Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 2

Plotting Vectors using NumPy and MatPlotLib

Submitted by:
Parco, Jovian Asher G.

Instructor:
Engr. Dylan Josh D. Lopez

October 3, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of vector operation and vector plotting using NumPy & MatPlotlib.

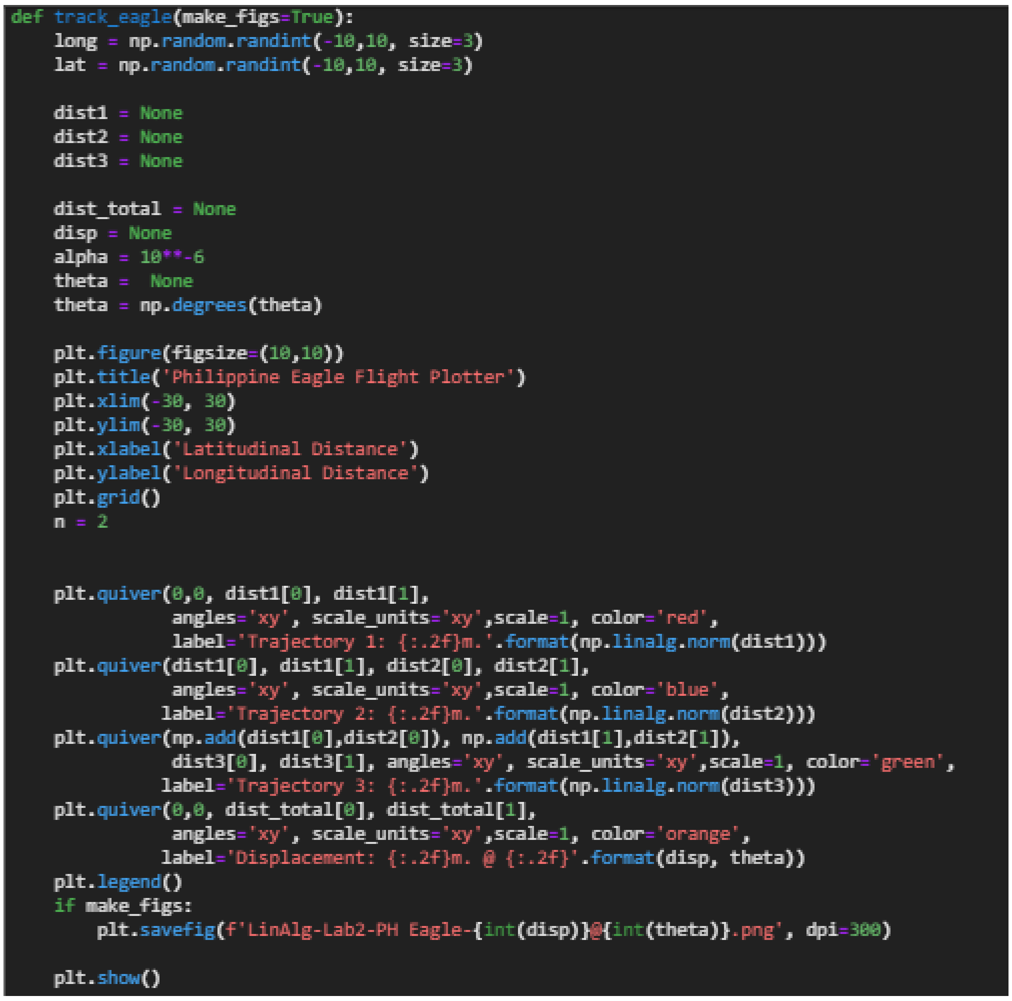
II. Methods

In this laboratory activity, the practices consist of three-vector manipulation using mathematical functions, different array() function & vector plotting. There are two ways of inputting mathematical functions in python using arithmetic symbols and NumPy mathematical functions; using arithmetic symbols is often used for short equations while NumPy mathematical functions are used for a more complex equation. Simple array functions like splicing, traversing arrays are often used when dealing with vectors and vector plotting. For vector plotting, the MatPlotlib is a standard for a data scientist to visualize their data coherently.

The deliverables of the activity are plotting data, deriving & analyzing arithmetic equations. The researcher used MatPlotlib with a combination of NumPy for the vector’s operations to visualize the data given. The deriving and analyzing equations were done by solving the equation and understanding where they came from and NumPy libraries to show it in a code form

III. Results

The complete repository for the activity are available at Github(https://bit.ly/30iSIxb) Part 1 of the lab activity consists of fixing the code and fill in the necessary code to get the necessary output(Figure 2). The code(Figure 1) is for plotting an eagles flight with only three sets of longitude and latitude to determine the relative displacement from the eagle to their nest.



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

    dist1 = None
    dist2 = None
    dist3 = None

    dist_total = None
    disp = None
    alpha = 10**-6
    theta =  None
    theta = np.degrees(theta)

