COMMUNICATION SYSTEM - DR.HADI

digital baseband communication system

Autumn 1401

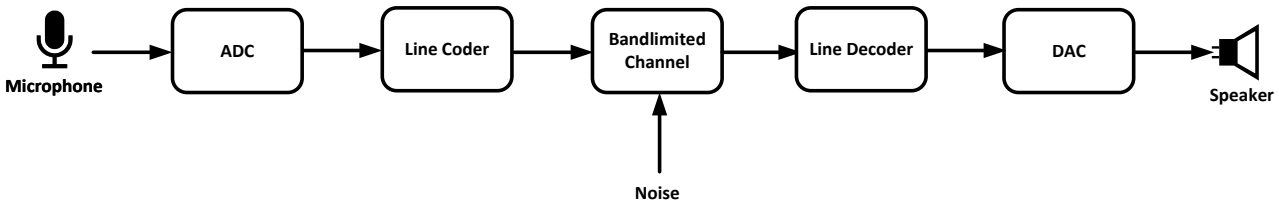
Amirhosein Moraveji, Arsham Lolohari, Mohamad Hosein Faramarzi

5. februar 2023

Innhold

1 Introduction

in this project we want to simulate a realtime digital baseband communication system in MATLAB.this implementation is based on analog to digital conversion and digital communication subjects that we learned about them in last seasons of course.this communication system is made of few parts including ADC(analog to digital converter) , line decoder ,bandlimited channel ,line decoder and finally DAC(digital to analog converter) ,we describe all in the following.our purpose in this project is implementing a system that gives a sound signal through Microphone and after passing this signal through communication system ,deliver it to the speaker. the general block diagram of this system is shown in Fig. 4



Figur 1: Block diagram of a digital baseband communication system.

1.1 ADC

ADC generates a PCM voice with sampling rate f_s and number of quantization bits ν.this part includes sampling , quantization ,and encoding together.the signal we give from microphone enters to ADC.this signal is already sampled and discrete so the sapling we use is actually re-sampling that change sample rate of signal.then we use quantizer and encoder to convert the primitive signal to a digital signal with sampling rate f_s and number of quantization bits ν.note that the new f_s should be greater than early f_S

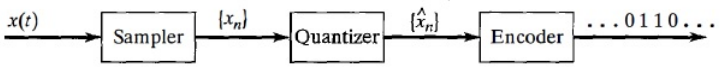


Figure: Block diagram of analog to digital converter.

1.2 line coder

A digital pulse amplitude modulation signal is expressed as:

$$x(t) = \sum_{-\infty}^{\infty} a_k p(t - kD)$$

The process of assigning pulses to the digital data is called line coding .Line coder generates a binary polar Nyquist line code with rolloff factor β, baud rate r, and amplitude $\pm A$.

The baud or symbol rate of a PAM signal is defined as:

$$r = \frac{1}{D}$$

in line coding the p(t) has this equation :

$$p(t) = \frac{cos(2\pi\beta t)}{1-(4\beta t)^2} sinc(rt)$$

1.3 Bandlimited Channel

Channel adds additive zero-mean Gaussian noise with variance $\sigma^2 = N_0 B$, where B is the channel bandwidth and N_0 denotes noise power spectral density. this channel is bandlimited so we should apply an ideal lowpass filter with bandwidth B too.

1.4 Line decoder

Line decoder has a perfect synchronization circuit.in this block first we sample input signal with new rate F_s and then it check the amplitude of signal to be positive or negative to find that the digitilized signal was 0 or 1.

1.5 DAC

the output of Line decoder is a digital sequence.it gives that binary signal then separate the bits of each level and through reading this separated bits reconstructs the amplitude of related level.this reconstruction can be a constant Coefficient with different quantized Initial value.

2 codes and implementation algorithms

at first we introduce functions of each block with description and then we introduce other parts and syntaxes we used in our implementation.

2.1 main

this function is the general overview of the system that includes all parts of the system in itself.this function gives all required inputs of all other functions and call every function in itself:

```
%% main function
function [outputSignal,x1,x4] = main(inputSignal ,
   inputSampleRate , requiredSampleRate , numberOfBits ,  beta , A
    , continuous_fs , N0 , B)
    r = numberOfBits*requiredSampleRate ;
    x1 = adc(inputSignal , inputSampleRate , requiredSampleRate ,
    numberOfBits );
    x2 = linecoder(x1 , beta , r , A , continuous_fs);
    x3 = channel(x2 , N0 , B , continuous_fs);
    x4 = linedecoder(x3 , r , continuous_fs );
    outputSignal = dac(x4 , numberOfBits , requiredSampleRate ,
   continuous_fs);
end
```

