University of Washington
Department of Electrical Engineering
EE 235 Lab 2
Continuous-Time Signals and Transformations in Time

In this lab, we will use Matlab to perform transformations in time on continuous-time signals. We will also introduce students to Matlab syntax for functions. **Note: all pre-lab exercises must be completed by each student individually and submitted via Canvas before walking in to the lab. No pre-lab -> no credit for entire lab!**

Important Concepts From Lecture You Will Use In Lab 2

- Performing transformations in time: time shift and time scale
- Performing multiple transformations in time using order of operations

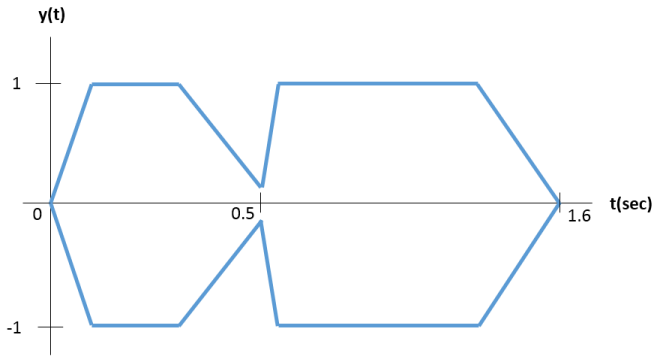
What is Expected From You In Lab 2

- Completion of 3 pre-lab exercises (5 points)
- Completion of 2 in-lab check offs with TA (5 points)
- Completion of a lab report (10 points)

EXERCISE #1: Time Scale Operation

In this exercise, we will learn how to call and test a function written in Matlab that performs time scaling. To test the time scale function, we will use the built-in Matlab file, **train.mat**.

1) ***Pre-Lab Questions***: The envelope of the audio signal **y(t)** in **train.mat** is approximated below:



   a) Let us first analyze the signal **y(2t)**.
      i) Sketch the signal **y(2t)**. Label your plots appropriately. In particular, label the critical times **t = 0**, **t = 0.5**, and **t = 1.6** after the time scaling.

      ii) Is this an example of an audio signal speeding up or slowing down?

   b) Repeat (a) for the signal **y(0.5t)**.

   c) To perform a time scaling and compute **y(t) = x(at)** given an input signal **x(t)** that has been sampled at rate **Fs**, we will use a function that has the following header:

[y, t] = timescale(x, Fs, a)

Read Background Sections (6) and (7) on functions, and answer the following:

i) What are the input variables of this function? What are the output variables of this function?

ii) How would you call this function if you wanted to time scale a signal vector **y** (sampled with a sampling rate of **Fs**) by a factor **2**, the result of which you want to store in the output variables **y1** and **t_y1**?

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

Matlab Concepts/Functions to Review	New Matlab Concepts/Functions You Will Learn
• Creating a variable • Loading Matlab data files • Computing time samples vector • Plotting a signal • Plotting multiple graphs using subplot • Adjusting x and y axes • Labeling plots • Playing sound files	• Syntax for Matlab functions • Calling Matlab functions with multiple outputs

2) ***Lab Exercise***: We will now test function **timescale** using the same signal from pre-lab.

a) Open Matlab. Then, download the Matlab script **timescale.m** from the course website and save the file in your working folder. Open the file in Matlab and read through the code and comments. Note: this script will NOT be modified in this lab exercise.

b) We will now test this function on a separate script. Open a script and call it **Ex1.m**.

c) To start with **Ex1.m**, write the lines of code to clear all figures and close all windows.

d) We will now load and plot the audio train file as follows:

i) Load the data file **train.mat** via the command **load**. The file will have two variables:
y: this is sound vector
Fs: this is sampling rate of sound vector **y**

ii) Compute corresponding time samples of this sound vector using **y** and **Fs**. Store the time samples in vector **t_y**. **If needed, refer to the Matlab Review section**.

iii) Load a new figure window.

iv) Using a 3 x 1 figure window, plot the signal **y vs. t_y** on the 1st subplot. Adjust the limits of the x-axis to be between 0 and 4, and the limits of the y-axis to be between -2 and 2. Make sure to provide labels and a title to your plot. **If needed, refer to the Matlab Review section**.

