EEE5502: FOUNDATION OF DIGITAL SIGNAL PROCESSING

PROJECT 3: CONVOLUTION (MATCHED FILTERING)

NAME: APARNA HARIYANI

UFID: 69185846

EMAIL: ahariyani@ufl.edu

Generating required signals:

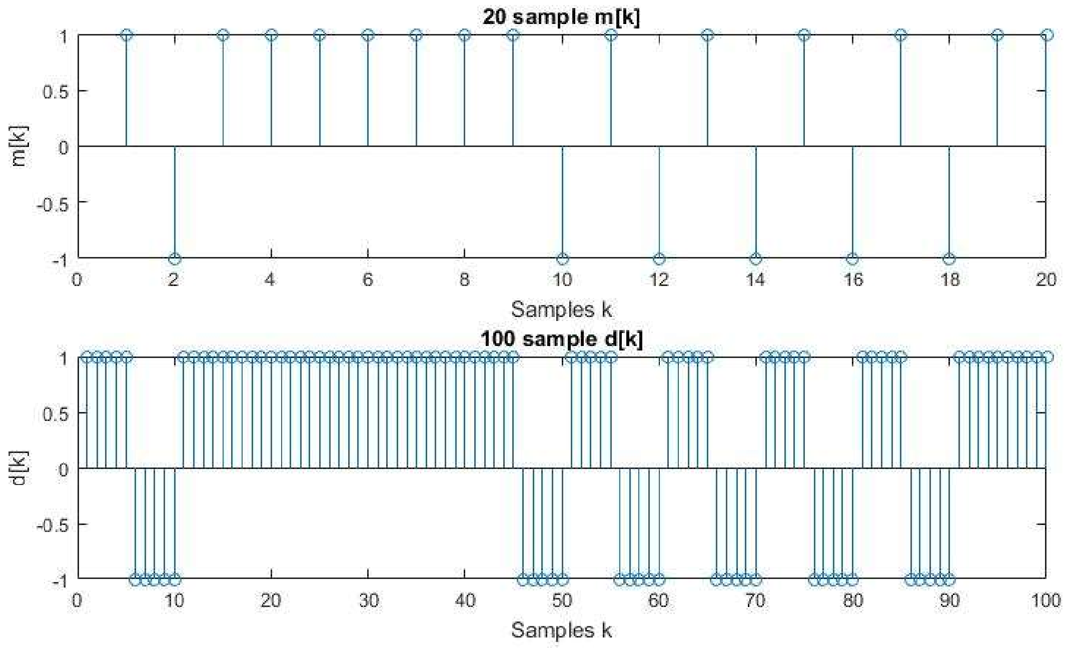

Fig. 1. The upper plot shows a 20 sample signal m[k] generated using random function in Matlab. The lower plot is m[k] upsampled by 5 i.e. d[k]

