University of Washington
Department of Electrical Engineering

EE 235 Lab 4

Convolution

In this lab, we will expand on what we did in the last lab on convolution. We will 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 statements.

<u>Important Concepts From Lecture You Will Use In Lab 4</u>

- ☐ Using and graphing unit step functions
- ☐ Performing graphical method of convolution

<u>What Is Expected From You In Lab 4</u>

- ☐ Completion of 4 pre-lab exercise (5 points)
- ☐ Completion of 2 in-lab check offs with TA (5 points)
- ☐ Completion of a lab report (10 points)

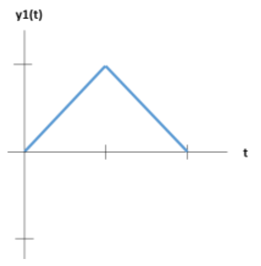
<u>EXERCISE #1: Convolution of Two Signals</u>

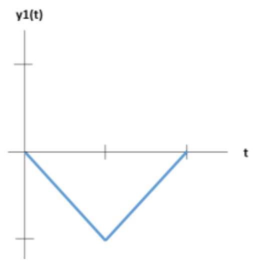
In this experiment, we will investigate what happens when you convolve a signal with itself and also with a different signal. The signals we will use involve unit step functions and serve as the basis for the digital communications problem we will explore later in this lab.

1) ***<u>Pre-Lab Questions</u>***: Digital communications involves sending messages consisting of binary values 0 and 1. To represent the high signal 1, we will use the signal **x1(t) = u(t) – u(t - 1)**. To represent the low signal 0, we will use the signal **x0(t) = -x1(t)**.

   a) Graph the signals **x1(t)** and **x0(t)** on separate plots over the range $0 \le t \le 1$.

   b) 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 corresponding time boundaries for **y1(t)**.





   c) Repeat (b) for **y0(t) = x1(t) ∗ x0(t)**. Which of the two plots is correct? Is it the same as the plot for **y1(t)**? What are the time boundaries?

   d) **<u>Read the Background Section</u>** on these topics to prep for the upcoming lab exercise:

Matlab Concepts/Functions to Review	New Matlab Concepts/Functions
• Creating a vector that lasts *t* seconds • Creating time samples vector • Plotting using subplots • Adding labels and a title to a plot	• Using the ones function • Scaling output of convolution

2) ***Lab Exercise***: Let us now perform these convolutions you analyzed using Matlab.
   a) Create a new working directory 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. **Refer to Background Section (3)**.

   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 all 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.

3) ***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.

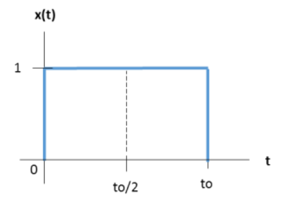
4) ***Lab Report Question #1 of 3***: An inquisitive student decides to perform the convolution for **y0(t)** in reverse order and computes: **y0(t) = x0(t) ∗ x1(t)**. The student claims s/he produced a completely different plot. Explain why this CANNOT be true.

EXERCISE #2: Convolution and Correlation

In this exercise, 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 x2(t), and sampling the convolution output at one time. Recall the convolution formula:

$$y(t) = x1(t) * x2(t) = \int_{-\infty}^{\infty} x1(\tau)x2(t-\tau)d\tau$$

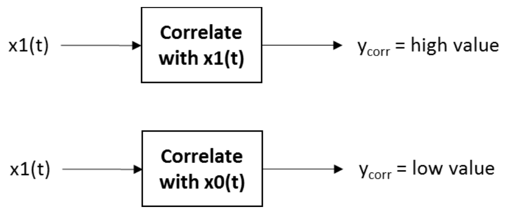
For this lab, we will focus on the case where the signals are both of the same duration and perfectly symmetric around their midpoint in time:



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 x2(t - τ):

$$y_{corr} = y(t)|_{t=full\ overlap\ case}$$

If the two signals are identical or very similar, the correlation measurement will be very high. If the two signals are different, the correlation measurement will be very low. You will verify this result in the upcoming pre-lab and lab exercise.



