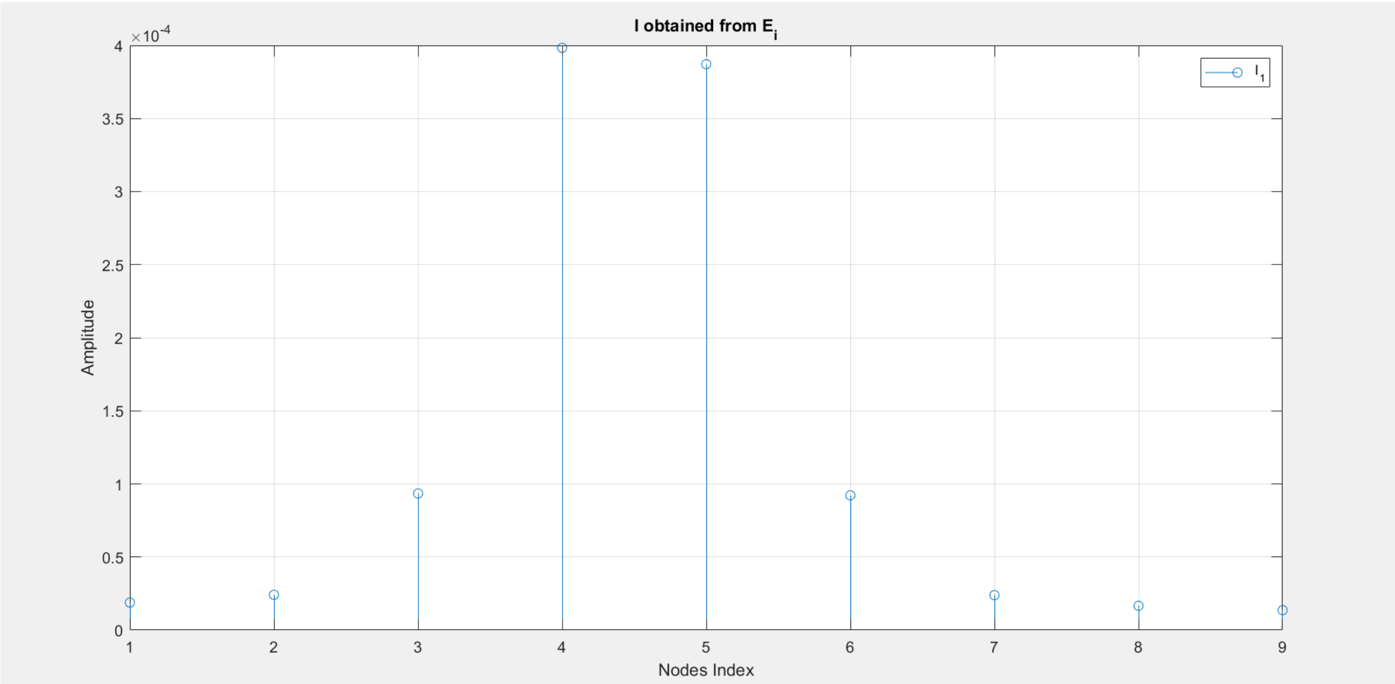


Figure 9

Current obtained based on given delta-form V_{Gap}:

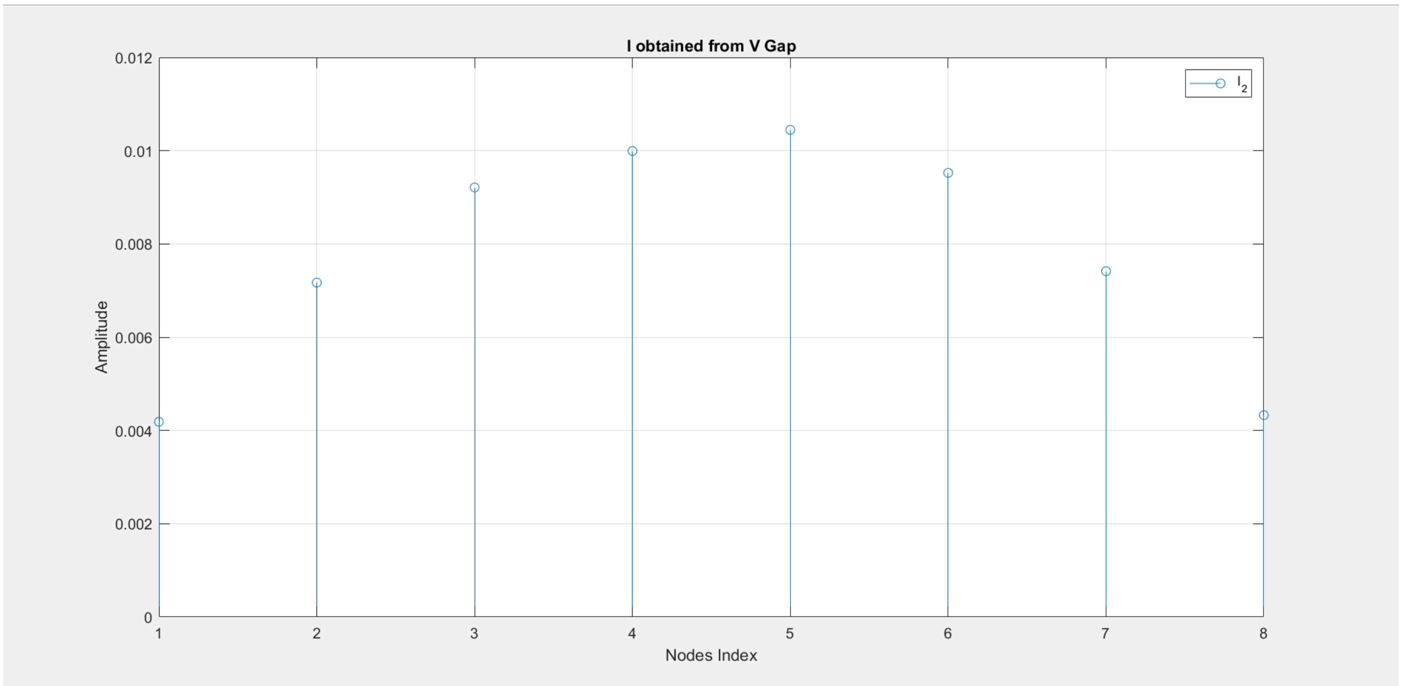


Figure 10

Z Matrix based on given delta-form E^i will look like:

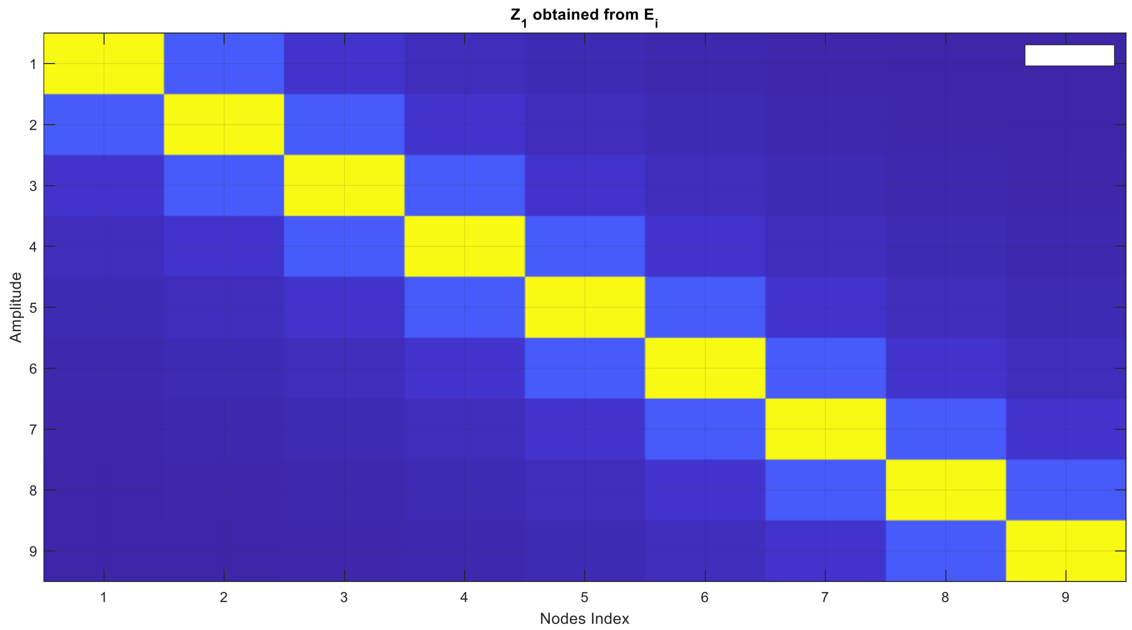


Figure 11

Z matrix, based on V_{gap} is depicted below:

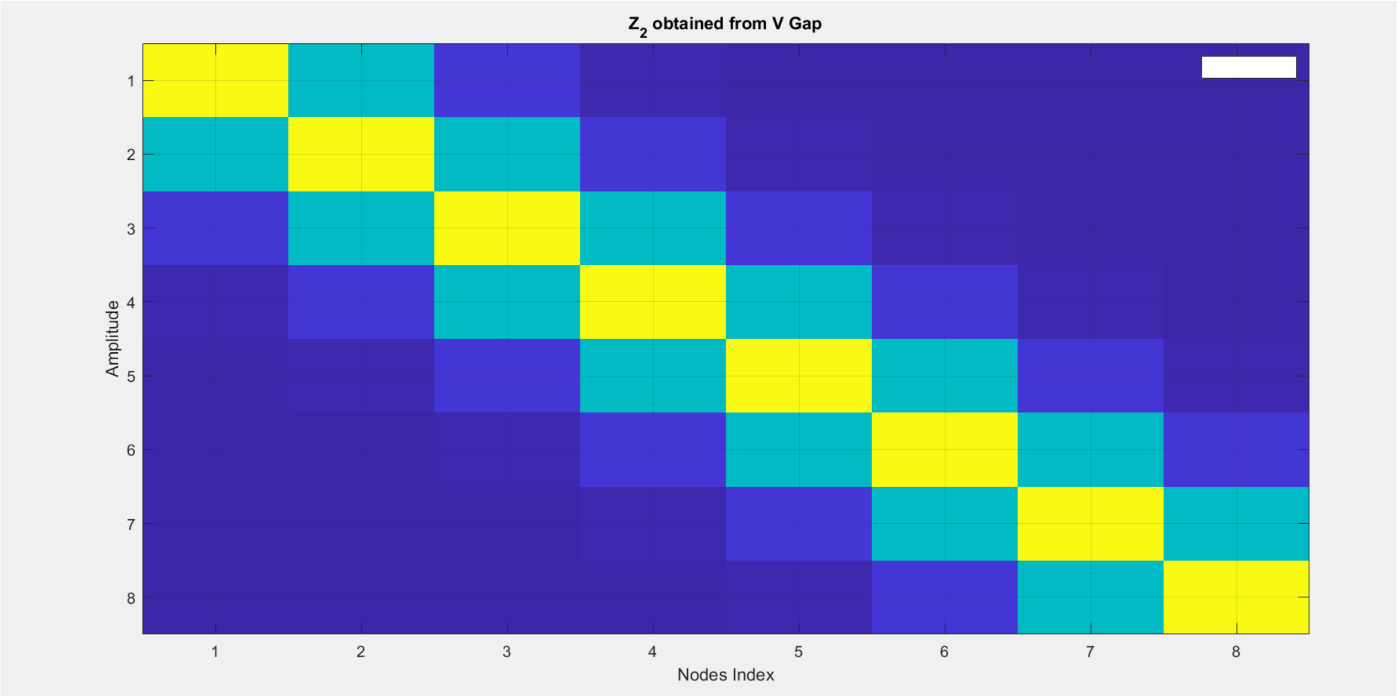


Figure 12

We find the answers to the V_{gap} more accurate and so we choose to use that as an excitation from now on.

