EITF75: Lab 1A & 1B instructions

Ashkan Sheikhi and Ove Edfors

September 12, 2023

Lab 1A

Introduction

In this laboratory exercise, you are going to learn about some basic applications of digital signal processing in audio and speech processing. We will start by introducing tools to import an audio file and extract its information in time and frequency. Then, we will study some basic filters to apply on the audio signals. There are some audio file examples provided for you to use during the lab. These audio files are some cropped music and speech examples. At the end of the lab, feel free to use your own audio files and music if you are interested.

Necessary functions and modules to perform each task will be mentioned in the task descriptions. However, you should be able to figure out the syntax to use them by looking at MATLAB help, or simply type help in command window. For example, if you are going to use the *poly* command, try `help poly` in the command window and read the short description.

It is highly recommended to use your headphones when listening to the audio signals during the lab. This will help you hear the effects more clearly.

Lab report

Each of the tasks in this lab includes some specific questions to be answered. After each problem or lab task, there is a red box containing instructions. Make sure you have answered those questions and included requested plots for each part.

Preparations

Before you start doing the lab tasks, you should make sure you have completed the following preparations.

Software and files

In all digital signal processing labs, you should have MATLAB software with the Signal Processing Toolbox installed on your computer. In order to make sure you can import and exploit the needed files to perform the tasks, make sure you have downloaded the folder *LabFiles_1* from canvas, and you have selected it as your current folder in MATLAB. You can do that by clicking on the *Browse for folder* symbol as shown in the figure below:

Preparation exercises

Before we start working with the lab tasks, there are a few preparation exercises to go through. Answering these problems are very important to get a better understanding of the upcoming lab tasks.

Let's assume we have an audio signal that has been sampled with sampling frequency of F_s Hz. After analyzing the signal, we realize that a one-tone sinusoid at $F = Fs/4$ Hz, is distorting the signal and we need a filter to eliminate the disturbance at this frequency. In the next two exercises, you will design two such filters, with different levels of 'sophistication'.

Problem 1. Design a second order FIR filter to remove the disturbance at $F = Fs/4$, by placing two zeros appropriately on the unit circle.

a) Determine the system function $H_1(f)$ of this filter, if we require that high-frequency components are unaltered, i.e. $H_1(f)|_{f=1/2} = 1$.

b) Plot the pole-zero diagram of the filter.

c) Plot the magnitude and phase responses of the filter, for $0 \leq f \leq 1/2$.

Problem 2. The second-order FIR filter designed in Problem 1 is deemed too unsophisticated for the job, since it will have too much of an influence on frequency components near $F_s/4$. To 'localize' the effect of the zeros in Problem 1, we also place two poles close to the zeros, inside the unit circle, at radius $r = 0.95$ from the origin. This will result in an IIR filter.

a) Determine the system function $H_2(f)$ of this filter, if we again require that high-frequency components are unaltered, i.e. $H_2(f)|_{f=1/2} = 1$.

b) Plot the pole-zero diagram of the filter.

c) Plot the magnitude and phase responses of the filter, for $0 \leq f \leq 1/2$, and comment on the differences you observe, compared to the filter in Problem 1.

Include in lab report:

- Solutions to each preparatory problem, with problem number included.
- The MATLAB code you have used when performing any computations or plotting.

Lab tasks

Lab task 1. In this task you will experiment with basic functionalities in MATLAB to process audio signals. You will use the file *HQmusic.wav* for this purpose. Use the `audioread` command to import the *HQmusic.wav* file. Make sure you have saved both the signal and its sampling frequency in two variables. Listen to the music file in MATLAB using the `soundsc` command. Note that you can set the reconstruction frequency as an argument to the `soundsc` command. Listen to the signal with reconstruction frequencies F_s, $F_s/2$, and $2F_s$.

Include in lab report:

- What is your observation? How does the reconstruction frequency affect the music you hear?

