



Linear Algebra

Laboratory Activity No. 2

Plotting Vectors using NumPy and Matplotlib

Submitted by:

Aquiro, Freddielyn E.

Instructor:

Engr. Dylan Josh D. Lopez

October 4, 2020

I. Objectives

This laboratory activity aims to implement the basic principles in using `matplotlib.pyplot` and `numpy`. To familiarize with the libraries in Python for numerical and scientific programming and to visualize the vectors, its plotting with its operation.

II. Methods

The practices of activity are used to implement the function of importing `numpy` and `matplotlib` to show its specific usage such as `plt.quiver()`, `np.`, `plt.figure()`, `plt.xlim()` and `plt.ylim()`, `plt.grid()`, `plt.legend()`, `shape`, and `size`.

The deliverables of this activity are to show the plotting of the eagle's flight and the online business of Bebang. For the eagle kinematics, is to provide proper documentation of the codes in order to understand the process of the code. In order to achieve them, I used different functions and objects in the problem and supplemented each missing code from its problem. Also, to understand the process of assigning and calling its variables and function. It took me a week to finalized its missing code and debug it properly.

III. Results

The first problem is to identify and show the trajectory of the eagle's flight. In order to get the values of longitude and latitude is to assign its random integers using the `np.randomint()`. Then I assigned the arrays starting from zero to two. Then I used the formula of Pythagorean's theorem using `np.sqrt()` to get the magnitude displacement of the vectors. I used the `plt.quiver()` to represent velocity vectors of the trajectory by arrows.

There 4 quiver in the problem, the first trajectory represented by the red arrow, the second trajectory is the arrow blue, the third trajectory is the green arrow and for the total displacement is the orange arrow. The three displacement are connected to each other and it is all add together to get the coordinates of the total displacement of the eagle's flight, "`theta = np.arctan(dist_total[1]/(alpha+ dist_total[0]))`" used to compute the angle of displacement followed by `theta = np.degrees(theta)`, to convert rad to degree.

Alpha is used to set the current objects to the axes. The function `angles = xy, scale_units = xy` generated to plot the vectors in the x and y plane with its units. I set different colors to properly read the trajectory of the eagle's flight. `plt.legend` represents its axis and its label. `plt.show()` function will display all the plotting of the eagle's flight. To further understand the process of the code I created a flow chart diagram.

See figure 1 , 1.1, 1.2, 1.3 and 1.4 for the flow chart, code, and result of eagle's flight.

$$dist\ total = (longitude\ total)\hat{x} + (latitude\ total)\hat{y}$$

$$disp = \sqrt{distx^2 + disty^2}$$

$$\theta = \tan^{-1}\left(\alpha + \frac{y}{x}\right)$$

Formulas for eagle's flight trajectory

```
[37]: import numpy as np
import matplotlib.pyplot as plt
import matplotlib

%matplotlib inline
def track_eagle(make_figs=True):
    long = np.random.randint(-10,10, size=3)
    lat = np.random.randint(-10,10, size=3)
    dist1 = np.array([lat[0], long[0]])
    dist2 = np.array([lat[1], long[1]])
    dist3 = np.array([lat[2], long[2]])

    dist_test_1_2 = np.add(dist1,dist2)
    dist_total = np.add(dist_test_1_2,dist3)
    disp = np.sqrt((dist_total[0]**2) + (dist_total[1]**2))
    alpha = 10**-6
    theta = np.arctan(dist_total[1]/(alpha+ dist_total[0]))
    theta = np.degrees(theta)

    n=2
    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()

    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()

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

    plt.show()
track_eagle(make_figs=False)
```

Figure 1: Eagle's flight trajectory code.

