Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 5

Multidimensional Vectors

Submitted by:

Parco, Jovian Asher G.

Instructor:

Engr. Dylan Josh D. Lopez

November 6, 2020

I. Objectives

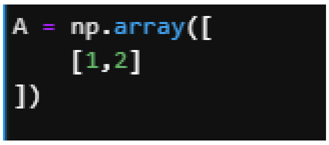
This laboratory activity aims to implement linear combinations in the 3-dimensional plane, plotting vectors in the three dimensions, and vector reshaping using Numpy.

II. Methods

In this laboratory activity, the practices consist of creating or declaring vectors by using the Numpy array() function to create a 3-dimensional matrix, using Matplotlib figure() function to create a blank plane and gca() function to specify the type of plot that will run and lastly the Matplotlib quiver() function to create the arrows and its properties from th created linear combinations.

III. Results

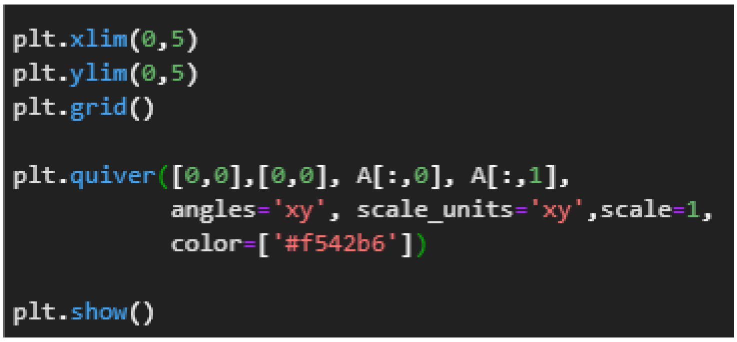
The complete repository for the activity is available at Github (https://bit.ly/36e8sng). The lab activity consists of 2 parts; the first part is creating a 2-dimensional plot with one rank vectors, and the second part is creating a 3-dimensional plot with two rank vectors using Numpy and Matplotlib.



```
A = np.array([
    [1,2]
])
```

Figure 1 Declared Value for Part 1

For the first part, the researcher declared system linear equation (Figure 1) with rank one & one vector, $A = \{x + 2y$ and represent it as a matrix form using the Numpy array() function.



```
plt.xlim(0,5)
plt.ylim(0,5)
plt.grid()

plt.quiver([0,0],[0,0], A[:,0], A[:,1],
           angles='xy', scale_units='xy',scale=1,
           color=['#f542b6'])

plt.show()
```

Figure 2 Code for Part 1

For the code in part 1 (Figure 2), the researcher used the last lab activity “Linear Combination and Vector Spaces” as the reference, for the first third line it is the properties for the plot which use the Matplotlib x/ylim() function to set the bounderies of the plot plane and the grid() function is use to create the horizontal & vertical lines, Matplotlib quiver() function is used to setup the arrows properties and lastly the Matplotlib show() function compile all the properties of the plot plane and the arrow properties.

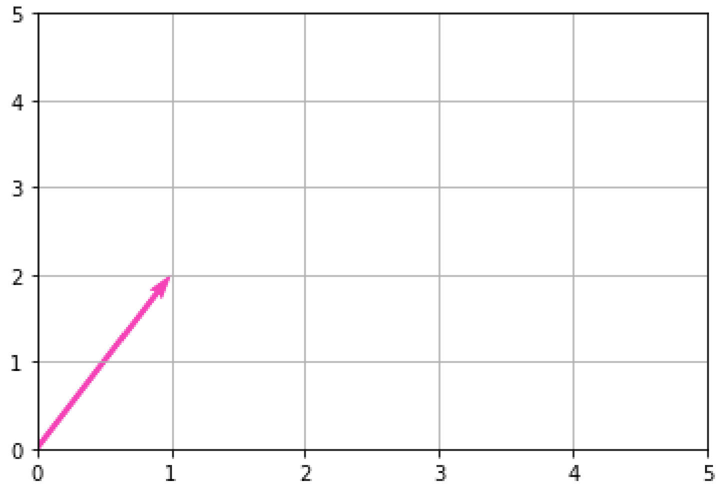


Figure 3 Output for part 1

The result of part 1 lab activity (Figure 3) outputs a 2-dimensional plane/plot with one arrow representing a linear equation.

The researcher created a simple flowchart of the function for part 1 of the lab activity(Figure 4).