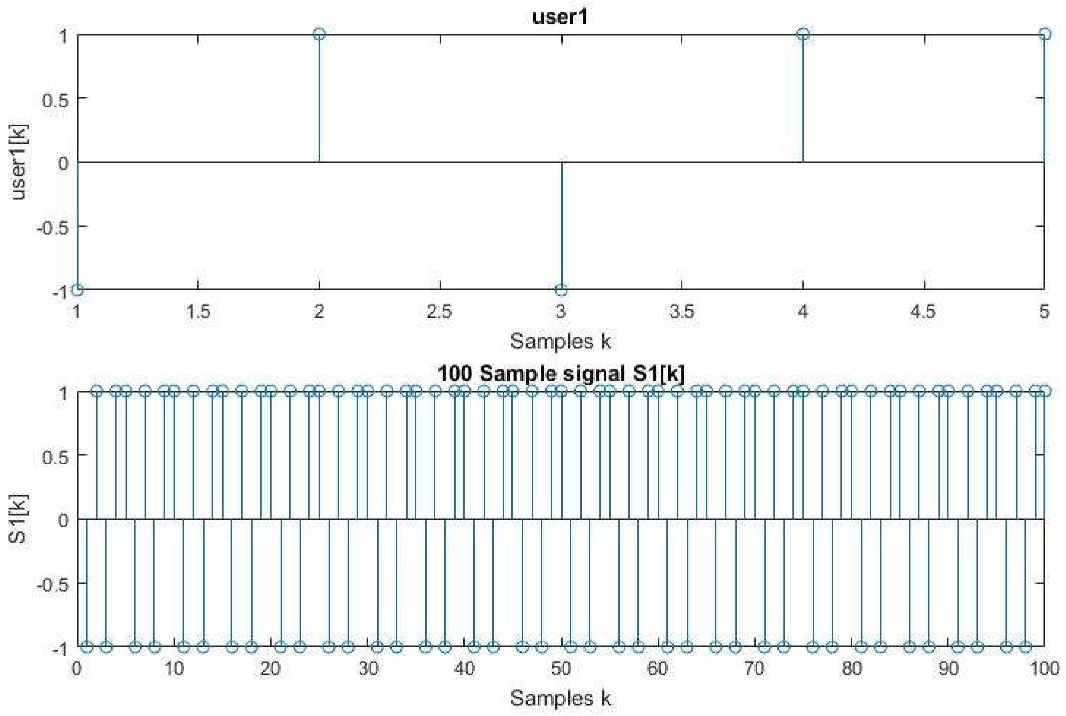

Fig. 2. The upper plot shows the Signal for user1. The lower plot is a 20 repetitions of user1 to make it a 100 sample signal

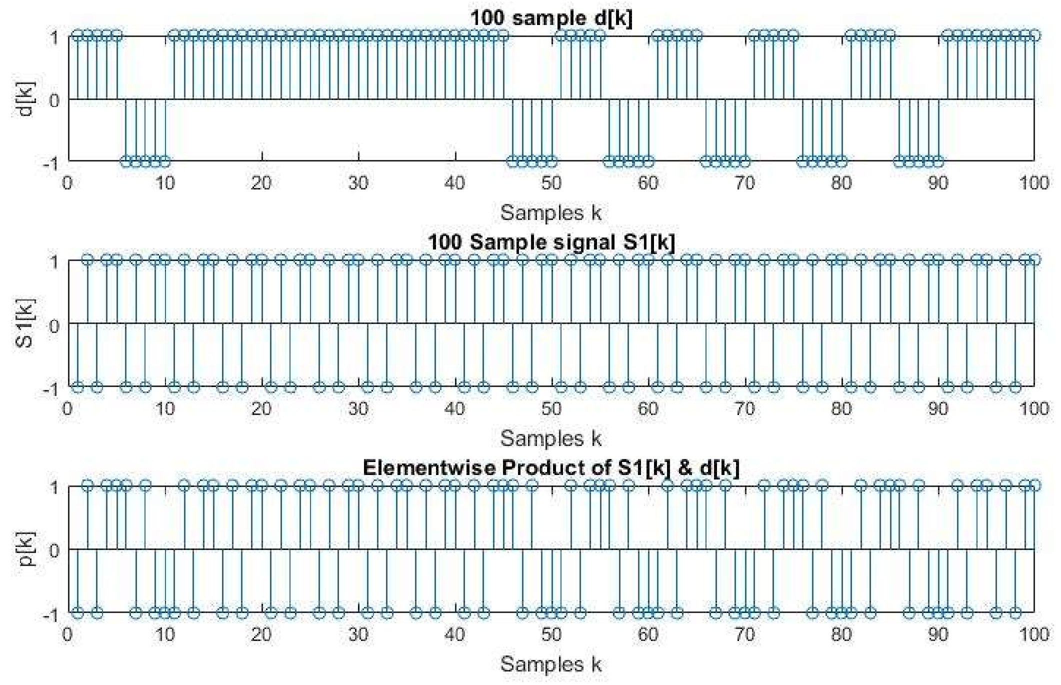

Fig. 3. The first plot is m[k] upsampled by 5 i.e. d[k], the second plot is 100 sample signal S1[k], the third plot is elementwise product p[k] = S1[k] * d[k]

Report 3.a:

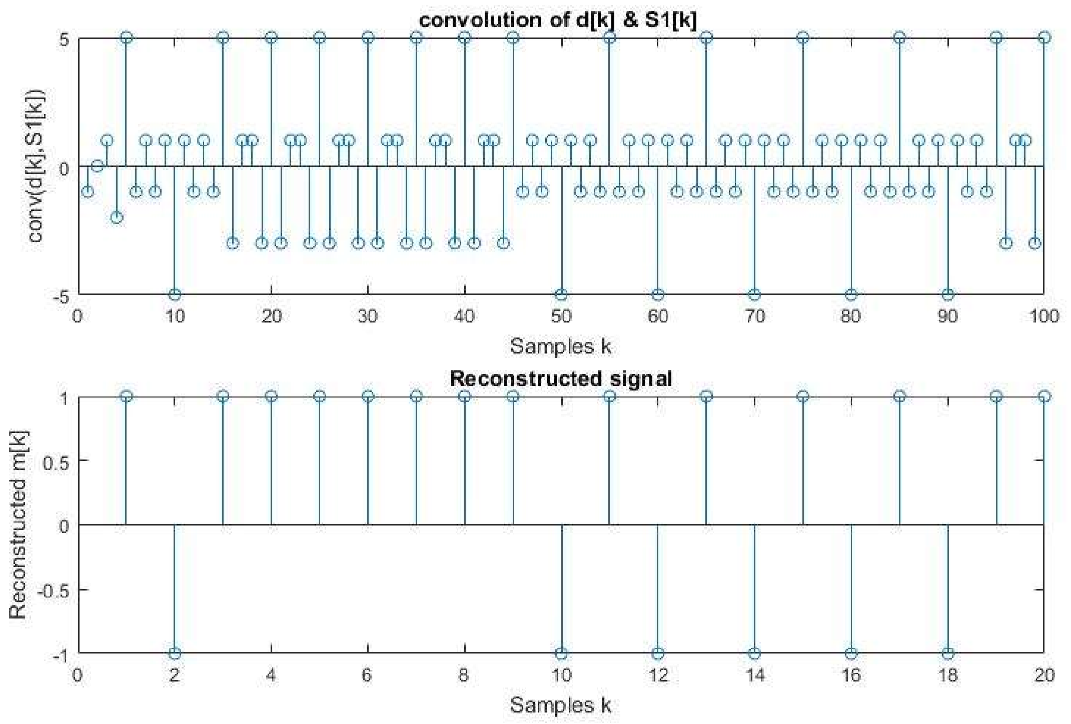

Fig. 4. The upper plot show convolution of d[k] and S1[k] which output at receiver of matched filter. The lower plot shows Reconstructed Original Signal

Report 3.b:

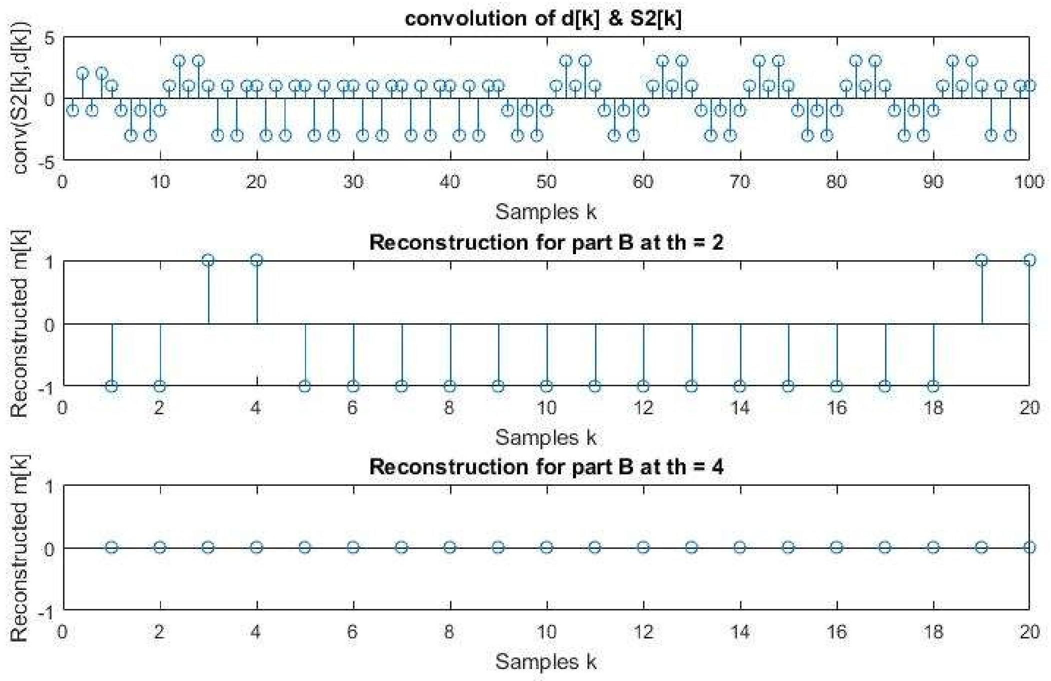

Fig. 5. The first plot shows convolution of d[k] with S2[k] which is repetition of user2[k] = `[-1 -1 -1 -1 1]`. The second plot is reconstructed signal with a threshold of 2. The third plot is reconstructed signal with threshold of 4.