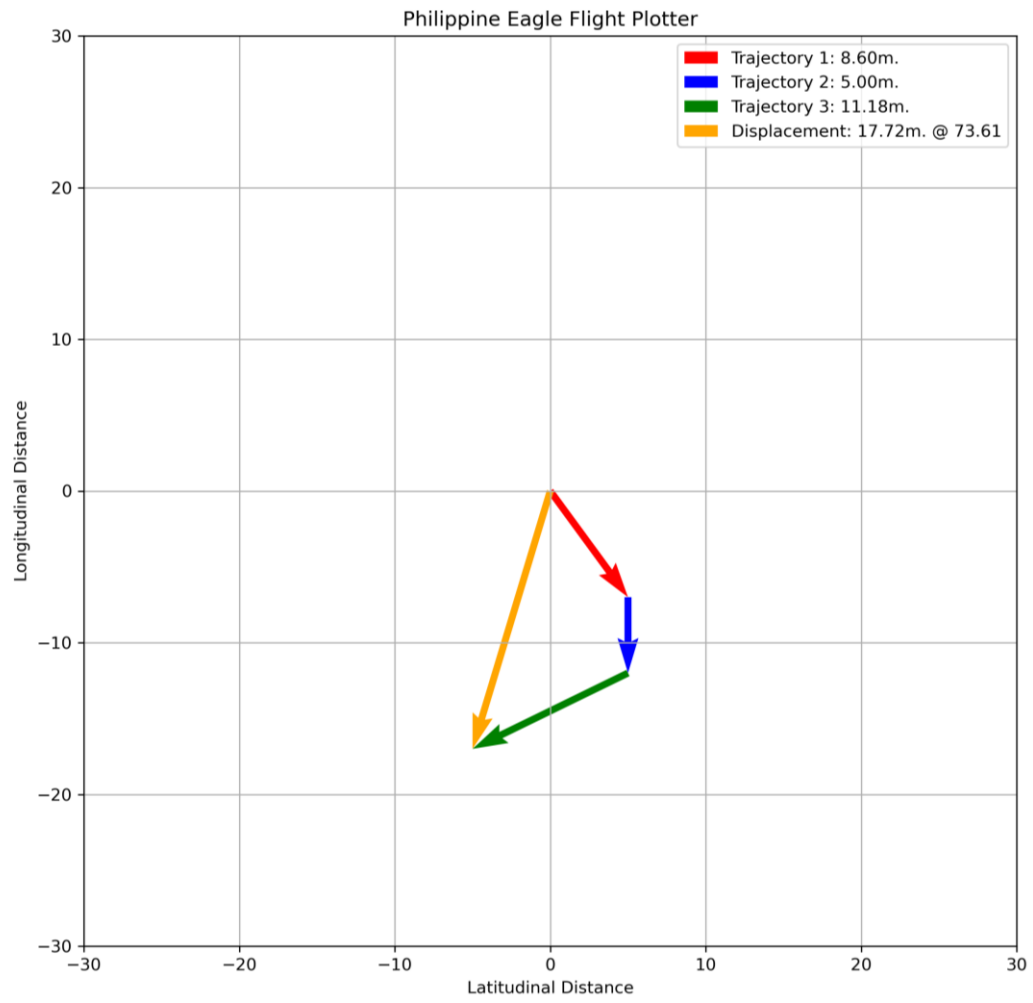


Figure 1.1: First result of eagle's flight trajectory

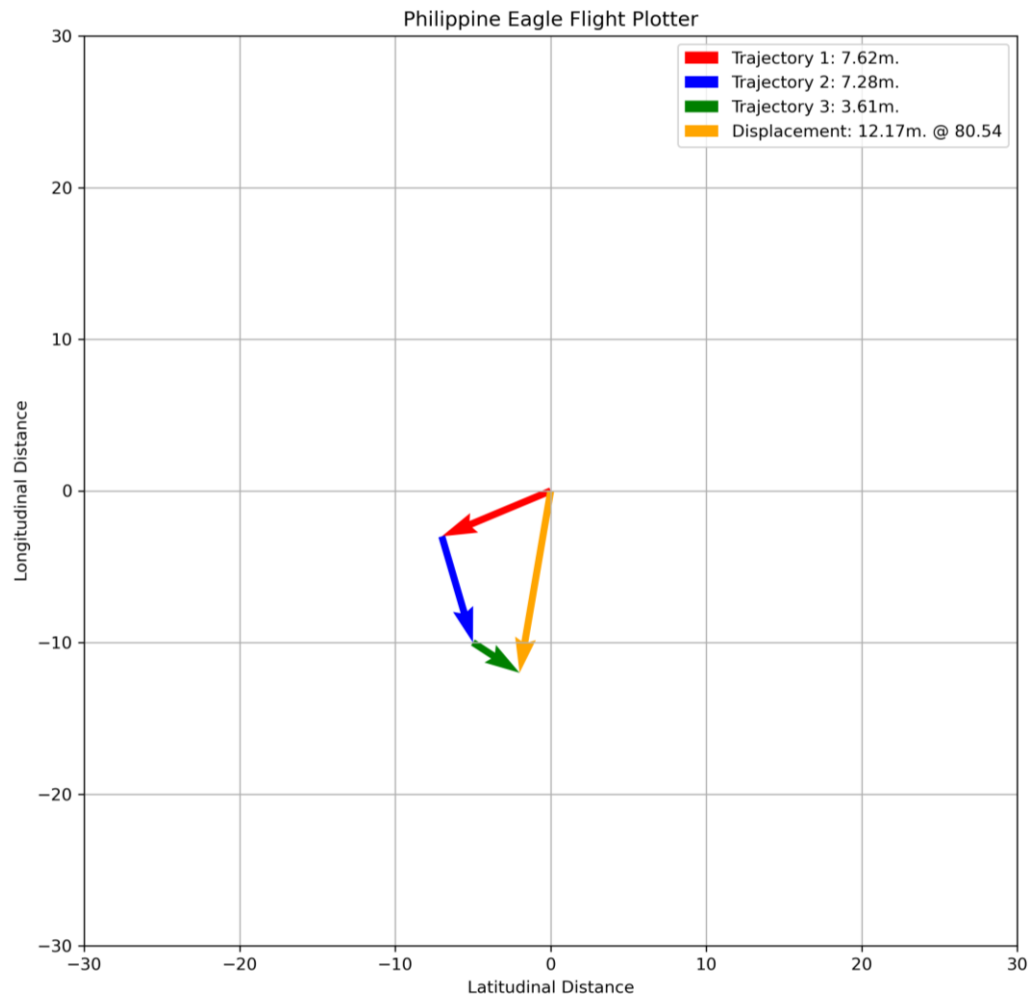


Figure 1.2: Second result of eagle's flight trajectory

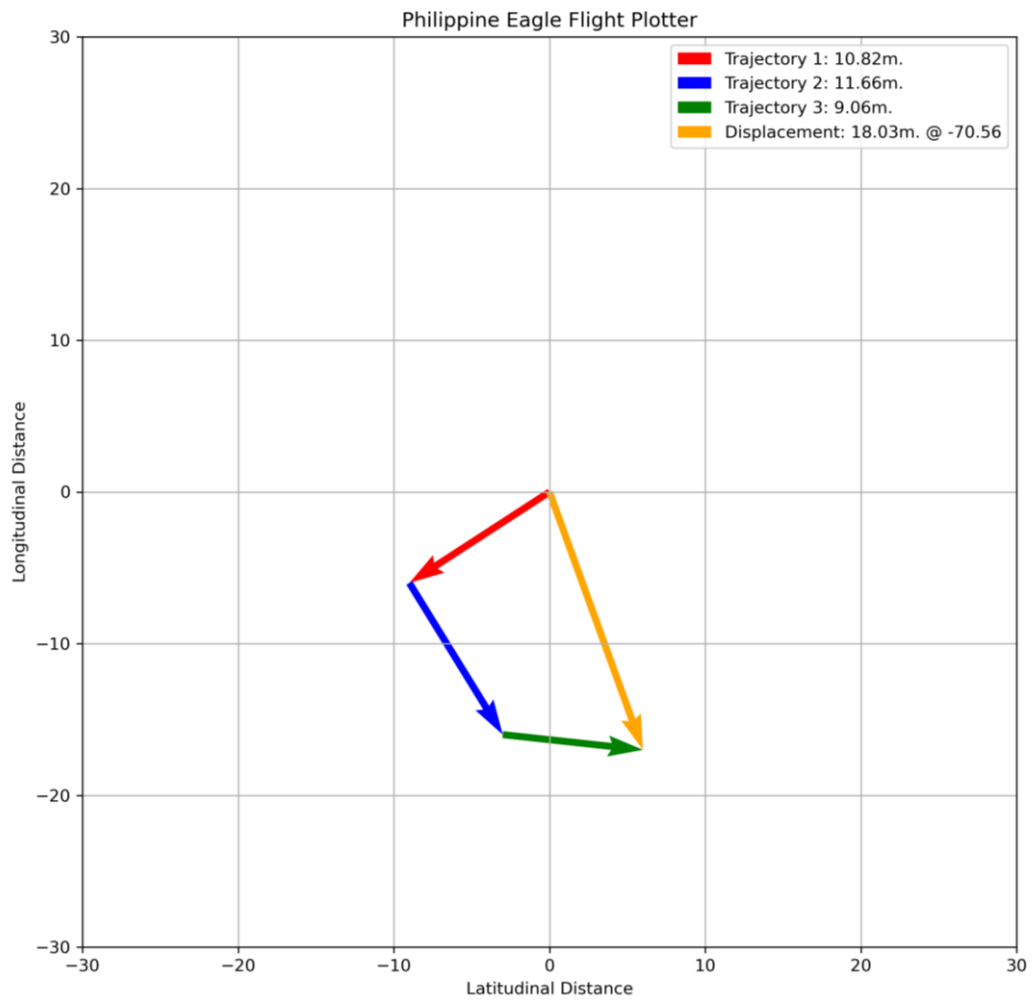


Figure 1.3: Third result of eagle's flight trajectory

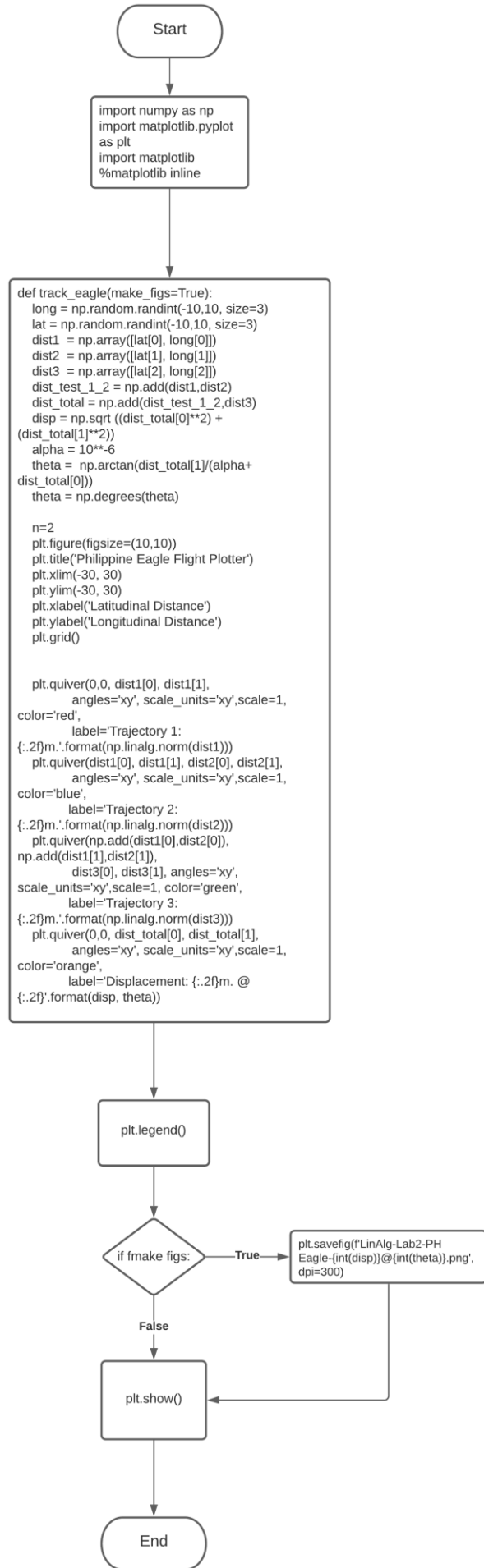


Figure 1.4: Flowchart for eagle's flight trajectory

The second problem is to fix the proper documentation of the eagle_kinematics, I used in assigning the variables for the time and position of the eagle_kinematics. The functions in eagle_kinematics are to get the eagle_total position, eagle_total_velocity, and the eagle_total_acceleration.

I assigned pst and res for the input eagle_position and time. Multiply the arrays of ([8,4,2,1] and [2,1,3,2]) is equivalent to the sum of [16,4,6,2] is equivalent to 28. The process in getting the eagle_total_position of the eagle's flight, followed by its eagle_total_velocity it multiplies the arrays of ([4,2,1] and [6,2,3]) is