

Southern University of Science and Technology

Speech Signal Processing

# Lab 10 Report

11510478 郭锦岳

Question 11.28

(a)

Code:

signal = audioread('s5.wav');

signal\_8 = fxquant(signal, 8, 'round', 'sat');

signal\_9 = fxquant(signal, 9, 'round', 'sat');

snr8 = snr(signal\_8, signal);

snr9 = snr(signal\_9, signal);

>>

snr8 = 37.8333

snr9 = 43.4176

snr9 – snr8 = 5.5843

The value roughly matches the expected amount 6 dB, as the bit number is increased by one.

(b)

To compute the values, I write a function called SNRplot, which plots two lines in a single run: the mu-law curve and the corresponding uniform curve.

Code:

function [ signal ] = SNRplot( signal, mu, bit)

%SNRPLOR plot a single line

% signal: input signal array

% mu: factor for mu-law

% bit: quantization bit number

% x-axis: 1/sigma, 13 samples

% y-axis: SNR, 13 samples

factor = 1\*2.^(0:-1:-12)';

signal = factor\*signal';

signal = signal';

SNR\_mu = zeros(1,13);

SNR\_uni = zeros(1,13);

standard = std(signal);

% mulaw and quantization

for i = 1:13

signal\_f = signal(:,i);

signal\_compress = mulaw(signal\_f, mu);

signal\_q = fxquant(signal\_compress, bit, 'round', 'sat');

signal\_expand = mulawinv(signal\_q, mu);

SNR\_mu(i) = snr(signal\_expand, signal\_f);

SNR\_uni(i) = snr(fxquant(signal\_f,bit,'round','sat'),signal\_f);

end

semilogx(1./standard, SNR\_mu);

semilogx(1./standard, SNR\_uni,'--');

grid;

Running:

clear;clc

mu = 500;

signal = audioread('s5.wav');

hold on;

for i = 10:-1:6

SNRplot(signal, mu, i);

end

hold off;

xlabel('1/sigmax');ylabel('SNR in dB');title(sprintf('mu=%d',mu));

legend('10bit','10bit','9bit','9bit','8bit','8bit','7bit','7bit','6bit','6bit');





As mu increases, the curves of mu-law quantizer could maintain as a flat line for a greater range. To reach the effect of 6bit mu-law quantizer, a uniform quantizer must have at least 9bit. The enhancement of mu-law quantizer is significant.

However, we can observe that at the beginning of the curves, SNR is slightly smaller when mu is bigger. I think this is because when variance is small, the quantization error might be amplified as mu gets bigger.

Question 11.30

To implement the ADPCM coders, I write two functions to realized the encoder and decoder.

Code:

function[ adpcm\_y ]= adpcm\_encoder(raw\_y)

IndexTable = [-1, -1, -1, -1, 2, 4, 6, 8, -1, -1, -1, -1, 2, 4, 6, 8];

StepSizeTable = [7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 19, 21, 23, 25, 28, 31, 34, 37, 41, 45, 50, 55, 60, 66, 73, 80, 88, 97, 107, 118, 130, 143, 157, 173, 190, 209, 230, 253, 279, 307, 337, 371, 408, 449, 494, 544, 598, 658, 724, 796, 876, 963, 1060, 1166, 1282, 1411, 1552, 1707, 1878, 2066, 2272, 2499, 2749, 3024, 3327, 3660, 4026, 4428, 4871, 5358, 5894, 6484, 7132, 7845, 8630, 9493, 10442, 11487, 12635, 13899, 15289, 16818, 18500, 20350, 22385, 24623, 27086, 29794, 32767];

prevsample = 0;

previndex = 1;

Ns = length(raw\_y);

n = 1;

raw\_y = 32767 \* raw\_y; % 16-bit operation

while (n <= Ns)

predsample = prevsample;

index = previndex;

step = StepSizeTable(index);

diff = raw\_y(n) - predsample;

if (diff >= 0)

code = 0;

else

code = 8;

diff = -diff;

end

tempstep = step;

if (diff >= tempstep)

code = bitor(code, 4);

diff = diff - tempstep;

end

tempstep = bitshift(tempstep, -1);

if (diff >= tempstep)

code = bitor(code, 2);

diff = diff - tempstep;

end

tempstep = bitshift(tempstep, -1);

if (diff >= tempstep)

code = bitor(code, 1);

end

diffq = bitshift(step, -3);

if (bitand(code, 4))

diffq = diffq + step;

end

if (bitand(code, 2))

diffq = diffq + bitshift(step, -1);

end

if (bitand(code, 1))

diffq = diffq + bitshift(step, -2);

end

if (bitand(code, 8))

predsample = predsample - diffq;

else

predsample = predsample + diffq;

end

if (predsample > 32767)

predsample = 32767;

elseif (predsample < -32768)

predsample = -32768;

end

index = index + IndexTable(code+1);

if (index < 1)

index = 1;

end

if (index > 89)

index = 89;

end

prevsample = predsample;

previndex = index;

adpcm\_y(n) = bitand(code, 15);

n = n + 1;

end

Decoder:

function [ decoded\_y ] = adpcm\_decoder( adpcm\_y )

IndexTable = [-1, -1, -1, -1, 2, 4, 6, 8, -1, -1, -1, -1, 2, 4, 6, 8];

StepSizeTable = [7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 19, 21, 23, 25, 28, 31, 34, 37, 41, 45, 50, 55, 60, 66, 73, 80, 88, 97, 107, 118, 130, 143, 157, 173, 190, 209, 230, 253, 279, 307, 337, 371, 408, 449, 494, 544, 598, 658, 724, 796, 876, 963, 1060, 1166, 1282, 1411, 1552, 1707, 1878, 2066, 2272, 2499, 2749, 3024, 3327, 3660, 4026, 4428, 4871, 5358, 5894, 6484, 7132, 7845, 8630, 9493, 10442, 11487, 12635, 13899, 15289, 16818, 18500, 20350, 22385, 24623, 27086, 29794, 32767];

prevsample = 0;

previndex = 1;

Ns = length(adpcm\_y);

n = 1;

while (n <= Ns)

predsample = prevsample;

index = previndex;

step = StepSizeTable(index);

code = adpcm\_y(n);

diffq = bitshift(step, -3);

if (bitand(code, 4))

diffq = diffq + step;

end

if (bitand(code, 2))

diffq = diffq + bitshift(step, -1);

end

if (bitand(code, 1))

diffq = diffq + bitshift(step, -2);

end

if (bitand(code, 8))

predsample = predsample - diffq;

else

predsample = predsample + diffq;

end

if (predsample > 32767)

predsample = 32767;

elseif (predsample < -32768)

predsample = -32768;

end

index = index + IndexTable(code+1);

if (index < 1)

index = 1;

end

if (index > 89)

index = 89;

end

prevsample = predsample;

previndex = index;

decoded\_y(n) = predsample / 32767;

n = n + 1;

end

end

Running:

signal1 = audioread('s1.wav');

signal1\_adpcm = adpcm\_decoder(adpcm\_encoder(signal1));

snr1 = snr(signal1\_adpcm, signal1');

repeat for s1 to s6.



As for the result, the SNR varies for different input signals; Averagely the value is around 20dB. Even though the method I use is 16bit quantizing, the result does not seem to be satisfying. Maybe there is some problem of my code, and I need to work for further investigation.