Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 2

Laboratory 2: Plotting Vectors Using Numpy and MatPlotLib

Submitted by:
Surio, Rovil Jr., M.

Instructor:
Engr. Dylan Josh D. Lopez

September 28, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of plotting a vector with the use of the MathPlot library for creating visualizations in Python and the NumPy library for fast operations on arrays.

II. Methods

- The practices of the activity include creating arrays and plotting of vectors.
  - The activity teaches and implies the techniques of manipulating arrays and plotting of vectors.
- The deliverables of the activity are to provide an array and formulas needed to completely plot the vectors.
  - It was achieved by reading the documentation of the functions.

III. Results

The results from the activities done by implementing the principles and techniques of representing the vectors through plotting using the MathPlot library and NumPy library to manipulate arrays were presented in this section. The first activity was focused on providing necessary formulas to complete the code while the second activity was all about code analyzation and how to transform a project that lack of coding conventions into a well created project that follows coding convention. Lastly, the third activity was about understanding the relationship of each vector used to represent data.

```
## START OF FUNCTION
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]])     ##arrays for each vector
    dist2 = np.array([lat[1],long[1]])
    dist3 = np.array([lat[2],long[2]])

    dist_total = dist1 + dist2 + dist3 ##computation for resultant vector
    disp = np.linalg.norm(dist_total)  ## The L2 Norm was used which is calculated as the square root-
                        ## -of the sum of the squared vector values and executed using function np.linalg.norm
    alpha = 10**-6
    theta = np.arctan(dist_total[1]/(dist_total[0]+alpha)) ##Computation for angle of displacement
    theta = np.degrees(theta)                              ##converting rad to degrees

    ## 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

    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()

## END OF FUNCTION
```