equivalent to the sum of [24,4,3] that will be equivalent to 31, while the eagle_total_acceleration multiply the elements in the arrays ([2,1] and [12,2]) is equivalent to the sum of [24,2] that will be equivalent to 26. Importing numpy as np will use the function np.array(), used to create and store arrays. np.multiply(), used to multiply the corresponding elements. np.sum(), is used to get all the results of arrays from the axis. For the last function np.zeros() used to return to all arrays that store zeroes. See figure 2 and 2.1

```
import numpy as np
def eagle_kinematics(eagle_position, time):

    req_shape = 4
    eagle_velocity = np.zeros((req_shape-1,))
    eagle_acceleration = np.zeros((req_shape-2,))
    eagle_total_vector = np.array([res**3, res**2, res, 1])

    if eagle_position.shape == (req_shape,):
        eagle_velocity = np.array([3*eagle_position[0], 2*eagle_position[1], eagle_position[2]])
        eagle_acceleration = np.array([2*eagle_velocity[0], eagle_velocity[1]])
        eagle_total_position = np.sum(np.multiply(eagle_position, eagle_total_vector))
        eagle_total_velocity = np.sum(np.multiply(eagle_velocity, eagle_total_vector[1:]))
        eagle_total_acceleration = np.sum(np.multiply(eagle_acceleration, eagle_total_vector [2:]))

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

    return eagle_total_position, eagle_total_velocity , eagle_total_acceleration

pst = np.array([2,1,3,2])
res = 2
print("Eagle kinematics:", eagle_kinematics(pst, res))
```

Figure 2: Eagle kinematics code

```
Eagle kinematics: (28, 31, 26)
```

Figure 2.1: Output for the Eagle kinematics

The last problem is getting Bebang's online FB reach post and profit by using the same concept from the first problem. But here in this problem it talks about the weeks, profit, and reach gradient. I assigned week_1_2 to at week1 and 2. Then another for week 3 and 4 to get resultant vectors. I used the Pythagorean formula to get the week_performance of her online business as function $\text{np.sqrt}((\text{week_total}[0]**2) + (\text{week_total}[1]**2))$.

Then I set the plt.xlim as $(0, 1.01 * \text{np.sum}(\text{reach}))$ and for the y axis as $\text{plt.ylim}(-\text{np.sum}(\text{np.abs}(\text{profit})), \text{np.sum}(\text{np.abs}(\text{profit})))$. Then for plotting the weeks I used $\text{plt.quiver}()$ initiating from its origin (0,0) followed by the $\text{week}[0]$ and $\text{week}[1]$. And getting its angles as function $\text{angles} = 'xy'$ and scale units, then I set its arrows color and set the width to 0.0025. Then to get its efficiency, I assigned it first from its origin (0,0), then I inputted the corresponding result as $\text{week_total}[0]$, $\text{week_total}[1]$. In order to show the result of the output assigned different values to the profit and FB reach post and to save the image of the result the if makes figs should be true and to print the results I used the function $\text{plt.show}()$. See figure 3, 3.1, 3.2, and 3.3 for the codes and output.

