Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 2

Plotting Vectors using NumPy and MatPlotLib

Submitted by:
Vallarta, Troy Joaquin G.

Instructor:
Engr. Dylan Josh D. Lopez

October 04, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of using NumPy functions and plotting vectors using MatPlotLib. The students are task to understand, code, and visualize the given vectors. Teaching the researchers in plotting graphs in Python.

II. Methods

- The practices of the activity are to create arrays and creating the vectors
    - They imply to teach that researchers can plot and visualize code in Python using MatPlotLib.
- The deliverables of the activity are to provide a deep understanding of vectors and coding arrays.
    - To achieve the lessons in Laboratory 2, the researchers are task to read about NumPy and MatPlotLib. The researchers are also tasked to learn the formulas and how to execute it.

III. Results

Laboratory report two consists of vectors and arrays that use Numpy and MatPlotLib. Numpy provides an array that includes mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, introductory linear algebra, basic statistical operations, random simulation, and much more [1]. Numpy is a powerful package in python that has almost everything that the programmers need in their projects; it is a powerful tool for the people learning machine learning. Handling arrays are not an easy task because it has many data. MatPlotLib is to visualize the data that the programmer encoded in python, it is also a powerful package in python, using this package the programmers can visualize their codes in simple lines of code. The "matplotlib.pyplot.plot()" function is the most useful when programmers want to visualize their works.

The programmer in this lab report added two additional import to clean the codes, as seen in Figure 1, the use of it is to be efficient in coding and cleaning the codes that can lead to long stretches of code in one line. Substituting the name of the import will be more comfortable in the programmer's eyes in the long run. The use of acronym should connect from the real use of what the programmer import.

```
import numpy as np
from numpy import random as ra
from numpy import linalg as alg
import matplotlib.pyplot as plt
import matplotlib

%matplotlib inline
```

Figure 1. Importing packages

The Laboratory report is on a budget; the programmer randomizes the path of how the eagle flew to its nest. The activity provides three random longitudes and latitudes of an eagles path. In Figure 2, the range of randomness of the eagle's path is from -10 to 10. np.array function is the process of making the first, second, and third set of longitude and latitude be in one list.

```
def track_eagle(make_figs=True):
    long = ra.randint(-10,10, size=3)
    lat = ra.randint(-10,10, size=3)



    dist1 = np.array([long[0],lat[0]])
    dist2 = np.array([long[1],long[1]])
    dist3 = np.array([long[2],lat[2]])
```

Figure 2. Random Eagle latitude and longitude

Figure 3 is where all the formula for the displacement, total distance, and the direction of the eagle. The total distance is easy to understand; it is the total of all the three distance of the eagle. However, it is separated longitude and latitude; the np.array function made it possible to separate it. To calculate the displacement, the programmer used the np.linalg.norm, the programmer substitute the np.linalg function to clean the code and more comfortable to understand when debugging. The process of the displacement is the total distance has two elements in an array and plug it in equation 2 in Figure 4, latitude and longitude are distance x and y respectively. The second part of the code block of Figure 3 is to visualize the grids, legends, and labels. Equation 3 in Figure 4, the variable α is necessary when the longitude and latitude randomly generate. The use of α is preventing error in the code when the randomly generated number turns out to be a zero.

```
dist_total = dist1 + dist2 + dist3
disp = alg.norm(dist_total)

alpha = 10 ** -6
theta =  np.arctan(dist_total[1] / dist_total[0] + alpha)

theta = np.degrees(theta)

## Plotting the PH Eagle flight vectors.
plt.figure(figsize=(10,10))
plt.title('Philippine Eagle Flight Plotter')
plt.xlim(-30, 30)
plt.ylim(-30, 30)
plt.xlabel('Latitudinal Distance')
plt.ylabel('Longitudinal Distance')
plt.grid()
n = 2
```