Report 3.c:

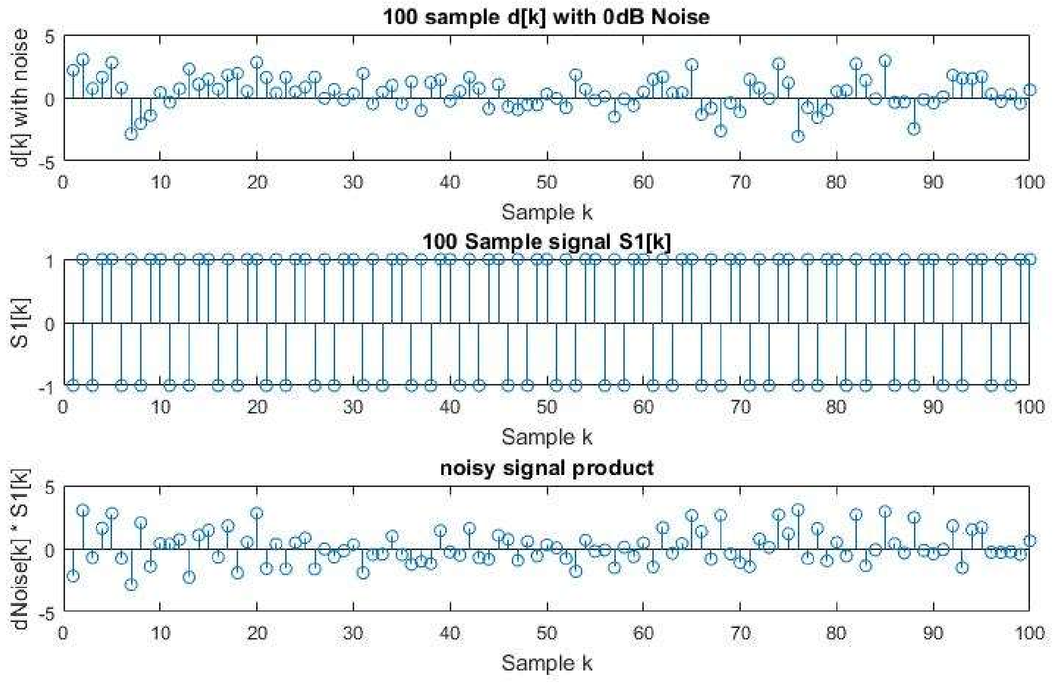

Fig. 6. The upper plot shows 100 sample d[k] with SNR 0dB added to it. The second plot shows S1[k]. The lower plot shows elementwise product of noisy d[k] and S1[k]

