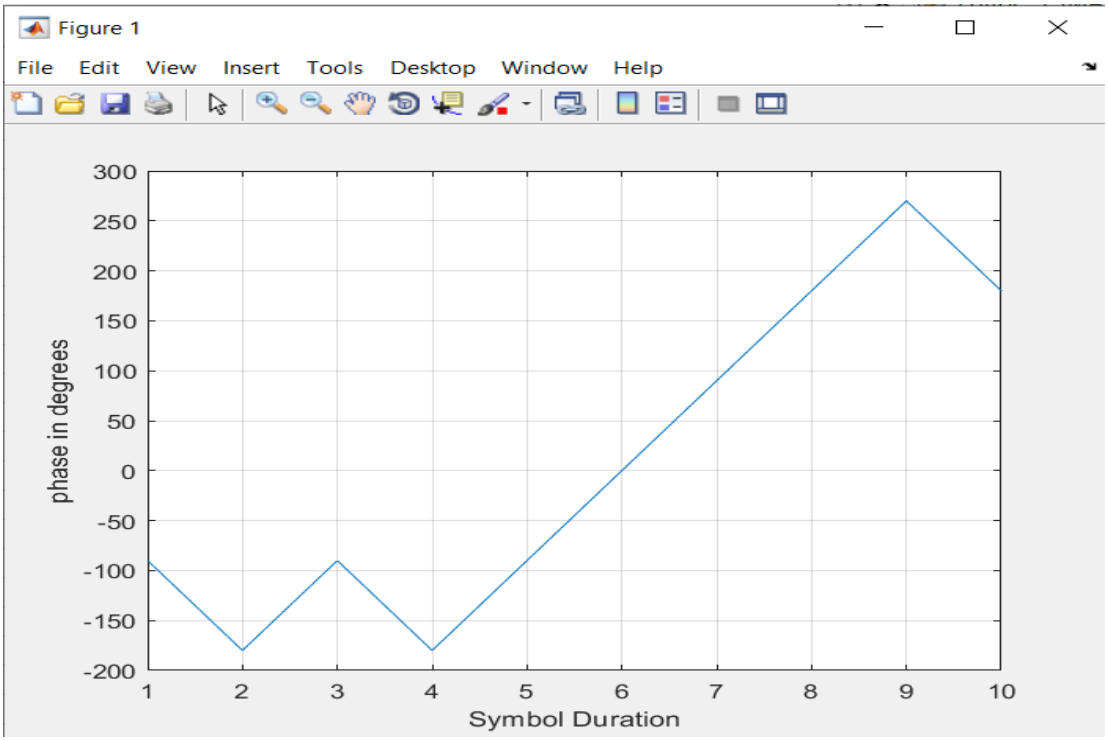
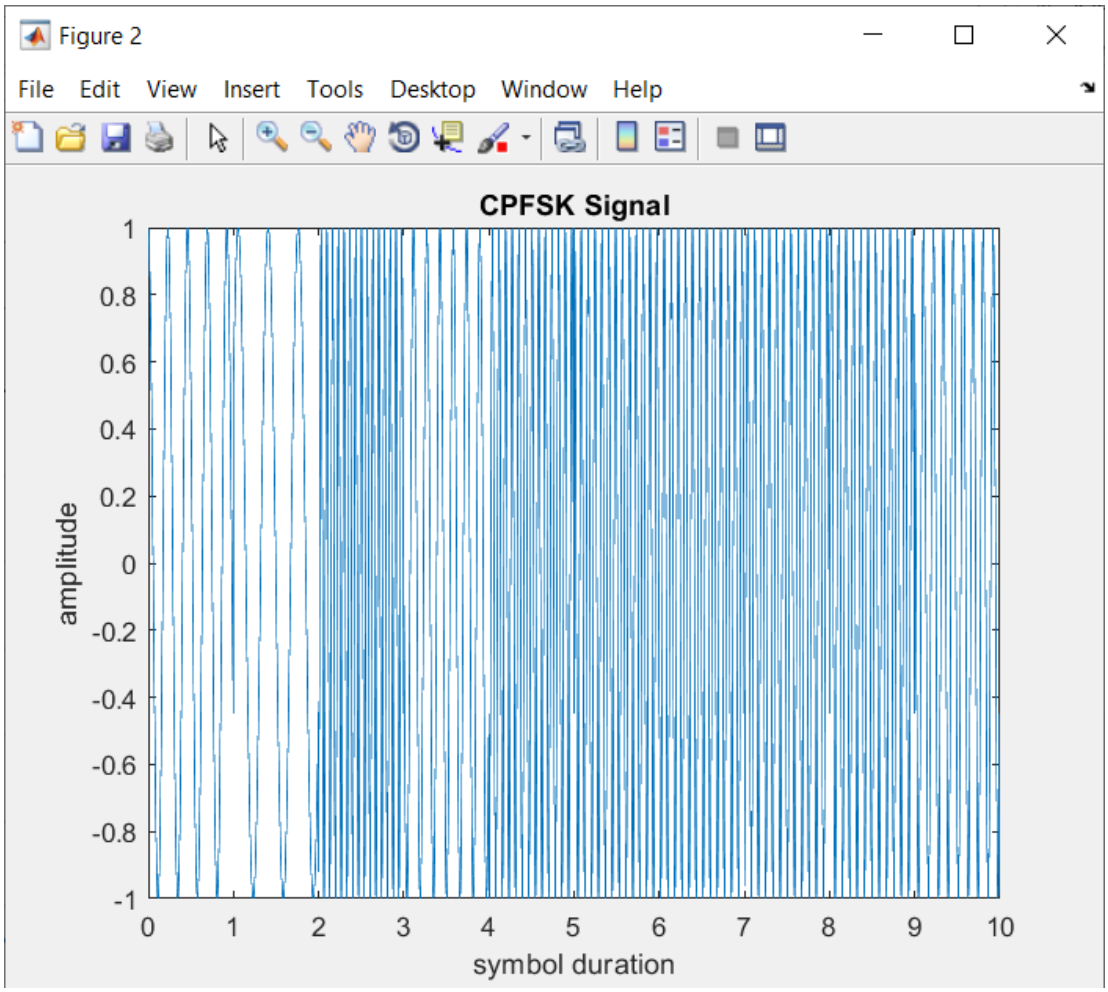
OUTPUTS: for input = [-1,-1, 1, -1, 1, 1, 1, 1, 1, -1]





