# ECE3101L

**Lab Report # 2**

# Abraham Nkum

# Lab 5– Sampling & Reconstruction

### Sampling pulses, Delta-Dirac functions or square wave pulses

#### Time domain waveform

Show the first 2 cycles of the time domain sinewave on the 3 input scope

A picture containing graphical user interface

Description automatically generated

#### Frequency domain Spectrum

Show the spectrum analyzer frequency spectrum with a span from DC to 500HzA screen shot of a computer

Description automatically generated

Try to identify the sync function modulating the peaks in the frequency spectrum and think about why it is there. Think of a square wave Fourier coefficient mathematical expression

Show the spectrum analyzer frequency spectrum with a span from DC to 61Hz

### Reducing the Spectrum analyzer Resolution Bandwidth

Change the Spectrum Analyzer resolutioA screenshot of a computer

Description automatically generatedn bandwidth to auto (about 0.5Hz) and simulation time to 3.1sGraphical user interface

Description automatically generated

#### Frequency domain Spectrum

Show the “spectrum analyzer” frequency spectrum with a span from DC to 500Hz

Show the “spectrum analyzer” frequency spectrum with a span from DC to 65Hz

### Sampling with a 30% duty cycle Square wave

Change the Spectrum Analyzer resolution bandwidth back to 0.001Hz and simulation time to 1510s

Change the square wave duty cycle to 30%. Note that you will just need to change the number of samples in the pulse generator pulse width

#### Sampling pulse generator settings

Graphical user interface, text, application, email

Description automatically generatedWe will still sample this sinewave at 0.1Hz (100ms period) with pulse generator square pulses but with a 30% duty cycle (or 30ms pulse width) square pulses

Set the pulse width to 30 samples

That will give us 100 samples per cycle for a 100ms period 10Hz square wave and 30 samples per sampling square pulse as a 30% duty cycle will result in a 30ms pulse width.

#### Time domain waveform

Show the first 2 cycles of the time domain sinewave on the 3 input scope

Graphical user interface

Description automatically generated

#### Frequency domain Spectrum

Show the spectrum analyzer frequency spectrum with a span from DC to 500Hz

A screenshot of a computer

Description automatically generated

Note that as you increase the duty cycle of the samples square pulses, the frequency of the sync function modulating the frequency spectrum peaks increases

Show the spectrum analyzer frequency spectrum with a span from DC to 61Hz

### A screenshot of a computer Description automatically generatedReducing the Spectrum analyzer Resolution Bandwidth

Change the Spectrum Analyzer resolution bandwidth to auto (about 0.5Hz) and simulation time to 3.1s

Show the “spectrum analyzer” frequency spectrum with a span from DC to 500HzGraphical user interface, text, application

Description automatically generated

Show the “spectrum analyzer” frequency span from DC to 65Hz

A screenshot of a computer

Description automatically generated

### Sampling with a 50% duty cycle Square wave

Change the Spectrum Analyzer resolution bandwidth back to 0.001Hz and simulation time to 1510s

Change the square wave duty cycle to 50%.

#### Graphical user interface, text, application Description automatically generatedSampling pulse generator settings

We will still sample this sinewave at 10Hz (100ms period) with pulse generator square pulses but with a 50% duty cycle (or 50ms pulse width) square pulses

Set the pulse width to 50 samples

#### Time domain waveform

Show the first 2 cycles of the time domain sinewave on the 3 input scopeA picture containing graphical user interface

Description automatically generated

#### Frequency domain Spectrum

Show the spectrum analyzer frequency spectrum with a span from DC to 500HzA screenshot of a computer

Description automatically generated

Show the spectrum analyzer frequency spectrum with a span from DC to 61HzA screen shot of a computer

Description automatically generated

### Sampling with a 75% duty cycle Square wave

Change the square wave duty cycle to 75%.

#### Sampling pulse generator settings

Graphical user interface, text, application, email

Description automatically generatedSet the pulse width to 75 samples

#### Time domain waveform

Show the first 2 cycles of the time domain sinewave on the 3 input scopeA picture containing graphical user interface

Description automatically generated

#### Frequency domain Spectrum

Show the spectrum analyzer frequency spectrum with a span from DC to 500HzA screen shot of a computer

Description automatically generated

Show the spectrum analyzer frequency spectrum with a span from DC to 61HzA screenshot of a computer

Description automatically generated

### Sampling with a 99% duty cycle Square wave

#### Sampling pulse generator settings

Graphical user interface, text, application

Description automatically generatedSet the pulse width to 99 samples

#### Time domain waveform

Show the first 2 cycles of the time domain sinewave on the 3 input scopeA screen shot of a computer