Figure 1 The code executed

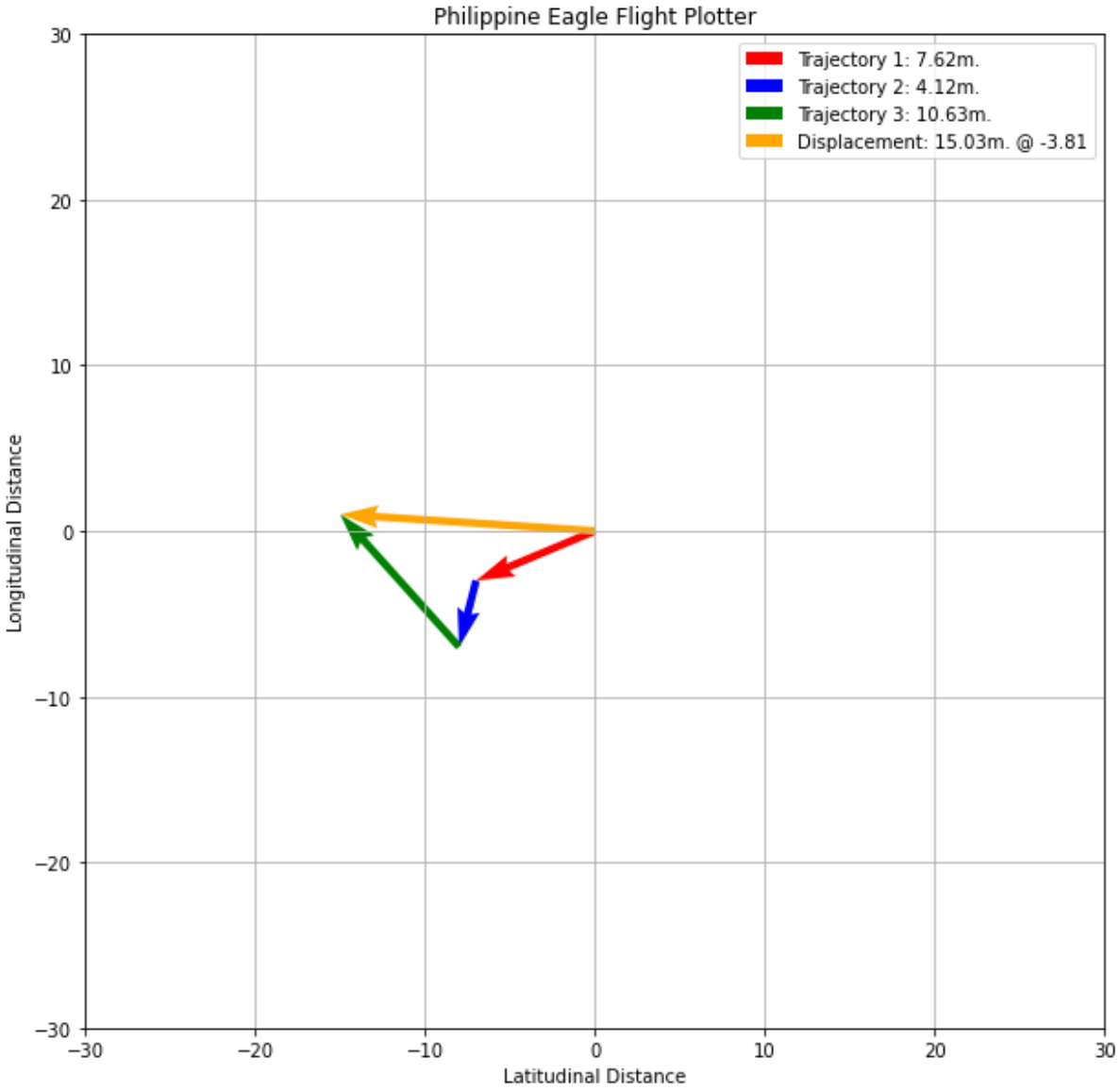

Figure 2 Sample Output

Figures 1 and 2 show the code executed and the sample output or the Philippine Eagle Flight Plotter. The task was to fill in the necessary codes to make the function works. Three 2-D arrays were created which is assigned into dist1, dist2, and dist3 variable from the long and lat variable which generates random integers. In computing the total distance, the computation of three arrays was performed in the dist_total variable therefore, it holds the value of the summation of dist1, dist2, and dist3. The magnitude of the displacement was computed next which is assigned into disp variable and the L2 norm was used which is calculated as the square root of the sum of the squared vector values and executed using the function np.linalg.norm [1]. The angle of the displacement was calculated next using the equation provided arctan $\left(\frac{y}{x+\alpha}\right)$ wherein Figure 1 is equal to np.arctan(dist_total[1]/(dist_total[0]+alpha)) and it

was assigned to theta variable wherein after getting the result, the value of theta was converted into degrees using the code provided np.degrees from the Numpy library.

Proceeding to the plotting of vectors, plt.quiver was used because it plots a 2D field of arrows based on the parameter provided [2]. As shown in Figure 1, the first vector was plotted using the line of code that includes plt.quiver (0,0, dist1[0], dist1[1]) this is the parameter used to identify the location of the arrow, the 0,0 is the origin of the tail of the arrow while the dist1[0] and the dist[1] is the head of the arrow. The code that follows this is for the modification of how the arrow will look and the code needed for the legend of the graph. The same concept was used in plotting the second vector and for the third, the addition of the first vector and the second is used as the origin of the tail and still used its value dist3[0] and dist3[1] for its arrowhead. The displacement was plotted using the values of variable dist_total and in the same line of code, for the legend, it includes the variable theta or the degree. The plt.legend() shows the legend at the right top side of the graph as shown in Figure 2 and the plt.show() show what is inside the plt or the vectors plotted in the canvas. Lastly, the track_eagle(make_figs=True) is the one that confirms if the saving of the graph into an image will be executed so, if it was set to False then no image will be saved.

```
def eagle_kinematics(velocity, eagle_time):
    req_shape = 4 ;                     ##set an array with 4 elements
    shape3 = np.zeros((req_shape-1,));  ##creates  [0.0.0.] from 4 zeros to minus 1
    shape2 = np.zeros((req_shape-2,))   ##creates  [0.0.] from 4 zeros to minus 2
    eagle_time = np.array([t**3, t**2, t, 1]); ##t=2 : [8 4 2 1]
    if velocity.shape == (req_shape,):        ##checks if the two arrays are equal
        velocity2 = np.array([3*velocity1[0],2*velocity1[1], velocity1[2]]) ##[6 2 3]
        velocity3 = np.array([2*velocity2[0],velocity2[1]])               ##[12 2]
        dist_total1 = np.sum(np.multiply(velocity1, eagle_time)) ;   ##sum of [8 4 2 1] x [2 1 3 2] = 28
        dist_total2 = np.sum(np.multiply(velocity2, eagle_time[1:])); ##sum of [4 2 1] x [6 2 3] = 31
        dist_total3 = np.sum(np.multiply(velocity3, eagle_time[2:])); ##sum of [2 1] x [12  2] = 26

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

    return dist_total1, dist_total2, dist_total3  ##returns 28,31,26
```

```
velocity1 = np.array([2,1,3,2])
t = 2
eagle_kinematics(velocity1, t)
```

