Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 2

Plotting Vectors using NumPy and Matplotlib

Submitted by:

Chipongian, John Patrick Ryan J.

.

Instructor:

Engr. Dylan Josh D. Lopez

October 6, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of using NumPy and Matplotlib in plotting vectors by using different functions and processes in it.

II. Methods

In this activity, we focus on using the NumPy library for plotting the vectors that would be computed. In computing the vectors, we've used many functions to get its value. We used both mathematical processes of NumPy to compute the values of it element wise and we also used arithmetic operators for a simpler equation. After computing the values of the vector arrays, we assign it to its plot points using the Matplotlib. Using Matplotlib, we were able to visualize the values computed.

Using the computed values, we were able to plot the values using Matplotlib and able to visualize it. To achieve this, we derived formulas using arithmetic methods and NumPy functions. After deriving the formulas, we formulate the values of vector to its respective place so that we would be able to plot it.

III. Results

In the first part of the activity, we were given already done codes and was tasked to input necessary codes to complete the function. First, we arrayed the distances of the eagle's flight, which is the latitude and longitude of it. The values inputted as the distances of the eagle's flight were randomized since we do not have eagle tracking data.

```
long = np.random.randint(-10,10, size=3)
lat = np.random.randint(-10,10, size=3)



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

Figure 1 Values of distances traveled by the eagle

After having the latitude and longitude of the eagle's flight, we then computed for the final vector which is the resultant vector. To plot the resultant vector, we first computed the needed values. We computed its total distance with the given formula which is: $dist_{total} = (long_{total})x + (lat_{total})y$. It is then derived in code which is "dist_total" which is shown in figure 2. Then we compute the displacement of the resultant vector with the formula: $disp = dist_x^2 +$

$dist_y^2$. It is then derived in code which would be “disp” and is shown in figure 2. Lastly, the angle of the resultant vector is computed using the formula: $\theta = arctan(y/\alpha x)$. It is ten derived in code which would be “theta” and is shown in figure 2. The alpha, which is a constant and has a value of 10^{-6}, is defined is “alpha = 10**-6” as there is no constant value of alpha in the python library. And lastly, the computed angle using the theta formula is converted from rad to degrees with the function degrees().

```
dist_total = long.sum(0), lat.sum(0)
disp = np.sqrt(np.add(lat.sum(0)**2,long.sum(0)**2))
alpha = 10**-6
theta =  np.arctan((np.divide((lat[0] + lat[1] + lat[2]),(long[0] + long[1] + long[2]))) + alpha )
theta = np.degrees(theta)
```

Figure 2 Computation for the Resultant vector

The next lines are the ones that are used to plot the points of the eagle flight vectors. These were done using the functions of matplotlib. In figure 3, the first lines of codes are what gives the properties of the graph. The function figure() gives the size of the graph. The title() is the title that is show in the graph. The xlim() and ylim() is the value limitations in axis x and y. The xlabel() is the label for the x-axis while the ylabel() is the label of the y-axis. [1] The grid() function is what sets the visibility of grids in the figure. Lastly, the “n = 2” indicates the dimension of the vectors that would be shown in it.

The function quiver() is the function that plots the points given in its syntax. In the figure 3, the first quiver() function is the plot of the first trajectory, the next would be the second trajectory, then next would be the third trajectory and lastly the fourth would be the resultant vector. The next function that were used is legend(), which prints the legend section in the figure that would show. Next is the statement that would save the figure which was done using the function savefig(). And last is the function show(), which would show the figure when the function is called.

```
## 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()
```

Figure 3 Plotting of the latitude and longitude of the distances and displacement of the eagle

Now to run the function, the code in figure 4 is used to call the function.