The corresponding pattern will be:

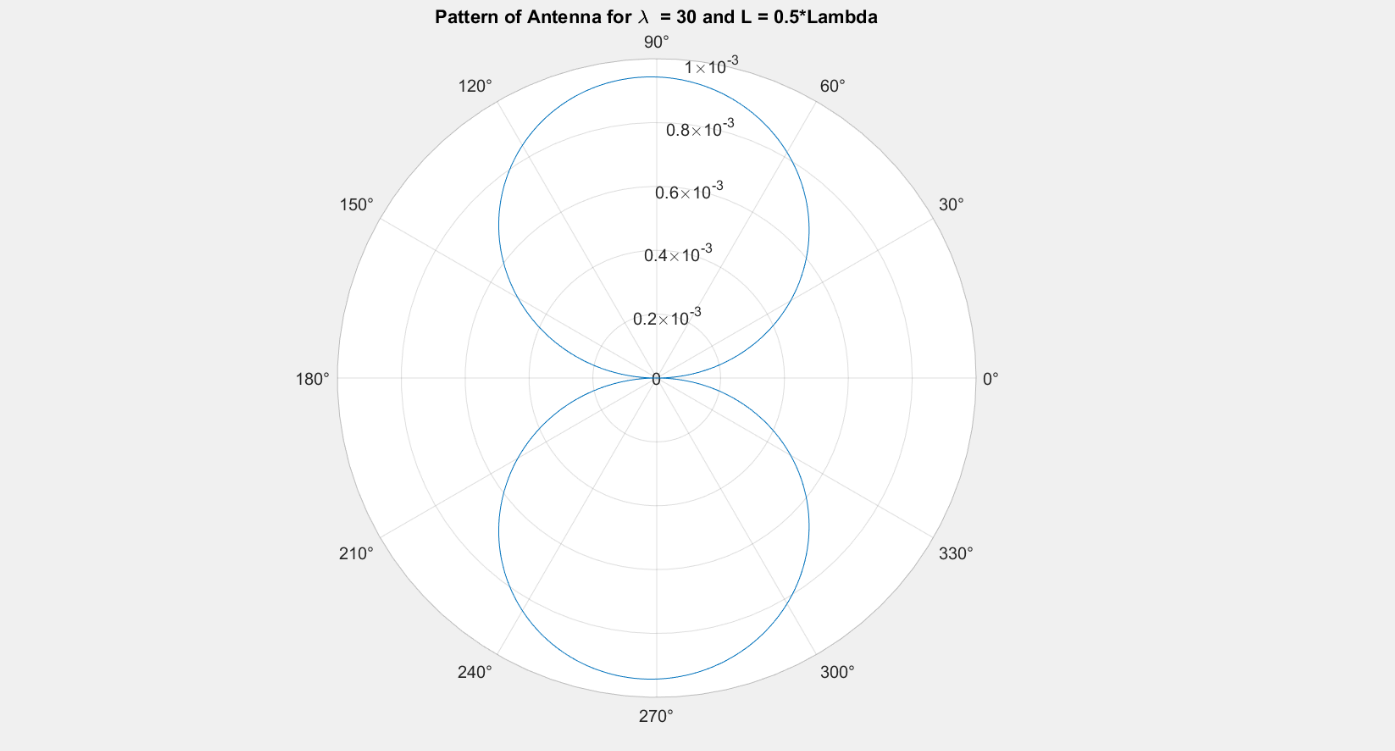

Pattern of Antenna for λ = 30 and L = 0.5*Lambda
90°
120°
60°
150°
30°
180°
0°
210°
330°
240°
300°
270°
1×10^-3
0.8×10^-3
0.6×10^-3
0.4×10^-3
0.2×10^-3
0

Figure 13

Part-2)

Enumeration over Antenna Length

The Pattern and antenna impedance for different antenna lengths is brought below:

- L = Lambda/4

The Z Matrix:

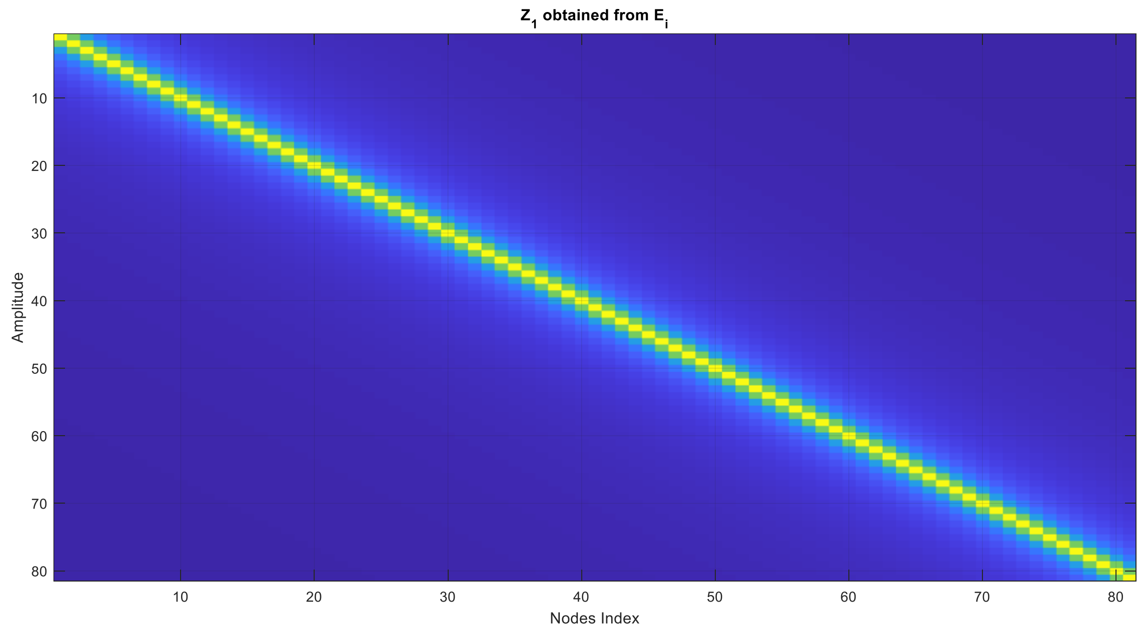


Figure 14

Input Impedance Value:

1.1576e+01 - 4.1315e+02i

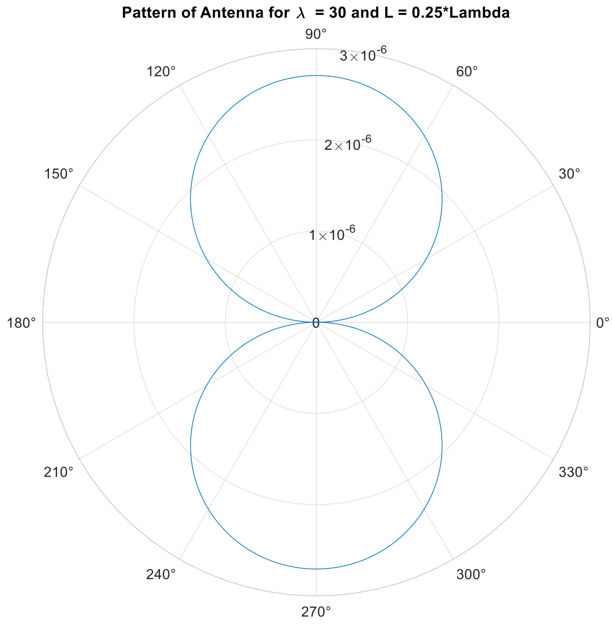


Figure 15

Current Distribution over the Antenna:

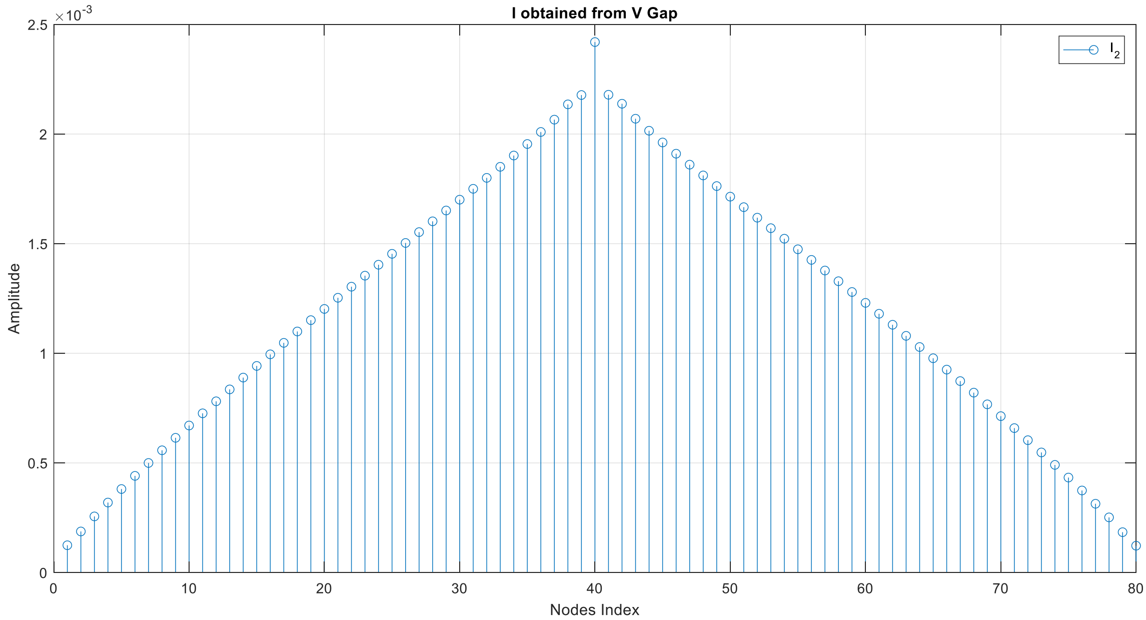


Figure 16

- L = Lambda/2

The Z Matrix:

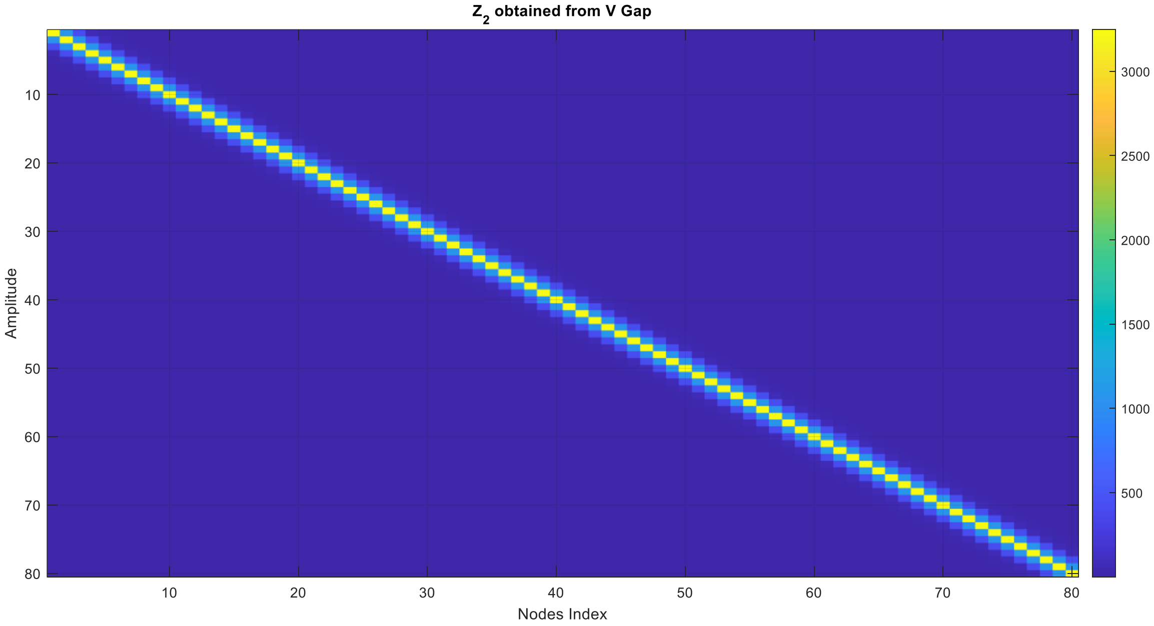


Figure 17

Input Impedance Value:

86.6484 +47.1377i

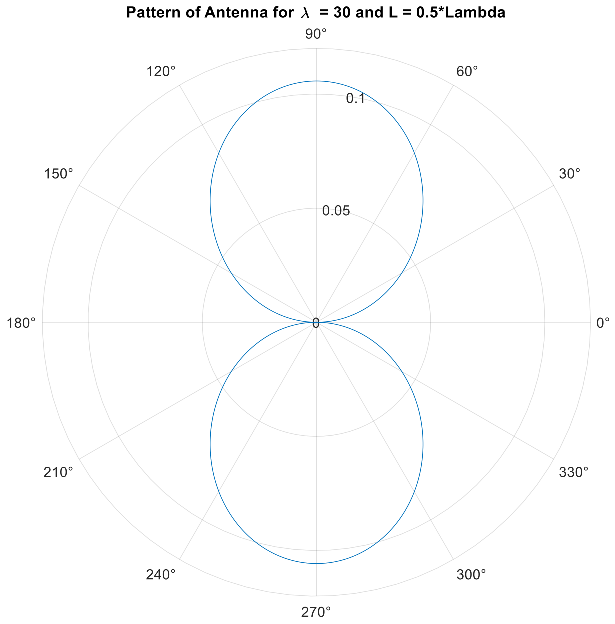


Figure 18

Current Distribution over the Antenna:

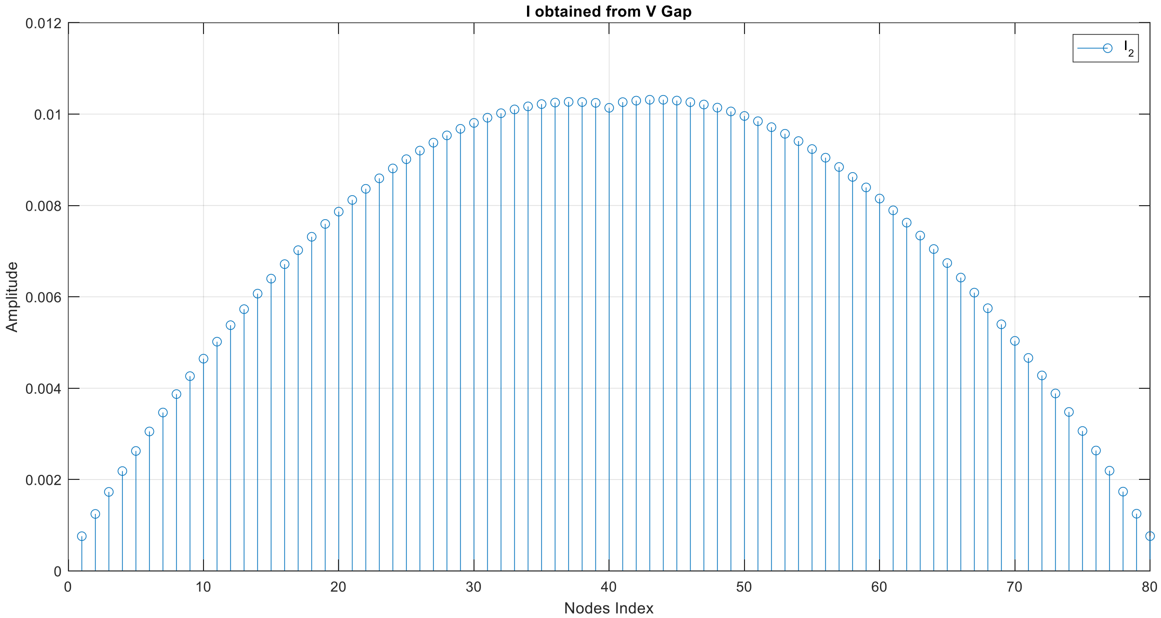


Figure 19

- L = 3*Lambda/4

The Z Matrix:

