**ENSC 180 – Introduction to Engineering Analysis Tools**

**Assignment #1: Basic MATLAB Syntax**

## Background

MATLAB’s ease of use can be attributed to several things:

1. A vast library of built-in functions accompanied by comprehensive documentation.
2. High-level data structures and handling of data types.
3. It’s an interpreted programming language, as opposed to a compiled language like C++.

The advantage of interpreted languages comes from their ability to execute code “on the fly”. Unlike a C++ program, a MATLAB script is simply a sequence of instructions that could very well have been typed by the user and executed line-by-line in the MATLAB terminal. A programmer can simply piece together a program by experimenting in the terminal and examining the outputs after each piece of code (the visualization and plotting functions in MATLAB makes the task of examining these outputs even easier in certain cases). Functional lines of code can then be dragged from the “Command History” pane into a MATLAB script or function.

The disadvantage of interpreted languages is slow performance. In MATLAB’s case, this issue can be circumvented by taking advantage of its built-in functions, many of which are highly optimized. Vector and matrix operations are also well-optimized, and thus, vectorization is preferred (see [[[1]](#footnote-1)]) instead of loop-based, scalar-oriented code.

## Objectives

This assignment is intended to:

1. provide practice with MATLAB syntax.
2. accustom the student with vectorization.
3. bring awareness to several commonly-used built-in MATLAB functions.

Additionally, Question 5 is meant to emphasize MATLAB’s numerical nature (closed-form symbolic functions are not always available in real life).

## Procedure

**Question 1:** Given the vectors x and y, use vector operations (no loops) to compute the vector z whose *i*th element is given by

The cosine function is in radians. You may find the MATLAB function length useful along with the colon operator.

**Question 2:** Without using loops, calculate the sum of the following series

for the first 10,000 terms [[[2]](#footnote-2)]. There are several ways to implement this. You may find functions such as sum, ones, or dot useful. Store the result in an array seriesSum.

**Question 3:** An M-by-M matrix X is given. Without using loops, extract values from matrix X to create the following:

i) matrix A – composed of all values in odd columns AND odd rows of X, e.g.:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

ii) matrix B – composed of all entries of X, except for the outside rows and columns, e.g.:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

iii) matrix C – composed of diagonals surrounding the middle diagonal of matrix X, e.g.:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

The 1st row of C should contain values from the upper diagonal. The 2nd row should contain values from the lower diagonal.

**Question 4:** The voltage across a capacitor is measured from *t* = 0 to 11ms. The voltage measurements are given in vector V and the corresponding times at which they are taken are given in vector t. Write code to create a line plot of voltage vs. time with the following requirements:

i) The x-axis must be labeled “time (ms)”.  
ii) The y-axis must be labeled “voltage (V)”.  
iii) The title must be “Capacitor Voltage”.  
iv) The x-axis must only span the time of the measurements, i.e., from 0 to 11ms.  
v) The numbers on the axes must have font size 14 and the labels/title must have font size 18.

**Question 5:** Consider the voltage data from the previous question. It would be convenient to be able to call V(t\_i) in MATLAB to get the voltage at some time t\_i. Unfortunately, this would not work since V is not a function, but rather, a vector full of measurements and t\_i is a time, not an index. One solution is to fit a function to the measurements in V and then call that function at any time t\_i. This will be covered later in the course, but for now we will implement a far simpler solution. Write a MATLAB function with the following description:

function values = getValues(y, t, t\_i)

* y is a vector of values sampled at times t.
* t\_i is a vector of times at which we want to fetch the corresponding values of y.
* values is a vector of the desired values from y at times t\_i.

