Assignment 1, Digital communication 1TE747,VT24

Anton Blaho Mildton
Department of Electrial engineering
Uppsala University
`Anton.blahomildton@gmail.com`

Gustaf Löfdahl
Department of Electrial engineering
Uppsala University
`Gustaf.lofdahl@hotmail.com`

1 Introduction

In the era of fast technological advancement, digital communication has emerged as a transformative force. Unlike traditional analog methods, digital communication relies on discrete signals, enabling more efficient and reliable data transmission. This shift has changed how we connect, share information, and collaborate globally. In the initial segment of the Digital Communication course, the focus lies on essential concepts such as quantization and the conversion processes between continuous signals and Gray code. This is within assignment 1 of the course.

2 Assignment 1 answers

In this assignment the signal/data from the load train command in MATLAB is used, see appendix A code. The figure of how the signal looks can be found in figure 1 below.

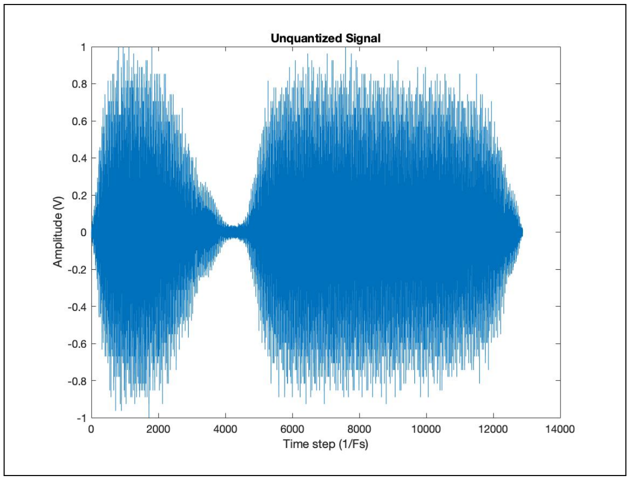


Figure 1: The unquantized signal (train)

The first part of assignment 1 is to write a function in equation 1. The function's goal is to implement a quantizer, which quantifies a unquantizedSignal (the train signal), based on a given quantification levels (V_p) and amount of bits 2^N (N changes the bit amount). When the function is called, it should provide the user with the quantizedsignal,variance of the linear error, variance of the saturation error and finally the Signal to quantization Noise power Ratio.

$$[quantizedSignal, varLin, varSat, SNqR] = MyQuantizer(unquantizedSignal, Vp, N) \tag{1}$$

An illustration of the quantized signal vs the unquantized signal is given in the figure below. On the x axis is the Time step ($1/F_s$), where F_s is the sample rate and on the y axis is the amplitude.

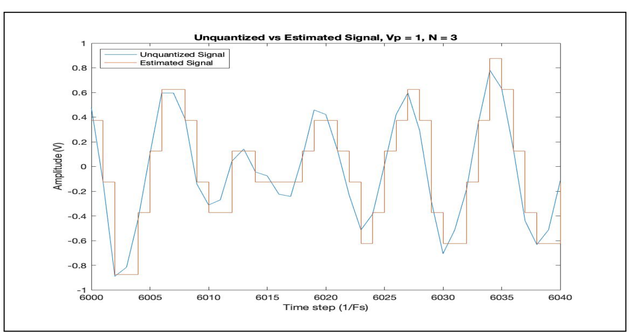


Figure 2: Quantized signal vs the Unquantized signal, N=3 and V_p=1

The second part of the assignment was to build a function that transforms the quantized signal to (binary reflected) gray-code and then create a decoder which transforms the gray-coded signal to a estimation signal (DAC). The two functions is given in equation 2 and 3.

$$[bitStream] = MyGraycode(quantizedSignal, Vp, N) \tag{2}$$

$$[estimatedSignal] = MyDAconverter(estimatedBitStream, Vp, N) \tag{3}$$

As can be seen in the equation the input to the second and third built function is the QuantizedSignal, the Quantizedlevel N bits and the estimatedBitStream. The estimatedBitStream is given from the output of the second equation.

Illustrations of that that both the written functions work can be seen in the figures below, in figure 3 the Quantized signal and in figure 4 the estimated signal.