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

Figure 3 The code executed and output

Figure 3 shows the organized eagle_kinematics function that computes the distance traveled by the eagle. The equations used are $distance = (velocity)(time)$ and $distance = distance1 + distance2 + ...$ To begin, the req_shape = 4 set an array that has 4 elements or set the shape into 4 and it was followed by the creation of shape3 and shape2 variables that returns an array of zeros since np.zeros are used. The eagle_time or the first array with 4 elements was created by getting the value of 't' which in this case was set to 2 by passing as an argument when calling the eagle_kinematics function and when it was substituted to the 't' in the line of operation provided it results in [8 4 2 1]. The if statement in Figure 3 checks if the number of elements of the velocity variable is equal to the req_shape which has a shape of 4. Besides, the variable velocity1 passed its value to velocity since it is used as an argument in calling the eagle_kinematics function. Therefore, the velocity now has 4 elements too and equal to the shape or req_shape. The next line of code is for making the second array of velocity. The velocity2 variable created an array with 3 elements by multiplying to 3 the first element of variable velocity1 or the 2 which results to 6 and multiplying to 2 the second element of variable velocity1 or the 1 which results to 2 and lastly, getting the third element of the variable velocity1 or the number 3. Overall, the velocity2 now has [6 2 3]. Next, the velocity3 created an array with 2 elements by multiplying to 2 the first element of velocity2 or the 6 and getting its second element which results in [12 2]. Now, the set of velocities are present, and the computation of distance is possible by manipulating the array of eagle_time to match the number of elements for each set of velocities. The values of eagle_time were sliced to match the number of elements or the shape of each array of velocities. The first total distance

computed using the equation provided from the first arrays with 4 elements results to 28 while the second total distance from the arrays with 3 elements results to 31 and lastly for the third total distance from the arrays with 2 elements results to 26. The three values from the total distance were returned so when the function was called, it will give the three total distance.

```
## START OF FUNCTION
def month_profit_trace(profit, reach, make_figs=True):
    if (profit.shape == (4,)) and (fbpost_reach.shape == (4,)): #checks if the number of elements are equal
        week1 = np.array((fbpost_reach[0], profit[0]))   ##arrays for each vector
        week2 = np.array((fbpost_reach[1], profit[1]))
        week3 = np.array((fbpost_reach[2], profit[2]))
        week4 = np.array((fbpost_reach[3], profit[3]))

        week_total = week1 + week2 + week3 + week4   ##computation for resultant vector
        week_performance = np.linalg.norm(week_total) ##computation for magnitude of displacement
        alpha = 10**-6
        fbpost_reach_gradient = np.arctan(week_total[1]/(week_total[0]+alpha))##Computation for angle of displacement
        fbpost_reach_gradient = np.degrees(fbpost_reach_gradient) ##converting to degrees
        ## Plotting the Bebang's Month Post Efficiency
        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='yellowgreen', 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='green', width=0.0025,
                   label='Week 2: {:.2f}'.format(np.linalg.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='blue', 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='orange', 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='red', width=0.005,
                   label='Efficiency: {:.2f}m. @ {:.2f}'.format(week_performance, fbpost_reach_gradient))

        plt.legend(loc='upper left')

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

        plt.show()
    else:
        print('Profit and reach dimention are not equal')

## END OF FUNCTION
```

