

EE3414 Multimedia Communication Systems I

Experiment One

Sampling and Quantization

Yao Wang and Xiaofeng Xu

1. Introduction

The purpose of this lab is for you to understand the principles of sampling a continuous time signal, increasing or decreasing the sampling rate of a discrete time signal, and for quantization. In this experiment you will record your own speech/music with different sampling rates and bits per sample, apply sampling rate conversion and quantization on digital signals. By comparing the sound quality obtained with different filters and quantizers at the same sampling frequency and quantization level, you should have a better appreciation of the differences in performances by using different prefilters, interpolation filters, and quantizers.

If you have, please bring a headphone, so you can listen to the sound with better quality, also without interfering others.

Part 1 of the lab can be done with your own computer. In order to save the time spent in the lab, you should try to do it before coming to the lab. (see detailed instruction below). If you have to do this in the lab, bring one of your favorite music CD, or your own portable CD player or MP3 player.

You should read this document and try to understand all the MATLAB programs before coming to the lab. The APPENDIX provides all the MATLAB functions with detailed documentations. To help you understand the program, flow-graphs of the programs are also given. If you don't understand a particular function call and the meanings of its input and output variables, type "help function_name" on the command window of MATLAB.

2. Experiment

- 1) Audio recording at different sampling frequencies and bit rate.
 - a. Start the CD player on the computer, or your own music player (CD Player and MP3 player).
 - b. If you are playing the music from an external player, then connect the sound output from the player to the sound input jack (for connecting microphone) on the computer. If you don't have a cable enabling the aforementioned connection, you can connect the sound output jack of the external player with the speaker and the sound input jack on the computer with the microphone, and move the microphone near the speaker.

- c. If you are playing a MP3 file on the same computer, you need to configure the “recording control” to change the source of sound input. Open the volume control panel (double-click the “speaker” icon in system tray), select “options->properties->recording”, then select “CD player, Microphone, Stereo Mix” to show all these three items in recording control panel, click “OK”. In recording control panel, select “Stereo Mix”. So you can record the internal sound of the computer.
- d. It is similar to record music from CD on the same computer. The difference is you need to select “CD player” instead of “Stereo Mix” in “Recording Control Panel”.
- e. Open the sound recorder (accessory -> entertainment -> sound recorder). Adjust the recorder properties to set the sampling rate be 11.025 KHz, 8 bits/sample. This can be done by selecting “file->properties->convert now”. Choose “PCM” for “format”, choose “11.025 KHz, 8 bits, mono” for “attributes”, type “.wav” for “save as”.
- f. Push the “record” button. Record 5 seconds of the played audio. Save it into a file using the “.wav” format.
- g. Open another sound recorder. Change the recorder properties to set the sampling rate be 44.100 KHz, 16 bits/sample. Record 5 seconds of roughly the same audio segment. Save it into a different file.
- h. Play back the recorded two files using the “play” button on the respective sound recorders. *Compare the sound quality. In your report, you should include a brief comment on the sound quality.*

Note: you should try to do this part on your own computer before coming to the lab. You are welcome to experiment with other combinations of sampling frequencies and bit resolutions. You can also try to record your own speech. But please make sure that you bring one file that is sampled at 44 KHz, quantized at 16 bits/sample, and that the file is fairly short (5 seconds).

2) Sampling rate conversion using different prefilters and interpolation filters.

You are given two MATLAB programs, `down4up4_nofilt()`, `down4up4_filt()`. These two programs are included in the Appendix of this document and briefly explained. The “`down4up4_nofilt`” program down sample a given sound file by a factor of 4, without prefiltering, and then interpolate the resulting sequence by a factor of 4 by repeating each sample 4 times. It also computes and displays the spectra of the original, down-sampled, and interpolated sequences. The “`down4up4_filt`” program uses a good low-pass filter for prefiltering before down-sampling, and uses a good interpolation filter for up-sampling.

You can apply the two programs on a sound file you recorded using a sampling frequency of 44KHz sampling frequency, 16 bits/second. Or you can use one of the sound files provided for this lab.

- a. Start the MATLAB program. Change directory to your own working directory. Copy all the program files and sample audio files to your directory.

- b. Use the “down4up4_nofilt” program on a sound file recorded using a sampling frequency of 44KHz sampling frequency. For example, if you want to use an input file with name “myinput.wav”, and save the interpolated sequence in an output file with name “myoutput.wav”, you should type

```
down4up4_nofilt('myinput.wav', 'myoutput.wav')
```

on the command window of MATLAB.

