Lab 0

Igor Fedorov
Yihan Hu

January 4, 2021

1 Grading

Each lab will consist of a series of tasks, which you must complete in order to receive full credit. Your work must be summarized in a written report, which will be graded by your TA. You may use your favorite word processor. We will give 5% extra credit per lab for reports completed in LaTex. You may use NeurIPS templates in LATEX and Word, available from the 2015 NeurIPS format site. The page limits mentioned there don't apply.

For **each** task, generate the required figures/plots/etc. and provide a brief explanation or summary of your findings if asked. You may also be asked to attach Matlab/C code to the end of your report.

2 Objectives

In this lab, you will learn how to

1. Generate a discrete-time domain signal in MATLAB
2. Write the result to file
3. Verify the results of your code in MATLAB.

3 Instructions

Let us begin by sampling a sinusoid with frequency (f) at a given sampling rate.

3.1 Task 1

Write a MATLAB function `generate_sigs()` which outputs L samples from a sinusoid of frequency f sampled at F_s. The function should have three arguments. Generate 3 plots showing the generated samples for $F_s = 48\ kHz$, $L = 100$, and $f = 2.3\ kHz$, $f = 23\ kHz$, and $f = 36\ kHz$. Use `subplot()`

command to place all 3 plots in the same figure window. Clearly label the horizontal and vertical axes of all plots and give titles for each plot with the value of f. Include your Matlab function code in the report.

3.2 Task 2

Use `fft()` function to find the length 64 Fast Fourier Transform of the signals generated in Task 1. Generate a figure with 3 subplots corresponding to the magnitude of the FFT of each signal from Task 1. Plot the first half of the FFT, with the horizontal axis scaled to be in units of radians. Label the axes for each subplot and give titles for each figure to denote f.

3.3 Task 3

Explain why the magnitude plots are not perfect delta functions.

3.4 Task 4

Describe how the plots in Task 2 relate to the famous Nyquist-Shannon sampling theorem, especially the phenomenon happening at $f = 36\ kHz$.

3.5 Task 5

Use your previous code to generate sinusoids of length 100 with frequency $f = 2.3, 23, 36\ kHz$ sampled at $F_s = 48\ kHz$. Write the generated signal to WAV files using `audiowrite()` function. Then, use the provided C skeleton code to decrease the amplitude of all 3 signals by a factor of 0.5. Confirm your C implementation by reading the C-generated WAV files back into MATLAB and plotting the processed signals on top of the original ones. Label all axes, generate titles for each plot, and add a legend specifying which signal is the original and which is the modified. Include the part of your C code that accomplishes the modification in the report.

4 Report Outline with Rubric (16 pts)

1. Title (1pt): The title has to be informative and not generic like 'ECE161B Lab0 Report'. Also you have to include author's list.
2. Introduction (1pt): A short description of what you did. Please mention any key findings and interesting insights.
3. Task 1 (1+3 pts): A figure that has three subplots of signals. You may have points deducted if you fail to include clear labels and axes. Include function code snippet in the appendix of the report.
4. Task 2 (3 pts): A figure that has three subplots of FFT with clear labels and titles. Axes need to be scaled to radians.

5. Task 3 (2 pts)
6. Task 4 (2 pts)
7. Task 5 (2+1 pts): Include the modified C code in the appendix of the report. Show a figure that has the original and the modified signal overlaid.
8. For the source code-the most points will be given for well-documented and clean code.

5 Additional Information

For the C programming task, we will use libsndfile library for reading and writing files containing sampled sound. You can read more about the library and its usage here: libsndfile.

Unfortunately, most systems do not have the library pre-installed, which would require us to install it manually and link the compiler to the library during compilation.

5.1 Installation

First, you need to install libsndfile. For macOS and linux system, you can simply do so in your terminal:

```
macOS: brew install libsndfile
Linux: sudo apt-get install libsndfile1-dev
```

For Windows, we highly recommend you do the assignment in Ubuntu, an OS based on the Debian Linux distribution, which can be found in the Windows store. Follow the steps below:

Start - Windows Features - Scroll down to turn on Windows Subsystem for Linux - Restart the system - Windows store - Install Ubuntu App - Start - Ubuntu - Create account - Command prompt - WSL

Then you can install the library and compile your code in the prompted terminal:

```
sudo apt-get update
sudo apt install gcc
sudo apt install pkg-config
sudo apt-get install libsndfile1-dev
```

5.2 Compilation

Once you have the library ready, you can run the following to compile your C code and run the executable (assuming you have a compiler installed, in this example, gcc is used):

```
# -lm flag links the compiler to the library
gcc -lm -lsndfile -o skeleton skeleton.c

# provide wav files as arguments to the executable
./skeleton input.wav output.wav
```

Some systems might still experience header linking problem, in that case, you can run the following to compile:

```
gcc -lm -o skeleton skeleton.c $(pkg-config sndfile --cflags --libs)
```