Figure 3. Formulas for Eagle's Flight

$$(Eq.1) dist_{total} = (long_{total})\hat{x} + (lat_{total})\hat{y}$$

$$(Eq.2) disp = \sqrt{dist_x{}^2 + dist_y{}^2}$$

$$(Eq.3) \theta = \arctan\left(\frac{y}{\alpha + x}\right)$$

Figure 4. Equations

Figure 5 is where the computations of the previous codes are visualized through Matplotlib plots. Using quiver to plot the arrow where the eagle flew. The first, second, and third directions are represented as red, blue, and green respectively. The displacement is color orange. The purpose of Figure 5 is to be easier for humans to visualize the track of the eagle. The result is creating unique patterns for every programmer, due to the randomness of the eagle's direction going to its nest.

```
plt.quiver(0,0, dist1[0], dist1[1],
           angles='xy', scale_units='xy',scale=1, color='red',
           label='Trajectory 1: {:.2f}m.'.format(np.linalg.norm(dist1)))
plt.quiver(dist1[0], dist1[1], dist2[0], dist2[1],
           angles='xy', scale_units='xy',scale=1, color='blue',
           label='Trajectory 2: {:.2f}m.'.format(np.linalg.norm(dist2)))
plt.quiver(np.add(dist1[0],dist2[0]), np.add(dist1[1],dist2[1]),
           dist3[0], dist3[1], angles='xy', scale_units='xy',scale=1, color='green',
           label='Trajectory 3: {:.2f}m.'.format(np.linalg.norm(dist3)))
plt.quiver(0,0, dist_total[0], dist_total[1],
           angles='xy', scale_units='xy',scale=1, color='orange',
           label='Displacement: {:.2f}m. @ {:.2f}'.format(disp, theta))

plt.legend()
```

Figure 5. Plotting the directions

Figure 6 activity one is to show the plot of the eagle's path. Setting the make_figs function is to export the graph generated by the code to the programmer's local drive. In Figure 7-10 are 4 random scenarios of the eagle going to its nest.

```
    if make_figs == True:
        plt.savefig(f'LinAlg-Lab2-PH Eagle-{int(disp)}@{int(theta)}.png', dpi=300)
    plt.show()

track_eagle(make_figs=True) ## Let 'make_figs' equal to False during debugging.
                            #Only flip it to True if the researcher wants to export to computer the graph created below
```

