Carnegie Mellon DSP Electrical & Computer ENGINEERING

Digital Signal Processing (18-491/18-691)
Spring Semester, 2023

The 18-691 Final Project: A DIY Audio Effect Processor (Preliminary version)

Issued: 4/21/23

Due: 4/30/23 at midnight via Gradescope

Note: This version is preliminary in that it is lacking data and infrastructure resources. These files will be provided tomorrow, April 22.

Background: At the beginning of the semester we noted that students enrolled in 18-691 will be required to complete one additional assignment in order to receive graduate credit for the course. This is necessary in order to comply with ECE Department rules that state that courses in multiple tiers with parallel numbers must have at least somewhat different content to justify the parallel numbering.

The final assignment for the 18-691 students will be an implementation of two algorithms that implement popular audio effects for musical performance. The project is intended to be easy and fun.

The final project will be given a grade equal to one homework assignment. Nevertheless, the grade for the final project will not be dropped from the final grade calculation, no matter how low it is. Each 18-691 student will work on his or her own on this project. The deadline for the final assignment will be April 30, the Sunday after the last day of classes at midnight.

Digital distortion

While audio processing and high fidelity most commonly strive for reproducing the signals as accurately as possible, in the late 1960s rock artists such as Jimi Hendrix, Eric Clapton, and others began introducing signal distortion deliberately as a form of expressiveness in music production. (For example, distortion can be heard in varying degrees in the Beatles songs "Think for Yourself," "Revolution #1," and "Helter Skelter," among others.) While this was first accomplished using analog circuitry (initially referred to as "fuzz boxes"), discrete-time implementations became increasingly popular as time went by for all the usual reasons.

Signal distortion can be produced in an almost unlimited number of ways. The simplest type of distortion is "hard limiting" or "clipping" in which the magnitude of the signal amplitude is clamped to a fixed limit. Consider a simple discrete-time nonlinear system with input $x[n]$ and output $y[n]$. Clipping the output input is easily described by the equation

$$y[n] = \begin{cases} C, & x[n] > C \\ x[n], & -C \leq x[n] \leq C \\ -C, & x[n] < -C \end{cases} \tag{1}$$

If the input signal $x[n]$ is itself normalized such that $-1 \leq x[n] \leq 1$, the severity of the clipping (and hence the amount of distortion produced) will increase as the limiting threshold C is decreased from 1 toward 0. Note that $0 \leq C \leq 1$

Unfortunately, the imposition of distortion even for a band-limited signal will produce an infinite number of "distortion products," which is the term that describes the additional high frequency components that results from the nonlinearity. In general, nonlinearities that are more severe will produce more and more distortion, and with the higher frequency components having increasingly higher magnitudes. This is a problem for systems implemented in discrete time because eventually there will be significant energy in the distorted signal at frequencies that exceed the Nyquist frequency (*i.e.* half the sampling rate), and the result will be a signal containing both the desired nonlinear distortion associated with the clipping process and also the undesired nonlinear distortion associated with the aliasing of components of the distorted signal.

The problem of the introduction of aliasing distortion is generally resolved by upsampling the signal, clipping the upsampled signal, and subsequently downsampling the clipped signal by an equal conversion ratio. Clipping the signal in upsampled form and clipping the signal directly are illustrated in Fig. 1.

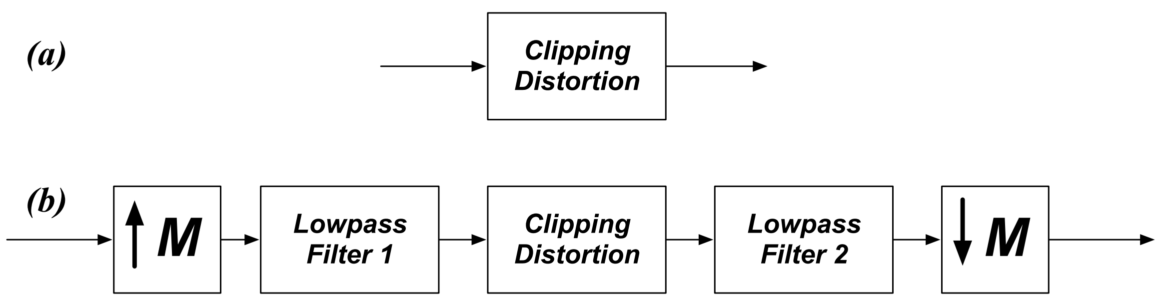

Figure 1. (a) A simple discrete-time clipping system. (b) A clipping system implemented with upsampling (oversampling) to reduce aliasing distortion.

Digital reverberation

The second discrete-time effect we will examine is simulated reverberation. If a sound source is produced in a normal room, the signal that arrives to the listener will include a direct component that travels directly from the sound source to the listener, but it will also include additional components of the sound that are delayed and attenuated. These are caused by components of the sound source reflecting off the surfaces of the walls, floors, and ceiling of the room, along with the people and objects in the room. An example of a typical room impulse response (RIR) is depicted in

Figure 2. Note that the initial pulse representing the direct component is followed by a number of early discrete reflections, and ultimately by a long decaying exponential as the discrete reflections become blurred together.