Description automatically generated

#### Frequency domain Spectrum

Show the spectrum analyzer frequency spectrum with a span from DC to 500Hz

Show the spectrum analyzer frequency spectrum with a span from DC to 61Hz

# Lab 6 – Image & Audio Quantization

## Image quantizing code with/without the matlab imquantize function

clc, clear all, close all;

imgfls\_name= {'moonImage.tif','skullImage.jpg'};

for imgfls\_index=1:length(imgfls\_name)

[imgfls\_content{imgfls\_index}, Fs{imgfls\_index}] = imread(imgfls\_name{imgfls\_index});

max\_value = max(imgfls\_content{imgfls\_index}(:));

min\_value = min(imgfls\_content{imgfls\_index}(:));

range = max\_value - min\_value;

% Shift down the data to start at 0 instead of at min\_value

down\_shifted=imgfls\_content{imgfls\_index}-min\_value;

% Number of bits used to encode the quantization levels

encode\_bits = 1:7;

% Create a figure to plot the quantized images

fsf(2\*imgfls\_index-1)=figure(2\*imgfls\_index-1);

% Enlarge to fullscreen

set(fsf(2\*imgfls\_index-1), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

% Quantization not using matlab's image imquantize function

% run through the i loop for every value of b

for encode\_bit\_index = 1:length(encode\_bits)

levels=2^encode\_bit\_index;

step = range/(levels-1);

% prepare the data to be quantized so it can be rounded

% normalize by the step size

% so every step contains data values of range 1

down\_range=down\_shifted/step;

% round to nearest integer

% if in the middle go to higher integer

rounded=round(down\_range);

% un-normalize by the step size

up\_range=rounded\*step;

% add back the minimum value to create the quantized image to N

% levels

quantized\_image= up\_range+min\_value;

% plot the quantized images

subplot((length(encode\_bits)+2)/3,3,encode\_bit\_index);

imshow(quantized\_image);

title([num2str(encode\_bit\_index) ' bit quantized ' imgfls\_name{imgfls\_index} ' not using quantiz']);

end

% plot the original image

subplot((length(encode\_bits)+2)/3,3,length(encode\_bits)+1);

imshow(imgfls\_content{imgfls\_index});

title(['Original ' imgfls\_name{imgfls\_index}]);

% Quantization using matlab's imquantize function

% Note that imquantize/multithresh allows only up to 20 levels.

for encode\_bit\_index = 1 : 4

numLevels = 2^(encode\_bit\_index);

partition = multithresh(imgfls\_content{imgfls\_index}, numLevels-1);

codebook = [partition max(imgfls\_content{imgfls\_index}(:))];

[imquantized\_image, index] = imquantize(imgfls\_content{imgfls\_index},partition,codebook);

%plot quantized signal-----------------------

fsf(2\*imgfls\_index)=figure(2\*imgfls\_index);

% Enlarge figure to full screen.

set(fsf(2\*imgfls\_index), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot((length(encode\_bits)+2)/3,3,encode\_bit\_index);

imshow(imquantized\_image);

title([num2str(numLevels) ' levels quantized ' imgfls\_name{imgfls\_index} ' using imquantize']);

end

% plot the original image

subplot((length(encode\_bits)+2)/3,3,4+1);

imshow(imgfls\_content{imgfls\_index});

title(['Original ' imgfls\_name{imgfls\_index}]);

end

An old photo of a living room

Description automatically generated

A picture containing photo, different, looking, room

Description automatically generated

## Audio Quantization

clc, clear all, close all;

audfls\_name = {'goreAudio.au','musicAudio.au','speechAudio.au','boygeorgeAudio.au'};

for imgfls\_index=1:length(audfls\_name)

[audfls\_content{imgfls\_index}, Fs{imgfls\_index}] = audioread(audfls\_name{imgfls\_index});

% to find the maximum value of a 2 dimensional array

max\_value = max(audfls\_content{imgfls\_index});

min\_value = min(audfls\_content{imgfls\_index});

% Shift down the data to start at 0 instead of at min\_value

shift=audfls\_content{imgfls\_index}-min\_value;

range = max\_value - min\_value;

encode\_bits = 1:7; % number of bits needed to encode the number of quantization levels

%Quantize not using the matlab 'quantiz' function

fsf(2\*imgfls\_index-1)=figure(2\*imgfls\_index-1);

set(fsf(2\*imgfls\_index-1), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

for encode\_bit\_index = 1:length(encode\_bits)

levels=2^encode\_bit\_index; %number of quantization levels

step = range/(levels-1); %step size or interval between two quantization levels

% prepare the data to be quantized so it can be rounded

% normalize by the step size

% so every step contains data values of range 1

down\_range=shift/step;

% round to nearest integer

% if in the middle go to higher integer

rounded=round(down\_range);

% un-normalize by the step size

up\_range=rounded\*step;

% add back the minimum value

quantized\_audio=up\_range+min\_value;

%plot quantized signal-----------------------

subplot((length(encode\_bits)+1)/2,2,encode\_bit\_index);

plot(quantized\_audio);

title([num2str(encode\_bit\_index) ' bit quantized ' audfls\_name{imgfls\_index} ' not using quantiz' ]);

xlabel('?'); ylabel('?');

axis tight;

if (imgfls\_index==2 && encode\_bit\_index==4)%k is the audio file, i is the number of bits

soundsc(quantized\_audio,16000,16)

% second argument is fs.

