CT216
introduction to communication system
Lab group 2

- We declare that
  - The work that we are presenting is our own work.
  - We have not copied the work (the code, the results, etc.) that someone else has done.
  - Concepts, understanding and insights we will be describing are our own.
  - We make this pledge truthfully. We know that violation of this solemn pledge can
  - carry grave consequences.

1. Divyarajsinh Chundavat – 202201155
2. Meet Katharotiya -202201157
3. Kishan Patel – 202201159
4. Smit Godhani-202201162
5. Jay Goyani-202201163
6. Krisha Brahmbhatt – 202201164
7. Het Gandhi – 202201167
8. Jyot vasava – 202201169
9. Parshwa Modi – 202201165
10. Parth Vadodaria – 202201174
11. Harshvardhan Vajani – 202201413

Convolution Encoder and BPSK/AWGN Simulation

```
clf;
clear all;
```

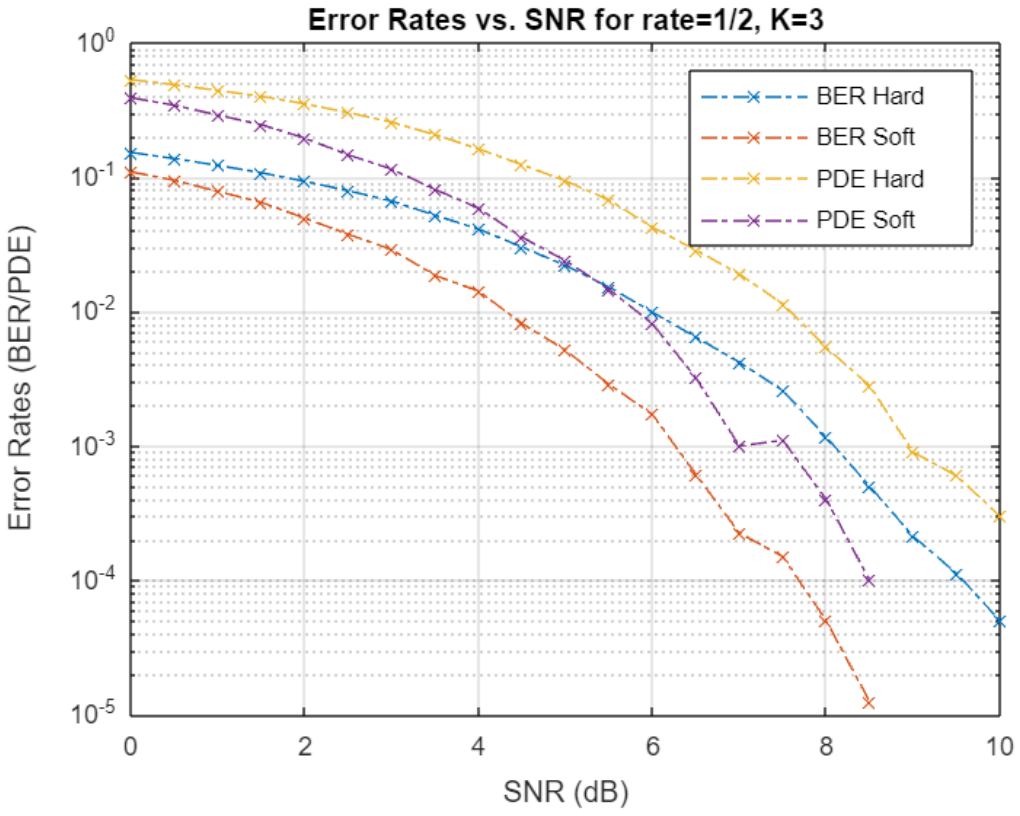
--> Common Inputs

```
EbNo_dB=0:0.5:10;
EbNo=10.^(EbNo_dB/10);
k=1;
[~,col_EbNo]=size(EbNo_dB);
Nsim=10000;

BER_uncoded=(1/2)*erfc(sqrt(EbNo));

M=create_msg(6);
```

