Ryerson University
Department of Electrical & Computer Engineering
Faculty of Engineering & Architectural Science

Course Title:	Signals & Systems I
Course Number:	ELE532
Semester/Year (e.g. F2016)	Fall 2023

Instructor:	Instructor: Professor Javad Alirezaie, TA: Brendan Wood

Assignment/Lab Number:	Lab 1
Assignment/Lab Title:	Working with Matlab Functions, Visualization of Signals, and Signals Properties

Submission Date:	Wednesday, Oct 4, 2023
Due Date:	Sunday, Oct 8, 2023

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Abbas	Syed Maisam	501103255	13	Maisam

Lab 1 - Working with Matlab Functions, Visualization of Signals, and Signals Properties

By: Syed Maisam Abbas

Table of Contents

Introduction

In his lab, the objective is to implement functions to generate graphs/plots, analyze data assigned in external files and/or matrices, compare data files/plots, and gain coding experience using MATLAB, the online programming software. These labs may prove to be beneficial in the future to analyze signals and different kinds of systems in the engineering field.

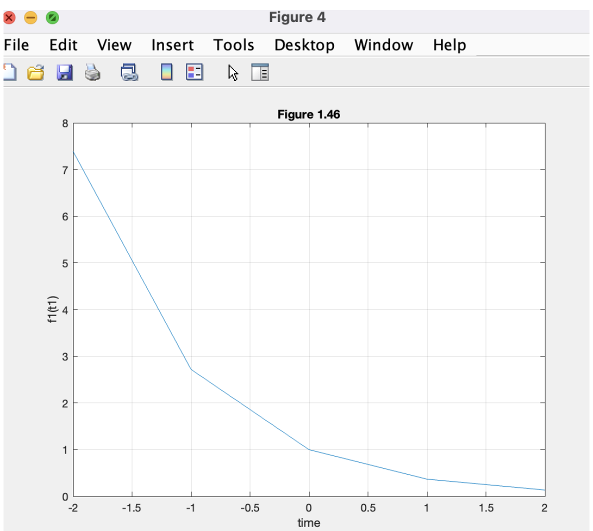
Lab Analysis

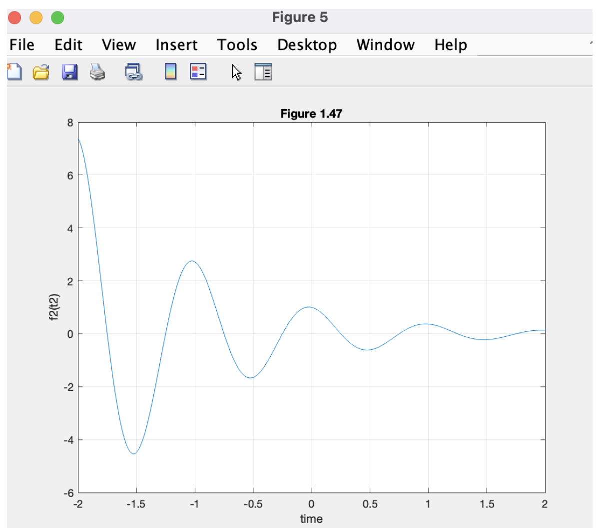
This lab is comprised of four individual parts (A, B, C, and D), each of which have their own assigned tasks and analysis. Completing each of these tasks provides greater insight and knowledge into the operations and commands that can be implemented on MATLAB.