% may need to change, eg k=2 needs to be played at 16000

end

end

subplot((length(encode\_bits)+1)/2,2,length(encode\_bits)+1);

plot(audfls\_content{imgfls\_index});

title([' Original ' audfls\_name{imgfls\_index}]);

xlabel('?'); ylabel('?');

axis tight;

%Quantize using quantiz

fsf(2\*imgfls\_index)=figure(2\*imgfls\_index);

set(fsf(2\*imgfls\_index), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

for encode\_bit\_index = 1:length(encode\_bits)

levels=2^encode\_bit\_index;

minv=min(audfls\_content{imgfls\_index});

maxv=max(audfls\_content{imgfls\_index});

range=maxv-minv;

step=range/(levels);

lowerpart=minv+step;

upperpart=maxv-step;

lowercode=minv+(step/2);

uppercode=maxv-(step/2);

[index,quants] = quantiz(audfls\_content{imgfls\_index},lowerpart:step:upperpart,lowercode:step:uppercode);

%plot quantized signal-----------------------

subplot((length(encode\_bits)+1)/2,2,encode\_bit\_index);

plot(quants);

title([num2str(encode\_bits(encode\_bit\_index)) ' bit quantized ' audfls\_name{imgfls\_index} ' using quantiz' ]);

xlabel('?'); ylabel(' ¬ø');

axis tight;

%{

if (k==2 && i==4)%k is the audio file, i is the number of bits

soundsc(quants,8000,16)

end

%}

end

subplot((length(encode\_bits)+1)/2,2,length(encode\_bits)+1);

plot(audfls\_content{imgfls\_index});

title([' Original ' audfls\_name{imgfls\_index}]);

xlabel('?'); ylabel('?');

axis tight;

end

A picture containing table

Description automatically generated

A picture containing table

Description automatically generated

Table

Description automatically generated

A picture containing chart

Description automatically generated

Lab 7 – Quantization Error

## Error Analysis Matlab code

clc, clear all, close all;

A = {'musicAudio.au'};

for k=1:length(A)

[y{k}, Fs{k}] = audioread(A{k});

end

for k = 1:length(A)

z = double(y{k});

max\_value = max(y{k});

min\_value = min(y{k});

range = max\_value-min\_value;

shift=z-min\_value;

b = 1:8;% bits to encode new quantizing levels

%Quantize not using quantiz

for i = 1:length(b)

N=2^i;% number of new quantizing levels (2,4,8,16,32...256)

delta = range/(N-1);%step size between two vels

norm=shift/delta;

rounded=round(norm);

unnorm=rounded\*delta;

quantized\_audio=unnorm+min\_value;

%plot quantized signal-----------------------

fsf(10\*k+1)=figure(10\*k+1);

set(fsf(10\*k+1), 'Units', 'Normalized', 'OuterPosition', [0 0 1

1]);%fullscreen

subplot(length(b)/2,2,i);

plot(quantized\_audio);

title([num2str(i) ' bit quantized ' A{k} ' not using quantiz' ]);

xlabel('Sample number'); ylabel('Quantized audio');

axis tight;

%error signal & histogram--------------------

E=quantized\_audio-y{k};

fsf(10\*k+2)=figure(10\*k+2);

set(fsf(10\*k+2), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot(length(b)/2,2,i);

hist(E,20);

title([A{k} ' Error Histogram not using quantiz for b = '

num2str(i)]);

xlabel('Error Magnitude'); ylabel('Sample instances');

%autocorrelation for error signal and plot---------

[r,lags]=xcorr(E,200,'unbiased');

fsf(10\*k+3)=figure(10\*k+3);

set(fsf(10\*k+3), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot(length(b)/2,2,i);

plot(lags,r);

title([A{k} ' Autocorrelation not using quantiz for b = '

num2str(b(i))]);

xlabel('lag'); ylabel('Auto-correlation');

axis tight;