e) Run the lines of code you have so far via **Editor -> Run** or by pressing **F5**. Verify that your plot matches the envelope signal **y(t)** you analyzed in pre-lab.

f) We will now write the code to implement the transformation for signal **y(2t)**

i) Write the line of code to call the function **timescale** so that you pass in the inputs **y**, the sampling rate **Fs**, and the correct time scale factor. Store the outputs of the function in the variables **y1** and **t_y1**.

ii) Using the same figure window you plotted **y vs. t_y**, plot **y1 vs. t_y1** on the 2nd subplot. Make sure you to not run the command **figure** again. Otherwise, you will load and plot on a new figure window. Adjust the x-axis so that limits are between 0 and 4, and adjust the y-axis to be between -2 and 2. Label your plot appropriately.

iii) Run the script you have so far via **Editor -> Run** or by pressing **F5**. Verify that your plot of **y(2t)** matches your pre-lab results.

g) Repeat (f) for the signal **y(0.5t)**. Store your output in the variables **y2** and **t_y2**. Plot **y2 vs. t_y2** on the same figure window as the 3rd subplot. Use the same axes limits, and label your plot appropriately. Run your script to verify that your plot of **y(0.5t)** matches your pre-lab results.

h) We will now play all three audio signals to confirm your answers to the pre-lab questions on whether a signal speeds up or slows down.

i) Using the function **sound**, write the line of code to play the original sound signal **y**. Then, with a pause of 4 seconds in between to avoid sound overlap, play the sound signal **y1**. Finally, play the signal **y2**.

ii) Run your entire Matlab script and pay close attention to the speed of each audio signal. Verify your answers in pre-lab.

iii) If, for some reason, you cannot hear your audio files over RDP, add the following lines of code to create two separate wav files for **y1** and **y2**, which you can then cover over and play on your main desktop:

- audiowrite('Ex1_y1.wav', y1, Fs);
- audiowrite('Ex1_y2.wav', y2, Fs);

i) Provide comments (using the symbol %) in your script.

3) ***Lab Check-Off #1 of 2***: Demonstrate your Matlab script **Ex1.m** to the lab TA and show you that you know how to perform time scaling in Matlab.

4) ***Lab Report Question #1 of 3***: Suppose a student runs the **figure** command before every call to **subplot**. When you run your script Ex1.m again, what changes do you expect to see? How will the plots change?

EXERCISE #2: Time Shift Operation

In this exercise, we will write our own Matlab function to perform a time shift. The built-in Matlab file, **train.mat**, will again be used to test this function.

1) ***Pre-Lab Questions***:
   a) Consider the same **y(t)** used from Exercise #1. Let us first analyze the signal **y(t – 3)**:

      i) Sketch the signal **y(t – 3)** from **0 ≤ t ≤ 5**. Make sure to label all important points, in particular, the locations of critical times **t = 0**, **t = 0.5**, and **t = 1.6** after time shifting. Note: we are sketching the signal only from **0 ≤ t ≤ 5** because all audio signals are assumed to start at time t = 0.

      ii) Is this signal getting delayed or advanced?

   b) Repeat (a) for the signal **y(t + 0.5)** from **0 ≤ t ≤ 5**.

   c) **Read Background Section (5)** and answer the following questions:

      i) Suppose we wish to create a vector that lasts for 3 seconds. How many samples **ny** do we need given that the sampling rate is Fs = 44,100?

      ii) Suppose we wish to create a vector that lasts for a variable **t_y_end** seconds. Write the general mathematical equation for the number of samples **ny** needed in terms of **t_y_end** and the sampling rate variable **Fs**.

      iii) Suppose we create a vector using a sampling rate of Fs = 44,100 that lasts a duration of 5 seconds. Answer the following questions:
         - At what index i is time t = 0.5 sec located in this vector?
         - Write a general mathematical equation for the value of index **i** that equals where time **t = to** seconds is located in this vector. Your equation should be in terms of **to** and sampling rate variable **Fs**.