    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()
```

Figure 1 Given problem in part 1

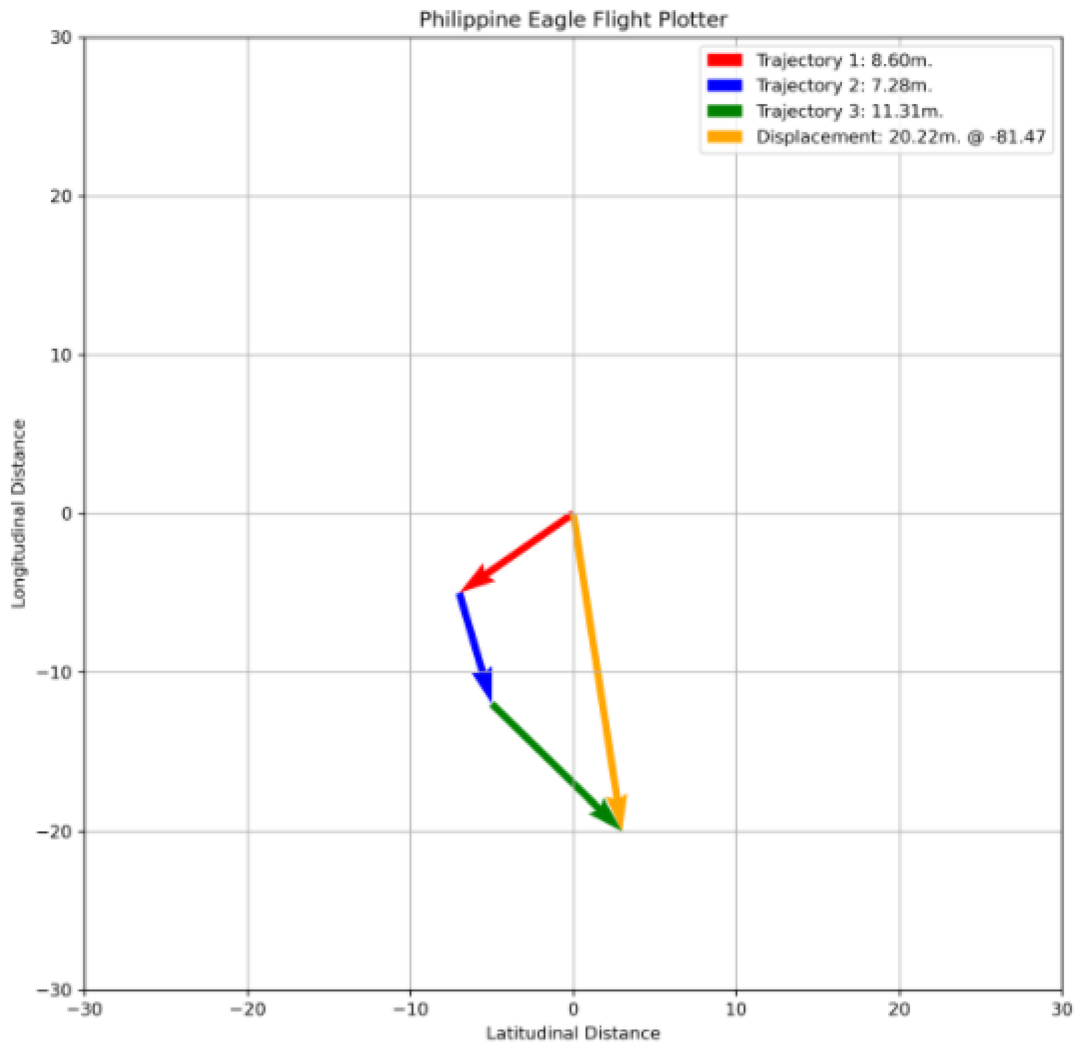
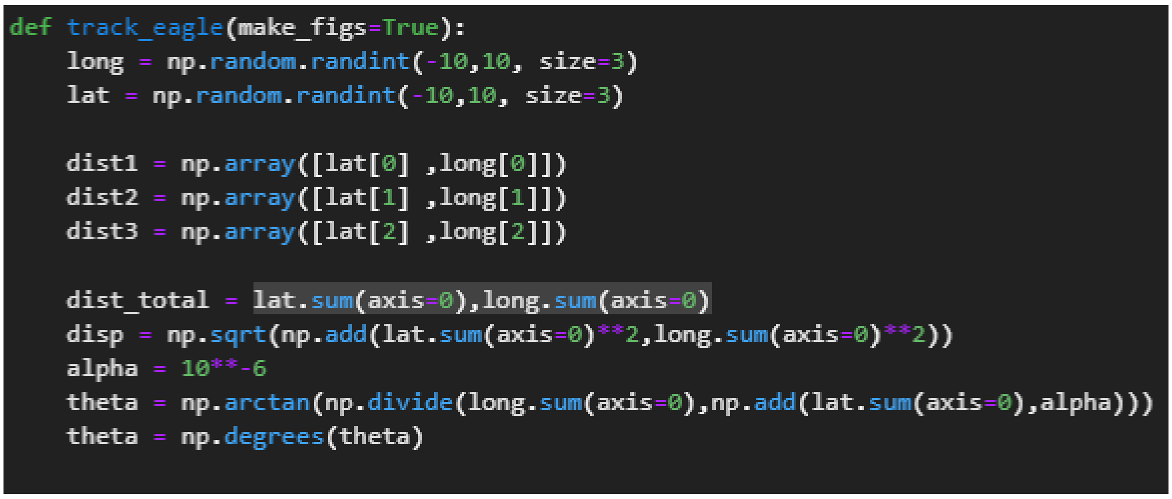


Figure 2 Example of desired output in part 1

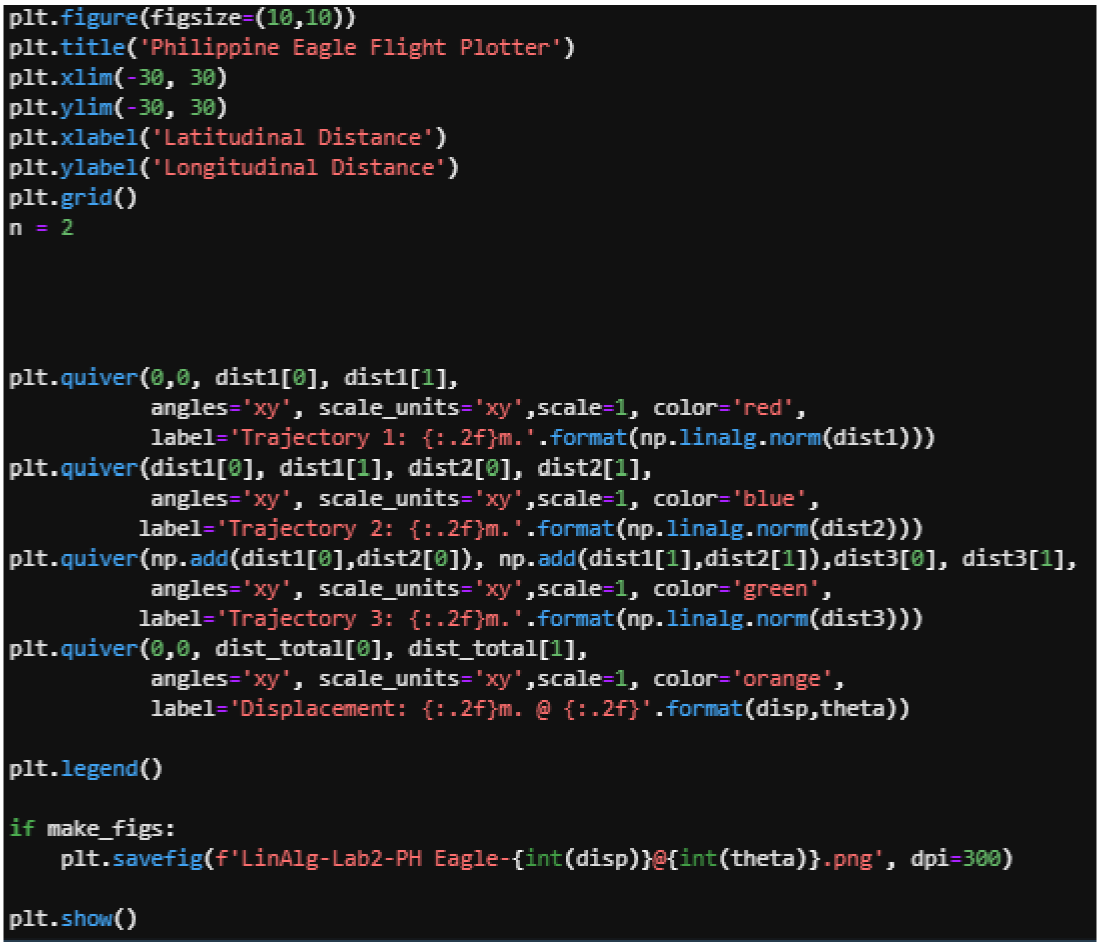
In the given code(Figure 1), the task is to replace the code "None" with a more appropriate code to get the desired output(Figure 2). The first 2&3 lines of the code use the random() function to get a random number from -10 to 10 with a size of 6, which is the data used for the longitude and latitude of the eagle flight path. The line from 6 to 8 in (Figure3) uses the array() function to declare the eagle's e longitude and latitude in every flight.

To calculate the total distance traveled, the researcher uses the equation "$\text{dist}_{\text{total}} = (\text{long}_{\text{total}})\hat{x} + (\text{lat}_{\text{total}})\hat{y}$" which convert to "lat.sum(axis=0),long.sum(axis=0)" for the Python implementation, The sum() function gives the Sum of array elements over a given axis [1]. To calculate the magnitude displacement, the researcher uses the given equation " $\text{disp} = \sqrt{\text{dist}_x{}^2 + \text{dist}_y{}^2}$ which convert to "np.sqrt(np.add(lat.sum(axis=0)**2,long.sum(axis=0)**2))" for the python implementation, the sqrt() function Returns a non-negative square-root of an array, element-wise [1].