Lab task 2. Now, you will plot the music signal in the time and frequency domains. To plot the signal in time, we need to specify the sampling time. To plot the signal in the time domain, you need to create a ⟨*time vector*⟩ with the same length as the signal. Make sure the distance between the samples in time is equal to the sampling period.
In order to see the signal in the frequency domain, we use `periodogram` command in MATLAB. Since you will learn about discrete Fourier transforms/fast Fourier transforms (DFTs/FFTs) only later in the course, we have provided a function, `Spectrum_PLOT`, to plot the signal power spectral density:

```
Spectrum_PLOT(⟨signal⟩,⟨sampling frequency⟩)
```

The function arguments are the ⟨*signal*⟩ vector and its corresponding ⟨*sampling frequency*⟩. This function plots the spectrum magnitude both in linear and log-scale.

Include in lab report:

- Your MATLAB code and the plot of the time domain signal.
- Your MATLAB code and the plot of the signal power spectral density.
- What is the largest frequency component of this music signal?

Lab task 3. We will now apply some simple filters to our audio signal. Let's start with a moving average of length 5. As you have seen some examples in the course lectures, the output of this filter at time n is a weighted sum of the last 5 samples. Use the `conv` command to apply this filter to your original ⟨*signal*⟩. Note that the length of the convolution is the input ⟨*signal*⟩ length plus the filter length, minus one. To avoid the transient at the beginning, we discard the first 4 samples of the ⟨*output signal*⟩. You can use the following command to discard those samples:

```
⟨output signal⟩=⟨output signal⟩(5:end);
```

Include in lab report:

- A plot the power spectral density of the filtered signal and comment on how it compares to the original signal spectrum, from Lab task 2. At what frequencies is the signal power spectral density affected? Did you expect this?
- Comment on the difference between the original and filtered signals, after listening to both. Did the filter remove BASS or TREBLE components?
- Compare the linear and dB-scale plots of the filtered signal. Do you see any specific structure in the dB-scale plot, that you can't see in the linear scale? How do you explain it?
- Increase the length of the moving average filter to 15, do the same experiment again, and explain your findings.

Lab task 4. We will now illustrate the application of the moving average filter in removing noise from a signal. To generate a noisy signal, add a white Gaussian noise, with variance of $\sigma^2 = 0.1^2 = 0.01$, i.e., add `0.1*randn(length(x),1)` to the original signal and save it in a new variable.

Include in lab report:

- A plot the power spectral density of the noisy signal and describe your findings. What has happened to the signal after adding the noise? Also, listen to the noisy signal and verify your answer. Compare the linear-scale and dB-scale power spectral density plots. What are the advantages of having a dB-scale plot?
- Define a moving average filter of length 5 again. Filter the noisy signal and listen to the filtered signal. Plot the spectrum of the filtered signal. Did you expect this? Describe your findings and include your plots. Did the filter completely remove the noise? Did it affect the music quality?

Lab task 5. As you have seen in the course, you can design a filter by placing poles and zeros on the z-plane. In this lab task, a MATLAB program called `mkiir` is provided for you to design those filters and see the filter frequency response at the same time. The `mkiir` graphic interface allows you to put as many poles and zeros as you want on the z-plane and see the effect. You can also move the zeros and poles and directly see the changes in the frequency response. Read the `mkiir` MATLAB program instructions at the end of this document, then launch the program. Place a couple of poles on the unit circle and move them along the unit circle. Look at the amplitude response and describe your findings. Remove the poles, place a couple of zeros, and study the effects.

Include in lab report:

- Describe the effect of placing poles close to zeros. Include an example screenshot from the `mkiir` program to illustrate your observations.
- Describe the effect of having a second-order pole on the unit circle. Include an example screenshot from the `mkiir` program to illustrate your observations.
- Describe what happens when you move a zero from $\omega = 0$ to $\omega = \pi$? How do you explain this?

