Signals and Systems
(10920EECS202002)
Computer Homework #2: Convolution sum, echo time estimation, and reverberator implementation

Due: 24:00, 04/18/2021

Part 1. Convolution sum and echo time estimation - 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 transmitted signal, e.g., **x** given in the provided sample codes, and then will receive an echo signal contaminated by noise, as **y** (get **y** by **load NoisyEchoSignal**). **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. As mentioned in our lecture (slide 59, Topic2_LTISystems_Part1_HandWriting0331_2021.pdf), we can design an FIR LTI system to suppress the noise and help the echo time estimation.

(a) (5%) Please elaborate why the impulse response of the noise remover is design in the form presented in the given sample codes (see **h** in the sample codes of Part 1).

(b) (10%) Implement your own convolution sum function – **MyConv()** based on the given function protoype defined in the provided **MyConv.m**. Note that please use Example 2.4 (slide 50, Topic2_LTISystems_Part1_HandWriting0331_2021.pdf) to valid your own MyConv() and also compare with the results from MATLAB built in function **conv()**.

(c) (10%) Run the noise remover with **y** as its input, plot the noise suppressed signal as a function of absolute time, and then estimate the echo time from the noise suppressed signal.

(d) (5%) Please tell what is the minimum achievable error (in seconds) of the estimated echo time and justify your answer.

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.

Part 2. Reverberation implementation

The linear constant coefficient difference equation (LCCDE) shown below models a digital recursive IIR reverberator, which is known as “comb reverberator” and thus, as the name indicates, will colorize the input sound. You will study the coloration issue further in an easier way in Topic 3. In this homework, you will try to implement and tune this digital recursive IIR reverberator which is modeled as follows.

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

where a is the attenuation and D is the reverberation time delay in normalized time. With this LCCDE modeling, please try to answer the following questions.

(a) (10%) Derive the impulse response of this 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) (10%) Please comment whether this reverberator can be potentially implemented in real time and under what condition of a and D this reverberator is stable. Prove them.

(c) (10%) Approximate this reverberator by an FIR system (see slide 46, in Topic2_LTISystems_Part2_withNotes.pdf) and implement it using your own convolution sum function **MyConv()** created in Part 1 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). Note that you may verify your results by the MATLAB built in convolution function **conv().** 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_withNotes.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 (i.e., y[n] and x[n], when n < 0**) is used when you implement it using your own function **MyConv()**.

(d) (10%) Write **your own MATLAB codes (see the pseudocodes in slide 48, in Topic2_LTISystems_Part2_withNotes.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) (10%) 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 runs too slow.

(f) (10%) Change the initial condition (e.g., y[n] = x[n] = K, 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, e.g., K = the mean value of the input) which you used in (d), see and comment how the initial condition affects 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) (10%) As mentioned in the very beginning of Part 2, the implemented reverberator is a "comb reverberator". That means the system will colorize the input sound. That is, some frequencies of the sound will be suppressed and some frequencies will be amplified (or emphasized). Use the single tone sinusoidal signal in your computer homework #1 to figure out what frequencies (or pitches) of the sinusoidal signals will be suppressed and what frequencies (or pitches) of the sinusoidal signals will be emphasized by changing the frequency from 0 to Fs/2. For this purpose, please plot magnitude response of the system for frequency ranging from 0 to Fs/2. The definition of the magnitude response will be explained in class (see the topic of "Explanation of ComputerHW2"). Note that the *a* and τ (used to find D which varies with different Fs) are the same values used in (c).

Bonus (5%): Do you know why the magnitude response required in (g) is only plotted for frequency from 0 to Fs/2, instead of from 0 to Fs or even higher frequency, e.g., 2Fs? Please comment it (Hint: think about what you learn in Topic

1 and computer HW1).

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/2021
   .