BELMONT UNIVERSITY, CURB COLLEGE OF ENTERTAINMENT AND MUSIC BUSINESS

AET 5410 Homework 2

Digital Audio, Computer Programming, Signal Analysis

Due: October 16th, 5:00 pm

1 DYNAMIC RANGE CREST FACTOR

One way to analyze the loudness of a signal is to consider the maximum signal level (peak amplitude) and the average signal level (RMS amplitude). A signal's *crest factor* is defined as the ratio of peak amplitude to RMS amplitude.

$$Crest\ Factor = \frac{|A_{peak}|}{A_{rms}} \tag{1.1}$$

By applying dynamic range reduction, with compressors/limiters, the Crest Factor is decreased because the a_{rms} is increased without changing a_{peak}. Therefore, the Crest Factor is an analysis used to describe the relative amount of dynamic range in a signal.

A common transformation of the calculation is to put the crest factor on the decibel scale.

$$Crest\ Factor_{dB} = 20 \cdot log_{10}(|A_{peak}|) - 20 \cdot log_{10}(\frac{A_{rms}}{0.7071}) \tag{1.2}$$

Audio engineers use this an an analysis of "mastered" music. Typical values range from DR4 (less dynamic range) to DR10 (more dynamic range).

1.1 PROBLEM

Create and save a **script** (m-file) in MATLAB that performs the following steps:

- Name the script: `crestFactor.m`
- Import the provided sound file, *'HW2.wav'*

- Determine the peak amplitude of the signal
- Calculate the RMS amplitude of the signal
- Calculate the Crest Factor of the signal using Equation 1.1
- Convert the linear Crest Factor to the decibel value using Equation 1.2
  - Print the decibel Crest Factor to the Matlab Command Window

Remember to add comments to your code to explain what each command is accomplishing. For this problem you will just submit the script.

2 Calculating the Crest Factor Over Time

Besides calculating the crest factor, peak amplitude, RMS amplitude, etc. for an entire signal, audio engineers may want to consider how theses measurements changes over time for short signal segments (see: DRMeter, https://www.maat.digital/drm2/).

In this case, the signal is divided into short segments, also sometimes called frames or *buffers*. The length of each segment is usually selected to be 512, 1024, 2048, or 4096 samples. Then, the crest factor is calculated for each buffer.

2.1 Problem

For this problem you will create a MATLAB **script** to calculate the crest factor of a signal over time based on the following specifications:

- Name the script: `cfSegments.m`
- Import the provided sound file, *'HW2.wav'*
- Create a variable called *bufferSize*
  - It is the number of samples to be included in each segment/window
  - Experiment with different values to see the result
- Initialize the following empty arrays to store the analyzed values for all the buffers:
  - *peakAmp* - array of peak amplitudes, each element belongs to a different segment
  - *rmsAmp* - array of rms amplitudes, each element belongs to a different segment
  - *DRdB* - array of crest factor values on a decibel scale (see Problem 1)
- Then perform the following steps:
  - Index a segment of the signal based on the *bufferSize* starting from the first sample
  - Analyze the various amplitude measurements for this segment
    - You can use the built-in MATLAB functions in this problem

  - Append the appropriate values to the corresponding arrays
  - Repeat these steps for another segment, starting with the sample after the end of the previous segment, until you reach the end of the signal
  - *Hint:* this will work best if you use a loop

- Create a plot similar to Figure 2.1 that shows the waveform along with the analyzed Peak array, RMS array, and Crest Factor array. (Use `subplot`)