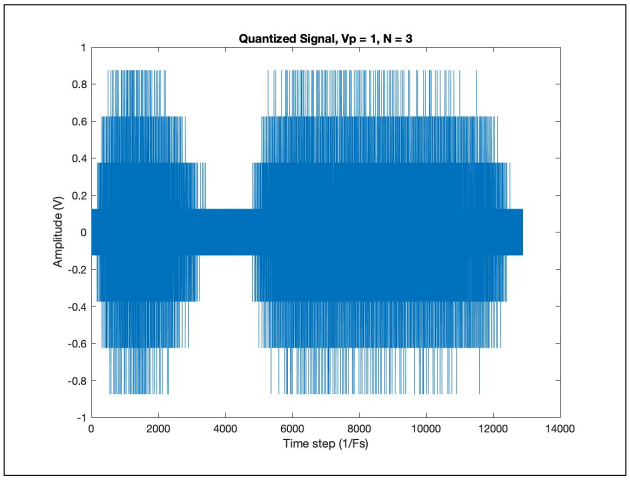


Figure 3: The Quantizied signal, N=3 and V_p=1

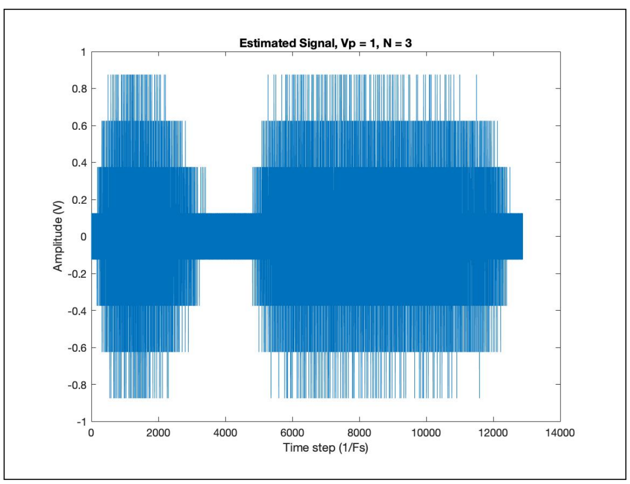


Figure 4: The Estimated signal, N=3 and V_p=1

The final part of the assignment is to analize the output of the function with different inputs. Firstly, N was set to 4 and (V_p) was varied with values. A comparison between the variance of the linear error and the variance of the theoretical error based on the Quantization Step Size (q) function was calculated. The equation used for the calculation is given in equation 4. The result of this can be seen in Table 1. For all the values in Table 1 N was set to 4.

$$LinearErrorVariance = \frac{q^2}{12} \tag{4}$$

Table 1: Theoretical vs simulated error of the signal

V_p	Linear Error (Var)	Theoretical Linear error (Var)
1	0.0013	0.0013
2	0.0057	0.0052
3	0.0130	0.0117
4	0.0241	0.0208

From table 1 it can be concluded that the theoretical error and linear error from the simulation differs more and more when V_p increases.

When listening to the signal one can conclude that a V_p=1 is the most fitted value on that parameter. This is due to the fact that the original train signal is varied between -1 and 1 which makes it natural to have that also as V_p. When the user listen to the sound it feels like when V_p is smaller than 1 it loses some peaks which leads to that high notes is suppressed. If it is too high V_p, then we do not use all the quantization steps which makes the sound choppy.

Furthermore, we set the V_p=1 and then play around with our value on N to see were we can not recognize the signal. When listening to the train signal the conclusion is that it loses it's recognizability when N=2.The SNqR is 13.5157 at this point. When N=5 it can be concluded that then listener can not differ the sound from the original one. SNqR is 50.982 at this point.

In table 2, the parameter N is varied and then the different values on SNqR theoretical and empirical is compared. The SNqR theoretical is calculated using the equation below, were L is given by 2^N.

The assumptions of the theoretical SNqR is that original signal has a mean of 0 and that the signal is uniform distributed between V_p and -V_p. In our case the mean is very close to 0 and the distribution is uniform distributed which would suggest a good approximation for the theoretical SNqR. Additionally, the assumption is that there is no saturation error.

$$SNqR = L^2 \tag{5}$$

Table 2: Theoretical vs empirical SNqR of the signal

N	SNqR Theoretical (dB)	SNqR Empirical (dB)
1	12.0412	0.6763
2	24.0824	13.5157
3	36.1235	26.1204
4	48.1648	38.6720
5	60.2060	50.9819
6	72.2472	62.8676
7	84.2884	74.7407

When comparing the SNqR for each value of N, we find the empirical decibel value to be 1/20 of the theoretical for N=1. With each step of N does the empirical value increase closer to the theoretical as N grow larger.