```
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_total = lat.sum(axis=0),long.sum(axis=0)
    disp = np.sqrt(np.add(lat.sum(axis=0)**2,long.sum(axis=0)**2))
    alpha = 10**-6
    theta = np.arctan(np.divide(long.sum(axis=0),np.add(lat.sum(axis=0),alpha)))
    theta = np.degrees(theta)
```

Figure 3 Data solution and variable declaring for part 1

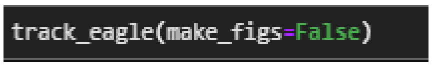


```
plt.figure(figsize=(10,10))
plt.title('Philippine Eagle Flight Plotter')
plt.xlim(-30, 30)
plt.ylim(-30, 30)
plt.xlabel('Latitudinal Distance')
plt.ylabel('Longitudinal Distance')
plt.grid()
n = 2




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()
```

Figure 4 Plotting and saving figure for part 1

The alpha(α) variable used within theta function to calculate the angle relative to the nest, and the equation used by the researcher "$\theta = \arctan\left(\frac{y}{x+\alpha}\right)$" which converted "np.arctan(np.divide(long.sum(axis=0),np.add(lat.sum(axis=0),alpha)))" where arctan() function is the trigonometric inverse tangent, element-wise[1]. And the add() function is used to add arguments element-wise[1]. And the last line in (Figure 3) uses the degree() function which convert angles from radians to degrees.

The researcher uses the Matplotlib for plotting(Figure 3); the first line indicated in code(Figure 4) uses the figure() function, which holds all plot elements with the arguments for the width and height of the figure with a specified aspect ratio[2]. The next line is the title() function, which is the plot's title and can be placed above the center's axes, flush with the left edge, and flush with the right edge [3]. The 3 to 6 line of the code in (Figure 4) uses x & y lim() function and x & y label() function; the lim() function sets the limit of the plot, and the adjacent of that is the label() which labels the function, the last two lines of code in plotting properties are grid() function and n; the grid() function configure the grid lines [4] and the "n" is used to indicate what vector shape will be inputted.