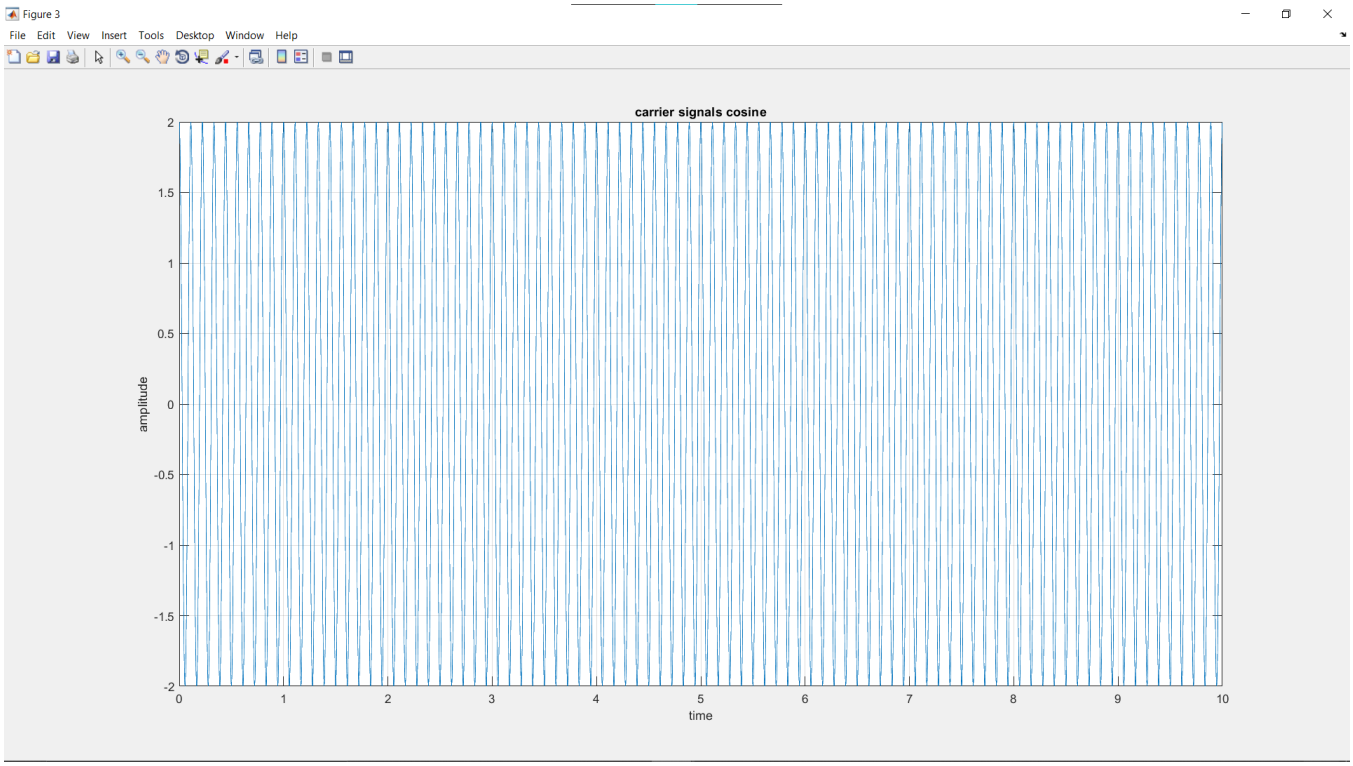


Figure 3
File Edit View Insert Tools Desktop Window Help
carrier signals cosine
amplitude
time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5
6
7
8
9
10

