# Bit Error Rate n Presence of AWGN

# Roll No.: 42239

clc

clear all

close all

bit\_count = 100000; %no. of random bits to be generated for a single shot of BER calculation

SNR = 0: 1: 10; %Range of SNR over which to simulate

for k = 1: 1: length(SNR)

tote = 0; %total error bits

totb = 0; %total bits

while tote < 100 %until you get 100 errors

rbits = round(rand(1,bit\_count)); %generate random bits

tx = -2\*(rbits-0.5); % BPSK Modulation: Directly to Bipolar NRZ

N0 = 1/10^(SNR(k)/10); %noise level

rx = tx + sqrt(N0/2)\*(randn(1,length(tx))+i\*randn(1,length(tx)));

rx2 = rx < 0; % BPSK demodulator logic at the Receiver

diff = rbits - rx2; % Calculate Bit Errors

tote = tote + sum(abs(diff)); %total errors

totb = totb + length(rbits); %total bits generated

end

BER(k) = tote / totb; % Calculate Bit Error Rate

end

semilogy(SNR,BER,'\*r');

hold on;

xlabel('Eb/No (dB)');

ylabel('BER');

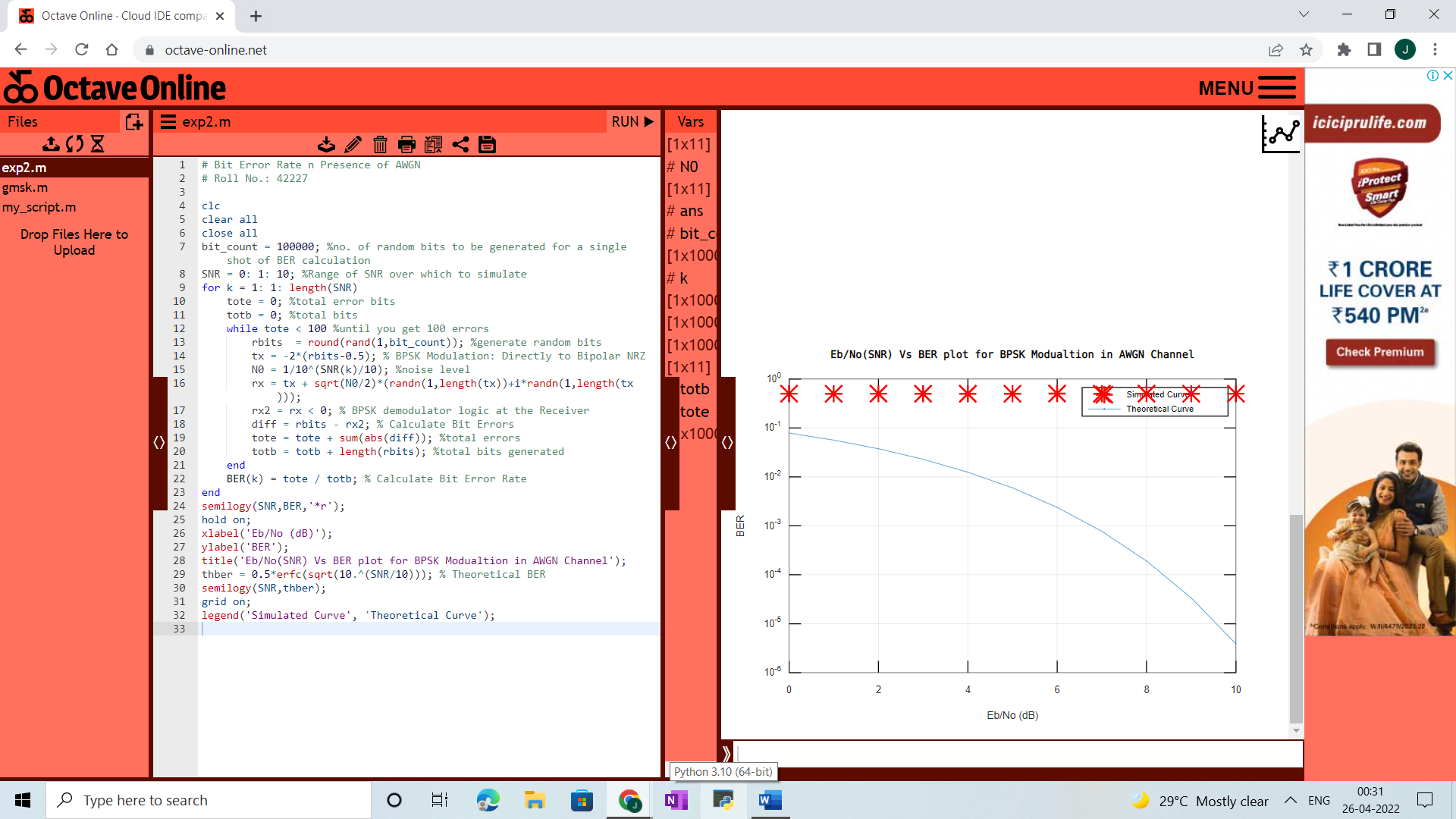
title('Eb/No(SNR) Vs BER plot for BPSK Modualtion in AWGN Channel');

thber = 0.5\*erfc(sqrt(10.^(SNR/10))); % Theoretical BER

semilogy(SNR,thber);

grid on;

legend('Simulated Curve', 'Theoretical Curve');



**CODE:**

# DSSS CDMA

# Roll No.: 42239

clc;

close all;

clear all;

%input('Enter the inpt Bits:');

b=[1 0 1 0 1 0 1 0 1 0]

ln=length(b);

% Converting bit 0 to -1

for i=1:ln

if b(i)==0

b(i)=-1

end

end

%Generating the bit sequence with each bit 8 sample ling

k=1;

for i= 1:ln

for j=1:8

bb(k)=b(i);

j=j+1;

k=k+1;

end

i=i+1;

end

len=length(bb);

subplot(2,1,1);

stairs(bb,'linewidth',2);

axis([0 len,-2 3]);

title('ORIGINAL BIT SEQUENCE b(t)');

%Generating the pseudorandom bit pattern for spreading

pr\_sig=round(rand(1,len));

for i=1:len

if pr\_sig(i)==0

pr\_sig(i)=-1;

end

end

subplot(2,1,2);

stairs(pr\_sig,'linewidth',2);

axis([0 len -2 3]);

title('PSEUDORANDOM BIT SEQUENCE pr\_sig(t)');