1) ***Pre-Lab Questions***:
   a) Consider our high signal **x1(t) = u(t) – u(t - 1)** and our low signal **x0(t) = -x1(t)**. What is the width (in sec) of both signals – that is, for how long (in sec) are both signals nonzero?

   b) Consider the convolution of two identical signals **y1(t) = x1(t) ∗ x1(t).** Using your pre-lab results from Exercise #1:

      i) At what time **t** in **y1(t)** is there full overlap between x1(τ) and x1(t - τ) in the convolution? Show that the time **t** when the full overlap case occurs is equal to the width of **x1(t)**.

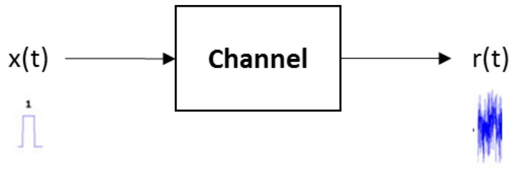
      ii) What is the corresponding correlation measurement between **x1(t)** and **x1(t)**? That, is what is the value of **y1(t)** at time **t** when there is full overlap?

c) Consider the convolution of two different signals **y0(t) = x1(t)** $*$ **x0(t).** Using your pre-lab results from Exercise #1:

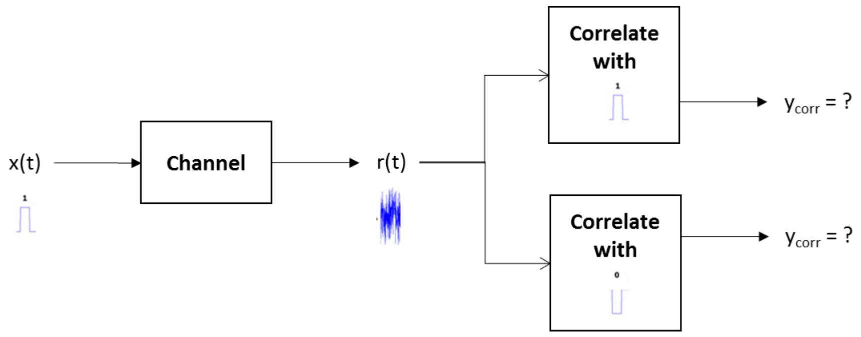
   i) At what time **t** in **y0(t)** is there full overlap between x1(τ) and x0(t - τ) in the convolution? Show that the time **t** when the full overlap case occurs is equal to the width of **x1(t)** (or equivalently, the width of **x0(t)**).

   ii) What is the corresponding correlation measurement **x1(t)** and **x0(t)**? That, is what is the value of **y0(t)** at the time **t** when there is full overlap? Is this measurement greater than or less than correlation between **x1(t)** and itself?

d) In digital communications, we can use correlation to determine which of two signals, high or low, was transmitted even in the presence of noise. Consider the following scenario where a high signal **x(t)** is transmitted over the air and a noisy version of the high signal **r(t)** is received:



You can correlate **r(t)** separately with **x1(t)** and then with **x0(t)** to decode which signal was transmitted.



Using your understanding of correlation, do you expect the correlation between **r(t)** and **x1(t)** to be greater than or less than the correlation between **r(t)** and **x0(t)**?

e) **Read the Background Section** on these topics to prep for the upcoming lab exercise:

Matlab Concepts/Functions to Review	New Matlab Concepts/Functions
• Extracting time sample t in a vector • Changing axes limits • Plotting two graphs on one plot • Changing color of a graph • Adding a legend to a plot • Displaying output on COMMAND window	• Scaling output of convolution

2) ***Lab Exercise***: Let us now use Matlab to verify our pre-lab results

   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 time t equal to the width of input **x1** in sec, similar to what you did in pre-lab. Store the correlation measurement in the variable **y1_corr**. Display the value of **y1_corr** on the COMMAND window. **Refer to Matlab Review Section if needed**.

   d) We will now compute the correlation between two different signals **x1(t)** and **x0(t)**:

      i) Compute the convolution **y0(t) = x1(t)** $*$ **x0(t)** and store the result in vector **y0**. Make sure to scale the output of the convolution appropriately.

      ii) To get the correlation measurement, extract the value of the convolution output **y0** at the index that corresponds to time t equal to the width of input **x0** in sec, similar to what you did in pre-lab. Store the correlation measurement in the variable **y0_corr**. Display the value of **y0_corr** on the COMMAND window.