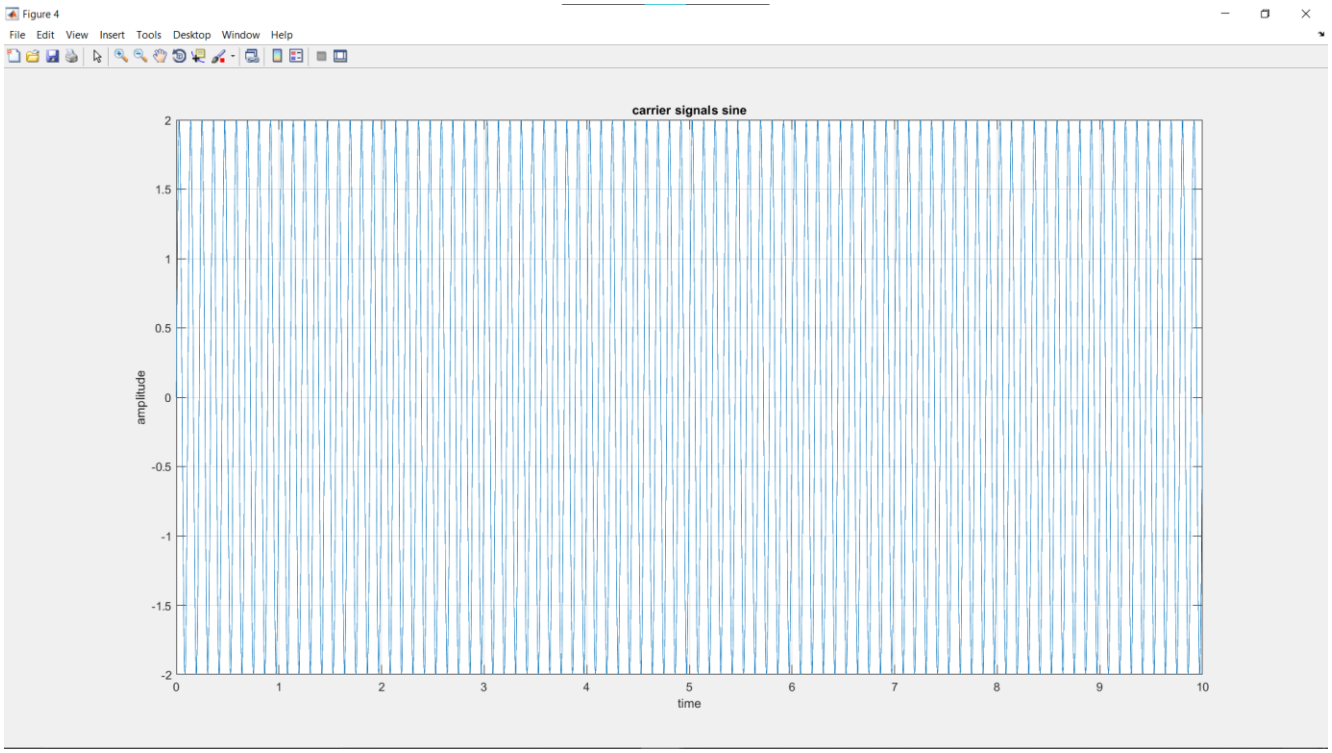


Figure 4
File Edit View Insert Tools Desktop Window Help
carrier signals sine
amplitude
time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5
6
7
8
9
10

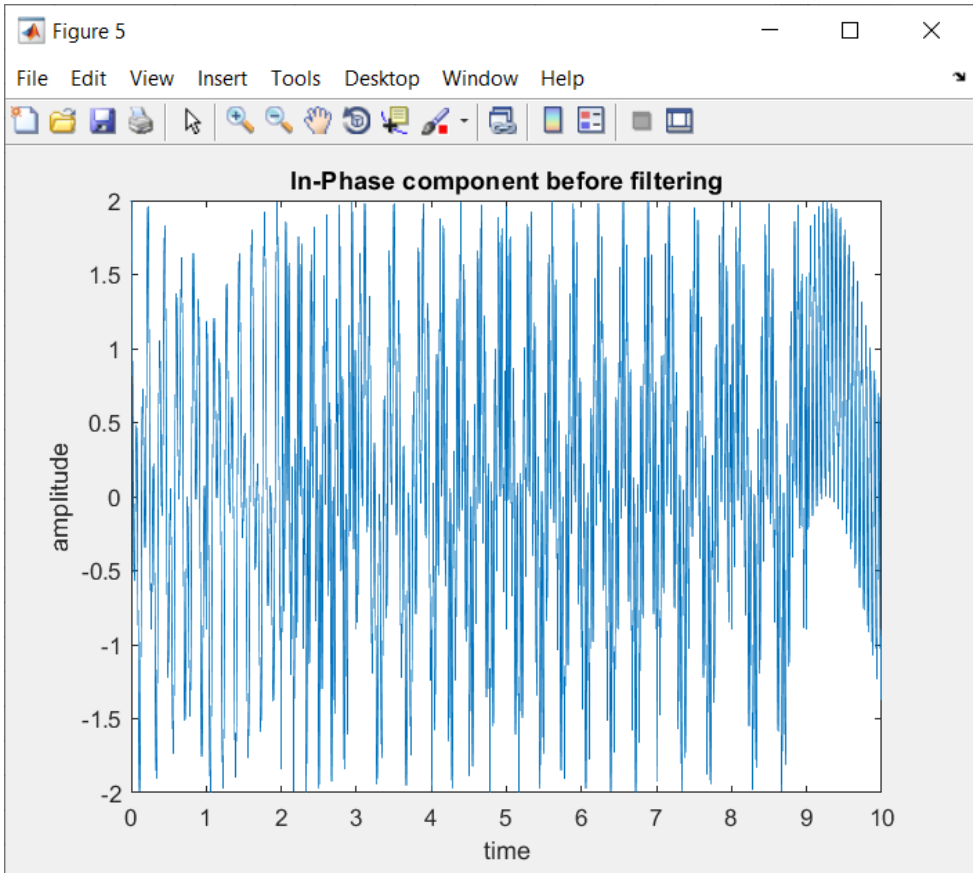


Figure 5
File Edit View Insert Tools Desktop Window Help
In-Phase component before filtering
amplitude
time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 1 2 3 4 5 6 7 8 9 10

