#### Texas Tech University

#### Department of Electrical and Computer Engineering

#### ECE 4264 / 5364 Digital Signal Processing

#### Fall 2019 Group D1

# Homework 6

Due October 18 before midnight CDT.

Please submit to Blackboard exactly two files:

1. A single ZIP file with all your MATLAB programs, along with any data files necessary to run your programs.
2. A single report document in Word or PDF. Your report should include procedures and solutions to analytical problems, screenshots of results and graphs from programs, and discussions. Please do not include copies of your code in the report.

Submit your solutions to Blackboard by the date and time listed to avoid penalties.

This homework is individual. You are welcome to discuss your approach with others, but your answers need to be unique. If your answers are found to be highly similar to others’ or to some material published, you may be subject to a review of academic integrity.

Undergraduate students only need to complete Part 1. Graduate students should complete both Part 1 and Part 2.

## Part 1 (Both ECE 4364 & 5364 students)

### Problem 1 (15 points)

We wish to create a solution to decimate audio files from 48000 samples/second to 8000 samples/second, without causing too much distortion to the frequencies that are kept.

1. Design an anti-aliasing FIR filter that you consider appropriate for this purpose.

* Explain your design choice: passband ripple, stopband attenuation, fs, fc, design method

**I used the standard fir1() function in MATLAB to design all of the filters on this assignment. For the anti-aliasing filter, I had a cutoff frequency of 0.16π normalized frequency since the signal needs to be decimated by a factor of 6. This filter also has a sampling frequency of 48 kHz. According the IFIR video, anti-aliasing filters need to be applied BEFORE they’re down sampled. My stopband attenuation ended up being around 50 to 60 dB. The only downfall of this design is the coefficient amount of 115.**

* Plot the frequency response of your filter

A close up of a piece of paper

Description automatically generated

1. Apply your filter to audio from toneramp.wav (still at 48000 samples/second)

* Plot the resulting amplitude over time

A picture containing screenshot

Description automatically generated

A screenshot of a cell phone

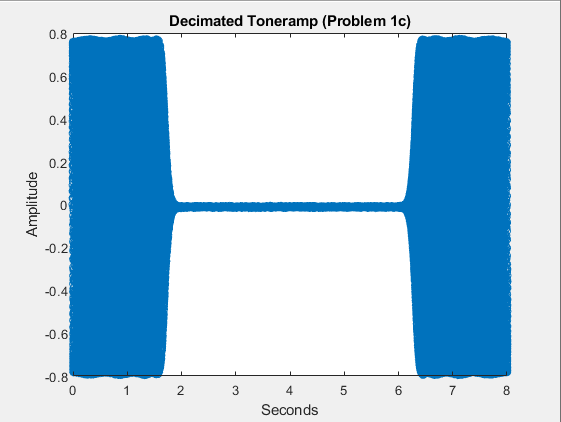
Description automatically generated

* Explain why this result makes sense

**I plotted two different graphs. The top graph is the amplitude response of the tone ramp signal and the second graph is zoomed in with x and y-axis limits from 0 to .0005 seconds so that more values would be visible and to help with Problem 1c solution. This result makes sense because of the number of samples to be taken is large due to the sampling rate.**

1. Implement the rest of the decimation process in MATLAB

* Save the decimated signal into a new wave file, which should have 8kHz sampling frequency. Submit this file as part of your solution.
* Plot the amplitude of the new wave file and compare it to your graph from part b). Explain what you observe, and why it makes sense.



A screenshot of a cell phone

Description automatically generated

**Because the signal was down sampled to a lower sampling rate, there is less information and values available to represent the original signal. It can’t really be seen in the first graph but in the second graph, the difference is obvious. In the zoomed in graph of 1b, there are more values available to represent the signal whereas in the zoomed in graph of 1c, the amount of samples has decreased tremendously.**

1. Apply your decimation to the audio file equinox-48kHz.wav.

* Save your decimated wave file and submit it as part of your solution.
* Listen to the audio file and compare it with the original 48 kHz version. Comment on what you hear, and why it makes sense.

**The decimated audio file sounds muffled and is lacking in the higher frequency range compared to the original. This makes sense because the original file had frequencies above 4 kHz that were successfully sampled but are not audible in the down sampled file due to the decreased sampling rate.**

### Problem 2 (15 points)

We wish to create a solution to interpolate audio files from 8000 samples/second to 48000 samples/second, without causing too much distortion to the signal present.

1. Implement the sample rate increase in MATLAB (by zero insertion)

* Apply this process to the audio file from problem 1, part c).
* Plot the spectrum of the signal before and after zero-insertion. Explain what you observe and why it makes sense. **When the spectrum is up sampled, images are created at a factor of 6. This is shown in the graphs below. This makes sense because every 8000 Hz, the original spectrum is repeated. This is resolved with an image-rejection filter.**

A close up of a map

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Design an FIR image-rejection filter that you consider appropriate for this purpose.