   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 time samples vector 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. **<u>Refer to Matlab Review section if needed</u>**.

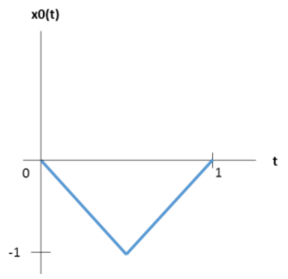
   v) Compute the correlation **yr1_corr** between **r** and **x1** by sampling **yr1** at the correct time. Display the output of **yr1_corr** on the COMMAND window.

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

h) Comment your script, and then run it to view the correlation measurements for the received signal. Verify your answer to pre-lab as to which correlation measurement is greater.

3) ***<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.

4) ***<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: Correlation and Signal Decoding**</u>

In this exercise, we will take what we learned in the last exercise about using correlation to decode a transmitted signal and write our own decode function.

1) ***<u>Pre-Lab Questions</u>***:
   a) Review **<u>Background Section (4)</u>** on functions and then answer this question. 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) Read **<u>Matlab Review Section</u>** on decision statements and then 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**

2) ***<u>Lab Exercise</u>***: We will now write and test our **decode** function.
   a) Open a new script and call it **<u>decode.m</u>**. As the first line, write your function header.

   b) Using the same procedure from Exercise #2, compute the correlation **y0_corr** between input signal **x** and low signal **x0**. Use the time width (in seconds) **d** to sample the output convolution, which you can call **y0**. Make sure to scale the output of the convolution.

   c) Repeat (b) to get the correlation **y1_corr** between input signal **x** and high signal **x1**.

   d) 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**.

   e) Write your function header comments and in-line comments.

   f) Let us now test your function on a separate Matlab script. Open a new script and call it **<u>Ex3.m</u>**. Make sure to clear all variables and close all figures.

   g) We will test the **decode** function you wrote by decoding the high signal **x1** and the low signal **x0**. 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.

      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.

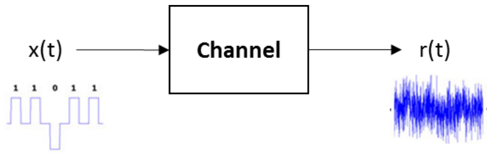
      ii) Decode the signal **x1** by calling the function **decode** and passing **x1** as the input signal, a duration of 1 second, and the low and the high signals **x0** and **x1**. Store the output in the variable **s1**. Display the output of **s1** on the COMMAND window.

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

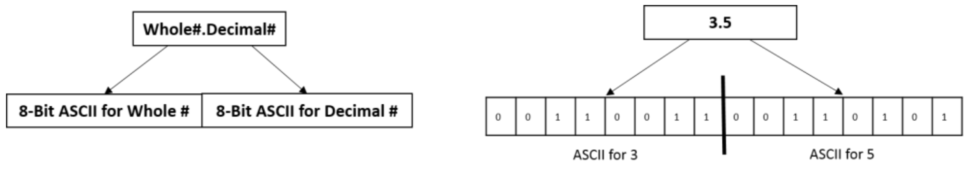
   h) Run your script, and verify you get the correct values for **s1** and **s0**.

EXERCISE #4: Deciphering Received Message in Digital Communication System

In this exercise, we will take what you learned in the previous exercises and apply it to a digital communication system application. One of the goals of a digital communication system is to decipher received signals that have been distorted by noise during transmission. As shown already, we can use correlation measurements to determine whether each part of a received signal is a 0 or 1.

