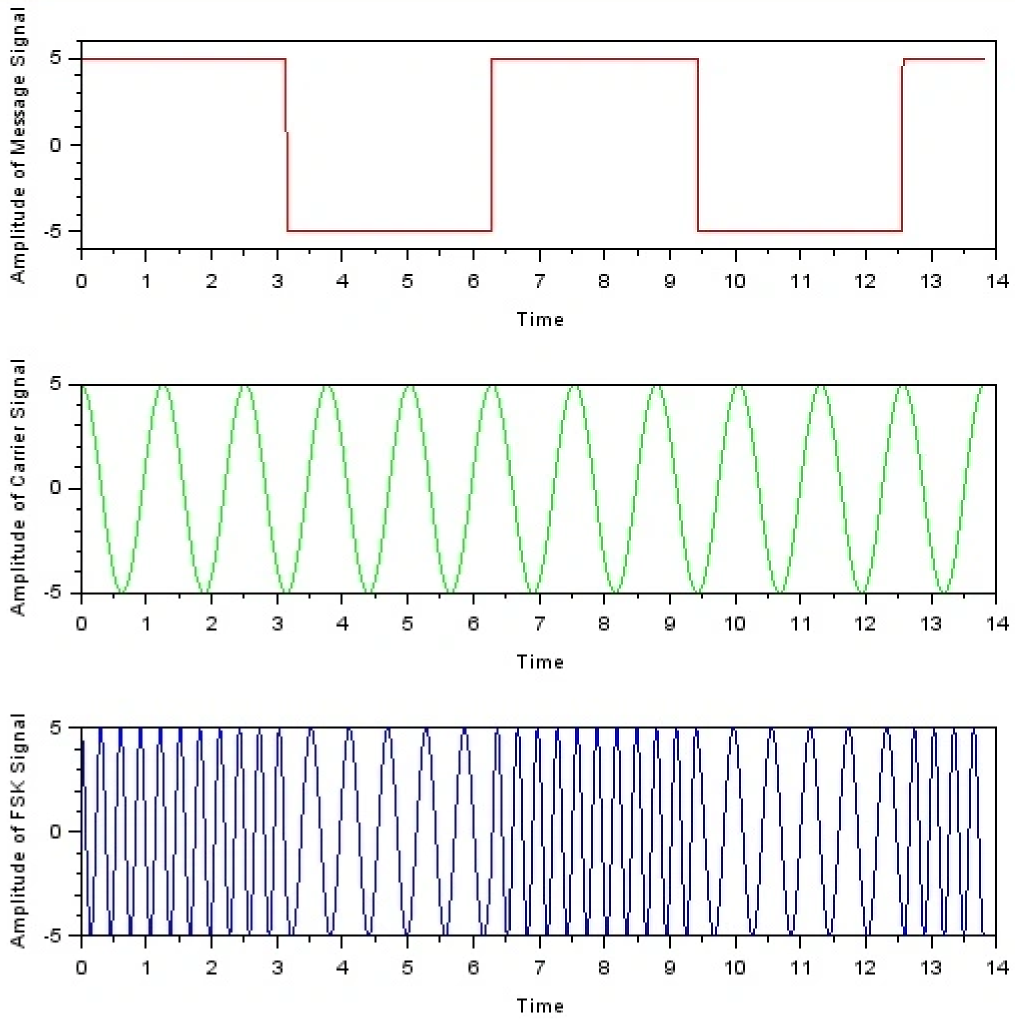
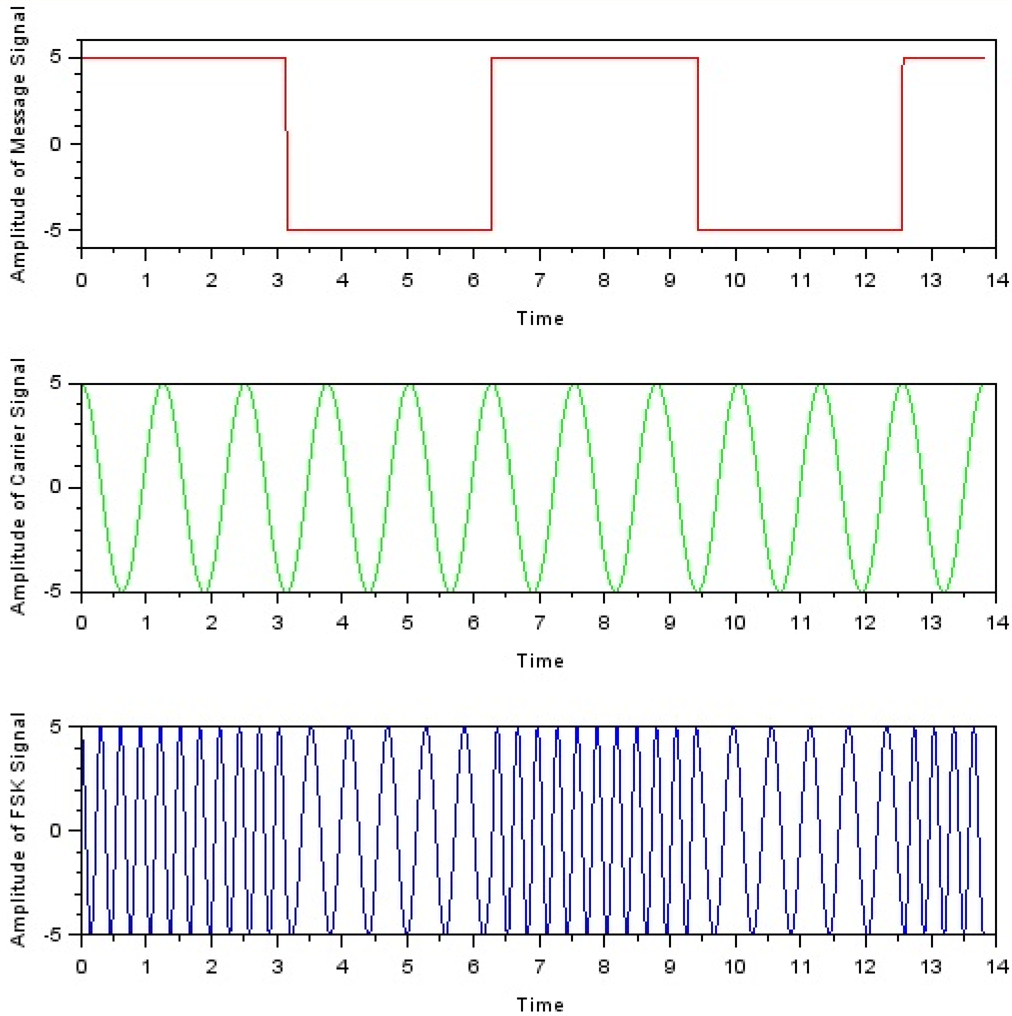
Digital Communication using Scilab