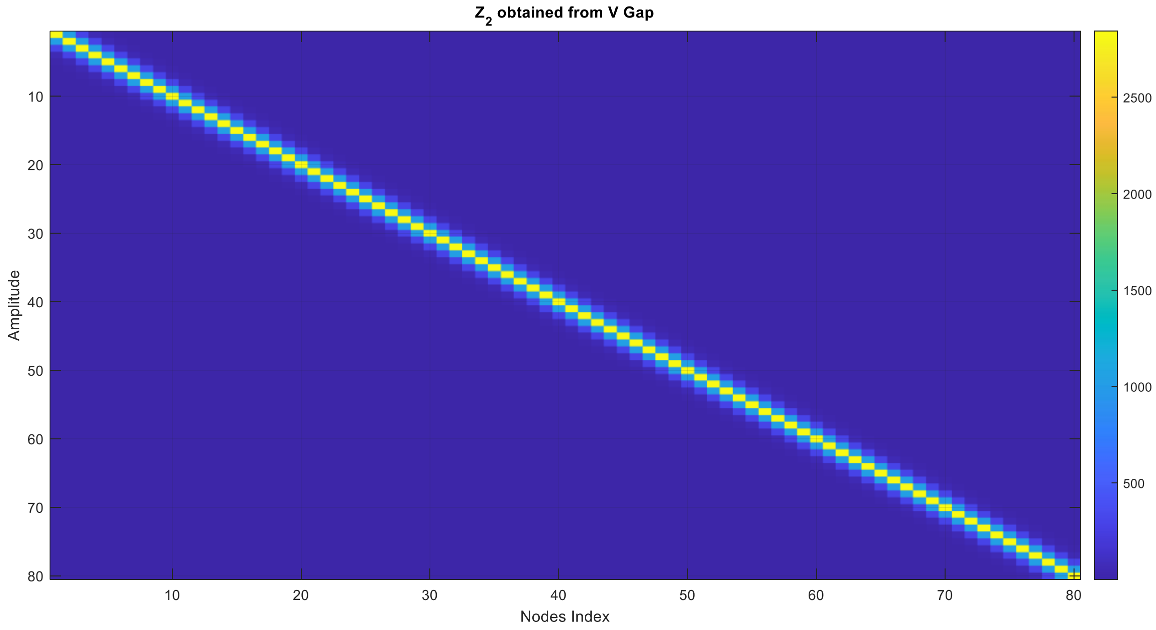


Figure 20

Input Impedance Value:

7.1280e+02 + 6.6702e+02i

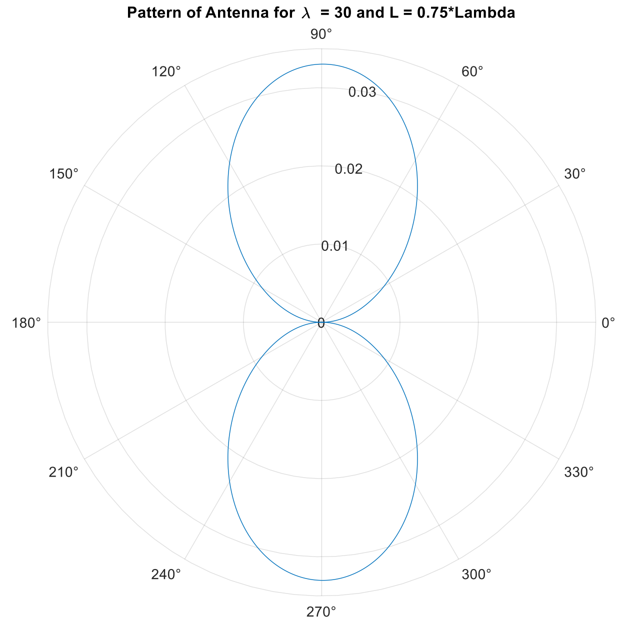


Figure 21

Current Distribution over the Antenna:

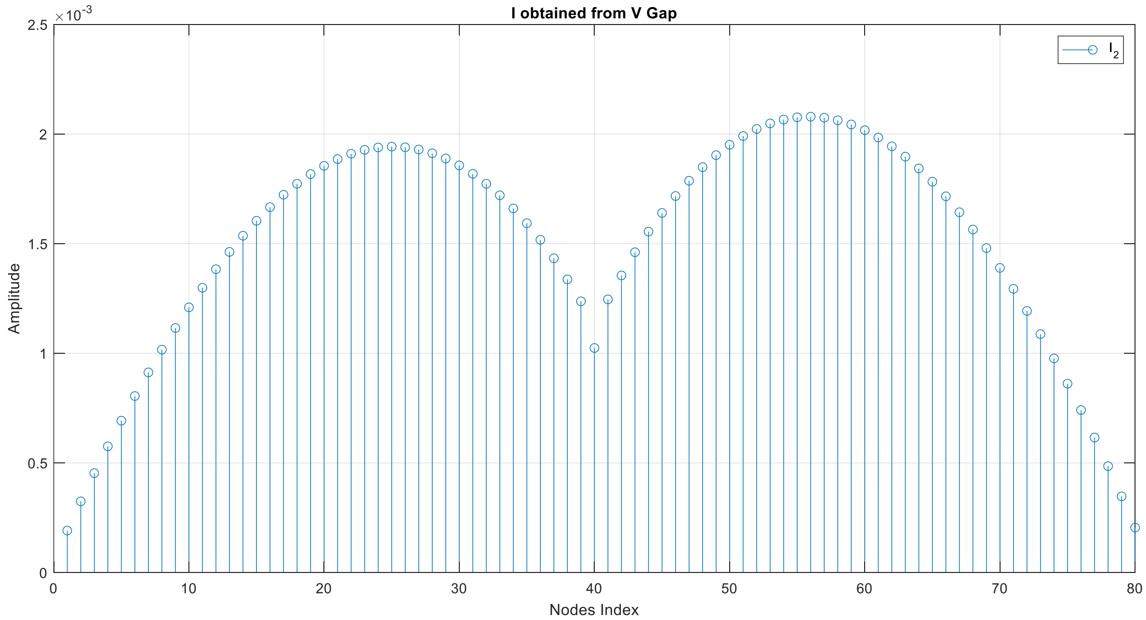


Figure 22

- L = Lambda

The Z Matrix:

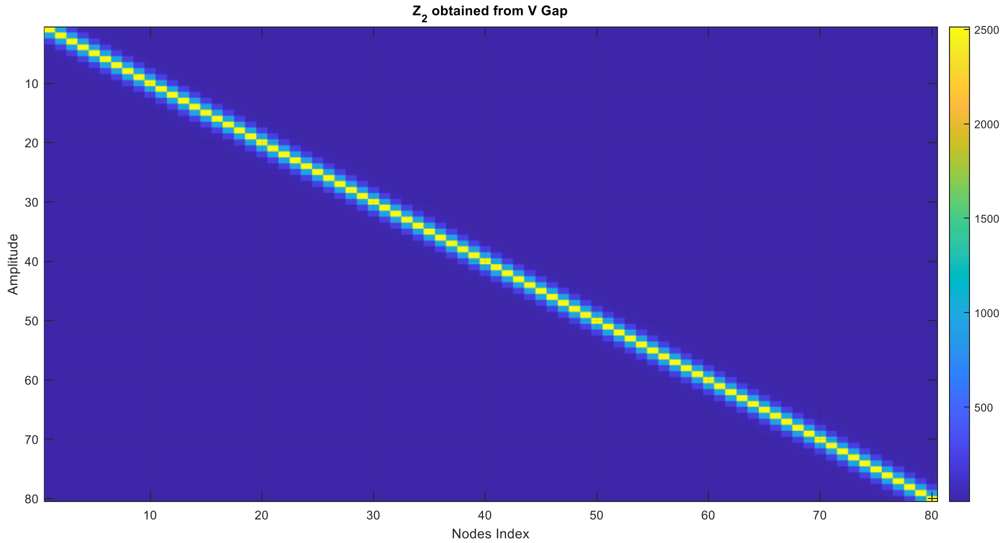


Figure 23

- Input Impedance Value:
  7.6199e+02 - 1.0435e+03i

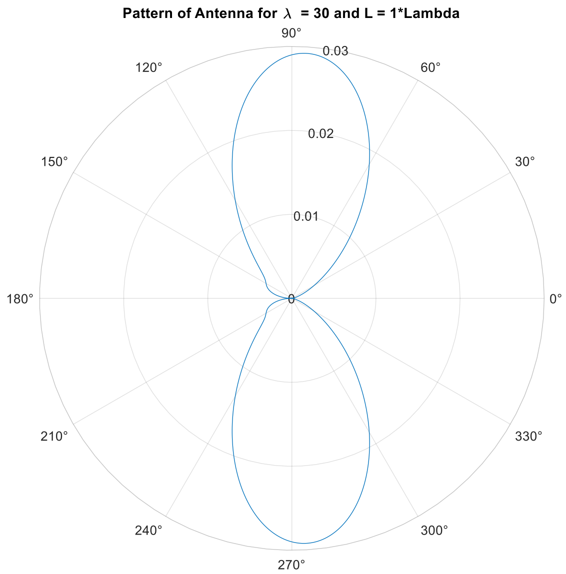


Figure 24

- Current Distribution over the Antenna:

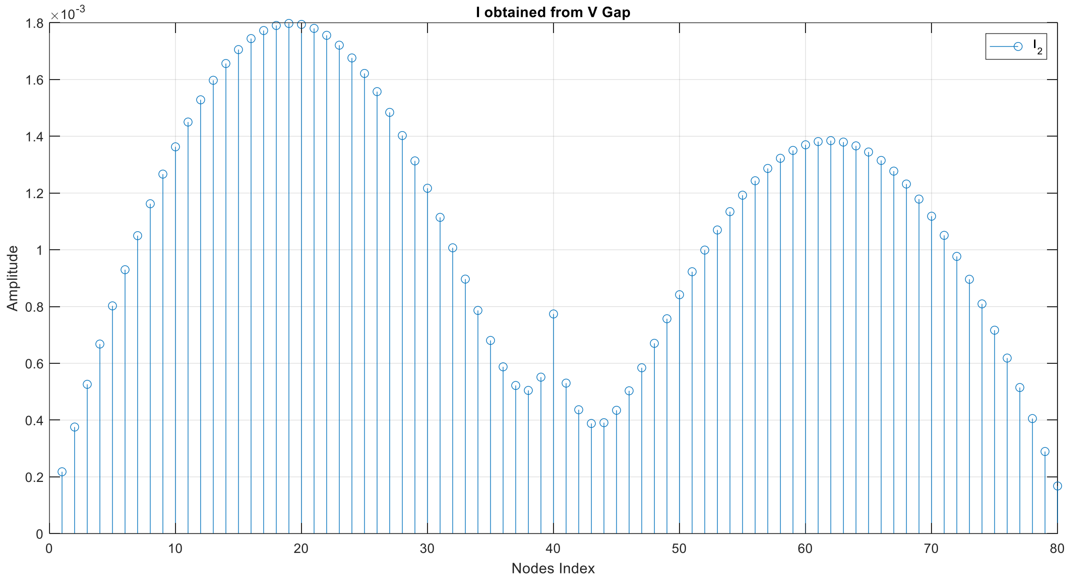


Figure 25

Convergence of the antenna Impedance:

- Testing convergence for L = Lambda/2

For different values of N we get different values of Z_in but they are almost the same:

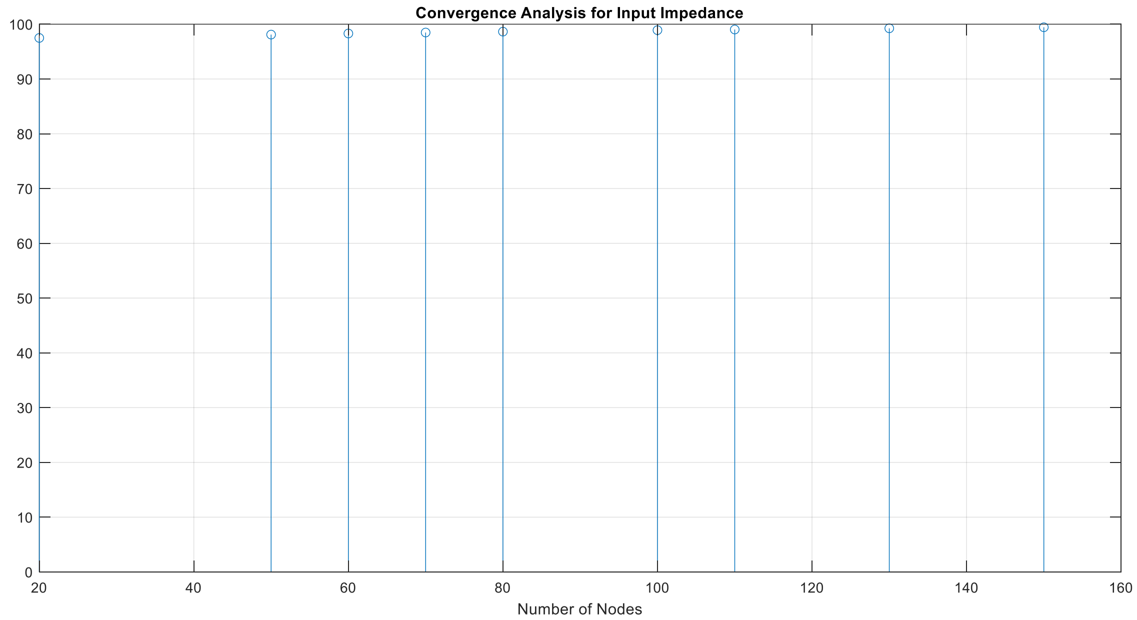


Figure 26

It is clear that, with this value as the impedance, we are not getting the exact answer, but we are converging to obtainable answer from MoM.