%Multiplying bit sequence with psseudorandom sequence

for i=1:len

bbs(i)=bb(i).\*pr\_sig(i);

end

%Modulating the hopped signal

dsss=[];

t=0:1/10:2\*pi;

c1=cos(t);

c2=cos(t+pi);

len=length(bb);

for k=1:len

if bbs(1,k)== -1

dsss=[dsss c1];

else

dsss=[dsss c2];

end

end

figure;

subplot(2,1,1);

stairs(bbs,'linewidth',2);

axis ([0 len -2 3]);

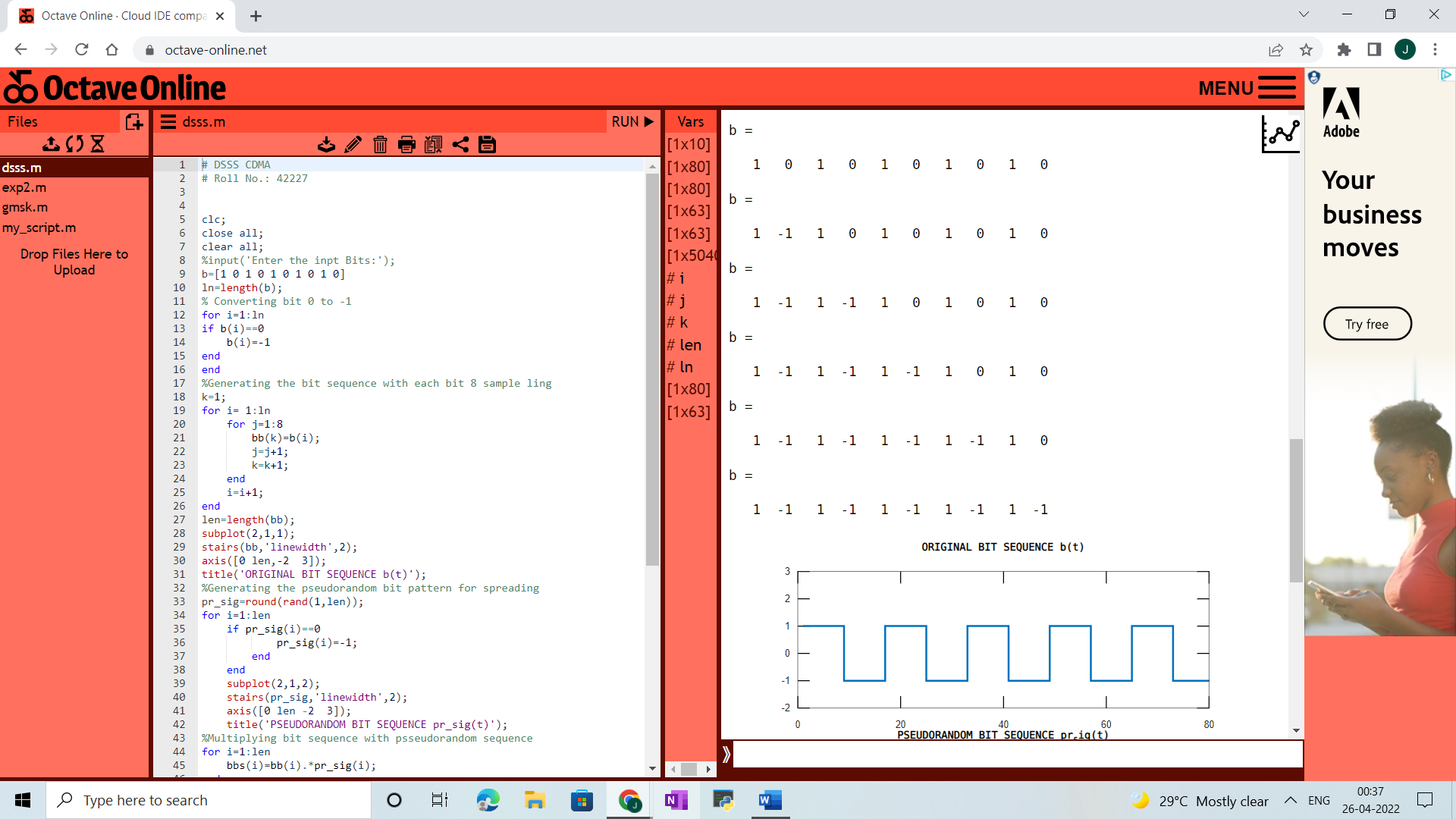
title('MULTIPLIER OUTPUT SEQUENCE b(t)\*pr\_sig(t)');