Lab 1B

Introduction

In this part of the lab you will learn about designing basic filters to remove unwanted effects from a distorted signal. You will also study the echo effect in audio recordings and implement a method to remove the echo effect, using inverse filtering.

It is highly recommended to use your headphones when listening to the audio signals in this part of the lab. This will help you hear the effects more clearly.

Preparations

Before you start the lab tasks, you should make sure you have completed the following preparations.

Software and files

You should already have the necessary software and files available on your computer, after preparing for Lab 1A. If not, see the Preparation instructions for Lab 1A, above.

Preparation exercises

Echos in a recording environment typically stem from acoustic waves bouncing off structures in the environment. We can model this phenomenon as multiple delayed copies of the original sound, arriving at the microphone with different delays and attenuations.

Problem 3. Assume a speech signal $x[n]$ has been recorded in an environment with an echo effect, such that a D-samples delayed version of the original signal, attenuated by a factor α, is added to the original signal. Define a system to model this effect.

a) Find the difference equation describing the input-output relation and the time-domain impulse response of this echo system.

b) Determine frequency response of the filter, for $D = 5$ and $\alpha = 0.8$. Plot the magnitude response and the pole-zero diagram of the echo system.

c) To remove this echo effect, one can apply the inverse of echo system to the echoed signal. Find the frequency response of this inverse filter and determine the poles for arbitrary D and α.

d) Determine the frequency response of the inverse filter for $D = 5$ and $\alpha = 0.8$. Plot the magnitude response and the pole-zero diagram of the echo-canceling filter. What do you observe when comparing it to part b).

e) It turns out that the mentioned, straightforward, inverse filtering approach can fail (in terms of stability) under certain conditions. For which values of α does this happen?

Include in lab report:

- Solutions to the preparatory problem with exercise numbers included.
- The MATLAB code you have used when performing any computations or plotting

Lab tasks

Lab task 6. We will now focus on our original high quality music (from Lab 1A) and study the effect of applying other simple filters on the signal. Import the music file again. Also, re-create the ⟨*time vector*⟩ you used for plotting the time-domain signal in Lab Task 2 of Lab 1A. You are going to add a distortion to the music, and remove it using a simple FIR filter with a couple of zeros on the unit circle. Add a one-tone distortion signal with a frequency of $F_s/4$ and amplitude 0.1 to the original signal (Hint: you can use the ⟨*time vector*⟩ you have created before).

Include in lab report:

- Listen to the distorted signal and plot its power spectral density (using the provided `Spectrum_PLOT` function). Include the frequency domain plots and describe your observations, when listening to the signals and looking at the plots.

Lab task 7. You will now implement the filter you designed in the first preparation problem from Lab 1A. Open the `mkiir` program again. Set the filter gain to 0 dB, add a couple of zeros on the unit circle at the 'right place'. Export the filter as *test_filter.mat* and save it in your current MATLAB directory. Load the filter coefficients into your workspace by giving the MATLAB command:

```
load(test_filter.mat)
```

Two new variables, `a` and `b`, should appear in your workspace. You can apply this filter to your distorted music, using the MATLAB command:

```
filter(b,a,⟨Input Signal⟩)
```

Include in lab report:

- Listen to the signal and plot its spectrum. Describe your observations. Did you manage to remove the distortion? Does the signal quality remain the same? If you have not removed the distortion, check again if you have placed the zeros at the right place.

Lab task 8. We will now try to manually design filters, without the `mkiir` program, to remove distortions from some music files. Select one of the music signals from the `distorted_music` folder. Import the signal and its sampling frequencies into MATLAB. Listen to the signal and plot its power spectral density using the `Spectrum_PLOT` function. Can you see the sources of distortion? Examine the plot and write down the frequencies of the distortions. You are going to design an FIR filter to remove those unwanted frequency components.