      iv) Suppose a signal **x(t)** undergoes a time shift to the right by 3 seconds to produce a new signal **y(t) = x(t - 3)**. If **x(t)** has a duration and end time of 2.3 seconds, what will the new duration and end time of **y(t)** be?

      v) Suppose a signal **x(t)** undergoes a time shift to the right by a variable **to** seconds to produce a new signal denoted by **y(t) = x(t - to)**. If **x(t)** ends at time **t_x_end** seconds, what is the formula for the new end time of **y(t)**? Call the value of this new end time **t_y_end**.

      vi) Suppose we know a signal vector **x** has **length(x)** number of samples. Write the general mathematical equation for the duration (or end time) of this signal in terms of **length(x)** and the sampling rate variable **Fs**. Call the value of this end time **t_x_end**.

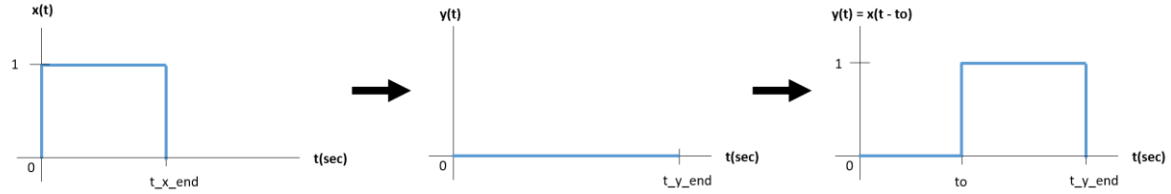
d) **Read the Background Section** on these topics to prep for the upcoming lab exercise:

Matlab Concepts/Functions to Review	New Matlab Concepts/Functions
• Creating variables • Getting length of vector • Extracting elements of a vector • Changing elements of a vector • Relationship between time and index • Plotting basics • Loading and playing sound files	• Using the zeros function • Using the ceil function • Accessing the end of a vector or matrix • Creating a vector that lasts **t** seconds • Converting number of samples to number of seconds

2) ***Lab Exercise***: We will now write and test the function to do a time shift.

a) Download and open the file **timeshift.m** from the course website. Unlike in Exercise #1, we will be writing a few lines of code in this file.

b) We will now fill in the code for **timeshift.m**. We will start with a time shift to the right under the "if" statement. We will simulate a time shift to the right by **to** seconds by creating a new vector **y** with an added duration of **to** seconds, and then inserting the values in vector **x** into the appropriate location in vector **y**. This process is shown below:



i) For the first lines of code under the "if" statement, let us create the new vector **y** with the appropriate size and initialize its values to zero by following these steps:

- Using your formula from pre-lab, compute the end time (in seconds) of input vector **x** using the length of **x** and the sampling rate **Fs**. Store the value in variable **t_x_end**.

- Compute what the new duration of vector **y** will be following a time shift of to seconds. Using your formula from pre-lab, write your formula in terms of the time shift value **to** and the end time value **t_x_end**, and store that value in variable **t_y_end**.

- Using your equation from pre-lab, compute the number of zeros needed to last **t_y_end** seconds given the sampling rate is **Fs**. Store this result in the variable **ny**. Use the function **ceil** to round your answer to an integer. **Refer to Background Section (3) on how to use the ceil function**.

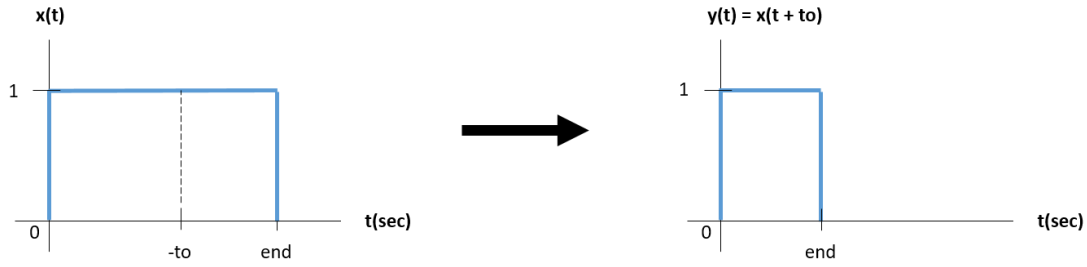
- Using the **zeros** function, create a column vector of zeros called **y** using the value **ny** as the number of rows. **Refer to Background Section (2)**.

ii) Now that we have our column vector **y**, the final step is to insert the input signal **x** in to the output signal **y** at time **t = to** seconds. To do this:

- Using your equation from pre-lab, compute the index where time **t = to** seconds given the sampling rate **Fs**. Store this result in the variable **index_to**.