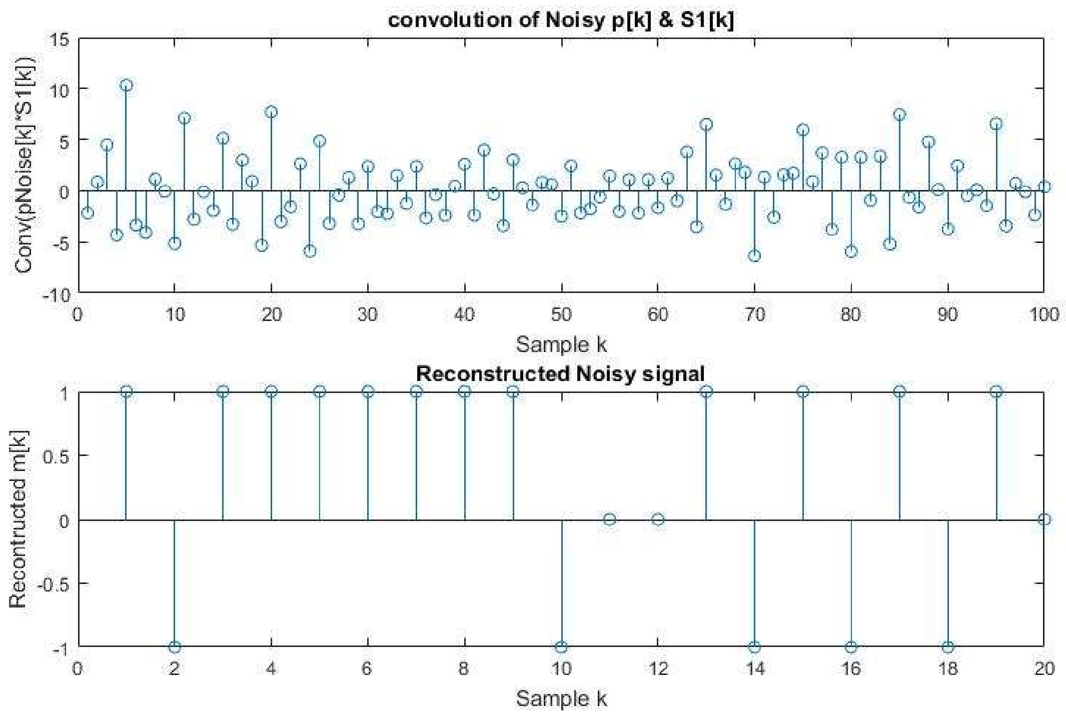

Fig. 7. The upper plot shows convoluted noisy product and h1[k] at the receiver. The lower plot shows the reconstructed m[k] with a threshold of 2

<u>BER for Part 1, Part 2, Part 3</u>:

	Part 1	**Part 2**		**Part 3**
Error rate	0	0.5	1	0.15
#Erroneous bits	0	10	20	3
Total #Samples	20	20	20	20
Threshold	4	2(Rational Choice)	4	2(Rational Choice)

We observe that the BER depends on the amount of noise, the threshold and the reverse sequence relative to user. Some noisy signals can be better reconstructed at a lower thresholds as compared to higher thresholds as noise might reduce the amplitudes of the received signals.

Matlab Code for Project 3

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: Aparna Hariyani
%UFID: 69185846
%Course: EEE 5502
%Project3 - Convolution (Matched Filtering)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;
clc;

N=5;

rng(1); %To ensure same sequence everytime
x = randi(2,1,20); %generate a random sequence of length 20

x(x==2) = [-1]; %Replace all 2's by -1 to get a random sequence of -1 & 1

d = upsample(x,N);
l1 = length(d);
for len = 1:l1  %removing zeros
    if d(len) == 0
        d(len) = d(len-1);
    end
end
figure(1);
subplot(2,1,1);
stem(x) %plotting m[k]
title('20 sample m[k]');
xlabel('Samples k');
ylabel('m[k]');
subplot(2,1,2);
stem(d) %plotting m[k]
title('100 sample d[k]');
xlabel('Samples k');
ylabel('d[k]');

user1= [-1 1 -1 1 1];    %user 1
h1 = [1 1 -1 1 -1 ];     % reverse image of user 1

user1N = (repmat(user1,1,20)); % user1 for 100 samples
figure(2);
subplot(2,1,1);
stem(user1);
title('user1');
xlabel('Samples k');
ylabel('user1[k]');
subplot(2,1,2);
stem(user1N)
title('100 Sample signal S1[k]');
xlabel('Samples k');
ylabel('S1[k]');

p = (d .* user1N); % pint wise multiplication of d[k] & user 1

figure(3);
subplot(3,1,1);
stem(d) %plotting m[k]
title('100 sample d[k]');
```

```
xlabel('Samples k');
ylabel('d[k]');
subplot(3,1,2);
stem(user1N)
title('100 Sample signal S1[k]');
xlabel('Samples k');
ylabel('S1[k]');
subplot(3,1,3);
stem(p)
title('Elementwise Product of S1[k] & d[k]');
xlabel('Samples k');
ylabel('p[k]');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part A
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

y1 = conv(p,h1);

th = N-1;
j = 1;
for i = 1:length(p)
    if (mod(i,N) == 0)
        if (abs(y1(i)) > th )
            r1(j) = y1(i)/(th+1); % m2k is the message signal obtained at the
receiver side, dividining by gain N to get original signal
            j = j+1;
        else
            r1(j) = 0; % for those values which are below the threshold and
should be rpresented to have the sample values k
            j = j+1;
        end
    end
end
j=1;
figure(4);
subplot(2,1,1);
stem(y1);
xlim([0 100]);
title('convolution of d[k] & S1[k] ');
xlabel('Samples k');
ylabel('conv(d[k],S1[k])');
subplot(2,1,2);
stem(r1);
title('Reconstructed signal');
xlabel('Samples k');
ylabel('Reconstructed m[k]');