1) Rate= 1/2 Constraint Length=3

```
K=3;
rate=1/2;
g=[1,0,1
   1,1,1];

[Org_msg,encoded_msg]=encode_msg(K,M,g);
s=create_symbols(encoded_msg);

sigma = sqrt(1./(rate.*EbNo));
BER1_hard=zeros(1,col_EbNo);
BER1_soft=zeros(1,col_EbNo);

PDE1_hard=zeros(1,col_EbNo);
PDE1_soft=zeros(1,col_EbNo);

for index_EbNo=1:col_EbNo
    SNR=sigma(1,index_EbNo);

    error_cnt_hard_ber=0;
    error_cnt_soft_ber=0;

    error_cnt_hard_pde=0;
    error_cnt_soft_pde=0;

    for index_Nsim=1:Nsim
        received_msg=pass_msg(s,SNR);
        digitized_msg=digitize(received_msg);
        decoded_msg_hard=viterbi_hard(g,digitized_msg);
        decoded_msg_soft=viterbi_soft(g,received_msg);
```

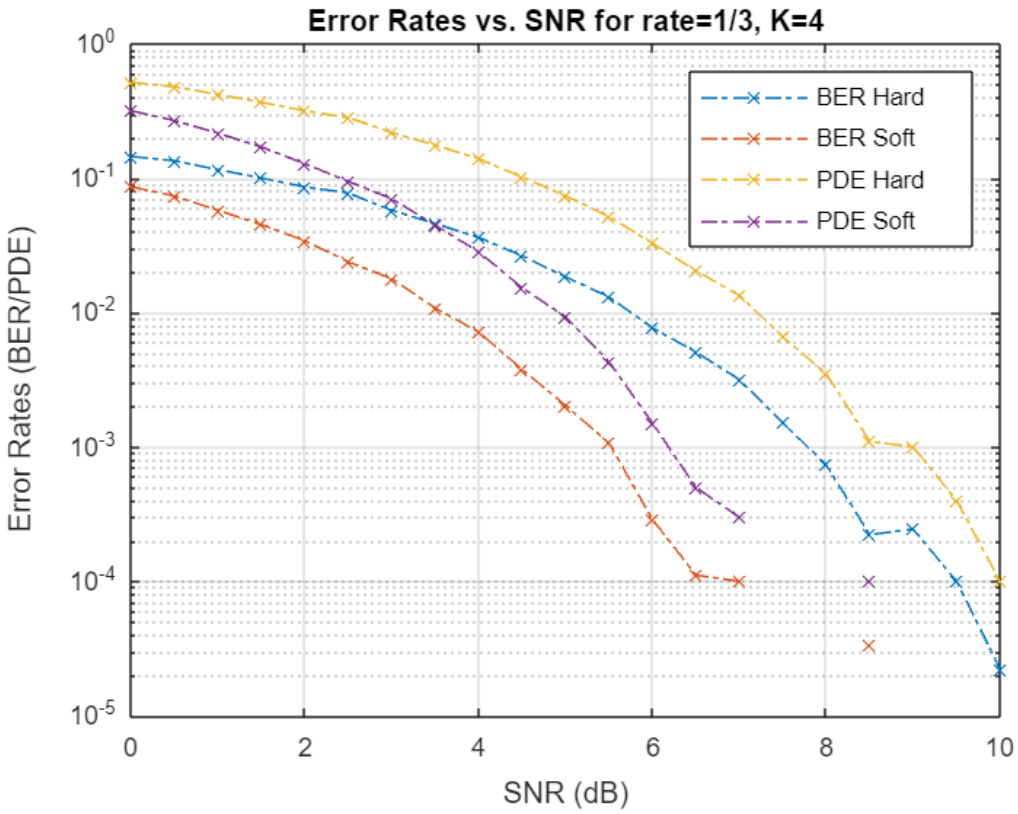
```
        error_cnt_hard_pde=error_cnt_hard_pde+(sum(decoded_msg_hard~=Org_msg)>0);
        error_cnt_soft_pde=error_cnt_soft_pde+(sum(decoded_msg_soft~=Org_msg)>0);

        error_cnt_hard_ber=error_cnt_hard_ber+(sum(decoded_msg_hard~=Org_msg));
        error_cnt_soft_ber=error_cnt_soft_ber+(sum(decoded_msg_soft~=Org_msg));
    end

    PDE1_hard(1,index_EbNo)=error_cnt_hard_pde/Nsim;
    PDE1_soft(1,index_EbNo)=error_cnt_soft_pde/Nsim;

    BER1_hard(1,index_EbNo)=error_cnt_hard_ber/(length(Org_msg)*Nsim);
    BER1_soft(1,index_EbNo)=error_cnt_soft_ber/(length(Org_msg)*Nsim);
end

figure(1);
semilogy(EbNo_dB, BER1_hard,'x-.',EbNo_dB, BER1_soft,'x-.', EbNo_dB,
PDE1_hard,'x-.',EbNo_dB, PDE1_soft,'x-.');
xlabel('SNR (dB)');
ylabel('Error Rates (BER/PDE)');
grid on;
legend('BER Hard','BER Soft','PDE Hard','PDE Soft');
title('Error Rates vs. SNR for rate=1/2, K=3');
```



