Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 4

Vector Operations

Submitted by:

Parco, Jovian Asher G.

Instructor:

Engr. Dylan Josh D. Lopez

October 26, 2020

I. Objectives

This laboratory activity aims to implement vector operation, visualize vector operation, perform vector operation, and visualize vector using a 3D plot.

II. Methods

In this laboratory activity, the practices consist of creating or declaring vectors, different vector operations using Numpy functions & the built-in function of Python, and plotting 3D vector using Matplotlib.

III. Results

The complete repository for the activity is available at Github (https://bit.ly/34vO8ys). The lab activity consists of 3 parts; the first part is solving a Euclidian distance or the Euclidean Norm without using the Numpy linalg.norm() function, the second part is solving two vector's inner product with using the Numpy inner() function, or @ operator, last is solving vector operation using the given formula and visualizing it using 3D plot.

```
def magnitude(array):
    mag = (sum(array*array))**(1/2)
    print(mag)
```

Figure 1 Function for Part 1

For the first part of the lab activity, the researchers create a function (Figure 1) with the name "magnitude" with one parameter as an array; in the second line, the researcher created an operation that will simulate a Euclidean Norm by using " $||\text{mag}|| = \sqrt{(\text{array}_{\text{index0}})^2 + (\text{array}_{\text{index1}})^2 \ldots}$" In the final line, the researcher shows the answer using the print() function.

```
A = np.array([1,2,3,4])
B = np.array([5,6,7,8])
C = np.array([9,10,11,12])
D = np.array([13,14,15,16])
E = np.array([17,18,19,20])
F = np.array([21,22,23,24])
```

Figure 2 Declared Vectors for Part 1

```
magnitude(A)
np.linalg.norm(A)
```

Figure 3 Executable code for Part 1

```
5.477225575051661
5.477225575051661
```

Figure 4 Output for Part 1

The researcher declared six vectors with four elements (Figure 2) and inputted it in the first line of (Figure 3) to use the created function(Figure 1) and in the second line uses the Numpy linalg.norm() function to check if the created function is working correctly (Figure 4).

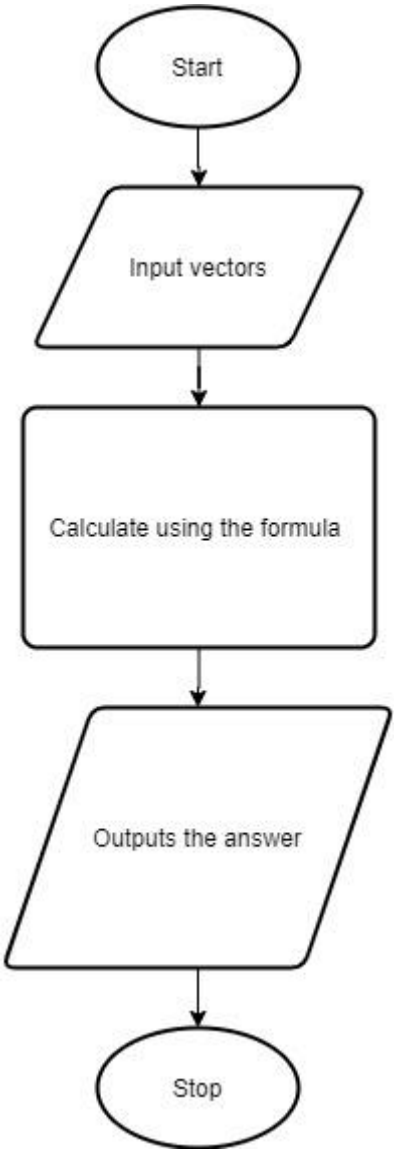

Figure 5 Flowchart for Part 1

The researcher created a simple flowchart of the function for part 1 of the lab activity(Figure 5).

