Lab 5 – Convolution
EE252L: Signals and Systems Lab

Dr. Basit Memon and Ms. Hira Mustafa

Name: _____

Objectives

By the end of this lab, you will be able to get familiarize with continuous and discrete time convolution, its relation with LTI Systems and its application on 1-D systems using MATLAB.

Instructions: After completion of each task get it marked by sharing your results and code with any of the RA, your code must be self-explanatory i.e. use proper comments. *Attach rubrics at the end of your report*.

1 Convolution

This lab is centered around another operation that can be performed on signals, called 'Convolution'. Convolution has a central role in signal processing too, and you'll discover the same in this lab. Given two continuous-time signals $u(t)$ and $v(t)$, the continuous-time convolution operation can be defined as:

$$y(t) = u(t) * v(t) = \int_{-\infty}^{\infty} u(\tau)v(t-\tau)\,d\tau, \tag{1}$$

Where the symbol * denotes convolution, and it must not be confused with the multiplication symbol. The value of the convolution output at some time t is defined as the integral of the products of the two input signals, where one of the signals has been shifted in time by t and flipped.

Task 1 - Warm up

In your first task, you're provided a demo of continuous-time convolution (courtesy of Mathworks "Convolution in Digital Signal Processing" Courseware), and you'll play around with the inputs to get more familiar with the convolution operation.

- Download and open the `convolution_basics_1.mlx` file, located in the files section in folder `Lab5` on LMS.
- Use the sliders and drop downs to set different parameters for the signals. Click Run to view the convolution process and its result. The demo provides some suggestions to try at the bottom too.
- Answer the following questions:
  - Does the order of inputs matter? Is convolution always commutative, i.e. is $u(t) * v(t) = v(t) * u(t)$?
  - Describe your general observations about the convolution output w compared to the inputs u and v?

The convolution operation can similarly be described in discrete-time too. Given two discrete-time signals, $u[n]$ and $v[n]$, their convolution can be defined as:

$$w[n] = u[n] * v[n] = \sum_{k=-\infty}^{\infty} x[k]x[n-k].$$

A demo similar to continuous-time case is provided for discrete-time too. While the previous demo used a summation to approximate a continuous integral, discrete-time convolution can be exactly computed in MATLAB using `conv` command.

Task 2 - Warm up continues

- The demo for discrete-time convolution is located in the same file. Play around with the various parameters to enhance your understanding of discrete-time convolution.
- Answer the following questions:
  - How is the length of convolution output related to the length L of the two input signals?
  - What do you think will happen if the input sequences had different lengths? say, M and N, instead of L.

Can you use the `conv` command to compute continuous-time convolution? Yes; you can find out how in Appendix-A.

2 Why do we care about convolution?

In this section, we'll find out that the convolution operation provides a remarkable way to implement signal processing through linear-time invariant systems. We'll discover this by looking at a discrete-time system, shown in Figure 1, which computes the moving average of an input signal $x[n]$ to generate the corresponding output signal $y[n]$.

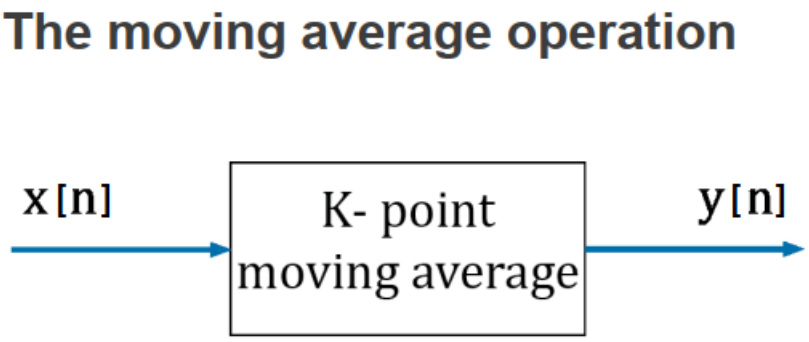

Figure 1: Moving Average Filter

The output of a K-point moving average system at any time n is simply the average (mean) of the input at the same time and some $K-1$ neighboring values of the input. For example, the output of a five-point centered moving average system is defined as:

$$y[n] = \frac{1}{5}\left(x[n-2] + x[n-1] + x[n] + x[n+1] + x[n+2]\right).$$

You may be surprised to know that this system acts as a simple low-pass filter on the data, i.e. it removes high frequency components from the signal or data. This is why the system is also known as a moving-average filter, as it removes unwanted noise. As the length of the filter increases, the smoothness of the output increases, whereas the sharp modulations in the data are made increasingly blunt.