- Assign the entire input signal **x** to the output signal **y** starting at **index_to** up to the **end** of vector **y**. That is, write the following line of code:
  y(FILL IN CODE) = FILL IN CODE;
  For help on accessing the end of a vector, **refer to Background Section (4)**.

c) Now let us perform a time shift to the left under the "else" statement. To distinguish a time shift to the left, we pass a **negative** value for **to**. We will simulate this shift by extracting the signal **x** starting at time **t = -to** seconds and then discarding the rest of **x**.



i) First, we need to determine at which index we will start extracting values from signal **x**. Using the sampling rate **Fs**, compute the index **start_index** where time **t = –to** seconds in vector **x**. Use the function **ceil** to round the answer to an integer.

ii) Extract samples of **x** starting at index **start_index** until the **end** of the signal **x**, and store this result in the vector **y**.

d) For the final line of code in **timeshift.m**, compute the corresponding time samples of output vector **y** using the sampling rate **Fs**. Store this result in the variable **t**.

e) Before you exit **timeshift.m**, add a function header comment, as well as in-line comments. **Refer to Background Section (7)** for an example on function comments.

f) We will now test your function by writing a test script **Ex2.m**. Open up a new script and call it **Ex2.m**. To start in **Ex2.m**, first clear all variables and close all figures.

g) Load data file **train.mat** via the command **load**.

h) Using the same procedure in **Ex1.m**, plot the audio signal **y** as the 1st plot in a 3 x 1 subplot figure window. Make sure you compute the corresponding time samples **t_y** before plotting. Adjust the limits of the x-axis to be between 0 and 5. The y-axis does not need to be adjusted. Make sure to label all your axes and title your plot.

i) We will now write the code to perform the time shift **y(t - 3)**

   i) Call the function **timeshift** so that you pass in the inputs **y**, the sampling rate **Fs**, and the correct time shift amount needed. Store outputs in the variables **y1** and **t_y1**.

   ii) Using the same figure window you plotted **y vs. t_y**, plot **y1 vs. t_y1** on the 2nd subplot. Adjust the x-axis so that limits are between 0 and 5. Label your plot appropriately.

   iii) Run the script you have so far via **Editor -> Run** or by pressing **F5**. Verify that your plot of **y(t - 3)** matches your pre-lab results.

j) Repeat (i) for the signal **y(t + 0.5)**. Store your output in the variables **y2** and **t_y2**. Plot **y2 vs. t_y2** as the 3rd subplot on the same figure window. Make sure to run your script to verify that your plot of **y(t + 0.5)** matches your pre-lab results.

k) We will now play all three audio signals to confirm your answers to pre-lab questions on whether a signal is delayed or advanced. Using the function **sound**, play the original sound signal **y**. Then, play the sound signal **y1**. Finally, play the sound signal **y2**. To avoid overlap between sound play, use a pause of 5 seconds between each function call.

l) Run your entire Matlab script and pay close attention when you play the three audio signals. Verify your answers to pre-lab. If, for some reason, you cannot hear your audio files over RDP, add the following lines of code to create two separate wav files for **y**, **y1**, and **y2**, which you can then cover over and play on your main desktop:
   - audiowrite('Ex2_y.wav', y, Fs);
   - audiowrite('Ex2_y1.wav', y1, Fs);
   - audiowrite('Ex2_y2.wav', y2, Fs)

m) Provide in-line comments for your script.

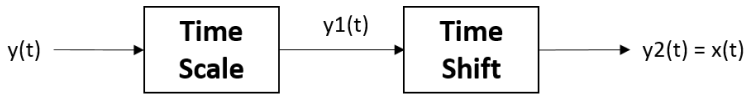
3) ***Lab Check-Off #2 of 2***: Demonstrate your Matlab script **Ex2.m** to the lab TA.