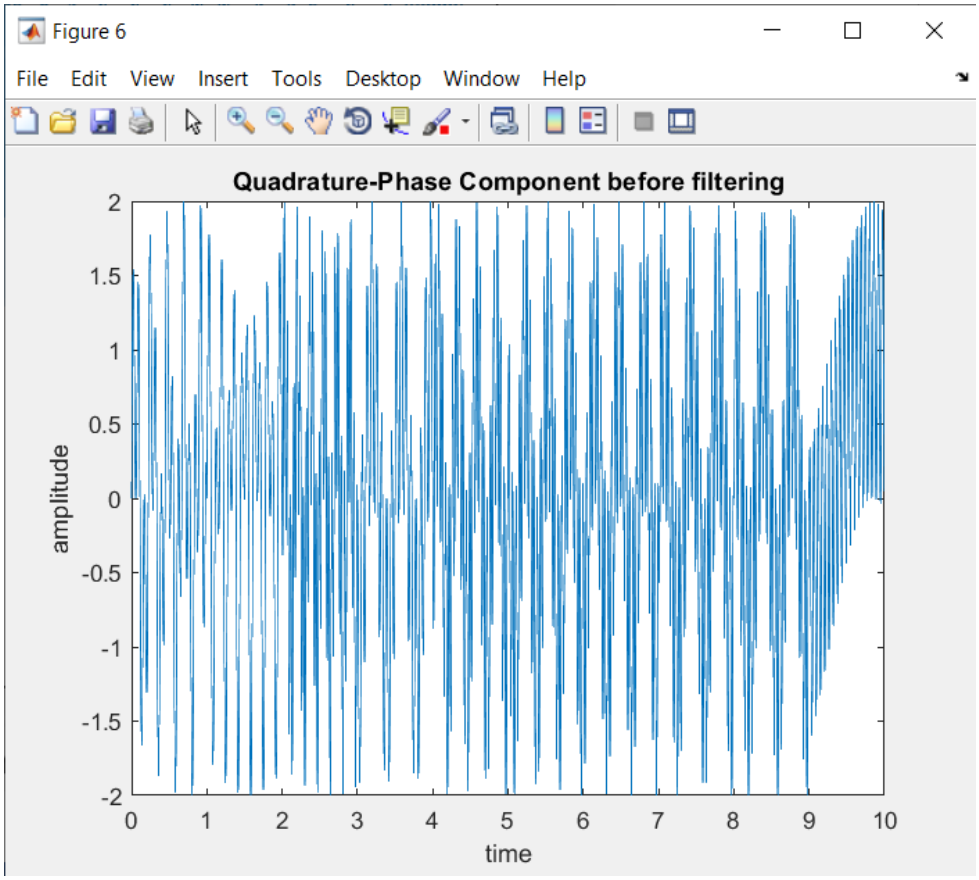


Figure 6
File Edit View Insert Tools Desktop Window Help
Quadrature-Phase Component before filtering
amplitude
time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 1 2 3 4 5 6 7 8 9 10

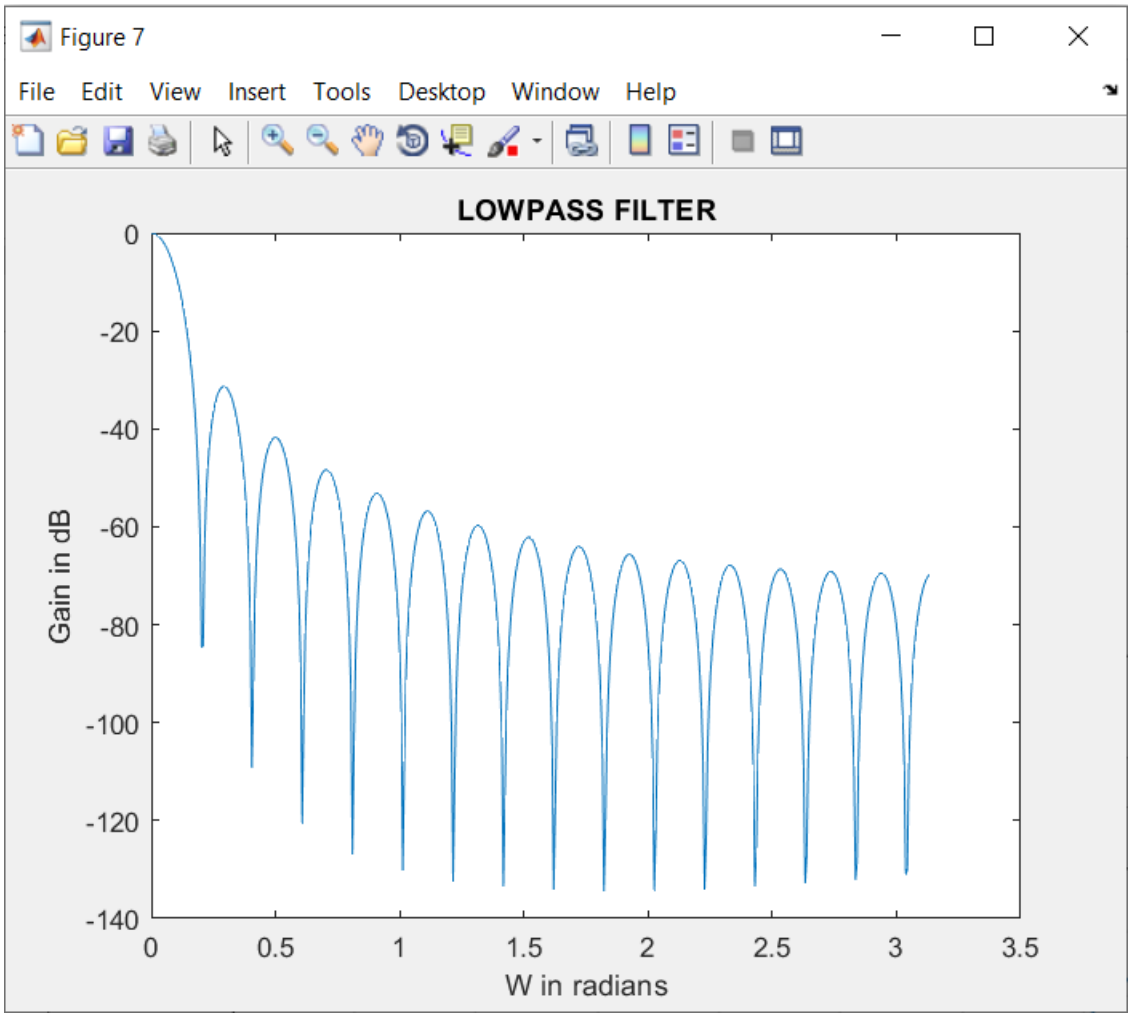


Figure 7
File Edit View Insert Tools Desktop Window Help
LOWPASS FILTER
Gain in dB
0
-20
-40
-60
-80
-100
-120
-140
0
0.5
1
1.5
2
2.5
3
3.5
W in radians