```
track_eagle(make_figs=False)
```

Figure 5 Running the function for part 1

The next line is the quiver() function which plots the arrow direction of the vector, with the size of the arrow related to the magnitude of the vector [5] and the arguments for quiver() function is the starting point of the arrow & the head location of the arrow, next argument is the properties of the arrow and the label of it, The legend() function displays all the labels of the quiver() function. The next line is an if statement, and the parameters "make_figs" if

(Figure5) issue will create a snapshot of the plot, the final line in (Figure 4) is show() function, which the researchers us show all the plots and labels.

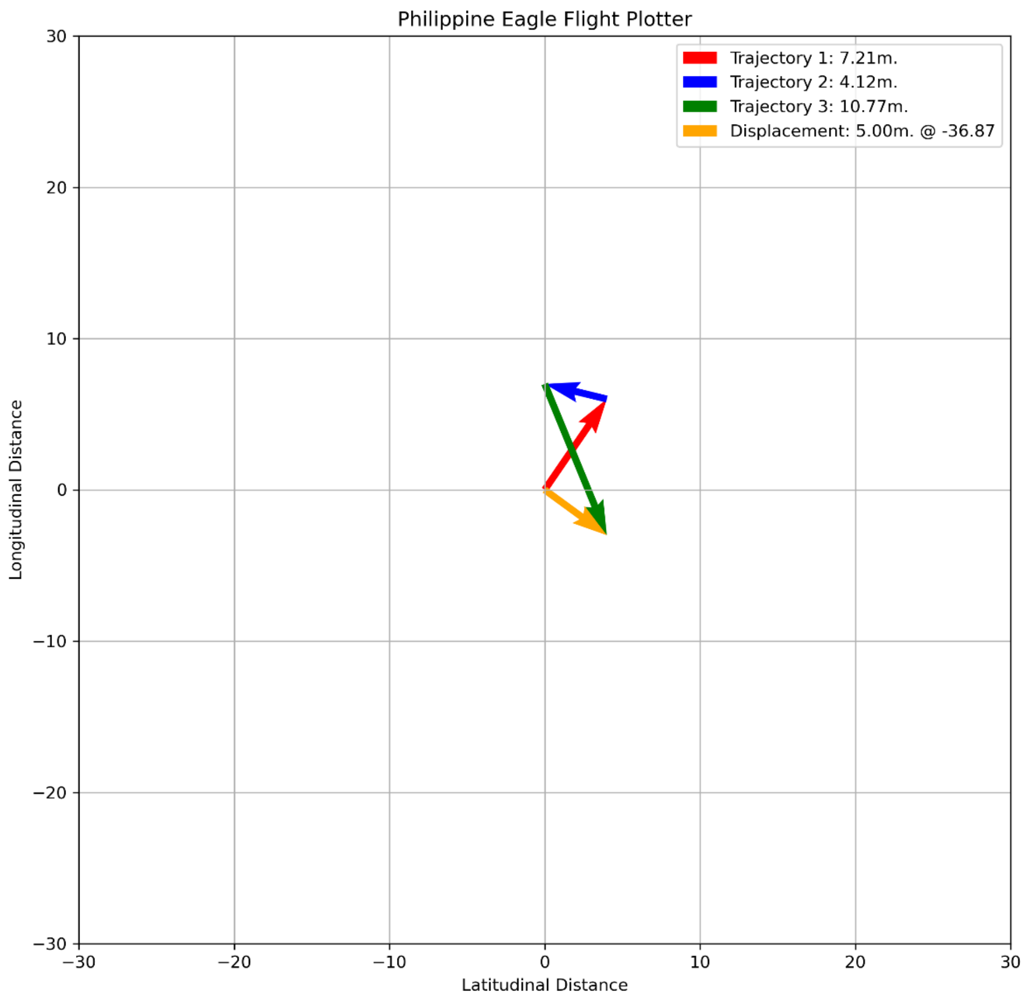


Figure 6 Plot result #1 in part 1

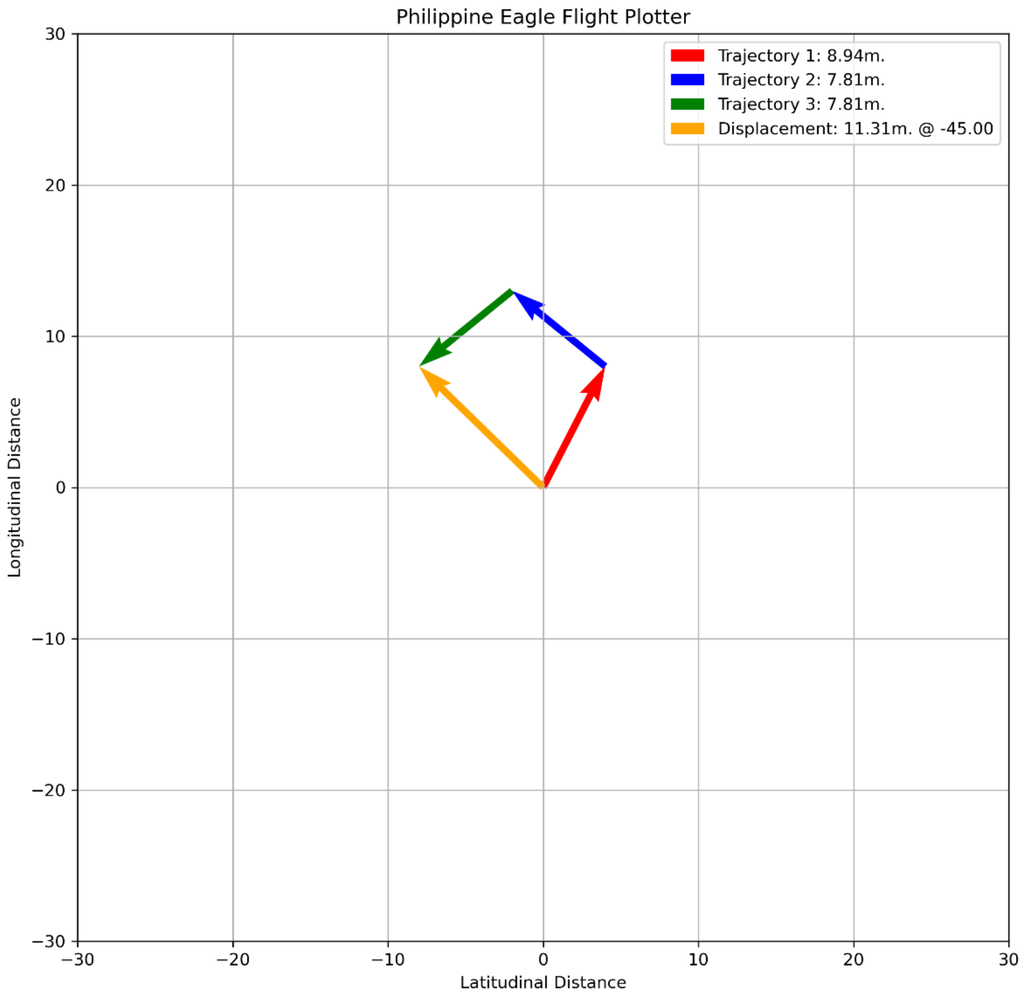


Figure 7 Plot result #2 in part 1

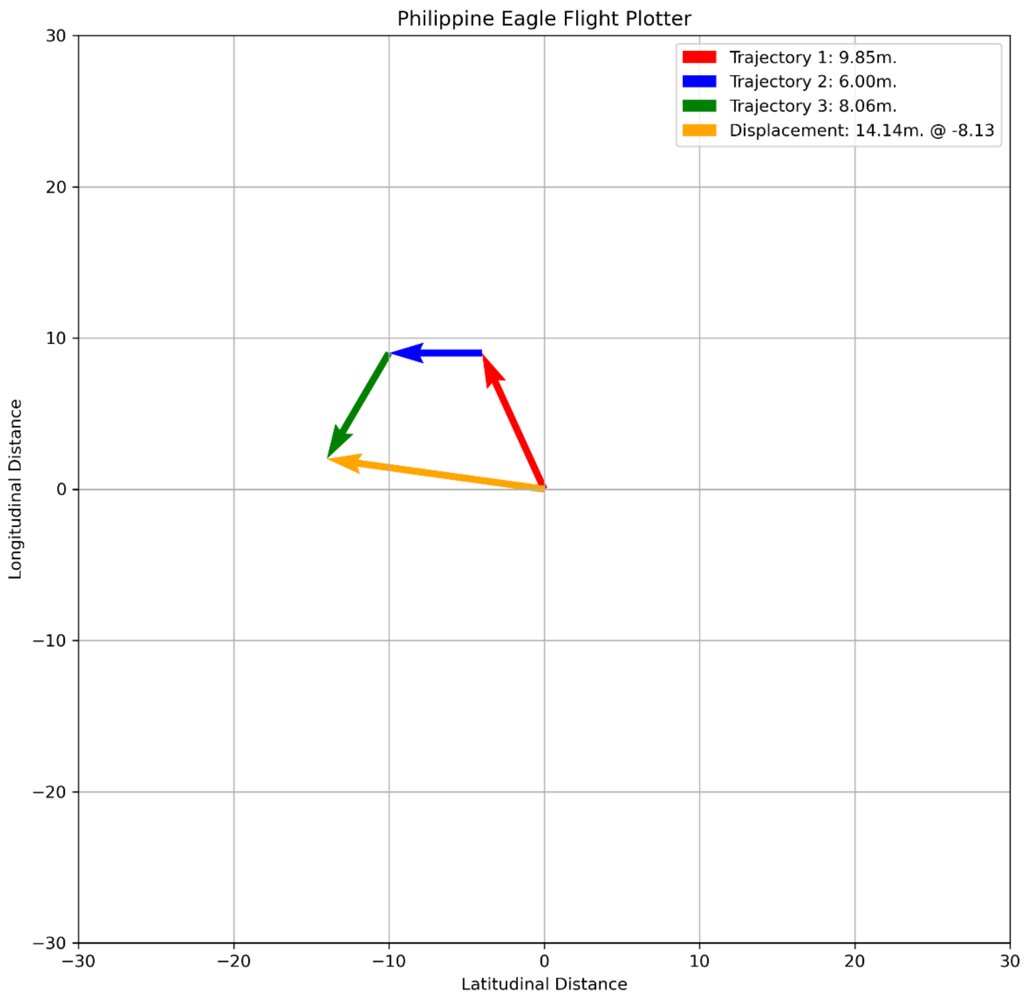


Figure 8 Plot result #3 in part 1

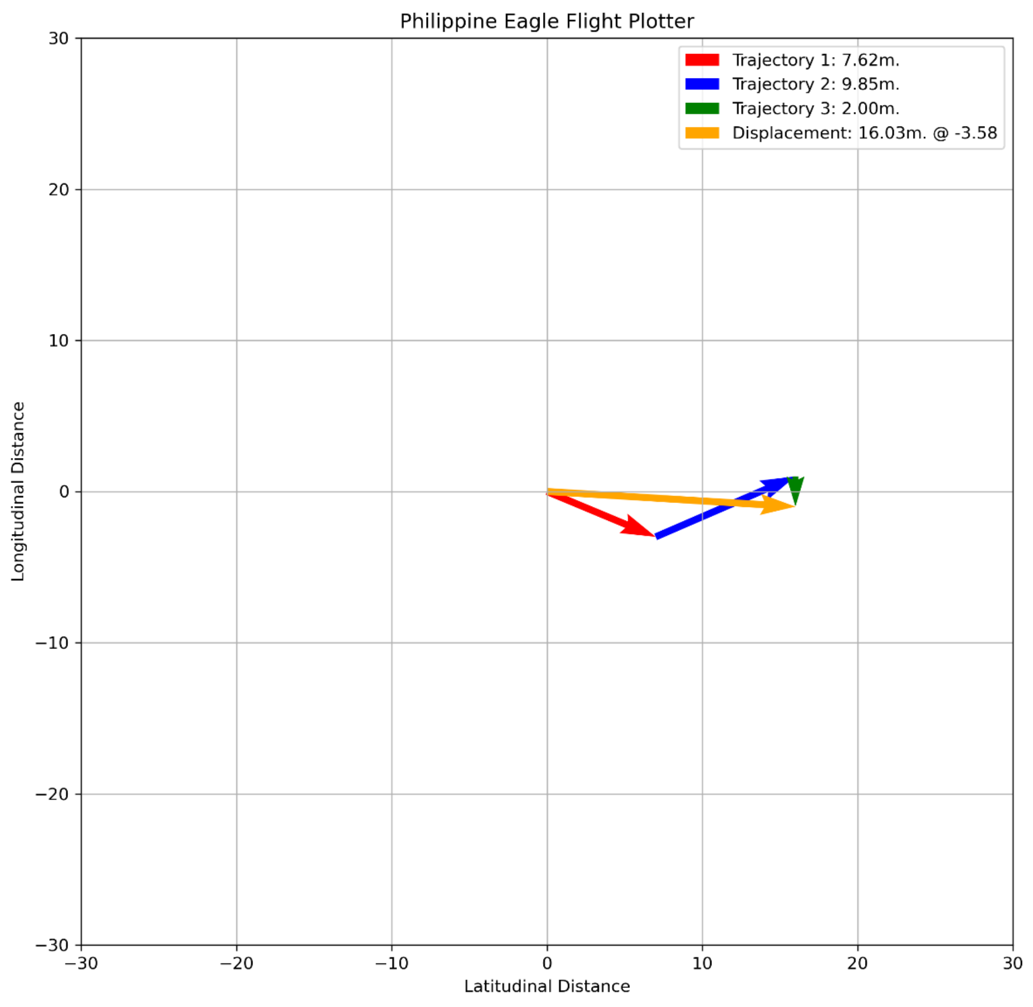
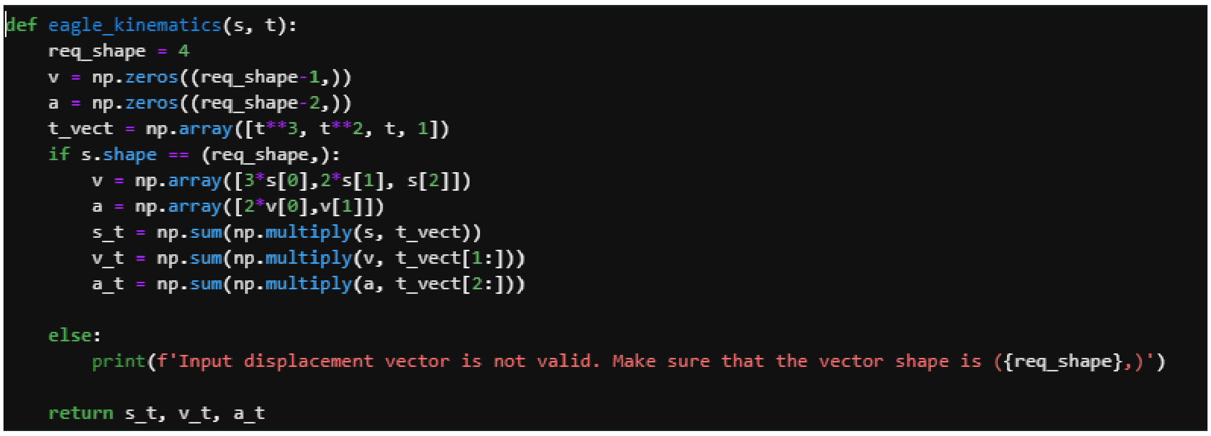


Figure 9 Plot result #4 in part 1

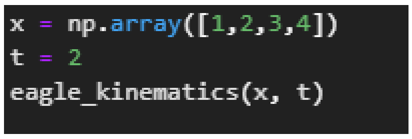
(Figure 6-8) Is the output of Philippine Eagle Flight Plotter, due to the use of random() function the flight pattern is also random.