EXERCISE #3: Recovering Popular TV/Movie Audio File

Mayday, mayday, mayday, [insert your name here]! The TA needs your help! All of the TA's favorite audio files of popular tv/movie quotes have been tampered with by an evil former EE235 student using what s/he knew about transformations in time. Your mission, should you choose to accept, (which you better accept or else you will not get credit for this lab, hence you will lose 20 points from your total lab score, and that may be the difference between you getting a 3.9 versus a 4.0 *wink wink*) is to undo the evil former student's transformations in time and recover the signals in the TA's prized possession of audio files.

We need to first prep and train you in the art of undoing transformations in time. Luckily, the TA knows the evil student failed his/her midterm question on time reversal, so the TA is certain the student only used time scaling and time shifting. The TA is also certain the student used transformation values of factors 0.5 or 2.0 because 2.0 / 0.5 = 4.0, which is the grade the former student was trying to get. Therefore, the audio signal either went through a sequence of transformations to produce **y(t) = x(2t – 0.5)** or **y(t) = x(0.5t – 2).**

Regardless of which output is correct, the transformations in time can be undone by performing the transformations in reverse order. If we say the signal **x(t)** first went through a TIME SHIFT and then a TIME SCALE to produce **y(t)**, then to undo the transformation, we need to perform a TIME SCALE on **y(t)** first followed by a TIME SHIFT to recover **x(t)** as shown below:



1) ***Pre-Lab Questions***: The goal now is to determine how much to scale and shift **y(t)** in order to recover **x(t)**.

   a) Consider the case where **y(t) = x(2t – 0.5)**.

      i) The goal of the TIME SCALE is to scale **y(t)** and produce a signal **y1(t)** that gets rid of the factor 2. That is, if we perform a TIME SCALE on **y(t)** with factor **"a"**, then **y1(t) = y(at) = x(2(at) – 0.5)** which needs to equal **x(t – 0.5)**. What scaling factor **"a"** produces this result?

      ii) The goal of the TIME SHIFT it to shift **y1(t)** and produce a signal **y2(t)** that removes the time shift 0.5. That is, if we perform a TIME SHIFT of **"to"** on **y1(t)**, then we should get **y2(t) = y1(t – to) = x((t – to) – 0.5)** which needs to equal **x(t)**. What time shift **"to"** produces this result?

   b) Repeat (a) for the other possible transformation **y(t) = x(0.5t – 2)**. What should the scaling factor **"a"** be? What should the time shift **"to"** be?

2) ***Lab Exercise***: We will now try to undo the transformations in time by testing the values you computed in pre-lab.

a) Ask the TA to assign you one of the 10 Matlab sound files on the class website. Download the file in to your workspace folder.

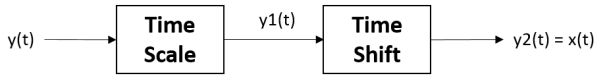
b) To understand your task for this exercise, let us load and play the corrupted sound file on your COMMAND window. Type the following two lines of code:
   i) Load your assigned Matlab data file
   ii) Play your sound file

c) Now, open up a new script file and call it **Ex3.m**. Clear all variables and close all figures.

d) Load your Matlab data file. The Matlab data file will again have two variables: **y** and **Fs**.

e) Plot the distorted audio signal **y** as the 1st subplot in a 2 x 1 subplot figure window. Make sure to compute the corresponding time sample **t_y** before plotting. No need to adjust the axes, but make sure to label and title your plot.

f) We will now undo the operations for **y(t) = x(2t – 0.5)** using the same order of operations shown below from pre-lab:



   i) Undo the time scale operation on audio signal vector **y** by calling the **timescale** function and passing the appropriate arguments using your transformation values from pre-lab. Call the output signal **y1** and the corresponding time samples **t1**.

   ii) Undo the time shift operation on audio signal **y1** by calling the **timeshift** function using your transformation values from pre-lab. Call the output signal **y2** and the corresponding times samples **t2**.

