Introduction

It is the utmost common issue occured in signals that were caused by unwanted disturbance known as Noises or Error in interrupting the communication framework reliability. On the other hand, it was proven by Shannon in the 1940s that it is possible to transmit information digitally without virtual errors. This can be achieved by adding redundant bits to boost error detection correcting code [1]. Error control codes such as Hamming Codes, Convolutional Codes and Block Codes are introduced. Hamming code can only detect and correct up to one error or detection at most 2 errors. The limitation is Hamming code cannot detect multiple bits error . The block codes are depending on the current input bit while convolutional codes refer to current input bits as well as previous input bits sequence . Block codes are therefore known as memoryless while convolutional code is memory based [2]. Convolutional codes are used extensively to achieve reliable data transfer in some applications such as digital video ,radio ,mobile communication and satellite communication [3].

Operating Principle of Convolution Code

To encode the data , shift registers will be introduced , each holding one bit of data . All registers will be initiated with a value of 0. The number of the shift registers are known as the constraint length ,K . Number of bits entering the encoder are known as k . Number of output bits are then known as n . Code rate is then determined by k/n . The value of each shift register will be added by using modulo-2 adders and produce the outputs. The addition of the input bits are determined by the generator polynomials . Generator polynomial is a function that determines which shift register value to be added up through modulo-2 adders and produces the output .

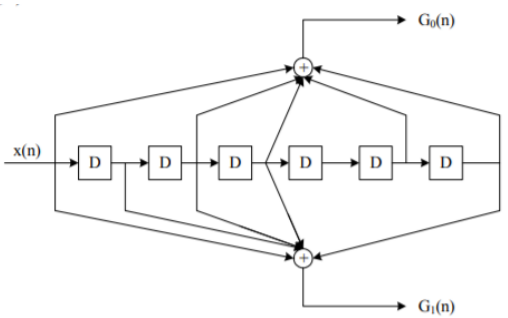

Figure 1 [3] Convolution Encoder

In the figure above , there are 7 shift registers therefore a constraint length of 7. Generator polynomial has function of $g_0(n)$ $=1+n^2+n^3+n^5+n^6$ which correspond to g_0 $=(1,0,1,1,0,1,1)$. $g_1(n) = 1+n+n^2+n^3+n^6$ which correspond to g1=(1,1,1,1,0,0,1)[4]. The input message bit will be $(m_0,m_1,m_2,m_3\ldots..m_{L-1})$ where L is the length of input message bits . Therefore, we can obtain our output by convoluting the input and generator polynomial , $u(n) = \sum^{x}_{k=0} m(k)*g_k(n)$, where k = 0,1,2…..L-1[4].

Viterbi Decoding Algorithm

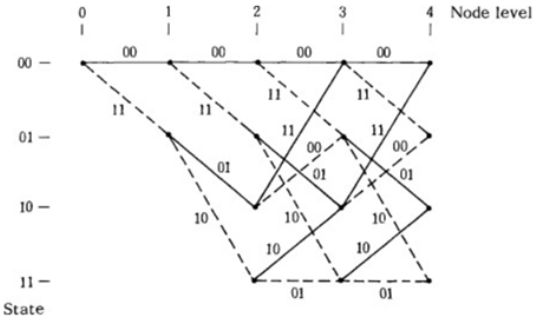

Figure 2 Viterbi algorithm trellis diagram [9]

Viterbi algorithm makes use of the trellis diagram to perform maximum likelihood decoding[5]. The algorithm operates on the basis that it goes through the trellis one stage at a time and will search for the most likely path and discard all other paths. $M_t(i)$ is the path metric at i-th node, and t-th stage in metric .At the beginning, set t=0 and $M_0(0)=0$. As we go along the trellis structure , at each stage of t and i , compare and find the number

of same bits between the number of received bits and code bits . The result is then added to the metric . At each stage , we will pick the mostly likely path which carries the largest value of metric and delete the other paths . This process will end when the path finally converges to its origin state and only one path will remain through the trellis . We will then find the path with the largest metric value and read the data bits from the trellis for decoding [6] .

Experiment Using Matlab

Performance of convolution code depends on the constraint length and coding rate . When the constraint length is longer, that means there are more parity bits and therefore is a more powerful code . However , a longer constraint length will result in a more complex decoder required and a longer delay . The smaller coding rate will also produce more powerful code due to extra redundancy . However , a smaller coding rate will cause less bandwidth efficiency as more bits need to be sent .

We can simulate and verify the performance of convolution code in Matlab . First, we will introduce random bit data sequences and pass them through the AWGN channel and determine the Bit Error Rate BER after decoding . Here, we will use the timer to record the time used for decoding .

