Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 2

Introduction to Vectors and Numpy

Submitted by:
Marquita, Bhea Marrianne J.

Instructor:
Engr. Dylan Josh D. Lopez

March 13, 2021

I. Objectives

This laboratory activity aims for the researcher to be familiar with Python libraries for numerical and scientific programming. This activity also aims to visualize vectors through Python programming. This laboratory activity seeks to create a program wherein the codes will perform the following operations in any combination and any number of trials: addition, subtraction, multiplication, division, squaring, square root, and summation.

II. Methods

The researcher reviewed about creating vectors using numpy and matplotlib.

- **Vectors**

  It is an array of numerical values or scalars that would represent any feature space. Feature spaces or simply dimensions or the parameters of an equation or function. A vector can contain several parameters or values that would describe what an object is doing. [1]

- **NumPy or Numerical Python**

  In Python, numpy holds most of the equations and functionalities for doing computations and advanced scientific computations. [1]

- **Scalars**

  Scalars are numerical entities that are represented by a single value. [1] Scaling or scalar multiplication takes a scalar value and performs multiplication with a vector. [1]

- **Shapes**

  The shape of a vector tells us how many rows and columns are there. [1]

- **MatPlotLib**

  MatPlotLib or MATLab Plotting library is Python's take on MATLabs plotting feature. MatPlotLib can be used vastly, from graphing values to visualizing several dimensions of data. [1] Matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. [2]

The researcher also created a program wherein the objectives are explicitly met: a program that will perform the following operations in any combination and any number of trials: addition, subtraction, multiplication, division, squaring, square root, and summation.

III. Results

For the code reference, please visit https://github.com/bheanne/LinearAlgebra/blob/main/LAB%20REPORT%202/Marq_Lab2.ipynb

```python
import numpy as np
import matplotlib.pyplot as plt

Vector1 = np.array([12,30])
Vector2 = np.array([11,21])
Vector3 = np.array([1,4])
Vector4 = np.array([-1,-25])
Vector5 = np.array([-12,-24])
```

Figure 1.0 Declaring Library and Values

In Figure 1.0, we can see that the researcher imported numpy and matplotlib python libraries in the program and called it np and plt. The researcher also declared the five vectors' values with two elements that are necessary for the program.

```python
##A Addition & Subtraction
Addition = np.add(Vector1, Vector2)
Subtraction = np.subtract(Vector1, Vector3)
Combi1 = np.subtract(np.add(Vector1, Vector2), Vector3)
print("\n⊹˙✶°••°✶˙⊹•°✶˙⊹•°✶˙⊹•°✶˙⊹•°✶˙⊹★⊹˙✶°•⊹˙✶°•⊹˙✶°•⊹˙✶°•⊹˙✶°••°✶˙⊹")
print("\nAdding Vector1 & Vector2 is equal to", Addition)
print("\nSubtracting Vector1 & Vector3 is equal to", Subtraction)
print("\nSubtracting", Addition, "to Vector3 is equal to", Combi1)
```

Figure 2.0 Addition and Subtraction

The researcher first created examples for addition and subtraction, then made a combination of both operations.

```
##B Multiplication & Division
Multiplication = np.multiply(Vector3, Vector5)
Division = np.divide(Vector3, Vector4)
Combi2 = np.multiply(np.divide(Vector3, Vector4), Vector5)
print("\n⊹˚⋆°••°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹★⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°••°⋆˚⊹")
print("\nMultiplying Vector3 & Vector5 is equal to", Multiplication)
print("\nDividing Vector3 & Vector4 is equal to", Division)
print("\nDividing", Multiplication, "to Vector5 is equal to", Combi2)
```

Figure 3.0 Multiplication and Division

The researcher also created examples of multiplication and division, and a combination of both is also present, as shown in Figure 3.0.

```
##C Square & Square Root
Square = np.square(Vector2)
Square_Root = np.sqrt(Vector2)
Combi3 = np.square(np.sqrt(Vector2))
print("\n⊹˚⋆°••°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹★⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°••°⋆˚⊹")
print("\nThe Square of Vector2 is", Square)
print("\nThe Square Root of Vector2 is", Square_Root)
print("\nThe Square of the Square Root of Vector2 is", Combi3)
```

Figure 4.0 Square and Square Root

A vector was squared, as shown in Figure 4.0, and the researcher also identified its square root. For both operations, the researcher gets the vector's square root and then squares it.

```
##D Summation
Summation = np.sum(Vector2)
print("\n⊹˚⋆°••°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹★⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°••°⋆˚⊹")
print("\nThe Summation of Vector2 is", Summation)

##Visualizing the Data
print("\n⊹˚⋆°••°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹•°⋆˚⊹★⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°•⊹˚⋆°••°⋆˚⊹")
plt.title("Visualization of the Data")
plt.xlim(-50, 100)
plt.ylim(-50, 100)

##Vectors
plt.scatter(Vector1[0], Vector1[1], label='Vector1', c='red')
plt.scatter(Vector2[0], Vector2[1], label='Vector2', c='orange')
plt.scatter(Vector3[0], Vector3[1], label='Vector3', c='yellow')
plt.scatter(Vector4[0], Vector4[1], label='Vector4', c='greenyellow')
plt.scatter(Vector5[0], Vector5[1], label='Vector5', c='lime')
```

