University of Washington ECE Department
EE 242 Lab 2 – Convolution

In this lab, we will implement convolution on the computer to find responses of an LTI system to different signals. In particular, we will look at convolving simple discrete-time signals with two different impulse responses. Then we'll look at removing noise from audio signals and detecting edges in images using convolution. This is a 2-week lab. It is recommended to work on the first 3 assignments in the first week and the remaining assignments in the second week.

Lab 2 Turn-in Checklist

- 3 pre-lab exercises, to be completed individually and uploaded to canvas before the first lab
- Lab report (Jupyter notebook), completed and submitted as a team
- Jupyter notebook for individual assignment to be uploaded to canvas

Pre-lab

Read the Lab 2 Background document, then complete the following exercises.

1. Let x[n]=u[n-2]-u[n-9]. Sketch the result of convolving x[n] with each of the following signals:

$$h_1[n] = u[n] - u[n-4] \qquad h_2[n] = \delta[n] - \delta[n-1]$$

2. You need to create a signal on the computer that is 2 seconds long and is based on a sampling rate of 1000 Hz. What python commands would you use to create the time vector?

3. Describe the difference between the numpy.random.rand() and the numpy.random.randn() functions.

Lab Assignments

This lab has 4 team assignments. Each should be given a separate code cell in your Notebook, and each should be associated with a markdown cell with subtitle and discussion. As in Lab 1, your notebook should start with a markdown title and overview cell, which should be followed by an import cell that has the import statements for all assignments. For this assignment, you will need to import: *numpy*, the *wavfile* package from *scipy.io*, *simpleaudio*, *matplotlib.pyplot*, the *ndimage* package from *scipy,* and *color* from *skimage*.

Assignment 1: Convolving Simple Signals

We will start by doing some simple convolutions similar to what you saw in class. Create a new cell in your Lab 2 notebook for Assignment 1. This assignment will have three parts, A-C.

A. Create the following three discrete-time signals, assuming a time range of [0,12]

   x = a box of height 1 starting at time n=2 and ending at n=8

   h1 = a pulse of length 4 & height 1 starting at time 0

   h2 = 1 at n=0, -1 at n=1, and 0 otherwise

B. Use numpy.convolve() function to find y1=x*h1 and y2=x*h2 (where * indicates convolution, not multiplication).

C. Create a time vectors nx, ny1 and ny2 for plotting x, y1 and y2. Use the stem plotting function to plot the signals on a **3x1** subplot, using a y-axis between -2 and 5 and a time axis from 0 to 20. Label and title the graphs. Verify that the signals for (y1) and (y2) match what you expect from your prelab.

Report discussion: The systems corresponding to impulse responses $h_1[n]$ and $h_2[n]$ capture different information about a signal. Comment on what aspects of the input signal correspond to the largest values of $y_1[n]$ and $y_2[n]$.

Assignment 2: Smoothing Signals

In this assignment, we'll implement a moving window smoothing function to show how you can use convolution to remove noise from a signal. We'll use a discrete signal associated with a sampling period, and plot signals as if they were continuous to make it easier to see the effect of smoothing. The base signal is generated randomly, so you can run the cell multiple times to see how the results look for different signals. This assignment will have three parts, A-C.

A. Using the starter code provided, create a **base** time signal and a noisy version of it by adding random noise generated with the numpy.random.randn() function (the standard normal distribution, which is zero mean and unit variance). Plot the original and noisy signals with **2x1** subplots, with the time axis labeled assuming a sampling rate of 1000 Hz. Constrain the y-axis to be [0,25] for all plots.

B. Create a smoothed version of the signal called **filtsig1** by computing the average value over a +/- k samples using the numpy.mean() function and k=20. You will need to make a decision as to how to handle the first and last k samples, for which there won't be a full k samples available in both directions. In a single plot, plot the noisy signal and the filtered signal overlaid on the original signal.