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

month_profit_trace(profit, fbpost_reach, make_figs=False)
```

Figure 4 The code executed

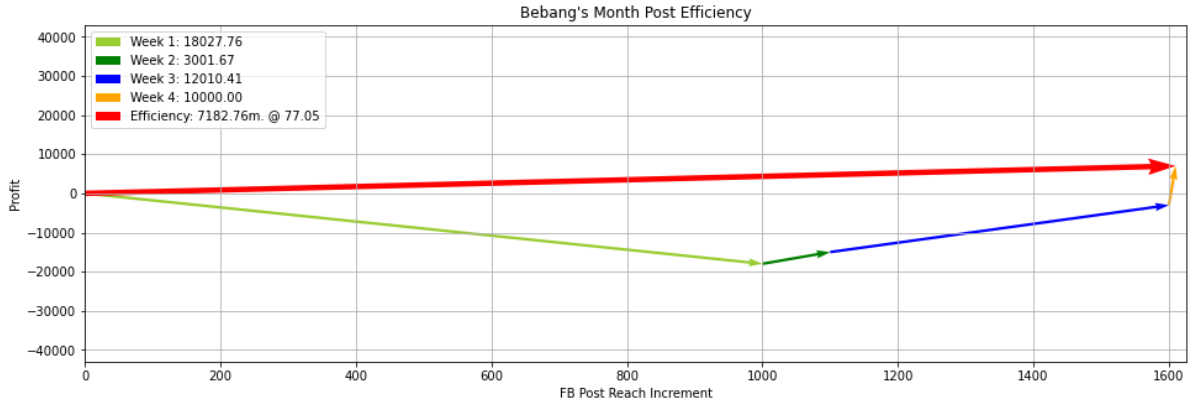

Figure 5 The output

Figures 4 and 5 show the completed code for Bebang's Project and the output or the result of Bebang's month post efficiency. The values of the vectors come from the arrays of profit and fbpost_reach by making an ordered pair of the 2 arrays. The computation of the resultant vector, magnitude of the displacement, angle of displacement, and even in plotting the vectors are the same as what is used or concept implemented in plotting Philippine Eagle Flight Plotter.

```
profit= np.array([-20000, 3000, 12000, 10000])   ##Original Values:-18000, 3000, 12000, 10000
fbpost_reach = np.array([1000, 100, 500, 10])    ##Original Values: 1000, 100, 500, 10