Figure 5.0 Summation, Visualization, and Vectors' colors

In Figure 5.0, the researcher generated the operation Summation, and the set of codes for the visualization of the data was also present, as well as the codes for the colors of the vectors.

```
##Results
plt.scatter(Addition[0], Addition[1], label='Addition', c='aquamarine')
plt.scatter(Subtraction[0], Subtraction[1], label='Subtraction', c='aqua')
plt.scatter(Combi1[0], Combi1[1], label ='Add and Subtract', c='deepskyblue')
plt.scatter(Division[0], Division[1], label='Division', c='blue')
plt.scatter(Multiplication[0], Multiplication[1], label='Multiplication', c='blue')
plt.scatter(Combi2[0], Combi2[1], label='Multiply and Divide', c='darkviolet')
plt.scatter(Square[0], Square[1], label='Square', c='fuchsia')
plt.scatter(Square_Root[0], Square_Root[1], label='Square Root', c='magenta')
plt.scatter(Combi3[0], Combi3[1], label='Square and Square Root', c='deeppink')

##Resultant Vector
plt.quiver(Vector1[0], Vector1[1], Vector2[0], Vector2[1], angles='xy', scale_units='xy', scale=1, color='mistyrose')
plt.quiver(5, 21, Vector4[0], Vector4[1], angles='xy', scale_units='xy', scale=1, color='violet')
Vector6 = Vector2 + Vector4
plt.quiver(Vector3[0], Vector3[1], Vector6[0], Vector6[1], angles='xy', scale_units='xy', scale=1, color='skyblue')
```

Figure 6.0 Results and Resultant Vectors

As shown in Figure 6.0, the researcher plotted the operations with colors and some of the vectors.

```
R_mag = np.sqrt(np.sum(Vector1 ** 2 + Vector4 ** 2))  ##Euclidean Distance / Euclidean Norm
rise = Vector6[1]
run = Vector6[0]
slope = rise / run
print("\nThe slope is: ", slope)
print("\n⊹˙⋆°••°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹★⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°••°⋆˙⊹")

plt.grid()
plt.legend()
plt.show()
print("\n⊹˙⋆°••°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹★⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°••°⋆˙⊹")
```

Figure 7.0 Slope and Graph

In Figure 7.0, the slope is created, and the graph for the visualization is called.

```
⊹˙⋆°••°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹★⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°••°⋆˙⊹

Adding Vector1 & Vector2 is equal to [23 51]

Subtracting Vector1 & Vector3 is equal to [11 26]

Subtracting [23 51] to Vector3 is equal to [22 47]

⊹˙⋆°••°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹•°⋆˙⊹★⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°•⊹˙⋆°••°⋆˙⊹
```

Figure 2.1 Output for Addition and Subtraction

The researcher showed the output for the set of codes for the operations, addition, and subtraction in Figure 2.1 and the researcher's fancy border. Additionally, the combination of operations addition and subtraction was also present.

```
Multiplying Vector3 & Vector5 is equal to [-12 -96]

Dividing Vector3 & Vector4 is equal to [-1.   -0.16]

Dividing [-12 -96] to Vector5 is equal to [12.    3.84]
```

Figure 3.1 Output for Multiplication and Division

The product of Vector3 and Vector5 was computed, and the researcher also presented the quotient of Vector3 and Vector4 in Figure 3.1 and the combination of both operations.

```
The Square of Vector2 is [121 441]

The Square Root of Vector2 is [3.31662479 4.58257569]

The Square of the Square Root of Vector2 is [11. 21.]
```

Figure 4.1 Output for Square and Square Root

In Figure 4.1, the researcher showed the output for the operations square and square root, as well as the square of the square root of Vector2.

```
The Summation of Vector2 is 32

The slope is:  -0.4
```

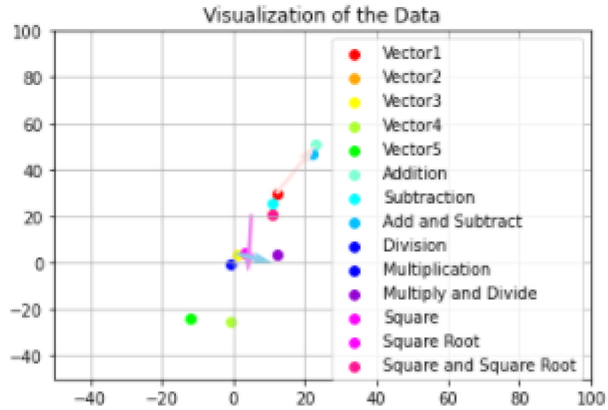

Figure 7.0 Summation, Slope and the Graph

The result of the summation and slope are shown in Figure 7.0. The graph for the visualization of the data inputted can also be seen in Figure 7.0.

IV. Conclusion

Using NumPy from the researcher's past course subjects was very confusing, for learnings from websites aren't enough for the researcher to understand the NumPy library. Still, because of this laboratory activity, new understandings have been encountered by the researcher. Firstly, the researcher can use NumPy to create dimensional arrays as well as advanced mathematical and scientific operations. NumPy can also be used on plotting graphs by using MatPlotLib. Hence, NumPy is a distinctive library to use for arrays. The researcher personally liked the part of the activity where the researcher can pick different colors to represent every element in the graph.

.

References

[1] Dylan Josh Domingo Lopez. (2021) GitHub. [Online]. https://github.com/dyjdlopez/linearAlgebra2021/blob/main/Week%202%20-%20Intro%20to%20Vectors%20and%20Numpy/LinAlg_Lab_2.ipynb

[2] John Hunter, Darren Dale, Eric Firing, and Michael Droettboom. (2021, January) MatPlotLib. [Online]. https://matplotlib.org/stable/tutorials/introductory/pyplot.html