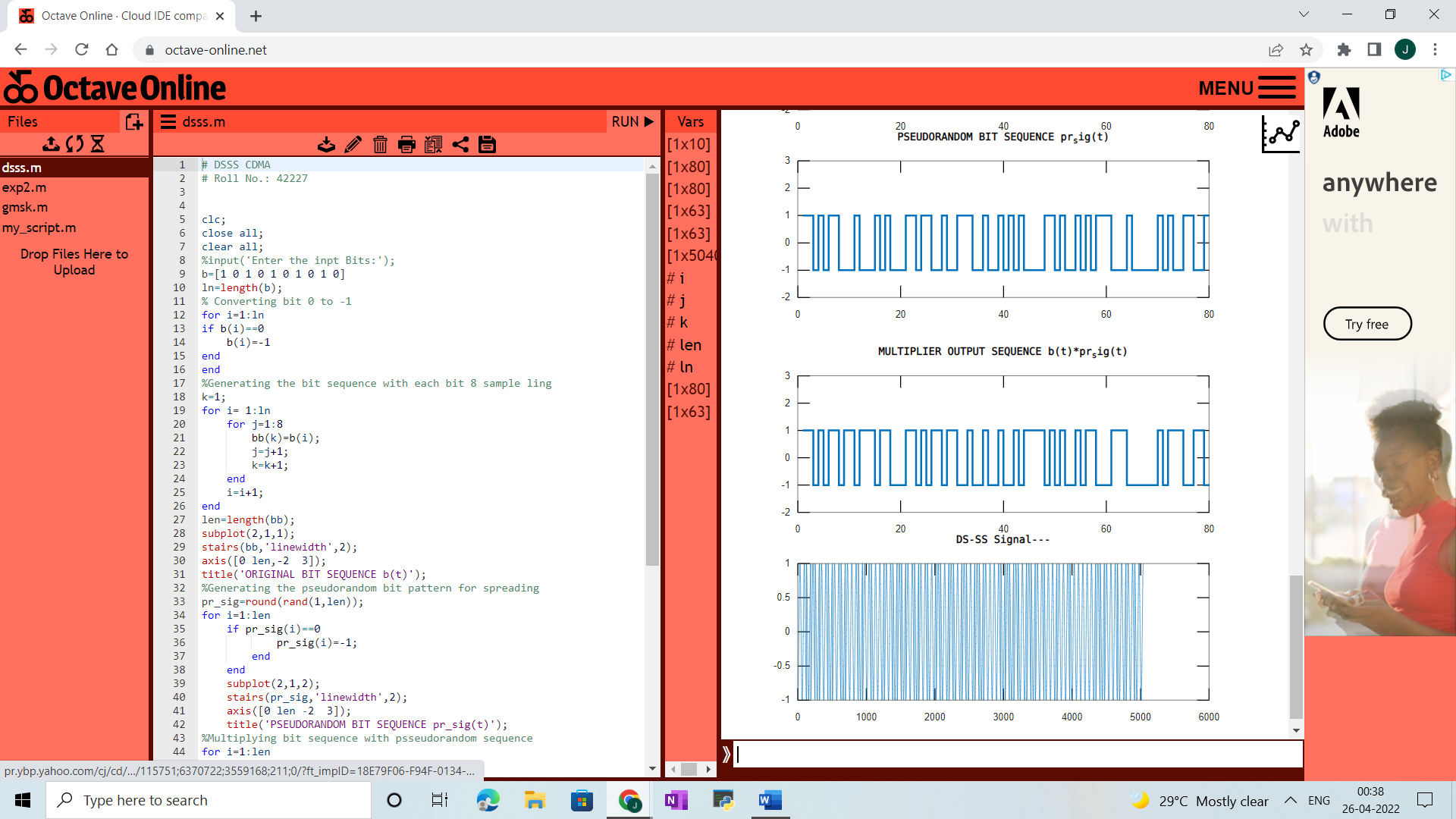
subplot(2,1,2);

plot(dsss);

title('DS-SS Signal---');

**OUTPUT:**





**CODE:**

# GMSK

# Roll No.: 42239

clc;

clear all;

close all;

DRate=1; %data rate or 1 bit in one second

M=18; %no of smples per bit

N=36; %no of bits for simulation [-18:18]

BT=0.5;%Bandwidth\*period (cannot change)

T=1/DRate; %data period ,i.e 1 bit in one second

Ts=T/M;

k=[-18:18]; %Chens value more than needed;%only introduces a little more delay

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%COSTRUCTION OF GAUSSIAN FILTER FOLLOWED BY SHAPING OF DATA BITS USING

%GAUSSIAN FILTER

alpha=sqrt(log(2))/(2\*pi\*BT);

h=exp(-(k\*Ts).^2/(2\*alpha^2\*T^2))/(sqrt(2\*pi)\*alpha\*T); %Gaussian filter response in time domain

figure(1);

plot(h,'\*r')

title('Response of Gaussian filter');

xlabel('Sample at Ts');

ylabel('Normalized Magnitude');

ipbit=[1 1 0 0 0 1 1 1];

m=filter(h,1,ipbit);

m\_sig=[];

fc=1

fs=10^3;

w=2\*pi\*fc;

T=1

%%Weighting function generator

t=-1:1/fs:7; %advanced inphase %length is 8fs+1

t1=0:1/fs:8; %quadrature %length is 8fs+1

cos\_w=cos(2\*pi\*t/2);%Inphase

sin\_w=sin(2\*pi\*t1/2); %quadrature

ai=ipbit(1:2:end);

aq=ipbit(2:2:end);

m1=filter(h,1,ai);

m2=filter(h,1,aq);

ai\_c=[];

aq\_s=[];

ai\_c=kron(m1(1:2:end),ones(1,4000));

aq\_s=kron(m2(2:2:end),ones(1,4000));

ai\_c=[ai\_c zeros(1,1)]; %to make length=8fs+1

aq\_s=[zeros(1,1) aq\_s];%to make length=8fs+1

aii=ai\_c.\*cos\_w;

aqq=aq\_s.\*sin\_w;

i=aii.\*cos(w\*t);

q=aqq.\*sin(w\*t);

m\_sig=i+q;

figure(2)

subplot(2,1,1)

plot(i);

subplot(2,1,2)