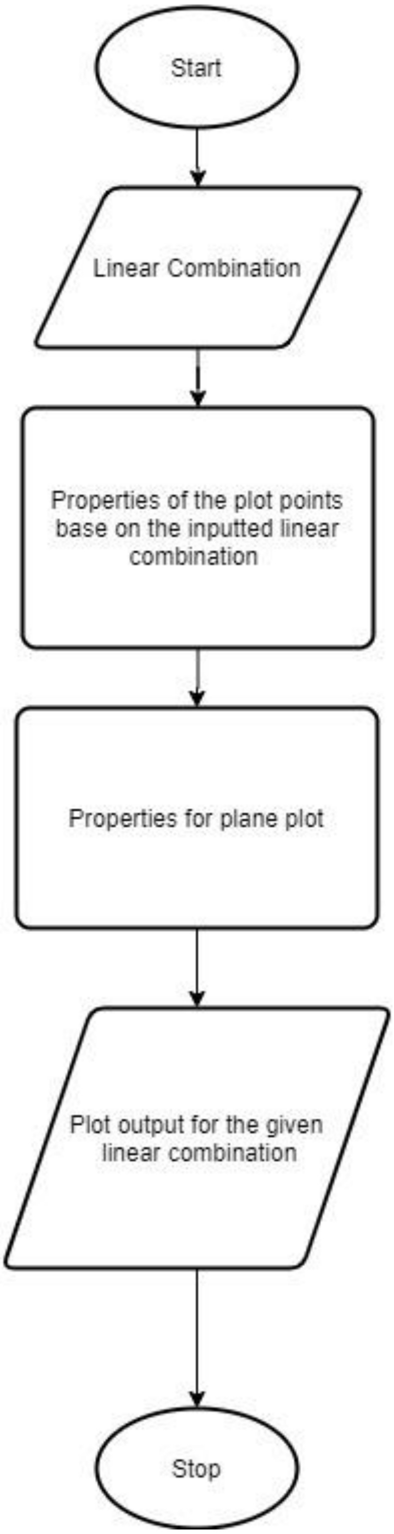
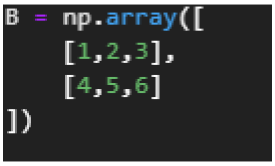

Start
Linear Combination
Properties of the plot points base on the inputted linear combination
Properties for plane plot
Plot output for the given linear combination
Stop

Figure 4 Flowchart for Part 1

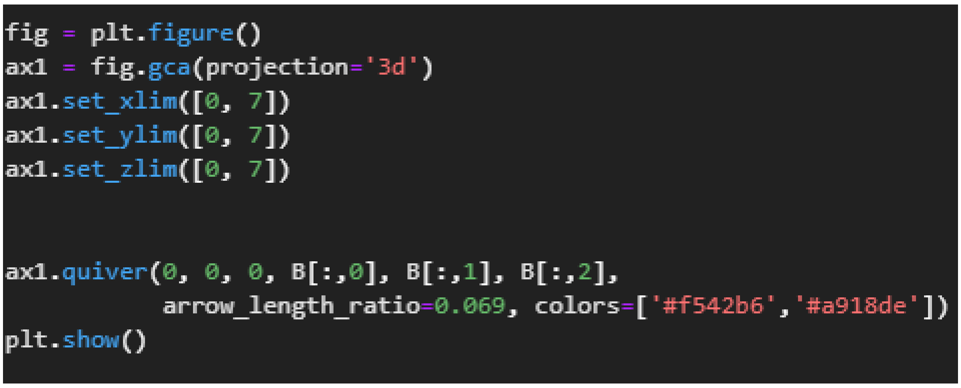
The researcher created a simple flowchart of the function for part 1 of the lab activity(Figure 4).



```
B = np.array([
    [1,2,3],
    [4,5,6]
])
```

Figure 5 Declared Value for Part 2

For part 2 of the lab activity, the reseachers decleared linear combination with rank two & two vectors, $B = \begin{cases} x + 2y + 3z \\ 4x + 5y + 6z \end{cases}$ which is represent in matrix form using Numpy array() function (Figure 5).



```
fig = plt.figure()
ax1 = fig.gca(projection='3d')
ax1.set_xlim([0, 7])
ax1.set_ylim([0, 7])
ax1.set_zlim([0, 7])


ax1.quiver(0, 0, 0, B[:,0], B[:,1], B[:,2],
           arrow_length_ratio=0.069, colors=['#f542b6','#a918de'])
plt.show()
```

Figure 6 Code for Part 2

For the executable code in part 2 of the lab activity (Figure 6), the reacher used the same function for the part one lab activity. However, the only added function is the gca() function that specifies the plane type, which is a 3-dimension plane that needs three vectors.

