Toronto Metropolitan University
Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering
& Architectural Science

ELE 532
Signals & Systems I

Laboratory Assignment 1
Working with MATLAB Functions, Visualization of Signals, and Signals Properties

Objective

In Lab 1 you will work with simple MATLAB functions and explore key MATLAB features such as *algorithm vectorization* and *array indexing*. You will use MATLAB to operate on signals and will visualize the effects of these operations.

Preparation

Read *Lathi, Section B.7 MATLAB: Elementary Operations*, pp. 42-53 and *Lathi, Chapter 1, Sections 1.1-1.5*, pp. 64-95. Note that MATLAB has an extensive help facility for every command available, and you can obtain help on the command you want to use by typing:

`>> help` *command*

at the MATLAB command line. You can also access on-line MATLAB manuals and documentation either by selecting *Help/Product Help* from the menu bar or by typing:

```
>> doc
```

at the MATLAB command line.

Lab Assignment

For this assignment, you will be modifying the code in the exercises such that the *Anonymous Function* is used to generate the mathematical functions.

A. Anonymous functions and plotting continuous functions

Problem A.1 [1 Mark] Section 1.11-1 (Lathi, page 126). Generate and plot Figures 1.46 and 1.47 on page 127.

Problem A.2 [1 Mark] Plot the function e^{-t} for five points [-2,-1,0,1,2] using `t=[-2:2]`.

Problem A.3 [1 Mark] Compare your result in Problem A.2 with Figure 1.46 in Problem A.1.

B. Time shifting and time scaling

Problem B.1 [1 Mark] Section 1.11-2 (Lathi, page 128). Generate and plot $p(t)$ as shown in Figure 1.50 on page 129.

Problem B.2 [2 Marks] Use $p(t)$ to generate and plot functions $r(t) = tp(t)$ and $n(t) = r(t) + r(-t+2)$.

Problem B.3 [2 Marks] Plot the following two signals: $n_1(t) = n(\frac{1}{2}t), n_2(t) = n_1(t + \frac{1}{2})$.

Problem B.4 [2 Marks] Plot the following two signals: $n_3(t) = n(t + \frac{1}{4}), n_4(t) = n_3(\frac{1}{2}t)$.

Problem B.5 [2 Marks] Compare $n_4(t)$ and $n_2(t)$ and explain their differences and/or similarities.

C. Visualizing operations on the independent variable and algorithm vectorization

Note that if **f** and **g** are two inline functions of **t**, then **f(t).*g(t)** is no more an inline function, but it is a vector with values of **f(t)**$\times$ **g(t)**.

Problem C.1 [1 Marks] Section 1.11-33 (Lathi, page 130). Follow the steps in this exercise, but to instead generate $g(t) = f(t)u(t)$ where $f(t) = e^{-2t}\cos(4\pi t)$.

Problem C.2 [1 Marks] Using $g(t)$ in C.1, generate and plot $s(t) = g(t+1) = e^{-2}e^{-2t}\cos(4\pi t)u(t+1)$ for `t=[-2:0.01:4]`.

Problem C.3 [3 Marks] Plot $s_\alpha(t) = e^{-2}e^{-\alpha t}\cos(4\pi t)u(t)$ for $\alpha = [1, 3, 5, 7]$ in one figure for `t=[0:0.01:4]`. For this plot you can use the `for` command for a loop structure (to learn more about this command type `help for` at the MATLAB prompt). Also try to use matrix and vector operations to generate and plot the desired functions by following the steps of Section B.7-6 (Lathi, page 49).

Problem C.4 [1 Mark] Determine the size of the matrix you generated in C.3.

D. Array indexing

Note: The Matlab data file `ELE532_Lab1_Data.mat` contains all data arrays (arrays **A**, **B** and **x_audio**) referenced in this section. You can download the data file from the course homepage on Blackboard. Alternatively, if you are using MATLAB on any of departmental computers, typing `load ELE532_Lab1_Data` at the MATLAB prompt will load all data arrays into your current MATLAB workspace.

Problem D.1 [1 Marks] Let A be a 5×4 matrix array with real-valued elements:

$$\mathbf{A} = \begin{bmatrix} 0.5377 & -1.3077 & -1.3499 & -0.2050 \\ 1.8339 & -0.4336 & 3.0349 & -0.1241 \\ -2.2588 & 0.3426 & 0.7254 & 1.4897 \\ 0.8622 & 3.5784 & -0.0631 & 1.4090 \\ 0.3188 & 2.7694 & 0.7147 & 1.4172 \end{bmatrix} \quad (1)$$

For the matrix **A** in Equation (1) implement the following operations:

(a) $\mathbf{A}(:)$
(b) $\mathbf{A}([\ 2\ \ 4\ \ 7\ ])$
(c) $[\ \mathbf{A} >= 0.2\ ]$
(d) $\mathbf{A}([\ \mathbf{A} >= 0.2\ ])$
(e) $\mathbf{A}([\ \mathbf{A} >= 0.2\ ]) = 0$

Describe the outcome of each operation stated in parts (a)–(e).

Problem D.2 [1 Mark] Let $\mathbf{B}$ be a 1024×100 data matrix representing 100 blocks of non-overlapping 1024-element input samples from a particular data source.

(a) Write a simple MATLAB program using two nested `for` loops that will set all elements of the data matrix $\mathbf{B}$ with magnitude values below 0.01 to zero:

$$\mathbf{B}(i, j) = 0, \quad \text{if } |\mathbf{B}(i, j)| < 0.01, \tag{2}$$

where $\mathbf{B}(i, j)$ is element of the data matrix $\mathbf{B}$ in *i-th* row and *j-th* column.

(b) Repeat part (a) using MATLAB's indexing features as described in Problem D.1.

(c) Use the MATLAB commands `tic` and `toc` to compare the execution time of the code you wrote in parts (a) and (b).

Problem D.3 [1 Mark] Let **x_audio** be a 20,000 sample-long row vector representing 2.5 sec of an audio signal sampled at 8 kHz. A simple data compression algorithm can be implemented by setting all elements of the data array **x_audio** with magnitude values below a threshold to zero. ***Note:*** *The actual compression algorithm will code the zero-valued samples more efficiently thus achieving a certain degree of compression. In this exercise, however, we only want to investigate the compressibility of the data array* ***x_audio*** *as measured by the number of samples with magnitude values below the threshold and the resulting sound quality as a function of the threshold.*

Write such a data compression algorithm and listen to the processed audio file. You may want to consider the following points:

- Do not work directly on the data array **x_audio**; otherwise after each process you will need to reload the data file. Instead, copy the data array **x_audio** to another working array and process that array.
- Devise a simple method of counting the number of elements of data array that are set to zero.
- You can listen to an audio array by using the MATLAB command `sound`. For example, if you want to listen to the original, unprocessed data array **x_audio**, issue the command `sound(x_audio,8000)` at the MATLAB prompt.