Figure 6. Showing the Graph

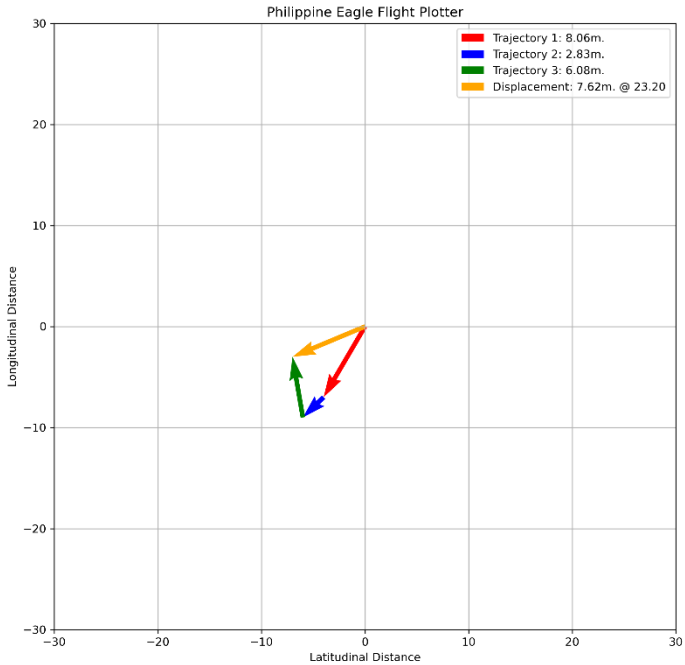

Figure 7. PH Eagle 7.62 m @ 23 °

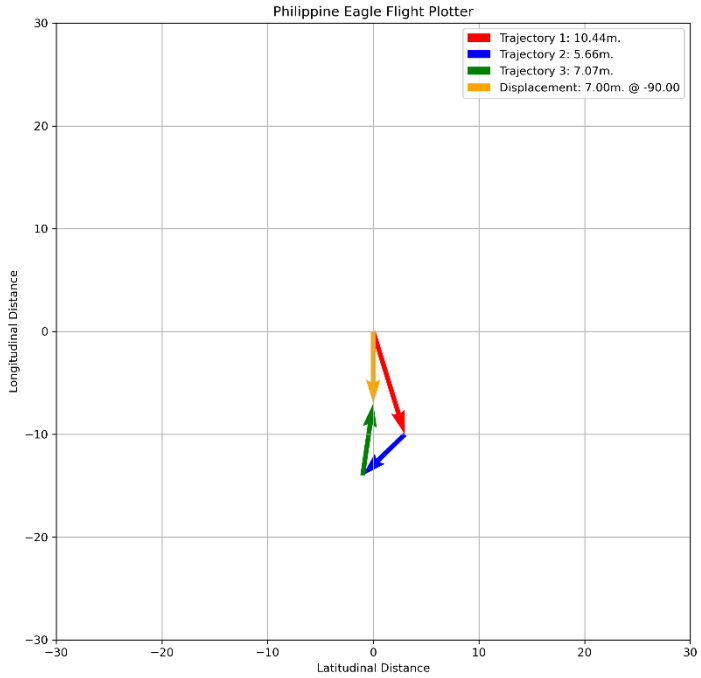

Figure 8. PH Eagle 7 m @90 °

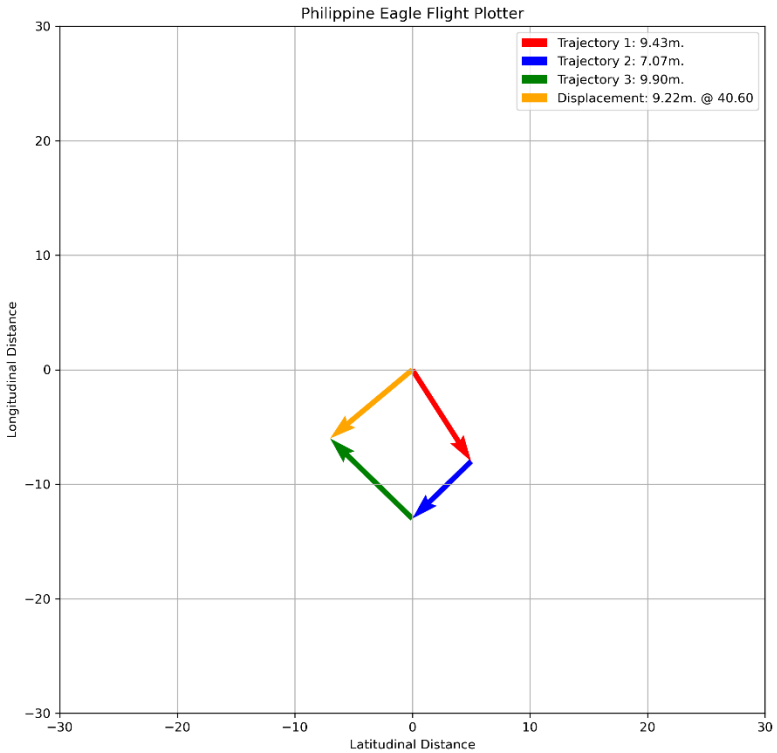

Figure 9. PH Eagle 9.22 m @ 40 °

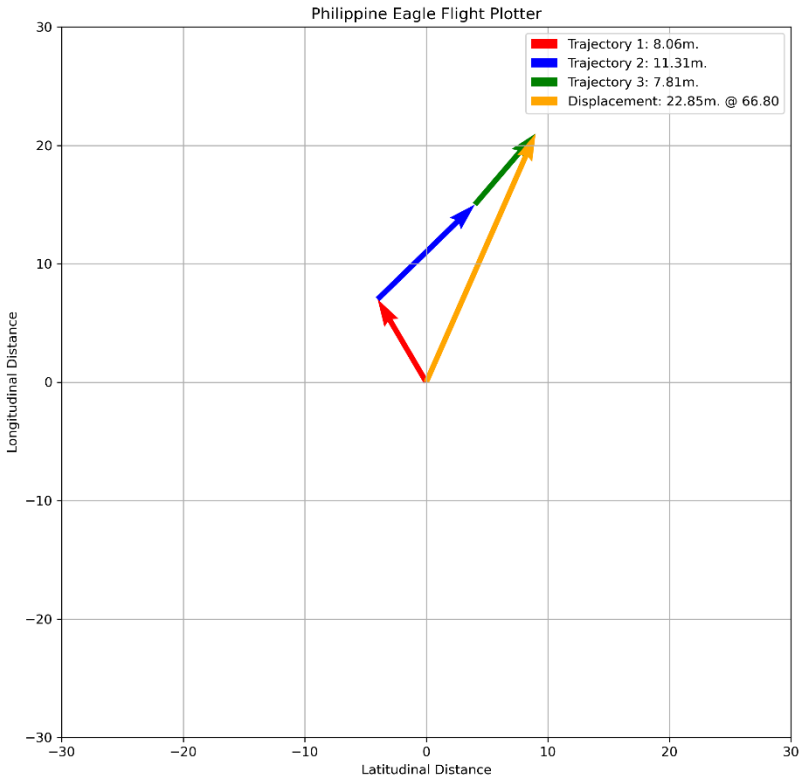

Figure 10. PH Eagle 22.85 m @ 66 °

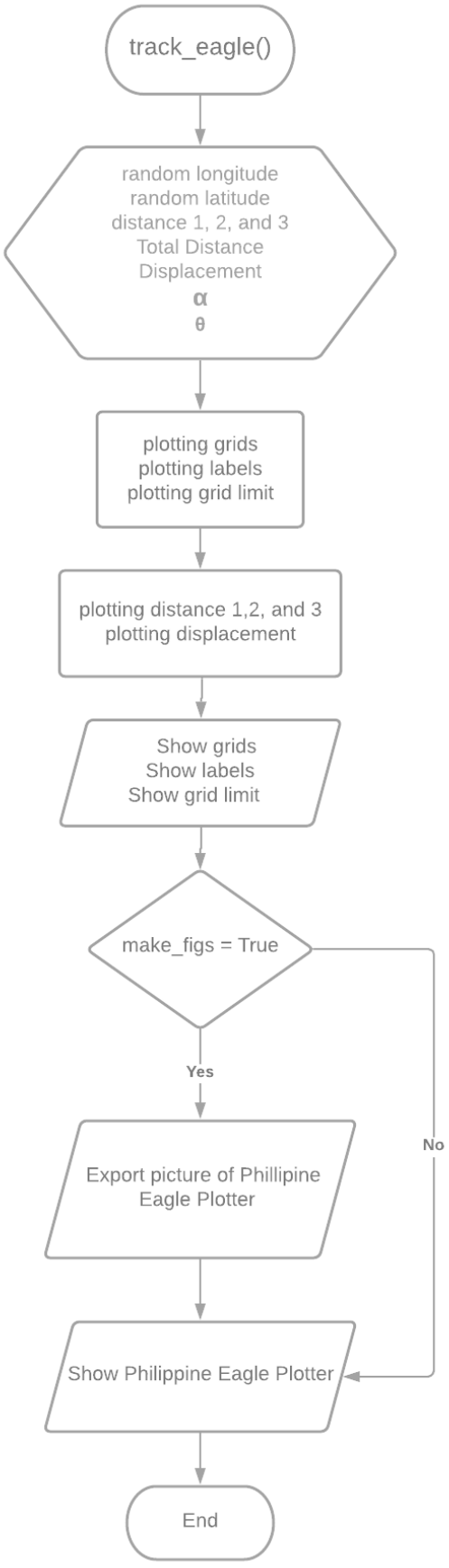

Figure 11. track_eagle flowchart

Equation 1,2, and 3:

$s_t = 2t^3 + t^2 + 3t + 2$

$v_t = 6t^2 + 2t + 3$

$a_t = 12t + 2$

Based on the formula of kinematics the programmer reversed engineered and interpret the bad coding habit in question two. The programmer changes the variables in Figure 12, other programmers can easily understand the code without testing it. In programming, it is commonly done in groups of people or teams. If the code is hard to understand because of bad coding habits, the workflow in a group will become sluggish. Learning about good coding habits is essential for programmers. The faster the programmers do their job the faster result. The result of the program is the speed(t), velocity (t), and accel(t) respectively.

```
def eagle_kinematics(speed, time,accel, velocity):
    req_shape = 4
    time_vect = np.array([time**3, time**2, time, 1])
    if speed.shape == (req_shape,) and velocity.shape == (req_shape-1,) and accel.shape == (req_shape-2,):
        speed_t = np.sum(np.multiply(speed, time_vect))
        velocity_t = np.sum(np.multiply(velocity, time_vect[1:]))
        accel_t = np.sum(np.multiply(accel, time_vect[2:]))

    else:
        print(f'Input displacement vector is not valid. Make sure that the vector shape is ({req_shape},)')

    return speed_t, velocity_t, accel_t
```

```
The speed, velocity, and acceleration value comes from the equations
```

```
speed = np.array([2,1,3,2])      #eq1
velocity = np.array([6,2,3])     #eq2
accel = np.array([12,2])         #eq3
time = 2
eagle_kinematics(speed, time)
```

```
(28, 31, 26)
```

