University of Washington
Department of Electrical Engineering
EE 235 Lab 4 Background: Convolution & Functions

Matlab Concepts/Functions To Review	New Matlab Concepts/Functions You Will Learn
• Using the zeros function • Determining number of samples needed • Changing value(s) in a vector • Creating time samples vector • Relationship between index and time • Plotting functions & using subplot	• Using the conv function with amplitude scaling • For loops and decision functions

BACKGROUND MATERIAL

1) **Matlab Review**

Concepts/Functions	Sample Code
Create vector with colon operator j:i:k	m = 0:2:10; % [0 2 4 6 8 10] n = 1:5; % [1 2 3 4 5]
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
Horizontal concatenation	z = [x, y]; % x and y must have the same number of rows
Vertical concatenation	z = [x; y]; % x and y must have the same number of columns

<table>
<tr><td>Using zeros and ones functions</td><td>x = zeros(10, 1); % Column vector with 10 elements all = 0
y = ones(10,1);   % Column vector with 10 elements all = 1</td></tr>
<tr><td>Opening a new figure window</td><td>figure;</td></tr>
<tr><td>Using a 2 x 1 subplot and plotting on 1^{st} figure</td><td>subplot(2, 1, 1);
plot(……..);</td></tr>
<tr><td>Plotting a signal x vs. t</td><td>plot(t, x);</td></tr>
<tr><td>Turn on grid lines</td><td>grid on;</td></tr>
<tr><td>Changing axes limits</td><td>xlim([0 10]);
ylim([-5 5]);</td></tr>
<tr><td>Labeling axis and adding plot title</td><td>xlabel(‘Time’);
ylabel(‘x(t)’);
title(‘Signal x(t)’);</td></tr>
<tr><td>Plotting two graphs on one plot with different colors</td><td>plot(t, x1, ‘r’);   % Use color red
hold on;
plot(t, x2, ‘g’);   % Use color green

<table>
<tr><th>Color Specifier</th><th>Color</th></tr>
<tr><td>r</td><td>Red</td></tr>
<tr><td>g</td><td>Green</td></tr>
<tr><td>b</td><td>Blue</td></tr>
<tr><td>c</td><td>Cyan</td></tr>
<tr><td>m</td><td>Magenta</td></tr>
<tr><td>y</td><td>Yellow</td></tr>
<tr><td>k</td><td>Black</td></tr>
<tr><td>w</td><td>White</td></tr>
</table></td></tr>
<tr><td>Adding a legend to a plot</td><td>legend(‘label1’, ‘label2’, ‘label3’);</td></tr>
<tr><td>Changing axes limits</td><td>xlim([0 10]);
ylim([-5 5]);</td></tr>
<tr><td>Labeling axis and adding plot title</td><td>xlabel(‘Time’);
ylabel(‘x(t)’);
title(‘Signal x(t)’);</td></tr>
<tr><td></td><td></td></tr>
</table>

Load 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) **Review: On Sampling Rate and Determining Number of Samples Needed**

- Recall what sampling rate **Fs** means and how we are using it in Matlab. It represents the number of data samples per second. Suppose we create a signal using a sampling rate of Fs = 4000. This means that, per second, we have 4,000 samples of data.

- Given a particular sampling rate **Fs**, suppose we want to create a vector with duration of **t** seconds. How many samples **m** do we need? We can create a formula by performing a simple unit conversion

$$samples = seconds \times \frac{samples}{second} + 1 \quad \rightarrow \quad \boxed{m = t \times F_S + 1}$$

- The additional + 1 is needed since we are using discrete-time signals and need to count the first element.

- Ex: Suppose Fs = 4000. How many samples **m** do we need for a signal to last 3 seconds?

$$m = t \times F_S + 1 = 3 \times 4000 + 1 = \boxed{12{,}001\ samples}$$

- Similar to the number of samples needed, we can access the index ***i*** of the time vector corresponding to time t_0 by the general formula

$$\boxed{i = t_0 \times F_S + 1}$$

- Ex: Extracting t = 5 and storing in y

```
index = 5 * Fs + 1;
y = x(index);
```

- Ex: Accessing t = 5 and changing value to 1

```
index = 5 * Fs + 1;
x(index) = 1;
```

- Ex: Extracting 0 ≤ t ≤ 5 and storing in y

```
start_index = 0 * Fs + 1;
end_index = 5 * Fs + 1;
y = x(start_index:end_index);
```

3) **More Plotting Commands**
   - To **plot multiple graphs** on one axis, use the **hold on** command

     ```
     figure;
     hold on;
     plot(t, x);
     plot(t, y);
     ```

   - To **change the color of your graph**, you can add a third parameter to the function call to **plot** specifying the color you want

     ```
     plot(t, x, 'r'); % Plot in red
     ```

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

   - To **add a label for each graph**, you can use the function **legend**:
     Syntax: legend(string1, string2, string3, …);
     Description: Puts a legend on the current plot using the specified strings as labels
     Usage: % Suppose you plotted x1(t) and x2(t)
     legend('x1(t)', 'x2(t)');

4) **The *conv* Function**
   - Matlab has a function called **conv** that you can use two convolve two signals x and h
     Syntax: y = conv(x, h)
     Description: Convolves x and h

   - It assumes that the sampling rate (and hence, sampling period) are the same for both signals. For this lab and the rest of this class, this will always be the case

   - In theory, the length of the output of the convolution for continuous-time signals should be: **length(x) + length(h)**. This is not the case in Matlab.

   - In Matlab, the length of output **y** is actually **length(x) + length(h) – 1,** where "length" refers to the length of the vector that is passed to the conv function. (The same will be true for the length of the non-zero part of the signal.) The reason for this is because again Matlab deals with discrete-time signals and not continuous-time signals. In EE 341, you will learn why the length of the output is slightly different for the case of discrete-time signals. For purposes of EE 235, you need to know that the result of a convolution will have a different length than its input signals, which you may need to account for in plotting.

5) **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 difference affects the convolution computation. For impulse functions, which have no width and only area, there is no impact if the height of the impulse in discrete time is the area of the continuous-time impulse. However, for convolutions of other signals, 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 sample period 1/Fs:
  y = (1/Fs) * conv(x, h)

6) **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
```

7) **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 = 0; i < 5; ++i)
{
    // Code
}
	for i = 1:5
	
	    % 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

8) **Summary of Element Extraction**

- Review So far:
  - To extract the 2^{nd} 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 3^{rd} 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 the first row
  - y = A(:, 1); % Extract the first column