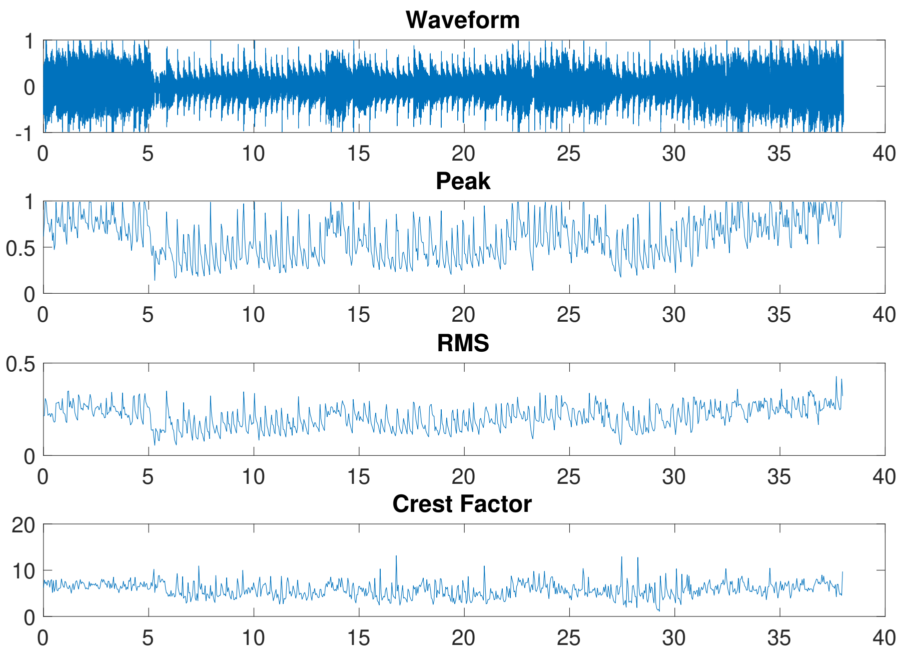

Figure 2.1: Output Plot from Test Script

3 Logarithmic Sine Sweep

One type of signal that is commonly used for tests in audio is the sine sweep. This is a signal with one frequency at any point in time (i.e. sine wave). The frequency of the sine wave changes (sweeps) throughout the duration of the signal from a low frequency (e.g. 20 Hz) to a high frequency (e.g. 20 kHz).

There are a couple of variations of the signal based on the rate at which the frequency sweeps from low frequencies to high frequencies. One type is the **linear sine sweep**, that increases at a linear rate. In other words, the time it takes to transition from 100 Hz to 200 Hz is the same time it takes to transition from 10,100 Hz to 10,200 Hz.

Another type is the **logarithmic sine sweep**. In this case, there is equal time spacing based on octaves. The time it takes to transition from 100 Hz to 200 Hz is the same time it takes to transition from 10,000 to 20,000 Hz.

When synthesizing any type of sine sweep, it is common to specify the starting (lowest) frequency and the ending (highest) frequency. Additionally, the length of the signal in seconds and the amplitude in decibels should be specified.

3.1 Problem: Synthesizing the Signal

For this problem you will write a MATLAB script to synthesize a logarithmic sine sweep.

- Name the script: `logSineSweep.m`
- Create the following variables near the top of your script to adjust to synthesized signal
  - `Fs` : pick a common audio sampling rate
  - `f1` : lowest frequency (Hz) at the start of the sweep
  - `f2` : highest frequency (Hz) at the end of the sweep
  - `sec` : duration in units of seconds
  - `dBAmp` : amplitude (dB)
- It is recommended to first figure out how to synthesize a linear sine sweep
  - Make use of the MATLAB `sin()` function
  - It will be necessary to convert the variables declared above to other units
  - Remember this is a sine wave that has to change frequency smoothly
  - Scale the overall amplitude appropriately
  - Listen to the result to see if it is working
  - Keep this section of code in your submitted script
- Next, figure out how to synthesize a logarithmic sine sweep
  - Convert `f1` and `f2` to a normalized frequency scale
    * $\texttt{normF1} = \frac{log_{10}(\frac{\texttt{f1}}{20})}{3}$
    * $\texttt{normF2} = \frac{log_{10}(\frac{\texttt{f2}}{20})}{3}$
  - Create an array (`normArray`) of linear-spaced values between `normF1` and `normF2`
  - Convert this entire array back to the frequency (Hz) scale
    * $\texttt{freqArray} = 20 \cdot 10^{(3*\texttt{normArray})}$
  - Use `freqArray` when synthesizing the sweep

4 SUBMISSION

To submit your homework, create a single zip file that contains the MATLAB m-file scripts and the sound files you used in the problems. Name the zip file: xxxxx_AET5410_HW2.zip, where *xxxxx* is your last name. Email the zip file to: eric.tarr@belmont.edu before 5:00 pm on October 16th.