Compare the original sound and the one after down-sampling and upsampling, in terms of perceptual sound quality, waveform, and spectrum. The program also computes the mean square error between the original and the interpolated signal. Record this error in your report.

- c. Use the “down4up4_filt” program on the same input sound file you used for part a). *Compare the original sound and the one after down-sampling and upsampling, in terms of perceptual sound quality, waveform, and spectrum. Also compare the resulting sequence from part a) with the resulting sequence from part b), in terms of perceptual sound quality, waveform, spectrum, and reconstruction error. Which program gives better results?* This program also plots the impulse response and frequency response of the pre-filter and interpolation filter used. *Ideally, what should be the cut-off frequencies of these filters? Do the filters used have sufficient attenuation at the desired frequencies?*

In your report, you should include the relevant waveform and spectral plots, also include comments on the differences between original and processed signals (in terms of perceptual sound quality, waveform, and spectrum). You can copy a MATLAB figure by choosing “Copy figure” in the “edit” sub-menu in the figure window, and then paste into a desired document. You should open a temporary document to save all the figures, so that you can use these figures when writing your report. Make sure you write notes on which figure is for what while doing so.

3) Quantization

You are given two MATLAB programs, `quant_uniform()` and `quant_mulaw()`. These two programs are included in the Appendix of this document and briefly explained. The “quant_uniform” program quantizes an input sequence using a uniform quantizer with a user-specified number of levels. The “quant_mulaw” program uses the mu-law quantizer.

- a) Apply “quant_uniform” function to a sound file recorded with 16 bits/sample, with quantization level set to 8 bits ($N=256$) and 4 bits ($N=16$), respectively. For example, if you want to use an input file with name “myinput.wav”, and save the quantized sequence in an output file with name “myoutput.wav”, and set $N=256$, you should type

```
quant_uniform('myinput.wav', 'myoutput.wav', 256)
```

on the command window of MATLAB.

Compare the original sequence and quantized sequences with different levels, in terms of perceptual sound quality and the waveform. Also record and compare the quantization error (mean square error between the original and quantized samples).

b) Apply “quant_mulaw” function to a sound file recorded with 16 bits/sample, with quantization level set to 8 bits (N=256) and 4 bits (N=16), respectively, with $\mu=16$. *Compare the original sequence and quantized sequences with different levels. Also compare the quantized sequence obtained with the uniform quantizer and that with the mu-law quantizer with the same number of quantization levels. Which quantizer yields smaller error for signal values that are small, and larger error for signal values that are large? (you should observe this from the waveform plot).*

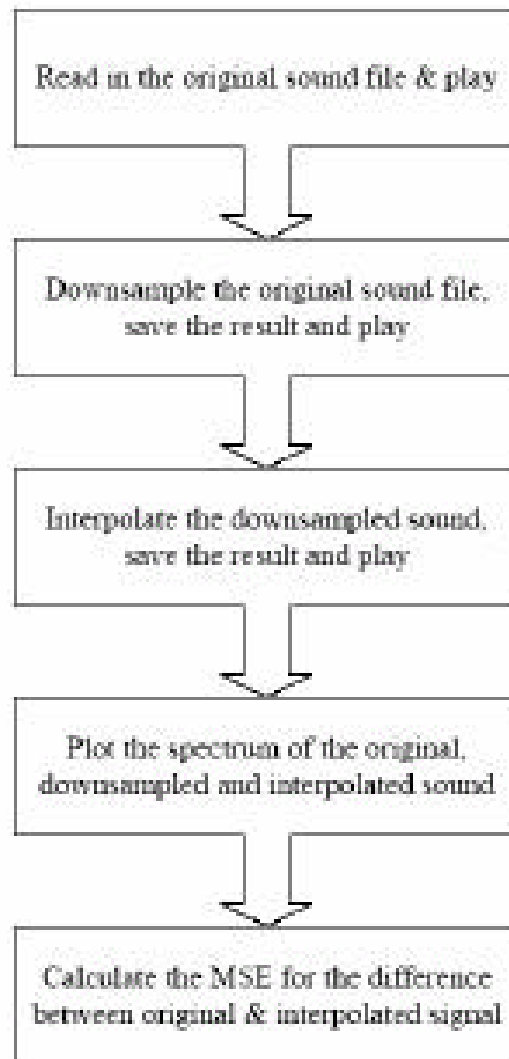
In your report, you should include the waveform plots, also include a comment on the difference between different quantized sequences (in terms of perceptual sound quality, waveform and mean square quantization error). Which method gives better sound quality for the same number of quantization levels?