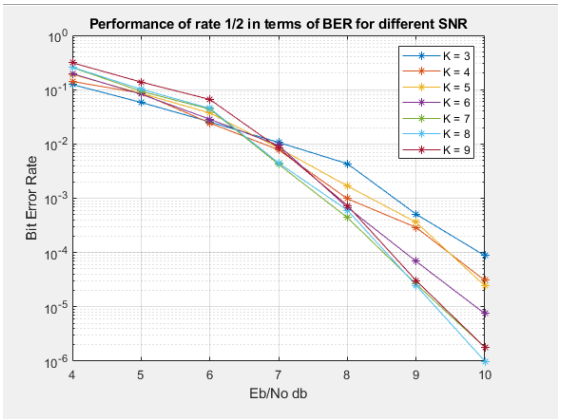

Figure 3 : Performance of CC's in terms of BER for different SNR at increasing constant

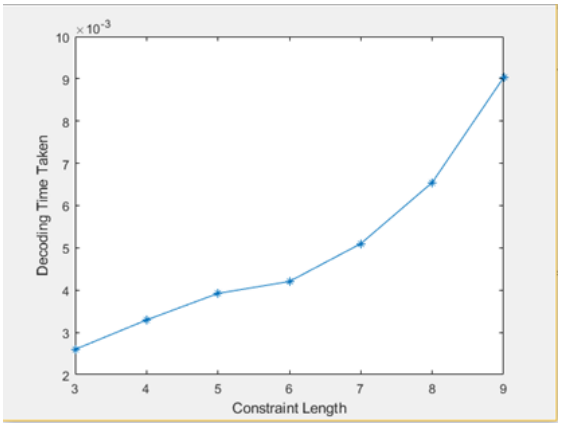

Figure 4 : Decoder time (in seconds) for a single frame data against constraint.

In figure 3, we can deduce a general conclusion that larger constraint length has a better performance at higher Eb/No in which they have a lower error bit rate . However , larger constraint length means a more complex decoder is required and hence will take a longer time to decode. As shown in figure 4 , the decoding time taken increases as the constraint length increases which indicates that complexity increases as well . Hence , there's a compromise for decoding time taken if we want a better performance of convolutional code or vice versa. If the delay is the priority , constraint length must be shorter and on the other hand the constraint length must be large enough if the quality and performance is prioritized.

Conclusion

Through this project , we got to learn the encoding and decoding process of convolutional coding . This project offered us a better understanding of the relationship between the performance , complexity and delay in decoding of convolution code . It's impossible that we achieve high performance with low complexity and low delay as there must be a cost to pay for when we choose one of the aspects that we want to maximise . Therefore as an engineer , we would always attempt to obtain optimum results after considering every aspect of the problem without neglecting the other factors.

References

[1] L. K. S. Tolentino, I. C. Valenzuela, and R. O. S. Juan, “Overhead Interspersing of Redundancy Bits Reduction Algorithm by Enhanced Error Detection Correction Code,” *Journal of Engineering Science and Technology Review*, vol. 12, no. 2, pp. 34–39, Apr. 2019.

[2] Baby, R. A. (2015). Convolution coding and applications: A performance analysis under AWGN channel. 2015 International Conference on Communication Networks (ICCN), 84–88. https://doi.org/10.1109/ICCN.2015.17

[3] Halonen, Timo, Javier Romero, and Juan Melero, eds. GSM, GPRS and EDGE performance: evolution towards 3G/UMTS. John Wiley & Sons, 2004. - p. 430

[4] Yin Xiang , Fang Li . Coding via Random Convolution .

[5] H. Li and R. Kohno, "An Efficient Code Structure of Block Coded Modulations with Iterative Viterbi Decoding Algorithm," *2006 3rd International Symposium on Wireless Communication Systems*, Valencia, 2006, pp. 699-703, doi: 10.1109/ISWCS.2006.4362391

[6] A. J. Viterbi, "A personal history of the Viterbi algorithm," in IEEE Signal Processing Magazine, vol. 23, no. 4, pp. 120-142, July 2006, doi: 10.1109/MSP.2006.1657823.

[7] Sweeney, P. (2002). Error control coding. Chichester: Wiley.

[8] Au.mathworks.com. (2019). Convolutionally decode binary data using Viterbi algorithm - MATLAB vitdec- MathWorks Australia. [online] Available at: https://au.mathworks.com/help/comm/ref/vitdec. html [Accessed 25 May 2019]

[9] Takahiro Yamada (1990) Essential of Error Control Coding https://www.sciencedirect.com/topics/engineering/constraint-length

Appendix

The generator polynomials code with rate ½ was utilised for the simulation (result : figure 3) were obtained from [7].

K	g(1)	g(0)
3	7	5
4	17	15
5	35	23
6	75	53
7	171	133
8	371	247
9	735	561

Table A1 : Polynomials generator with ½ rate. It is used to connect the decoding process with the constraint length.