2.2 ADC

this part includes few functions that are related to each other.the functions are including changesamplerate , digitalize ,quantize , changeAmplitude.

```
%% ADC function
function y = adc( inputSignal , inputSampleRate ,
   requiredSampleRate , numberOfBits );
    x = changeSampleRate(inputSignal , inputSampleRate ,
   requiredSampleRate);
    y = digitalize( x , numberOfBits);
end
```

the changesamplerate function gives main signal and its early sample rate and change its sample rate with following algorithm:

```
function y = changeSampleRate( signal , f1 , f2) % change sample
   rate from f1 to f2
    L = length(signal);
    T = L/f1 ; % duration of signal
    Ly = round(T*f2) ; % length of y
    t = round((0:1:Ly-1)*f1/f2)+1; % scale index
    y = signal(t);
end
```

the digitalize function calls quantize and de2bi(decimal to binary) functions and change the sampled signal to a sequences of bits:

```
%% digitalize and quantize and changeamplitude functions
function y = digitalize( signal , numberOfBits) % find binary
    code for one quantized number
     x = quantize( signal , numberOfBits);
     z = de2bi( x , numberOfBits );
     y = reshape(z.' , length(signal)*numberOfBits , 1);
end
function y = quantize( signal , numberOfBits )
     n = 2^(numberOfBits-1) ; % n = number of codes / 2
     x = changeAmplitude( signal )*n; % x => (-n , n)
     y = (floor(x)+n); % x => [0 , 1 , ... , 2n-1]
end
function y = changeAmplitude( x ) % change amplitude of x to a
    little less than 1
     y = x/(max(abs(x),[],'all')*1.0001);
end
```

2.3 line coding

in line coding as we said in introduction section we have a convolution with p(t) that we described in introduction.in this part at last we remove negative parts. other processes is clear based on the code we bring in the following:

```
%% line coding function code
function lineCode_x = linecoder(signal, beta,r,A,continuous_fs)
     %*****better to choose r as a fraction of system_fs, for
    example r=system_fs/1000******
     D = 1/r;
     L = length(signal);
     T = D*(L-1);
     t = 1/continuous_fs:1/continuous_fs:T; %WARNING: THIS t IS
    DIFFERENT FROM t OF THE MAIN SIGNAL!!!
     t_len = length(t);
     t_posNeg=cat(2,-fliplr(t),0,t); %t_posNeg contains time from
    -t to +t

     lineCode_x_ampl = (signal-0.5)*2*A; %creating amplitude +-A
    from bits 1&0


     lineCode_x_zeroIntrp = zeros(t_len+1,1);
     %lineCode_x_zeroIntrp(1:t_D_len:end) = lineCode_x_ampl; %zero
     interpolation for convolving
     for i=1:1:L
         lineCode_x_zeroIntrp(round((i-1)*continuous_fs/r)+1)=
    lineCode_x_ampl(i);
     end

```

```
    p = cospi(2*beta*t_posNeg) ./ (1-(4*beta*t_posNeg).^2) .*
   sinc(r*t_posNeg); %p(t), a column vector. length = 2*length(t)
   +1 = length(t_posNeg)
    p=p.'; %converting to row vector

    %lineCode_x = zeros((encode_vect_len-1)*t_D_len+1,1);
    %disp("line 190");
    lineCode_x = conv(lineCode_x_zeroIntrp,p);
    lineCode_x = lineCode_x(t_len+1:2*t_len+1); %removing
   negative parts and last part (that exceeds t_len)

    %******* at the end, sample 1 from encode_vect corresponds to
    sample 1
    %from lineCode_x (or t), sample 2 corresponds to sample 101
   from
    %lineCode_x,... and sample i corresponds to i*(t_D_len-1)+1
   **********

    %*** lineCode_x starts with the first bit peak and finishes
   with last bit peak ***
end
```

2.4 channel

in bandlimited channel we described as channel function first with syntax "wgn"we add an zero mean Gaussian noise to our signal and then we use an ideal filter with bandwidth B.

```
%% channel function code
function y = channel(signal , N0 , B , fs)
    noiseVar = N0*B ;
    noise = wgn(size(signal,1),size(signal,2),noiseVar,'linear');
    % noise = sqrt(noiseVar)*randn(size(signal,1),size(signal,2))
   ;
    y = signal + noise;
    L = length(y);
    if fs>B
        f = fft(y);
        n = round(B/(fs/2)*L);
        f(n+2:L-n) = 0;
        y = ifft(f);
    end
end
```

