University of Washington
Department of Electrical Engineering
EE 235 Lab 5 Background:
Time Domain to Frequency Domain

Matlab Concepts/Functions to Review	New Matlab Concepts/Functions
• Generating a signal in the time domain • Creating time samples vector • Playing sound signal • Using the **pause** function • Plotting basics and subplots • Using the **zeros** function • For loops • Vector extraction and concatenation • Writing functions	• Using **fft** and **fftshift** functions • Using **abs, find** and **min** functions • Creating string variables • String concatenation

BACKGROUND SECTION

1) **Matlab Review**

Concepts/Functions	Sample Code
Creating Matlab variables	x = 1; % Scalar y1 = [2; 4 ; 6]; % Column vector y2 = [1 3 5]; % Row vector z = [3 6 9; 1 1 1]; % 2 x 3 Matrix
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 b2 = y1(:); % Extract all elements c = z(1,2); % Extract one element d = z(2, 1:2); % Extract multiple elements e = z(2, :); % Extract all elements in 2nd row
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
Vector concatenation	

	z = [x, y]; % horizontal: x & y must have the same # rows
	z = [x; y]; % vertical: x & y must have the same # columns

Generating a signal in time domain	% Generate time samples t = 0:1/Fs:5 % 0 ≤ t ≤ 5 with Fs = 2 % Generate actual signal x = cos(pi*t); % x(t) = cos(πt)
Creating a vector that lasts t seconds	x = zeros(1, 3*Fs + 1); % zeros vector that lasts 3 sec

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);
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	x = x + 1 % Display output of calculation A % Display contents of matrix A

for loops	for i = 1:5 % Code end
Function Header Examples	function y = myexample(x) function [y1, y2] = myexample(x) function [y1, y2] = myexample(x1, x2)

Example Function with Header	% 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

2) **Function fft: Converting Signal from Time Domain to Frequency Domain**
   - We will use a function called **fft**, where **fft** stands for Fast Fourier Transform. The FFT is an efficient implementation of a discrete Fourier transform, which is a discretized version of the Fourier Transforms that we are learning about in class. Similar to how time domain signals can only be defined at discrete samples, the **fft** can only compute samples of the signal in the frequency domain as well. You will learn about the FFT in more detail in EE 341.

   - Usage of **fft**: Suppose we have a signal **x(t)** that is sampled at some sampling rate **Fs**. Also, suppose we want to generate **N** = 1024 samples in the frequency domain. To convert **x** to the frequency domain, we would call **fft** as such:
     ```
     N = 1024;
     X_fft = fft(x, N);
     ```

   - The period between the frequency samples is given by: $\frac{2\pi F_S}{N}$

   - The **fft** computes samples over the range: $0 \leq w < 2\pi F_s$. Note that the upper bound is only less than and not less than or equal to, which means the last frequency sample actually occurs at $2\pi F_s - \frac{2\pi F_S}{N}$

3) **Function fftshift: Centering Frequency Domain at w = 0**
   - The **fft** actually starts computing samples in the frequency domain starting at w = 0. To center our frequency samples around w = 0, we need to use a function called **fftshift**.

   - We can convert x to the frequency domain and make sure the signal is centered at w = 0 with:
     ```
     N = 1024;
     X_fft = fftshift(fft(x, N));
     ```

- The period between the frequency samples is still given by: $\frac{2\pi F_S}{N}$

- After the **fftshift**, the range of frequency samples is now: $-\pi F_S \leq w < \pi F_s$

- We can define the frequency samples vector, given the sampling rate **Fs** and the number of frequency samples **N**:

```
w_period = 2*pi*Fs/N;
w = (-N/2:(N/2)-1)*w_period;
```

4) **Summary of Element Extraction**

- Review So far:
  - 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 columns from 1st row of matrix: y = A(1, :)

- New Scenario: Suppose we want to extract the third, fourth, and sixth elements a vector
  - Instead of using the colon operator, you would list out the elements in a vector:

```
y = x([3 4 6]);
```

  - Alternatively, you could define the list of elements in a vector and then extract:

```
index_x = [3 4 6];
y = x(index_x);
```

5) **Other New Functions in Matlab**

- Function **abs**: Used to compute the magnitude of a complex number
  - Usage: Suppose we want to compute the magnitude of a variable X

```
X_abs = abs(X)
```

- Function **find**: In the last lab, we used this function to find the maximum of a vector. In this lab, we will use the function find to locate indices of a vector where the values of a vector cross a certain threshold. Consider the following row vector X:

$$X = [1 \quad 3 \quad 4 \quad 2 \quad 1 \quad 3]$$

  - Suppose we want indices in X where X > 2
  - Then, we would use the **find function** as such:

```
index_X = find(X > 2)
```

  - In this example, index_X = [2 3 6], which are the indices in X where X > 2

- Function **min**: Used to find the minimum value and corresponding index location in a vector
  - Usage: Find minimum and location of minimum in vector x

```
[min_x, index_min_x] = min(x);
```

6) **String Variables**

- Like other programming languages, a string in Matlab is a vector of characters
- In Matlab, string values are enclosed in single quotes
- To create a string, we use the same format **<name> = <initial_value>**:

```
s = 'Any characters';
```

- We can concatenate two strings together using square brackets:

```
s = ['Any ', 'characters'];
```

- Alternatively:

```
str1 = 'Any ';
```

```
str2 = 'characters';
str = [str1, str2];
```

- To convert a number to a string, use the function **num2str** as such:

```
x = 1;
str = ['x is ', num2str(x)];    % Value of str is "x is 1"
```