Part 2 of the lab activity is a reverse engineering activity. The researcher needs to be analyzed and recreate a code with proper documentation, comments, and vague variable declarations for other researchers or clients to understand and properly use it.



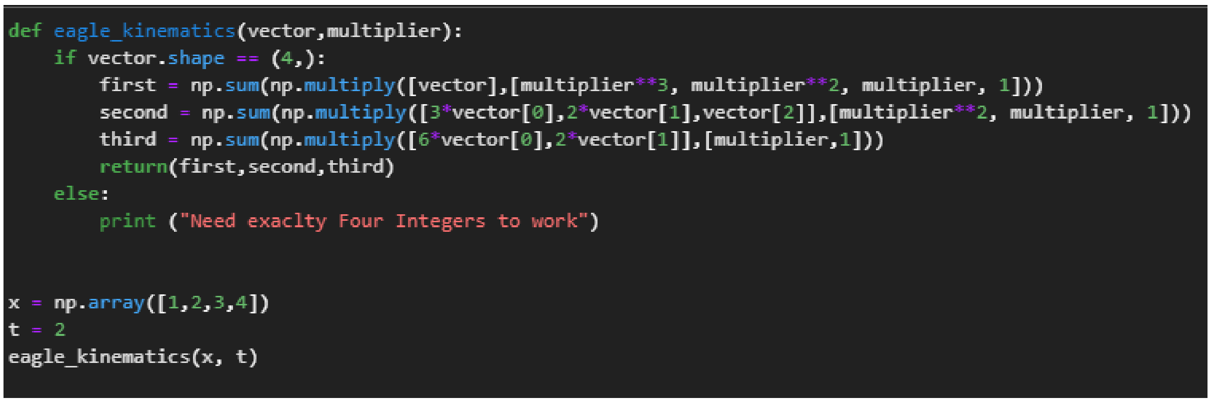
```
def eagle_kinematics(s, t):
    req_shape = 4
    v = np.zeros((req_shape-1,))
    a = np.zeros((req_shape-2,))
    t_vect = np.array([t**3, t**2, t, 1])
    if s.shape == (req_shape,):
        v = np.array([3*s[0],2*s[1], s[2]])
        a = np.array([2*v[0],v[1]])
        s_t = np.sum(np.multiply(s, t_vect))
        v_t = np.sum(np.multiply(v, t_vect[1:]))
        a_t = np.sum(np.multiply(a, t_vect[2:]))