```
track_eagle(make_figs=True)
```

Figure 4 Calling of the function

The following is the figures made using this function:

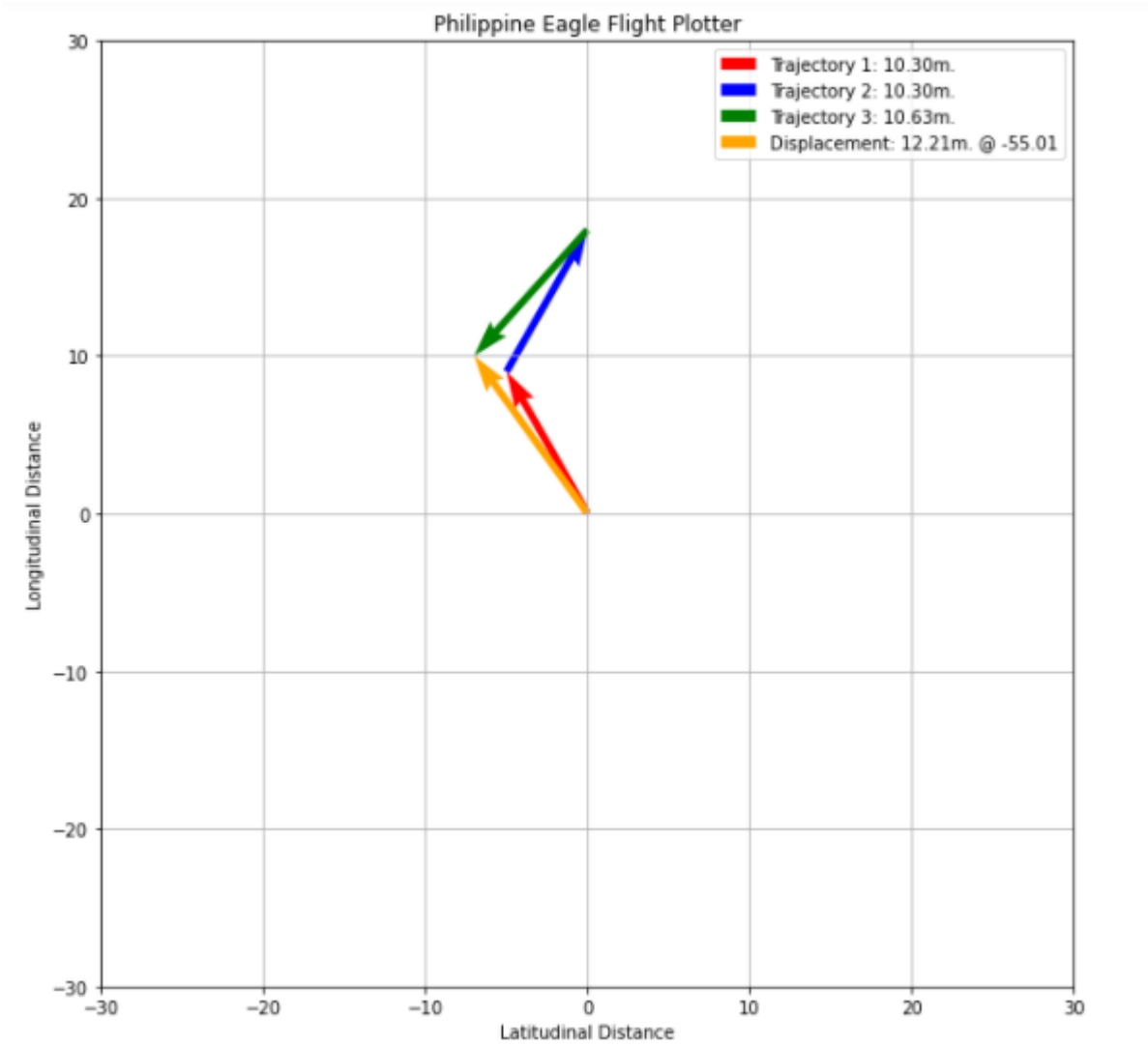

Figure 5 First figure plotted