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

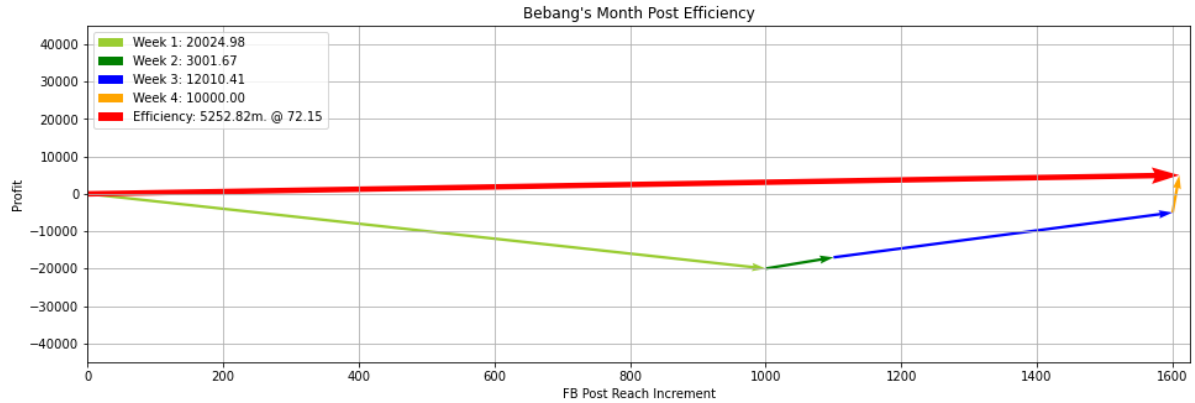

Figure 6 First scenario - lower profit

Figure 6 shows the first scenario wherein the total value of profit was decreased. It can be observed in this scenario that the efficiency of Bebang's post also decreased.

```
profit= np.array([-15000, 3000, 12000, 10000])   ##Original Values:-18000, 3000, 12000, 10000
fbpost_reach = np.array([1000, 100, 500, 10])    ##Original Values: 1000, 100, 500, 10

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

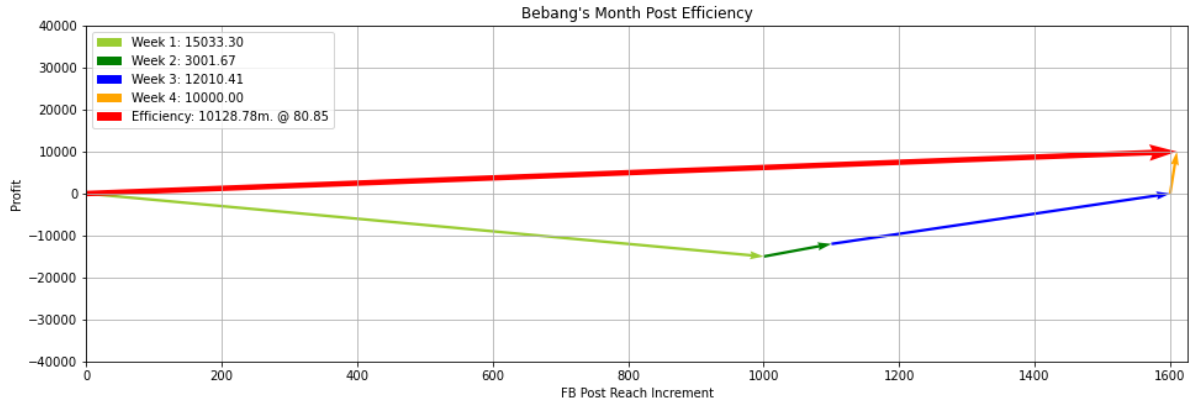

Figure 7 Second scenario – higher profit

Figure 7 shows the result of the second scenario wherein the total value of profit is increased. The efficiency of Bebang's post also increased.

```
profit= np.array([-15000, 3000, 12000, 10000])   ##Original Values:-18000, 3000, 12000, 10000
fbpost_reach = np.array([1000, 100, 200, 10])    ##Original Values: 1000, 100, 500, 10