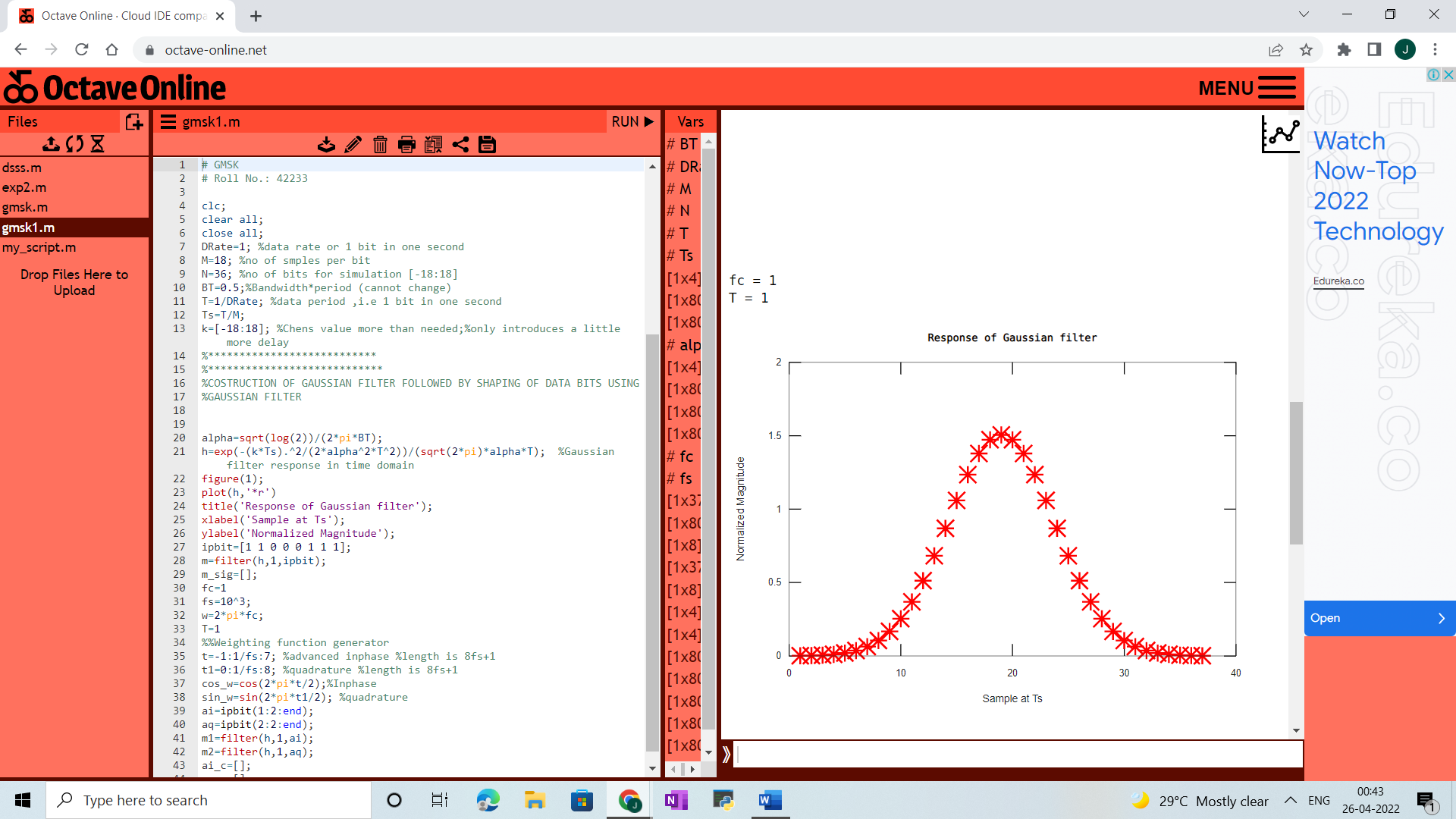
plot(q);

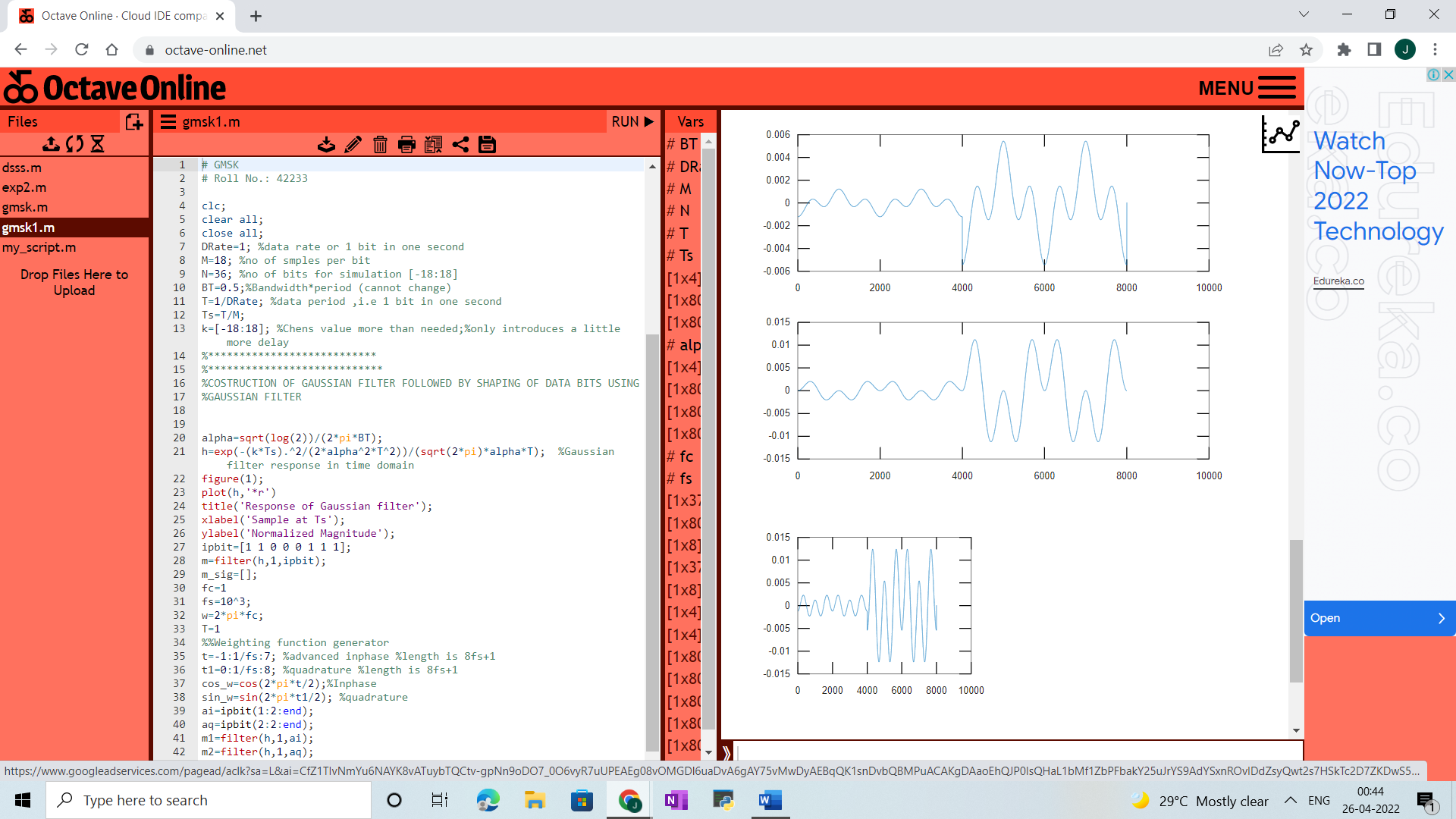
figure(3)

subplot(2,2,1);

plot(m\_sig);

**OUTPUT:**





**CODE:**

# HATA/Okumura Model

# Roll No.: 42239

% Okumura/Hata Model 3

clc;

close all;

clear all;

d = 1:0.01:20;

hm = 5;

hb1 = 30;

hb2 = 100;

hb3 = 200;

fc = 1000;

