Signals and Systems

(10820EECS202002)

Computer Homework #2: Reverberator implementation and your first experience of system design

Due: 24:00, 04/26/2020

Part 1. Reverberator implementation

At the end of Topic 2, we discussed a digital recursive IIR reverberator, which is also known as "comb reverberator" and as the name indicates, will colorize the input sound. You will study the coloration issue later on in Topic 3. In this homework, you will try to implement an "allpass reverberator" which is introduced by Schroeder and Logan (1961) to prevent "coloration" of the input sound by the comb reverberator. The allpass reverberator can be modeled or implemented via the following recursive LCCDE.

$$y[n] = ay[n-D] - ax[n] + x[n-D],$$

where all the symbols in the LCCDE share the same definitions as we discussed in the class (see slides 44 to 47 in Topic2_LTISystems_Part2_wNote.pdf). With this LCCDE modeling, please try to answer the following questions.

(a) Derive the impulse response of this allpass reverberator, and then use the MATLAB function **filter()** to compute and plot the impulse responses of this reverberator to verify your derivation , where $a = 0.7$, and $D = 5$.

(b) Please comment whether this reverberator can be potentially implemented in real time and under what condition of a and D this allpass reverberator is stable. Prove them.

(c) Approximate this allpass reverberator by a FIR system (see slide 46, in Topic2_LTISystems_Part2_wNote.pdf) and implement it using the MATLAB built in convolution function **conv()** with $a = 0.7$ and $D = \tau Fs$ where D is an integer representing the digital time delay given the continuous-time delay $\tau = 0.1$ sec. and sampling rate Fs (in Hz). Given the sound file **Halleluyah.wav**, you can load the file, play the sound and save the output sound by using the following MATLAB codes:

```
[x, Fs] = audioread(' Halleluyah.wav');
sound(x, Fs);
audiowrite(y,Fs, ' Halleluyah_FIRecho.wav');
```

where x is the input sound (a column vector), y is the output sound (a column vector, too), and Fs is the sampling rate in samples/sec (i.e., in Hz).

Please elaborate how you approximate the reverberator by an FIR system. That is,

elaborate how you determine the order of the FIR system (i.e., the M in the general LCCDE in slides 21 and 22, in Topic2_LTISystems_Part2_wNote.pdf). Also plot x and y together as a function of time and check the changes in x done by your FIR system. You may need to zoom in the plot to see the changes in details. The simplest way to see the change is to show y – x (which I call trashogram). Please tell what initial condition is used when you implement it using the ready-to-use MATLAB function **conv()**.

(d) Write **your own MATLAB codes (see the pseudocodes in slide 48, in Topic2_LTISystems_Part2_wNote.pdf)** to implement this reverberator using the provided recursive LCCDE with *a* and *D* being the same values used in (c). To generate digital reverberation, we will use the same sound file in (c), and save your output sound as **Halleluyah_IIRecho.wav**. Please verify your result using MATLAB function **filter()**. Note that you have to provide the **initial condition** (i.e., y[n] and x[n], when n < 0) used in your implementation in the report. Also plot x and y together as a function of time, check the changes in x done by your IIR filter, and comment the output differences between the two different implementations in (c) and (d). The simplest way to see the difference is to show the result of (c) minus that of (d) (which I call trashogram). Note that you may need to zoom in the plot to see the changes in details.

(e) Change the *a* in (d) to the values resulting in an unstable reverberator, and grasp the feeling about what the output of an unstable reverberator sounds like, also with the sound file used in (d) as the system input. Here you can use MATLAB function **filter()** directly if your own IIR system implantation is too slow.

(f) Change the initial condition (e.g., y[n] = x[n] = 1, when n < 0, or y[n] and x[n] are random numbers (try MATLAB function **rand()**), n < 0. Note that you have to try to scale the initial conditions y[n] or x[n] to the values close to your input) which you used in (d) and see how the initial condition affect the results. Plot the outputs with different initial condition together as a function of time and tell the difference, also with the sound file used in (c) as the system input and the *a* and *D* the same values used in (c).

(g) Following (f), verify whether the system output y[n] is equal to the zero-input response plus zero-state response.

Part 2. Your first experience of system design – Noise remover for echo time estimation

Parking sonar estimates the echo time of the sonar sound propagating back and forth between a car and an obstacle to figure out the car-to-obstacle distance. Generally, the parking sonar will prefer to employ a windowed/weighted linear FM

signal as the transmit signal, e.g., **x** given in the provided sample codes, and then will receive an echo signal contaminated by noise, as **y** given in the provided sample codes. **y** is contaminated by Gaussian white noise and is recorded at a sampling rate of **Fs** in Hz, also given in the provided sample codes. With the unprocessed **y**, you will find out it is impossible to figure out the echo time. Note that there is only one reflector for y. To perform the echo time estimation, the first step will be noise removal. Design an FIR LTI system which can suppress the noise and help the echo time estimation, elaborate how you design the system (i.e., impulse response) and why it can suppress the noise, and verify your idea using MATLAB (see slide 61, Topic2_LTISystems_Part1_HandWriting0408_2020.pdf).

Note that as for the definition of echo time and how to find out echo time, related codes have been provided in the sample codes. You also can find the ground truth of the echo time in the provided sample codes. For the system design and implementation, you may need MATLAB built in function conv() for convolution sum and fliplr() (or flipud()) for time reversal operation.

Notice:

1. Please hand in your solution files to the LMS elearning system, including your word or pdf file of the detailed solutions, the associated Matlab codes, and all the related materials. It would be nice that you can put your "KEY" code segment with comments side by side along with your answer in the word or pdf file whenever necessary.
2. Name your solution files "EECS2020_HW2_StudentID.doc" (or "EECS2020_HW2_StudentID.pdf") and "EECS2020_HW2_StudentID.m", and archive them as a single zip file: EECS2020_HW2_StudentID.zip.
3. The first line of your word/pdf or Matlab file should contain your name and some brief description, e.g., % EECS2020 Calvin Li u9612345 HW2 MM/DD/2020
   .