Part-3)

Input impedance and pattern of the dipole in MATLAB using exact formulas obtained from Constantine A. Balanis Antenna Book 3rd edition:

Figure 27

Compare our results with analytical solution obtained from classical analysis of the dipole antenna:

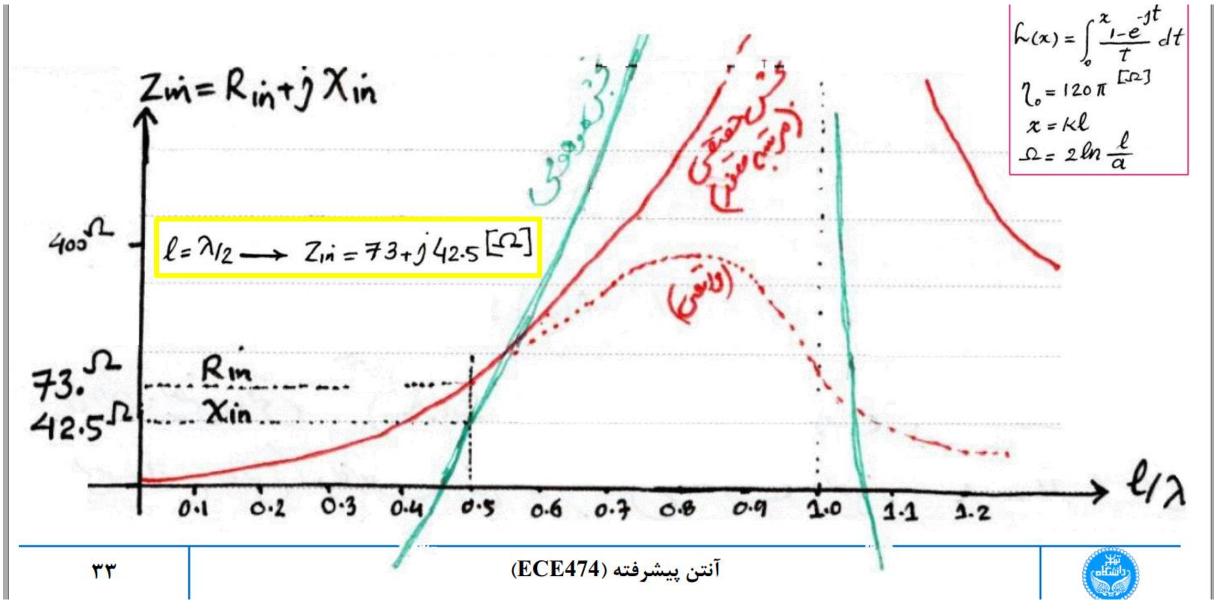


Figure 28

From Balanis we have also:

$$R_{in} = \left[\frac{I_0}{I_{in}}\right]^2 R_r \tag{4-77a}$$

where

R_{in} = radiation resistance at input (feed) terminals
R_r = radiation resistance at current maximum Eq. (4-70)
I_0 = current maximum
I_{in} = current at input terminals

Figure 29

where the radiation resistance of dipole antenna is considered:

The radiation resistance can be obtained using (4-18) and (4-68) and can be written as

$$\boxed{\begin{aligned} R_r = \frac{2P_{\text{rad}}}{|I_0|^2} = \frac{\eta}{2\pi}\{&C + \ln(kl) - C_i(kl) \\ &+ \tfrac{1}{2}\sin(kl) \times [S_i(2kl) - 2S_i(kl)] \\ &+ \tfrac{1}{2}\cos(kl) \times [C + \ln(kl/2) + C_i(2kl) - 2C_i(kl)]\} \end{aligned}} \tag{4-70}$$

Shown in Figure 4.9 is a plot of R_r as a function of l (in wavelengths) when the antenna is radiating into free-space ($\eta \simeq 120\pi$).

and for different values of ℓ, in figure 4.9 of Balanis' book we get to see the variation of Radiation resistance and input resistance of the dipole antenna:

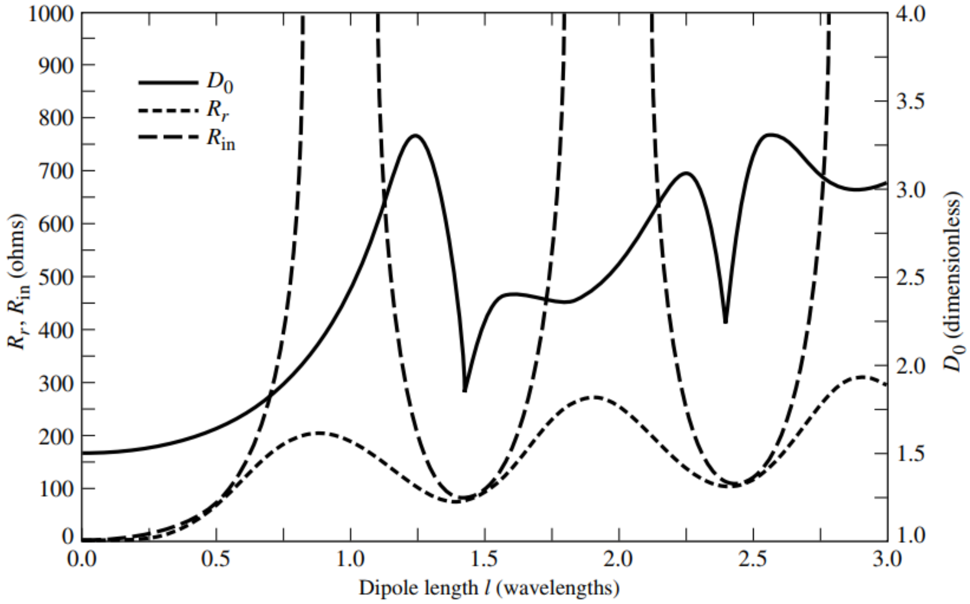


Figure 4.9 Radiation resistance, input resistance and directivity of a thin dipole with sinusoidal current distribution.

Figure 30

and for the Imaginary part of the input impedance, we can use the below equation:

$$X_m = \frac{\eta}{4\pi}\left\{2S_i(kl) + \cos(kl)[2S_i(kl) - S_i(2kl)] - \sin(kl)\left[2C_i(kl) - C_i(2kl) - C_i\left(\frac{2ka^2}{l}\right)\right]\right\} \quad \text{(4-70a)}$$

Figure 31

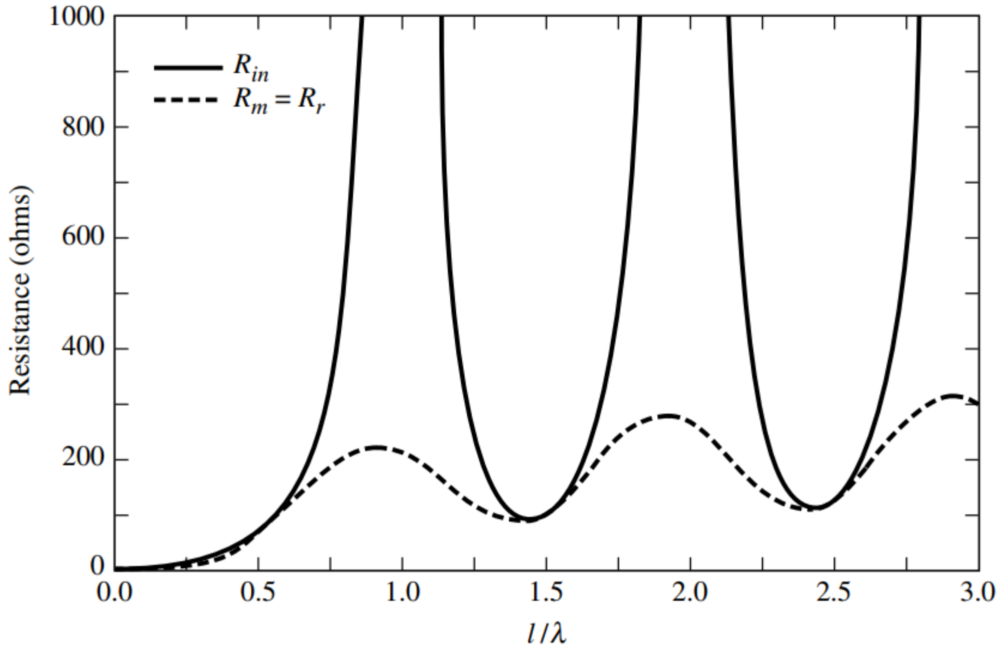
in above equations, Ci(x) and Si(x) are used which are fresnel cosine and sine integrals defined as:

$$C_i(x) = -\int_x^{\infty} \frac{\cos y}{y}\,dy = \int_{\infty}^{x} \frac{\cos y}{y}\,dy \tag{4-68a}$$

$$S_i(x) = \int_0^x \frac{\sin y}{y}\,dy \tag{4-68b}$$

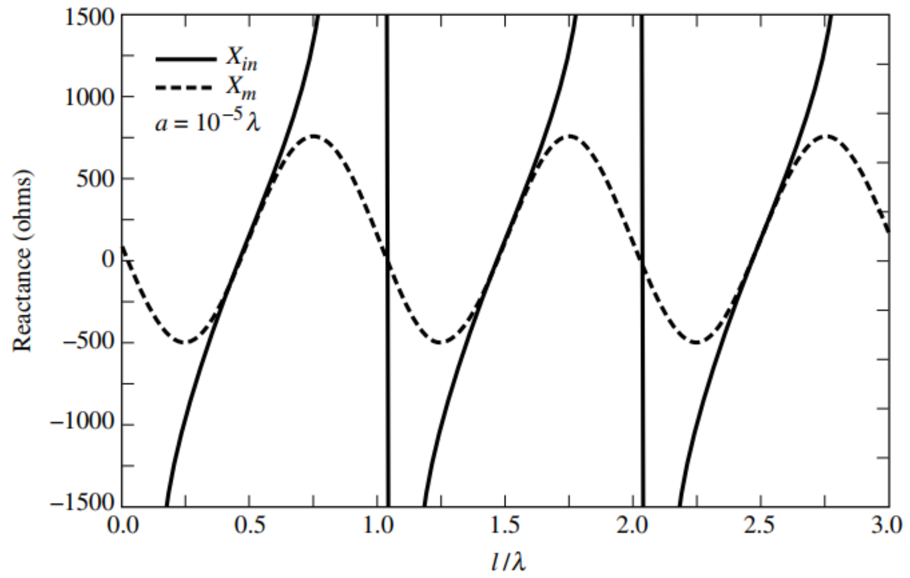
Figure 32

For l/Lambda from 0 to 3 we get the figures below to express the difference between input resistance and the radiation resistance:



(a) Resistance

Figure 33



(b) Reactance

Figure 8.16 Self-resistance and self-reactance of dipole antenna with wire radius of 10^{-5} λ.

Figure 34

Input Impedance using Exact Formulas:

For Dipole with length equal to: Lambda/4

33.7808 +64.3415i <exact>

VS

1.1576e+01 - 4.1315e+02i <MoM>

For Dipole with length equal to: Lambda/2

58.4540 +13.9715i<exact>

VS

86.6484 +47.1377i<MoM>

For Dipole with length equal to: 3*Lambda/4

2.3536e+02 + 9.3291e+01i<exact>

VS

7.1280e+02 + 6.6702e+02i<MoM>

For Dipole with length equal to: Lambda

1.5614e+08 + 2.6760e+33i

VS

7.6199e+02 - 1.0435e+03i

<u>MATLAB’s Antenna-Tool Box:</u>

Also using MATLAB’s Antenna-Tool-Box, we get:

For Lambda = 30 m, L = Lambda/2:

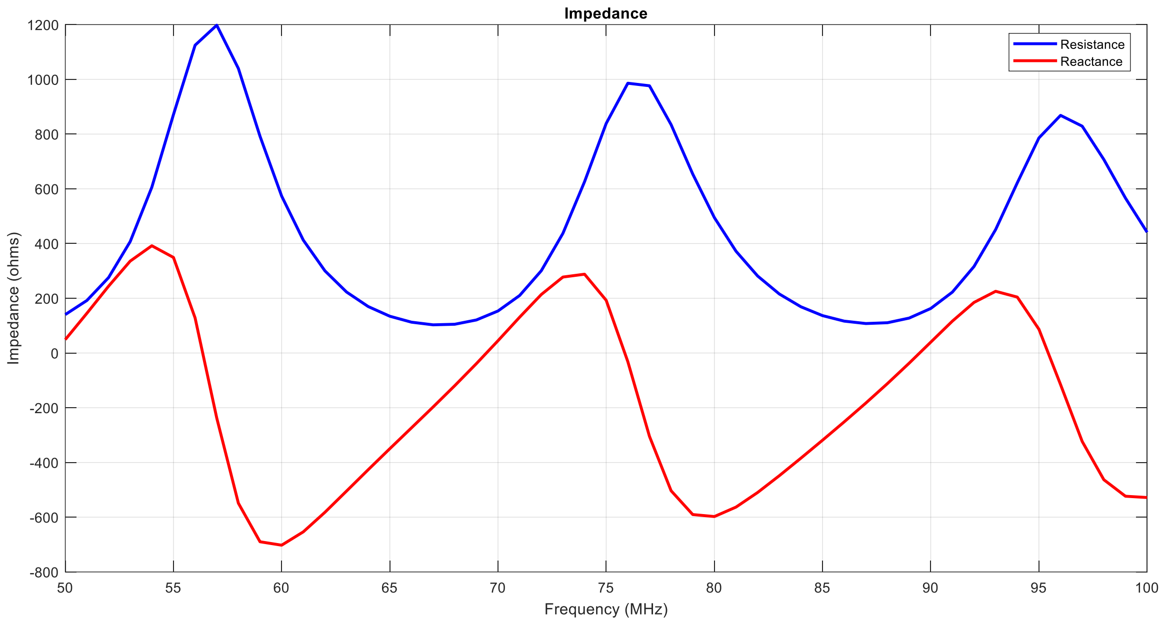


Figure 35

Also, Antenna Patterns are available for different Lengths of Antenna:

- For antenna with L = Lambda/4:

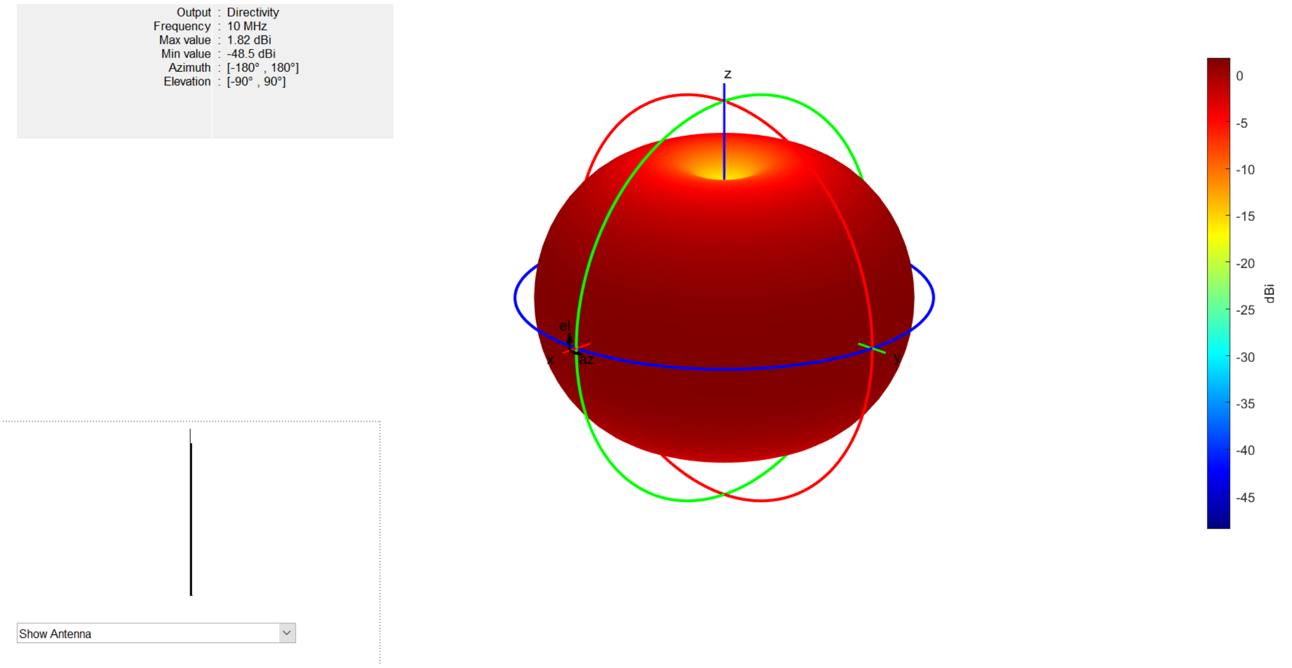


Figure 36

- For antenna with L = Lambda/2:

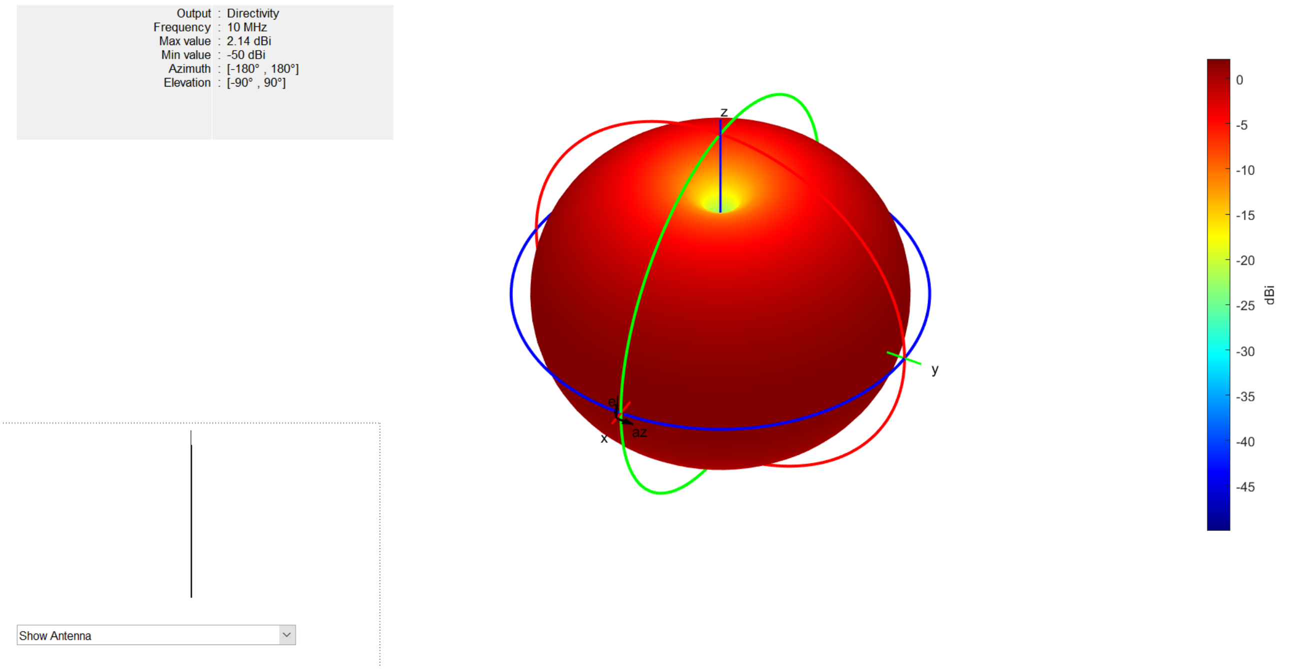


Figure 37

- For antenna with L = 3*Lambda/4:

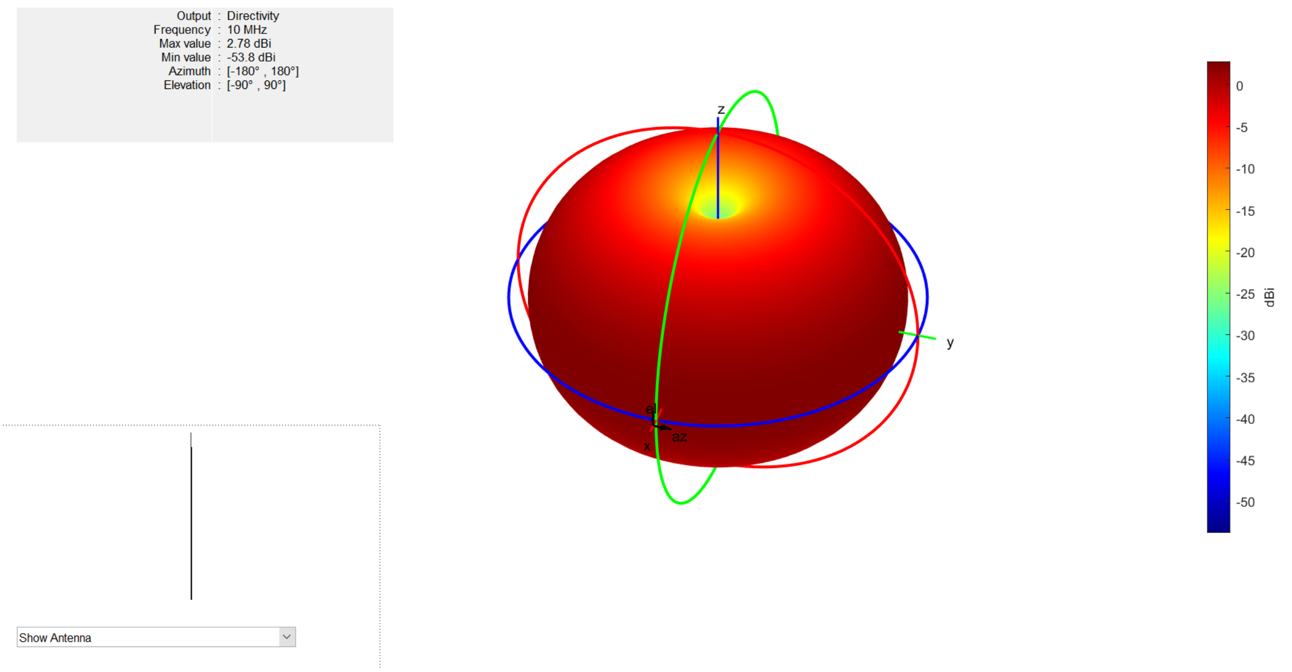


Figure 38

- For antenna with L = Lambda:

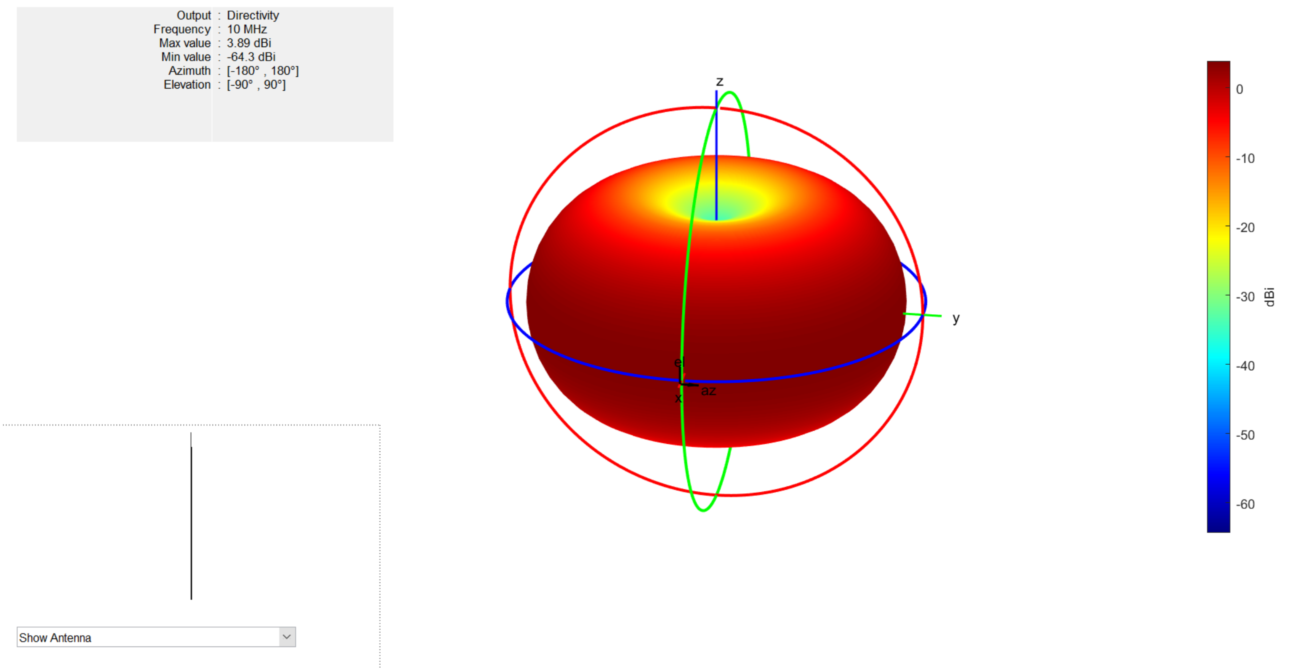


Figure 39

Q2:

۲- قسمت الف مسئله قبل را برای آنتنی که بجای تغذیه از وسط از نقطه‌ای واقع در $\frac{1}{3}$ طول آن تغذیه می‌گردد حل نمائید.

Enumeration over antenna lengths:

- For Antenna with L = Lambda/4

Current Distribution:

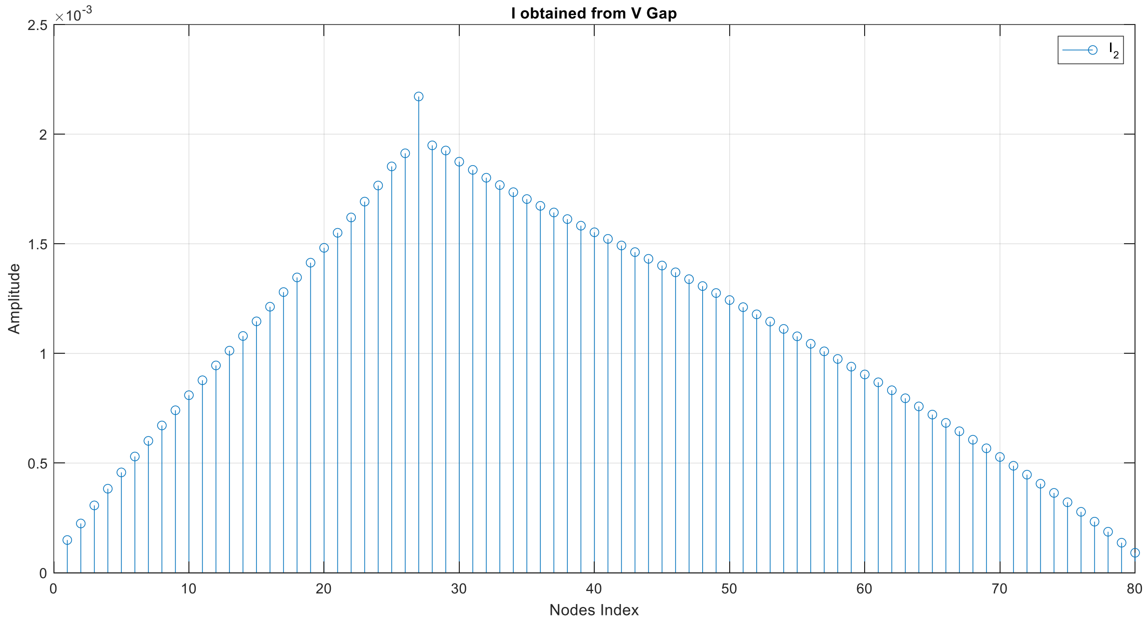


Figure 40

Z Matrix:

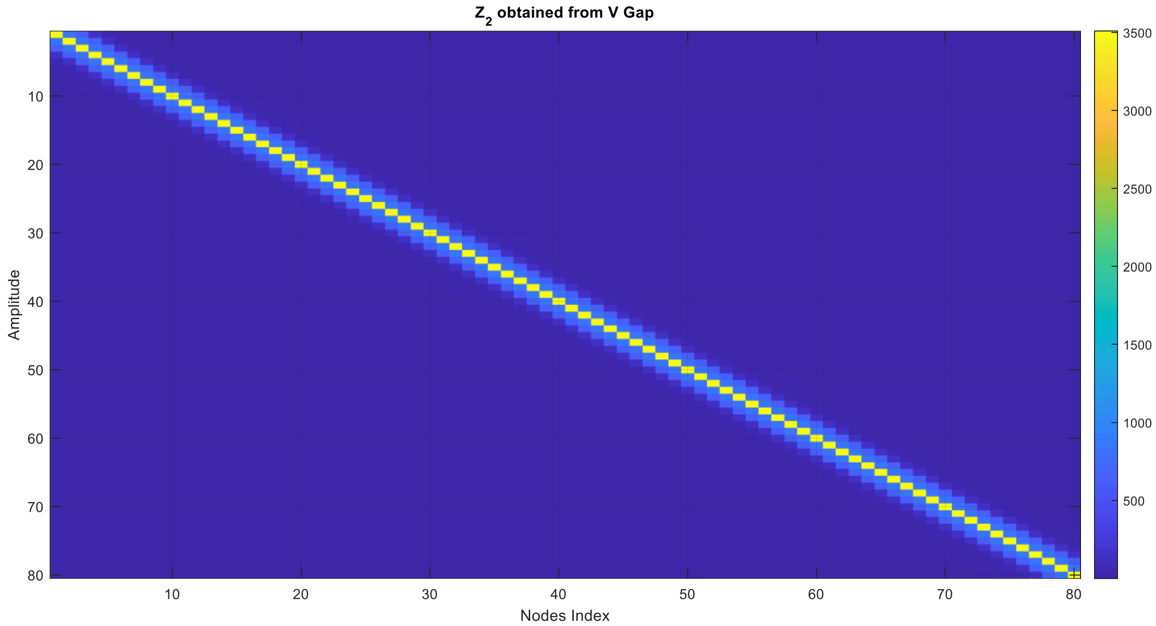


Figure 41

Input Impedance:

1.1607e+01 - 4.6027e+02i

- For Antenna with L = Lambda/2

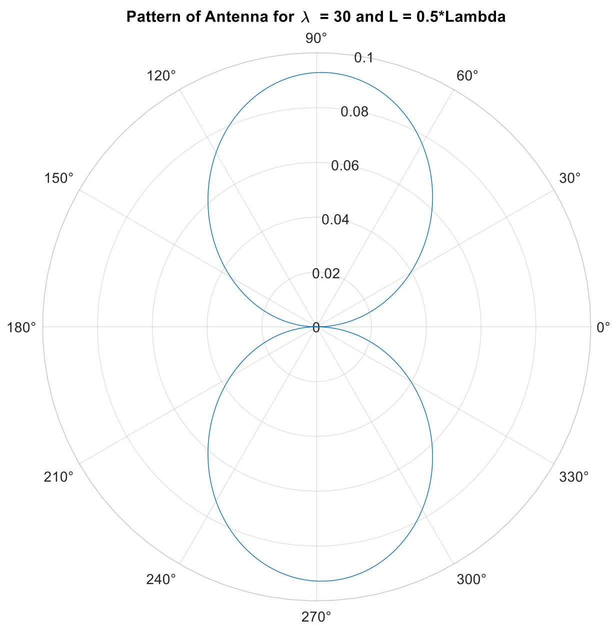


Figure 42

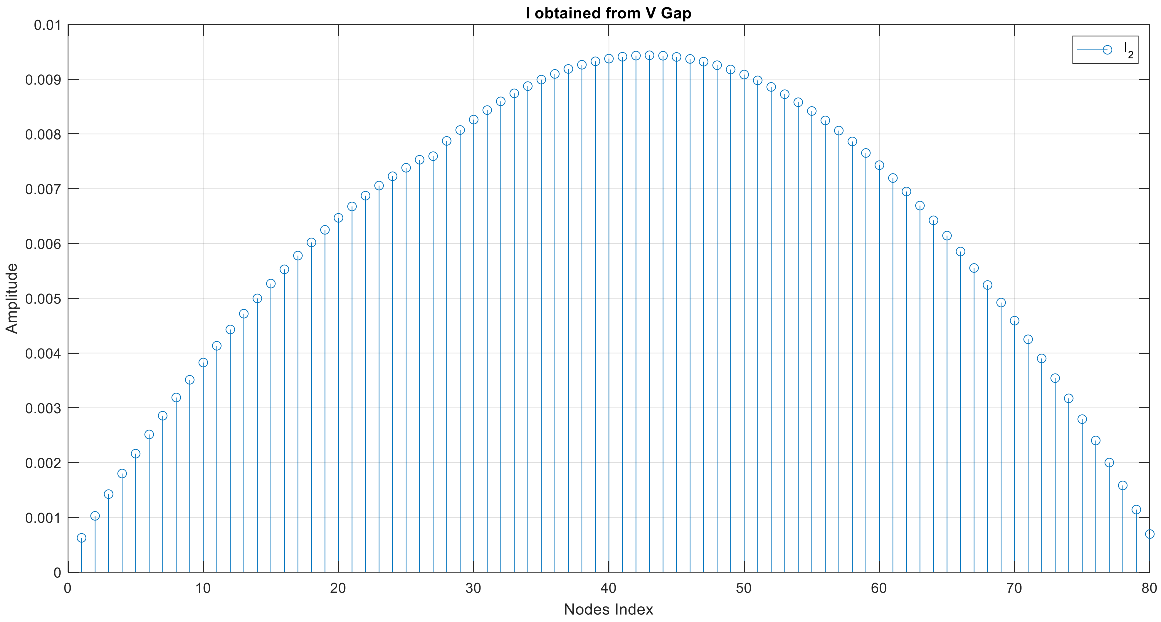


Figure 43

Input Impedance:

1.1908e+02 + 5.6212e+01i

- For Antenna with L = 3*Lambda/4

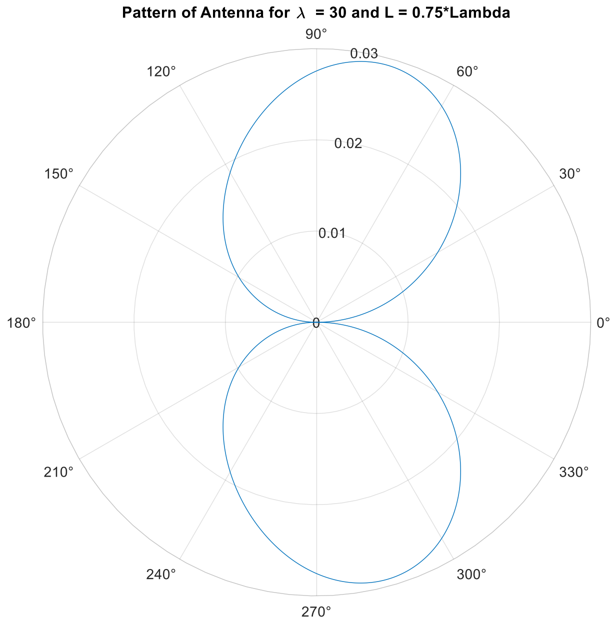


Figure 44

Current Distribution:

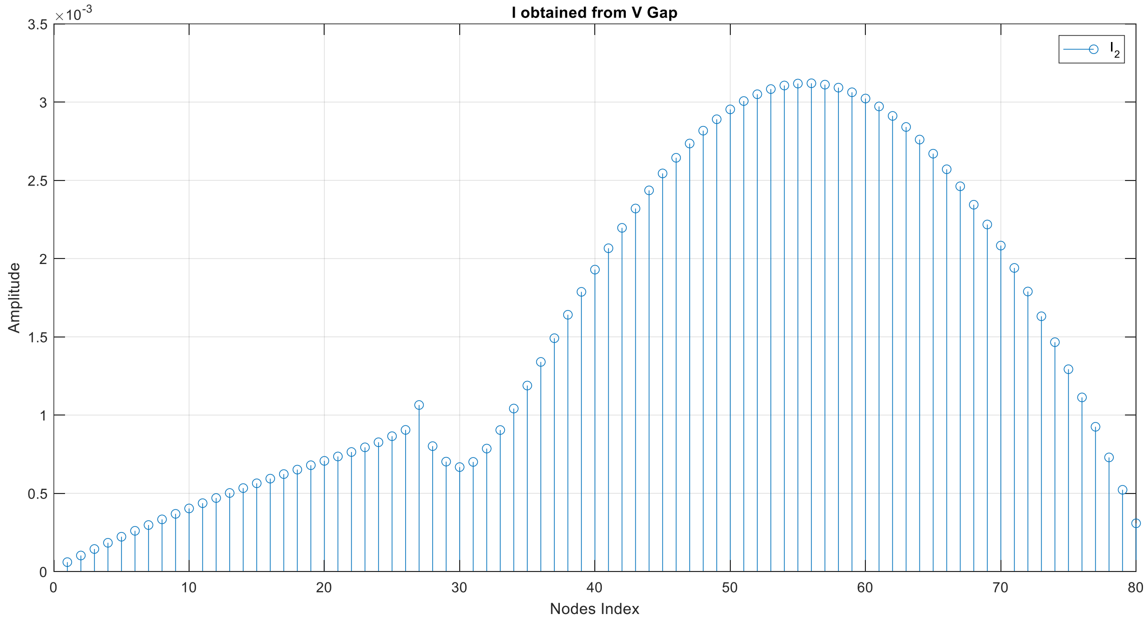


Figure 45

Input Impedance:

5.7372e+02 - 7.4394e+02i

- For Antenna with L = Lambda

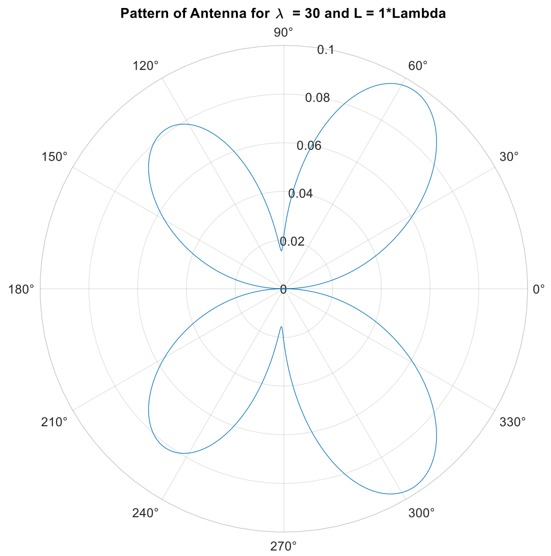


Figure 46

Current Distribution:

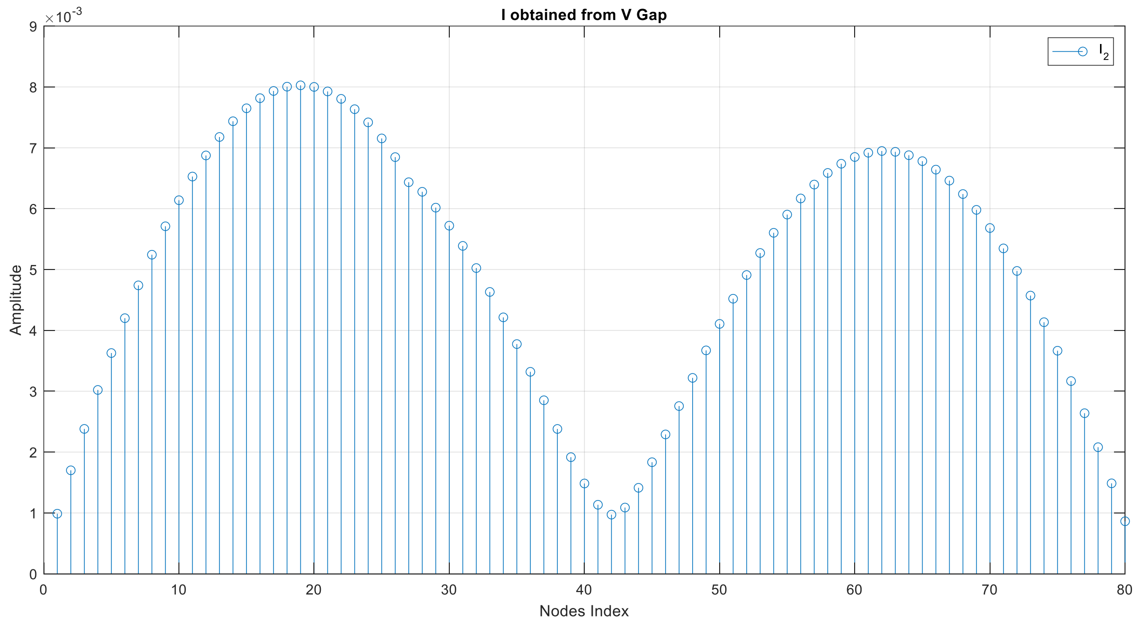


Figure 47

Input Impedance:

$$1.4120e+02 + 6.4949e+01i$$

Q3:

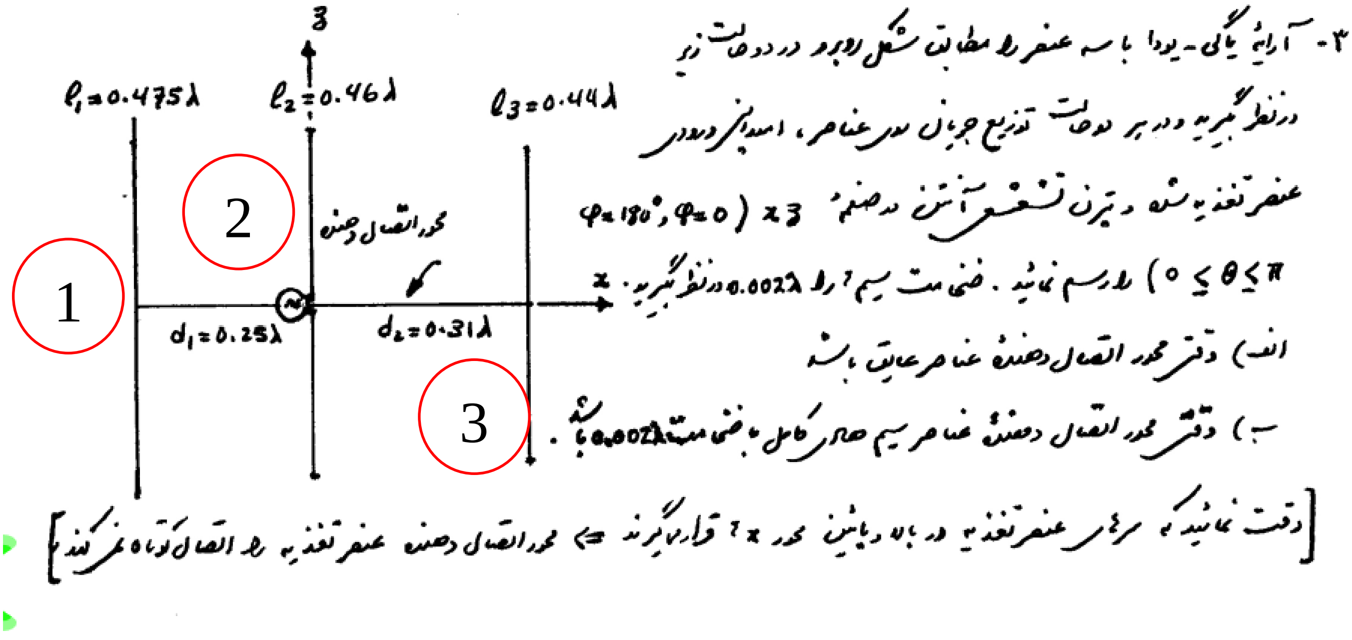


Figure 48

Part-1)

To solve this problem, we again use previous codes to find self-terms for each antenna element and then use R_{eff} idea to model the mutual terms between each of these three antenna elements.

<Grover formulation is used for calculation of R_{eff} >

Total Pattern of this array of dipole antenna is:

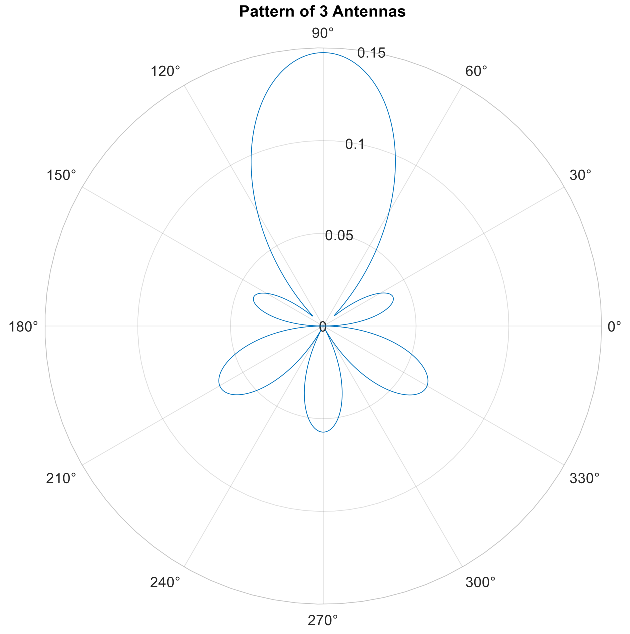


Figure 49

From literature we notice the Yagi-Uda antenna Pattern below:

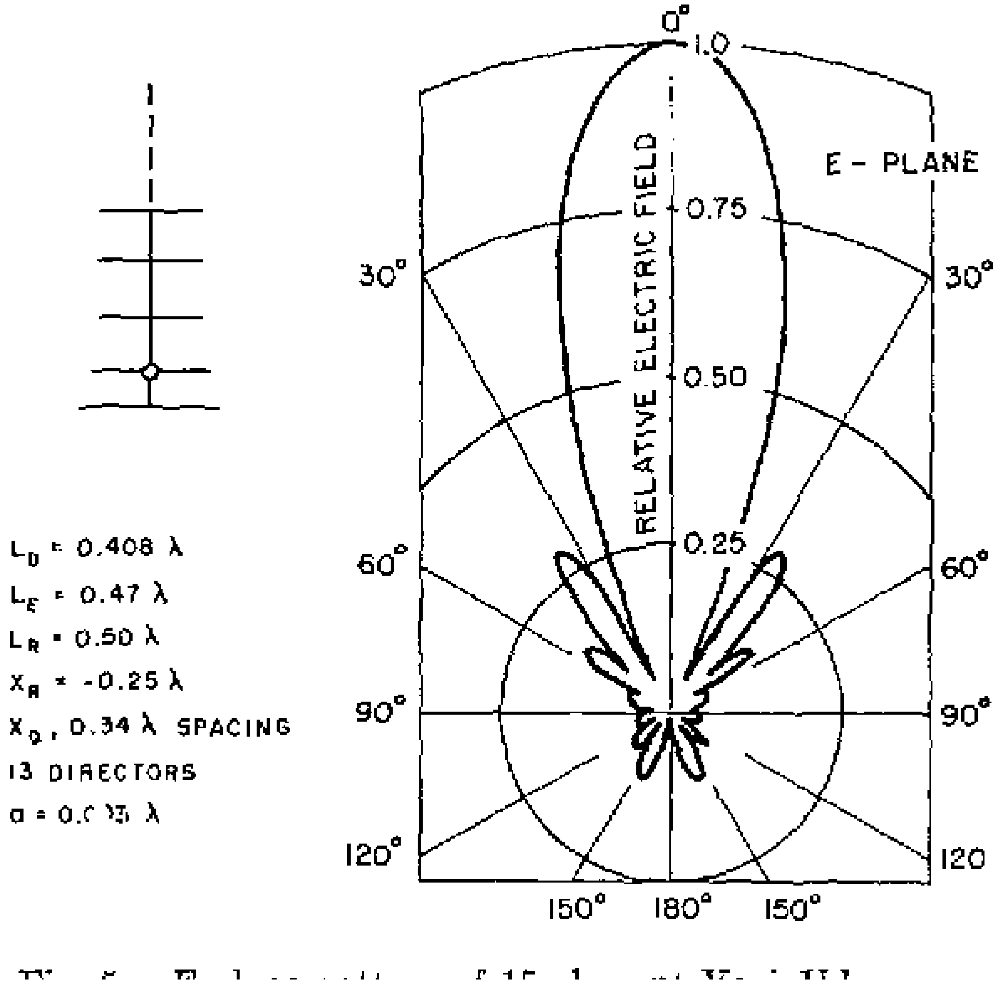


Figure 50

1969: Analysis of yagi-uda-type antennas from **G. Thiele**

- Current Distribution for each Antenna:

1) First Antenna:

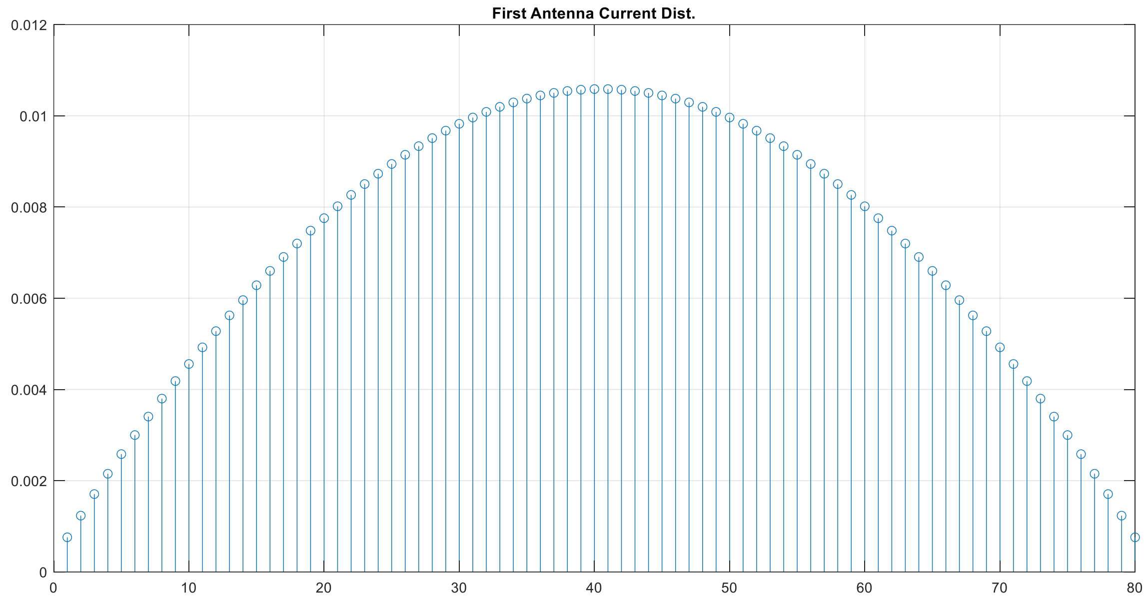


Figure 51

2) Second Antenna:

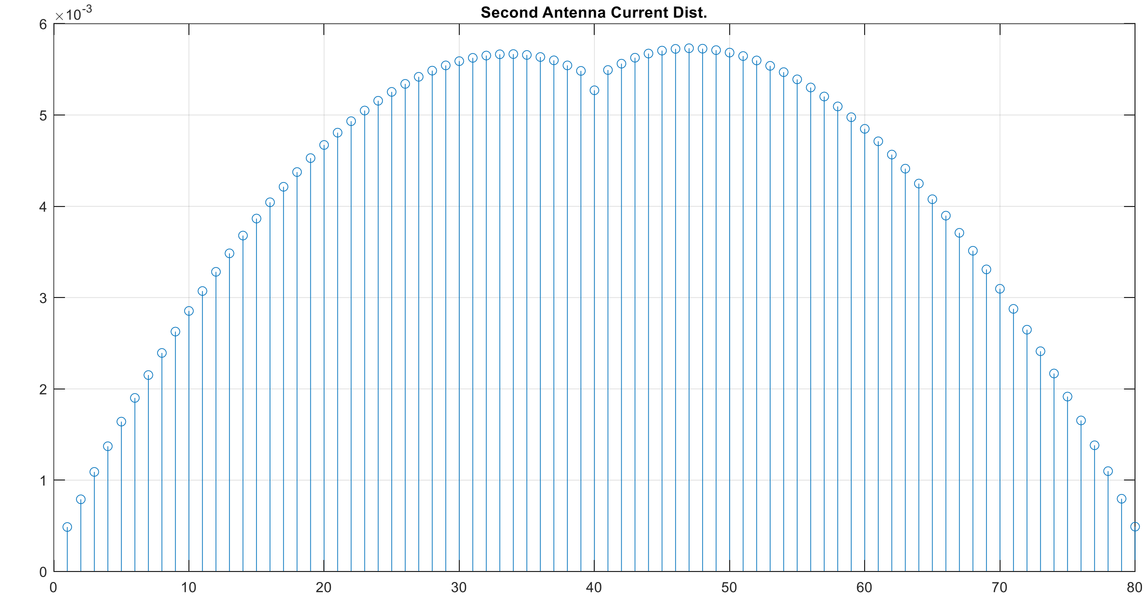


Figure 52

3) Third Antenna:

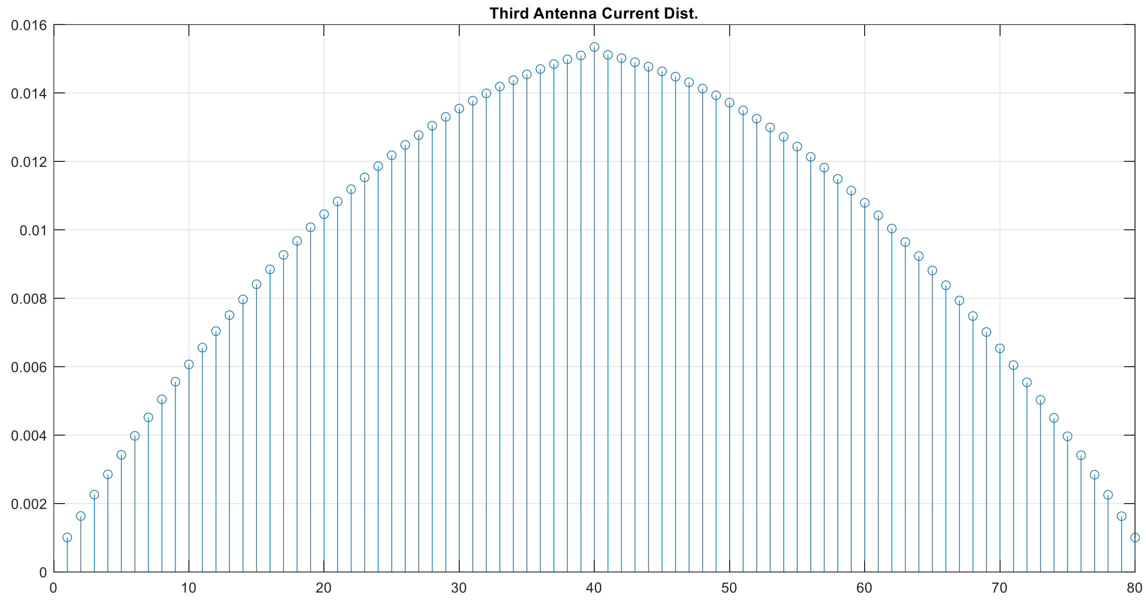


Figure 53

Z Matrix for the total Geometry:

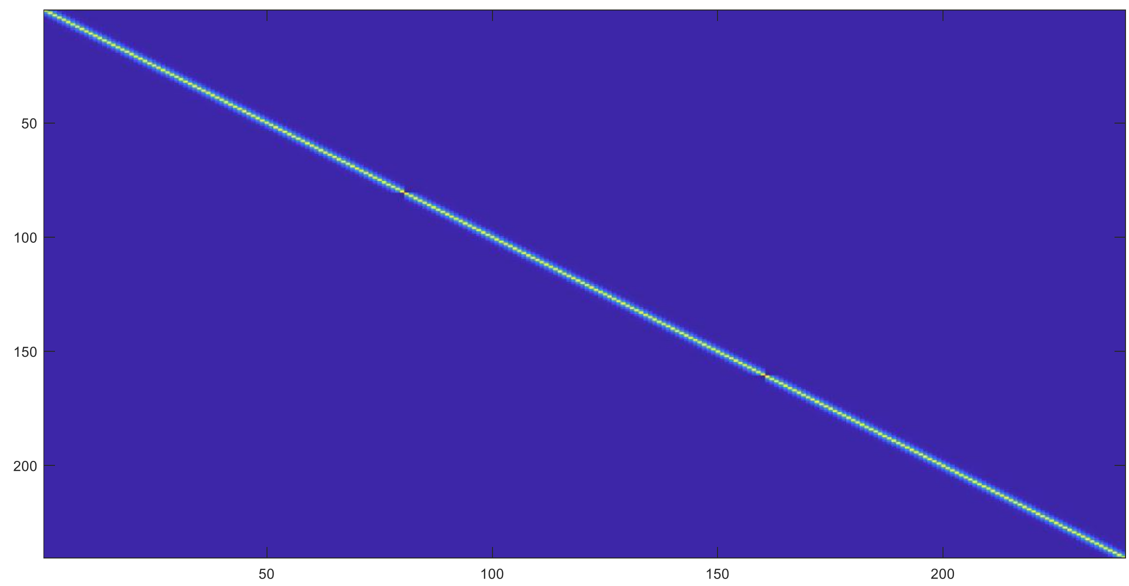


Figure 54

- Each Antenna Pattern:

1) First Antenna:

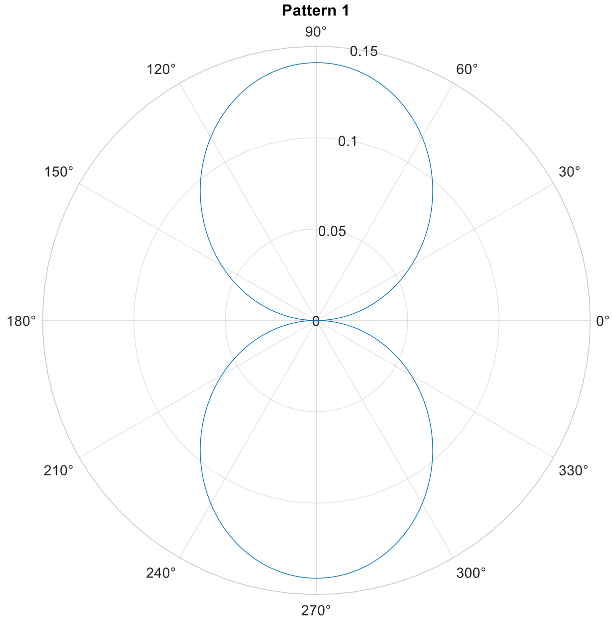


Figure 55

2) Second Antenna:

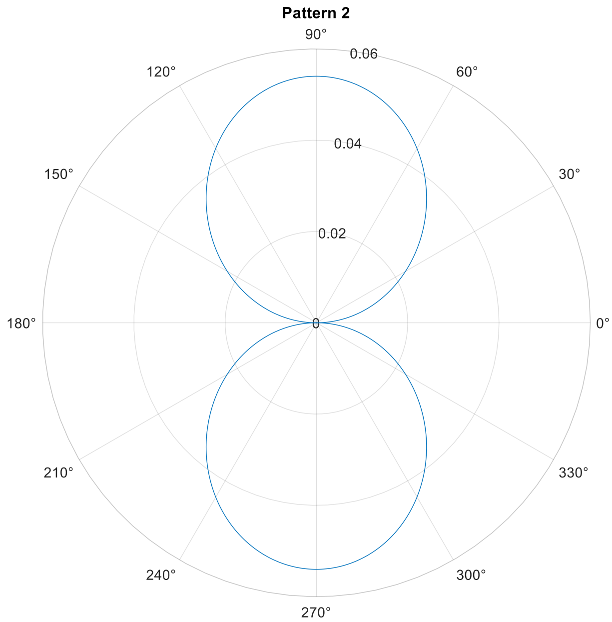


Figure 56

3) Third Antenna:

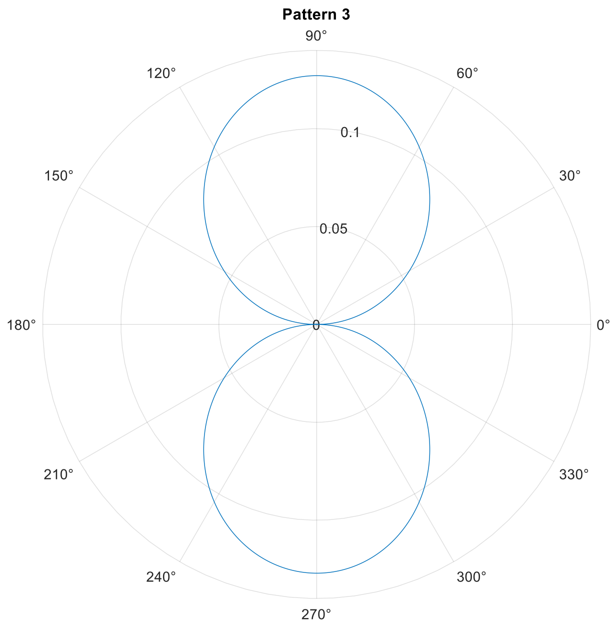


Figure 57

Achieving a good pattern like Yagi-Uda pattern, is due to good arrangement of antenna elements position and their uniform feed.

Better patterns can be reach using non-uniform feeding and weighting in array form of this geometry.

By better patterns, it is conventional to have better gain, better directivity and lower side lobe level and also back lobe level.

- It is necessary to mention that every single pattern can be the best pattern for a certain application but generally, a pencil-beam pattern is a desired one.

Part-2)

I really was trying to solve this sub question, but I ran out of time and energy…

The total idea of this question is the same as multi-object model in MoM which was previously deployed in Part-1 and, also previous homework (Hw6) to calculate the mutual and self-capacitance of 2 siding flat plate capacitors.

We used the model below to solve the part-1 question:

$$Z_{tot} = [[Z_{11}]\,[Z_{12}]\,[Z_{13}];\; [Z_{21}]\,[Z_{22}]\,[Z_{23}];\, [Z_{31}]\,[Z_{32}]\,[Z_{33}]]$$

Where we get:

$$Z_{13} = Z_{31};$$

$$Z_{12} = Z_{21};$$

$$Z_{23} = Z_{32};$$

$$Z_{11}\;,\; Z_{22}, Z_{33} = \;Each\,elements,\,from\,Q1$$

New Code:

```
L_tot            = [0.457 ; 0.46 ; 0.44 ]*Lambda;
Feed_portion_tot = [2 ; 2 ; 2 ];
% Antenna_01_N1 = Total_Worker_Multi_object(N1,L1,a,f,c,0,Feed_portion);
Total_Object = Total_Worker_Multi_object(N1,L_tot ,a ,f,c,Feed_portion_tot );

%%

I_TOT = Total_Object.I_TOT;

I1 = I_TOT(1:N1);
I2 = I_TOT(N1+1:2*N1);
I3 = I_TOT(2*N1+1:3*N1);


Total_Object.Super_I = [I1 ; I2 ; I3] ;

%% Pattern Draw:

Total_Object = Pattern_draw_Total(Total_Object);
```

New_Functions:

Pattern_draw_Total:

```
function Total_Object =Pattern_draw_Total(Total_Object)


L1 =Total_Object.L1 ;
L2 =Total_Object.L2 ;
L3 =Total_Object.L3 ;

Super_I = Total_Object.Super_I;
N1 = Total_Object.N;

I1 = Super_I(1:N1);
I2 = Super_I(N1+1:2*N1);
I3 = Super_I(2*N1+1:3*N1);

D = Total_Object.D;

Lambda = Total_Object.Lambda;
delta_l =Total_Object.delta_l;
k =Total_Object.k;

theta = -180: 0.1 :180 ;



zn1 = linspace(-L1/2,L1/2,length(I1))';
Pattern1 = sind(theta).*sum( delta_l*I1.*exp(1j*k*zn1*cosd(theta)) ) ;

zn2 = linspace(-L2/2,L2/2,length(I2))';
Pattern2 = sind(theta).*sum( delta_l*I2.*exp(1j*k*zn2*cosd(theta)) ) ;
```

```
zn3 = linspace(-L3/2,L3/2,length(I3))';
Pattern3 = sind(theta).*sum( delta_l*I3.*exp(1j*k*zn3*cosd(theta)) ) ;


Pattern_TOT = exp(1j*k*D(1,1)*sind(theta)).*Pattern1 + ...
              exp(1j*k*D(2,1)*sind(theta)).*Pattern2 + ...
              exp(1j*k*D(3,1)*sind(theta)).*Pattern3 ;


Total_Object.theta = theta;
Total_Object.Pattern1_theta = Pattern1;
Total_Object.Pattern2_theta = Pattern2;
Total_Object.Pattern3_theta = Pattern3;
Total_Object.Pattern_TOT = Pattern_TOT  ;

figure()
polarplot(pi*theta/180, abs(Pattern_TOT))
title("Pattern of 3 Antennas")
grid on


  end
```

Total_Worker_Multi_object:

```
function Total_Objects = Total_Worker_Multi_object(N,L_tot ,a ,f,c,Feed_portion_tot )

L1 = L_tot(1,1);
L2 = L_tot(2,1);
L3 = L_tot(3,1);

Total_Objects.L1 = L1;
Total_Objects.L2 = L2;
Total_Objects.L3 = L3;

Feed_portion1 = Feed_portion_tot(1,1);
Feed_portion2 = Feed_portion_tot(2,1);
Feed_portion3 = Feed_portion_tot(3,1);

Total_Objects.Feed_portion1 = Feed_portion1;
Total_Objects.Feed_portion2 = Feed_portion2;
Total_Objects.Feed_portion3 = Feed_portion3;



Antenna_element_1 = Total_Worker(N,L1,a,f,c,0,Feed_portion1);
Antenna_element_2 = Total_Worker(N,L2,a,f,c,0,Feed_portion2);
Antenna_element_3 = Total_Worker(N,L3,a,f,c,0,Feed_portion3);


Total_Objects.Antenna_element_1 = Antenna_element_1;
Total_Objects.Antenna_element_2 = Antenna_element_2;
Total_Objects.Antenna_element_3 = Antenna_element_3;


Total_Objects.delta_l  = Antenna_element_3.delta_l;
Total_Objects.f = f;
Total_Objects.k = Antenna_element_3.k;
```

```
Z_11  = Antenna_element_1.Z2;
Z_22  = Antenna_element_2.Z2;
Z_33  = Antenna_element_3.Z2;

Total_Objects.Z_11 = Z_11;
Total_Objects.Z_22 = Z_22;
Total_Objects.Z_33 = Z_33;


Lambda = Antenna_element_1.Lambda;

Total_Objects.Lambda = Lambda;
% Mutual Term between Antenna 1 and 2:
d1_2 = 0.25*Lambda;
Antenna_element_1_2 = Total_Worker(N,L1,a+d1_2,f,c,0,Feed_portion1);
Z_12 = Antenna_element_1_2.Z2;

Total_Objects.d1_2 = d1_2;
Total_Objects.Antenna_element_1_2 = Antenna_element_1_2;
Total_Objects.Z_12 = Z_12;

% Mutual Term between Antenna 1 and 3:
d1_3 = 0.25*Lambda + 0.31*Lambda ;
Antenna_element_1_3 = Total_Worker(N,L1,a+d1_3,f,c,0,Feed_portion1);
Z_13 = Antenna_element_1_3.Z2;


Total_Objects.d1_3 = d1_3;
Total_Objects.Antenna_element_1_3 = Antenna_element_1_3;
Total_Objects.Z_13 = Z_13;

% Mutual Term between Antenna 2 and 3:
d2_3 =  0.31*Lambda;
Antenna_element_2_3 = Total_Worker(N,L2,a+d2_3,f,c,0,Feed_portion2);
Z_23 = Antenna_element_2_3.Z2;


Total_Objects.d2_3 = d2_3;
Total_Objects.Antenna_element_2_3 = Antenna_element_2_3;
Total_Objects.Z_23 = Z_23;


Total_Objects.D = [0 ; d1_2 ; d1_3];

Z_TOT = [ Z_11 , Z_12 , Z_13 ;...
          Z_12 , Z_22 , Z_23 ;...
          Z_13 , Z_23 , Z_33 ] ;

V1 = Antenna_element_1.V2;
V2 = Antenna_element_2.V2;
V3 = Antenna_element_3.V2;

Total_Objects.Z_TOT = Z_TOT;
Total_Objects.V1 = V1;
Total_Objects.V2 = V2;
Total_Objects.V3 = V3;



V_TOT = [ V1(1:end-1) ; V2(1:end-1) ; V3(1:end-1)  ];
```

```
I_TOT   =  inv(Z_TOT) * V_TOT     ;


Total_Objects.V_TOT = V_TOT;
Total_Objects.I_TOT = I_TOT;
Total_Objects.N = N;

end
```

The End