2) Rate= 1/3 Constraint Length=4

```
K=4;
rate=1/3;
g=[1,0,1,1
   1,1,0,1
   1,1,1,1];

[Org_msg,encoded_msg]=encode_msg(K,M,g);
s=create_symbols(encoded_msg);

sigma = sqrt(1./(rate*EbNo));

BER2_hard=zeros(1,col_EbNo);
BER2_soft=zeros(1,col_EbNo);

PDE2_hard=zeros(1,col_EbNo);
PDE2_soft=zeros(1,col_EbNo);


for index_EbNo=1:col_EbNo
    SNR=sigma(1,index_EbNo);

    error_cnt_hard_ber=0;
    error_cnt_soft_ber=0;

    error_cnt_hard_pde=0;
    error_cnt_soft_pde=0;

    for index_Nsim=1:Nsim
        received_msg=pass_msg(s,SNR);
        digitized_msg=digitize(received_msg);
        decoded_msg_hard=viterbi_hard(g,digitized_msg);
        decoded_msg_soft=viterbi_soft(g,received_msg);

        error_cnt_hard_pde=error_cnt_hard_pde+(sum(decoded_msg_hard~=Org_msg) > 0);
        error_cnt_soft_pde=error_cnt_soft_pde+(sum(decoded_msg_soft~=Org_msg) > 0);

        error_cnt_hard_ber=error_cnt_hard_ber+(sum(decoded_msg_hard~=Org_msg));
        error_cnt_soft_ber=error_cnt_soft_ber+(sum(decoded_msg_soft~=Org_msg));
    end

    PDE2_hard(1,index_EbNo)=error_cnt_hard_pde/Nsim;
    PDE2_soft(1,index_EbNo)=error_cnt_soft_pde/Nsim;

    BER2_hard(1,index_EbNo)=error_cnt_hard_ber/(length(Org_msg)*Nsim);
    BER2_soft(1,index_EbNo)=error_cnt_soft_ber/(length(Org_msg)*Nsim);

end

figure(2);
```