g) Repeat (f) for **y(t) = x(0.5t – 2)** using your transformation values from pre-lab. To prevent you from overriding your previous results, call the outputs of the **timescale** function **y3** and **t3** and the outputs of the **timeshift** function **y4** and **t4**.

h) Run your script and make sure you do not have any errors.

i) Now that we have the two recovered signals **y2** and **y4**, we need to determine which one is correct. On the COMMAND WINDOW, play your audio signals. First, play **y2**. Then, play **y4**. Only one will produce an audible result. Which was the correct distorted signal? **y(t) = x(2t – 0.5)** or **y(t) = x(0.5t – 2)**?

j) Once you have figured out which recovered signal is correct (either **y2** or **y4**), go back to your script **Ex3.m** and plot that signal as the 2nd subplot. No need to adjust the axes. Just make sure to label and title your plot.

k) Write the line of code to play the correct audio signal.

l) Provide comments for your script.

3) ***Lab Report Question #2 of 3***: What popular TV/movie quote did you receive? Can you identify the movie or TV show your quote came from? If you are having trouble identifying some of the words, feel free to ask your TA for help. Here are some hints and fun facts about each quote.

- **S1**: This quote comes from a popular science-fiction movie franchise directed by James Cameron that chronicles the ongoing battle between humans and Skynet. The film series stars a former Governor of California. Five films have been made so far with a sixth film is expected in 2019. Your TA has not seen these movies in ages! Time to give up an upcoming Friday night and weekend to binge watch them again!

- **S2**: This quote comes from the third entry in a popular movie franchise based on a series of British fantasy books chronicling the life of a boy wizard and his friends. Your TA is a huge fan of the movie and the director, who won a 2014 Academy Award for Best Director for Gravity, which was your TA's favorite movie from 2013.

- **S3**: This quote comes from a 2016 standalone spinoff film to a popular space opera film series that your TA was a little late in watching. Your TA just didn't want to wait in line and have to sit in a crammed movie theatre. Boy, it was worth the wait! The movie ended up being an unexpected prequel!

- **S4**: This quote comes from a hugely popular HBO fantasy drama series that is also on your TA's weekly appointment viewing list. Your TA is a huge fan that he waited 6+ hours in line to attend its San Diego Comic-Con program panel and was elated when HBO handed out free gift bags.

- **S5**: This quote comes from an AMC post-apocalyptic horror drama TV series, and is currently one of your TA's favorite shows. The TA likes it so much that at the 2013 San Diego Comic-Con, the TA and a group of friends started standing in line at 4am to gain entrance in to the show's program panel that started at 2pm!

- **S6**: This quote comes from the 55th Disney animated feature film about an unlikely friendship between a rabbit and a fox working to solve a crime. The movie is so good that its creators won a Golden Globe Award for best animated film. Your TA agrees with that win because the TA has already watched this film twice!

- **S7**: This quote comes from an animated sequel to a classic fish tale from Pixar. It is voiced by a famous talk show host and comedian recently honored by a Presidential Medal of Freedom. What an honor indeed!

- **S8**: This quote comes from a web-slinging superhero movie trilogy that has been rebooted so many times that your TA has lost count. Regardless of this, your TA is a huge fan and has plans to watch every movie this superhero is in.

- **S9**: This quote comes from a television sitcom on CBS about a group of four socially awkward geeks (ahem, intellectuals) living in Pasadena, CA. The word you hear is a regularly used catch phrase on the show that means "fooled you!" Your TA has not yet watched this show, but in honor of this show, your TA promises to use this catch phrase throughout the quarter.

- **S10**: This quote comes from a popular, female-led book-turned-movie set in the fictional country of "Panem." Your TA had a goal to finish the book before watching the movie, but was too excited to see the film and only finished 75% of the book. Don't worry, your TA finished the rest of the book soon after and was surprised how many things the movie left out!

4) ***Lab Report Question #3 of 3***: Suppose a student accidentally flips the order of function calls in step (e) when performing the time transformations, and calls **timeshift** first before calling **timescale**. Clearly, the output of the sequence of transformations will no longer be **x(t)**. What will the output be in terms of signal **x**? Define your answer mathematically, and also describe answer in words. As an example, the answer mathematically could be x(0.5t), and the description in words would be the signal x slows down by factor 2.

