**ECE 1111: Engineering Computation I**

**Laboratory No. 4: Vector and Matrix Manipulations**

**Goals:** Help you develop an understanding of how to do basic engineering mathematics using data-driven programming techniques.

**Deliverables:**

1. *Source Code:* Do all your work in this directory:

/data/courses/ece\_1111/current/labs/lab\_04/<lastname\_firstname>

You must comment your code, following the guidelines and templates provided in class. Create directories *p01*, *p02*, and *p03* for the corresponding parts below.

1. *Check Off:* You will arrange a meeting with your TA during the lab session. The TA will ask you to make a small modification to your print statement, compile and run your program, and manipulate the output file using command line tools.

**Description:**

In this lab, we will integrate several concepts we have discussed in class and demonstrated during the lectures or labs. The specific task is to implement a program that reads a vector or matrix from a file and does some simple linear algebra computations.

1. (p01) Vector-Vector Multiplication

Create a Python program that does the following:

**vmult <filename> constant**

This program should read the contents of a file specified from the command line and perform the following computation:

Where is specified from the command line as the second argument. Create a text file that contains the two vectors:

ece\_000\_[1]: more x.txt

1.0, 0.0, 0.0

0.0, 1.0, 0.0

The interface to your program should be:

vmult x.txt 2.0

The output of your program should be:

x = [...print the values of the first vector ...]

y = [...print the values of the second vector ...]

z = ...print the value of the above computation...

If the files fail to open properly, print an error message and exit. If the vectors are not the same lengths, print an error message and exit. In short, make sure your program catches as many errors as possible.

Print the result using four decimal places of precision. Demonstrate that your code produces the same result as what you get in Excel. Note that the input data can be any length.

1. (p02) Matrix-Vector Multiplication

Write a program that loads a vector and a matrix from a text file and multiplies them. Here is an example of the interface you must support:

nedc\_000\_[1]: more x.txt

1.0 0.0 0.0

0.0 1.0 0.0

0.0 0.0 1.0

nedc\_000\_[1]: more y.txt

1.0, 0.0, 0.0

The interface to your program should be:

vmmult x.txt y.txt

The output of your program should be a nicely formatted display of the inputs and the outputs, and should compute:

where is a matrix and is a vector. The result, , is a vector.

Note that the vector and matrix can be any length or dimension. Also note that matrix multiplication of an identity matrix with a vector produces the input vector, so you can use the above example for the initial set of test cases to debug your code. Make sure that you do proper error checking so that a result is produced only if the data has the proper dimensions.

1. (p03) Matrix-Matrix Multiplication

Write a program that loads two matrices from a text file and multiplies them. Here is an example of the interface you must support:

nedc\_000\_[1]: more x.txt

1.0 0.0 0.0

0.0 1.0 0.0

0.0 0.0 1.0

nedc\_000\_[1]: more y.txt

1.0 0.0 0.0

0.0 1.0 0.0

0.0 0.0 1.0

The interface to your program should be:

mmmult x.txt y.txt

The output of your program should be a nicely formatted display of the inputs and the outputs, and should compute:

where is a matrix, is a matrix and is a matrix.

Note that your program should work for all types of matrices that can be multiplied. They need not be rectangular or have the same dimensions. Make sure that you do proper error checking so that a result is produced only if the data has the proper dimensions.

You will probably find this tutorial helpful: <https://www.geeksforgeeks.org/python-program-add-two-matrices/>.

**Summary:**

Vectors, or an array as we often refer to it, and matrices (technically simply a multidimensional array) are fundamental to the way we write code to implement engineering mathematics. In this lab, we cover the basics of how these types can be used in simple computations.

This lab demonstrates several key concepts in programming: (1) data-driven programming: reading data from a file rather than hardcoding it in your program; (2) dynamic memory allocation: sizing your program to the data so you don't run out of memory; and (3) verification: using MATLAB or Excel to verify your results.