University of Washington ECE Department
EE 235 Lab 4 – Convolution (II)

In the lab we will continue our study of convolution to learn how it can be used to measure the similarity between two signals, which is useful in decoding transmitted messages in digital communications. This lab will also be used to review functions and formally introduce loops and decision statements.

Lab 4 Turn-in Checklist

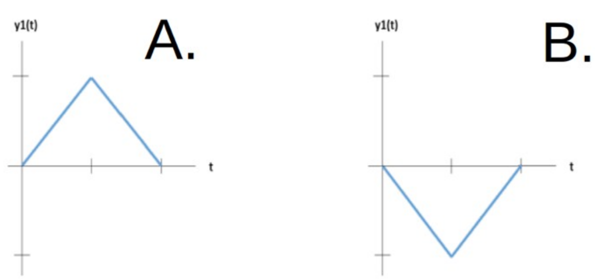
- 3 pre-lab exercises
- 2 lab assignment check-offs with TA
- Lab report, submitted as a Jupyter notebook following the format provided in earlier labs

Note: All assignments except the prelab should be completed in groups of 2-3 people.
The pre-lab exercises are to be completed individually and submitted via Canvas before lab section.

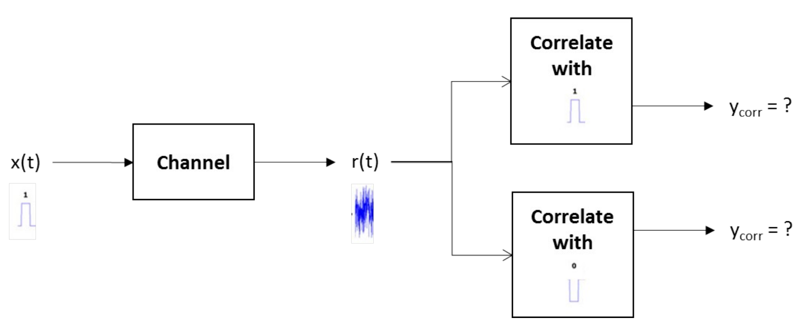
Pre-lab

Read the Lab 4 Background document, then complete the following exercises.

Let **s1(t) = u(t) – u(t – 1)** and **s0(t) = -s1(t)**. Which of the plots below is the correct plot for **y1(t) = s1(t) ∗ s1(t)**? Find the peak value and critical time points for **y1(t),** including start time, peak time and end time. Repeat for **y0(t) = s1(t) ∗ s0(t)**.



In digital communications, we can use can convolve a received signal **r(t)** separately with **s1(t)** and then with **s0(t)** to determine which signal was transmitted.



After convolving the received signal with a signal template, you can find the correlation by extracting the signal value at its midpoint. We will refer to the correlation values as **y1_corr** and **y0_corr,** for the correlation with s1 and s0, respectively. Write a Python decision statement (if/else) to meet the following conditions: Let **s** be 0, unless the variable **y1_corr** is greater than the variable **y0_corr** – in that case, assign the value 1 to **s**.

Write a function **decode** for deciding which of two signals was sent. Your function should take the following inputs: a signal **r**, the sampling rate **fs**, the 0-signal **s0**, and the 1-signal **s1**. It should return the decoded symbol, which is either 0 or 1. In the function, you implement the correlation using convolution (don't forget to apply the amplitude scaling factor) and then extracting the peak value at the midpoint of the resulting signals. The decoded symbol will be decided based on the correlation measurements **y0_corr** (between **r** and **s0**) and **y1_corr** (between **r** and **s1**). If **y1_corr** exceeds **y0_corr**, return 1. Otherwise, return 0. For this lab, you can return 0/1 either as a number or as a string, but it is a little more convenient to use strings.

Lab Assignments

This lab has 3 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 previous labs, 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. In this lab, you will need to use most of the import commands that you have used in earlier labs, plus "import csv". As always, you should add comments to your code for clariy.

You will again be working with concepts from previous labs, so you may want to refer back to the background files for those labs or the py_ref document.