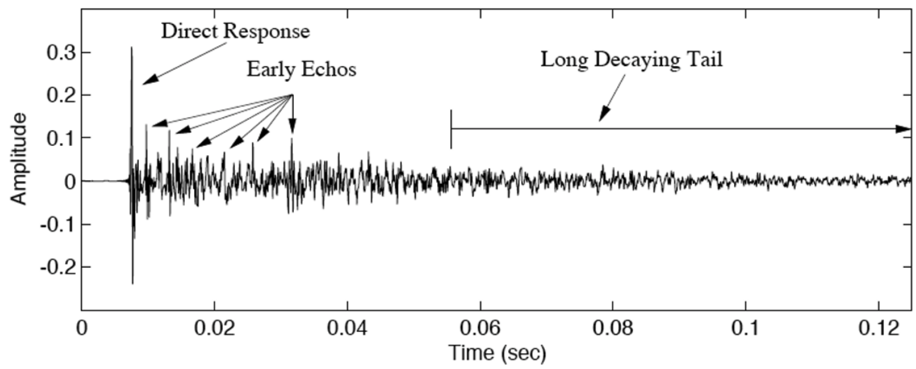

Figure 2. Example of a unit sample response of sound recorded in a room.

In practice, estimates of the RIR can be obtained in multiple ways. The RIR can be estimated directly by recording the response to a spark or a punctured balloon. Alternatively, it can be inferred from the response to a sine wave generator with a frequency that slowly sweeps through the audio frequency band. The RIR can also be estimated by passing white noise through a broadband loudspeaker and then computing the cross-correlation of the generated signal and the signal that is picked up by the microphone. Measured RIRs for a number of rooms are available for download at various sites on the worldwide web.

Simulated reverberation is accomplished directly by convolving a “clean” source of speech or music with the unit sample response of the room. Because the input signals can be arbitrarily long, this convolution is typically implemented using FFTs and overlap-save or overlap-add. algorithms.

For your work in implementing discrete-time distortion and reverberation, we will provide some general resources and guidance, but it will be up to you to make reasonable design decisions. Your work will be evaluated based on whether your systems work properly, and on your description of how you arrived at your critical design choices.

Note: You should implement these problems for the most part “by hand,” or without using MATLAB routines. This should not be difficult for the most part. MATLAB **may** be used for filter design ysubg the routine `filterDesigner`. The MATLAB routines `fft` and `ifft` may also be used. Let the teaching staff know via Piazza if you have any questions about these boundaries.

Problem C10.1: (Digital distortion)

What is provided to you:

- A sample of a sine wave of 5 kHz, with amplitude 1. All waveforms in this problem are sampled at 16 kHz.
- A sample of a guitar riff at the same sampling frequency

- A distorted version of the guitar riff, generated according to the principles described in this problem

(a) Pass the sine wave through the clipping distortion specified by Eq. (1) at the top of page 2 of this assignment. As you decrease the clipping threshold parameter C from 1 to 0, describe how the upper frequency of distortion changes. What range of clipping thresholds will be likely to cause harmonics with substantial power to fall beyond the Nyquist frequency?

(b) Suppose that we select a clipping threshold of approximately $C = 0.15$. Based on the distortion frequencies that you observed in your answers to part (a) for that value of C, choose an upsampling/downsampling ratio M that will ensure that the higher distortion frequencies fall well below the Nyquist frequency at the upsampled sampling rate. Justify your choice.

(c) Write the specifications of the Lowpass Filters 1 and 2 in Fig.1, that appear to be reasonable for the upsamplng and downsampling based on your answers in parts (a) and (b). With the help of the MATLAB package `filterDesigner`, design two filters that can meet the desired specifications:

1. an IIR filter using an elliptic prototype
2. an FIR linear-phase equiripple filter using the Parks-McClellan algorithm

What are the orders of the two filters? Plot the magnitude and phase of each of the two filters using the MATLAB routine `freqz`.

(d) Obtain audio results for the distortion systems using the IIR and FIR filters by implementing the difference equation that is specified by the two filters. Listen to the audio. Does the distorted signal sound like the signal that we provided as an example? Do you hear differences between the FIR implementation (which uses linear-phase filters) and the IIR implementation (which is more efficient but has nonlinear phase)?

Problem C10.2: (Digital reverberation)

What is provided to you:

- A sample room impulse response (RIR)
- Longer segments of music to be reverberated. These signals will have been recorded in a relatively non-reverberant (or "dry") environment.

(a) Using the RIR provided, estimate the reverberation time of the room. By definition, the reverberation time is the time that it takes for the impulse response to decay by 60 dB (*i.e.* a factor of 1000). To estimate the RIR, compute and plot the magnitude of the RIR in decibels. Plot the instantaneous amplitude in dB as a function of time. There should be some initial threshold followed by a section of linear decay followed by a section that is approximately constant as the response falls to the noise floor. The linear portion of the response most likely will fall for less than 60 dB, but you can estimate the reverberation time by examining the slope of the linear portion of the curve and estimating how long it would take for an extrapolation of the linear curve to fall by 60 dB.

(b) Now convolve the impulse response (RIR) that is provided with the input signal that is provided. Because the RIR is finite in duration you can (and should!) use the overlap-add or overlap-save algorithm to perform the convolution. State how you choose the DFT size, the block size of each section, and the amount of overlap. Write the code to perform the convolution yourself, and do not use MATLAB routines for the convolutions. Listen to the audio. It should sound exactly as it would if you used the MATLAB routine `conv` to convolve the input and sample response. Does it? Does the signal sound as if it is in a reverberant room? How does the execution time using overlap-add or overlap-save compare to what is obtained convolving the input with the RIR directly in time?

(c) Repeat parts (a) and (b) using an RIR that you have obtained from the internet. (They are provided in multiple locations; try searching for "room impulse response.") Try to use an RIR that has a relatively long reverberation time. Listen to the resulting audio and provide your assessment of whether the reverberation sounds realistic.