The MATLAB command `filter` (as described in Lab Task 7) needs the coefficients of the numerator and denominator of its system function, in vectors `a` and `b`, respectively. To find those coefficients,

we use the MATLAB commands `b=poly(z)` and `a=poly(p)`, where vectors `z` and `p` contains the zeros and poles of the filter. Apply the filter to the distorted music file.

If you listen to the signal after the filter, you should notice that most of the distortion (beep sounds) is gone. However, you may still hear some distortion! The reason is that the `Spectrum_PLOT` function doesn't have a high enough resolution and therefore it is not accurate enough for some of the music files, and when you are plotting the spectrum, you can only see the most dominant distortion in the signal. However, there may be a couple of more distortions added to the music that you will not be able to see clearly. If you are interested in finding all the distortion frequencies and remove all the unwanted frequencies, you should use the following MATLAB command for plotting the signal spectrum:

```
figure
pwelch( ⟨Input Signal⟩ , [] , [] , [] , ⟨Sampling Frequency⟩ )
```

Using this function, you should be able to see the distortion frequencies more clearly now, and if you place the zeros and poles of the filter correctly, you can hear a clear undistorted music.

We have provided one more MATLAB function, `Spectrum_DoublePLOT`, making it possible to compare the power spectral densities of input and output signals of a filter:

```
Spectrum_DoublePLOT(⟨Input Signal⟩,⟨Output Signal⟩,⟨Sampling Frequency⟩)
```

Include in lab report:

- Listen to the filtered signal and plot its power spectral density. Have you removed the distortion frequencies? Did this filter affect the music quality?
- Use the function `Spectrum_DoublePLOT` to plot the power spectral density the original and filtered signal together. Include your plot, describe your findings, and compare to the listening experience.

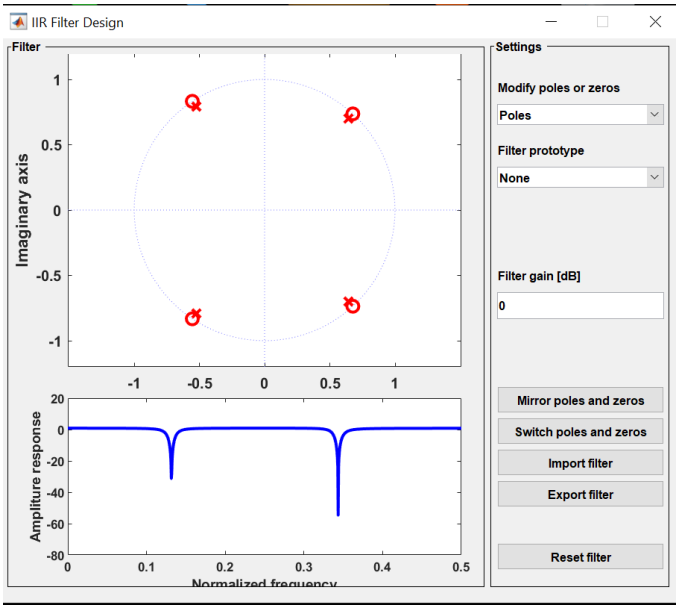
Lab task 9. The problem with the filter designed above is that, in addition to filtering the unwanted frequencies, it also attenuates the neighbor frequencies, which reduces the music quality. To overcome this issue, we will use an IIR notch filter. An IIR notch filter only removes the unwanted frequencies at more exact locations than the FIR filter from previous Lab task, reducing the impact on other frequency components. To create such a filter, after adding a zero on the unit circle, we also add a pole inside the unit circle close to the zero (e.g. at the radius of 0.95 from the origin). As an example, see the figure below which is taken from the `mkiir` program.

Now let's get back to the filter design from previous Lab task. Set the poles at the proper locations close to the zeros and define the filter coefficients again in vectors `a` and `b`. Repeat the previous Lab task, with this new filter.