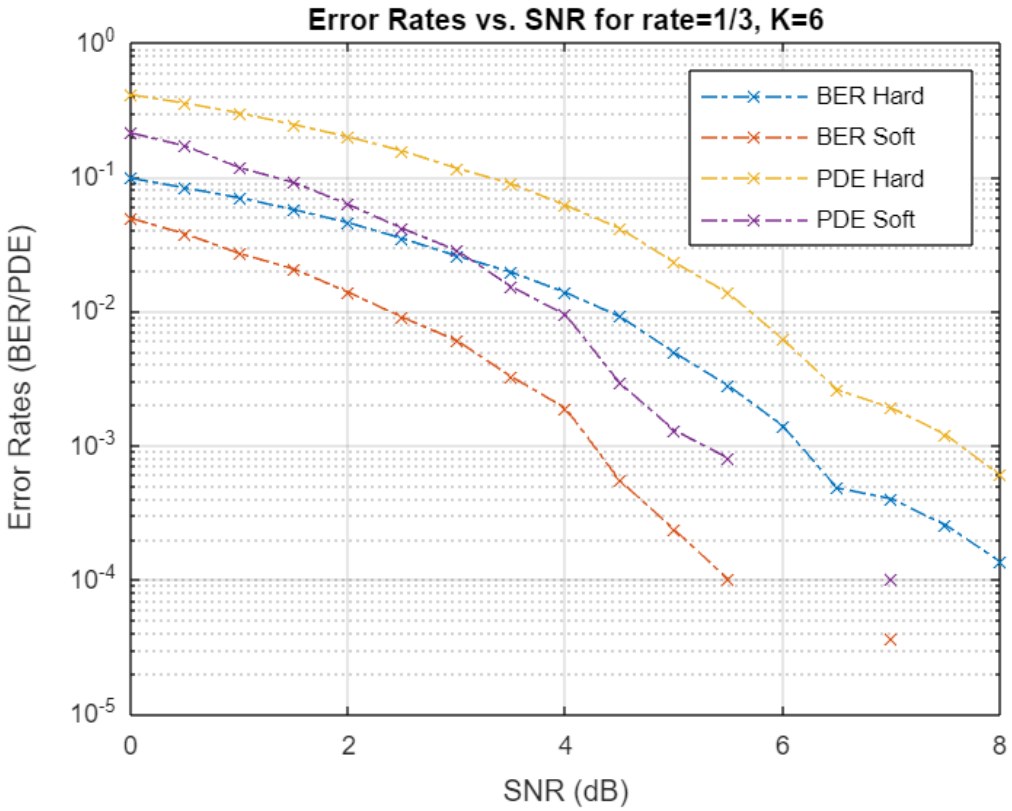
```
semilogy(EbNo_dB, BER2_hard,'x-.',EbNo_dB, BER2_soft,'x-.', EbNo_dB,
PDE2_hard,'x-.',EbNo_dB, PDE2_soft,'x-.');
xlabel('SNR (dB)');
ylabel('Error Rates (BER/PDE)');
grid on;
legend('BER Hard','BER Soft','PDE Hard','PDE Soft');
title('Error Rates vs. SNR for rate=1/3, K=4');
```