Task 3: Experimenting with moving-average filter

In this task, you are going to implement a 5-point moving average filter. The MATLAB command `movmean` can be used for its implementation. Use MATLAB `help` to explore this command.

(a) Find the output of this moving average filter for the input `x[n]`, where $x[n] = z[n] + noise[n]$ and `z[n]=sin(0.3142*n)`, First generate a noisy signal using the command `randn` and add it to $z[n]$ to finally acquire $x[n]$, where `n=-20:20`. Store the output in a variable named `y1`.

(b) Find the output of this moving average filter for a unit-impulse input $x[n] = \delta[n]$. Store the resulting output in the variable `h`.

(c) Convolve `x[n]` from (a) and `h[n]` from (b), and store the output in `y2`.

(d) Plot $x[n]$, $y1[n]$, $h[n]$, and $y2[n]$ in the same figure and comment on the differences/similarities between $y1$ and $y2$. **Preffered plotting approach:**Use plot command instead of stem, you will be able to observe the graphs clearly.

The observation that $y1[n] = y2[n]$ is not a coincidence, and will hold for any $x[n]$. This is because the moving average system is a linear-time invariant (LTI) system, and it is a property of LTI systems that they are completely specified by their impulse responses, which is usually denoted by $h[n]$. More specifically, if the impulse response of an LTI system is known, one can compute the system output for any input signal. We now present on of most important topics in signals and systems theory:

"The response (or output) of an LTI system to any input signal is computed by the convolution of the input signal with the impulse response of the system, i.e.

$$y[n] = x[n] * h[n],$$

where $h[n]$ is the impulse response of the system, $x[n]$ is any input, and $y[n]$ is the corresponding output."

What is the impulse response? The impulse response of an LTI discrete-time system is the output of the system when the unit impulse is applied as input, as was the case in (b) of the previous task. Even though the unit impulse input consists of one nonzero term, the impulse response signal $h[n]$ usually consists of more than one nonzero elements. The explanation is that the system is dynamic (with memory); that is, the system responds over various time instances to the input applied at $n = 0$.

Task 4: Is moving-average filter an LTI system?

Prove that the moving-average filter is a linear time-invariant system analytically.

In the next task, you will see a powerful application of the output principle of an LTI system using convolution. You would have surely noticed that the same audio (you speaking or music) played in different physical environments sounds different. This is called reverberation, an audio

effect, resulting from the interaction of sound waves with the environment. By using our output principle, we can artificially introduce reverberation to make our sound appear to be playing in a different environment than the one in which it was recorded. For example, in the following task you will make a ringtone recorded in a recording studio (anechoic chamber) sound as if it were playing in a church balcony in Nashville.

Task 5: Reverberation

Let's import a recording of an impulse sound signal reverberated from a church balcony[a]. This will be the impulse response of the church modeled as an LTI system.

```
[h, Fh] = audioread("1st_baptist_nashville_balcony.wav");
disp(size(h))
```

Notice (by observing the size) that this audio has two channels (left and right). It is a stereophonic sound. Let's play this audio

```
soundsc(h,Fh) % Play the sound
pause(length(h)/Fh) % Wait for audio to finish playing
```

Let's import the ringtone to which we want to apply the same reverberation effect. Notice that this file has just one channel. It is a monophonic recording. Let's play it.

```
[x, Fx] = audioread("ringtone.wav"); % Read input
disp(size(x))
soundsc(x,Fx) % Play the sound
pause(length(x)/Fx) % Wait for audio to finish playing

h1 = h(:,1); % Extract the first channel from the impulse response
%In simple words extracting all the rows of first column
h2 = h(:,2); % Extract the second channel from the impulse respose
%ISimilarly extracting all the rows of second column of h
```

- Using `conv` command, compute convolution of `h1` and `h2` separately with `x`, i.e. `y1=h1*x` and `y2=h2*x`. Concatenate the two outputs, `y=[y1,y2]`. Play the resulting sound `y` using `soundsc` and sampling frequency `Fx`.
- Plot the audio signals.
- Apply this effect to your own sound files! You can find impulse responses of more interesting places at `https://www.openair.hosted.york.ac.uk/?page_id=36` and some anechoic recordings at `https://www.openair.hosted.york.ac.uk/?page_id=310`.