A. Anonymous functions and plotting continuous function

- **A.1**

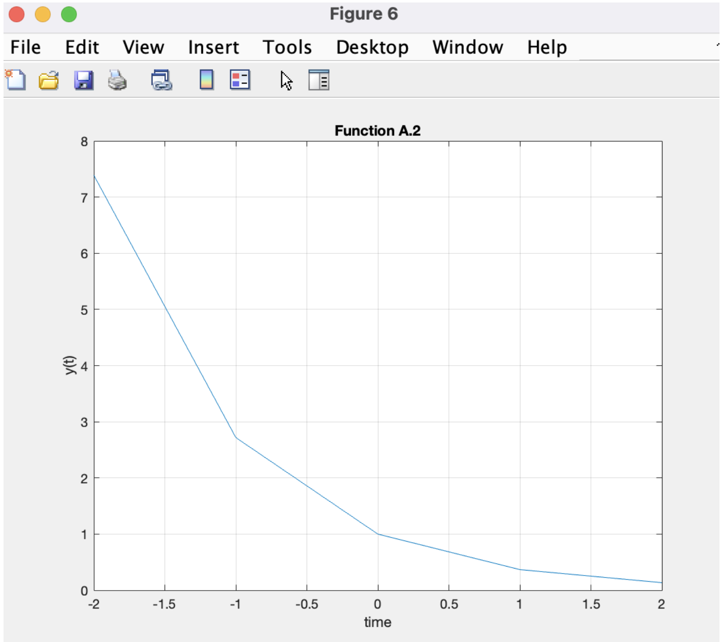
In this problem, figures 1.46 and 1.47 (on page 126) of the textbook were illustrated graphically as shown below:





- **A.2**

In this problem, the following graph was depicted provided the given function and time domain in the lab manual. Function A.2: e^(−t) for t = (-2:2)



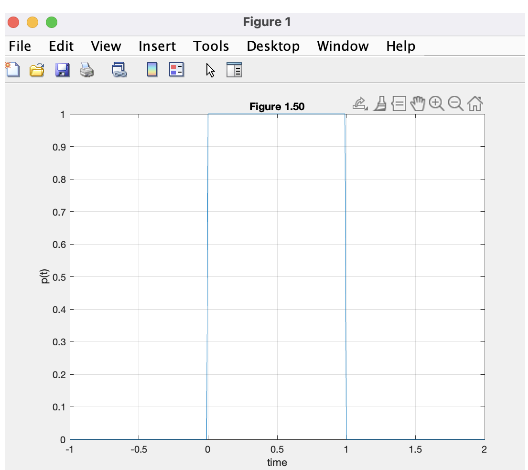
- **A.3**

As shown above, the function A.2 and the function depicted in figure 1.46 can be visually compared. After thoroughly examining the graphs, it is clear that these two graph are the exact same as they share the same time domains and in return the same outputs.

B. Time shifting and time scaling

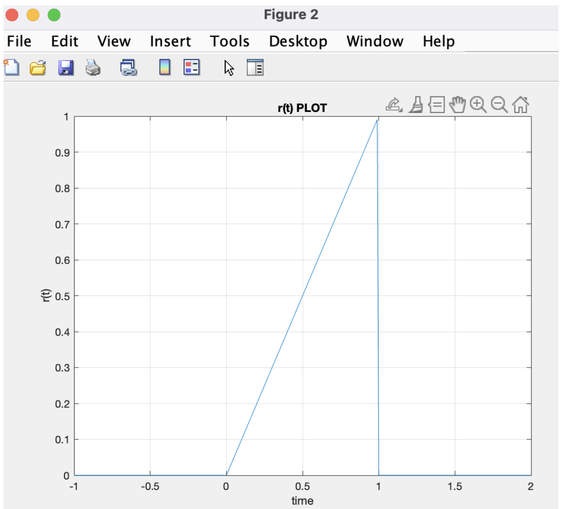
- **B.1**

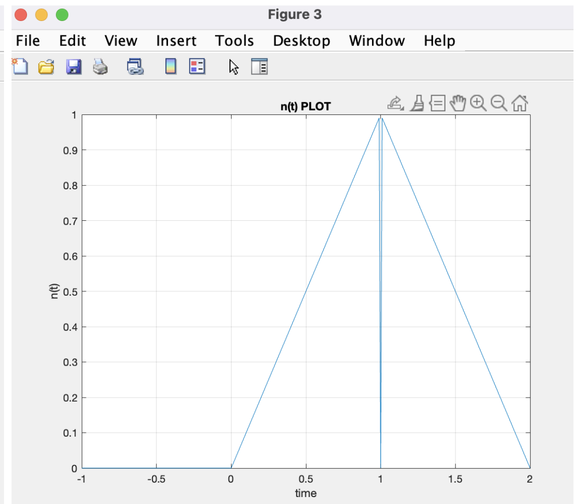
In this problem, figure 1.50 (on page 128) of the textbook is illustrated graphically as p(t), as shown below:



- **B.2**

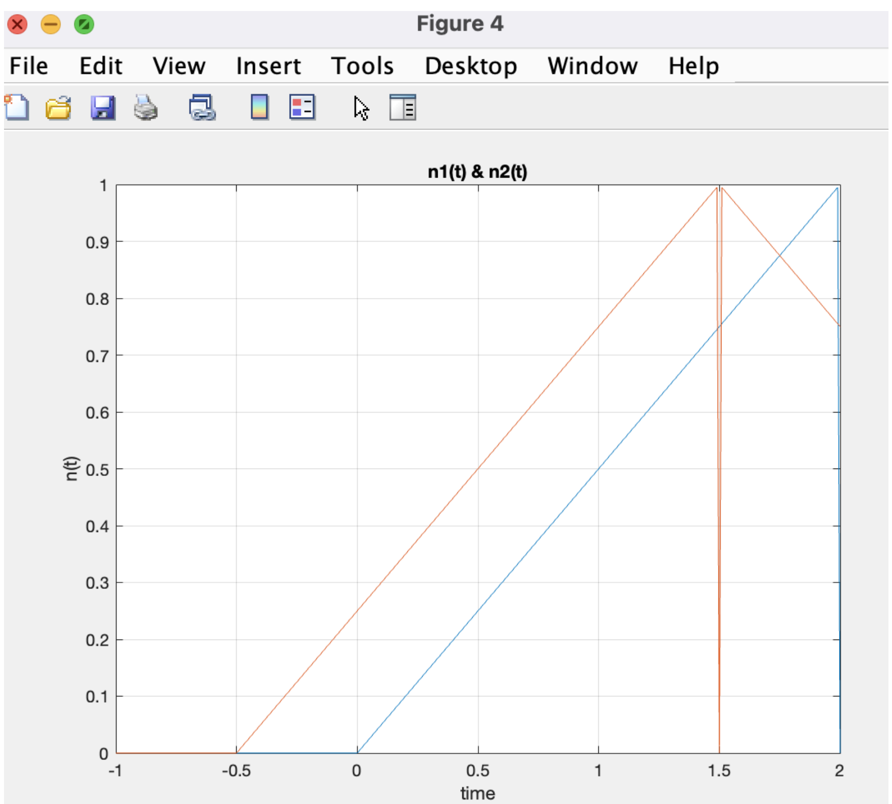
Using the same function from the previous plot [p(t)], the functions r(t) = tp(t) and n(t) = r(t) + r(-t+2) are implemented as stated in the lab manual and depicted below:





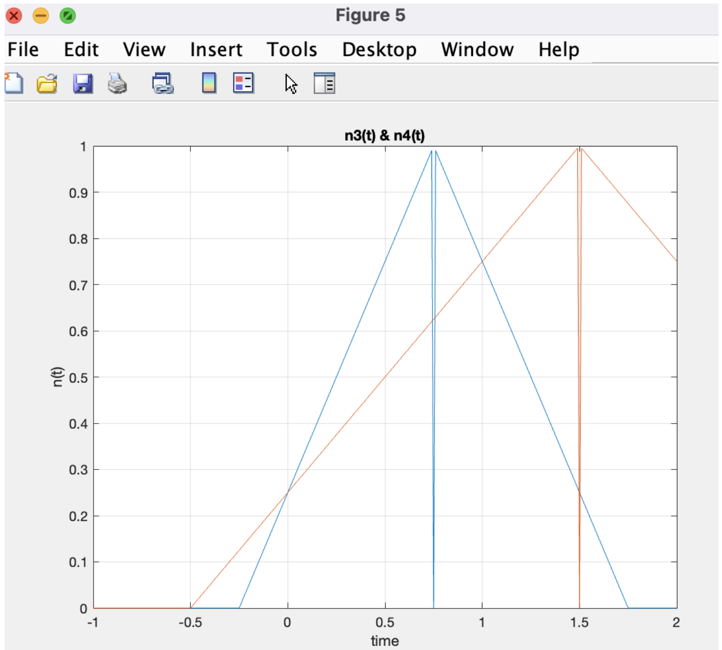
- **B.3**

In this problem, the objective is to plot the two provided signals, as stated in the lab manual: n1(t) = n((½)t) and n2(t) = n1(t + (½)), these functions are depicted below:



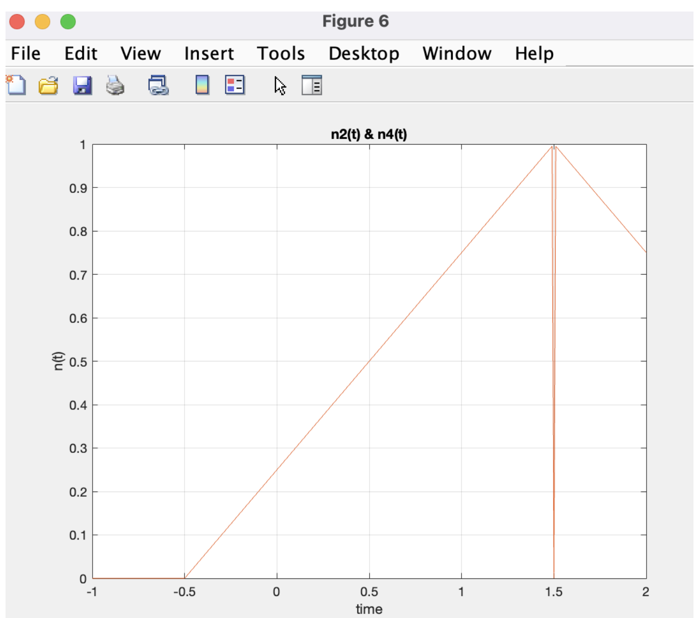
- **B.4**

In this problem, the objective is to plot the two provided signals, as stated in the lab manual: n3(t) = n(t +(¼)), n4(t) = n3((½) t), these functions are depicted below:



- **B.5**

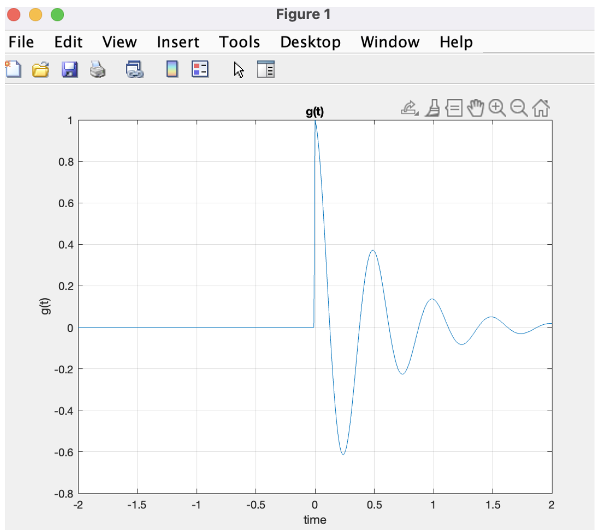
The two function n2(t) and n4(t) are the exact same functions/plots as they share the same intersecpts, slope, and time domain. For visual representation, the two graphs are depicted below:



C. Visualizing operations on the independent variable and algorithm vectorization

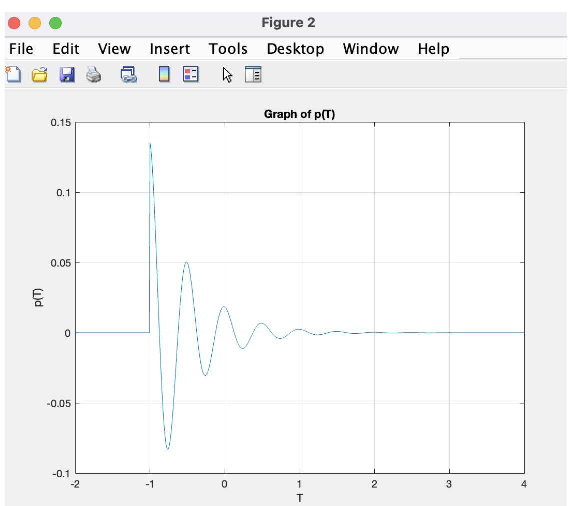
- **C.1**

In this exercise, page 130 of the textbook was referenced to generate the plot g(t) = f(t)u(t), where f(t) = (e^(-2t))cos(4πt). The plot of this function is shown below:



- **C.2**

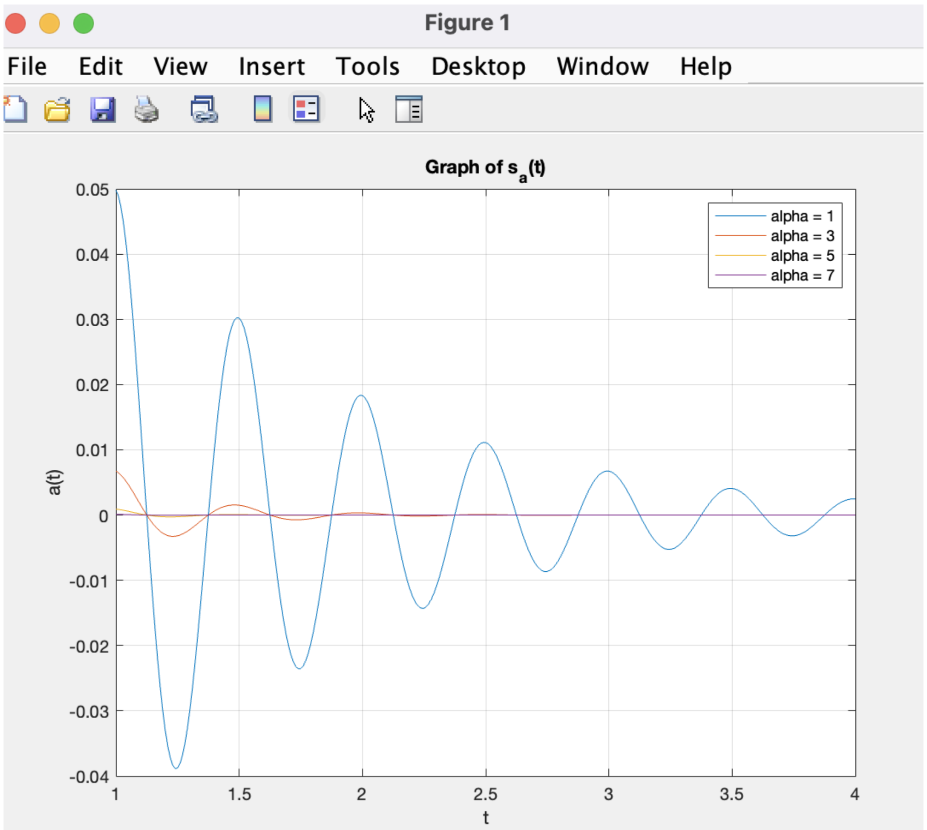
Using the previous function, the plot of s(t) = g(t+1) = [e^(-2)][e^(-2t)]cos(4πt)u(t+1) for t=[-2:0.01:4] was implemented as shown below:



Note that this graph was named p(T) to avoid confusion between similar functions and time domains.

- **C.3**

In this exercise, the objective is to plot the function, sα(t) = [e^(−2)][e^(−αt)]cos(4πt)u(t) for α = [1, 3, 5, 7] in one figure (i.e.: using a legend), over the domain of t=[0:0.01:4]. Note that this plot was illustrated using nested for loops and is depicted below:



- **C.4**

To determine the size of the matrix generated from the plot above, the number of elements must be calculated using the provided domain, t=[1:0.01:4]. Since 4 is the final value in the x-domain and the provided increment is 0.01, (4/0.01) - 1 = 399 (where 1 is the initial value in this domain). This means that the matrix is comprised of 399 elements in a given row, meaning that there are 399 columns. Since the provided values of α = [1, 3, 5, 7], meaning that there are 4 assigned rows in this matrix. Therefore, it can be said that the generated matrix will be a 4 x 399 matrix and is comprised of ((4)(399) =) 1596 elements.