3) Rate= 1/3 Constraint Length=6

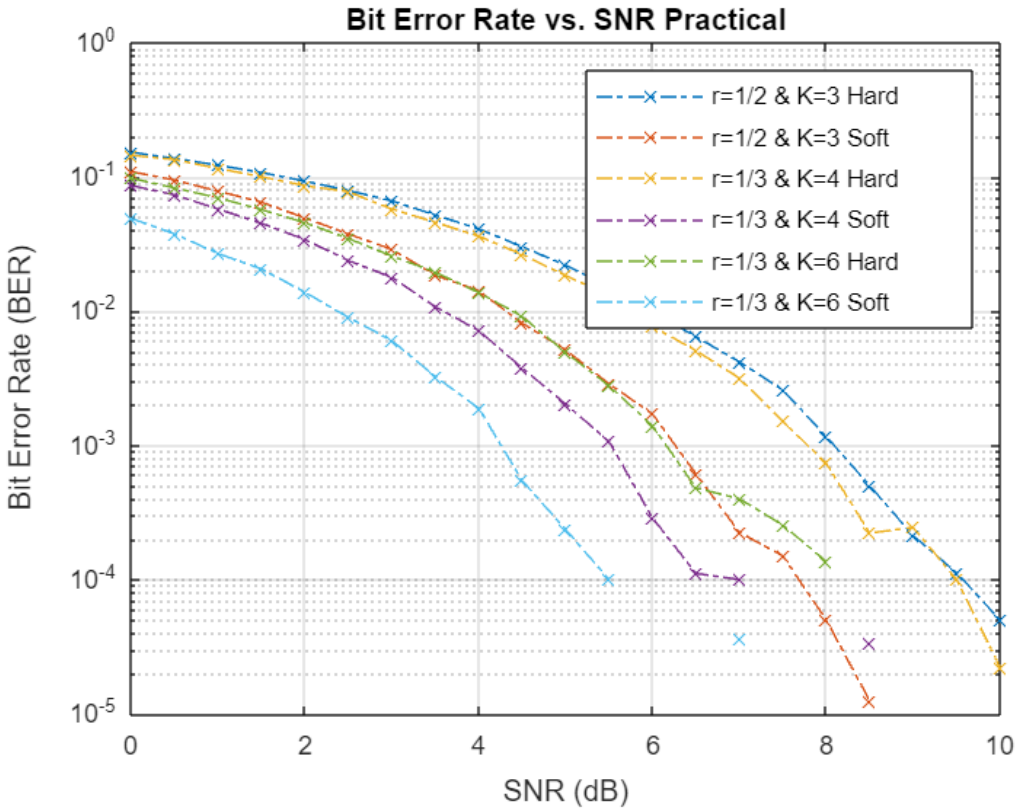
```
K=6;
rate=1/3;
g=[1,0,0,1,1,1
   1,0,1,0,1,1
   1,1,1,1,0,1];

[Org_msg,encoded_msg]=encode_msg(K,M,g);
s=create_symbols(encoded_msg);

sigma = sqrt(1./(rate*EbNo));

BER3_hard=zeros(1,col_EbNo);
BER3_soft=zeros(1,col_EbNo);

PDE3_hard=zeros(1,col_EbNo);
PDE3_soft=zeros(1,col_EbNo);
```

```
for index_EbNo=1:col_EbNo
    SNR=sigma(1,index_EbNo);

    error_cnt_hard_ber=0;
    error_cnt_soft_ber=0;

    error_cnt_hard_pde=0;
    error_cnt_soft_pde=0;

    for index_Nsim=1:Nsim
        received_msg=pass_msg(s,SNR);
        digitized_msg=digitize(received_msg);
        decoded_msg_hard=viterbi_hard(g,digitized_msg);
        decoded_msg_soft=viterbi_soft(g,received_msg);

        error_cnt_hard_pde=error_cnt_hard_pde+(sum(decoded_msg_hard~=Org_msg) > 0);
        error_cnt_soft_pde=error_cnt_soft_pde+(sum(decoded_msg_soft~=Org_msg) > 0);

        error_cnt_hard_ber=error_cnt_hard_ber+(sum(decoded_msg_hard~=Org_msg));
        error_cnt_soft_ber=error_cnt_soft_ber+(sum(decoded_msg_soft~=Org_msg));
    end

    PDE3_hard(1,index_EbNo)=error_cnt_hard_pde/Nsim;
    PDE3_soft(1,index_EbNo)=error_cnt_soft_pde/Nsim;

    BER3_hard(1,index_EbNo)=error_cnt_hard_ber/(length(Org_msg)*Nsim);
    BER3_soft(1,index_EbNo)=error_cnt_soft_ber/(length(Org_msg)*Nsim);

end

figure(3);
semilogy(EbNo_dB, BER3_hard,'x-.',EbNo_dB, BER3_soft,'x-.', EbNo_dB,
PDE3_hard,'x-.',EbNo_dB, PDE3_soft,'x-.');
xlabel('SNR (dB)');
ylabel('Error Rates (BER/PDE)');
grid on;
legend('BER Hard','BER Soft','PDE Hard','PDE Soft');
title('Error Rates vs. SNR for rate=1/3, K=6');
```