* Explain your design choice: passband ripple, stopband attenuation, fs, fc, design method

**The filter used here is the exact same filter as the one used for anti-aliasing in the first problem. The stopband attenuation, coefficient amount, sampling rate and cutoff frequency are the same and helps eliminate those images.**

* Plot the frequency response of your filter

A close up of a piece of paper

Description automatically generated

* Apply it to the result of part a)
* Plot the spectrum of the signal over frequency in Hz after filtering. Explain what you observe and why it makes sense.

A screenshot of a social media post

Description automatically generated

**In this plot, we can see that the images have been filtered out and the image-rejection filter was successful. The shape of this graph is the same shape as the original just mapped out over the larger sampling rate.**

Plot the amplitude over time of the filtered signal. Explain what you observe and why it makes sense.

A picture containing screenshot

Description automatically generated

**The amplitude response is very similar to the one at the beginning of Problem 1b. This response is showing how the signal at 8 kHz would be represented at a 48 kHz sampling rate. This makes sense because the response has been stretched to match these parameters.**

1. Apply your interpolation solution to the audio file from problem 1, part d)

* Save your interpolated wave file and submit it as part of your solution. This should be a 48 kHz file.
* Listen to the audio file and compare it with the original 48 kHz version. Comment on what you hear, and why it makes sense.

**The new 48k file sounds exactly like the 8k down sampled equinox from problem 1. This makes sense because with zero insertion, zeros are inserted instead of values. This gives the impression that it was sampled at 48k where it was sampled at 8k. When you down sample, information is reduced and compressed to fit an 8k sampling window. In upsampling, the information remains the same but is stretched to fit a larger sampling window.**

### Problem 3 (15 points)

We wish to create a solution that can split audio files at 48000 samples/second, into two audio files, each at 24000 samples/second. The first file should contain the lower half of the representable frequencies, and the second file the upper half of the frequencies.

1. Design a solution to extract the lower-frequency band (up to 12 kHz), decimate to 24000 samples/second, and save it to a wave file.

* Show the frequency response of your anti-aliasing filter

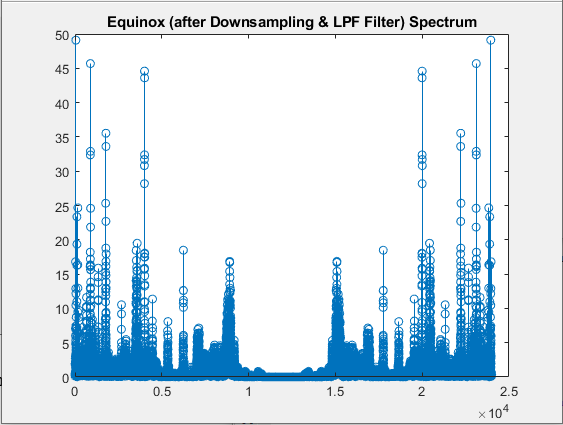
A screenshot of a cell phone

Description automatically generated

* Apply it to the audio file equinox-48kHz.wav
* Show the spectrum of the signal before and after conversion. Explain why this makes sense.

A screenshot of a cell phone

Description automatically generated



**This makes sense because the lower 12 kHz stays the same while the higher frequency content is filtered out.**

* Listen to the resulting audio file. Explain what you hear and why it makes sense.

**It sounds almost as full as the original but it’s obviously missing a lot of that higher frequency content. This makes sense and proves the anti-aliasing filter works.**

1. Design a solution to extract the upper-frequency band (from 12kHz onward), decimate to 24000 samples/second, and save it to a wave file.

* Show the frequency response of your image-rejection filter

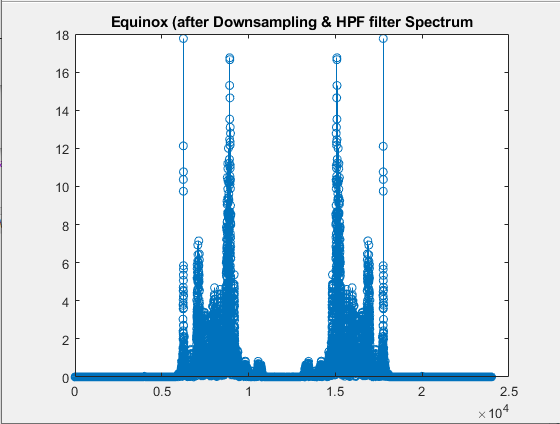
A close up of a piece of paper

Description automatically generated

* Apply it to the audio file equinox-48kHz.wav
* Show the spectrum of the signal before and after conversion. Explain why this makes sense.

A screenshot of a cell phone

Description automatically generated

\

**This makes sense because all of the lower frequency content has been filtered out and the high content remains. The amplitude content has been halved as expected.**

* Listen to the resulting audio file. Explain what you hear and why it makes sense.

**When listening to the audio file, it mainly consists of high frequency content. Most of it is not audible because the frequencies of our ears don’t go past 20 kHz. This makes sense because the bulk of the sound is in the lower range and not in those upper frequencies thus successfully proving that all of the high frequency content has been separated.**