To explain our findings, we start with producing a histogram of the analog signal, for the original signal and when N=3 and N=7, figure 5 and 6.

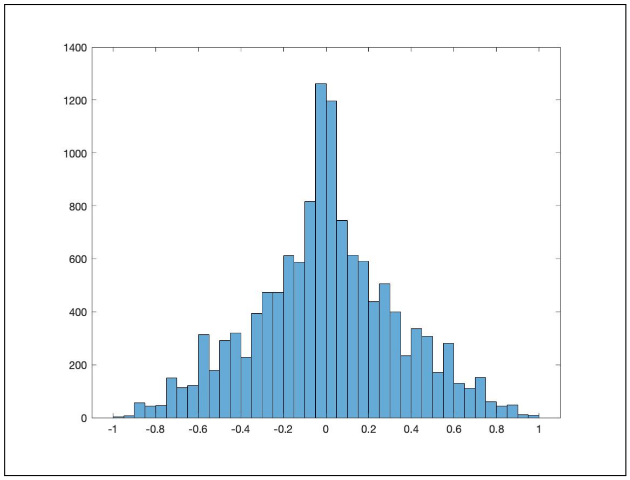
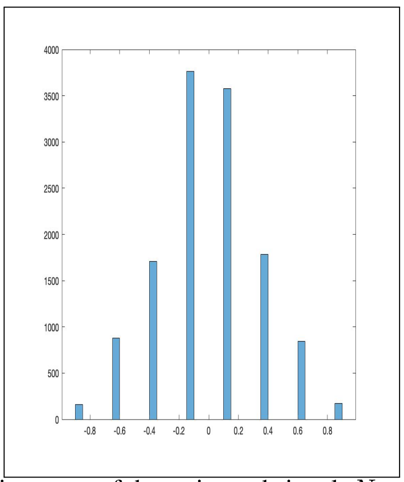
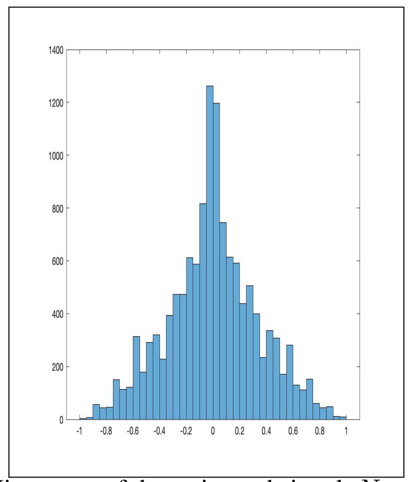


Figure 5: Histogram of the original signal



Histogram of the estimated signal, $N = 3$ and $V_p = 1$



Histogram of the estimated signal, $N = 7$ and $V_p = 1$

Figure 6: Combined caption for the histogram for N=3 and N=7.

As can be seen in the figures when is low, it has just a frequency response at the quantization levels (Figure 6, N=3) and when N is higher then more and more of the original spectra is captured. With a higher N the capturing of the whole spectra leads to that the sound looks very similar to the original one.

References

[1] *Wireless Communication, Andrea Goldsmith*, Available at: `http://fa.ee.sut.ac.ir/Downloads/AcademicStaff/1/Courses/7/Andrea%20Goldsmith-Wireless%20Communications-Cambridge%20University%20Press%20%282005%29.pdf`, Accessed: January 29, 2024.