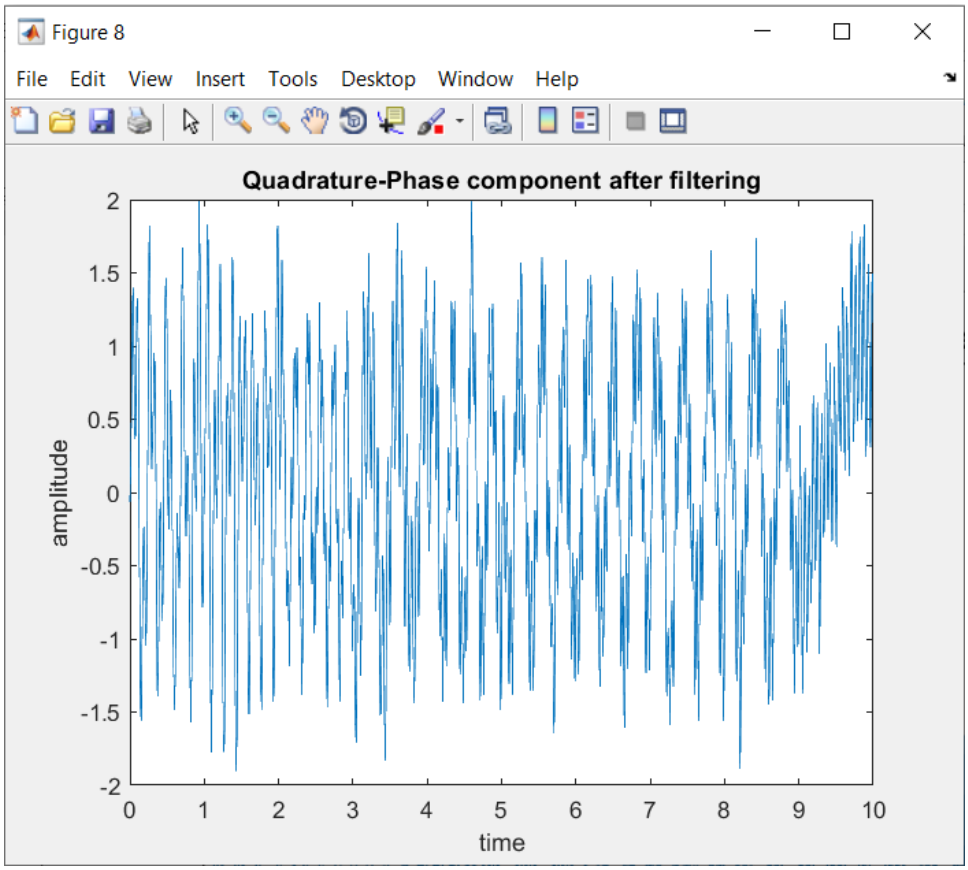


Figure 8
File Edit View Insert Tools Desktop Window Help
Quadrature-Phase component after filtering
amplitude
time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 1 2 3 4 5 6 7 8 9 10

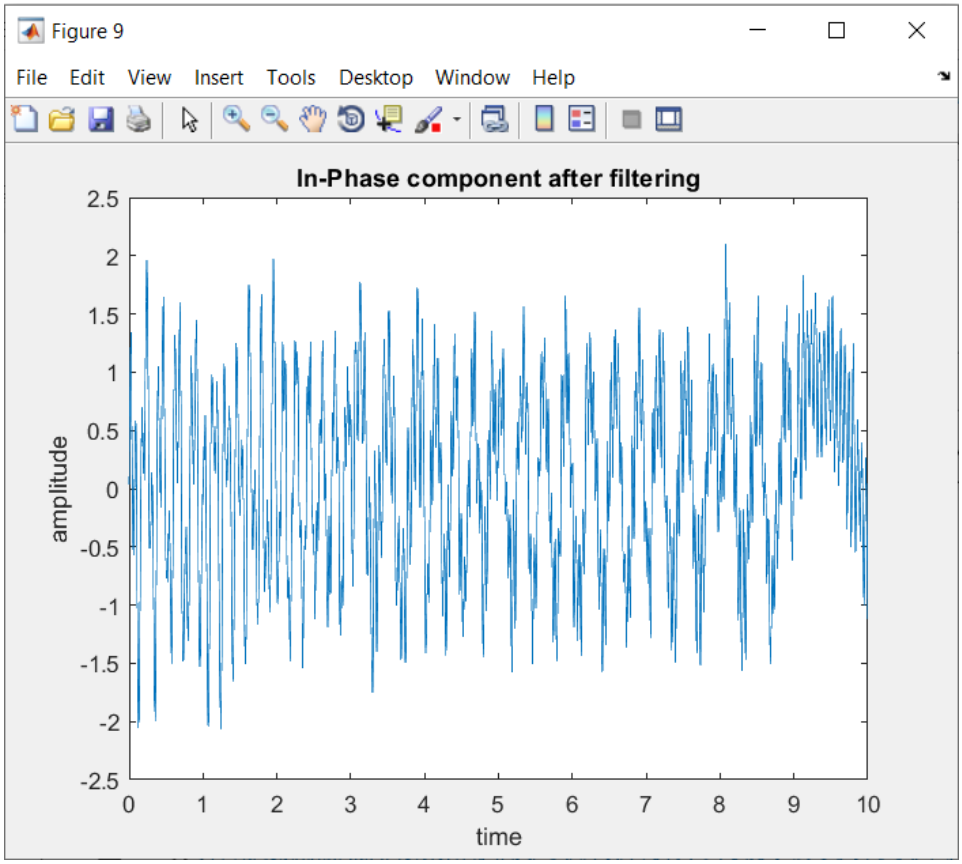


Figure 9
File Edit View Insert Tools Desktop Window Help
In-Phase component after filtering
amplitude
time
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0 1 2 3 4 5 6 7 8 9 10