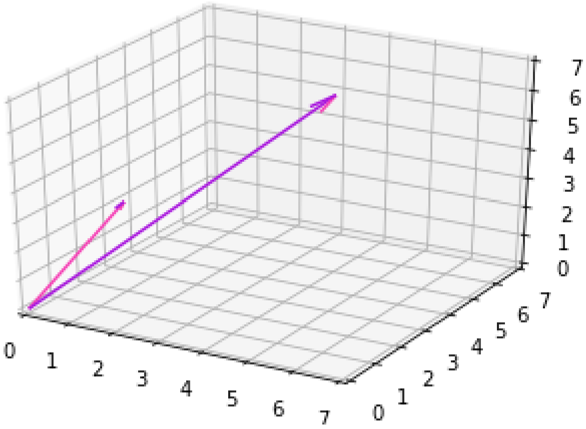


Figure 7 Output for Part 2

For the output for part 2 (Figure 7), the code created by the researchers outputs a plane/plot in 3-dimensions with two arrows representing the two declares linear equations.

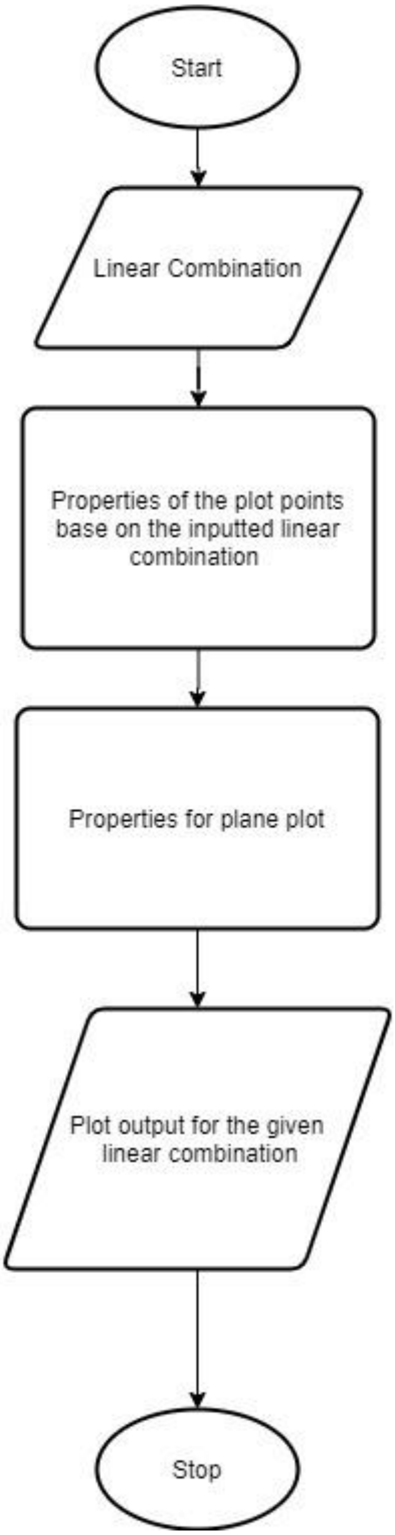

Start
Linear Combination
Properties of the plot points base on the inputted linear combination
Properties for plane plot
Plot output for the given linear combination
Stop

Figure 8 Flowchart for Part 2

The flowchart in part 2 (Figure 8) is the same as the flow chart in part 1.

A two-dimensional plot/plane is the most useful graph because it is used to correlate, compare or even identify changes over time or activities. A 4-dimensional or above plane is mathematically possible, but 4-d space is hard to visualize or comprehend 4n dimension because we live in a 3-dimensional plane.

IV. Conclusion

The researcher used Matplotlib and Numpy functions to represent a linear combination in 2-dimensional and 3-dimensional plane and learned that plotting plane is simple, ploting vectors are useful in different areas, or occasions such as knowing if the business is getting better or even going bankrupt also it is useful to today’s ongoing situation to track the pandemics movement to avoid more infections .

References

[1] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[2] D. Blaheta, "A visual proof of amortised-linear resizable arrays", ACM SIGCSE Bulletin, vol. 41, no. 3, pp. 338-338, 2009. Available: 10.1145/1595496.1562979.