Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 4

Vector Products

Submitted by:
Chipongian, John Patrick Ryan J.

Instructor:
Engr. Dylan Josh D. Lopez

November 05, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of vector operations. It aims to familiarize the user from vector operations mostly in new operations such as product.

II. Methods

The practices that were used in this activity are the numpy, matplotlib and math libraries. The functions used in numpy library are np.array() and np.linalg.norm while in the matplotlib the functions with the modules of axes used were plt.figure(), fig.gca(), set_xlim, set_ylim, set_zlim, quiver() and plt.show(). New libraries of function were used which are the math libraries. The function used in math libraries was math.sqrt().

$$||X|| = \sqrt{\sum_{n=1}^{N} x_n^2}$$

Figure 1

In task 1, the task was to compute the modulus of six different vectors and to make the values into one vector. The use of numpy preset functions were not allowed to be used in this task

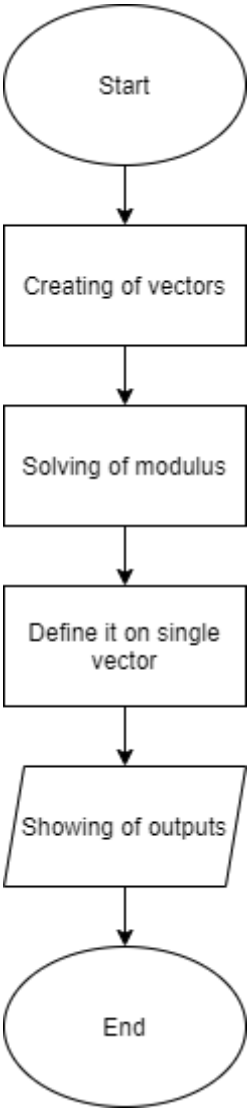

Figure 2

$$A \cdot B = \sum_{n=1}^{N} a_n \times b_n$$
$$whereas: N = len(A) = len(B)$$

Figure 3

In task 2, the task was to find and compute the inner product of five pairs of vectors. The use of numpy present function and the use of "@" operator were not allowed in this task.

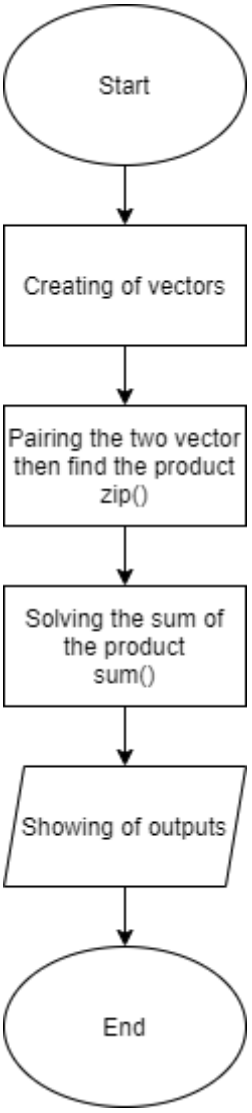

Figure 4

$$((A^2 + B^2 + C^2) \cdot (A * (B + A * B)/C)) * ||A + B + C||$$

$$A = \begin{bmatrix} -0.4 \\ 0.2 \\ -0.6 \end{bmatrix}, B = \begin{bmatrix} -0.2 \\ 0.2 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 0.2 \\ -0.1 \\ -0.5 \end{bmatrix}$$

Figure 5

In task 3, the task was to solve the given vector operations using the given three vectors then to compare it to the expected output and visualize it.

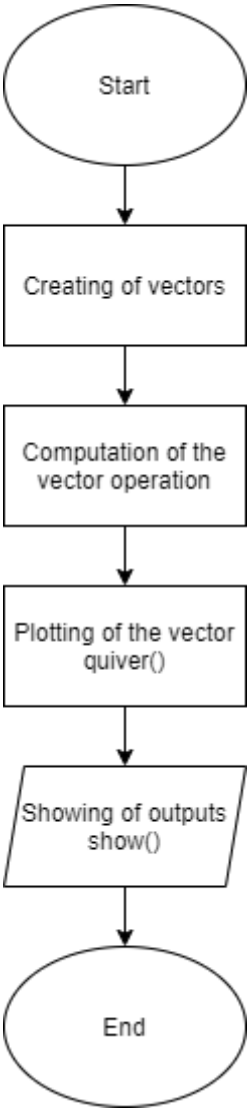
Start
Creating of vectors
Computation of the vector operation
Plotting of the vector quiver()
Showing of outputs show()
End

Figure 6

III. Results

```
from math import sqrt

vect1 = [1,1,1,1,1]
vect2 = [5,3,4,2,7]
vect3 = [9,5,2,1,3]
vect4 = [4,4,2,2,6]
vect5 = [2,1,1,0,4]
vect6 = [5,5,5,0,2]

t = sqrt(vect1[0]**2 + vect1[1]**2 + vect1[2]**2 + vect1[3]**2 + vect1[4]**2)
v = sqrt(vect2[0]**2 + vect2[1]**2 + vect2[2]**2 + vect2[3]**2 + vect2[4]**2)
w = sqrt(vect3[0]**2 + vect3[1]**2 + vect3[2]**2 + vect3[3]**2 + vect3[4]**2)
x = sqrt(vect4[0]**2 + vect4[1]**2 + vect4[2]**2 + vect4[3]**2 + vect4[4]**2)
y = sqrt(vect5[0]**2 + vect5[1]**2 + vect5[2]**2 + vect5[3]**2 + vect5[4]**2)
z = sqrt(vect6[0]**2 + vect6[1]**2 + vect6[2]**2 + vect6[3]**2 + vect6[4]**2)

vectorA = [t,v,w,x,y,z]

for i in range(6):
    print(vectorA[i])
```