$$\text{week_performance} = \sqrt{\text{week_total}[0]^2 + \text{week_total}[1]^2}$$

$$\text{reach_gradient} = \tan^{-1}\left(\alpha + \frac{\text{week_total}[1]}{\text{week_total}[0]}\right)$$

Formula for the week_performance and reach_gradient

```

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

%matplotlib inline
def month_profit_trace(profit, reach, make_figs=True):
    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_1_2 = np.add(week1, week2)
        week_3_4 = np.add(week3, week4)
        week_total = np.add(week_1_2, week_3_4)
        week_performance = np.sqrt((week_total[0]**2) + (week_total[1]**2))
        alpha = 18**-.6
        reach_gradient = np.arctan((week_total[1]) / (alpha + week_total[0]))
        reach_gradient = np.degrees(reach_gradient)

        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

        plt.quiver(0,0, week1[0], week1[1],
            angles='xy', scale units='xy', scale=1, color='blue', width=0.0025,
            label='Week 1: {:.2F}'.format(np.linalg.norm(week1)))

        plt.quiver(week1[0], week1[1], week2[0], week2[1],
            angles='xy', scale units='xy', scale=1, color='crimson', width=0.0025,
            label='Week 2: {:.2F}'.format(np.linalg.norm(week2)))

        plt.quiver((week1[0] + week2[0]), (week1[1] + week2[1]), week3[0], week3[1],
            angles='xy', scale units='xy', scale=1, color='coral', width=0.0025,
            label='Week 3: {:.2F}'.format(np.linalg.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='magenta', width=0.0025,
            label='Week 4: {:.2F}'.format(np.linalg.norm(week4)))

        plt.quiver(0,0, week_total[0], week_total[1],
            angles='xy', scale units='xy', scale=1, color='cyan', 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")

profit = np.array([-1000, 19000, 22000, 900])
reach = np.array([2000, 175, 99, 666])

month_profit_trace(profit, reach, make_figs=False)