Assignment 1: Matched Filter with Ideal Signals

In this Assignment, we'll use a received signal that has no noise and convolve it with itself and with the template for a different signal. Our signals involve unit pulses, and they will serve as the basis for the digital communications problem we will explore later in this lab. Write your code in a new cell for Assignment 1, inside your Lab 4 notebook.

A. Implement the signals **s1(t) = u(t) – u(t-1)** and **s0(t) = -s1(t)** with **fs=1000 and 0 < t < 1**.

B. Use *numpy**.convolve()*** to compute the convolution **y1(t) = s1(t) ∗ s1(t)** and **y0(t) = s1(t) ∗ s0(t)**. Don't forget to apply the amplitude scaling factor in implementing convolution.

C. Compute the time samples vector **t_y** for **y1** and **y0** (they should be the same length). On a **2x1** subplot, plot **y1 vs. t_y** and **y0 vs. t_y**. Title and label each subplot. Verify that the plots match your pre-lab calculations.

D. To get the correlation measurement **y1_corr**, extract the value of **y1** at the index which corresponds to when time **t** equals the width of input **s1** in seconds, as in pre-lab. Print **y1_corr**. Repeat to get **y2_corr.** Compare your correlation measurements to the peak values you found in pre-lab problem 1. (Note: We are using discrete-time signals while pre-lab used continuous-time ones, so correlation measurements may not exactly match.)

Assignment Check-Off #1 of 3: Demonstrate this Assignment to the lab TA

Report discussion: In this Assignment, we assumed signals s1(t) and s0(t) were both box signals, but with opposite (+/-) amplitudes. Suppose instead that s0(t)=v(t)-v(t-0.5), where v(t)=u(t)-u(t-0.5). Using your understanding of correlation, do you expect the correlation measurement between s1(t) and s0(t) to be greater or less than the correlation measurement was when both signals were box signals? Which s0(t) would be more useful for communications?

Assignment 2: Signal Decoding with a Matched Filter

As always, start a new cell in your Jupyter notebook for this assignment. In this assignment, you will use a random noise generator. Note that each time you call it, the function will use a different random seed, so if you run the cell multiple times without changing your code, you will get different results for the plots. Don't forget to title and label all plots.

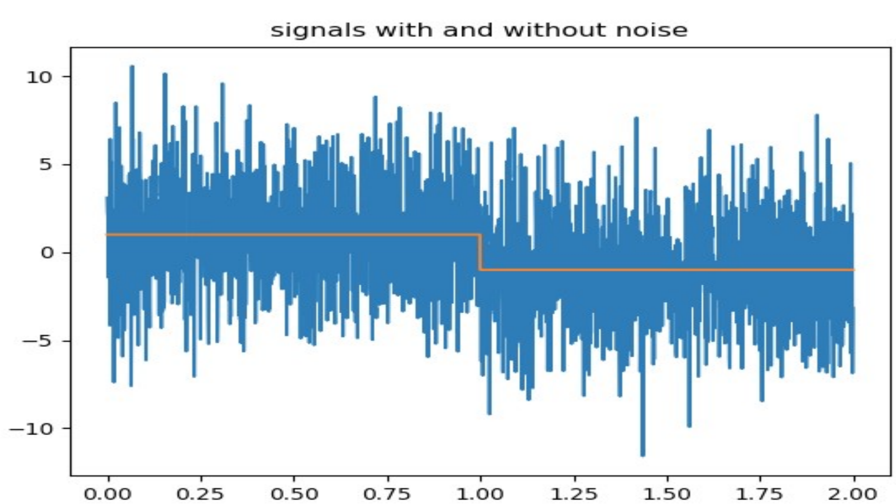
A. Using a separate cell, implement the **decode** function that you wrote in the pre-lab.

B. Using the definitions of **fs**, and the **s1** and **s0** signals from **Assignment 1**, create a received signal **r(t)** by adding signal **s1(t)** with a random noise signal **n(t)**.

   a. We create a noise vector of the same length as **s1**, and standard deviation 10 with:
   `n = np.random.normal(0,3,len(s1))`

   b. Create received signal vector **r** using **r = s1 + n**, a second received signal vector **r0 = s0 + n**.

