

## LAB 1: Working with MATLAB, Visualization of Signals

- **Report Due:** Sunday,
- **NOTE:** You must submit ONE report per group and it must be submitted online through the course D2L page.

### 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, 3rd Ed., Section B.7: MATLAB Elementary Operations*, pp. 42–53 and *Lathi, 3rd Ed., Chapter I*, sections 1–5, pp. 64–94. Note that MATLAB has an extensive on-line help facility; you can obtain help on the MATLAB command you want to use by typing

```
>> help command
```

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

```
>> doc
```

at the MATLAB prompt.

### Lab Assignment

#### A. Inline functions and plotting continuous functions

**Problem A.1 [0.5 Marks]** Section 1.11-1 Anonymous Functions, *Lathi, 3rd Ed., page 126*. Generate and plot the graphs as shown in Figures 1.46 and 1.47 on page 127.

**Problem A.2 [1 Mark]** Plot the function  $e^{-t}$  for  $t$  taking on integer values contained in  $-2 \leq t \leq 2$ ; you can generate these values using the MATLAB command `tt=[-2:2]`.

**Problem A.3 [0.5 Marks]** Compare the results of 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, 3rd Ed., page 128*. Generate and plot  $p(t)$  as shown in Figure 1.50 on page 129.

**Problem B.2 [2 Marks]** Use  $p(t)$  defined in Problem B.1 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  $n_2(t)$ ; explain any observed differences and/or similarities.

### C. Visualizing operations on the independent variable and algorithm vectorization

**Problem C.1 [0.5 Marks]** Section 1.11-3, *Lathi, 3rd Ed.*, page 130. Follow the steps, but instead, generate  $g(t) = f(t)u(t)$  where  $f(t) = e^{-2t} \cos 4\pi t$ .

**Problem C.2 [0.5 Marks]** Using  $g(t)$  as described in Problem C.1, generate and plot  $s(t) = g(t + 1)$  for  $t = [0 : 0.01 : 4]$ .

**Problem C.3 [0.5 Marks]** Plot  $s_\alpha(t) = e^{-2}e^{-\alpha t} \cos(4\pi t)u(t)$  for  $\alpha \in \{1, 3, 5, 7\}$  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, 3rd Ed.*, page 49.

**Problem C.4 [0.5 Marks]** Determine the size of the matrix  $s(t)$  generated in Problem 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 webpage on D2L. 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 [0.5 Marks]** Let **A** be a  $5 \times 4$  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 (??) implement the following operations:

- $\mathbf{A}(:)$
- $\mathbf{A}([2 \ 4 \ 7])$
- $[\mathbf{A} \geq 0.2]$
- $\mathbf{A}([\mathbf{A} \geq 0.2])$
- $\mathbf{A}([\mathbf{A} \geq 0.2]) = 0$

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

**Problem D.2 [1 Mark]** Let **B** be a  $1024 \times 100$  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 **B** with magnitude values below 0.01 to zero:

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

where  $\mathbf{B}(i, j)$  is element of the data matrix **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.