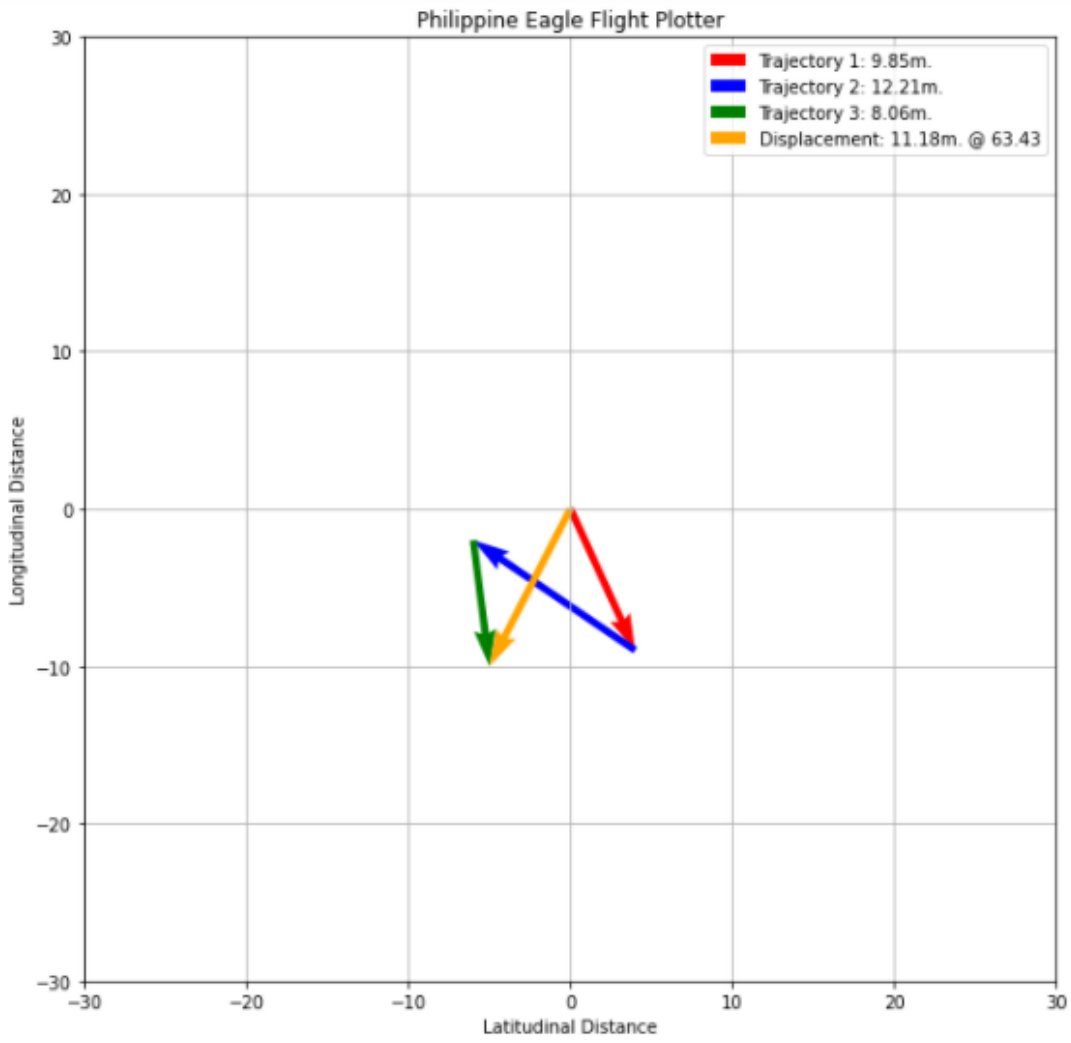

Figure 6 Second figure plotted

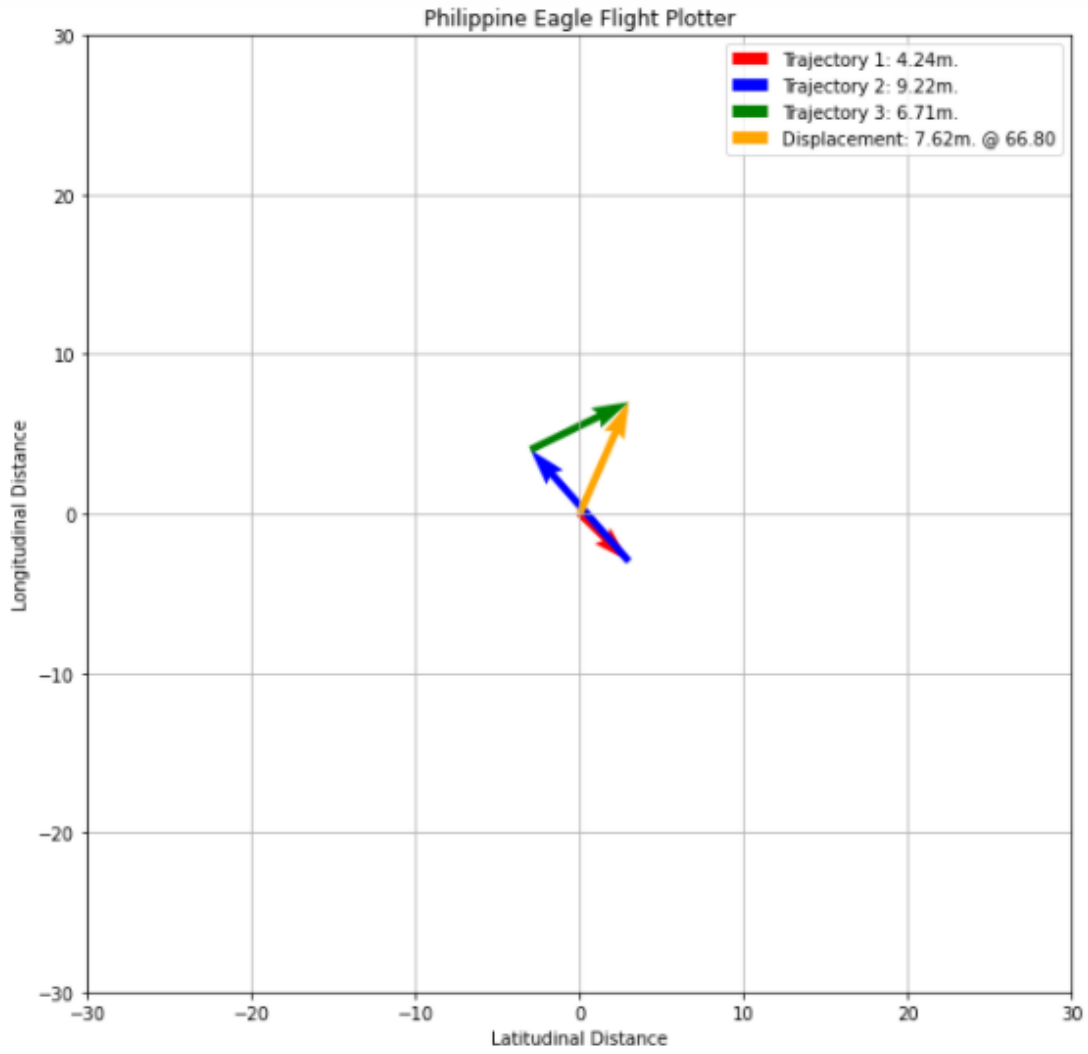

Figure 7 Third figure plotted

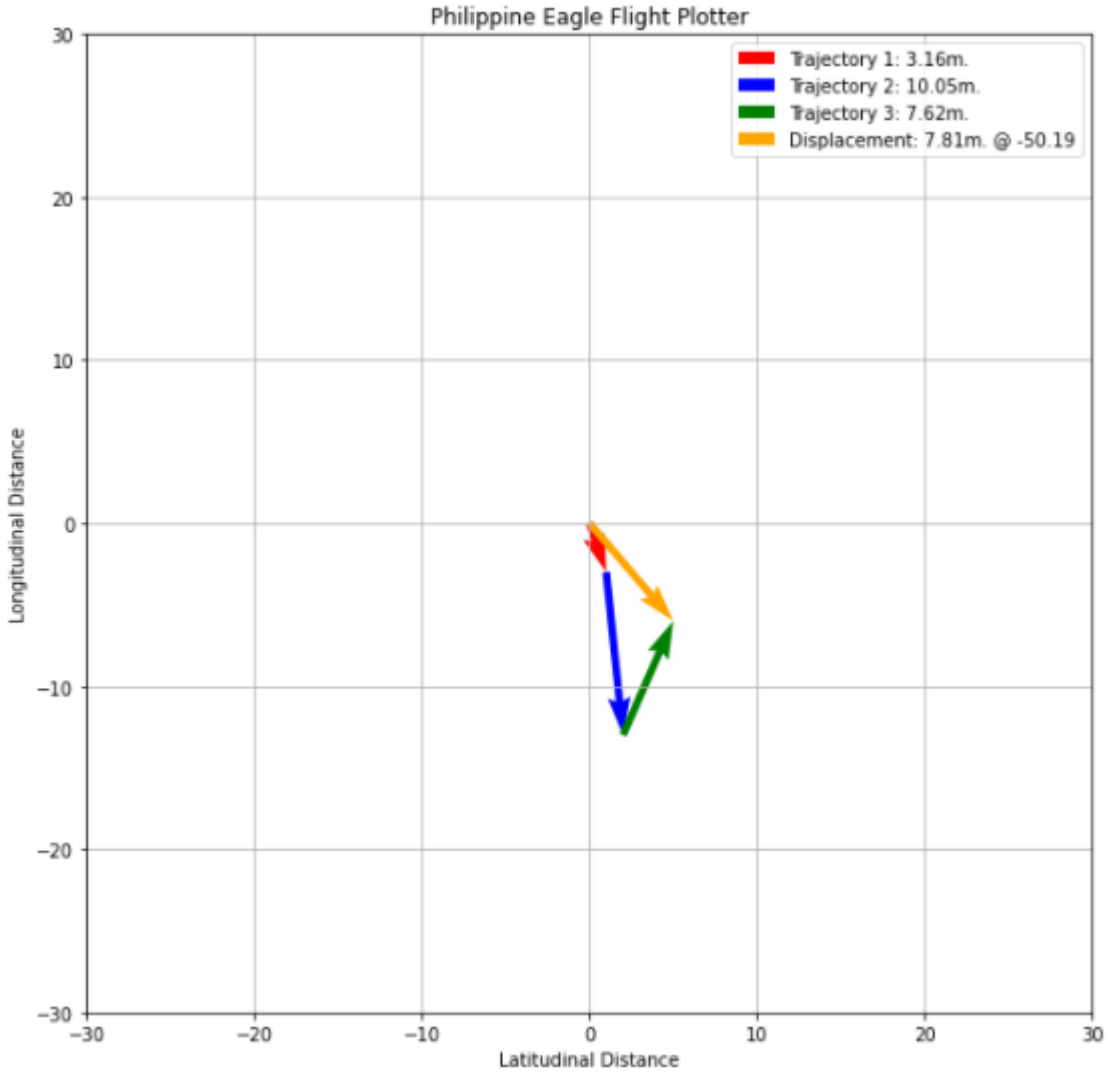

Figure 8 Fourth figure plotted

The next part of the activity, we were tasked to add appropriate documentation, proper comments, and to change the variable names in to a more understandable names to the given codes to us.