[a]The audio file 1st_baptist_nashville_balcony.wav by Adam Townsell is licensed under CC-BY and was obtained from https://openairlib.net/.

Post Lab - Task 6: Write your own convolution function

To make sure that you understand discrete-time convolution, you are now to write your own MATLAB function `w = discrete_convo(u,v)` to compute discrete-time convolution of two input signals `u` and `v`. Your function should not use MATLAB's built-in command `convo`.

Test your function by convolving $h[n] = [1, 2, 1], -1 \leq n \leq 1$ with $x[n] = [1, 2, 3, 4, 5], 0 \leq n \leq 4$. Convolve the same two signals using MATLAB's built-in `conv`. Compare the two outputs by plotting them in the same figure along with the inputs. Set `axis([-2 5 0 8])`.

Hint: The process that has to be followed to compute the convolution sum is as follows. First, the two input signals are plotted on the k-axis. Then, one of the two signals is reflected about the amplitude axis. The convolution output at any time n is computed by shifting the reflected signal along k-axis by the value n, finding the product of overlapping values of the two signals, and computing the sum of product terms, according to the convolution sum

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k].$$

A Continuous-time Convolution using the *conv* command

The computational process followed in demo approximates the analytical way of deriving the convolution between two signals. In MATLAB, the command `conv` allows the direct computation of the convolution between two signals.

There are four rules that have to be applied for successfully computing the convolution between two continuous-time signals.

First rule: The two signals (input and impulse response) should be defined on the same time interval. Say, both signals are defined on the time interval $a \leq t \leq b$, where $t = a$ is the first time instance that at least one of the signals $x(t)$ or $h(t)$ is not zero and $t = b$ is the last time instance that at least one of the two signals is non-zero.

Example:

$$h(t) = \begin{cases} 1 - t & 0 \geq t \geq 1 \\ 0 & \text{elsewhere} \end{cases} \tag{2}$$

$$x(t) = \begin{cases} 1 & 0 \geq t \geq 2 \\ 0 & \text{elsewhere} \end{cases} \tag{3}$$

Making the intervals same by writing $h(t)$ as:

$$h(t) = \begin{cases} 1 - t & 0 \geq t \geq 1 \\ 0 & 1 > t \geq 2 \end{cases} \tag{4}$$

Second rule: When a signal consists of multiple parts, the time intervals in which each part is defined must not overlap.

$$h(t) = \begin{cases} 1-t & 0 \geq t \geq 1 \\ 0 & 1 > t \geq 2 \end{cases} \tag{5}$$

Notice that the equality at t=1 is placed only at the upper part. In MATLAB, it will be defined as:

```
dt=0.1;
t1=0:dt:1;
t2=1+dt : dt : 2;  %time does not overlap it starts from (1+dt = 1.1 second)
h1=1-t1;
h2=zeros(1,length(t2));
h=[h1 h2];
```

Third rule: The output of the `conv` command has to be multiplied with the time step used in the definition of the input and impulse response signals, in order to correctly compute the output of the system. This rule emerges from the fact that the convolution integral is being approximated by discrete sum in MATLAB. It is written as: `y=conv(x,h)*dt`.

The only thing left to do is to plot the output. However, the number of elements of the output vector y is not equal to the number of elements of the vectors `x` or `h`. The precise relationship is: $length(y) = length(x) + length(h) - 1$.

Fourth rule: The output of the system is plotted in the double time interval of the one in which the input and impulse response signals are defined. i.e for the above stated example `ty=0:dt:4`.

Assessment Rubric
Lab 5: Convolution

Name:	**Student ID:**

Marks distribution:

	Tasks	**LR2 Code**	**LR5 Results**	**LR9 Report**	**AR4 Report Submission**	**AR7 Comments**
In-Lab	**Task 1**			/4	/8	
	Task 2			/4		
	Task 3	/8	/4	/8		/2
	Task 4		/8			
Post Lab	**Task 5**	/4	/4			
	Task 6	/8	/4			/4
Total =70		/20	/20	/16	/8	/6
CLO Mapped		**CLO2**	**CLO2**	**CLO2**	**CLO3**	**CLO3**

CLO	**Total Points**	**Points Obtained**
2	56	
3	14	
Total	**70**	

For description of different levels of the mapped rubrics, please refer the provided Lab Evaluation Assessment Rubrics and Affective Domain Assessment Rubrics.

