Lab 1: MATLAB and Python Basics

1. Introduction

The goal of this lab is to give you a quick start with MATLAB and Python. They are powerful computing environments for numeric computation and visualization. They are also useful tools that are widely used in diverse areas.
After this lab, you will be able to write simple MATLAB and Python functions. You will learn how to express discrete-time signals in MATLAB and Python. Then you will exercise writing a function simulating an echo effect.
** If you are not familiar with MATLAB and Python, there are **tutorials** in Blackboard, which cover necessary basics.*

2. Grading

Each lab is divided into two sections: Pre-Lab (4 points) and In-Lab (6 points). You are required to complete **both MATLAB and Python**. For the Pre-Lab component, 4 points will be awarded for the hand-in answer sheet, graphs, and code (must be turned in at when coming to the lab at 1:25 PM). For the In-Lab component, in addition to showing your results (running MATLAB and Python scripts, NOT typing individual commands) to the TA, you also need to be ready to explain your answers to the TA, and be prepared to answer questions.

TIPS FOR A SUCCESSFUL EXPERIENCE IN LAB

- Always complete your pre-lab. Besides earning credit, this prepares you for the tasks in the lab. Hand in the pre-lab at the beginning of lab.
- Feel free to work on the lab outside of lab time but remember that you need to attend lab to get your work checked in order to get full credit.
- Always label plots: x-axis, y-axis, title (and legend if needed). A plot is meaningless if it is not labeled!
- Always use script and/or function files. Not only does it save you time from reentering commands, but it also helps you (and others) organize and debug.
- Feel free to discuss the labs with your classmates but work should be individually completed.

3. Pre-lab: MATLAB

A. Comment

Briefly comment every line of the following (no more than one line on the pre-lab sheet). Comment what each line is doing or write down the expected outcome. Tip: the **help** function might be useful.

1. a = zeros(1,5)
2. b = [3:1:5]
3. c = [1:-3.5:-9]
4. d = [1.4, 2.3; 5.1, 7.8]
5. e = d(1,2)
6. f = d(:,2)
7. d.^2
8. comp = 3+4i
9. real(comp)
10. abs(comp)
11. 3 ~= 5

B. Plot graphs

Variables x and y are given below. Plot the followings. You could copy and paste your plots into Microsoft Word (Select Edit→Copy Figure). Submit the plots with your codes.

** Don't forget to label axes and title using xlabel, ylabel, and title function.*

x = [1:5];

y = [3 5 7 6 8];

B.1. Plot y vs x (x: x-axis and y: y-axis) with a default style using **plot** function. You are expected to see a blue solid line.

B.2.Plot y vs x (x: x-axis and y is y-axis) with a default style using **stem** function. You are expected to see five blue circle markers and five blue solid lines.

B.3. Briefly comment (no more than one line) a difference between plot and stem function.

B.4. Plot y vs x (x: x-axis and y is y-axis) with a custom style of "**red plus '+'**" using **stem** function. You are expected to see five red plus markers and five red solid lines when you copy the plot to Word.

C. Function

Given an input signal x = [3 8 6 2 7], write a function that clips the elements larger than 5 to 5, and store it in y and outputs y. In other words, the output y should be [3 5 5 2 5]. Write the function in a script and save it as "clip.m". Test the function by running it. Submit your code.

```
function y = clip(x)
  Your code goes here
end
```

4. Pre-Lab: Python

A. Briefly comment every line of the following (no more than one line on the pre-lab sheet). Comment what each line is doing or write down the expected outcome. Assume we imported NumPy module by: **import numpy**

1. a = numpy.zeros(3)
2. b = a.shape
3. numpy.absolute([-5.2, -3])
4. c = numpy.arange(5,8,1)
5. d = numpy.arange(1,-16.5,-3.5)
6. e = numpy.array([[1.4, 2.3], [5.1, 7.8]])
7. f = e[0][0]
8. g = e[0][:]
9. comp = 5+12j
10. numpy.imag(comp)
11. numpy.absolute(comp)
12. 5 == 5

B. Variables x and y are given below. Plot the following graphs using **matplotlib.pyplot module**. To assist you with the lab, the template is given. Save the plot and insert it into Microsoft Word. Submit the plots with your codes. ** Don't forget to label axes and title using xlabel, ylabel, and title function.*

```
# Import module
import matplotlib.pyplot as plt
import numpy

# Define variables
x = numpy.array([1,2,3,4,5])
y = numpy.array([2,2,2,5,5])

# Plot
Your code starts from here
```

B.1. Plot y vs x (x: x-axis and y: y-axis) with a default style using **plot** function in matplotlib.pyplot.

B.2. Plot y vs x (x: x-axis and y is y-axis) with a default style using **stem** function in matplotlib.pyplot.

B.3. Plot y vs x (x: x-axis and y is y-axis) with a custom style of "**yellow, circle, dashed**" using **plot** function.

C. Given an input signal x = numpy.array([3,8,6,2,7]), write a function that clips the elements **less** than 5 to 5, and save it in y and returns y. In other words, the output y should be [5 8 6 5 7]. Save the function as "clip" and save the file as "prelab1_c.py". Test the function by running it. Submit your code.

```
def clip(x):
  Your code goes here
  return y
```

5. In-Lab

Some problems ask you to write down your answer. Some ask you to save your code in a file. Some ask you to copy and paste the result into Word. To avoid doing the problem again, make sure you know what to show the TA for each problem. You must do the problems sequentially, starting from the first problem. Unless mentioned, complete all problems ***both MATLAB and Python****. It is recommended to complete all problems in MATLAB and then do the same task by Python.*

A. Concatenation

Write a function that takes a 2-D input matrix and outputs a row vector that is a concatenation of each column vector in the input matrix. For example, if the input is [1 2 ; 3 4], the output should be [1 3 2 4]. Make sure it works with matrix of any size.

B. Impulse signal

A discrete-time signal can be represented by a vector. In this problem, we will generate a discrete-time impulse. Create a row vector of length 100. Make the 5th element value one, and everywhere else value zero. Make a graph using stem and show the graph.

C. Generate signal

Generate the following sample signals with 100 Hz sampling rates for 10s:

C.1. $\cos(2\pi t)$

C.2. $e^{-0.9t}\cos(2\pi t)$

D. Subplot

Plot all the signals in problem C in a single window (figure) with the command **subplot for MATLAB** and **subplot in matplotlib.pyplot module for Python**. Use **stem** for C.1. and **plot** for C.2. Label axes and title for each subplot. Copy the figure to Word.

E. my_echo

Given a discrete-time signal contained in a vector **x**, write a function **my_echo(x,d,a)** to generate a vector that includes the original signal plus an echo, which occurs **d** seconds after the end of the original signal. The echo is just like the original signal but has a smaller amplitude and duration (echo amplitude = **a** * signal amplitude, echo duration = half of signal duration). The sampling frequency is fixed as 100 Hz, and the first sample occurs at time t = 0. You might need to concatenate the original signal and/or its delayed copy with zeros to avoid length mismatches. Remember that echoes are additive. Then execute x = 1:15000 (Python: numpy.arange(1,15001)), d = 50 seconds, and a = 0.5, and plot the result (x-axis: time vs y-axis: signal value).

** Use [ ] for Matlab and numpy.concatenate for Python.*