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 prior to the start of your next lab section

BACKGROUND SECTION

1) **Matlab Review**

Concepts/Functions	Sample Code
Clear all variables and close all figures	clearvars; close all;
Creating a variable	x = 1; % Scalar y1 = [2; 4 ; 6]; % Column vector y2 = [1 3 5]; % Row vector z = [3 6 9; 1 1 1]; % Matrix
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 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
Length of a vector	length(y1);
Size of a vector/matrix	size(y1); % Vector size(z); % Matrix
Num of rows or columns in vector/matrix	size(z, 1); % Num of rows size(z, 2); % Num of columns
Transposing a vector/matrix	a = y1';
Scalar multiplication	% Perform scalar operation 2A B = 2*A;

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

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;
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 playing sound file	load file.mat; % Contains variables y and Fs sound(y, Fs);

2) **zeros Function**
   - We will use the function **zeros** throughout Lab 2 to initialize vectors.
     Syntax: x = zeros(m, n)
     Description: Return an m x n matrix of zeros
     Usage: x = zeros(10, 1) % Creates a column vector of zeros of length 10

3) **ceil Function**
   - In some cases, we need to round a calculation to an integer. The function **ceil** can accomplish that:
     Syntax: y = ceil(x)
     Description: round x to the nearest integer greater than or equal to x
     Usage: n = ceil(i * Fs + 1) % Round calculation to next integer

4) **Accessing End of a Vector or Matrix**
   - You can use the keyword **end** to access the end of a vector or matrix
   - Examples:
     y = x(end); % Access end of vector x
     z = x(3:end); % Access all elements of vector starting at index 3

5) **Sampling Rate, Number of Samples Needed to Represent t Seconds, and Number of Seconds**
   - 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 sampling rate **Fs**, suppose we want to create a vector with duration of **t** seconds. How many samples **n** do we need? We can use the same formula from Lab 1:
     $$\boxed{n = t \times F_S + 1}$$

   - Ex: Suppose Fs = 4000. How many samples **m** do we need for a signal to last 3 seconds?
     $$m = t \times F_S + 1 = 3 \times 4000 + 1 = \boxed{12{,}001\ samples}$$

   - Now let us consider the reverse problem. Suppose we know a vector has **n** number of samples. Given a sampling rate **Fs**, can we determine the duration length of the vector in seconds? How do we do that? We just need to convert from number samples to seconds. We can do that by using the formula above and solving for **t**:
     $$\boxed{t = (n - 1)\ \times \frac{1}{Fs}}$$

   - Ex: Suppose Fs = 4000. What is the duration of the signal in seconds if we know the signal 4001 samples?
     $$t = (4001 - 1)\ \times \tfrac{1}{4000}\ = \boxed{1\ sec}$$

6) **Matlab Function Files**
   - Matlab function files are similar to the scripts you wrote in Lab 1. Function files also have the file extension *.m and thus are also called M-files.

   - The difference between a function and a script is that a script is simply a sequence of commands to be run, while a function (like in other programming languages) can take parameters and return values. A function is therefore a generalized and flexible script.

- Like in other programming languages, a side effect of functions is variable scope. That is, variables created within a function are not available after function has finished running. Therefore, you will not see the variables in your workspace window.

- By contrast, variables created in a script are added to your workspace window and you can look at them anytime afterwards.

7) **Matlab Function Syntax**
   - The syntax to declare a function (or define a function header) is similar to other programming languages. The main differences are the inclusion of the keyword *function* in the function declaration and the ability to return more than one output:

	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)*

- When saving a function in an M-file, you need to make sure the M-file has the same name as the the function. As an example, if you write a function called **addme**, the M-file must have the name **addme.m**.

```
function c = addme(a, b)

    c = a + b

end
```

- Using the **addme** function as an example, on the command window (or on a script M-file), you would type something like: >> *c = addme(1,2)*

- Function files should include a function header and in-line comments. A function header in Matlab is placed above the function declaration. With this header, you can actually view your header as help comments

```
% 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
```

- To view your function header on the COMMAND window, type the following: >> *help addme*