%cross-correlation and plot------------------

[c,lags]=xcorr(E,quantized\_audio,200,'unbiased');

fsf(10\*k+4)=figure(10\*k+4);

set(fsf(10\*k+4), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot(length(b)/2,2,i);

plot(lags,c);

title([A{k} ' Cross-correlation not using quantiz for b = '

num2str(b(i))]);

xlabel('lag'); ylabel('Cross-correlation');

axis tight;

end

%Quantize using quantiz

for i = 1:length(b)

levels=2^i;

minv=min(z);

maxv=max(z);

range=maxv-minv;

delta=range/(levels);

lowerpart=minv+delta;

upperpart=maxv-delta;

lowercode=minv+(delta/2);

uppercode=maxv-(delta/2);

[index,quants] =

quantiz(z,lowerpart:delta:upperpart,lowercode:delta:uppercode);

%plot quantized signal-----------------------

fsf(10\*k+5)=figure(10\*k+5);

set(fsf(10\*k+5), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot(length(b)/2,2,i);

plot(quants);

title([num2str(b(i)) ' bit quantized ' A{k} ' using quantiz' ]);

xlabel('Sample number'); ylabel('Quantized audio');

axis tight;

%error signal & histogram--------------------

E=quants'-y{k};

fsf(10\*k+6)=figure(10\*k+6);

set(fsf(10\*k+6), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot(length(b)/2,2,i);

hist(E,20);

title([A{k} ' Error Histogram using quantiz for b = '

num2str(b(i))]);

xlabel('Error Magnitude'); ylabel('Sample instances');

%autocorrelation for error signal and plot---------

[r,lags]=xcorr(E,200,'unbiased');

fsf(10\*k+7)=figure(10\*k+7);

set(fsf(10\*k+7), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot(length(b)/2,2,i);

plot(lags,r);

title([A{k} ' Autocorrelation using quantiz for b = '

num2str(b(i))]);

xlabel('sample lag'); ylabel('Auto-correlation');

axis tight;

%cross-correlation and plot------------------

[c,lags]=xcorr(E,quants',200,'unbiased');

fsf(10\*k+8)=figure(10\*k+8);

set(fsf(10\*k+8), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

subplot(length(b)/2,2,i);

plot(lags,c);

title([A{k} ' Cross-correlation using quantiz for b = ' num2str(i)]);

xlabel('sample lag'); ylabel('Cross-correlation');

axis tight;

end

end

Power Signal to Noise Ratio (PSNR) Matlab code

clc, clear all, close all;

A = {'goreAudio.au','musicAudio.au','speechAudio.au','boygeorgeAudio.au'};

for k=1:4

[y{k}, Fs{k}] = audioread(A{k});

end

fsf=figure();

set(fsf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

for k = 1:4

max\_value = max(max(y{k}));

min\_value = min(min(y{k}));

range = max\_value-min\_value;

shift=y{k}-min\_value;

b = 1:7;

%Quantize not using quantiz

for i = 1:length(b)

N=2^i;

delta = range/(N-1);

norm=shift/delta;

rounded=round(norm);

unnorm=rounded\*delta;

quantized=unnorm+min\_value;

%error signal---------------------------------

quantized\_audio=quantized;

E=quantized\_audio-y{k};

%signal to noise ratio------------------------

L1=length(quantized\_audio);

L2=length(E);

Py=(1/L1)\*sum(quantized\_audio.^2);

Pe=(1/L2)\*sum(E.^2);

PSNR=Py/Pe;

distortion(i)=1/PSNR;

Fs=8000;

bit\_rate(i)=Fs\*i;

end

%plot distortion curve not using quantiz

subplot(4,2,(2\*k)-1);

plot(bit\_rate,distortion);

title(['Distortion Curve not using quantiz on ',A{k}]);

xlabel('bit rate (bits/s) - Fs(sample/s) \* b(bits/sample)');

ylabel('Signal distortion');

axis tight;

%Quantize using quantiz

for i = 1:length(b)

levels=2^i;

minv=min(y{k});

maxv=max(y{k});

range=maxv-minv;

delta=range/(levels);

lowerpart=minv+delta;

upperpart=maxv-delta;

lowercode=minv+(delta/2);

uppercode=maxv-(delta/2);

[index,quants] = quantiz(y{k},lowerpart:delta:upperpart,lowercode:delta:uppercode);

%error signal---------------------------------

E=quants'-y{k};

%signal to noise ratio------------------------

L1=length(quants');

L2=length(E);

Py=(1/L1)\*sum(quants'.^2);

Pe=(1/L2)\*sum(E.^2);

PSNR=Py/Pe;

distortion(i)=1/PSNR;

Fs=8000;

bit\_rate(i)=Fs\*i;

end

%plot distortion curve using quantiz------------------

subplot(4,2,2\*k);

plot(bit\_rate,distortion);

title(['Distortion Curve using quantiz on ',A{k}]);

xlabel('bit rate (bits/s) - Fs(sample/s) \* b(bits/sample)');

ylabel('Signal distortion');

axis tight;

end

Chart, line chart

Description automatically generated