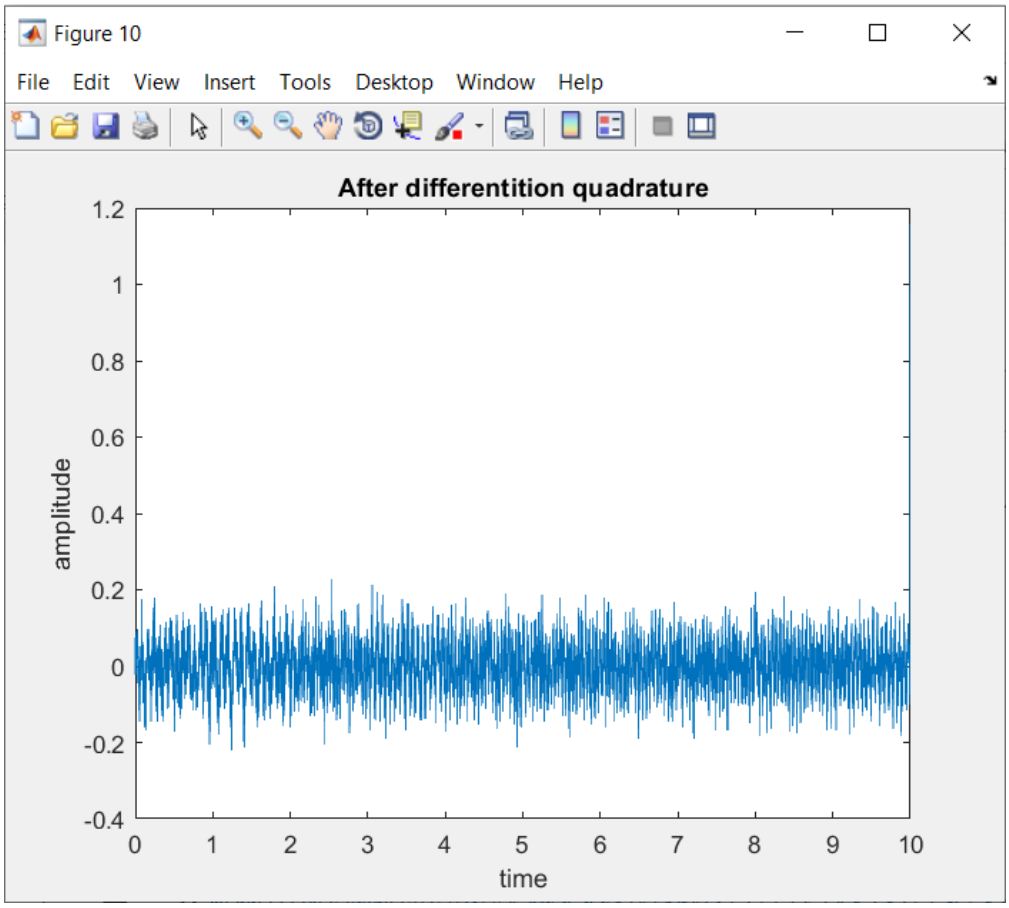


Figure 10
File Edit View Insert Tools Desktop Window Help
After differentition quadrature
amplitude
time
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0 1 2 3 4 5 6 7 8 9 10

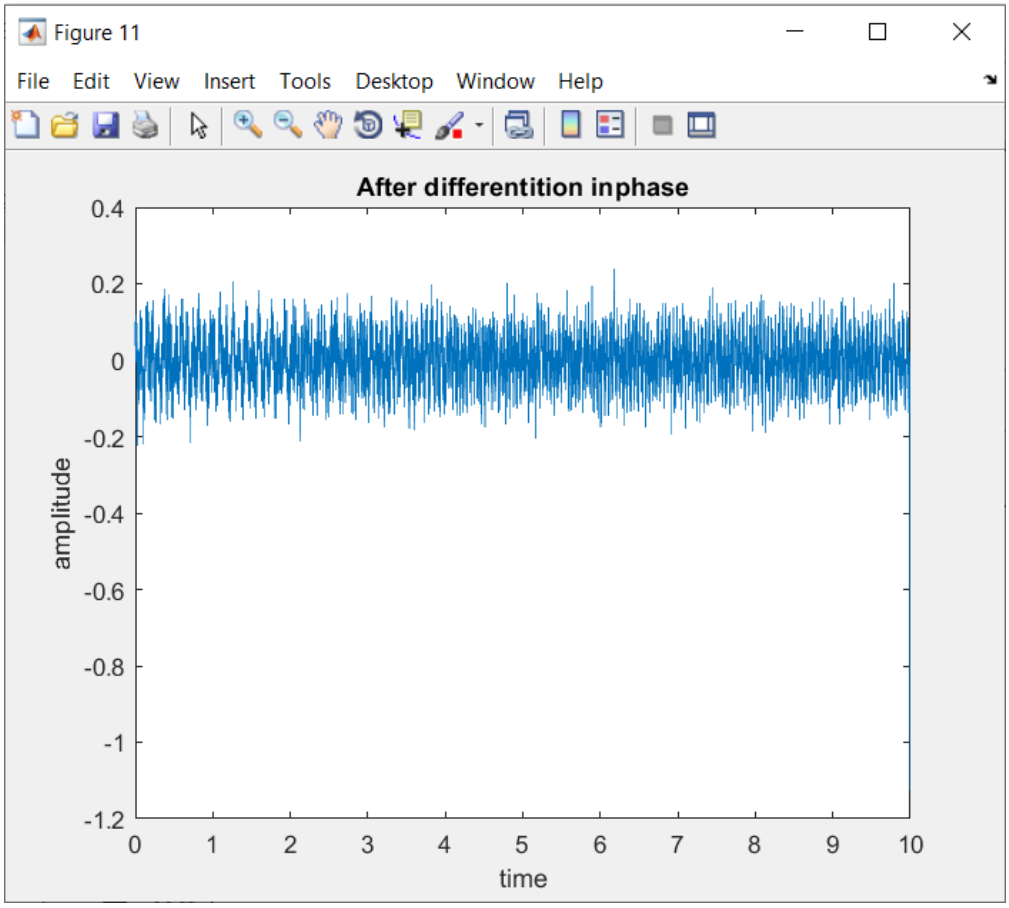


Figure 11
File Edit View Insert Tools Desktop Window Help
After differentition inphase
amplitude
time
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0 1 2 3 4 5 6 7 8 9 10

