Alexandria University
Faculty of Engineering
Electrical and Electronics Engineering Department
Fall semester, 2022/2023

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
الفصل الدراسى الثانى , 2022/2023

INTRO TO DIGITAL COMMUNICATIONS LAB

سيف الدين زكريا محمد	19015813
محمد عيد عبد المجيد	19016463
تهاني ياسر مرسى أحمد	18010503

Lab 2

Pulse Code Modulation (PCM)

Objective:

(1) Investigate the PCM system components.

(2) Investigate the effect of changing the number of levels.

(3) Investigate the oversampling, critical sampling and undersampling cases.

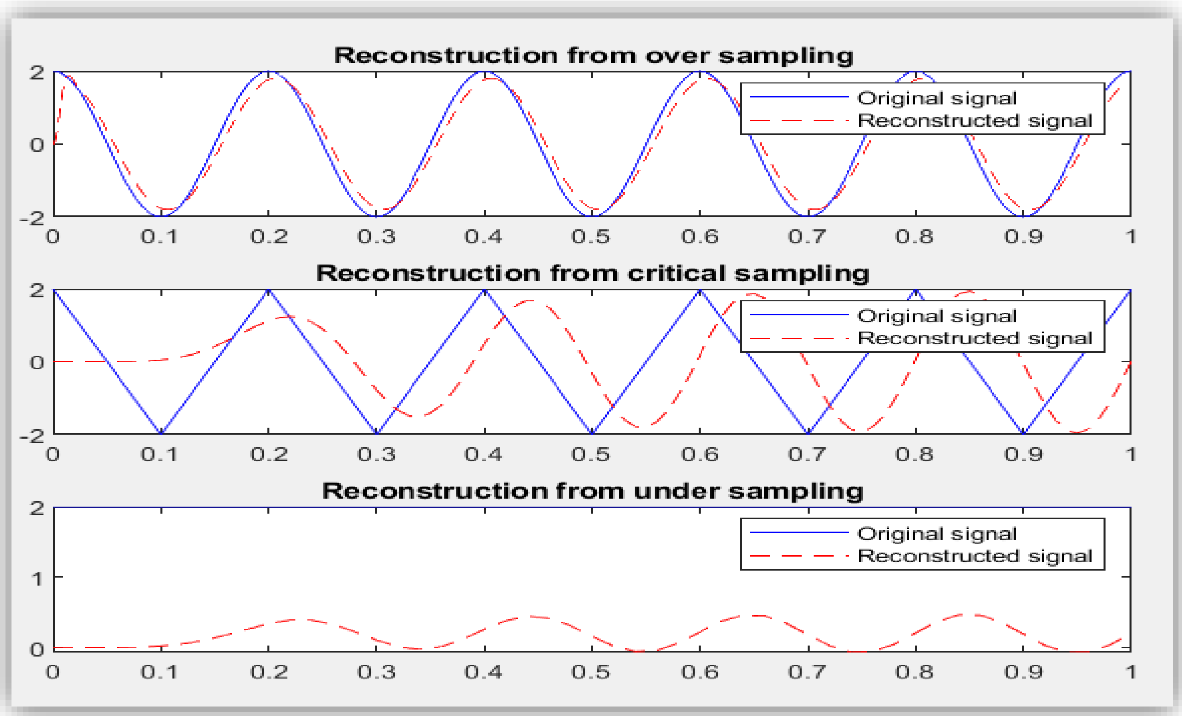
(4) Calculate the quantization error of the transmitted signal.