(a) Frequency Shift Keying (FSK) using Scilab

```
clc;clear all;clf;
t=[0:0.01:4.4*%pi];
A=5;
wc=5;

Vm=A.*squarewave(t);
Vc=A.*cos(wc.*t);
fc=wc/(2*%pi);

subplot(3,1,1);
plot(t,Vm, 'red');
xlabel("Time")
ylabel("Amplitude of Message Signal")

subplot(3,1,2);
plot(t,Vc, 'green');
xlabel("Time")
ylabel("Amplitude of Carrier Signal")

fd=0.5; //frequency deviation
subplot(3,1,3);
Vf=A.*cos(2.*%pi.*(fc+Vm.*fd).*t);
plot(t,Vf, 'blue');
xlabel("Time")
ylabel("Amplitude of FSK Signal")
```

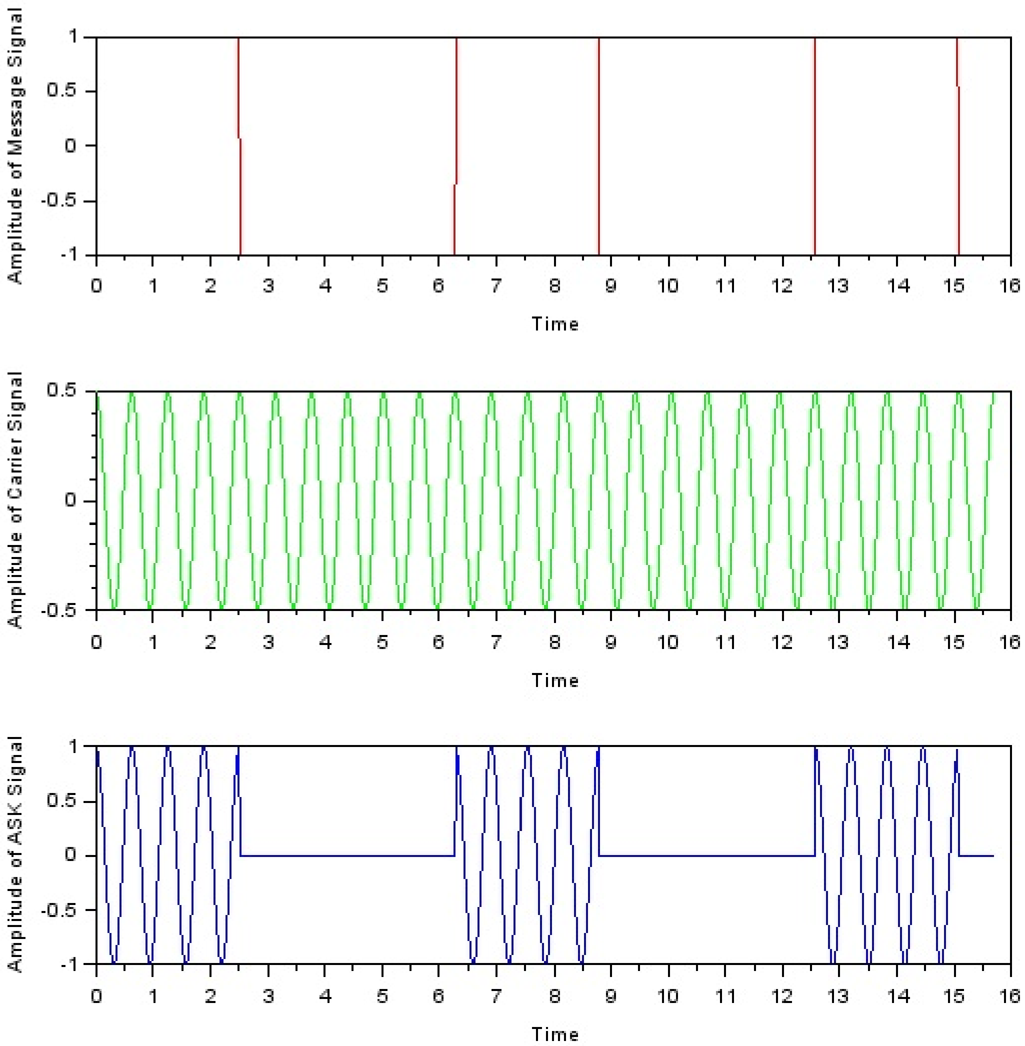
Output:


Amplitude of Message Signal
Time
Amplitude of Carrier Signal
Time
Amplitude of FSK Signal
Time

(b) Amplitude Shift Keying (ASK) using Scilab

```
clc;clear all;clf;
t=[0:0.02:5*%pi];
fc=10;
A=1;

Vm=squarewave(t,40); // The second parameter in the squarewave
                    //function is the percent of the period in
                    //which the signal is positive.

Vc=A/2.*cos(fc.*t);
Va=(1+Vm).*(Vc);

subplot(3,1,1);
plot(t,Vm,'red');
xlabel("Time")
ylabel("Amplitude of Message Signal")

subplot(3,1,2);
plot(t,Vc, 'green');
xlabel("Time")
ylabel("Amplitude of Carrier Signal")

subplot(3,1,3);
plot(t,Va, 'blue');
xlabel("Time")
ylabel("Amplitude of ASK Signal")
```

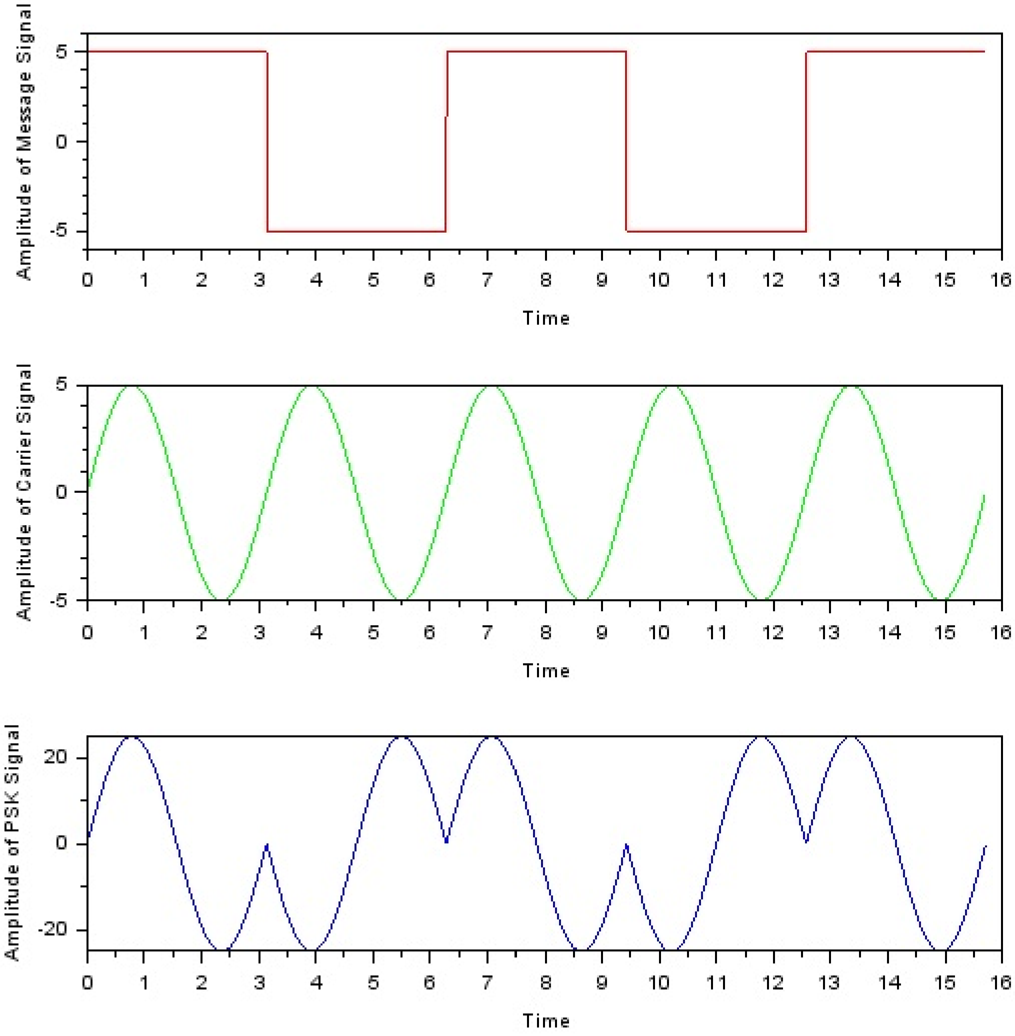
Output:


Amplitude of Message Signal
Time
Amplitude of Carrier Signal
Time
Amplitude of ASK Signal
Time

(c) Phase Shift Keying (PSK) using Scilab

```
clc;clear all;clf;
t=[0:0.01:5*%pi];
A=5;
fc=2;

Vm=A.*squarewave(t);
Vc=A.*sin(fc.*t);
Vp= Vm.*Vc;

subplot(3,1,1);
plot(t,Vm, 'red');
xlabel("Time")
ylabel("Amplitude of Message Signal")

subplot(3,1,2);
plot(t,Vc, 'green');
xlabel("Time")
ylabel("Amplitude of Carrier Signal")

subplot(3,1,3);
plot(t,Vp, 'blue');
xlabel("Time")
ylabel("Amplitude of PSK Signal")
```