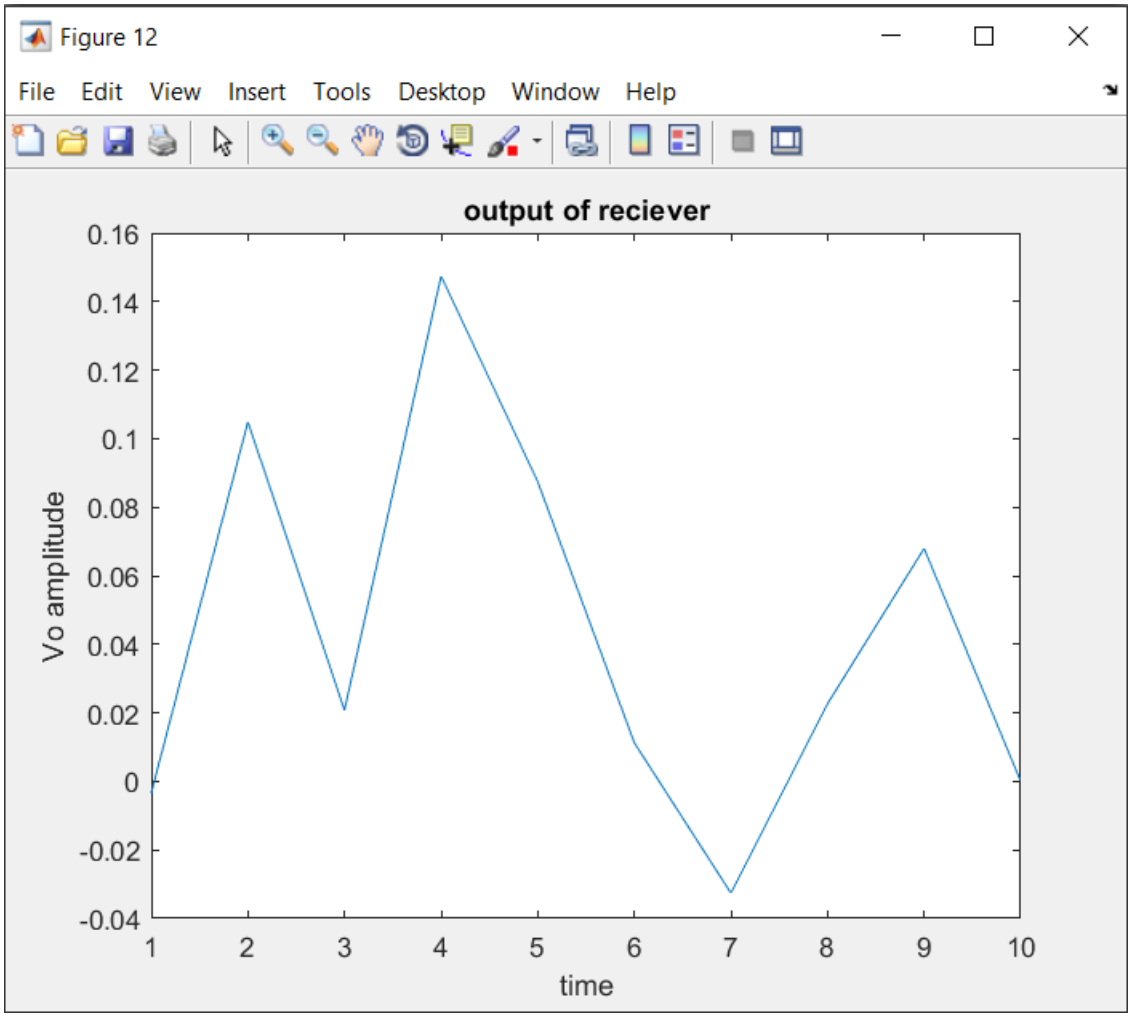


Figure 12
File Edit View Insert Tools Desktop Window Help
output of reciever
Vo amplitude
time
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
1
2
3
4
5
6
7
8
9
10