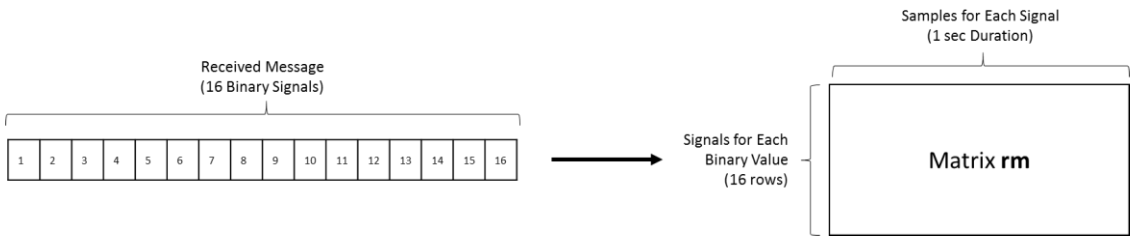


For this exercise, you will decipher an entire message sent from the future. This message consists of 16 bits, where each bit is represented as the same high and low signals we have already been using, **x1(t)** and **x0(t)**, both of duration 1 second for a total duration of 16 seconds for the whole message. In this message from the future, your lab TA has been tasked with sending you your final grade in EE 235. The TA chose to have some fun and separately encode each part of your grade (the whole number and the decimal number) in to a secret binary message using ASCII.



Unfortunately, the secret message had to travel far and wide (from the EE building through the HUB and also the Suzzallo and Allen libraries) to reach your computer, and as a result, was corrupted by random noise in the air. Your mission this week is to decode this secret message and view your final grade in EE 235!

1) ***<u>Pre-Lab Questions</u>***: To simplify the message decoding process, the received message **r(t)** has already been graciously 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:



Because the message consists of 16 binary signals, we 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**? **Refer to Matlab Review Section**.

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**? **Refer to Matlab Review Section**.

c) **Read Background Section (5)** on Matlab for loops and then answer the following question: 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) **Read Background Section (6)** on element extraction and answer the following question: 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**?

e) **Read the Background Section** on these topics to prep for the upcoming lab exercise:

Matlab Concepts/Functions to Review	New Matlab Concepts/Functions
• Getting dimensions of a matrix • Creating a zeros vector • Storing values in a vector • Displaying output on COMMAND winodw	• Writing for loops • Extracting all elements in a vector or matrix

2) ***Lab Exercise***:

a) Open up a new script and call it **Ex4.m**. Clear all variables and close all figures.

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

c) Download **Ex4.mat** 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) Using your pre-lab code, get the number of rows in matrix **rm** and store that value in variable **rm_rows**.

g) Using your pre-lab code, create a row vector 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. **Refer to Matlab Review Section if needed**.

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

3) ***Lab Report Question #3 of 3***: Decipher your grade for EE 235 from the 16-bit message.

NUMBER	ASCII CODE IN BINARY
0	00110000
1	00110001
2	00110010
3	00110011
4	00110100
5	00110101
6	00110110
7	00110111
8	00111000
9	00111001

a) Split the message in to two even parts.
b) Use the first 8 bits and the ASCII table to deduce the whole number of your final grade
c) Use the last 8 bits and the ASCII table to deduce the decimal number of your final grade.
d) What is your final grade?

FOR NEXT WEEK: LAB REPORT DUE

1) Use the lab report format on the class website
2) Turn in a PDF of your lab report (one per team) online via link on class website
3) Remember to include your M-files with your submission
4) Report is due in one week prior to the start of your next lab section

BACKGROUND SECTION

1) **Matlab Review**

Concepts/Functions	**Sample Code**

Extract elements in a vector/matrix	a = y1(1); % Extract one element
	b = y1(1:2); % Extract multiple elements
	b1 = y1(3:end); % Extract until end of vector
	
	c = z(1,2); % Extract one element
	d = z(2, 1:2); % Extract multiple elements

Changing elements or storing values in a vector/matrix	y1(1) = 3; % Change one element y1(1:2) = -1; % Change multiple elements z(1,2) = 0; % Change one element z(2, 1:2) = 2; % Change multiple elements
Dimensions of a vector or matrix	length(y1) % Num elements in vector size(y1) % Dimensions (rows, columns) size(z, 1) % Num of rows size(z, 2) % Num of columns

Creating time samples vector	% Given start and end time
	t = 0:1/Fs:5 % 0 ≤ t ≤ 5 with Fs = 2
	
	% Using vector length
	t_x = (0:length(x)-1) * (1/Fs) % Vector is x

Creating a vector that lasts ***t*** seconds	% Create zeros vector that lasts 3 seconds x = zeros(1, 3*Fs + 1)