Figure 12. Kinematics in Code

The third activity in the laboratory report is almost the combination of the activity one and two. The use of MatPlotLib to visualize the relation of profit and the reach of Bebang's store in her Facebook account. The programmer's task to plot the reach and profit of the store per week in a month. It means that the programmer is tasked to plot 4 weeks and find the efficiency of the store. Efficiency in a month is where it tallies everything if the store is making a profit. Figure 13, in one week the program records the profit and reach of Bebang's store on Facebook. The week's performance is similar to Figure 3 activity one of the laboratory reports,

in Eagle plotter, it is the displacement, but in this activity, it measures the performance of the store in a month.

```
def month_profit_trace(profit, reach, make_figs=True): ## You can simplify/
    if (profit.shape == (4,))  and (reach.shape == (4,)) :
        week1 = np.array((reach[0], profit[0]))
        week2 = np.array((reach[1], profit[1]))
        week3 = np.array((reach[2], profit[2]))
        week4 = np.array((reach[3], profit[3]))


        week_total = week1 + week2 + week3 + week4
        week_performance = alg.norm(week_total)
        alpha = 10**-6
        reach_gradient = np.degrees(np.arctan(week_total[1]/week_total[0]))


        plt.figure(figsize=(16,5))
        plt.title('Bebang\'s Month Post Efficiency')
        plt.xlim(0,1.01*np.sum(reach))
        plt.ylim(-np.sum(np.abs(profit)),np.sum(np.abs(profit)))
        plt.xlabel('FB Post Reach Increment')
        plt.ylabel('Profit')
        plt.grid()
        n = 2
```

Figure 13. Formulas of Reach and profit

Equations:

$$Performance_{week} = \sqrt{(Reach_{Total})^2 + (Profit_{Total})^2}$$

$$Reach_{gradient} = \theta = \tan^{-1}(\frac{Profit_{Total}}{Reach_{Total}})$$

Figure 14 is where the plot of the profit and reach of Bebang's store per week. The first to fourth quiver functions shows the plotting of Bebang store's direction if she had profit in those weeks. The last quiver function is the performance of Bebang's store in a month. The performance is the most important part of Bebang's store, it will show if her Facebook store is making a profit. If the performance in the graph is rising, the performance is great otherwise, it is not.

```
    ## put necessary vector plotting code here
    plt.quiver(0,0, week1[0], week1[1],
               angles='xy', scale_units='xy',scale=1, color='yellowgreen', width=0.0025,
               label='Week 1: {:.2f}'.format(alg.norm(week1)))

    plt.quiver(week1[0],week1[1], week2[0], week2[1],
               angles='xy', scale_units='xy',scale=1, color='lime', width=0.0025,
               label='Week 2: {:.2f}'.format(alg.norm(week2)))
    plt.quiver(np.add(week1[0], week2[0]), np.add(week1[1], week2[1]), week3[0], week3[1],
               angles='xy', scale_units='xy',scale=1, color='green', width=0.0025,
               label='Week 3: {:.2f}'.format(alg.norm(week3)))

    plt.quiver(week1[0]+week2[0]+week3[0],week1[1]+week2[1]+week3[1], week4[0], week4[1],
               angles='xy', scale_units='xy',scale=1, color='darkgreen', width=0.0025,
               label='Week 4: {:.2f}'.format(alg.norm(week4)))

    plt.quiver(0,0, week_total[0], week_total[1],
               angles='xy', scale_units='xy',scale=1, color='red', width=0.005,
              label='Efficiency: {:.2f} @ {:.2f}'.format(week_performance, reach_gradient))

    plt.legend(loc='upper left')

    if make_figs:
        plt.savefig(f'LinAlg-Lab2-Bebang Post Eff-{int(week_performance)}@{int(reach_gradient)}.png', dpi=300)

    plt.show()

else:
    print('Dimension error') ## Make a more appropriate error statement.
```