```

Figure 3: Bebang's Online Business code

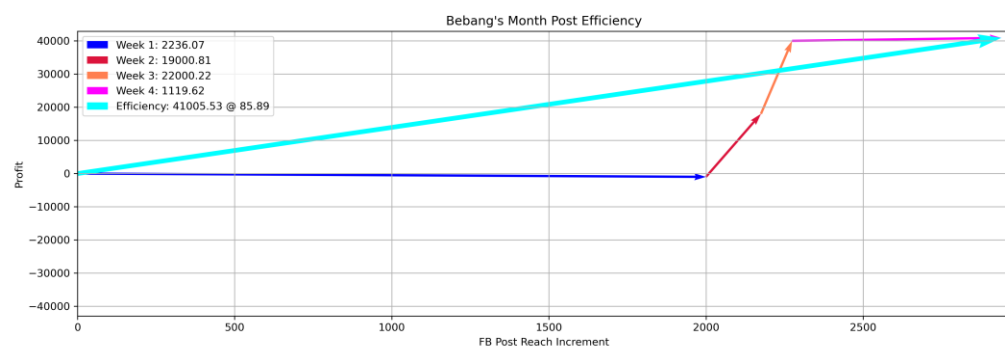


Figure 3.1: First scenario output

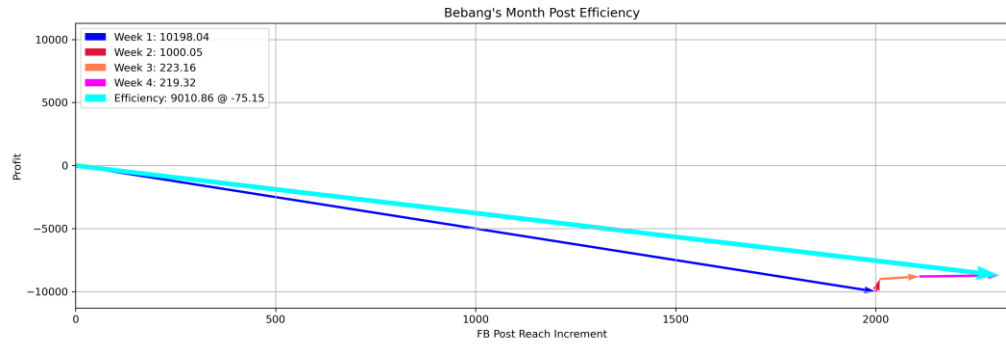


Figure 3.2: Second scenario output

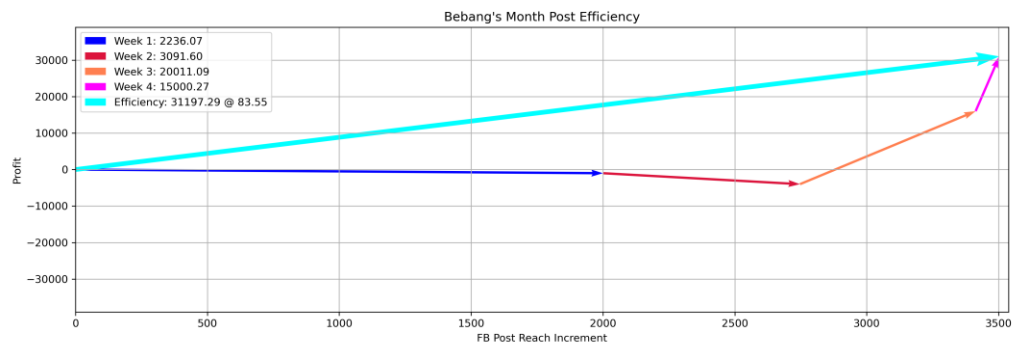


Figure 3.3: Third scenario output

IV. Conclusion

Based from laboratory no.2, I used the functions imported from numpy and matplotlib such as `plt.quiver()`, used to displays its velocity vectors represented as arrows with its corresponding component values as (u,v) that will set points of the x and y axis which implemented in getting the trajectory of the eagle's flight and in plotting the profit, FB reach posts and efficiency of Bebang's online business.[1]. The numpy as `np` set large numeric data types in the arrays. which is used in adding the vectors and getting its magnitude for displacement in the first problem similar to the third problem and getting the values of eagle kinematics.[2]. In the problem of one and three, in order to get its magnitude, I used the Pythagorean formula function as, `np.sqrt()` [3]. Then I used `tan inverse` to get the angle of each vector implied to the problem one and three. The `alpha` variable is used to set the current axes of objects and to avoid the error in the program [4]. The function `plt.figures()` used to set all the current values of arrays.[5]. `Plt x and y lim`, set the current limit of the x and y-axis in the

vectors implied to the problem one and three. [6] `Plt.grid()` used to set the visibility of the axes from its objects like shown in the results of the flow of arrow directions in the graph with the different colors represented in each displacement and FB reach posts, efficiency, and profits [7]. `Plt.legend()`, used to setting and visualizing the elements of the graph followed by its `plt.label()` providing its label plot that shown in each result in the problem one and three.[8] Shapes are used to set how many rows and columns present in the vector [9] and the size used to present the total number of elements in the vector [10]. `Plt.show()`, is used to display all the possible values in the arrays to get the visual representation of the graph without this, I can't able to read the results and understand the process of the vectors in the program.

Vectors are composed of magnitude and directions, it is present in our daily lives without appreciating the beauty of it. It applies in our daily lives such as walking from home to another place, it calculates the direction of your path and over an amount of time or a collision of a car and a truck resulting in its speed, force and direction.