3. Report

Your report should start with a general description of what you did in this experiment, followed by the results and observations you obtained. You should answer all the questions (written in italics in the preceding instructions), with the corresponding figures (waveforms or spectrums) next to your answers.

APPENDIX MATLAB PROGRAMS AND THEIR FLOW-GRAPHS

A. Flow-graph of the MATLAB function “down4up4_nofilt” and “down4up4_filt”



B. MATLAB FUNCTION “down4up4_nofilt”

```
% This function downsample a sound file by a factor of 4 and then  
% interpolate it back to the original sampling rate.  
% No prefilter is applied for down-sampling  
% Interpolation is done by repeating every sample 4 times  
% The program plays the sound file to allow perceptual evaluation of  
% sound quality.  
% It also displays the waveforms and spectrums of original, down-  
% sampled, and upsampled signal  
% Yao Wang, Polytechnic University, 1/11/2004
```

```
function []=down4up4_nofilt(InFilename,OutFilename);
```

```
% Usage : down4up4_nofilt('Infilename','Outfilename')
```

```

fprintf('\n Read in the original sound \n')
[y,fs]=wavread(InFilename);

%first shorten the sequence so that the length is multiple of 4
orig_length=length(y);
N=floor(orig_length/4)*4;
y = y(1:N);
disp([fs,N]);

% Play it
sound(y,fs);
figure(1);
subplot(1,2,1),bar(y(2000:2060),0.02);axis tight;title('Waveform of
Original');
subplot(1,2,2),psd(y,256,fs);axis([0,fs/2,-
60,0]);ylabel('');xlabel('');title('Spectrum of Original');

fprintf('\n Hit a key to continue\n');
pause

% Downsample without prefiltering: take every 4th sample
fprintf('\n The downsampled sound \n')
x=y(1:4:N);
sound(x,fs/4);
figure(2);
subplot(1,2,1),bar(x(2000/4:2060/4),0.02);axis tight;title('Waveform of
Down4');
subplot(1,2,2),psd(x,256,fs/4);axis([0,fs/2,-
60,0]);ylabel('');xlabel('');title('Spectrum of Down4');

fprintf('\n Hit a key to continue\n');
pause;

%Interpolation by repeating each sample 4 times
fprintf('\n The interpolated sound \n')
z=zeros(N,1);
%interpolate by first copy the samples in x to every 4th sample to z
starting from 1st sample
% then starting at 2nd sample, and so on
z(1:4:N)=x;
z(2:4:N)=x;
z(3:4:N)=x;
z(4:4:N)=x;
%interp(x,4);

sound(z,fs);
figure(3);
subplot(1,2,1),bar(z(2000:2060),0.02);axis tight;title('Waveform of
Up4');
subplot(1,2,2),psd(z,256,fs);axis([0,fs/2,-
60,0]);ylabel('');xlabel('');title('Spectrum of Up4');

fprintf('\n Hit a key to continue\n');
pause;
% Save the interpolated sequence

```

```
wavwrite(z,fs,OutFilename);

% Calculate the MSE
D=y-z;
MSE=mean(D.^2);
fprintf('\n Error between original and interpolated = %g\n\n',MSE )
```