A Appendix

```
clc
clearvars
close all

load train;
unquantizedSignal = y;
mean = mean(y);
Vp = 1;
N = 4;


[quantizedSignal,varLin,varSat,varTeo,SNqR,SNqRTeo] = MyQuantizer(unquantizedSignal,Vp,N);
estimatedBitStream = MyGraycode(quantizedSignal,Vp,N);
estimatedSignal = MyDAconverter(estimatedBitStream,Vp,N);

sound(estimatedSignal)

%results
figure
plot(unquantizedSignal)
title("Unquantized Signal")
ylabel("Amplitude (V)")
xlabel("Time step (1/Fs)")

figure
plot(quantizedSignal)
title(['Quantized Signal, Vp = ',num2str(Vp),', N = ',num2str(N)])
ylabel("Amplitude (V)")
xlabel("Time step (1/Fs)")

figure
plot(estimatedSignal)
title(['Estimated Signal, Vp = ',num2str(Vp),', N = ',num2str(N)])
ylabel("Amplitude (V)")
xlabel("Time step (1/Fs)")

x = 6000:6041;
figure
plot(x,unquantizedSignal(6000:6041))
hold on
stairs(x,estimatedSignal(6000:6041))
xlim([6000,6040])
ylim([-Vp,Vp])
title(['Unquantized vs Estimated Signal, Vp = ',num2str(Vp),', N = ',num2str(N)])
legend("Unquantized Signal","Estimated Signal")
legend('Location','northwest')
ylabel("Amplitude (V)")
xlabel("Time step (1/Fs)")

% figure
% plot(SNqR)
% title(['Signal to Noise Ratio, Vp = ',num2str(Vp),', N = ',num2str(N)])
% ylabel("Amplitude (dB)")
```

```
% xlabel("Time step (1/Fs)")

function [quantizedSignal,varLin,varSat,varTeo,SNqR,SNqRTeo] = MyQuantizer(unquantizedSignal,Vp,N)
    %quantization setup
    levels = 2^N;
    step = 2*Vp/levels;
    varTeo = step^2/12;
    SNqRTeo = mag2db(levels^2);
    level = linspace(-(levels/2-0.5)*step,(levels/2-0.5)*step,levels);
    quantizedSignal = nan(1,length(unquantizedSignal));

    %quantization
    for i = 1:length(unquantizedSignal)
        for j = 1:length(level)
            if unquantizedSignal(i) <= level(j)+step/2 && unquantizedSignal(i) > level(j)-step/2
                quantizedSignal(i) = level(j);
            end
        end
        if unquantizedSignal(i) < level(1)
                quantizedSignal(i) = level(1);
        elseif unquantizedSignal(i) > level(length(level))
                quantizedSignal(i) = level(length(level));
        end
    end

    %Variance linear
    varLin=var(quantizedSignal.' - unquantizedSignal);

    % Saturated error variance
    satError = quantizedSignal.' - unquantizedSignal;
    satError = min(max(satError, -Vp), Vp);
    varSat = var(satError);

    % Signal to Quantization Noise power Ratio (SNqR) in dB
    SNqR = 20 * log10(var(unquantizedSignal) ./ varSat);
end

function [bitStream] = MyGraycode(quantizedSignal,Vp,N)
    levels = 2^N;
    step = 2*Vp/levels;
    bitStream = nan(1,N*length(quantizedSignal));
    for i = 1:length(quantizedSignal)
        bit = (quantizedSignal(i)+(levels/2-0.5)*step)/step;
        bin = dec2bin(bit,N);
        bitStream(1,1+N*(i-1)) = str2double(bin(1));
        for j = 2:N
            bitStream(1,j+N*(i-1)) = bitxor(str2double(bin(1,j)), str2double(bin(1,j-1)));
        end
    end
end

function [estimatedSignal] = MyDAconverter(estimatedBitStream,Vp,N)
    levels = 2^N;
    step = 2*Vp/levels;
    level = linspace(-(levels/2-0.5)*step,(levels/2-0.5)*step,levels);
    signalbits = nan(1,length(estimatedBitStream));
    estimatedSignal = zeros(1,length(estimatedBitStream)/N);
    for i = 1:length(estimatedSignal)
        signalbits(1+N*(i-1)) = estimatedBitStream(1+N*(i-1));
```

```
        for j = 2:N
            signalbits(j+N*(i-1)) = bitxor(estimatedBitStream(j+N*(i-1)),signalbits(j+N*(i-1)-1));
        end
        for k = 1:N
             estimatedSignal(i) = estimatedSignal(i) + signalbits(k+N*(i-1))*2^(N-k);
        end
    end
    for l = 1:length(estimatedSignal)
        estimatedSignal(l) = level(estimatedSignal(l)+1);
    end
end
```

Listing 1: MATLAB code