Figure 14. Plotting the weeks and performance

```
profit= np.array([-18000, 3000, 12000, 10000])
reach = np.array([1000, 100, 500, 10])

month_profit_trace(profit, reach, make_figs=True)
```

Figure 15. Scenario 0 Input

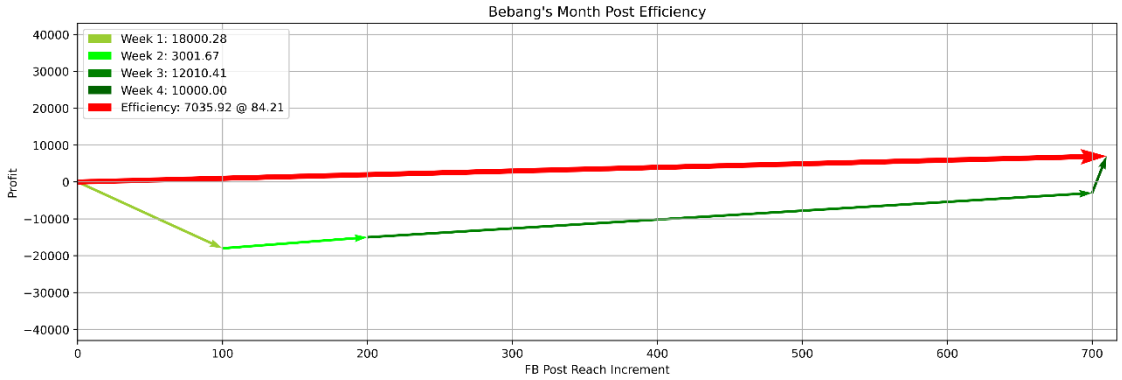

Figure 16. Scenario 0 Output

Figure 16 is the demonstration scenario wherein Bebang's store has a debt of 18,000, but her reach is gaining a lot. Therefore, while her reach is increasing, the profit is also increasing. The more people that her store reaches the more profit will return in the long run. In Figure 16, it shows that her growth is normal. Like many stores, at first, there will be debt, but as the brand grows the larger the reach, the more profit will arrive.

The relationship between Facebook post reach and Bebang's store shows that marketing is one of the efficient ways to make a profit. The more consumers that Bebang's store reaches, the more profit she will get. Many companies use this tactic, these companies spend thousands or even millions to reach consumers. It also creates familiarization and reliability, when products reach many consumers they will tend to choose the brand that they are familiar with.

The y-axis and FB post reach on the x-axis and not the other way around is because the programmer and Bebang want to see the efficiency of her business in terms of profit. The reach is a meter on how many people in a week are getting reach by her marketing. In this method, Bebang and other people involved in this business can check the efficiency and read if it is efficient or not.

```
profit= np.array([18_000, -8_000, -22_000, -9_000])
reach = np.array([2_000, 350, 500, 50])

month_profit_trace(profit, reach, make_figs=True) ##
```

Figure 17. Scenario 1 Input

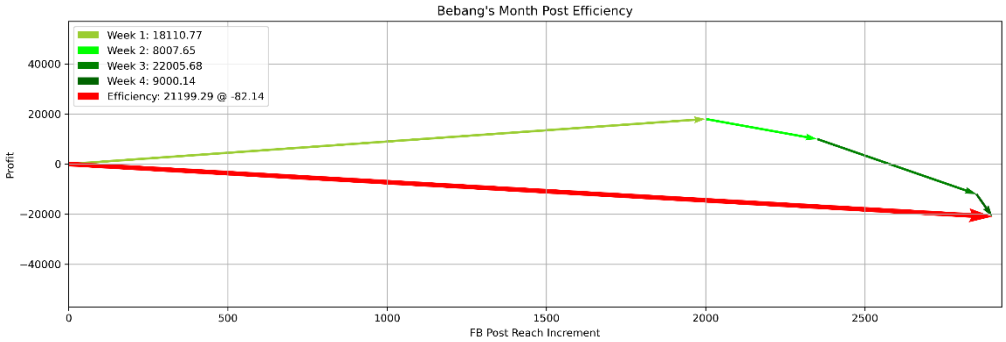

Figure 18. Scenario 1 Output

In Figure 18 the result is a negative efficiency, this type of graph is what business people should avoid always. In this scenario Bebang's store is a one-hit-wonder, for example, she is mentioned by influencers that her doughnuts are delicious. The rampant rise in the first week is a good sign, but consumers tasted her doughnuts but did not become a regular. Therefore, Bebang's store came crashing down its efficiency.

```
profit= np.array([-10_000, -20_000, 25_000, 5_000])
reach = np.array([200, 920, 1_500, 500])

month_profit_trace(profit, reach, make_figs=True) ##
```