D. Array Indexing

- **D.1**

In this exercise, using the provided 5x4 matrix, A (also provided in the workspace):

$$\mathbf{A} = \begin{bmatrix} 0.5377 & -1.3077 & -1.3499 & -0.2050 \\ 1.8339 & -0.4336 & 3.0349 & -0.1241 \\ -2.2588 & 0.3426 & 0.7254 & 1.4897 \\ 0.8622 & 3.5784 & -0.0631 & 1.4090 \\ 0.3188 & 2.7694 & 0.7147 & 1.4172 \end{bmatrix}$$

The following operations were performed in order to determine their functions:

Operations of the provided functions on matix A	
A(:)	This operation orders the elements in the matrix from top to bottom in order of the rows (left to right).
A([2 4 7])	This operation picks out elements 2, 4, and 7 from the provided matrix. The elements are order from top to bottom, then in terms of columns.
[A >= 0.2]	Converts the postive numbers to 1 and negative numbers to zero in the matrix.
A([A>=0.2])	This operation orders the elements in the matrix from top to bottom in order of the rows (only including positive numbers).
A([A>=0.2]) = 0	Sets all the positive numbers to zero and keeps the negative numbers the same in the matrix.

- **D.2**

This problem is comprised of three parts (a, b, and c)

- **Part A**

The objective of this exercise is to implement matrix B = 1024 x 100 using two nested for loops that will set all elements in this matrix to a value, x, that is 0.01< x <= 0

```
% Problem D.2
    %file accessed from workspace (Matrix B)
    %Matrix B = 1024x100

%PART A
Num_rows = size(B,1); %Allocaing Matrix Size
Num_cols = size(B,2); %Allocaing Matrix Size

for i = 1:1:Num_rows %First For loop for Rows
    for j = 1:1:Num_cols %Second For loop for columns
        if (abs(B(i,j)) < 0.01) % Absolute function of B(i,j) < 0.01
            B(i,j) = 0; % Returning magnitude values below 0.01 to zero
        end
    end
end
```

- **Part B**

The objective of this exercise is similar to part A, however, instead of using nested for loops, the code is implemented using the operations that were analyzed in problem D.1

```
%PART B
B([abs(B)>= 0.01]) = 0;
```

- **Part C**

Finally, in this part of the problem the MATLAB commands 'tic' and 'toc' were used to compare the execution times of the codes implemented in parts A and B.

```
%PART C
    %part Ci
tic
Num_rows = size(B,1); %Allocaing Matrix Size
Num_cols = size(B,2); %Allocaing Matrix Size

for i = 1:1:Num_rows %First For loop for Rows
    for j = 1:1:Num_cols %Second For loop for columns
        if (abs(B(i,j)) < 0.01) % Absolute function of B(i,j) < 0.01
            B(i,j) = 0; % Returning magnitude values below 0.01 to zero
        end
    end
end

fprintf('\nPART A execution time: ')
toc

    %part Cii
tic
B([abs(B)>= 0.01]) = 0;
fprintf('PART B execution time: ')
toc
```

Based on the implementation of this code, the outputs display the execution time of generating the assigned matrix, B, with values between 0.01 and 0:

```
PART A execution time: Elapsed time is 0.012172 seconds.
PART B execution time: Elapsed time is 0.003989 seconds.
```

Therefore, it is clear that the code implemented in part B is much more efficient and easier to work with in comparison to using two nested for loops.

- **D.3**

Finally, in this exercise the objective is to find the threshold of the provided x_audio file that was attached to lab manual and can be accessed through the workspace.

```
% Problem D.3
    %file accessed from workspace (x_audio)
    %x_audio: 20000x1 matrix

Num_rows = size(x_audio,1); %Allocaing Matrix Size
Num_cols = size(x_audio,2); %Allocaing Matrix Size

threshold = 0;

for i = 1: Num_rows
    for j = 1: Num_cols
        if(abs(x_audio(i,j) == 0))
            threshold = threshold + 1;
        end
    end
end

f = @(x_audio) sum(~x_audio(:));
f(x_audio)
fprintf("For the audio data set, the threshold is: " + threshold);

sound(x_audio,8000);
```

Based on the implementation of this code, the output displays the threshold of this audio file and is depicted below:

```
ans =

    58

For the audio data set, the threshold is: 58>>
```

Conclusions

In conclusion, this lab proved to be very beneficial as it provided greater insight into the MATLAB commands, graphing tools, and data analysis. These commands and operations will come of use in future applications of MATLAB.