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

    return s_t, v_t, a_t
```

Figure 10 Given problem in part 2



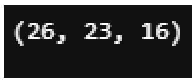
```
x = np.array([1,2,3,4])
t = 2
eagle_kinematics(x, t)
```

Figure 11 Given input in part 2



```
def eagle_kinematics(vector,multiplier):
    if vector.shape == (4,):
        first = np.sum(np.multiply([vector],[multiplier**3, multiplier**2, multiplier, 1]))
        second = np.sum(np.multiply([3*vector[0],2*vector[1],vector[2]],[multiplier**2, multiplier, 1]))
        third = np.sum(np.multiply([6*vector[0],2*vector[1]],[multiplier,1]))
        return(first,second,third)
    else:
        print ("Need exaclty Four Integers to work")


x = np.array([1,2,3,4])
t = 2
eagle_kinematics(x, t)
```

Figure 12 Solution for part2



```
(26, 23, 16)
```

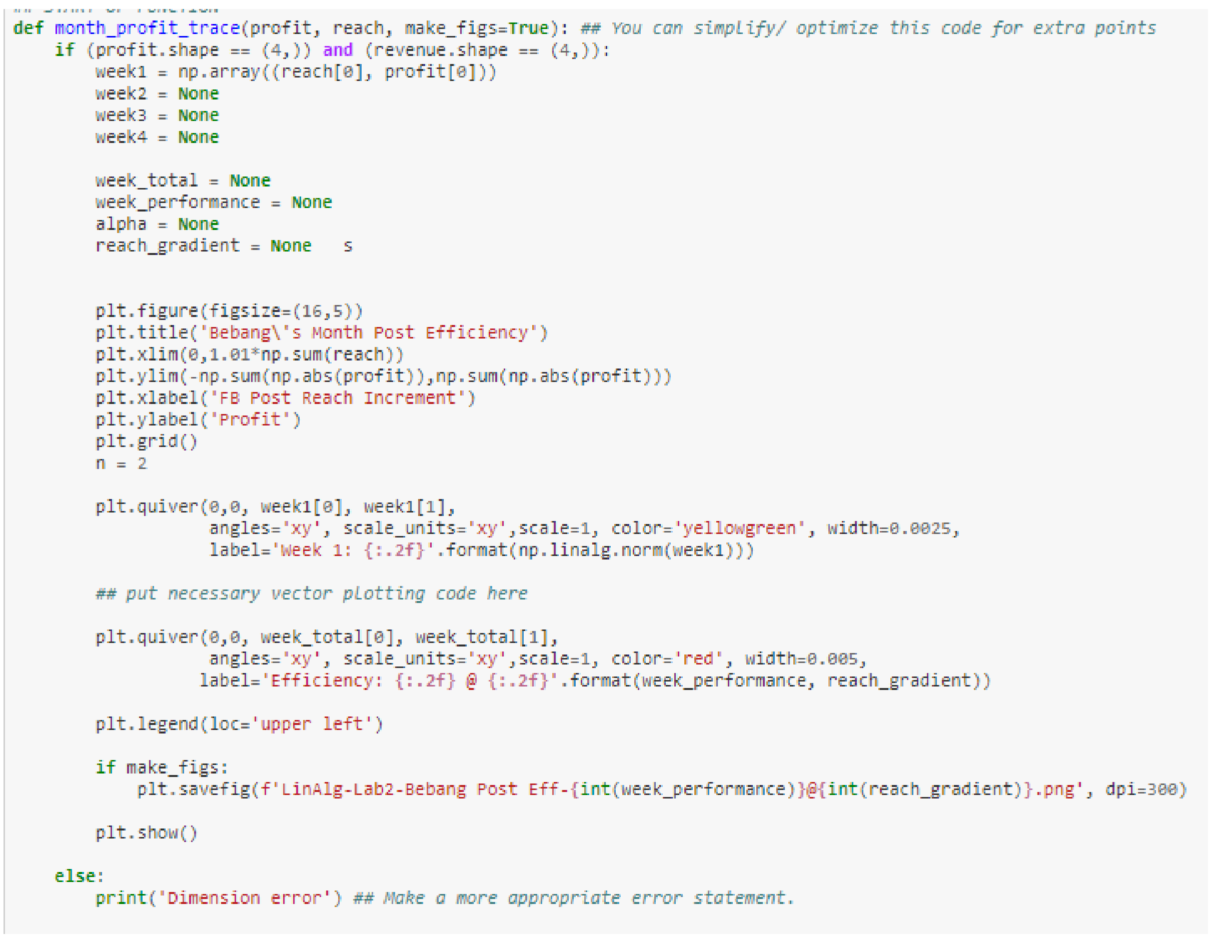
Figure 13 Output for part 2

For part 2 lab activity, the researcher need to use pen and paper to properly optimized and understand the given problem (Figure 10) and recreate it with proper comment and variable, the input(Figure 11) need one set of number with a size of 4 and one number as a multiplier. The researcher cames out with this formula to solve the three equations.

$$\left(\hat{A}x^3\right) + \left(\hat{b}x^2\right) + (\hat{c}x) + \left(\hat{d}\right) = FirstEq$$
$$(3\hat{a}x^2) + (2\hat{b}x) + (\hat{c}\ ) = SecondEq$$
$$(6\hat{a}\ x) + (2\hat{b}\ ) = ThirdEq$$

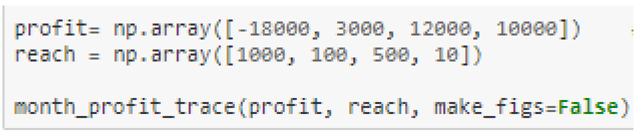
This applied to the function & recreate transform code (Figure 12), where it checks first for the “vector shape” if it is four; if not, it does not function and will output (Figure 13) with the answer from every equation combine as a list.

For part 3 of the lab activity, the researcher needs to create a plotter that computes profit and online reach using the Part 1 solution as or reference to create the plot.



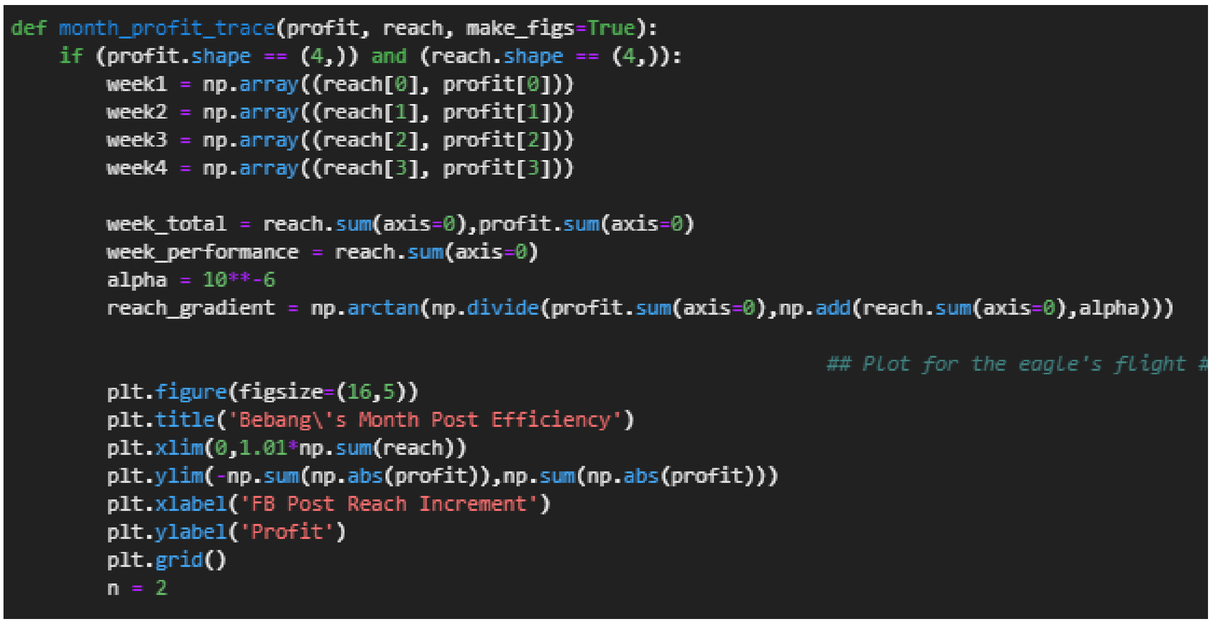
```
def month_profit_trace(profit, reach, make_figs=True): ## You can simplify/ optimize this code for extra points
    if (profit.shape == (4,)) and (revenue.shape == (4,)):
        week1 = np.array((reach[0], profit[0]))
        week2 = None
        week3 = None
        week4 = None

        week_total = None
        week_performance = None
        alpha = None
        reach_gradient = None   s


        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)))