Lab Evaluation Assessment Rubric

#	Assessment Elements	Level 1: Unsatisfactory Points 0-1	Level 2: Developing Points 2	Level 3:Good Points 3	Level 4:Exemplary Points 4
LR2	**Program/Code/ Simulation Model/ Network Model**	Program/code/simulation model/network model does not implement the required functionality and has several errors. The student is not able to utilize even the basic tools of the software.	Program/code/simulation model/network model has some errors and does not produce completely accurate results. Student has limited command on the basic tools of the software.	Program/code/simulation model/network model gives correct output but not efficiently implemented or implemented by computationally complex routine.	Program/code/simulation /network model is efficiently implemented and gives correct output. Student has full command on the basic tools of the software.
LR3	**Troubleshooting**	Unable to identify the fault/minimal effort show in troubleshooting.	Able to identify the fault but unable to remove it.	Able to identify the fault but partially removes it.	Able to identify the fault and takes necessary steps and actions to correct it.
LR4	**Data Collection**	Measurements are incomplete, inaccurate and imprecise. Observations are incomplete or not included. Symbols, units and significant figures are not included.	Measurements are somewhat inaccurate and imprecise. Observations are incomplete or vague. Major errors are there in using symbols, units and significant digits.	Measurements are mostly accurate. Observations are generally complete. Minor errors are present in using symbols, units and significant digits.	Measurements are both accurate and precise. Data collection is systematic. Observations are very thorough and include appropriate symbols, units and significant digits and task completed in due time.
LR5	**Results & Plots**	Figures/ graphs / tables are not developed or are poorly constructed with erroneous results. Titles, captions, units are not mentioned. Data is presented in an obscure manner.	Figures, graphs and tables are drawn but contain errors. Titles, captions, units are not accurate. Data presentation is not too clear.	All figures, graphs, tables are correctly drawn but contain minor errors or some of the details are missing.	Figures / graphs / tables are correctly drawn and appropriate titles/captions and proper units are mentioned. Data presentation is systematic.
LR6	**Calculations**	Formulae used and/or calculations are incorrect. Units are not mentioned with results.	Formulae used are correct but there are few mistakes in calculation which lead to incorrect results.	Formulae used and end results are correct. Complete working steps are not shown. Proper units are not mentioned with results.	Formulae used, end results and all calculations are correct with all the intermediate steps clearly shown. Units are mentioned with results.
LR9	**Report**	All the in-lab tasks are not included in report and / or the report is submitted too late.	Most of the tasks are included in report but are not well explained. All the necessary figures / plots are not included. Report is submitted after due date.	Good summary of most the in-lab tasks is included in report. The work is supported by figures and plots with explanations. The report is submitted timely.	Detailed summary of the in-lab tasks is provided. All tasks are included and explained well. Data is presented clearly including all the necessary figures, plots and tables.
LR11	**Design**	Proposed design is unsubstantiated or does not satisfy most of given constraints. The breakdown of system into features is incomplete and still at lower resolution.	Justification for proposed design is weak, and it only satisfies some constraints. The breakdown of system is missing some features and resolution is not appropriate at some places.	Proposed design is substantiated, preferably through proper analysis, and satisfies most given constraints. The breakdown of system into features seems complete, but resolution is not appropriate at some places.	Proposed design is substantiated, preferably through proper analysis, and satisfies all given constraints. The breakdown of system into features seems complete and at appropriate resolution, given students' level.
AR4	***Report Submission**	Not accepted after 1 week.	Late submission after 2 days and within 1 week.	Late submission after the lab timing and within 2 days of the due date.	Timely submission of the report and in the lab time.
AR7	**Report Content/Code comments**	Most of the questions are not answered / figures are not labelled/ titles are not mentioned / units are not mentioned. No comments are present in the code.	Some of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Few comments are stated in the code.	Majority of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Comments are stated in the code.	All the questions are answered, figures are labelled, titles are mentioned and units are properly mentioned. Proper comments are stated in the code.