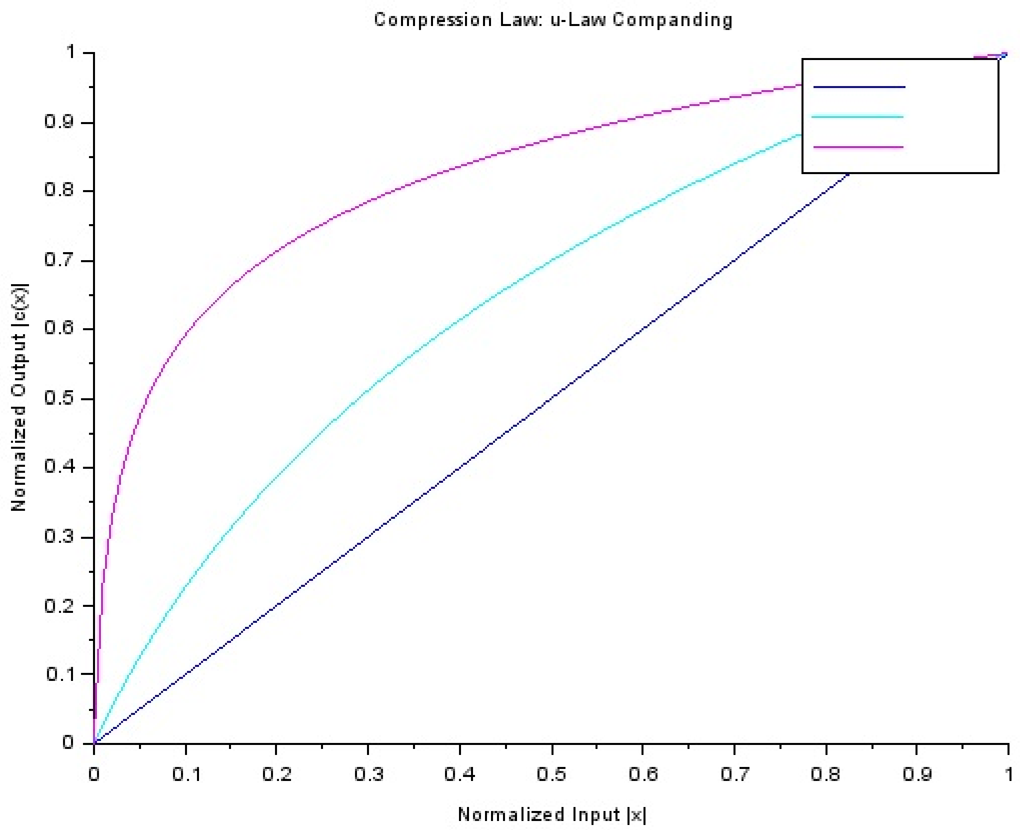
Output:



(d) mu-law companding using Scilab

```
clc;clf;
function [Cx, Xmax]=mulaw(x, mu)
Xmax = max(abs(x));
if(log(1+mu)~=0)
    Cx = (log(1+ mu*abs(x/Xmax )).∕ log(1+ mu));
else
    Cx = x/Xmax ;
end
Cx = Cx/Xmax ; // Normalization of output vector
endfunction
x = 0:0.01:1; // Normalized input
mu = [0,5,255]; // different values of mu
for i = 1:length(mu)
    [Cx(i,:),Xmax(i)] = mulaw (x,mu(i));
end
plot2d (x/Xmax(1),Cx(1,:),2)
plot2d (x/Xmax(2),Cx(2,:),4)
plot2d (x/Xmax(3),Cx(3,:),6)
xtitle ('Compression Law: u-Law Companding','Normalized Input
|x|','Normalized Output |c(x)|');
legend(['u=0'],['u=5'],['u=255']);
```

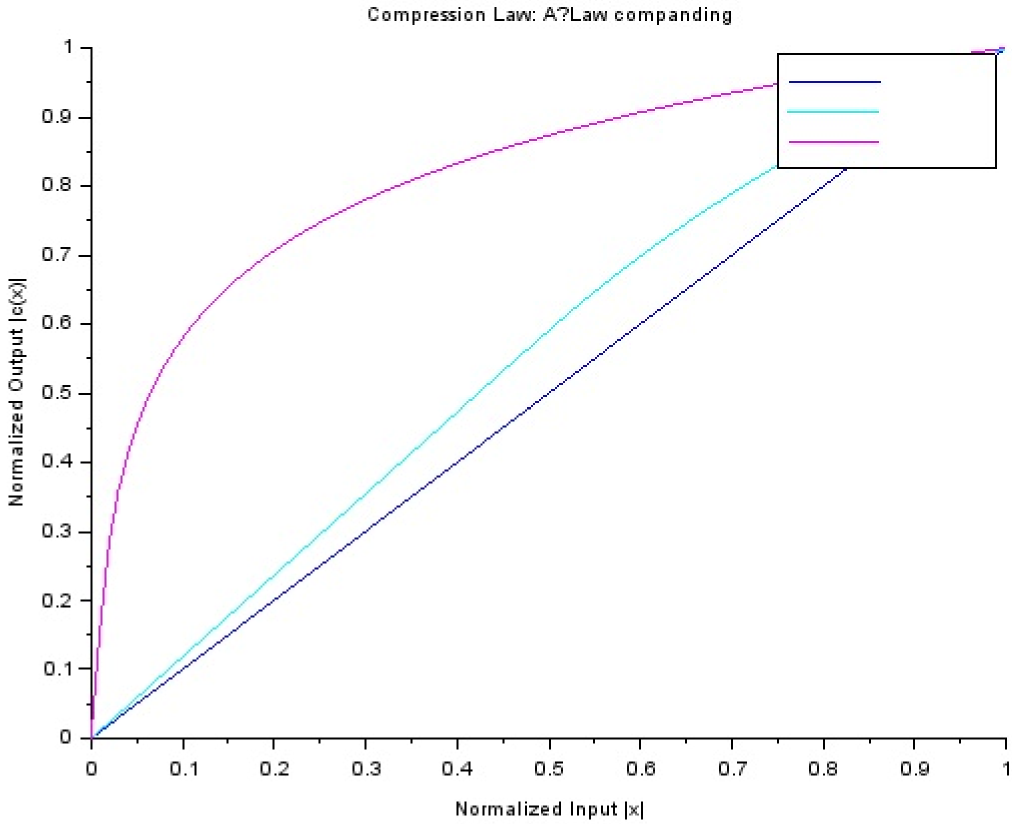
Output:



(e) A-law companding using Scilab

```
clc ;
function [Cx, Xmax]=Alaw(x, A)
Xmax = max(abs(x));
for i = 1:length(x)
    if(x(i)/Xmax <= 1/A)
        Cx(i) = A*abs(x(i)/Xmax)./(1+log(A));
    elseif(x(i)/Xmax > 1/A)
        Cx(i) = (1+log(A*abs(x(i)/Xmax)))./(1+log(A));
    end
end
Cx = Cx / Xmax ;
Cx = Cx';
endfunction
x = 0:0.01:1; // Normalized input
A = [1 ,2 ,87.56]; // Different Values of A
for i = 1:length(A)
    [Cx(i,:),Xmax(i)] = Alaw (x,A(i));
end
plot2d (x/Xmax(1),Cx(1,:),2)
plot2d (x/Xmax(2),Cx(2,:),4)
plot2d (x/Xmax(3),Cx(3,:),6)
xtitle ('Compression Law: A?Law companding','Normalized Input
|x|','Normalized Output |c(x)|');
legend (['A=1'],['A=2'],['A=87.56'])
```

<u>Output:</u>

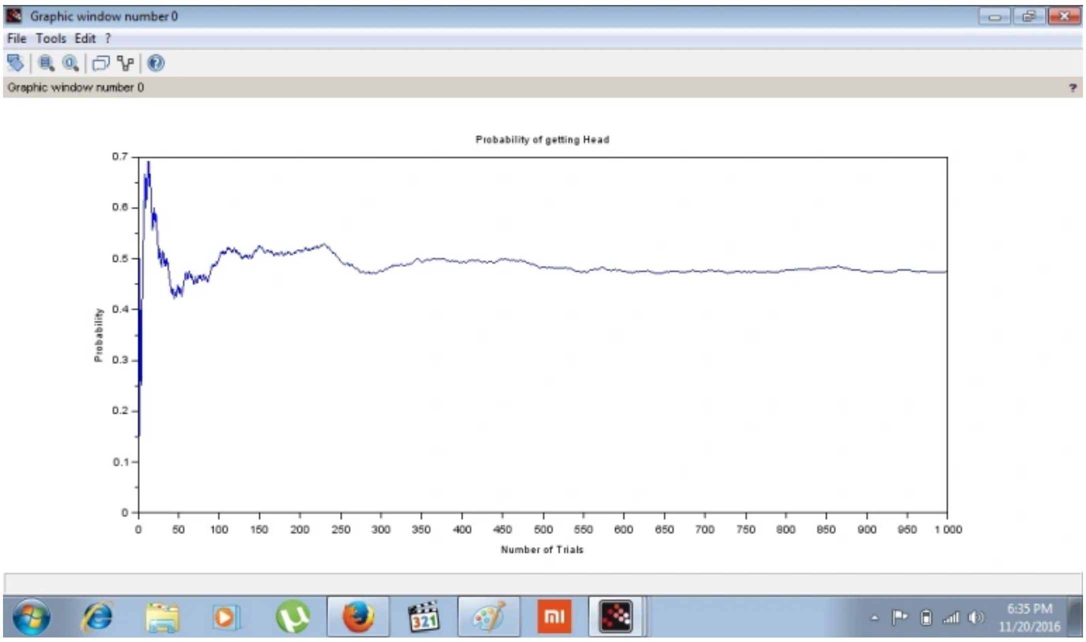


Compression Law: A?Law companding
Normalized Output |c(x)|
Normalized Input |x|
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

(f) Headcount (Probability of Getting Head) using Scilab

```
clc;clear all;
clf ;
a = 1000;
count=0;
for i=1:a;
    x=round(rand(1));     //"round" the elements to nearest integer
                          and "rand" returns a pseudo random scalar
                          value drawn from an uniform distribution of
unit interval
    if (x==1)//Head Count ==1
        count=count+1;    //increment the count value when head occurs
end
p(i)=count/i;             //probability of head occurring at ith
interval
end
plot(1:a,p)               //plot the probability at ith trail(discrete
sequence)
xlabel ("Number of Trials")
ylabel ("Probability")
title ("Probability of getting Head")
```