% a. For Large Cities

% fc >= 400MHz

ahm = 3.2\*(log10(11.75\*hm)).^2 - 4.97;

% A. Typical Urban

L50urban1 = 69.55 + 26.16\*log10(fc) + (44.9 - 6.55\*log10(hb1))\*log10(d) - 13.82\*log10(hb1) - ahm;

L50urban2 = 69.55 + 26.16\*log10(fc) + (44.9 - 6.55\*log10(hb2))\*log10(d) - 13.82\*log10(hb2) - ahm;

L50urban3 = 69.55 + 26.16\*log10(fc) + (44.9 - 6.55\*log10(hb3))\*log10(d) - 13.82\*log10(hb3) - ahm;

% B. Typical Suburban

L50suburban1 = L50urban1 - 2\*(log10(fc/28)).^2 - 5.4;

L50suburban2 = L50urban2 - 2\*(log10(fc/28)).^2 - 5.4;

L50suburban3 = L50urban3 - 2\*(log10(fc/28)).^2 - 5.4;

% C. Typical Rural

L50rural1 = L50urban1 - 4.78\*(log10(fc)).^2 + 18.33\*log10(fc) - 40.94;

L50rural2 = L50urban2 - 4.78\*(log10(fc)).^2 + 18.33\*log10(fc) - 40.94;

L50rural3 = L50urban3 - 4.78\*(log10(fc)).^2 + 18.33\*log10(fc) - 40.94;

figure(1);

plot(d, L50urban1, 'r', d, L50urban2, '--r', d, L50urban3,':r');

hold on;

legend('large urban hb=30', 'large urban hb=100', 'large urban hb=200', 'suburban hb=30', 'suburban hb=100', 'suburban hb=200', 'rural hb=30', 'rural hb=100','rural hb=200');

figure(2);

plot(d, L50suburban1, 'b', d, L50suburban2, '--b', d, L50suburban3, ':b');

hold on;

legend('large urban hb=30', 'large urban hb=100', 'large urban hb=200', 'suburban hb=30', 'suburban hb=100', 'suburban hb=200', 'rural hb=30', 'rural hb=100','rural hb=200');

figure(3);

plot(d, L50rural1, 'g', d, L50rural2, '--g', d, L50rural3, ':g');

hold on;

legend('large urban hb=30', 'large urban hb=100', 'large urban hb=200', 'suburban hb=30', 'suburban hb=100', 'suburban hb=200', 'rural hb=30', 'rural hb=100','rural hb=200');

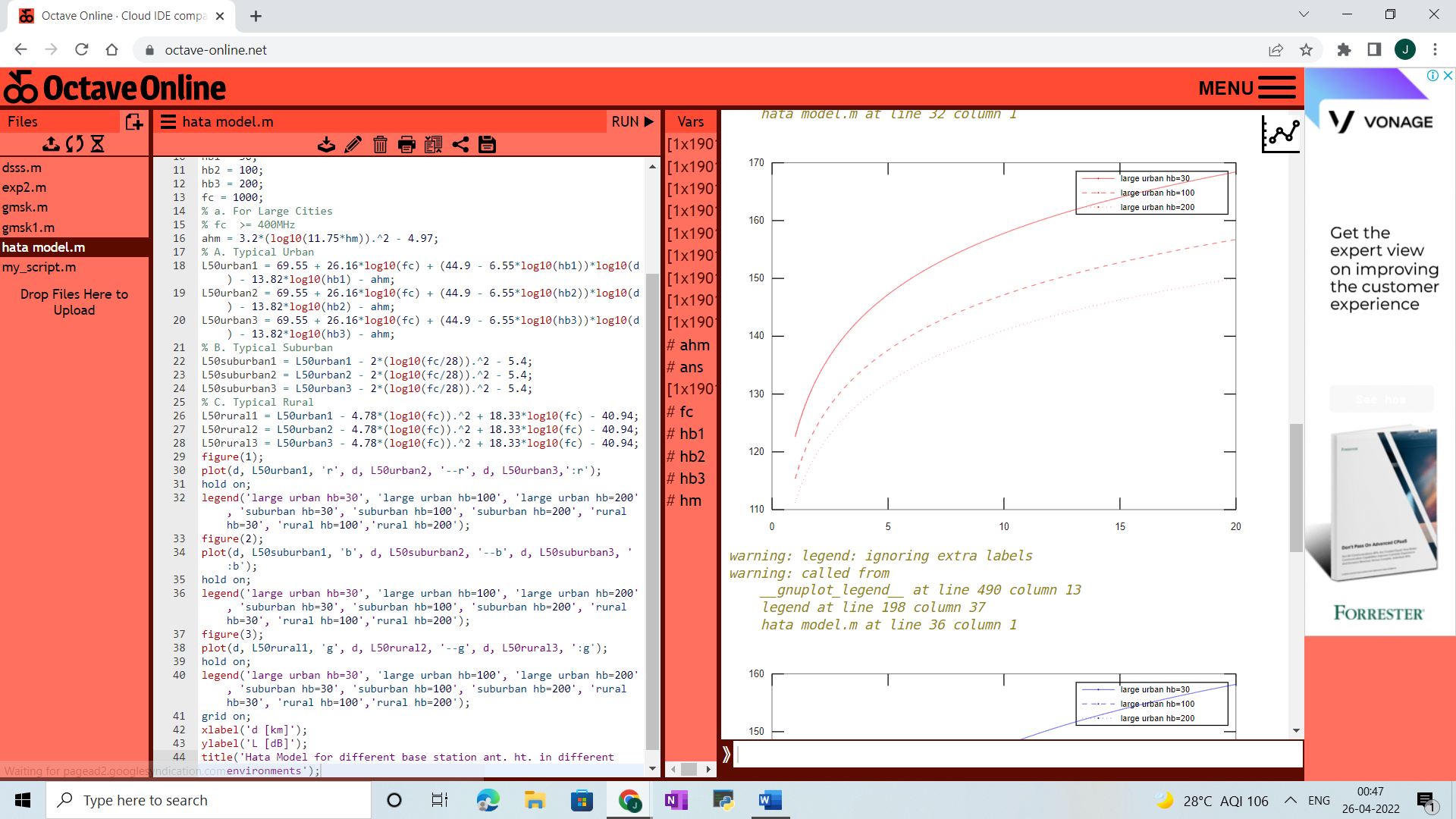
grid on;