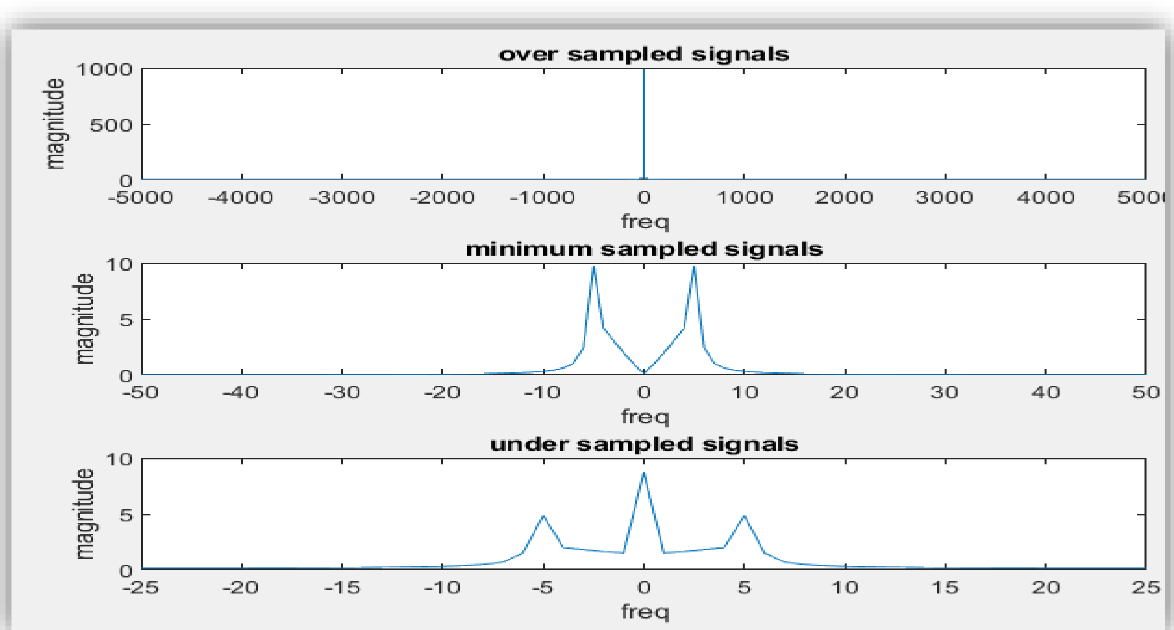
THE SAMPLING CODE :

```
clear, clc ,close all;

% reconstruction from oversampling
t1=0:0.001:1;                          % time signal
y=2*cos(2*pi*5*t1);
[B,A] = butter(3,1000/100000,'low' ); % butterworth  filter
zero_added_signal=zeros(1,length(y)*10);
for i=1:length(y)
    zero_added_signal(i*10)=y(i);
end
zero_added_signal(1:9)=[];
% Adding zeros enhances the signal display and don't change the spectrum, it changes sampling freq. only
t2=linspace(0,1,length(zero_added_signal));
filtered_signal = 9*filter(B,A,zero_added_signal);
figure(1),subplot(3,1,1);
plot(t1, y, 'b', t2, filtered_signal, 'r--');
title('Reconstruction from over sampling');
legend('Original signal', 'Reconstructed signal');

s=fft(filtered_signal);
s=fftshift(s);
fs=10000;
freq=linspace(-fs/2,fs/2,length(s));
figure(2),subplot(311)
plot(freq,abs(s))
xlabel('freq')
ylabel('magnitude')
title('over sampled signals')

% construction from minimum sampling
t1=0:1/(2*5):1;                              % fs=2*fm
y=2*cos(2*pi*5*t1);
[B,A] = butter(10,0.1,'low' );
zero_added_signal=zeros(1,length(y)*10);
for i=1:length(y)
    zero_added_signal(i*10)=y(i);
end
zero_added_signal(1:9)=[];
t2=linspace(0,1,length(zero_added_signal));
filtered_signal = 7*filter(B,A,zero_added_signal);
figure(1),subplot(3,1,2);
plot(t1, y, 'b', t2, filtered_signal, 'r--');
title('Reconstruction from critical sampling');
legend('Original signal', 'Reconstructed signal');
```

```
s=fft(filtered_signal);
s=fftshift(s);
fs=100;            % fs=100 as the signal frequency =10 &and we add 10 zeros between every 2 samples that make
the total frequency =100
freq=linspace(-fs/2,fs/2,length(s));
figure(2),subplot(312)
plot(freq,abs(s))
xlabel('freq')
ylabel('magnitude')
title('minimum sampled signals')

% construction from undersampling sampling
t1=0:0.2:1;
y=2*cos(2*pi*5*t1);
[B,A] = butter(10,0.2,'low' );
zero_added_signal=zeros(1,length(y)*10);
for i=1:length(y)
        zero_added_signal(i*10)=y(i);
end
zero_added_signal(1:9)=[];
t2=linspace(0,1,length(zero_added_signal));
filtered_signal = filter(B,A,zero_added_signal);
figure(1),subplot(3,1,3);
plot(t1, y, 'b', t2, filtered_signal, 'r--');
title('Reconstruction from under sampling');
legend('Original signal', 'Reconstructed signal');

s=fft(filtered_signal);
s=fftshift(s);
fs=50;
freq=linspace(-fs/2,fs/2,length(s));
figure(2),subplot(313)
plot(freq,abs(s))
xlabel('freq')
ylabel('magnitude')
title('under sampled signals')
```

The output:


Reconstruction from over sampling
Original signal
Reconstructed signal
Reconstruction from critical sampling
Original signal
Reconstructed signal
Reconstruction from under sampling
Original signal
Reconstructed signal


over sampled signals
magnitude
freq
minimum sampled signals
magnitude
freq
under sampled signals
magnitude
freq