Figure 19. Scenario 2 Input

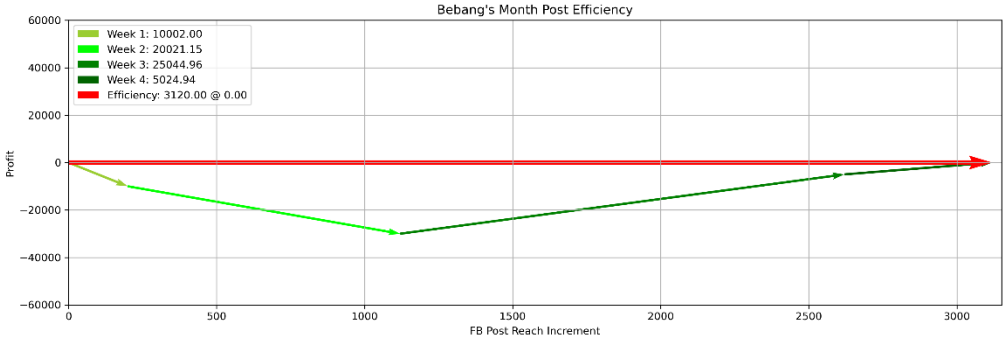

Figure 20. Scenario 2 Output

Figure 20 shows that it has an efficiency of zero. This is an example of a neutral graph. At first Bebang's store is losing money, but in the third week, she is gaining a lot of consumers and making money. In the last two weeks of the month is great, but in the first two weeks has a lot of wasted money. She eventually recovered from losing money. This graph shows that it has a great future ahead in profit even if she has an efficiency of zero this month.

```
profit= np.array([20_000, 3_000, 10_000, 5_000])
reach = np.array([500, 120, 840, 420])

month_profit_trace(profit, reach, make_figs=True)
```

Figure 21. Scenario 3 Input

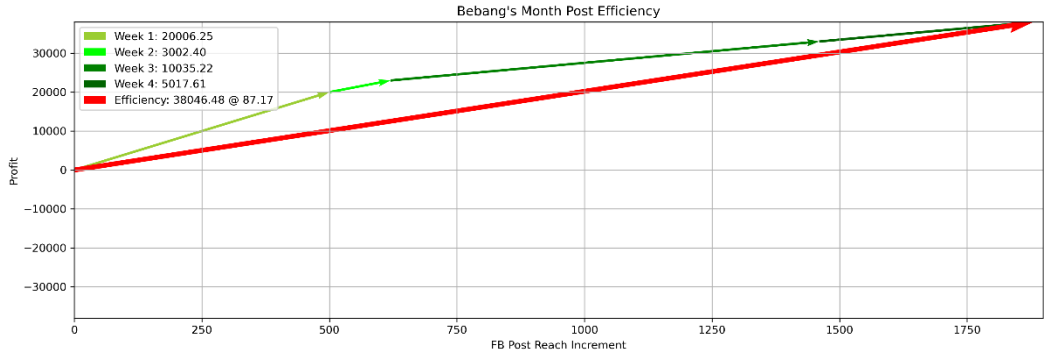

Figure 22. Scenario 3 Output

Figure 22 shows that it has an efficiency of 87%, in this scenario, for example, Bebang is a famous influencer/streamer on Facebook. Then suddenly she produces her own Bebang store to sell doughnut. This will result in large profits and efficiency in her store because she has a following before making the Store. In this case, it is a good marketing strategy and a profitable business.

IV. Conclusion

The second laboratory report in linear algebra is challenging but fun. I learned to use NumPy.random.randint() function, before I learned about this I was thinking of making a card game in my spare time. One of the struggles is using lists and random parts of the attack of a certain card. Although the random function is not the reason for stopping in making that game (mainly about the artwork) it is good to know that I learned about that little function. There are many things that I learned, MatPlotLib is godsent to someone like me. When I browse into their website, there is so much graph that I want to learn more fundamentals in making a graph. I like statistics and I know why, it is because of basketball and when I looked at how I can make my graphs and do it in code, it is a blessing. This Laboratory report is hard to understand if I don't want to learn about this. I also implemented my learning in physics in this laboratory. Especially kinematics, good thing that I did not throw my notebook in physics, or else I would search about this topic on the internet.

