University of Washington
Department of Electrical Engineering
EE 235 Lab 4
Convolution for Digital Communications

In this lab, we will investigate convolution, starting with using unit impulses and convolution to implement time delay, and then learn how convolution can be used to decode transmitted messages in digital communications. This lab will also be used to review functions and introduce students to loops and decision functions.

<u>Important Concepts From Lecture You Will Use In Lab 4</u>
- Describing LTI systems using the impulse response **h(t)**
- Computing the output **y(t)** of an LTI system using the graphical method of convolution

<u>What is expected from you in Lab 4</u>
- Completion of 3 pre-lab exercises (3 points total)
- Completion of 2 in-lab check offs with TA (4 points)
- Completion of a lab report (3 points)

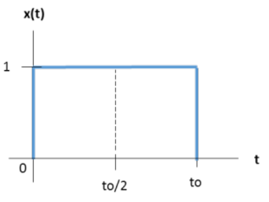
Note: all pre-lab exercises must be completed by each student <u>individually</u> before walking in to the lab. Each student will be checked off by the TA at the beginning of lab section.

<u>PRE-LAB:</u>
Read the Matlab Lab 4 tutorial, particularly the sections on scaling the convolution output, for loops and decision functions.

In this lab, we will see how we can use convolution to measure the similarity between two signals. This important measurement is called correlation. The correlation between two signals can be computed from the convolution between signal x1(t) and signal h(t), and sampling the convolution output at a particular time point. Recall the convolution formula:

$$y(t) = x1(t) * h(t) = \int_{-\infty}^{\infty} x1(\tau)h(t-\tau)d\tau$$

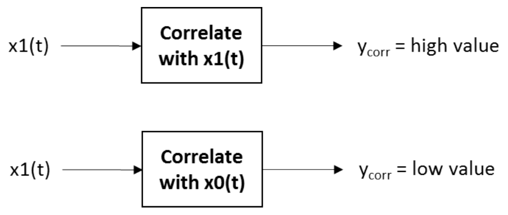
For this lab, we will consider the case where the signals are both of the same duration and perfectly symmetric around their midpoint in time:



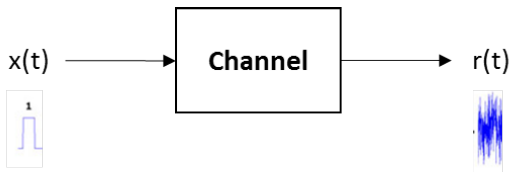
In this case, the correlation is computed from the convolution by sampling the y(t) at time t when there is full overlap between x1(τ) and h(t - τ):

$$y_p = y(t)\big|_{t=full\ overlap\ case}$$

The correlation measurement will be maximum when the two signals are identical, h(t)=x1(t), so this system is referred to as a *matched filter*. If the two signals are different, the correlation measurement will be lower, and it can be negative, as when h(t)=-x1(t).

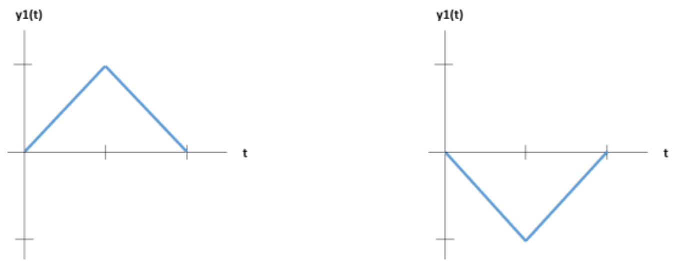


You can use this in digital communications, where messages consisting of binary values 0 and 1 are sent by using a signal x(t) that can take two different forms (x0(t) or x1(t)) depending on which value is being sent. After the signal is transmitted over the air, the signal received r(t) will have added noise, so it will not be perfectly matched.

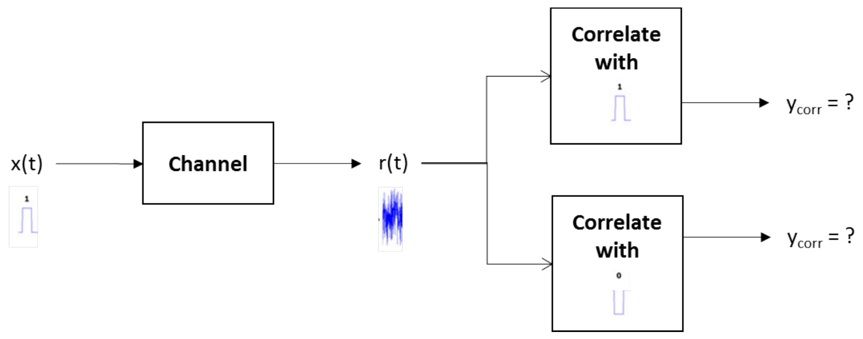


If the signals are chosen to be sufficiently different and the noise is not too high, the correlation measurement will still be higher for the matching signal. In this lab we will use the signals **x1(t) = u(t) – u(t – 1)** and **x0(t) = -x1(t)** to represent 1 and 0, respectively.