The PCM Code :

```
clear, clc ,close all;

% Parameters
A = 1;                    % amplitude of sinusoidal wave
f = 2;                    % frequency of sinusoidal wave
Fs = 4000;                % sampling frequency
m = 7;                    % total number of bits, including the sign bit
n = floor((m-1)/2);       % number of bits for integer value and fraction part

% Generate the signal
t1 = 0:1/Fs:1/f;           % time vector
x = A*sin(2*pi*f*t1);      % sinusoidal wave

% Quantize the signal using fi command
vmax = max(abs(x));        % maximum absolute value of input signal (m_P)
q = 2*vmax/(2^n);          % quantization step size
xq = fi(x,1,m,n);          % quantize signal using fi command
xq = double(xq);           % convert to double precision for further processing

% Convert the quantized samples to binary
x_bin = de2bi((xq + 1)*(2/q), m, 'left-msb');

% Calculate the mean square quantization error
n_values = [3, 4, 5, 6, 7, 8, 9, 10];
mse = zeros(size(n_values));
for i = 1:length(n_values)
   n = n_values(i);
   m = 2*n + 1;
   quantized_signal = double(fi(x,1,m,n));
   quantization_error = x - quantized_signal;
   mse(i) = sum(quantization_error.^2)/length(quantization_error);
   fprintf('n = %d, MSE = %f\n', n, mse(i));
end
```

The output:

```
n = 3, MSE = 0.001193
n = 4, MSE = 0.000306
n = 5, MSE = 0.000078
n = 6, MSE = 0.000020
n = 7, MSE = 0.000005
n = 8, MSE = 0.000001
n = 9, MSE = 0.000000
n = 10, MSE = 0.000000
```

PCM using quantizer:

```
clear, clc ,close all;

% Define the parameters
f_s = 4000;       % Sampling frequency (Hz)
f_sig = 2;        % Signal frequency (Hz)
A_sig = 1;        % Signal amplitude (V)

% Generate the time vector
t = 0:1/f_s:1-1/f_s;

% Generate the sinusoidal signal
x = A_sig*sin(2*pi*f_sig*t);

% Define the number of bits for quantization
n_bits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

% Initialize the mean squared error (MSE) vector
mse = zeros(size(n_bits));

% Loop over each number of bits
for i = 1:length(n_bits)
  n = n_bits(i);                                        % Number of bits
  L = 2^n;                                              % Number of quantization levels
  step_size = (2*A_sig)/L;                              % Quantization step size
  partition = [-1:step_size:1];                         % Define the partition for the quantization
  start = -A_sig - step_size;                           % Define the start of the codebook
  codebook = [start:step_size:1];                       % Define the codebook for the quantization
  [index,x_q,distor_linear] = quantiz(x,partition, codebook);   % Quantize the signal
  quantization_error = x - x_q;                         % Calculate the quantization error
  mse(i) = sum(quantization_error.^2)/length(quantization_error); % Calculate the MSE
```

```
  fprintf('n = %d, MSE = %f\n', n, mse(i));  % Print the results
end
```

The output:

```
n = 1, MSE = 0.363881
n = 2, MSE = 0.088820
n = 3, MSE = 0.021817
n = 4, MSE = 0.005385
n = 5, MSE = 0.001334
n = 6, MSE = 0.000331
n = 7, MSE = 0.000083
n = 8, MSE = 0.000021
n = 9, MSE = 0.000005
n = 10, MSE = 0.000001
```

PCM using non-uniform quantizer {using compand}:

```
clear, clc ,close all;

% Parameters
A = 1;                  % amplitude of sinusoidal wave
f = 2;                  % frequency of sinusoidal wave
Fs = 4000;              % sampling frequency
m = 7;                  % total number of bits, including the sign bit
n = floor((m-1)/2);     % number of bits for integer value and fraction part

% Generate the signal
t1 = 0:1/Fs:1/f;            % time vector
x = A*sin(2*pi*f*t1);       % sinusoidal wave

% Quantize the signal using compand command
vmax = max(abs(x));         % maximum absolute value of input signal (m_P)
mu = 255;                   % companding law parameter

% Calculate the mean square quantization error
n_values = [3, 4, 5, 6, 7, 8, 9, 10];
mse = zeros(size(n_values));
```

```
for i = 1:length(n_values)
   n = n_values(i);
   m = 2*n + 1;
   xq = compand(x,mu,vmax,'mu/compressor');% quantize signal using compand command
   xq = fi(xq,1,m,n);        % convert to fi object
   xq = double(xq);              % convert to double precision for further processing
   exband = compand(xq,mu,vmax,'mu/expander');% quantize signal using compand command
   quantization_error = x -exband;
   mse(i) = sum(quantization_error.^2)/length(quantization_error);
   fprintf('n = %d, MSE = %f\n', n, mse(i));
end
```

The output:

```
n = 3, MSE = 0.012561
n = 4, MSE = 0.003661
n = 5, MSE = 0.001016
n = 6, MSE = 0.000272
n = 7, MSE = 0.000071
n = 8, MSE = 0.000018
n = 9, MSE = 0.000005
n = 10, MSE = 0.000001
```