Lab 1

Igor Fedorov
Yihan Hu

February 15, 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. Compose a song in Matlab
2. Perform filtering operations on time domain signals

3 Instructions

Build upon the code from Lab 0 to complete the following tasks

3.1 Task 1

Write a MATLAB function `generate_melody()` which generates a time domain signal given the sampling rate F_s, a list of notes $\{z_i\}_{i=1}^{N}$ (in Hz), and the duration of each note (in seconds) t. You can reuse `generate_sigs()` from Lab 0 to generate each individual note. Use your code to generate a time domain signal consisting of the notes in Table 1 below, in that order, with $F_s = 8e3$ and $t = 0.5$. Write the resulting signal to a WAV file and describe what you hear. Include the function code in your report.

3.2 Task 2

Use MATLAB's `spectrogram()` function to generate a *time-frequency* representation of the signal generated in Task 1. Scale your axes so that x-axis represents time and y-axis represents frequency. Include the spectrogram in your report and describe how the spectrogram validates your implementation.

do	*re*	*mi*	*fa*	*so*	*la*	*ti*	*nu*
440	495	550	587	660	733	825	0

Table 1: Frequencies of notes in melody (Hz)

do	nu	do	nu	so	nu	so	nu	la	nu	la	nu	so	nu	nu	nu
fa	nu	fa	nu	mi	nu	mi	nu	re	nu	re	nu	do	nu	nu	nu

Table 2: Melody of Song

do_b	so_b	mi_b	so_b	do_b	so_b	mi_b	so_b	do_b	la_b	fa_b	la_b	do_b	so_b	mi_b	so_b
ti_b	so_b	fa_b	so_b	do_b	so_b	mi_b	so_b	ti_b	so_b	fa_b	so_b	do_b	so_b	mi_b	so_b

Table 3: Chorus of Song

do_b	re_b	mi_b	fa_b	so_b	la_b	ti_b
220	248	275	293	330	367	206

Table 4: Frequencies of notes in chorus (Hz)

3.3 Task 3

Use your code from Task 1 to generate the melody (x_{melody}) of Twinkle Twinkle Little Start using Table 2 and chorus (x_{chorus}) using Table 3. Then, form the complete song by combining the melody and chorus using

$$x = 0.6x_{melody} + 0.4x_{chorus}. \tag{1}$$

Plot the spectrogram of x with x-axis representing time and y-axis representing frequency. Describe how the spectrogram confirms your implementation. And lastly, write the result to a WAV file and describe what you hear.

3.4 Task 4

In MATLAB, write a function `iir_filter(x,a,b)` which filters a given signal x with a IIR filter h. h is defined by its numerator coefficients a and denominator coefficients b. Then, pass the song you created in Task 3 through the filter, defined by

$$\begin{aligned} &b_0y[n] + b_1y[n-1] + b_2y[n-2] + b_3y[n-3] + b_4y[n-4] = \\ &a_0x[n] + a_1x[n-1] + a_2x[n-2] + a_3x[n-3] + a_4x[n-4] \end{aligned} \tag{2}$$

where $b_0 = 1,\ b_1 = -3.1820023,\ b_2 = 3.9741082,\ b_3 = -2.293354,\ b_4 = 0.52460587,\ a_0 = 0.62477732,\ a_1 = -2.444978,\ a_2 = 3.64114,\ a_3 = -2.444978,\ a_4 = 0.62477732$. You are not allowed to use any built-in filter functions. Write the result to a WAV file and describe what you hear. Attach your function code in the report.

3.5 Task 5

Use MATLAB's `freqz()` function to generate a magnitude response plot of the IIR filter defined by (2). Explain how the plot relates to what the filtered song sounds like.

3.6 Task 6

Write a C program to generate the song Twinkle Twinkle Little Star, just like what you have done earlier in Task 3. A skeleton code has been provided to you and you are free to edit the script. You are allowed to make your own variables and functions and it is not required to follow the provided logic.

After your C implementation is complete, Write the result to a WAV file. Read the WAV file into MATLAB and compare with your MATLAB result from Task 3 using normalized ℓ_2 error as the distance measure. Include your error in the report.

Note that because of the difference in how MATLAB and C array indexing works, your signals may be misaligned by 1 sample. Make sure to correct for this in your error calculation.

3.7 Task 7

Implement IIR filtering operation in C using the same coefficients from Task 4. Writing the result to a WAV file. Read the resulting WAV file into MATLAB and compare with your MATLAB result from Task 4 using normalized ℓ_2 error as the distance measure. Include your error in the report.

4 Report Outline with Rubric (21 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+2 pts): A short description of what your hear. Include function code in the appendix of the report.
4. Task 2 (2 pts): A spectrogram figure with correct axes and labels. A short description of the spectrogram.
5. Task 3 (1+1+1 pts): A spectrogram figure of Matlab generated song with correct axes and labels. A short description of the spectrogram and what you hear.
6. Task 4 (1+2 pts): Describe the content of the WAV file and include the function code in the appendix of the report.
7. Task 5 (2 pts): Include the magnitude plot of the filter and your explanation.
8. Task 6 (3 pts): Full completion of programming and include the error.
9. Task 7 (3 pts): Full completion of programming and include the error.
10. For the source code-the most points will be given for well-documented and clean code.
11. Please submit your C source code as well as tex file if you'd like the extra credit.