1) Of the two plots below, which one is the correct plot for **y1(t) = x1(t) ∗ x1(t)**? Using the graphical method of convolution, find the peak amplitude and label critical time points for **y1(t)**. Repeat for **y0(t) = x1(t) ∗ x0(t)**. At what time point would you take the correlation measurement?



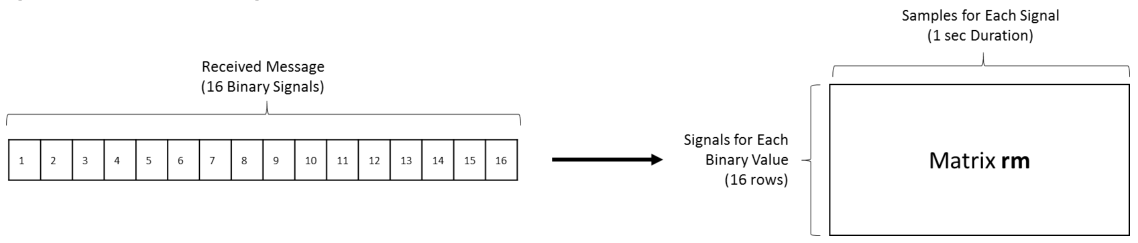
2) In digital communications, we can use can correlate **r(t)** separately with **x1(t)** and then with **x0(t)** to decode which signal was transmitted.



a) Write the header for the following desired function: a function called **decode** that takes in the input signal **x**, the width **d** of signal x, the sampling rate **Fs**, the low signal **x0**, and the high signal **x1**, and then outputs the decoded symbol to the variable **s**.

b) Write the Matlab decision statement for the following scenario:
   - If the variable **y1_corr** is greater than the variable **y0_corr**, assign the value 1 to **s**
   - Otherwise, assign the value 0 to **s**

3) A useful digital message typically has multiple bits of information. In this lab, you will decode a message with 16 bits, where the received signal has been reshaped in to a matrix **rm** to separate the binary signals. Each row of **rm** contains one of the 16 binary high or low signals in the message:



You will need to extract each binary signal from the matrix and perform a separate correlation to decipher each bit. We will employ a for-loop to aid us in this task.

a) What is the line of code to get the number of rows in matrix **rm**, the value of which will be stored in the variable **rm_rows**?

b) What is the line of code to create a row vector of zeros called **message_bits** with the same number of elements as **rm_rows**?

c) Using **rm_rows** and the loop index variable **i**, write the for-loop statement to loop through all the rows in matrix **rm**, starting from i = 1 to the last row of matrix **rm**. In other words, fill in the following line of code:

   for FILL IN CODE

d) Assuming your loop variable is **i**, what is the line of code to extract all the columns in the **ith** row of matrix **rm**, the values of which will be stored in the variable **signal**?

EXERCISE #1: Convolution of Two Signals

In this experiment, we will investigate what happens when you convolve a signal with itself and also with a different signal. The signals we will use involve unit step functions and serve as the basis for the digital communications problem we will explore later in this lab.

Lab Exercise: Let us now perform these convolutions you analyzed using Matlab.

a) Create a new working directory in your U Drive called **Lab 4**. In this directory, open up a script file and call it **Ex1.m**. Make sure to clear all variables and close all figures.

b) We will first implement the high and low signals **x1(t) = u(t) – u(t-1)** and **x0(t) = -x1(t)**:

   i) Using the **ones** function, create a vector of ones called **x1** that lasts for 1 second with a sampling rate of **Fs** = 8000.

   ii) Using the vector **x1**, create the vector **x0**.

c) Using the **conv** function, compute the convolution of the two high signals **y1(t) = x1(t) ∗ x1(t)**, as well as the convolution between the high and the low signal **y0(t) = x1(t) ∗ x0(t)**. Store your results in vectors **y1** and **y0**, respectively. For each convolution, make sure to apply the correct amplitude scaling factor.

d) Using **y1**, compute the corresponding time samples vector **t_y**. Since **y1** and **y0** are the same length, we will use **t_y** for both vectors.

e) Using a 2 x 1 subplot, plot the following:
   - **y1 vs. t_y**
   - **y0 vs. t_y**

   You do not need to adjust the axes, but title and label each subplot.

f) Comment your script, and then run it. Verify that your plots and time boundaries match your answers from pre-lab. Also, verify your answer as to whether convolving a signal with itself produces the same plot as the convolution of two different signals.