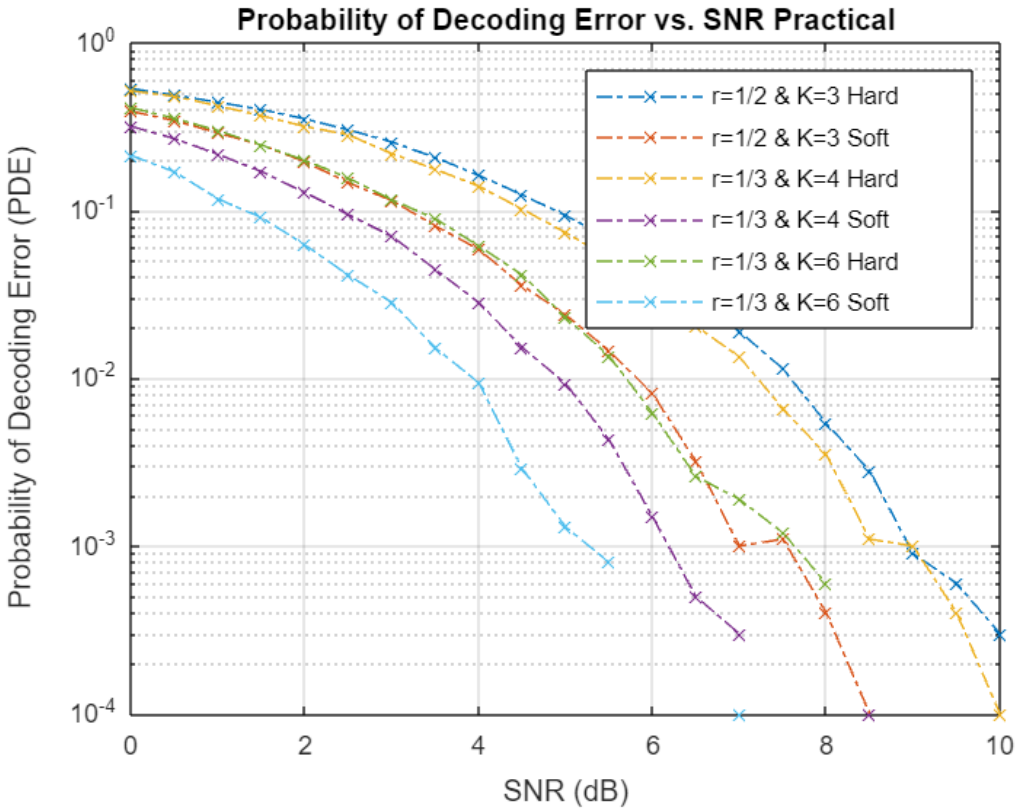


--> Plots for all of them together with BER and PDE plots separate.

```
figure(4);
semilogy(EbNo_dB, BER1_hard,'x-.',EbNo_dB, BER1_soft,'x-.', EbNo_dB, BER2_hard,
'x-.',EbNo_dB, BER2_soft, 'x-.',EbNo_dB, BER3_hard,'x-.', EbNo_dB, BER3_soft,'x-.');
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
grid on;
legend('r=1/2 & K=3 Hard','r=1/2 & K=3 Soft','r=1/3 & K=4 Hard','r=1/3 & K=4
Soft','r=1/3 & K=6 Hard','r=1/3 & K=6 Soft');
title('Bit Error Rate vs. SNR Practical');
```




```
figure(5);
semilogy(EbNo_dB, PDE1_hard,'x-.',EbNo_dB, PDE1_soft,'x-.', EbNo_dB,
PDE2_hard,'x-.', EbNo_dB, PDE2_soft,'x-.', EbNo_dB, PDE3_hard,'x-.', EbNo_dB,
PDE3_soft,'x-.');
xlabel('SNR (dB)');
ylabel('Probability of Decoding Error (PDE)');
grid on;
legend('r=1/2 & K=3 Hard','r=1/2 & K=3 Soft','r=1/3 & K=4 Hard','r=1/3 & K=4
Soft','r=1/3 & K=6 Hard','r=1/3 & K=6 Soft');
title('Probability of Decoding Error vs. SNR Practical');
```



--> Function to Encode Message

```
function [X,encode] = encode_msg(K,M,g)
    [~,Mnum_cols]=size(M);

    sizeX=Mnum_cols+K-1;
    X=zeros(1,sizeX);
    for i=1:Mnum_cols
        X(i)=M(i);
    end

    [gnum_rows,~]=size(g);

    encode=-ones(1,gnum_rows*sizeX);

    z=1;
    for n=1:sizeX
        for i=1:gnum_rows
            val=0;
            for j=1:K
                if n+1-j>0
                    val=val+g(i,j)*X(n+1-j);
                end
```

```
            end
            encode(z)=mod(val,2);
            z=z+1;
        end
    end
end
```

--> Function to create a message

```
function M=create_msg(input_bits_length)
    M=zeros(1,input_bits_length);
    for i=1:input_bits_length
        M(i)=randn > 0.5;
    end
end
```

--> BPSK Simulation

```
function s=create_symbols(encoded_msg)
    s=1-2*encoded_msg;
end
```

--> AWGN Simulation

```
function noisy_msg=pass_msg(s,sigma_n)
    [~,col_s]=size(s);
    % Noise Creation
    us= randn([1,col_s]);
    noise=sigma_n * us;

    % Noise AdditionBm1
    noisy_msg=s+noise;
end
```

--> Digitizing the received message

```
function digitized_bits = digitize(received_bits)
    digitized_bits=received_bits<0;
end
```