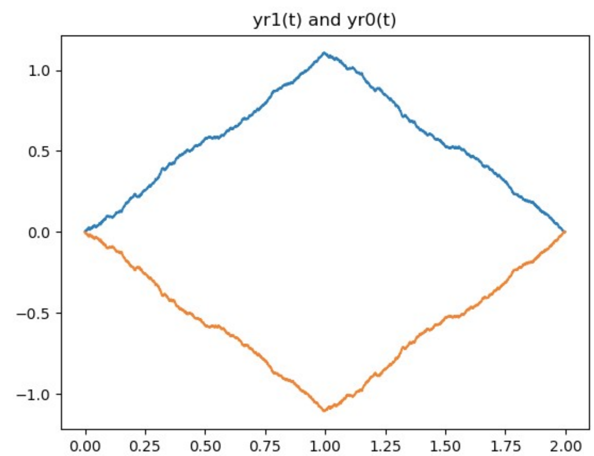
   c. Concatenate the two signals s1 and s0 in sequence and repeat for r and r0. Create a time samples vector and plot them in one plot with the clean signal overlaying the noisy version. The plot should look roughly like:



C. Now compute the correlation measurements between **r** and **s1,** and **r** and **s0**:

   a. Compute **yr1 = r ∗ s1**, and **yr0 = r ∗ s0**. Create **t_yr,** the time samples vector associated with **yr1** and **yr0**.

   b. In Plot 2, plot **yr1 vs. t_yr** and **yr0 vs. t_yr** in different colors. Provide a legend. The plot should look roughly as follows.



   c. Compute **yr1_corr** as the correlation between **r** and **s1**, by sampling **yr1** at the correct time. Print **yr1_corr.** Repeat for **y0_corr**, the correlation between **r** and **x0**, by sampling **yr0**.

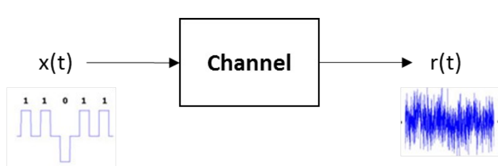
D. Test **decode**, by decoding the 1-signal **s1** and the 0-signal **s0**. If the function works as expected, passing the input signal **r=s1+n** should produce the symbol **1**, while passing **r=s0+n** should produce the symbol **0**.

Assignment Check-Off #2 of 3: Demonstrate this Assignment to the lab TA

Report discussion: In a real communication system, having shorter signals means that you can communicate information faster. How would the correlation functions yr1 and yr0 change if the signal was only 20 ms long?

Assignment 3: Decipher Received Message

In this exercise, we apply what we learned in previous exercises to a digital communication system. A primary goal of a digital communication system is to decipher received signals that have been distorted with noise during transmission. As we’ve seen, we can use correlation measurements to determine whether the received signal is 0 or 1 at a given time.



You will decipher a secret message by decoding a bit sequence from the received message **r(t),** and translating the bits to ASCII characters.

Start a new cell for Assignment 3. You will use the **decode** function from the previous assignment. Since the message was created with a higher noise level and sampling rate, you will need to rebuild **s0** and **s1** using **fs=8000.**

A. The received message **r(t)** has been reshaped as an array with N indices, one for each binary signal and saved as a CSV file. Download the file 'receivedmsg.csv' from the class web page. Using the guidelines in the Lab 4 background document, read in the CSV file containing the points from the received message and call it **rm**. Each row of the CSV file corresponds to one of the message's bits worth of data (a 1 second time slice), transmitted and received as **ri(t)**, i=1, ..., N.

B. To decode the message, you will need to loop over each row, and use **decode** to determine if the data for the given time slice constitutes a 1 or 0 signal, then save the appropriate value. If your **decode** function produces a binary output, you can store it in an array. If it produces a string, then you can just concatenate each new bit to the string of previous results.

C. Print the sequence of bits.

Report discussion: Provide the decoded message in your lab report. You can copy the result and use this link: http://www.binaryhexconverter.com/binary-to-ascii-text-converter to translate the bits to ASCII symbols.

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 Lab4-XYZ.ipynb, where XYZ are the initials of the lab partners. You may want to also download the file as pdf to have a nicer documentation of your records.

Submit the .ipynb file via Canvas.