Lab Check-Off #1 of 2: Show your script **Ex1.m** to a lab TA and demonstrate you know how to perform convolution in Matlab.

Lab Report Question #1 of 3: An inquisitive student decides to perform the convolution for **y0(t)** in reverse order and computes: **y0(t) = x0(t) ∗ x1(t)**. The student claims s/he produced a completely different plot. Explain why this CANNOT be true.

EXERCISE #2: Convolution and Matched Filtering

Lab Exercise:

a) Open up a script file and call it **Ex2.m**. Clear all variables and close all figures.

b) Your communication high and low signals are stored in a Matlab data file **CommsSignals.mat**. Download the file from the class website, and then load the file. The file will contain the following three variables: **x1**, **x0**, and **Fs**.

c) We will first compute the correlation between identical signals **x1(t)** and **x1(t)**:

   i) Compute the convolution **y1(t) = x1(t) $*$ x1(t)** and store the result in vector **y1**. Make sure to scale the output of the convolution appropriately.

   ii) To get the correlation measurement, extract the value of the convolution output **y1** at the index that corresponds to correlation measurement time that you found in the prelab. Store the correlation measurement in the variable **y1_corr**. Display the value of **y1_corr** on the COMMAND window.

d) Repeat the steps in part (c) for the correlation between two different signals **x1(t)** and **x0(t)**, storing the resulting signal in vector **y0** and the correlation measurement in variable **y0_corr**.

e) Comment your script, and then run your script to view your correlation measurements on the COMMAND window. Verify that your measurements are close to your answers from pre-lab. Note: we are using discrete-time signals in Matlab, so the correlation measurements will not be the same as your pre-lab answers (where continuous-time signals were analyzed).

f) Let us now analyze the digital communication scenario from pre-lab. We will create **r(t)** by adding the high signal **x1(t)** with a random noise signal **n(t)**:

   i) Download **Noise.mat** from the class website and load the file. The file will contain two variables: **n** and **Fs**. The sampling rate **Fs** is identical to **x1** and **x0**.

   ii) Using the high signal **x1** and the noise signal **n**, produce the received signal **r(t) = x1(t) + n(t)**. Store the result in variable **r**.

   iii) Create the corresponding time samples vector **t_r**.

   iv) On a 2 x 1 figure window, plot **r vs. t_r** as the 1st subplot. Adjust your x-axis to be between 0 and 2. Leave y-axis unchanged. Title and label your plot

g) Let us now compute the correlation measurement between **r** and **x1**, as well as **r** and **x0**:

   i) Compute the convolution between **r** and **x1**, and store the output in **yr1**. Make sure to scale the output of the convolution.

   ii) Compute the convolution between **r** and **x0**, and store the output in **yr0**. Make sure to scale the output of the convolution.

   iii) Using variable **yr1**, compute the time samples vector **t_yr**. We will use this vector of time samples for **yr0** as well.

iv) On the 2nd subplot, plot both **yr1 vs. t_yr** (using the color magenta) and **yr0 vs. t_yr** (using the color red) on the same plot. No need to adjust your axes. Title and label your plots. Also, provide a legend to label each signal.

v) Compute the correlation **yr1_corr** between **r** and **x1** by sampling **yr1** at the correct time. Display the output of **yr1_corr** on the COMMAND window.

vi) Compute the correlation **yr0_corr** between **r** and **x0** by sampling **yr0**. Display the output of **yr0_corr** on the COMMAND window.

h) Write a function to decode a received signal.

i) Open a new script and call it **<u>decode.m</u>**. As the first line, write your function header.

ii) Using your code above, compute the correlation between an input signal **x** and both the signals **x0** and **x1**, and sample to get the correlation measurement.

iii) Using your decision statement from pre-lab, write the if-else statement to determine the output decoded symbol **s** from the correlation measurements **y0_corr** and **y1_corr**.

i) Let us now test your **decode** function on a separate Matlab script. Open a new script and call it **<u>TestDecode.m</u>**. Make sure to clear all variables and close all figures.

i) Load **<u>CommsSignals.mat</u>** to get access to the variables: **x0**, **x1**, and **Fs**. The signals **x0** and **x1** will again have the same duration of 1 second.