Figure 7

Figure 7 are the codes that are used to implement the first task that were given in this activity. The modulus were computed using the six vectors. Firstly, the sum of the squared elements of the vectors were computed then using the function [1] sqrt() from the math library, the square root is computed. The computed modulus is then defined in a single vector and is shown. Figure 8 shows the computed modulus from task 1.

```
2.23606797749979
10.14889156509222
10.954451150103322
8.717797887081348
4.69041575982343
8.888194417315589
```

Figure 8

```
V1a = [2,1,5,7,5]
V1b = [6,5,3,9,1]
V2a = [1,2,2,2,2]
V2b = [3,2,4,5,6]
V3a = [9,8,7,6,6]
V3b = [5,4,6,4,3]
V4a = [6,5,4,5,4]
V4b = [9,7,2,5,1]
V5a = [1,1,1,1,1]
V5b = [2,6,4,5,5]

def dot_prod(V1a,V1b):
        product = [a * b for a, b in zip(V1a, V1b)]
        for i in range(len(product)):
            Sum = sum(product);
            return(Sum)

print(dot_prod(V1a,V1b))
print(dot_prod(V2a,V2b))
print(dot_prod(V3a,V3b))
print(dot_prod(V4a,V4b))
print(dot_prod(V5a,V5b))
```

Figure 9

The codes that is used to implement the problem in task 2 is shown in figure 9. Being not to be allowed to use the numpy preset function and "@" operator, a new function was used in making this task. The created function dot_prod() is created to compute the inner product of the paired vector. Firstly, it compiles the paired vector and multiplies it, element wise. Then, the sum is computed using the sum() function. After that, the inner product of the paired vector is computed. The inner product is shown in figure 10.

```
100
37
161
126
22
```

Figure 10

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

Equation = (((A@A)+(B@B)+(C@C)) * (A*(B+A*B)/C)) * (np.linalg.norm(A + B + C))
print(Equation)
```

Figure 11

```
vector = np.array([0.23741035, -0.4748207, 0.4748207])

fig = plt.figure()
ax = fig.gca(projection = '3d')
ax.set_xlim([0, 0.5])
ax.set_ylim([-0.5, 0.5])
ax.set_zlim([-0.5, 0.5])
ax.quiver(0,0,0,vector[0],vector[1],vector[2],arrow_length_ratio=0.20,colors='g')
plt.show()
```

Figure 12

Figure 11 shows the code in computing vector operation was solved while figure 12 shows the code in visualization of the vector. The algorithm for the vector operation solves the given vector operation using the three vectors given. Then, the computed from it is plotted using matplotlib. Using [2] plt.figure(), a new figure is created that would be where the vector would be plotted. [3] It is plotted in 3d using the axes classes. [4] The set_xlim(), set_ylim(), and set_zlim() sets the limit for x, y and z axes. [4] The quiver() plots the 3D field of the arrow. And lastly, [5] plt.show() is used to display all the figures. Figure 13 shows visualization of the vector.

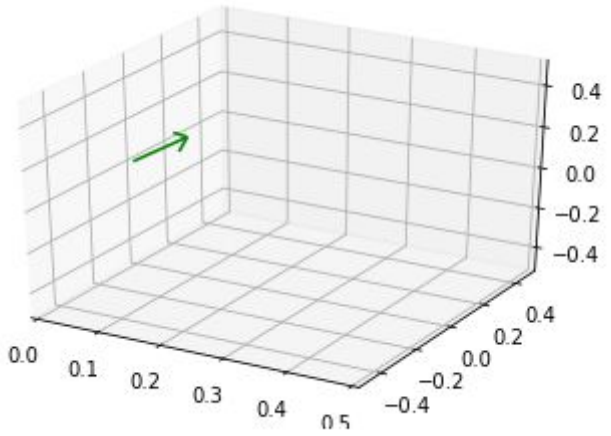

Figure 13

IV. Conclusion

In conclusion, different vector operations can be solved manually but that would produce harder and longer codes and by using the presets and operators that are usable for us, we would be able to produce more efficient and shorter algorithms. And by having the presets available for us, we were able to visualize the vector using 3D plotting.

References

[1] Tutorialspoint.com. 2020. *C Library Function - Sqrt() - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/c_standard_library/c_function_sqrt.htm> [Accessed 4 November 2020].

[2] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Figure() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-figure-in-python/> [Accessed 4 November 2020].

[3] Sodocumentation.net. 2020. *Matplotlib - Three-Dimensional Plots | Matplotlib Tutorial*. [online] Available at: <https://sodocumentation.net/matplotlib/topic/1880/three-dimensional-plots> [Accessed 4 November 2020].

[4] Matplotlib.org. 2020. *Mplot3d API — Matplotlib 2.0.2 Documentation*. [online] Available at: <https://matplotlib.org/mpl_toolkits/mplot3d/api.html> [Accessed 4 November 2020].

[5] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Show() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-show-in-python/> [Accessed 4 November 2020].