2.5 line decoder

in line docoder we do not have an special process and as we said in introduction we just change the sample rate of signal and then make a sequence of 0 and 1 based on amplitude of signal.

```
%% line decoder function code
```

```
function y = linedecoder( signal , r , continuous_fs )
    x = changeSampleRate(signal , continuous_fs , r); % sampling
    L = length(x);
    y = zeros(L,1);
    y(x>0)=1; % binary signal
end
```

2.6 DAC

at last we have digital to analog converter that in this function we change sample rate of smpled signal with a High new f_s and use binary to decimal process to reconstruct an analog signal.

```
%%DAC function code
function z = dac(signal , numberOfBits , Fsignal , continuous_f)
    L = length(signal);
    Ly = round(L/numberOfBits);
    x = reshape(signal , numberOfBits , []); % divide symbols
    y = bi2de(x.')+(-2^(numberOfBits-1)+0.5); % decode binary to
   quantized numbers
    z = increaseSampleRate( y , Fsignal , continuous_f);
    z = z/max(abs(z),[],'all');
    % lowpass filter
    f = fft(z);
    n = round(4000/continuous_f/2*L);
    f(n+2:L-n) = 0;
    z = real(ifft(f));

end
function y = increaseSampleRate( signal , f1 , f2) % change
   sample rate from f1 to f2
    L = length(signal);
    T = L/f1 ; % duration of signal
    Ly = round(T*f2) ; % length of y
    t = round((0:1:L)*f2/f1)+1; % scale index
    y = zeros(Ly,1);
    for i=1:1:L
        y(t(i):t(i+1)-1) = signal(i);
    end
end
```

3 plots and results and questions

3.1

a)Write a MATLAB code to simulate the communication system. Create separate functions for each block and then, connect them in a main mfile. Name the functions adc, linecoder, channel, linedecoder, and dac. Each function might have an arbitrary number of input arguments. For example, adc function might accept ν and f_s as its input arguments.
in previous section we describe all codes and functions clearly and we explain about blocks and etc.

3.2

b)Feed your simulation setup with a recorded audio file and play the received voice and hear it for different noise level N_0 and channel bandwidth B. Assume that the ADC/DAC parameters are suitably tuned such that aliasing and SQNR have acceptable values and the channel bandwidth is enough such that no ISI appears. How do you feel when you hear the received voice? Note that you can record your voice from your laptop microphone and feed it to the communication system. You can also hear the received voice via your laptop speaker. MATLAB has useful internal commands for working with microphones and speakers!

we control the effect of noise with parameter N_0.first we put 0 instead of N_0.the output voice is in file "outvoice _ N0_ 0.mat".we repeat this saving files with different values of N_0 and save the results in "outvoice _ N0_ 1e-7.mat"and "outvoice _ N0_ 1e-5.mat".by hearing the results with different noises we find that with N0=0 the sound is mostly clear. but by increasing N0 the result begin to become noisy and unrecognizable.

for effect of B we change value of B from 15000 to 40000 and the results is kept in "outvoice _ B_ B_value.matthat we attached in file of project to.by increasing the B the quality of sound become more unrecognizable

3.3

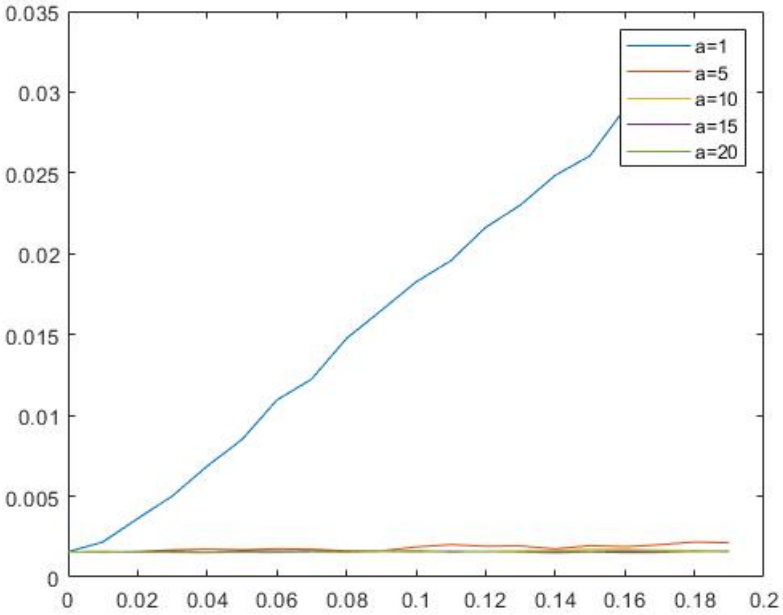
c)Use the simulation setup to calculate the BER of the system and plot it in terms of noise variance σ^2 and amplitude A. Assume that the ADC/DAC parameters are suitably tuned such that aliasing and SQNR have acceptable values and the channel bandwidth is enough such that no ISI appears. Discus the obtained plots.