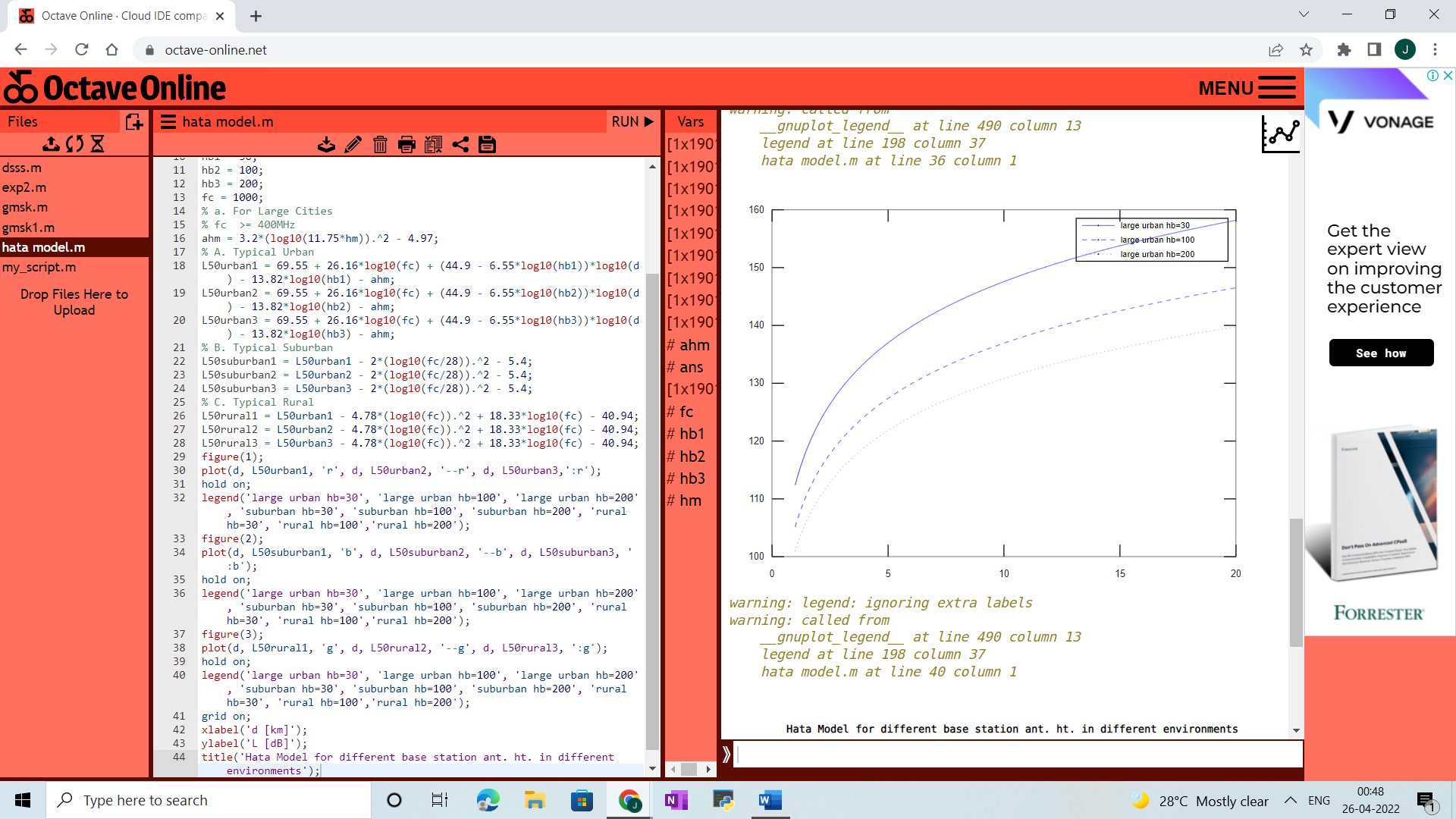
xlabel('d [km]');

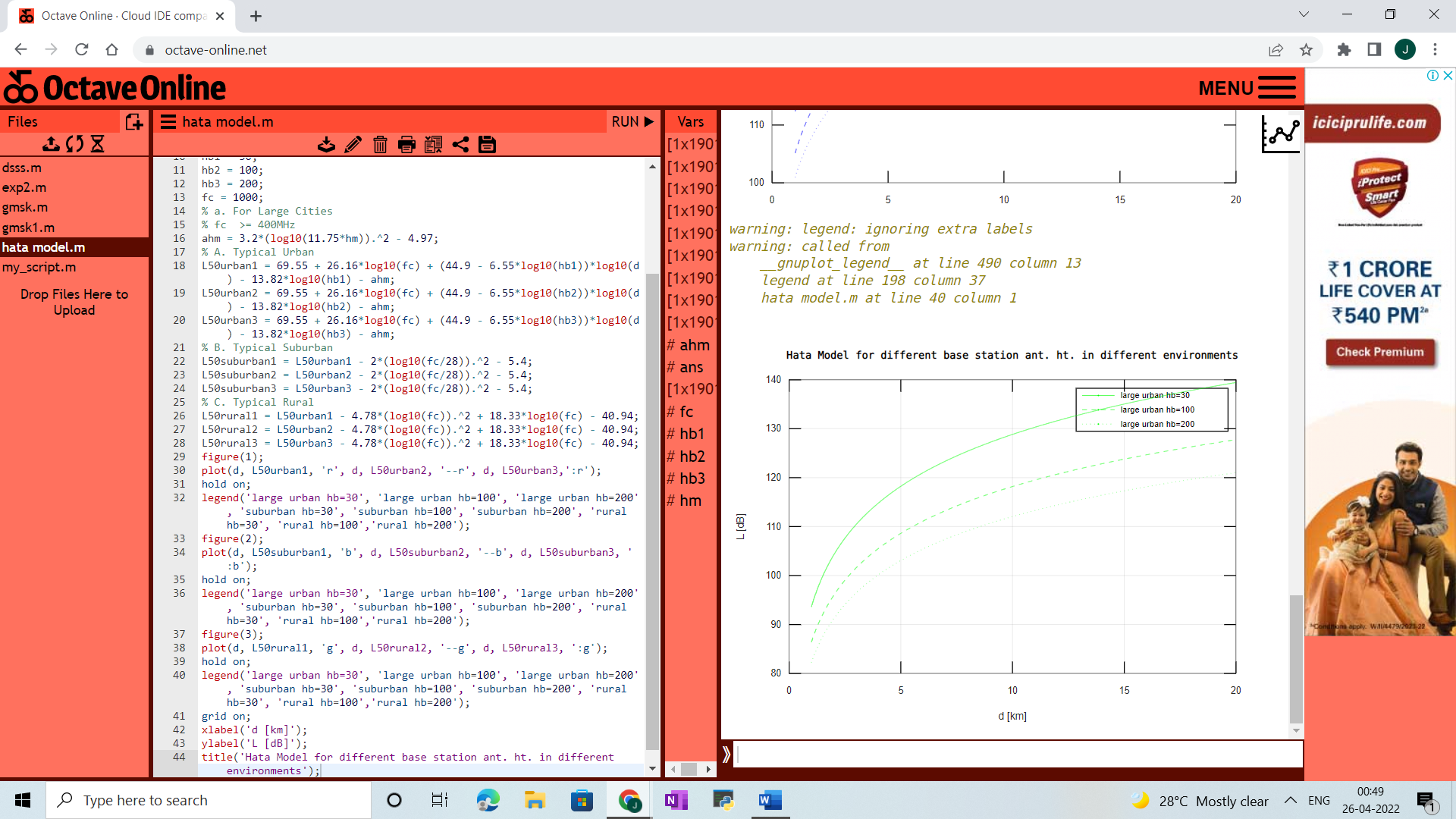
ylabel('L [dB]');