Output

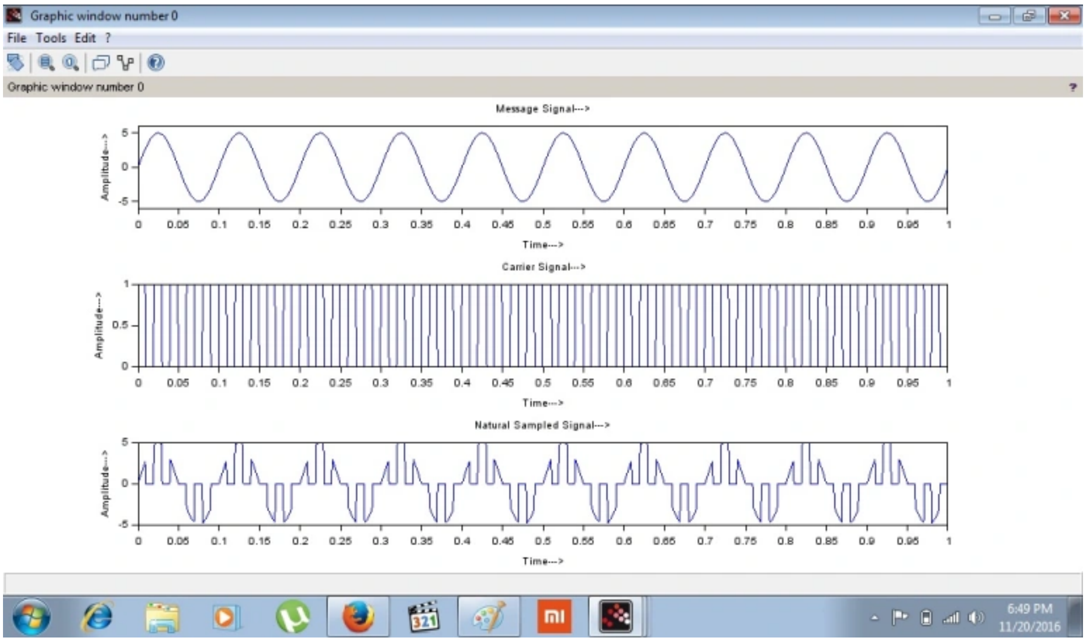


(g) Natural Sampling using Scilab

```
clc;
clear all;
clf;
t=0:0.001:1;
fc=input('Enter the frequency of carrier signal (square wave):');
fm=input('Enter the frequency of message signal (sine wave):');
a=input('Enter the amplitude of message signal:');
vc=squarewave(2*%pi*fc*t);
vm=a*sin(2*%pi*fm*t);
n=length(vc);
for i=1:n
    if (vc(i)<=0)
        vc(i)=0;
    else
        vc(i)=1;
    end
end
y=vc.*vm;

subplot(3,1,1);
plot(t,vm);
xlabel('Time--->');
ylabel('Amplitude--->');
title('Message Signal--->');

subplot(3,1,2);
plot(t,vc);
xlabel('Time--->');
ylabel('Amplitude--->');
title('Carrier Signal--->');

subplot(3,1,3);
plot(t,y);
xlabel('Time--->');
ylabel('Amplitude--->');
title('Natural Sampled Signal--->');
```

Output:

```
Enter the frequency of carrier signal (square wave):50
Enter the frequency of message signal (sine wave):10
Enter the amplitude of message signal:5
```



(h) PCM Modulation Output Signal to Noise Ratio with Bandwidth using Scilab

```
clear all;
clc;
n = input('Enter the number of bits to encode: ')
W = input('Enter the message signal bandwidth: ')
B = n*W;
disp(B,'Channel Width in Hertz: ')
SNRo = 6*n - 7.2;
//SNRo = 4.8 - 6*n;
//SNRo = 1.8 + 6*n;
disp(SNRo ,' Output Signal to Noise Ratio in dB : ')
```

Output:

```
Enter the number of bits to encode: 4
Enter the message signal bandwidth: 4000

 Channel Width in Hertz:

    16000.

  Output Signal to Noise Ratio in dB :

    16.8
```

(The answer can vary depending upon the input values applied)

(i) Output Signal to Noise Ratio of Delta Modulation using Scilab

```
clc;
clear all;
a=input('Enter the amplitude of input signal: ');
fm=input('Enter the modulating frequency in Hz: ');
fs=input('Enter the sampling frequency in samples/second: ');
ts=1/fs;                         //sampling interval
delta=2*%pi*a*fm*ts;             //step size to avoid slope overload
P0max=(a^2)/2;                   //maximum permissable output power
sigma_q=(delta^2)/3;             //quantization error
W=fm;                            //maximum message BW
N=W*ts*sigma_q;                  //average output noise power
SNR0=P0max/N;
SNR_dB=10*log10(SNR0);
disp(SNR_dB,'Maximum Output Signal to Noise Ratio for Delta
Modulation in dB: ');
```

Output:

```
Enter the amplitude of input signal: 5
Enter the modulating frequency in Hz: 2000
Enter the sampling frequency in samples/second: 10000

Maximum Output Signal to Noise Ratio for Delta Modulation in dB:
6.7664154
```

(The answer can vary depending upon the input values applied)

(j) Hamming Distance (error detecting technique) using Scilab

```
clc ;
clear all;
// Getting Code Words
code1 = input ('Enter the 1st Code Word ') ;
code2 = input ('Enter the 2nd Code Word ') ;
Hamming_Distance = 0;
for i = 1:length (code1)
 Hamming_Distance = Hamming_Distance + bitxor(code1(i),code2(i)) ;
end
disp (Hamming_Distance, 'Hamming Distance')
```

Output:

```
Enter the 1st Code Word [1 1 1 0 0 0 1 0]
Enter the 2nd Code Word [0 0 1 0 1 0 0 1]
Hamming Distance
5
```

(The answer can vary depending upon the input values applied)