in this part we plot 2 charts to show the effect of values of A and σ^2 on BER in fig2:

3.4

d)Repeat the previous two parts when the channel bandwidth limitation imposes ISI. Investigate the impact of rolloff factor β on the imposed ISI.

in this part we repeat out process but now with ISI and the result is shown in fig3:



Figur 2: BER plot based on a value of A

3.5

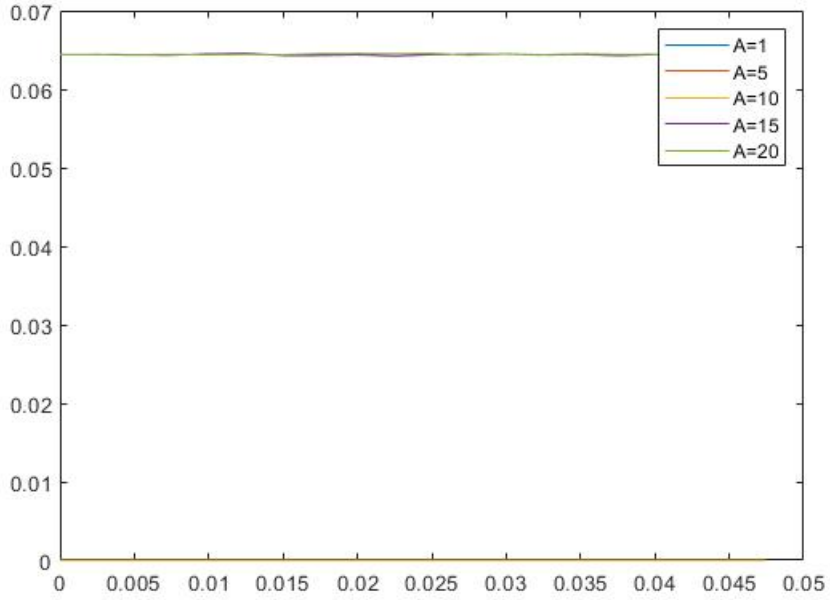
e)Investigate the impact of the sampling rate f_s and number of quantization bits ν on the performance of the communication system.

for this section we give output with different f_s in outvoice_fs_value_e and with different ν in few files named in outvoice_vnumber_e formats and we attached the files in project folder. as we knew f_s and ν increase the quality of sounds because this 2 parameter can improve the performance of ADC and DAC and we can keep the information of analoge signal better so as we expect the quality of song become more clear when we increase the sample rate of number of bits.

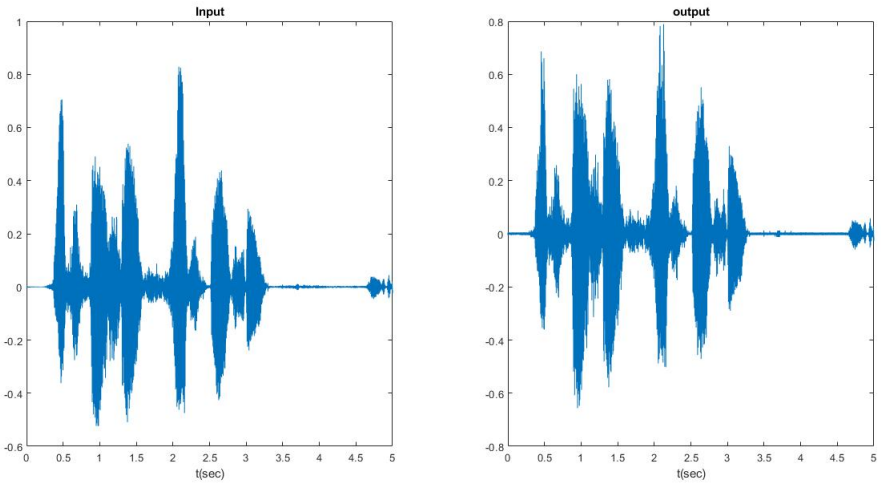
3.6

f)Prepare a short report and describe your work concisely. Use suitable figures to better describe the developed codes and to make your report more readable and understandable. Attach samples of the recorded audios as well as the developed codes to your sent report.

the input and out put of this system is shown in fig4.this plots show the similarity between input voice that we record with microphone and the output of communication system



Figur 3: BER plot based on value of A with ISI



Figur 4: input and output of communication system