--> Viterbi Decoding (Hard decision Decoding)

```
function Decoded_msg_hard =viterbi_hard(g,received_bit)

No_Of_Gen=size(g,1);
```

```
Constraint_Length=size(g,2);
NumStates=2^(Constraint_Length-1);
NextStates=zeros(2^(Constraint_Length-1), 2);
Outputs=zeros(2^(Constraint_Length-1), 2);


% calculating nextstate and Outputs for all states
for state = 0:(2^(Constraint_Length-1)-1)
    for input = 0:1
        nextState = bitshift(state, -1);
        if(input==1)
            nextState=nextState+(2^(Constraint_Length-2));
        end
        NextStates(state+1, input+1) = nextState;

        % calculation of Outputs for particular input
        state2= dec2bin(state,log2(NumStates))-'0';
        state2=[input,state2];

        temp=[];
        for j=1:No_Of_Gen
            temp=[temp,mod(sum(bitand( g(j,:),state2)),2)];
        end

        temp_string = num2str(temp);
        sequence = strrep(temp_string, ' ', '');
        Outputs(state+1,input+1) = bin2dec(sequence);

    end
end

%% computation of Trellis Graph

column=length(received_bit)/No_Of_Gen+1;
dp = repmat(1000, NumStates, column);
dp(1,1)=0;
bit_idx=1;
for i=1:column-1
    temp = [];
    for j=1:No_Of_Gen
        temp=[temp,received_bit(bit_idx)];
        bit_idx = bit_idx + 1;
    end

    for st=1:NumStates
        out= bi2de(temp,'left-msb');

        hamming_dist_0=sum(bitget(bitxor(out,Outputs(st,1)),1:32));
        hamming_dist_1=sum(bitget(bitxor(out,Outputs(st,2)),1:32));
```

```
        dp(NextStates(st,1)+1,i+1)=min(dp(NextStates(st,1)+1,i+1),dp(st,i)
+hamming_dist_0);
        dp(NextStates(st,2)+1,i+1)=min(dp(NextStates(st,2)+1,i+1),dp(st,i)
+hamming_dist_1);
    end
end

%% Backtracking for Original Message
next_st=0;
Decoded_msg_hard=[];

for i=column-1:-1:1
    prev_st1=[];
    prev_st2=[];
    f=0;
    for st=1:NumStates

        if( f==0 )
            if(NextStates(st,1)==next_st )
                prev_st1=[st,1];
                f=1;
            end
            if(NextStates(st,2)==next_st)
                prev_st1=[st,2];
                f=1;
            end
        end

        if(f==1)
            if(NextStates(st,1)==next_st )
                prev_st2=[st,1];
            end
            if(NextStates(st,2)==next_st)
                prev_st2=[st,2];

            end
        end

    end

    temp=[];
    for j=No_Of_Gen-1:-1:0
        temp = [temp,received_bit(No_Of_Gen*i-j)];
    end

    out=bi2de(temp,'left-msb');
    Bm_0=sum(bitget(bitxor(out,Outputs(prev_st1(1,1),prev_st1(1,2))),1:32));
    Bm_1=sum(bitget(bitxor(out,Outputs(prev_st2(1,1),prev_st2(1,2))),1:32));
```

```
    if((dp(prev_st1(1,1),i)+Bm_0)<(dp(prev_st2(1,1),i)+Bm_1))
        Decoded_msg_hard=[Decoded_msg_hard,prev_st1(1,2)-1];
        next_st=prev_st1(1)-1;
    end

    if((dp(prev_st1(1,1),i)+Bm_0)>(dp(prev_st2(1,1),i)+Bm_1))
        Decoded_msg_hard=[Decoded_msg_hard,prev_st2(1,2)-1];
        next_st=prev_st2(1)-1;
    end

    if((dp(prev_st1(1,1),i)+Bm_0)==(dp(prev_st2(1,1),i)+Bm_1))

        if(Bm_0<Bm_1)
            Decoded_msg_hard=[Decoded_msg_hard,prev_st1(1,2)-1];
            next_st=prev_st1(1)-1;
        else
            Decoded_msg_hard=[Decoded_msg_hard,prev_st2(1,2)-1];
            next_st=prev_st2(1)-1;
        end
    end
end

Decoded_msg_hard = flip(Decoded_msg_hard);
end
```