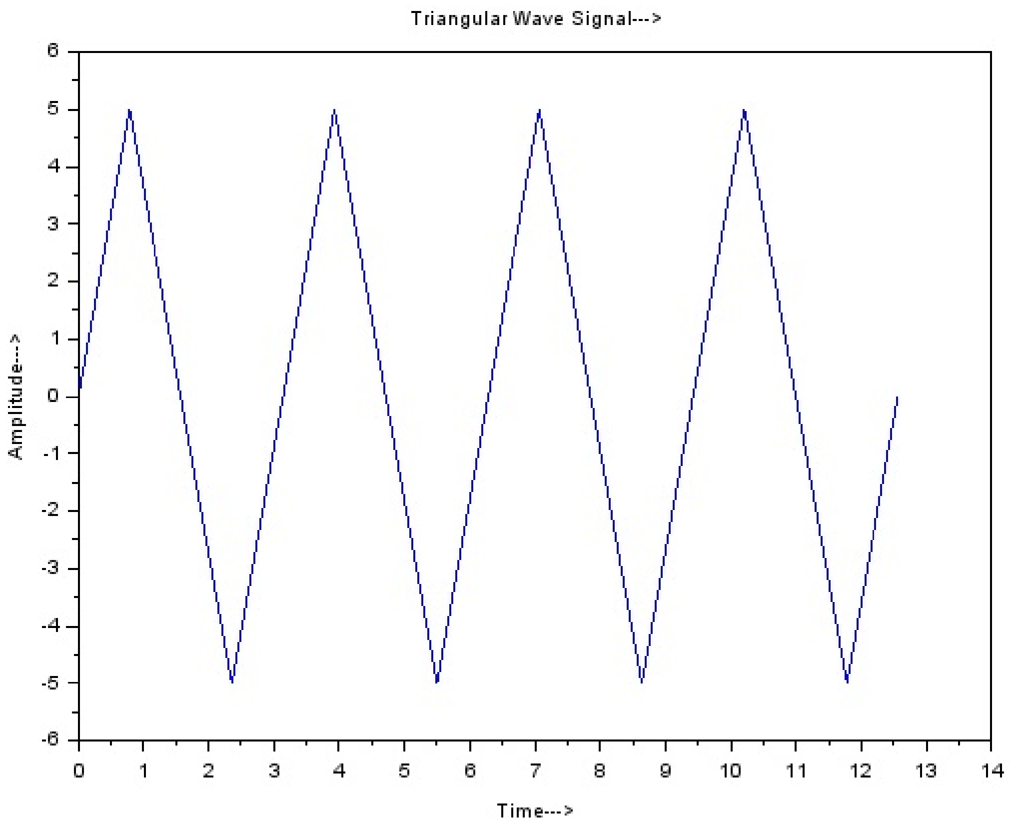
(k) Some Basic Functions using Scilab

(i) Triangular Pulse

```
clc ;clf ;clear all;
a =input('Enter the amplitude of message signal:');
t =0:( %pi /4) :(4* %pi );
y = a *sin (2* t );
plot (t,y);
xlabel('Time--->');
ylabel('Amplitude--->');
title('Triangular Wave Signal--->');
```

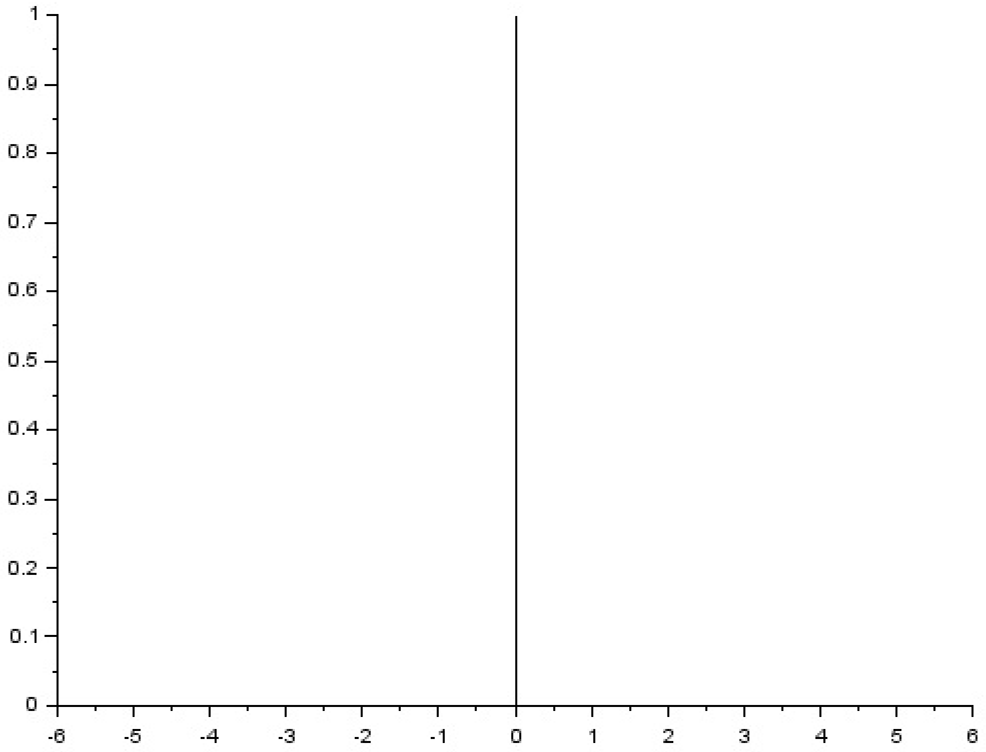
Output:

Enter the amplitude of message signal: 5 (can change the value)