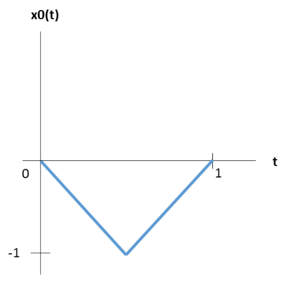
ii) Decode the signal **x1** by calling the function **decode** and passing **x1** as the input signal, a duration of 1 second, and the low and the high signals **x0** and **x1**. Store the output in the variable **s1**. Display the output of **s1** on the COMMAND window.

iii) Repeat (ii) for input signal **x0** and store the output in the variable **s0**. Display the output of **s0** on the COMMAND window.

iv) Run your script, and verify you get the correct values for **s1** and **s0**. If your function works as expected, passing the input signal **x1** should produce the symbol 1, while passing the signal **x0** should produce the symbol 0.

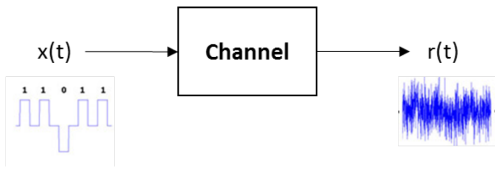
<u>Lab Check-Off #2 of 2</u>: Show your plots to a lab TA and demonstrate you know how to compute correlation measurements in Matlab.

<u>Lab Report Question #2 of 3</u>: In this exercise, we assumed signals **x1(t)** and **x0(t)** were both box signals but with different amplitudes. Suppose **x0(t)** had a different shape, say a triangle like the one below:



Using your understanding of correlation, do you expect the correlation measurement now between **x1(t)** and **x0(t)** to be greater or less than the correlation measurement for the case when both signals were box signals? Explain your answer.

<u>**EXERCISE #3: Deciphering Received Message in Digital Communication System**</u>
In this exercise, we will take what you learned in the previous exercises and apply it to a digital communication system application. As shown already, we can use correlation measurements to determine whether each part of a received signal is a 0 or 1.



For this exercise, you will decipher an entire message consisting of 16 bits, where each bit is represented as the same high and low signals we have already been using, **x1(t)** and **x0(t)**, both of duration 1 second for a total duration of 16 seconds for the whole message.

<u>***Lab Exercise***</u>:

a) Open up a new script and call it <u>**Ex3.m**</u>. Clear all variables and close all figures.

b) Load the data file <u>**CommsSignals.mat**</u>. This will again contain: **x0**, **x1**, and **Fs**.

c) Download <u>**L4Ex3signal.mat**</u> from the class website and load it. This file will contain:
   - **r**: This is the entire received messaged vector
   - **rm**: This is the formatted received message as a matrix
   - **Fs**: Sampling rate of received signal (same as x0 and x1)

d) Compute the time samples vector **t_r** for the received signal vector **r**.

e) To see what kind of signal you will be working with, plot **r** vs. **t_r**. No need to adjust the axes, but title and label your plot.

f) Get the number of rows in matrix **rm** and store that value in variable **rm_rows**.

g) Create a <u>row vector</u> of zeros called **message_bits** with the dimensions 1 x **rm_rows**. You will use this vector to store every bit you decipher.

h) We will now decipher each bit in the received message. Here is the outline of the loop:

```
% LOOP through all rows in matrix rm using variable index i

    % Extract all the columns in the ith row of matrix rm and store
    % the values in the variable signal

    % Call the decode function to decipher the binary value of signal and
    % store the output in the variable symbol

    % Store symbol in the ith element of message_bits

% END of loop
```

i) Display the values of **message_bits** on the COMMAND window.

j) Comment your script, and then run it. Make sure the values of **message_bits** are displayed on the COMMAND window.

2) ***Lab Report Question #3 of 3***: Provide the decoded message in your lab report. What is the hidden message? You can use this link: http://www.binaryhexconverter.com/binary-to-ascii-text-converter to translate the bits to ASCII symbols.

FOR NEXT WEEK: LAB REPORT DUE

Turn in a PDF of your lab report (one per team) online via the link on the class website. A sample format for the lab report is on the class website. You may vary from this format but please include the basic sections. The report is due 24 hours prior to the start of your next lab section. Lab turn-in times will be checked, so no late lab reports will be accepted unless arranged in advance with the instructor.