C. MATLAB FUNCTION “down4up4_filt”

```
% This function downsample a sound file by a factor of 4 and then
% interpolate it back to the original sampling rate.
% A length 31 FIR lowpass filter is applied for down-sampling
% Interpolation is done by using a length 33 FIR interpolation filter .
% The program plays the sound file to allow perceptual evaluatio of
% sound quality.
% It also displays the waveforms and spectrums of original, down-
% sampled, and upsampled signal
% The filter response is also displayed
% Yao Wang, Polytechnic University, 1/11/2004

function[]=down4up4_filt(InFilename,OutFilename);
% Usage : down4up4_nofilt('Infilename','Outfilename')

fprintf('\n Read in the original sound \n')
[y,fs]=wavread(InFilename);

%first shorten the sequence so that the length is multiple of 4
orig_length=length(y);
N=floor(orig_length/4)*4;
y = y(1:N);
disp([fs,N]);

% Play it
sound(y,fs);
figure(1);
subplot(1,2,1),bar(y(2000:2060),0.02);axis tight;title('Waveform of
Original');
subplot(1,2,2),psd(y,256,fs);axis([0,fs/2,-
60,0]);ylabel('');xlabel('');title('Spectrum of Original');

fprintf('\n Hit a key to continue\n');
pause

% Downsample with a default length 31 FIR prefilter:
fprintf('\n The downsampled sound \n')
x=decimate(y,4,30,'FIR');
sound(x,fs/4);
figure(2);
subplot(1,2,1),bar(x(2000/4:2060/4),0.02);axis tight;title('Waveform of
Down4');
subplot(1,2,2),psd(x,256,fs/4);axis([0,fs/2,-
60,0]);ylabel('');xlabel('');title('Spectrum of Down4');

fprintf('\n Hit a key to continue\n');
pause;
```

```

%display the prefilter used
figure(3);
hdec=fir1(30,1/4);[hdecspec,f]=freqz(hdec,1);

subplot(1,2,1),bar(-15:1:15,hdec,0.02);axis tight;title('Prefilter
Impulse Response ');
subplot(1,2,2),
plot(f/pi,20*log10(abs(hdecspec)));axis tight;title('Prefilter
Frequency Response');ylabel('');xlabel('');

fprintf('\n Hit a key to continue\n');
pause;

%Interpolation using a default FIR interpolation filter
fprintf('\n The interpolated sound \n')
[z,hintp]=interp(x,4);
%interp uses a default filter, which is returned in hintp

sound(z,fs);
figure(4);
subplot(1,2,1),bar(z(2000:2060),0.02);axis tight;title('Waveform of
Up4');
subplot(1,2,2),psd(z,256,fs);axis([0,fs/2,-
60,0]);ylabel('');xlabel('');title('Spectrum of Up4');

fprintf('\n Hit a key to continue\n');
pause;

%display the prefilter used
figure(5);
[hintpspec,f]=freqz(hintp,1);

subplot(1,2,1),bar(-16:1:16,hintp,0.02);axis tight;title('Intp Filter
Impulse Response ');
subplot(1,2,2),
plot(f/pi,20*log10(abs(hintpspec)));axis tight;title('Intp Filter
Frequency Response');ylabel('');xlabel('');

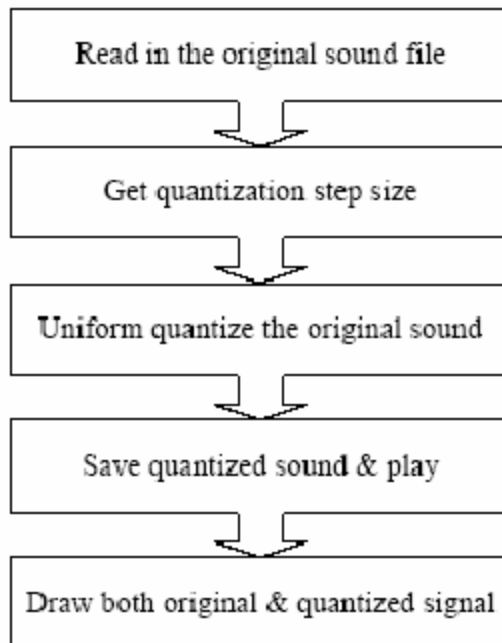
fprintf('\n Hit a key to continue\n');
pause;
% Save the interpolated sequence
wavwrite(z,fs,OutFilename);

% Calculate the MSE
D=y-z;
MSE=mean(D.^2);
fprintf('\n Error between original and interpolated = %g\n\n',MSE )

```

D. Flow-graph for MATLAB function “quant_uniform”

(note, the flow-graph for quant_mulaw is very similar)



E. MATLAB function “quant_uniform”

```
% This program quantizes a sound signal using uniform quantization  
% Yao Wang, Polytechnic University, 2/11/2004
```

```
function []=quant_uniform(inname,outname, N);  
if nargin < 3  
    disp('Usage: quant_uniform(inname,outname, N)');  
    disp('inname: input .wav file name');  
    disp('outname: output .wav file name');  
    disp('N: quantization level, N should be a positive integer');  
    return;  
end;
```

```
%read in input signal  
[x,fs,R]=wavread(inname);
```

```
magmax=max(abs(x));  
xmin=-magmax, xmax=magmax;  
Q=(xmax-xmin)/N;  
disp('R,xmin,xmax,N,Q');  
disp([R,xmin,xmax,N,Q]);
```

```
%apply uniform quantization on each sample
```

```
xq=floor((x-xmin)/Q)*Q+Q/2+xmin;
```

```
%compare sound quality
```

```

wavwrite(xq,fs,R,outname);
sound(x,fs);
fprintf('\n Hit a key to continue');
pause;
sound(xq,fs);

%plot waveforms over the entire period
t=1:length(x);
figure; plot(t,x,'r:');
hold on; plot(t,xq,'b-');
axis tight; grid on;
legend('original','quantized')
%plot waveform over a selected period
t=2000:2200;
figure; plot(t,x(2000:2200),'r:');
hold on; plot(t,xq(2000:2200),'b-');
axis tight; grid on;

% Calculate the MSE
D=x-xq;
MSE=mean(D.^2);
fprintf('\n Error between original and quantized = %g\n\n',MSE )