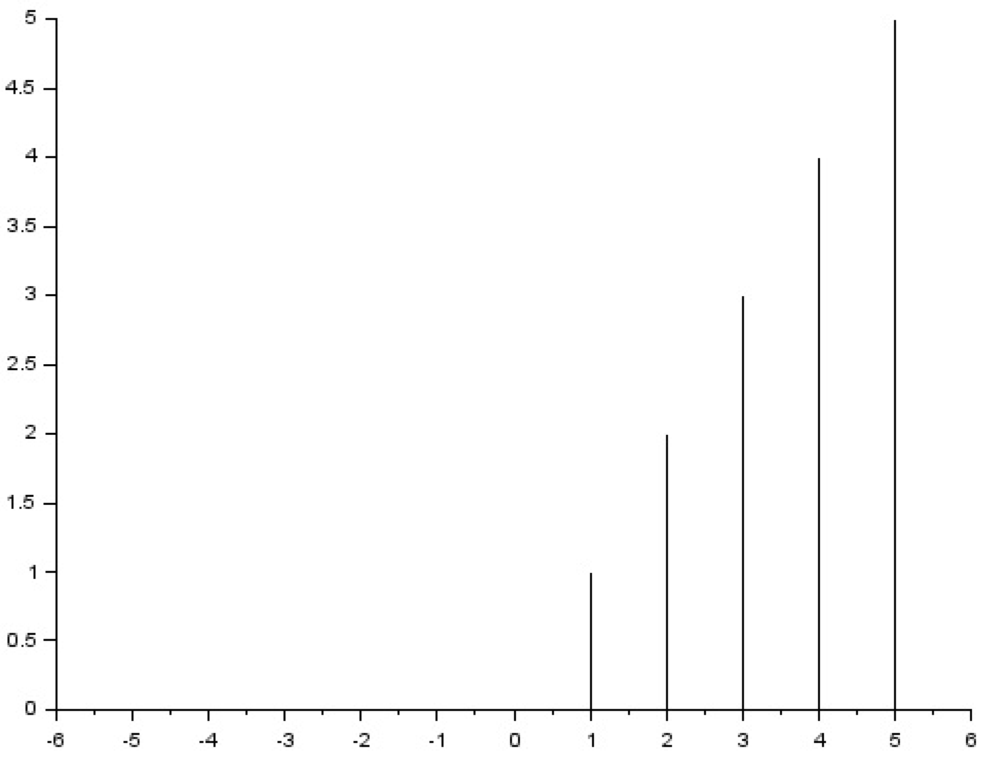
(ii) Impulse Signal

```
l=5;
n=-l:l;
x=[zeros(1,l),ones(1,1),zeros(1,l)];
plot2d3(n,x)
```



(iii) Ramp Signal

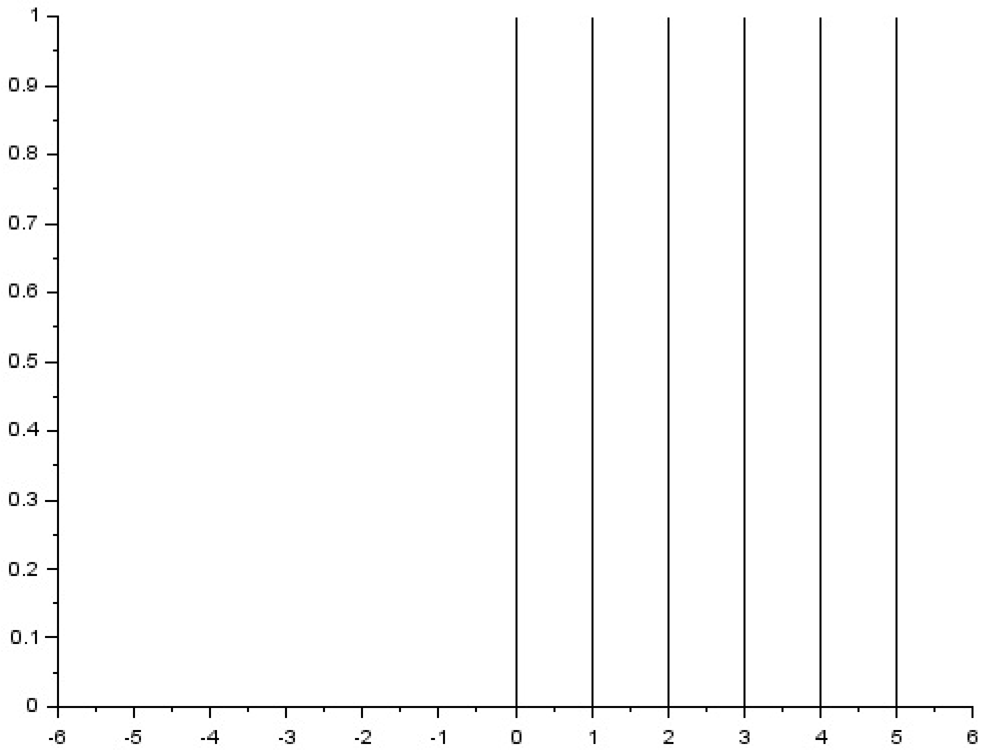
```
l=5;
n=-l:l;
x=[zeros(1,l),0:5]
plot2d3(n,x)
```



(iv) Step Signal

```
clc;clear all;clf;
L=5;
n=-L:L;
```

```
x=[zeros(1,L),ones(1,L+1)]
plot2d3(n,x)
```



(v) Exponential Signal

```
clc;clear all;clf;
a=-2:0.1:2;
b=exp(a);
plot2d3(a,b);
```