Include in lab report:

- Listen to the filtered signal and plot its spectrum. Have you removed the distortion frequencies? Did this filter affect the music quality?
- Use the function `Spectrum_DoublePLOT` to plot the power spectral densities of the original and filtered signals together. Include your plot, describe your findings, and compare to the listening experience.
- Compare the performance of this IIR filter to that of the FIR filter from the previous Lab Task, in terms of removing the distortion.

Lab task 10. In this task, we will study the effects of echo on the audio signals. You will also learn how to remove an echo from an audio signal. Since the echo effect happens as a result of sound



reflecting off of different objects in the environment, we simulate the echo effect by adding a delayed version of the signal to itself.

Let's assume that you have recorded speech in a room with an echo coming from one reflection with an excess delay (compared to the wanted speech signal) of D samples, corresponding to a sound traveling an extra 50 meters at 340 m/sec. Select one of the speech examples from `LabFiles_1` folder and import it into MATLAB. Add an echo effect to the signal, with the same strength as in part b) of preparation Problem 3, and select the excess delay D based on the description above.

Add the echo effect to the speech file you selected and save it in another variable. Listen to the signal with the echo included.

Include in lab report:

- Describe how you calculate the excess delay D.
- Plot the original and the echo version of the signal in the time domain and compare them.
- Add the echo effect to other speech files and apply echos with different delay values.
- Include your MATLAB code, plots, and describe your listening experience.

Lab task 11. We can now implement a filter to remove the echo effect. The method we use is called inverse filtering, which is basically applying a filter to the echoed signal with a frequency response equal to the inverse of the echo effect. Take the echo effect from previous task and design the proper IIR filter to remove the echo. Specify the vectors `a` and `b` in the `filter` command and apply it to the echoed signal and listen to the output.

Include in lab report:

- Plot the original signal, the echoed version, and the signal after removing the echo effect (in the time domain) and compare them.
- Add the echo effect to other speech files and apply the corresponding inverse filters.
- Listen to the signals and describe your listening experience.

MKIIR Program

IIR Filter Design Application

The program *mkiir*, allows you to visually design an IIR filter by placing poles and zeros in the z-plane. The program shows a complex pole-zero plane (top) and the filter amplitude response (bottom) on the left hand side, and program options on the right hand side.

Pole-Zero Plot

The pole-zero plot shows the unit circle and the real and imaginary axes. This area is where you place, move and delete your poles and zeros. Poles and zeros are automatically placed in complex conjugate pairs. Therefore, only IIR filters with odd-ordered numerator and denominator polynomials can be designed by and imported to the program.

Amplitude Response

The amplitude response updates continuously with the current filter response as you add, move or delete poles or zeros from the filter. This area can also display a prototype filter specification you can attempt to design using poles and zeros.

Program Options

The settings panel on the right hand side allows you to control the behaviour and actions of the filter design application.

Modify Poles or Zeros

Select whether to add, move and delete poles or zeros. Poles or zeros, depending on this option, are added to the filter by left clicking on the pole-zero plot. Poles or zeros are moved by left clicking and dragging an existing pole or zero, and are removed by left clicking near an existing pole or zero.

Poles and zeros can only be placed on or inside the unit circle on the z-plane. Poles or zeros placed outside the unit circle will be projected onto the unit circle so that you can place a point exactly on the unit circle.

Filter Prototype

Show a filter prototype in the amplitude response area. Selecting a filter prototype overlays a pre-defined design constraint on the amplitude response for you to design a filter against.

Filter Gain

Set a global filter gain or attenuation to vertically shift the amplitude response. This option allows you to scale the frequency response to fit within the design constraint.

Mirror and Switch Poles and Zeros

Mirror poles and zeros across the imaginary axis, or switch poles to zeros and zeros to poles.

Import and Export Filter

Import pre-designed filter coefficients from MATLAB, or export your filter to MATLAB. This allows you to design a filter in MATLAB and import it into the design application to see how different filter designs places the poles and zeros, or to export your filter coefficients for use in MATLAB.