For part two of the lab activity, the researcher replicated the Numpy inner() function or the "@" operator, which calculates the Vector Dot Product / Inner Product.

```
def dot_product(first_array,second_array):
    ans = sum(first_array*second_array)
    print(ans)
```

Figure 6 Function for Part 2

The researcher created a function (Figure 6) named dot_product with two parameters of the first array and the second array; in the second line, the researcher created an operation to solve the vector dot product by using "$\sum_{n=1}^{N} \text{first array} + \text{second array}$" formula and show the answer using the print() function.

```
A = np.array([1,2,3,4,5])
B = np.array([6,7,8,9,10])
C = np.array([11,12,13,14,15])
D = np.array([16,17,18,19,20])
E = np.array([21,22,23,24,25])
```

Figure 7 Declared Vectors for Part 2

```
dot_product(A,B)
print(A@B)
```

Figure 8 Executable code for Part 2

```
130
130
```

Figure 9 Output for Part 2

The researcher declared five vectors with fiive elements (Figure 7) and inputted it in the first line of (Figure 8) to use the created function(Figure 6) and in the second line uses the "@" operator function to check if the created function is working correctly (Figure 9).

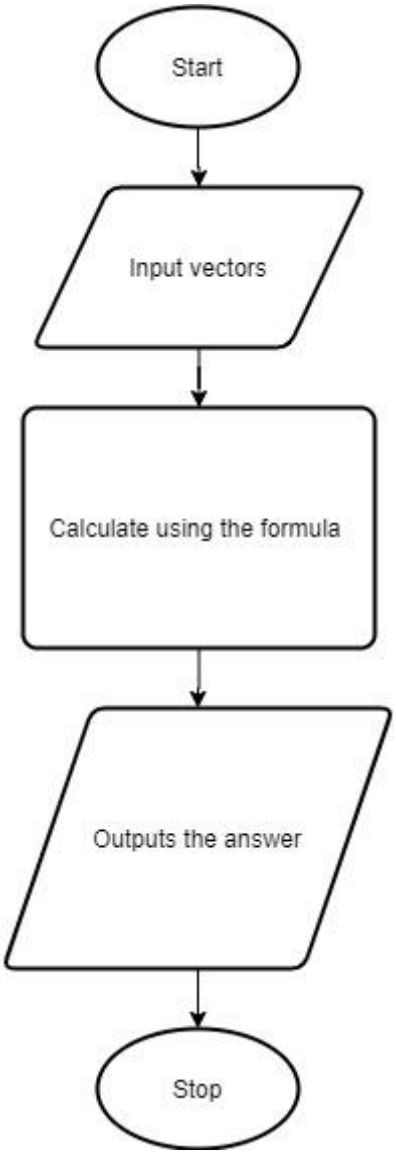

Figure 10 Flowchart for Part 2

The researcher created a simple flowchart of the function for part 2 of the lab activity(Figure 10).

For part 3 of the lab activity, the researcher created a code for the following vector operation "$((A^2 + B^2 + C^2)*(A*(B + A*B)/C))*||A + B + C||$" with the given vectors "$A = \begin{bmatrix} -0.4 \\ 0.3 \\ -0.6 \end{bmatrix}, B = \begin{bmatrix} -0.2 \\ 0.2 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 0.2 \\ 0.1 \\ -0.5 \end{bmatrix}$" and the expected answer [0.34769805, 1.13001866, 0.6953961] and visualize it in a 3D plot.

```
A = np.array([-0.4,0.3,-0.6])
B = np.array([-0.2,0.2,1])
C = np.array([0.2,0.1,-0.5])
```

Figure 11 Declared Vectors for Part 3

```
my_result = (((A*A)+(B*B)+(C*C))*(A*(B+A*B)/C))*np.linalg.norm(A+B+C)
expected_result = (((A@A)+(B@B)+(C@C))*(A*(B+A*B)/C))*np.linalg.norm(A+B+C)
```

Figure 12 Vector Operations for Part 3

```
Result using the formula [0.04193343 0.0794988  0.56260689]
Expected answer [0.34769805 1.13001866 0.6953961 ]
```