C. Define a vector **hfilt** that corresponds to box of length N=2k+1 and height 1/N. Create a second smoothed version of the signal called **filtsig2** by convolving the base signal with **hfilt** using the numpy.convolve() function. Plot the two different filtered signal outputs overlaid on each other. Note that the convolve function will change the length, so you will need to define a new time vector for that.

Report discussion: Describe the differences in the results using the two methods and explain these differences in terms of system properties. Comment on how the results and plots change when you amplify the noise more and also change the value of k.

Assignment 3: Removing Noise from an Audio Signal

In this assignment, we'll artificially add noise to an audio signal and then apply the system you used in Assignment 2 to remove it. You will need the audio packages that you used in Lab 1. This assignment will have three parts, A-C.

A. Read in the trombone sound file provided and name it **tr_orig**. This is a mono signal, so we will only need to worry about the single channel. Create a noise sequence that is the length of **tr_orig** using the numpy.random.randn() noise generation function with a scaling factor of 100. (It needs to be larger to be audible given the range of **tr_orig.**) Then add the two signals to create **tr_noisy**. Save **tr_noisy** to a new wav file.

B. Apply the convolution filter from Assignment 2 to remove the noise from **tr_noisy,** creating a signal called **tr_filt.** Save this signal as a wav file.

C. Read in both the noisy and filtered versions using simpleaudio and play the two files and the original to hear the effect of the noise and noise removal.

Report discussion: Comment on the differences in how the original and noise-removed signals sound. Comment on the impact of large increases or decreases in the value of k.

Assignment 4: Convolution with Images

This assignment is meant to introduce you to convolution of signals in higher dimensions. A canonical example of a 2D signal is an image. With a time signal, the impulse response has a single independent variable (time), so a discrete-time convolution has a sum over that variable. With a 2-D image, the impulse response is 2-D, so the convolution involves a double sum. Be sure to import the *scipy ndimage* and *skimage color* packages. This assignment will have four parts, A-D. Put part A (a function you will define) in its own cell and the other parts together in a

separate cell. Be sure to provide a title for all the images you are plotting in each part.

A. Write a function that takes an image as input and runs the Sobel edge detector (described in the background document) and returns 3 images: the horizontal edges, the vertical edges and the combined result.

B. Start a new cell and read in the image of a dragonfly provided, and convert it to grayscale, as follows:

```
from skimage import color

# Load the image and convert to grayscale
image = color.rgb2gray(plt.imread('dragonfly.jpg'))
```

Note that the variable ‘image’ will have normalized 8-bit (0 - 1) pixel brightness values.

Run the edge detector function on the dragonfly image and display the original with the 3 outputs in a 2 by 2 figure.

C. Define a 10x10 kernel where all elements have value 0.01. This impulse response is giving an average over the 10x10 window, and it should have a smoothing effect on the image, similar to the moving window for the time signal. Convolve the image with this filter using **ndimage.convolve**(). Plot the original and the smoothed image side by side. You should notice that smoothing blurs the image a little.

D. Run the edge detector function on the smoothed dragonfly image and plot the result side-by-side with the result from the edge detector on the original image.

Report discussion: Describe the differences in the results using the edge detector on the original vs. smoothed image. Comment on how the results change if you use a larger size smoothing filter.

Team Report

When you've tested and cleaned up all your code (remember, you should only submit code for the Assignments, each in their own cell), go to 'File' then 'Download as', then select '.ipynb'. The file you download is a Notebook that your TA will be able to open and grade for you, once you submit it on Canvas. Remember, only one notebook per team! Make sure that your notebook is titled Lab2-XYZ.ipynb, where XYZ are the initials of the lab partners.

Individual Assignment:

In this assignment, take a picture of two things with your phone (or a friend's phone), where one has **sharp edges** that you'd expect to detect and the other does not. Download the pictures to your lab directory and convert them to grayscale. Run the edge detection function on the two images and comment on the results. If you have a phone with a fancy camera, you may want to downsample the image to reduce the computational costs.