**LINEAR ALGEBRA**

Laboratory No. # 3

**VECTOR OPERATIONS**

|  |
| --- |
|  |

Score

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CRITERIA** | **Exceeds Expectations** | **Meets Expectations** | **Needs Improvement** | **Unsatisfactory** |
| Functionality  (60 points) |  |  |  |  |
| Completeness  (20 points) |  |  |  |  |
| Structure  (20 points) |  |  |  |  |

**Remarks:**

*Submitted by:*

**Ugot, Aaron Paul M.**

**Monday 7:00–10:00/ 58013**

*Submitted to*

**Engr. Maria Rizette Sayo**

Facilitator

*Date Performed:*

**15-09-2023**

*Date Submitted*

**16-09-2023**

**Objective**

1. Recall knowledge on vector operations while being familiar with new operations such as products.
2. Visualize vector operations.
3. Perform vector operations using Python.

**Algorithm**

1. Type the main title of this activity as "Vector Operations”
2. On your GitHub, create a repository name Linear Algebra 58019
3. On your Colab, name your activity as Python Exercise 3.ipynb and save a copy to your GitHub repository

**Discussion**

We have dealt with some of the vector operations in the last module, now we will dwell into more operations. In this laboratory, we will tackle addition, multiplication, division and the inner product of a vector.

**Coding Activity 3**

*Vector Addition*

We have encountered vector before especially with your last activity. Vector addition is simply the element-wise addition of the scalar values of the vectors. Let's take the following vectors as a sample:

A black text on a white background

Description automatically generated

So if do a vector addition of these two vectors we'll get:

A black lines with numbers

Description automatically generated with medium confidence

We can programmatically solve this using np.add() or simply using +.

|  |
| --- |
| A = np.array([1,2,0])  B = np.array([3,1,-2])  C = np.array([0,0,1])  A+B+C  np.add(A,np.add(B,C))  np.sum(B) |

*Vector Subtraction*

Vector subtraction is similar to your vector addition but you would need to scale the second vector using a negative scalar that is usually -1. So if we subtract vector B from vector A we get:

A black text on a white background

Description automatically generated

In Python, this can be achieved by using np.subtract() or –

|  |
| --- |
| import numpy as np  A = np.array([1,2,0])  B = np.array([3,1,-2])  C = np.array([0,0,1])  np.subtract(A,B)  A-B |

*Vector Multiplication*

Vector multiplication, like addition does its operations element-wise. So basic vector multiplication can be achieved by multiplying the elements or the scalars of the vectors individually. So:

A black text with numbers and a square

Description automatically generated with medium confidence

We can implement this in code by using np.multiply() or simply operating with \*.

|  |
| --- |
| import numpy as np  A = np.array([1,2,0])  B = np.array([3,1,-2])  C = np.array([0,0,1])  np.multiply(A,B)  A\*B |

*Vector Division*

Dividing a vector by another is uncommon, but if the situation specifies that each elements or scalars of the vector would be divided individually we can perform this by:

A black text on a white background

Description automatically generated

Take note that the notation we used here is ./ for element-wise division, this notation is based in a MATLab notation for element-wise division. This can be achieved in Python using np.divide() or /.

|  |
| --- |
| import numpy as np  A = np.array([1,2,0])  B = np.array([3,1,-2])  C = np.array([0,0,1])  np.divide(A,B)  A/B  C = np.array([3.9,1.8,7.7])  D = np.array([1,1,1])  C//D  np.array(A/B,dtype=int) |