Figure 13 Output for Part 3

The researcher inputted the given vectors "$A = \begin{bmatrix} -0.4 \\ 0.3 \\ -0.6 \end{bmatrix}, B = \begin{bmatrix} -0.2 \\ 0.2 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 0.2 \\ 0.1 \\ -0.5 \end{bmatrix}$" into a code (Figure 11), Using the give vector operation "$((A^2 + B^2 + C^2)*(A*(B + A*B)/C))*||A + B + C||$" the researcher inputted into a code (Figure 12) in line one which yields a different answer (Figure 13) but to mitigate the researcher uses the "@" operator or the vector dot product (Figure 12 "second line") instead of square power and which gives the expected result (Figure 13 "second line").

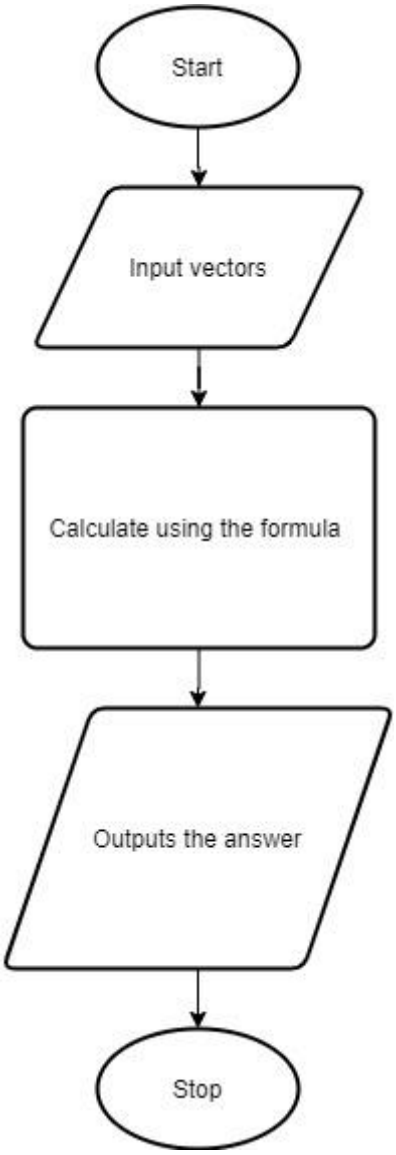

Figure 14 Flowchart for Part 3

The researcher created a simple flowchart of the function for part 3 of the lab activity(Figure 14). Modeling the expected answer and the answer based on the formula in a 3D for the researcher used the Matplotlib.

```
fig = plt.figure()

ax = fig.gca(projection='3d')
ax.set_xlim([0, 1])
ax.set_ylim([0, 1])
ax.set_zlim([0, 1])
ax.text2D(0, 1, "Expected Result", transform=ax.transAxes)
ax.quiver(0, 0, 0, expected_result[0], expected_result[1], expected_result[2], arrow_length_ratio=0.2, colors='r')

plt.show()
```

Figure 15 Plotting Part 3

Plotting the answers for part 3, the researcher created a code (Figure 15) with the first line is creating a blank figure as a plot, the second is setting the created plot as a 3D model, while set_x/y/zlim() function sets the plots limit or the border, while the third from the end set up the title of the plot and sets up the arrow point in the second to the last line and finally show() function to output all the settings for the arrow point and the plot.