--> Viterbi Decoding (Soft decision Decoding)

```
function Decoded_msg_soft =viterbi_soft(g,received_bit)

No_Of_Gen=size(g,1);
constraint_length=size(g,2);
NumStates=2^(constraint_length-1);
NextStates=zeros(2^(constraint_length-1), 2);
Outputs=zeros(2^(constraint_length-1), 2);



%% calculating nextstate and Outputs for all states

for state = 0:(2^(constraint_length-1)-1)
    for input = 0:1
        nextState = bitshift(state, -1);
        if(input==1)
            nextState=nextState+(2^(constraint_length-2));
        end
        NextStates(state+1, input+1) = nextState;


        state2= dec2bin(state,log2(NumStates))-'0';
```

```
        state2=[input,state2];

           temp=[];
           for j=1:No_Of_Gen
               temp=[temp,mod(sum(bitand( g(j,:),state2)),2)];
           end

        temp_string = num2str(temp);
        sequence = strrep(temp_string, ' ', '');
        Outputs(state+1,input+1) = bin2dec(sequence);

    end
end



%% computation of Trellis Graph

column=length(received_bit)/No_Of_Gen+1;
dp = repmat(500, NumStates, column);
dp(1,1)=0;
bit_idx=1;
for i=1:column-1
    out = [];
   for j=1:No_Of_Gen
       out=[out,received_bit(bit_idx)];
       bit_idx = bit_idx + 1;
   end

    for st=1:NumStates

        output1=dec2bin(Outputs(st,1),No_Of_Gen);
        output1=output1-'0';
        output1=1-2.*output1;

        output2=dec2bin(Outputs(st,2),No_Of_Gen);
        output2=output2-'0';
        output2=1-2.*output2;

        euclidean_dist_0=sum((out-output1).^2);
        euclidean_dist_1=sum((out-output2).^2);
        dp(NextStates(st,1)+1,i+1)=min( dp(NextStates(st,1)+1,i+1),dp(st,i)
+euclidean_dist_0);
        dp(NextStates(st,2)+1,i+1)=min( dp(NextStates(st,2)+1,i+1),dp(st,i)
+euclidean_dist_1);
    end
end
```

```
%% Backtracking for Original Message

curr_st=0;
Decoded_msg_soft=[];
for i=column-1:-1:1
    prevst1=[];
    prevst2=[];
    f=0;
    for st=1:NumStates
        if( f==0 )
            if(NextStates(st,1)==curr_st )
                prevst1=[st,1];
                f=1;
            end
            if(NextStates(st,2)==curr_st)
                prevst1=[st,2];
                f=1;
            end

        end
        if(f==1)
            if(NextStates(st,1)==curr_st )
                prevst2=[st,1];


            end
            if(NextStates(st,2)==curr_st)
                prevst2=[st,2];

            end
        end

    end

        out=[];
        for j=No_Of_Gen-1:-1:0
        out = [out,received_bit((No_Of_Gen)*i-j)];
        end

        output1=dec2bin(Outputs(prevst1(1,1),prevst1(1,2)),No_Of_Gen);
        output1=output1-'0';
        output1=1-2.*output1;

        output2=dec2bin(Outputs(prevst2(1,1),prevst2(1,2)),No_Of_Gen);
        output2=output2-'0';
        output2=1-2.*output2;

        Bm_0=sum((out-output1).^2);

        Bm_1=sum((out-output2).^2);
```

```
    if((dp(prevst1(1,1),i)+Bm_0)<(dp(prevst2(1,1),i)+Bm_1))
        Decoded_msg_soft=[Decoded_msg_soft,prevst1(1,2)-1];
        curr_st=prevst1(1)-1;
    end
    if((dp(prevst1(1,1),i)+Bm_0)>(dp(prevst2(1,1),i)+Bm_1))
        Decoded_msg_soft=[Decoded_msg_soft,prevst2(1,2)-1];
        curr_st=prevst2(1)-1;

    end
    if((dp(prevst1(1,1),i)+Bm_0)==(dp(prevst2(1,1),i)+Bm_1))

        if(Bm_0<Bm_1)
            Decoded_msg_soft=[Decoded_msg_soft,prevst1(1,2)-1];
            curr_st=prevst1(1)-1;
        else
            Decoded_msg_soft=[Decoded_msg_soft,prevst2(1,2)-1];
            curr_st=prevst2(1)-1;
        end
    end
end
Decoded_msg_soft=flip(Decoded_msg_soft);
end
```