Appendix 2

Convolution_main, code to initialize the convolutional coding simulation.

```
clc; clear all; close all
%% ECE3141 - Information and Networks Project : Convolutional Coding
%S2 2020 Monash University Malaysia
% modified codes from (Au.mathworks.com)
% modified by Eccel Gunawan, Chua Kah Liang
rng default
ModOrder = 64;                  % Modulation order
m = log2(ModOrder);             % Bits per symbol
EbNoVec = (4:10)';              % Eb/No values (dB)
numSymFrame = 1000;             % Number of QAM symbols per frame

% Following trellis adapted from: (Sweeney, 2002)

trellis1 = poly2trellis(3,[7 5]);
trellis2 = poly2trellis(4,[17,15]);
trellis3 = poly2trellis(5,[35 23]);
trellis4 = poly2trellis(6,[75,53]);
trellis5 = poly2trellis(7,[171 133]);
trellis6 = poly2trellis(8,[371,247]);
trellis7 = poly2trellis(9,[753 561]);
% Traceback Length
tbl = 32;
trellis = [trellis1; trellis2; trellis3; trellis4; trellis5; trellis6;
trellis7];
%% Initialize BER result vectors
berEst = zeros(length(EbNoVec), length(trellis));

%% Calling function to start convolutional coding in 10 iteration
for x = 1:10
   [berEst_func, time] = Convolution_function(trellis, EbNoVec, berEst, tbl,
m, numSymFrame, ModOrder);
   x = x+1;
end
%% Plots
% plotting performance rate in BER for several SNR into graph
for tr = 1:length(trellis)
    graph(tr) = semilogy(EbNoVec,berEst_func(:,tr),'-*');
    hold on
end

% Graphs Customization
title('Performance of rate 1/2 in terms of BER for different SNR');
grid;
xlabel('Eb/No db');
ylabel('Bit Error Rate');
legend(graph,'K = 3','K = 4','K = 5','K = 6','K = 7','K = 8','K = 9');
hold off

% Plot of estimated decoding time
figure
title ('estimated decoding time');
plot([3, 4, 5, 6, 7, 8, 9], time,'-*');
xlabel('Constraint Length');
ylabel('Decoding Time Taken');
```

“Convolution_function”(adapted from [7]) is to simulate the performance of the convolution codes in rate of ½ with multiple constraints.

```
%%

function [berEst_Func, time_plot] = Convolution_function(trellis, EbNoVec,
berEst, tbl, m, numSymFrame, ModOrder)
%% Inititalize variables
targeterrors =100;
maxnumTransitions =1e7;
timer = 0;
numFrames = 0;
for i = 1:length(trellis)
     k = log2(trellis(i).numInputSymbols);
     n = log2(trellis(i).numOutputSymbols);
     r = k/n;
%% Converting EbNoVec to form of SNR
    for j = 1:length(EbNoVec)
     % Convert Eb/No to SNR
     snr_dB = EbNoVec(j) + 10*log10(m*r);
     % Noise variance calculation for unity average signal power.
     noiseVariance = 10.^(-snr_dB/10);
     % Reset number of errors and bit counter
     [numErrors,numBitsInFrame] = deal(0); %optimizing processing speed
%% Generating Binary data,convert to symbol convolution encoding and QAM
modulate
     while numErrors < targeterrors && numBitsInFrame < maxnumTransitions

        % Generate binary data (random bits) and convert to symbols
        data_In = randi([0 1],numSymFrame*m,1);
        % Convolutionally encode the data(binary message) using matlab
        % built-in Function convenc
        data_Enc = convenc(data_In,trellis(i));
        % QAM modulate
        txSignal =
qammod(data_Enc,ModOrder,'InputType','bit','UnitAveragePower',true); %RX=
receive, tx = transfer
        % Pass through AWGN channel
        rxSignal = awgn(txSignal,snr_dB,'measured'); %Using Additive White
Gaussian Noise
        %demodulate signals that contains noises using bit and approximate
LLR
        rx_Data =
qamdemod(rxSignal,ModOrder,'OutputType','approxllr','UnitAveragePower',true,'
NoiseVariance',noiseVariance);
        tic %Start timer
        %using viterbi algorithm to decode data that has been demodulated
        %using matlab built-in function vitdec
        decData = vitdec(rx_Data,trellis(i),tbl,'cont','unquant'); % Decoded
data
        % Stop timer
        timer = timer + toc;
            for x = 1:10
            %calculating number of bit errors in frame and adjusting
            %decoding delay where it is equivalent with the depth of
            %traceback
                numErrsInFrame = biterr(data_In(1:end-
tbl),decData(tbl+1:end));
                numErrsInFramefin(x) = numErrsInFrame;

            end



            % Increment the error and
            numErrors = mean(numError
            numBitsInFrame = mean(num
            numFrames = numFrames + 1

      end

    % Estimate the BER for both metho
    berEst_Func(j,i) = mean(numErrors

    end

time_plot(i) = timer/numFrames;

end
```