% Finding error rate

hError = comm.ErrorRate;  % Using communication toolbox
input = reshape(x,length(x),1);
output = reshape(r1,length(r1),1);
errorRate1 = step(hError,input,output);

display('Part 1');
display(errorRate1(1),'Error rate for Part 1 ');
display(errorRate1(2),'No. of erroneous bits for Part 1 ');
display(errorRate1(3),'Total no. of samples ');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part B
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
user2= [-1 -1 -1 -1 1];    %user 1
h2 = [1 -1 -1 -1 -1 ];     % reverse image of user 1

y2 = conv(p,h2);
th = N-3;
for i = 1:length(p)
     if (abs(y2(i)) > th )
        r2(j) = y2(i)/(th+1); % m2k is the message signal obtained at the
receiver side, dividining by gain N to get original signal
        j = j+1;
     end
end
j = 1;
th = N-1;
for i = 1:length(p)
    if (mod(i,N) == 0)
        if (abs(y2(i)) > th )
            r21(j) = y2(i)/(th+1); % m2k is the message signal obtained at
the receiver side, dividining by gain N to get original signal
            j = j+1;
        else
            r21(j) = 0; % for those values which are below the threshold and
should be rpresented to have the sample values k
            j = j+1;
        end
    end
end

r2 = r2(1:length(x));

figure(5);
subplot(3,1,1);
stem(y2);
xlim([0 100]);
title('convolution of d[k] & S2[k] ');
xlabel('Samples k');
ylabel('conv(S2[k],d[k])');
subplot(3,1,2);
stem(r2);
title('Reconstruction for part B at th = 2');
xlabel('Samples k');
ylabel('Reconstructed m[k]');
subplot(3,1,3);
stem(r21);
title('Reconstruction for part B at th = 4');
xlabel('Samples k');
ylabel('Reconstructed m[k]');

hError = comm.ErrorRate;  % Using communication toolbox
input2 = reshape(x,length(x),1);
output2 = reshape(r2,length(r2),1);
errorRate2 = step(hError,input2,output2);

disp('Part 2');
display(errorRate2(1),'Error rate for Part 2 ');
display(errorRate2(2),'No. of erroneous bits for Part 2 ');
display(errorRate2(3),'Total no. of samples ');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part C
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dNoise = awgn(d,0,'measured');

pNoise = (dNoise .* user1N); % pint wise multiplication of d[k] & user 1

figure(6);
subplot(3,1,1);
stem(dNoise) %plotting m[k]
title('100 sample d[k] with 0dB Noise');
ylabel('d[k] with noise');
xlabel('Sample k');
subplot(3,1,2);
stem(user1N)
title('100 Sample signal S1[k]');
ylabel('S1[k]');
xlabel('Sample k');
subplot(3,1,3);
stem(pNoise)
title('noisy signal product');
ylabel('dNoise[k] * S1[k]');
xlabel('Sample k');

yNoise = conv(pNoise,h1);

th = N-3;
j = 1;
for i = 1:length(p)
    if (mod(i,N) == 0)
        if (abs(yNoise(i)) > th )
            rNoise(j) = yNoise(i)/(th+1); % m2k is the message signal
obtained at the receiver side, dividining by gain N to get original signal
            j = j+1;
        else
            rNoise(j) = 0; % for those values which are below the threshold
and should be rpresented to have the sample values k
            j = j+1;
        end
    end
end
j=1;

for i = 1:length(rNoise)
    if (rNoise(i) > 0)
        rNoise(i) = 1;
    else
        if (rNoise(i) < 0) % else it will remove the 0s place in previous
steps too, so check required
            rNoise(i) = -1;
        end
    end
end
figure(7);
subplot(2,1,1);
stem(yNoise);
xlim([0 100]);
title('convolution of Noisy p[k] & S1[k]');
ylabel('Conv(pNoise[k]*S1[k])');
```

```
xlabel('Sample k');
subplot(2,1,2);
stem(rNoise);
title('Reconstructed Noisy signal');
ylabel('Recontructed m[k]');
xlabel('Sample k');

% Finding error rate

hError = comm.ErrorRate;  % Using communication toolbox
input3 = reshape(x,length(x),1);
output3 = reshape(rNoise,length(rNoise),1);
errorRate3 = step(hError,input3,output3);

disp('Part 3');
display(errorRate3(1),'Error rate for Part 3 ');
display(errorRate3(2),'No. of erroneous bits for Part 3 ');
display(errorRate3(3),'Total no. of samples ');
```