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

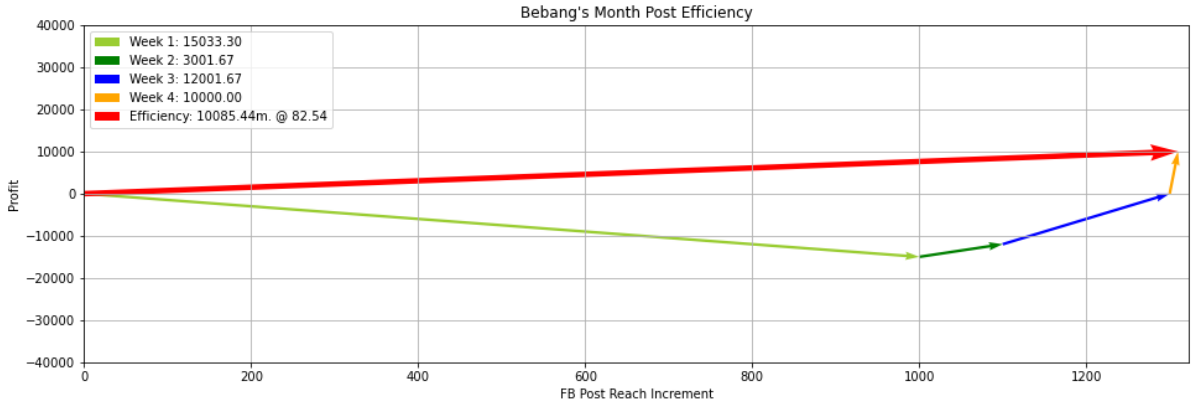

Figure 8 Third scenario – lower fbpost_reach

Figure 8 shows the result of the third scenario wherein the fbpost_reach was manipulated into a lower value. The result proves that the lower fbpost_reach means lower efficiency of Bebang's post.

```
profit= np.array([-15000, 3000, 12000, 10000])   ##Original Values:-18000, 3000, 12000, 10000
fbpost_reach = np.array([1000, 100, 890, 10])    ##Original Values: 1000, 100, 500, 10

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

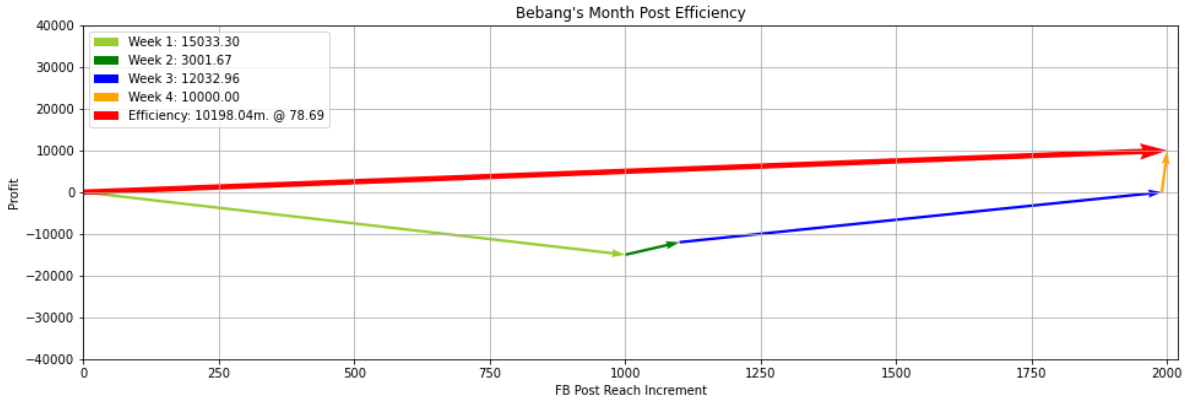

Figure 9 Fourth scenario – higher fbpost_reach

Figure 9 shows the result of the fourth scenario wherein the fbpost_reach was adjusted into higher value and this results in the higher efficiency of Bebang's post.

To conclude, as shown in Figures 6 to 9, the higher the profit or the fbpost_reach of Bebang means, the higher the efficiency of its month post while the lower profit or fbpost_reach means lower efficiency of the month post. Therefore, the Bebang's month post efficiency is directly proportional to profit and fb_post reach.

IV. Conclusion

The first function was used is the np.array which is used for creating an array [3]. The next is the one used in executing the L2 norm for computation of magnitude of displacement which is np.linalg.norm. It was defined according to its documentation [4], “This function is able to return one of eight different matrix norms.” It contains ord parameter which in this laboratory, the ord holds the L2 norm which is one of the eight different matrix norms and this is calculated as the square root of the sum of the squared vector values which is the formula in computing the magnitude of displacement [1]. The next function used is the np.arctan which is used in performing trigonometric inverse tangent [5] and the np.degrees which is used for converting angles from radians to degrees [6]. Moreover, in plotting the figures, plt.figure is used to create a new figure or activate an existing one [7]. The other function from the Mathplot library used in this laboratory includes plt.title – “Set a title for the axes.” [8], plt.xlim – “Get or set the x limits of the current axes.” [9], plt.ylim – “Get or set the y-limits of the current axes.” [10], plt.xlabel – “Set the label for the x-axis.” [11], plt.ylabel – “Set the label for the y-axis.” [12], plt.grid() – “Configure the grid lines.”, therefore if plt.grid is not used, there are no grid lines on the canvas [13]. Also, plt.quiver is used to plot a 2D field of arrows [2], plt.legend() is the one placing a legend on the axes [14], plt.show() is the one who display all open figures [15], plt.savefig is used to save the current figure [16]. Moreover, the np.zeros is used in the second problem in this laboratory and this function is used to return a new array of given shape and type which is filled with zeros [17]. The np.sum and the np.multipy are used to add the element on the array [18] or multiply the elements [19]. Lastly, the np.abs is used to calculate the absolute value of an element [20].

The vector is defined by mathematicians and scientists as a quantity that depends on the direction and when there is direction then also a force, acceleration, and displacement are involved [21]. These values in real-life are can be used to study one's behavior just like the Philippine eagle flight plotter, by knowing the values researcher can come up with many conclusions. Vectors are also applied in playing basketball because the ball was being thrown into the air and this creates projectile and projectile can be represented with vectors.

Other examples of how vectors are used or other real-life situations that can be modeled using vectors are how the forces acting on a moving boat in the river. “The boat’s motor generates a force in one direction, and the current of the river of the river generates a force in