title('Hata Model for different base station ant. ht. in different environments');

**OUTPUT:**







**CODE:**

# LOST CALL SYSTEM

# Roll No.: 42239

%Data and voice traffic analysis in Lost call System

clc;

clear all;

close all;

N=input('Enter the number of trunks');

A=input('Enter the value of ''A');

kk=1;

%GOS will be used for performance analysis

for n=1:N %For N trunks we will have N grade of service

num=power(A,n)/factorial(n);

den=0;

for k=0:n

den=den+power(A,k)/factorial(k);

end

final(kk)=num/den;

kk=kk+1;

end

disp(final);

n=1:N

stem(n,final());

xlabel('Number of trunks');

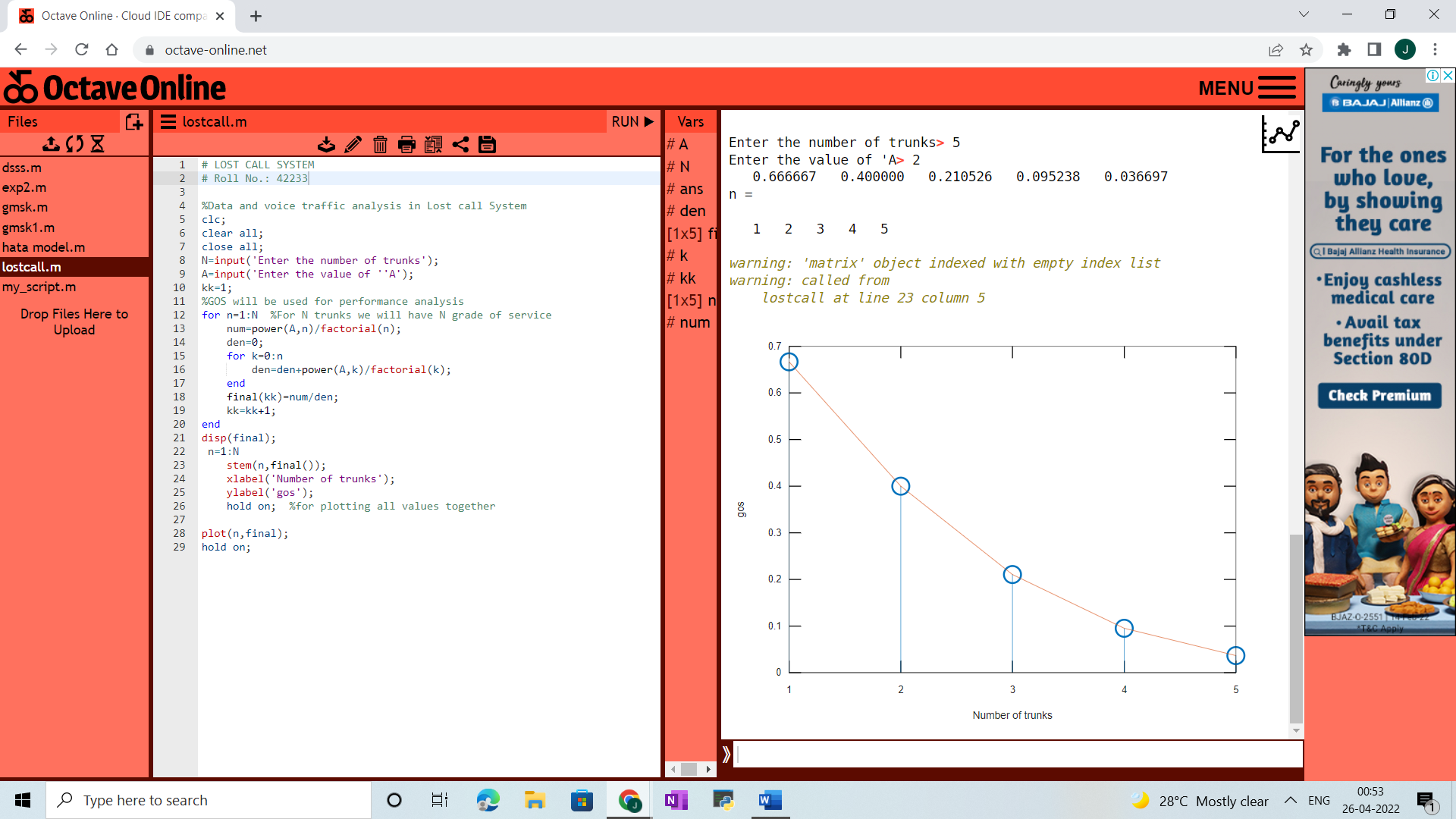
ylabel('gos');