MATLAB Code:

```
clc
fs = 1e3;
T = 15;
fc = 10;
t = 0:1/fs:T-1/fs;
sim_bits = 15;
mixingfreq = 9;
h = 0.5;
x = randi([0 1],sim_bits,1);
mapper =  mapping(x);
phasetrajectory = tree(mapper,0.5);
phaseshift = 0;
figure(1);
plot(phasetrajectory);
xlabel('Symbol Duration');
ylabel('phase in degrees');
grid on;
k = 1;
q = zeros(1,15000);


for i = 1:1:T
    if(mapper(i)==1)
      phaseshift = 180*h;
    end
    if(mapper(i)==-1)
        phaseshift = -180*h;
    end
    for t1=0.001+i-1:1/fs:i
        phase1(i)=phaseshift;
        s = cos(2*pi*fc*t1+(phaseshift*t1)/i);
        q(k) = s;
        k = k+1;
    end
end
figure(2);
plot(t,q);
xlabel('symbol duration');
ylabel('amplitude');
title('CPFSK Signal');

%Demodulation/Rx

[in_phase,quadrature_phase] = mixer(q,mixingfreq,T,fs);
figure(5);
plot(t,in_phase);
```

```
%subplot(2,1,1);
xlabel('time');
ylabel('amplitude');
title('In-Phase component before filtering');
%subplot(2,1,2);
figure(6);
plot(t,quadrature_phase);
xlabel('time');
ylabel('amplitude');
title('Quadrature-Phase Component before filtering');

% %Addition of noise
% quadrature_phase = quadrature_phase+randn(1,10000);
% in_phase = in_phase+randn(1,10000);

quadrature_phase_filter = filter_1(quadrature_phase);
figure(8);
plot(t,quadrature_phase_filter);
xlabel('time');
ylabel('amplitude');
title('Quadrature-Phase component after filtering');
in_phase_filter = filter_1(in_phase);
figure(9);
plot(t,in_phase_filter);
xlabel('time');
ylabel('amplitude');
title('In-Phase component after filtering');

quad_diff = fofa(quadrature_phase_filter);
in_diff = fofa(in_phase_filter);
figure(10);
plot(t,quad_diff);
xlabel('time');
ylabel('amplitude');
title('After differentition quadrature');
figure(11);
plot(t,in_diff);
xlabel('time');
ylabel('amplitude');
title('After differentition inphase');

oneterm = in_diff.*quadrature_phase_filter;
twoterm = quad_diff.*in_phase_filter;
summing = twoterm-oneterm;
rx_output = sampling(summing,T);
figure(12);
plot(rx_output);
```

```
xlabel('time');
ylabel('amplitude');
title('output of reciever');
Vavg = mean(rx_output)
function mapper_out = mapping(input_signal)
    n = length(input_signal);
    mapper_out = zeros(1,n);
    for i = 1:1:n
       if(input_signal(i)==1)
           mapper_out(i) = 1;
       else
           mapper_out(i) = -1;
       end
    end
end

function phase = tree(input,h)
temp = 0;
for i = 1:1:length(input)
   if(input(i)==1)
       temp = temp+h*180;
   else
       temp = temp-h*180;
   end
   phase(i) = temp;
end
end

function [inphase,quadphase] =
mixer(input_signal,fc,T,fs)
    k = 1;
    m = 1;
    qp = zeros(1,10000);
    inphase = zeros(1,length(input_signal));
    quadphase = zeros(1,length(input_signal));
    ip = zeros(1,10000);
    t = 0:1/fs:T-1/fs;
    for t1=0:1/fs:T-1/fs
        z = 2*cos(2*pi*fc*t1);
        ip(k) = z;
        k = k+1;
    end
    for t1=0:1/fs:T-1/fs
        y = 2*sin(2*pi*fc*t1);
        qp(m) = y;
        m = m+1;
    end
```

```
    inphase = input_signal.*ip;
    quadphase = input_signal.*qp;
    figure(3);
    plot(t,ip);
    grid on;
    title('carrier signals cosine');
    xlabel('time');
    ylabel('amplitude');
    %hold on;
    %       subplot(2,1,1);
    figure(4);
    plot(t,qp);
    %       subplot(2,1,2);
    grid on;
    title('carrier signals sine');
    xlabel('time');
    ylabel('amplitude');
end

function filter_out = filter_1(input_signal)
 n =  30;
 Fc = 1;
 FT = 1e3;
 %Rp = 0.005;
 %Rs = 0.005;
 wc = 2*(Fc/FT);
 window = kaiser(n+1);
 b = fir1(n,wc,'low',window);
 [h,w] = freqz(b,1);
 figure(7);
 plot(w,20*log(abs(h)));
 filter_out = filter(b,1,input_signal);

end

%first order finite approximation

function differentia = fofa(input_signal)
differentia(1) = input_signal(1);
for i=2:1:length(input_signal)-1
    differentia(i)=input_signal(i+1)-input_signal(i);
end
differentia(length(input_signal)) =
input_signal(length(input_signal));
end

function output = sampling(input_signal,T)
```

```
    for i=1:1:T
        output(i)=input_signal(i*1000);
    end
end
```

CONSIDERING NOISE:

```
clc
fs = 1e3;
T = 10000;
fc = 1000;
t = 0:1/fs:T-1/fs;
sim_bits = 10000;
snr_db = -4:1:10;
mixingfreq = 999;
h = 0.5;
x = randi([0 1],sim_bits,1);
mapper =  mapping(x);
phasetrajectory = tree(mapper,0.5);
phaseshift = 0;
figure(1);
plot(phasetrajectory);
xlabel('Symbol Duration');
ylabel('phase in degrees');
grid on;
for a = 1:1:length(snr_db)
k = 1;
q = zeros(1,1e7);
count = 0;
for i = 1:1:T
    if(mapper(i)==1)
      phaseshift = 180*h;
    end
    if(mapper(i)==-1)
        phaseshift = -180*h;
    end
    for t1=0.001+i-1:1/fs:i
        phase1(i)=phaseshift;
        s = cos(2*pi*fc*t1+(phaseshift*t1)/i);
        q(k) = s;
        k = k+1;
    end
end

[in_phase,quadrature_phase] = mixer(q,mixingfreq,T,fs);
snr = snr_db(a);
[in,qd] = noise(snr);
```

```
in_phase = in_phase+in;
quadrature_phase = quadrature_phase+qd;
quadrature_phase_filter = filter_1(quadrature_phase);
in_phase_filter = filter_1(in_phase);
quad_diff = fofa(quadrature_phase_filter);
in_diff = fofa(in_phase_filter);
oneterm = in_diff.*quadrature_phase_filter;
twoterm = quad_diff.*in_phase_filter;
summing = twoterm-oneterm;
rx_output = sampling(summing,T);

vk = 0.0276+rx_output;
for b = 1:1:10000
    if(vk>0)
        s(b) = 1;
    else
        s(b) = 0;
    end
end

for i=1:1:10000
   if(s(i)==x(i))
       count = count;
   else
       count = count+1;
   end

end
pe(a) = count/sim_bits;
end
figure(2);
plot(snr_db,pe);
```

//generation of noise

```
function [in,qd] = noise(snr_db)
snr_lin = 10^0.1.*snr_db;
in = zeros(1,1e7);
qd = zeros(1,1e7);
for i =1:1:1e7
    n(i) =
sqrt(1/(2*snr_lin))*randn(1)+1i*sqrt(1/(2*snr_lin))*randn
(1);
    in(i) = real(n(i));
    qd(i) = imag(n(i));
end
end
```