One of the examples based from on of the website I read, is when the boat is crossing a river the current of the water forces the boat in another direction, therefore if the boat has its motor, the force will be present in the situation because it is moving into one direction. In order to know where the boat will go, we need to get the result of the magnitude and direction of each force[12]. Another example given a situation about the cup location and table location you have to lift a cup of coffee with your eyes closed, the cup is in the table and it is at a one-arm distance away from you. Therefore, it will take a couple of tries before you lift the cup because you may end up by swinging your one arm in an arc to locate the cup[13].

References

- [1] “Quiver – (MATHLAB Functions)”
<http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/ref/quiver.html> (accessed Oct 4,2020)
- [2] “Numpy – (Python Numpy Tutorial with Jupyter and Colab))”
<https://cs231n.github.io/python-numpy-tutorial/>(accessed Oct 4,2020)
- [3] “np.sqrt() – (Geeksforgeeks)”
<https://www.geeksforgeeks.org/numpy-sqrt-in-python/>(accessed Oct 4,2020)
- [4] “Alpha – (Math works)”
<https://www.mathworks.com/help/matlab/ref/alpha.html> (accessed Oct 4,2020)
- [5] “ plt.figure() – (What is the necessity of plt.figure() in matplotlib?)”
<https://stackoverflow.com/questions/38666527/what-is-the-necessity-of-plt-figure-in-matplotlib>
(accessed Oct 4,2020)
- [6] “ plt.xlim() – (How to Set Limits for Axes in Matplotlib | Delft Stack)”
[https://www.delftstack.com/howto/matplotlib/how-to-set-limit-for-axes-in-matplotlib/#:~:text=Xlim\(\)%20and%20Ylim\(\)%20to%20Set%20Limits%20of%20Axes%20in%20Matplotlib,-matplotlib.&text=xlim\(\)%20and%20matplotlib.,range%20of%20the%20respective%20axes.](https://www.delftstack.com/howto/matplotlib/how-to-set-limit-for-axes-in-matplotlib/#:~:text=Xlim()%20and%20Ylim()%20to%20Set%20Limits%20of%20Axes%20in%20Matplotlib,-matplotlib.&text=xlim()%20and%20matplotlib.,range%20of%20the%20respective%20axes.)(accessed Oct 4,2020)
- [7] “ grid – (Matplotlib-Grids)”
https://www.tutorialspoint.com/matplotlib/matplotlib_grids.htm(accessed Oct 4,2020)
- [8] “plt.legend() – (Customizing Plot Legends)”
<https://jakevdp.github.io/PythonDataScienceHandbook/04.06-customizing-legends.html>(accessed Oct 4,2020)
- [9] “Shapes – (numpy.ndarray.shape — NumPy v1.19 Manual)”
<https://numpy.org/doc/stable/reference/generated/numpy.ndarray.shape.html>(accessed Oct 4,2020)
- [10] “Size - (Numpy size() function | Python)”
<https://www.geeksforgeeks.org/numpy-size-function-python/>(accessed Oct 4,2020)
- [11] “Plt.show() – (Visualization with Matplotlib)”
<https://jakevdp.github.io/PythonDataScienceHandbook/04.00-introduction-to-matplotlib.html#:~:text=is%20your%20friend,-plt.,display%20your%20figure%20or%20figures.&text=python%20myplot.py-,The%20plt.,your%20system's%20interactive%20graphical%20backend.> (accessed Oct 4,2020)
- [12] “ example of vectors in real life situation- (Vectors in space)”
[https://math.libretexts.org/Bookshelves/Calculus/Book%3A_Calculus_\(OpenStax\)/12%3A_Vectors_in_Space](https://math.libretexts.org/Bookshelves/Calculus/Book%3A_Calculus_(OpenStax)/12%3A_Vectors_in_Space)(accessed Oct 4,2020)
- [13] “ another example of vectors in real life situation – (Applications of vectors in real life)”

<https://www.freeaptitudecamp.com/applications-of-vectors-in-real-life/>(accessed Oct 4,2020)