```

F. MATLAB FUNCTION “quant_mulaw”

```

% This program quantizes a sound signal using mu-law quantization
% Yao Wang, Polytechnic University, 2/11/2004

```

```

function []=quant_mulaw(inname,outname, N,mu);
if nargin < 3
    disp('Usage: quant_mulaw(inname,outname, N, mu)');
    disp('inname: input .wav file name');
    disp('outname: output .wav file name');
    disp('N: quantization level, N should be a positive integer');
    disp('mu: 1<=mu <=255');
    return;
end;

%read in input signal
[x,fs,R]=wavread(inname);
xmin=min(x); xmax=max(x);

magmax=max(abs(x));
xmin=-magmax, xmax=magmax;
Q=(xmax-xmin)/N;
disp('R,xmin,xmax,N,Q,mu');
disp([R,xmin,xmax,N,Q,mu]);

%apply mu-law transform to original sample
y=xmax*log10(1+abs(x)*(mu/xmax))/log10(1+mu);

%apply uniform quantization on the absolute value each sample
yq=floor((y-xmin)/Q)*Q+Q/2+xmin;

%apply inverse mu-law transform to the quantized sequence

```

```

%also use the original sign
xq=(xmax/mu)*(10.^((log10(1+mu)/xmax)*yq)-1).*sign(x);

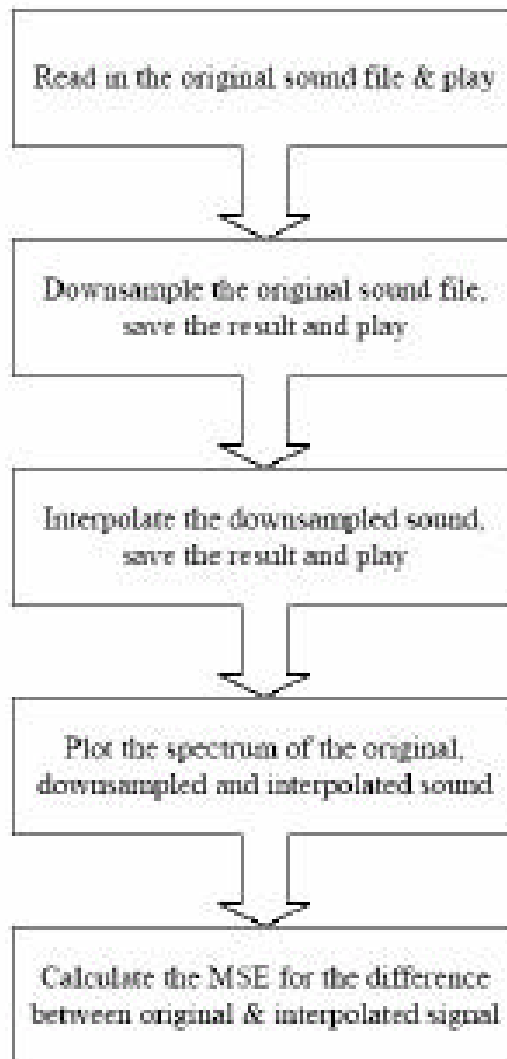
%compare sound quality
wavwrite(xq,fs,R,outname);
sound(x,fs);
fprintf('\n Hit a key to continue');
pause;
sound(xq,fs);

%plot waveforms over the entire period
t=1:length(x);
figure; plot(t,x,'r:');
hold on; plot(t,xq,'b-');
axis tight; grid on;
legend('original','quantized')
%plot waveform over a selected period
t=2000:2200;
figure; plot(t,x(2000:2200),'r:');
hold on; plot(t,xq(2000:2200),'b-');
axis tight; grid on;

% Calculate the MSE
D=x-xq;
MSE=mean(D.^2);
fprintf('\n Error between original and quantized = %g\n\n',MSE )

```

APPENDIX B FLOW GRAPHS OF MATLAB PROGRAMS



Flow-graph of the MATLAB function “down4up4_nofilt” and “down4up4_filt”

Flow-graph of the MATLAB function “quant_uniform”

Left is for