This function should be stored in a file named “getValues.m” and should perform the following steps:

1. Pre-allocate space to the output vector values. It should be initialized to be the same size as t\_i with every element as 0. Use the MATLAB functions zeros and either length or size and to perform this step.
2. For each element in vector t\_i (for-loop permitted here):
   1. Determine the index at which the current element of t\_i is located in vector t and store this index in idx. In other words, t(idx) should return the current element of t\_i. If the current element of t\_i is not found in vector t, then t(idx) should yield the closest available sample time after t\_i. For example:

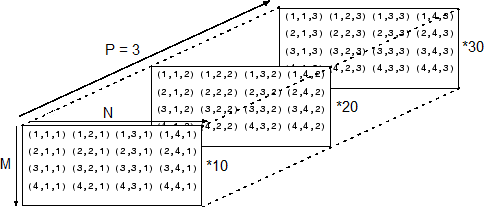
t\_i = 0.23 for  
t = 0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000  
would instead yield sample time 0.3000.

* 1. Using idx, fetch the value corresponding to the time in the current element of t\_i from vector y and store it in vector values.

You will need to use logical operators as well as the MATLAB function find (read its documentation carefully). Implementing this function should not take more than 5 or 6 lines of code.

Call the function with t\_i = [0.45, 2.003, 7] and the data V and t from Question 3. You can quickly check that your output is correct using the data cursor and arrow keys on the plot from Question 3. Store your output in a vector values.

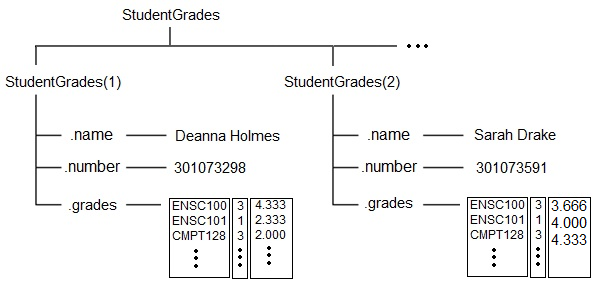
**Question 6:** You are given an M-by-N-by-P matrix Y. Without using loops, scale Y such that every element in the *i*th page is scaled by 10*i* for *i* =1,2,…,P. For example, given an M-by-N-by-3 matrix, you would apply the following scaling [[[3]](#footnote-3)]:



You will find the MATLAB functions reshape and repmat useful. Store your result in a matrix Yscaled.

**Hint:** To accomplish this task without loops, you need to generate a matrix such that an element-wise multiplication between this matrix and Y results in the desired scaling.

**Question 7:** You are given a structure called StudentGrades that is organized as follows:



The fields are:

* name – a string specifying the student’s name.
* number – a scalar specifying the student’s number.
* grades – a 1-by-3 cell array where:
  + the 1st cell is a 9-by-7 char matrix specifying the course names.
  + the 2nd cell is a 9-by-1 vector with the corresponding credits for each course of the 1st cell.
  + the 3rd cell is a 9-by-1 vector with the corresponding grade that the student earned for each course of the 1st cell. These are in the form of a 4.333 grade scale (e.g., 4.333 is an A+, 4.000 is an A, 3.666 is an A-, 3.333 is a B+, etc.).

Your task is to calculate each student’s cumulative GPA (CGPA). For a set of courses, let the grades for the course be denoted by and the number of credits of the course be denoted by , then CGPA can be calculated as:

In other words, this is simply a weighted average of the grades and the credits corresponding to each course. Calculate each student’s CGPA and store it in a new field called cgpa. For example, calling StudentGrades(1).cgpa would give the 1st student’s CGPA, calling StudentGrades(2).cgpa would give the 2nd student’s CGPA, etc.

After completing each questions, run the entire script Basic\_Syntax.m using F5. The entire script should run without error. Do not modify any code in the “Save Outputs” section.

1. [1] http://www.mathworks.com/help/matlab/matlab\_prog/vectorization.html [↑](#footnote-ref-1)
2. [] B. Hahn & D. Valentine. *Essential MATLAB for Engineers and Scientists 3e*, Oxford: Newnes, 2007. (Modified slightly to fix error in the original problem). [↑](#footnote-ref-2)
3. [] http://www.mathworks.com/help/matlab/math/multidimensional-arrays.html [↑](#footnote-ref-3)