1. Enumerate and briefly discuss the functions you have used in the laboratory exercise, please cite their usage using their respective documentations. (min of 200 words)

- The first function that I used in this program is the numpy.random.randint()[5] it is used to make random numbers, the parameter can have the range of numbers that will be randomized. Second is the numpy.array()[6] in which it makes an array the variables that I input inside the parameter, it is a useful tool to understand that the programmer are making an array and easy to read. The benefit is that the more value inside of an array the harder it is to keep track, in this function it is much easier to find a certain element. Third one is the numpy.linalg.norm()[4] function wherein the two variables that the programmer input inside is that the function will execute the displacement formula. Next is the numpy.arctan()[9] function wherein the values that the programmer put inside the parameter will be multiplied to arctan. The next one is numpy.degrees()[10] where the value will be converted into a degree.

Creating the plotter is all about the MatPlotLib, this package is full of graphs and many many things to learn. The use of quiver function is to make an arrow to the grid. The essential function in this package is the matplotlib.pyplot.show() and matplotlib.pyplot.legend() function where I can make the graph that I wanted. Without these functions it is hard to visualize the code, especially when it is plotting something.

2. How do vectors relate to real-life values? (min of 50 words)

- Vectors relate to the real life values when we need to visualize the graphs or statistics in life. For example kinematics, it is used in engineering and in physics. Vectors are everywhere in our life, the most probably I am excited about vectors is the kinematics, when I stumbled upon kinematics it is when I am goofing around trying to make games. It is important for the motion of the character, for me vectors are essential to learn.

3. Kindly give other examples of how vectors are used or other real-life situations that can be modeled using vectors? (min of 100 words and do proper citation)

- The use of vectors are everywhere. For example in the crime scene, what if the crime scene has a broken glass and a huge rock that is clearly thrown in the direction of the victim. Vectors are needed to solve that case. Motion of the character in video games, it uses kinematics and use vector. It is either Vector2 when in 2 dimensional and Vector3 when in 3D. Pilots use a lot of vectors to find the shortest path to their destination and save fuel. Vectors are responsible for that. In the world of basketball, trajectory has vectors and they can provide a deep understanding in learning how to shoot the basket. I remembered when I was young the perfect shot is when your arm shoots at a 40 – 45 degrees. That is how the vectors are used in real-life.

References

[1] The SciPy community, “What is NumPy?” - NumPy v1.19 Manual. [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html. [Accessed: 04-Oct-2020].

[2] J. Hunter, “Visualization with Python,” Matplotlib, 15-Sep-2020. [Online]. Available: https://matplotlib.org/. [Accessed: 04-Oct-2020].

[3] “Legend guide,” Legend guide - Matplotlib 3.3.2 documentation, 15-Sep-2020. [Online]. Available: https://matplotlib.org/tutorials/intermediate/legend_guide.html. [Accessed: 04-Oct-2020].

[4] The SciPy community , “numpy.linalg.norm,” numpy.linalg.norm - NumPy v1.19 Manual, 29-Jun-2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html. [Accessed: 04-Oct-2020].

[5] “numpy.random.randint”, NumPy. [Online]. Available: https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html. [Accessed: 04-Oct-2020].

[6] “numpy.array”, NumPy. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.array.html. [Accessed: 04-Oct-2020].

[7] “numpy.add”, NumPy. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.add.html. [Accessed: 04-Oct-2020].

[8] “numpy.sqrt”, NumPy. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.sqrt.html. [Accessed: 04-Oct-2020].
[9] “numpy.arctan”, NumPy. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.arctan.html. [Accessed: 04-Oct-2020].
[10] “numpy.degrees”, NumPy. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.degrees.html. [Accessed: 04-Oct-2020].

Appendix A

Github Repository Link:

https://github.com/Zofserif/Linear-Algebra/tree/master/Lab%202%20-%20Coding%20Vectors%20using%20NumPy%20and%20Matplotlib/lab2