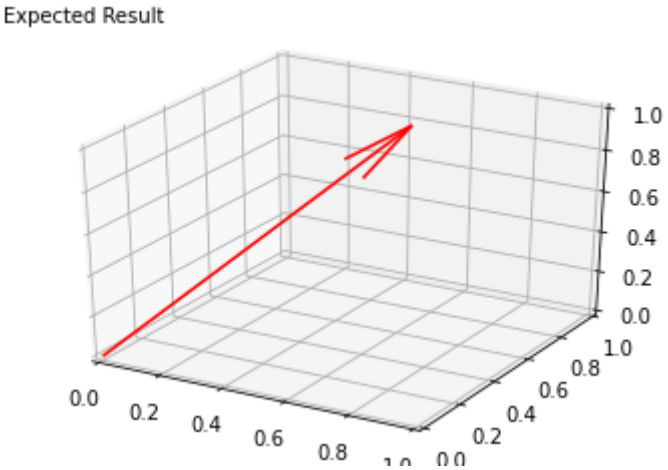

Figure 16 Expected Plot for Part 3

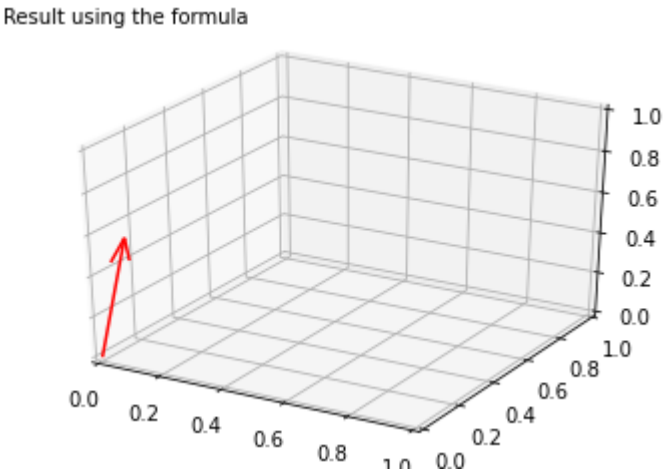

Figure 17 Plot using the Formula for Part 3

The plot from the expected result (Figure 16) is longer than the plot using the formula (Figure 17), which is near the origin.

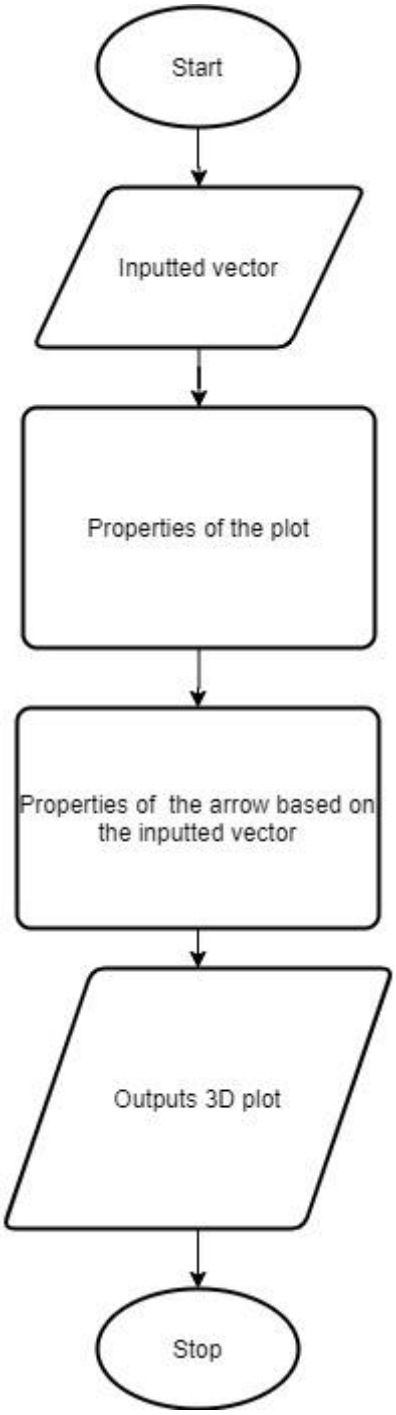

Figure 18 Flowchart for Plotting in Par 3

The researcher created a flowchart (Figure 18) for plotting the expected answer and the answer from the formula.

IV. Conclusion

The researcher used different vector operations build-in and Numpy function; Using the built-in function is complicated. It leads to a greater understanding of mathematical operations in general, plotting 3D vectors used the advance topics to prepare us for the next lab activity, which is “Multi-dimensional Vectors.”

References