Relationship between index and time:
$i = t \times F_S + 1$
	% Extracting t = 5 and storing in y
	index = 5 * Fs + 1;
	y = x(index);
	
	% Accessing t = 5 and changing value to 1
	index = 5 * Fs + 1;
	x(index) = 1;
	
	% Extracting 0 ≤ t ≤ 5 and storing in y
	start_index = 0 * Fs + 1;
	end_index = 5 * Fs + 1;
	y = x(start_index:end_index);

Opening a new figure window	figure;

Using a 2 x 1 subplot and plotting on 1st figure	subplot(2, 1, 1);
	plot(……..);

Plotting a signal x vs. t	plot(t, x);
Turn on grid lines	grid on;
Plotting two graphs on one plot with different colors	plot(t, x1, ‘r’); % Use color red hold on; plot(t, x2, ‘g’); % Use color green

Color Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Adding a legend to a plot	legend(‘label1’, ‘label2’, ‘label3’);

Changing axes limits	xlim([0 10]);
	ylim([-5 5]);

Labeling axis and adding plot title	xlabel(‘Time’);
	ylabel(‘x(t)’);
	title(‘Signal x(t)’);

Loading and play sound file	load file.mat; % Contains variables y and Fs sound(y, Fs);
Display to COMMAND window	% Display output of calculation x = x + 1 % Omit semicolon % Display contents of matrix A A % Omit semicolon

if-else Decision Statements	if x == -2
	  % Code
	elseif x > 2 && x < 5
	  % Code
	else
	  % Code
	end

2) **New Matlab Functions for Lab 4**

- **Function *ones***: We will use the function ***ones*** to create a unit step function.

Syntax: x = ones(1, 10)
Description: Creates a vector or matrix of ones
Usage in Lab 4: x = ones(1, 3*Fs +1);   % Create a row vector of ones that lasts 3 seconds

3) **Scaling Convolution Output from Matlab for Continuous-Time Signals**

- Since Matlab only deals with digitized representations of a signal, the built-in convolution function in Matlab assumes that the period between time samples is 1. For our continuous-time signals, however, the period between samples is 1/Fs.

- This discrepancy affects the convolution computation. In particular, the height of the output is incorrect and must be scaled appropriately. You will learn more about the difference between discrete-time convolution and continuous-time convolution in EE 341.

- To correct the convolution for continuous-time signals, you must scale the convolution output by the period 1/Fs:
  y = (1/Fs) * conv(x, h)

4) **Review of Matlab Functions**
- Function header declaration:

	Example Declaration	**Example Function Call**
One output, one input	*function y = myexample(x)*	*A = myexample(B)*
More than one output: Must enclose output in square brackets	*function [y1, y2] = myexample(x)*	*[A, B] = myexample(C)*
More than one output, more than one input	*function [y1, y2] = myexample(x1, x2)*	*[M, N] = myexample(C, D)*

- The name of a function file must be the same as the name of a function. As an example, if you write a function called **addme**, the M-file must have the name **addme.m**.

- Function files should include a function header and in-line comments. A function header in Matlab is placed above the function declaration. Example **addme** function:

```
% ADDME  Add two values together.
%  USAGE: C = ADDME(A,B) adds A and B together
% AUTHOR: [FILL IN NAME HERE]
function c = addme(a, b)

% Add a and b together and store in c
c = a + b

end
```

5) **Matlab for Loops**
- A for-loop in Matlab is similar to other programming languages. Below is an example of looping through a body of code 5 times:

In Java	**In Matlab**
int i;	for i = 1:5
for(i = 0; i < 5; ++i)	
{	% Code
// Code	
}	end

- Note some of the major differences:
  - No need to enclose your lines of code in a loop inside a set of curly braces
  - Must provide an **end** statement to the end of your loop

6) **Summary of Element Extraction**

- Review So far:
  - To extract the 2nd element of a vector: y = x(2);
  - To extract the first three elements: y = x(1:3);
  - To extract all the elements starting from the 3rd element: y = x(3:end);

- To extract ALL the elements (from a row or column), use the colon operator:
  - y = x(:) ; % Extract all elements of vector
  - y = A(1, :); % Extract all columns from the first row
  - y = A(:, 1); % Extract all rows from the first column