another direction." [22]. In this example, the direction of where the boat will go can be calculated when both the magnitude and direction of both forces from the current of the river and the force generated by the boat's motor are known. Lastly, momentum vectors are applied in playing billiard since the players are trying to predict what will happen when the balls collide with the specific force exerted to the ball, the ball passes some of its momentum to the other ball when it collides because of the impact [23].

References

[1] J. Brownlee, "Gentle Introduction to Vector Norms in Machine Learning", *Machine Learning Mastery*, 2020. [Online]. Available: https://machinelearningmastery.com/vector-norms-machine-learning/. [Accessed: 26- Sep- 2020].

[2]"matplotlib.pyplot.quiver — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.quiver.html. [Accessed: 26- Sep- 2020].

[3]"numpy.array — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.array.html. [Accessed: 26- Sep- 2020].

[4]"numpy.linalg.norm — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html#numpy.linalg.norm. [Accessed: 26- Sep- 2020].

[5]"numpy.arctan — NumPy v1.15 Manual", *Docs.scipy.org*, 2020. [Online]. Available: https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.arctan.html. [Accessed: 26- Sep- 2020].

[6]"numpy.degrees — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.degrees.html. [Accessed: 26- Sep- 2020].

[7]"matplotlib.pyplot.figure — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.figure.html. [Accessed: 26- Sep- 2020].

[8]"matplotlib.pyplot.title — Matplotlib 3.1.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.title.html. [Accessed: 26- Sep- 2020].

[9]"matplotlib.pyplot.xlim — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.xlim.html. [Accessed: 26- Sep- 2020].

[10]"matplotlib.pyplot.ylim — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.ylim.html. [Accessed: 26- Sep- 2020].

[11]"matplotlib.pyplot.xlabel — Matplotlib 3.1.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.xlabel.html. [Accessed: 26- Sep- 2020].

[12]"matplotlib.pyplot.ylabel — Matplotlib 3.1.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.ylabel.html. [Accessed: 26- Sep- 2020].

[13]"matplotlib.pyplot.grid — Matplotlib 3.1.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.grid.html. [Accessed: 26- Sep- 2020].

[14]"matplotlib.pyplot.legend — Matplotlib 3.1.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.legend.html. [Accessed: 26- Sep- 2020].

[15]"matplotlib.pyplot.show — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.show.html. [Accessed: 26- Sep- 2020].

[16]"matplotlib.pyplot.savefig — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.savefig.html. [Accessed: 26- Sep- 2020].
[17]"numpy.zeros — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.zeros.html. [Accessed: 26- Sep- 2020].
[18]"numpy.sum — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.sum.html. [Accessed: 26- Sep- 2020].
[19]"numpy.multiply — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.multiply.html. [Accessed: 26- Sep- 2020].
[20]"numpy.absolute — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.absolute.html. [Accessed: 26- Sep- 2020].
[21]"Scalars and Vectors", *Grc.nasa.gov*, 2020. [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/vectors.html. [Accessed: 27- Sep- 2020].
[22]"12: Vectors in Space", *Mathematics LibreTexts*, 2020. [Online]. Available: https://math.libretexts.org/Bookshelves/Calculus/Book%3A_Calculus_(OpenStax)/12%3A_Vectors_in_Space. [Accessed: 27- Sep- 2020].
[23] T. Editors, "What Are Vectors, and How Are They Used?", *Scientific American*, 2020. [Online]. Available: https://www.scientificamerican.com/article/football-vectors/. [Accessed: 27- Sep- 2020].

Appendix

This is the GitHub Repository https://github.com/RovilSurioJr/Laboratory-2