        ## put necessary vector plotting code here

        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 Given problem for part 3



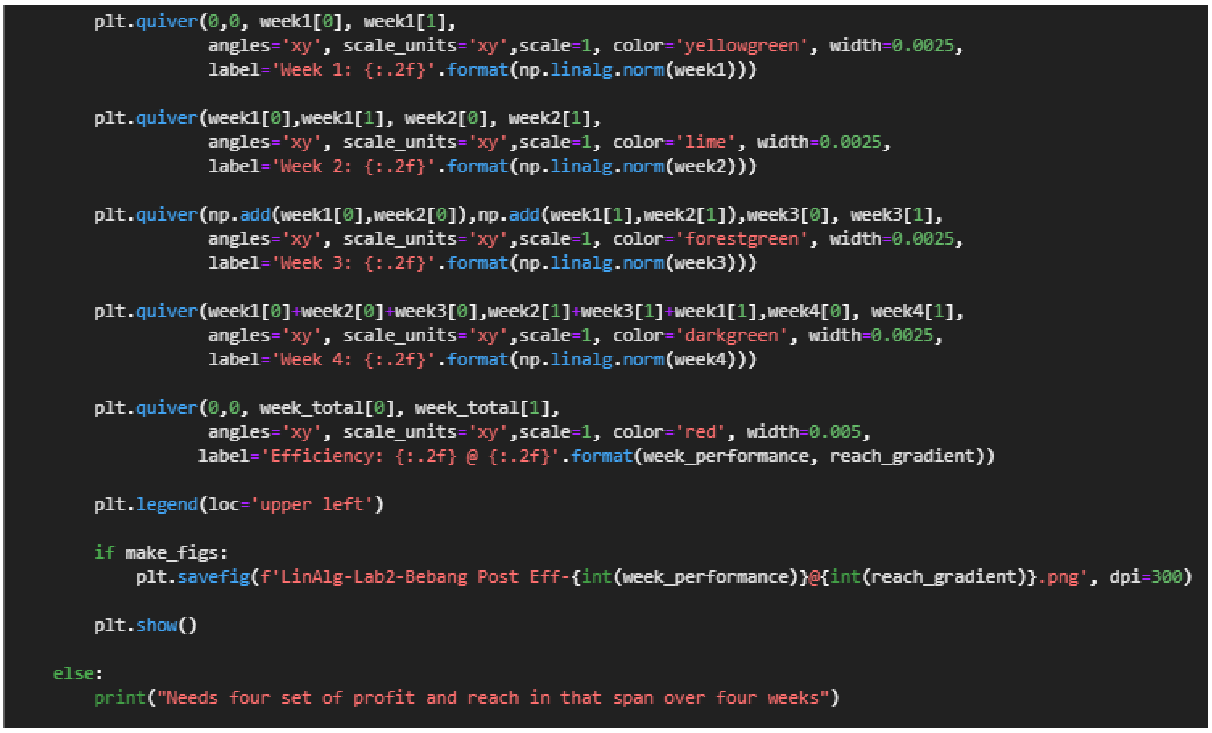
```
profit= np.array([-18000, 3000, 12000, 10000])
reach = np.array([1000, 100, 500, 10])

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

Figure 15 Given input for part 3



```
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_total = reach.sum(axis=0),profit.sum(axis=0)
        week_performance = reach.sum(axis=0)
        alpha = 10**-6
        reach_gradient = np.arctan(np.divide(profit.sum(axis=0),np.add(reach.sum(axis=0),alpha)))

                                                    ## Plot for the eagle's flight #
        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 16 Data solution and variable declaring for part 3



```
        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='lime', 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='forestgreen', width=0.0025,
                   label='Week 3: {:.2f}'.format(np.linalg.norm(week3)))

        plt.quiver(week1[0]+week2[0]+week3[0],week2[1]+week3[1]+week1[1],week4[0], week4[1],
                   angles='xy', scale_units='xy',scale=1, color='darkgreen', 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} @ {:.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("Needs four set of profit and reach in that span over four weeks")
```

Figure 17 Plotting and saving figure for part 3

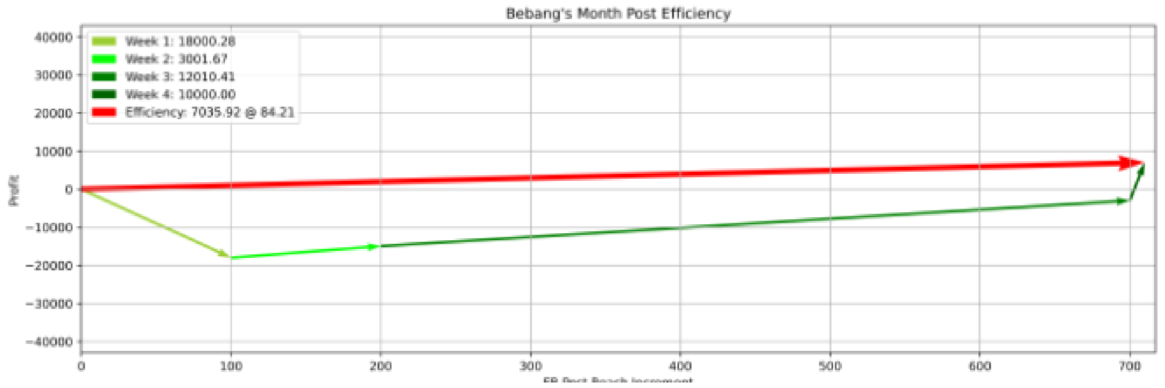


Figure 18 Expected output in part 3

Similar to part 1(Figure 1), the problem is to create a plot (Figure 18) using the code (Figure 14); the researcher reuses all the code and functions in part 1 problem; the only difference is in the input and the plotting. In part 1, the input uses the random() function which gives a random number, while in part 3 the input is given (Figure 15), and the plot properties change the figsize() function to “16,5” & the limit of the figure is set with respect to the given input.

IV. Conclusion

All of the problems in this lab activity are pretty challenging because the libraries are not familiar with the researcher like the Numphy and MatPlotlib, but their documentation is straight forward like when the researcher was searching for a function description, it is apparent & concise while also giving an example. The function used in this lab activity is the sum() function gives the Sum of array elements over a given axis [1], the sqrt() function returns a non-negative square-root of an array, element-wise [1], arctan() function is the trigonometric inverse tangent, element-wise[1], add() function is used to add arguments element-wise[1]. In the MatPlotlib, the researcher used the figure() function holds all plot elements with the arguments for the figure's width and height with a specified aspect ratio[2], the grid() function configures the grid lines [4], the quiver() function, which plots the arrow direction of the vector, with the size of the arrow related to the magnitude of the vector [5], and many more.

As a student like myself, vectors can help me view my academic purposes like knowing how much time you study in a week & make a plot in it. Another example is managing money; knowing how much I spend in a month and where that money is wasted can help you budget. Vector can also help when traveling; knowing the shortest path can save money and time. Last but not least, vectors can help visualize physics problems by showing a simulated space or the object created by computer software with the help of vectors.

References

[1]"Mathematical functions — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/routines.math.html?highlight=operation. [Accessed: 30- Sep- 2020].

[2]"matplotlib.figure — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/figure_api.html?highlight=fig#module-matplotlib.figure. [Accessed: 30- Sep- 2020].

[3]"matplotlib.pyplot.title — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.title.html?highlight=title#matplotlib.pyplot.title. [Accessed: 30- Sep- 2020].

[4]"matplotlib.pyplot.grid — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.grid.html?highlight=grid#matplotlib.pyplot.grid . [Accessed: 30- Sep- 2020].

[5]"matplotlib.quiver — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/quiver_api.html?highlight=quiver#module-matplotlib.quiver. [Accessed: 30- Sep- 2020].