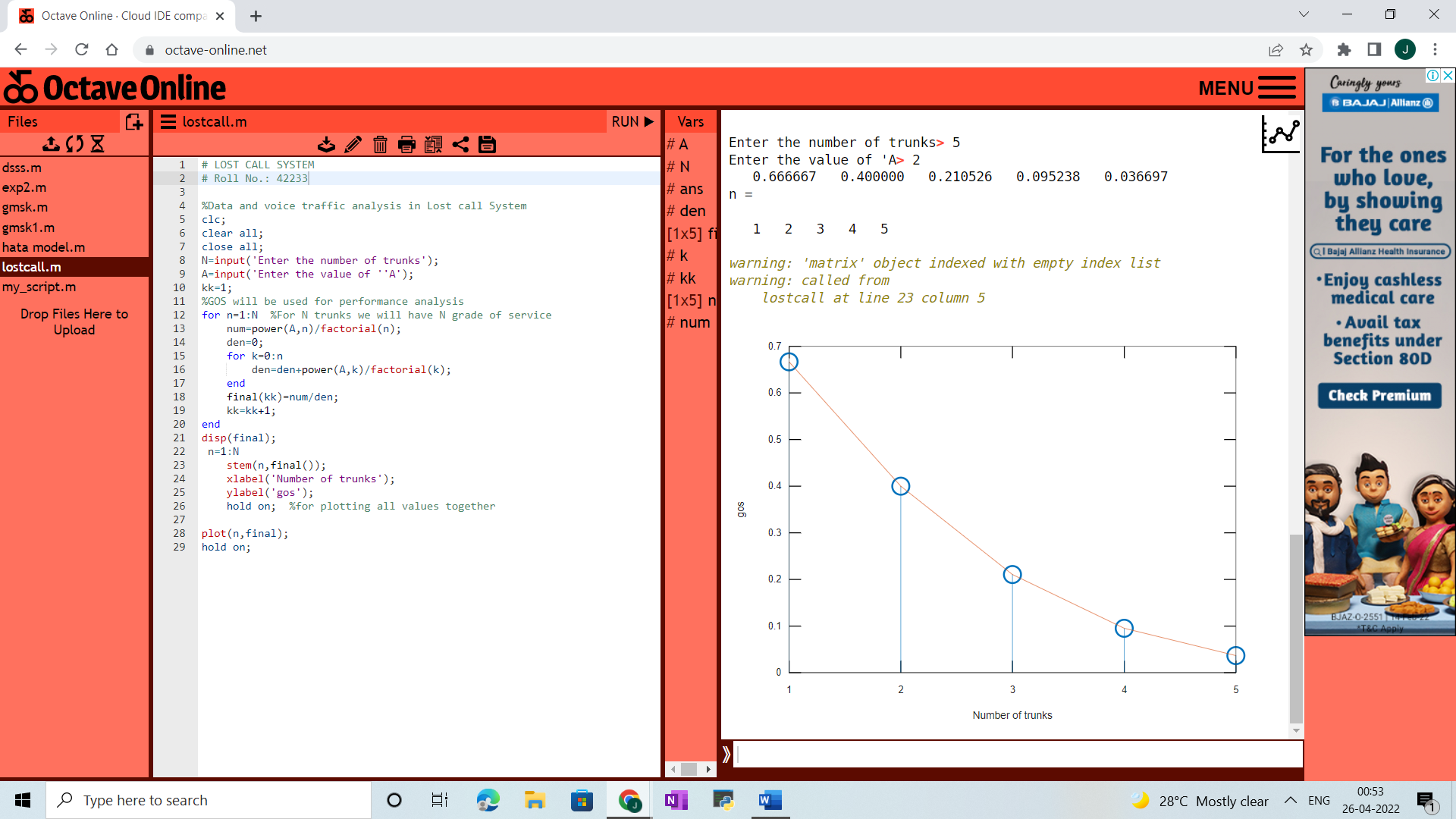
hold on; %for plotting all values together

plot(n,final);

hold on;

**OUTPUT:**





**CODE:**

# Speech Coding And Decoding

# Roll No.: 42239

clc;

close all;

clear all;

n=input('Enter n value for n-bit PCM system : ');

n1=input('Enter number of samples in a period : ');

L=2^n;

% % Signal Generation

% x=0:1/100:4\*pi;

% y=8\*sin(x); % Amplitude Of signal is 8v

% subplot(2,2,1);

% plot(x,y);grid on;

% Sampling Operation

x=0:2\*pi/n1:4\*pi; % n1 nuber of samples have tobe selected

s=8\*sin(x);

subplot(3,1,1);

plot(s);

title('Analog Signal');

ylabel('Amplitude--->');

xlabel('Time--->');

subplot(3,1,2);

stem(s);grid on; title('Sampled Sinal'); ylabel('Amplitude--->'); xlabel('Time--->');

% Quantization Process

vmax=8;

vmin=-vmax;

del=(vmax-vmin)/L;

part=vmin:del:vmax; % level are between vmin and vmax with difference of del

code=vmin-(del/2):del:vmax+(del/2); % Contaion Quantized valuses

[ind,q]=quantiz(s,part,code); % Quantization process

% ind contain index number and q contain quantized values

l1=length(ind);

l2=length(q);

for i=1:l1

if(ind(i)~=0) % To make index as binary decimal so started from 0 to N

ind(i)=ind(i)-1;

end

i=i+1;

end

for i=1:l2

if(q(i)==vmin-(del/2)) % To make quantize value inbetween the levels

q(i)=vmin+(del/2);

end

end

subplot(3,1,3);

stem(q);grid on; % Display the Quantize values

title('Quantized Signal');

ylabel('Amplitude--->');

xlabel('Time--->');

% Encoding Process

figure

code=de2bi(ind,[],'left-msb'); % Convert the decimal to binary

k=1;

for i=1:l1

for j=1:n

coded(k)=code(i,j); % convert code matrix to a coded row vector

j=j+1;

k=k+1;

end

i=i+1;

end

subplot(2,1,1); grid on;

stairs(coded); % Display the encoded signal

axis([0 100 -2 3]); title('Encoded Signal');

ylabel('Amplitude--->');

xlabel('Time--->');

% Demodulation Of PCM signal

qunt=reshape(coded,n,length(coded)/n);

index=bi2de(qunt','left-msb'); % Getback the index in decimal form

q=del\*index+vmin+(del/2); % getback Quantized values

subplot(2,1,2); grid on;

plot(q); % Plot Demodulated signal

title('Demodulated Signal');

ylabel('Amplitude--->');

**OUTPUT:**