```
def eagle_kinematics(displacement, time):
    req_shape = 4
    velocity = np.zeros((req_shape-1,))
    acceleration = np.zeros((req_shape-2,))
    total_vect = np.array([time**3, time**2, time, 1])
    if displacement.shape == (req_shape,):
        velocity = np.array([3*displacement[0],2*displacement[1], displacement[2]])
        acceleration = np.array([2*velocity[0],velocity[1]])
        displacement_total = np.sum(np.multiply(displacement, total_vect))
        velocity_total = np.sum(np.multiply(velocity, total_vect[1:]))
        acceleration_total = np.sum(np.multiply(acceleration, total_vect[2:]))

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

    return displacement_total, velocity_total, acceleration_total

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

Figure 9 Codes for part 2 of the activity

By deconstructing the code given to us, we were able to determine the function of the it. The function of variable “v” is to array the values of velocity while the variable “a” arrays the value of acceleration. The variables “s” and “t” is changed into displacement and time respectively. Using the variable displacement, the values of the array of velocity is computed in code and after computing it, it was then used for the computation for the values of acceleration. Now having all the values, the total displacement, total velocity, and total acceleration, which is the displacement_total, velocity_total, and acceleration_total, is then computed using the values we have computed earlier. After the computations, the values of the total displacement, velocity, and acceleration is then returned and is printed in array.

The last and next part of the activity given to us, we were tasked to create a program that would monitor the increase and decrease of profit with respect to the reach of the FB post online.

We based our codes to the first part of the activity and added necessary codes for the computation for the final vector, plotting of the points to the figure and printing error code if there is wrong with the inputs.

```
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]))
```

Figure 10 Array of the values of weeks 1-4

```
week_total = np.array([np.add(week1[0] + week2[0], week3[0] + week4[0]),np.add(week1[1] + week2[1], week3[1] + week4[1])])
week_performance = np.sqrt([np.add(week_total[0]**2, week_total[1]**2)])
week_performance = np.linalg.norm(week_performance)
alpha = 10**-6
reach_gradient =  np.arctan([np.divide(np.add(week1[1] + week2[1], week3[1] + week4[1]),
                                       np.add(week1[0] + week2[0], week3[0] + week4[0])) + alpha ])   # Computation for an
reach_gradient = np.linalg.norm(np.degrees(reach_gradient))

## Plotting the the Bebang's month vectors
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 11 Computation for the resultant vector and plotting of the figure

```
    ## Plotting of the vectors of weeks 1 to 4
    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='blue', 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='pink', width=0.0025,
               label='Week 3: {:.2f}'.format(np.linalg.norm(week3)))
    plt.quiver(np.add(week1[0] + week2[0], week3[0]), np.add(week1[1] + week2[1], week3[1]), week4[0], week4[1],
               angles='xy', scale_units='xy',scale=1, color='brown', width=0.0025,
               label='Week 4: {:.2f}'.format(np.linalg.norm(week4)))

    ## Plotting of the resultant vector
    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()

## Error Statements
else:
    print('Invalid number of profit values')
    print('Invalid number of reach values')
```

Figure 12 Plotting of weeks 1-4 and the resultant vector and saving the showing the figure and printed if an error occurred

Using the codes in figure 13, we would be to call the function month_profit_trace and to input the values needed. With the inputs needed present, the figures would be now showed.

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

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

Figure 13 Inputs for the function and calling of the function

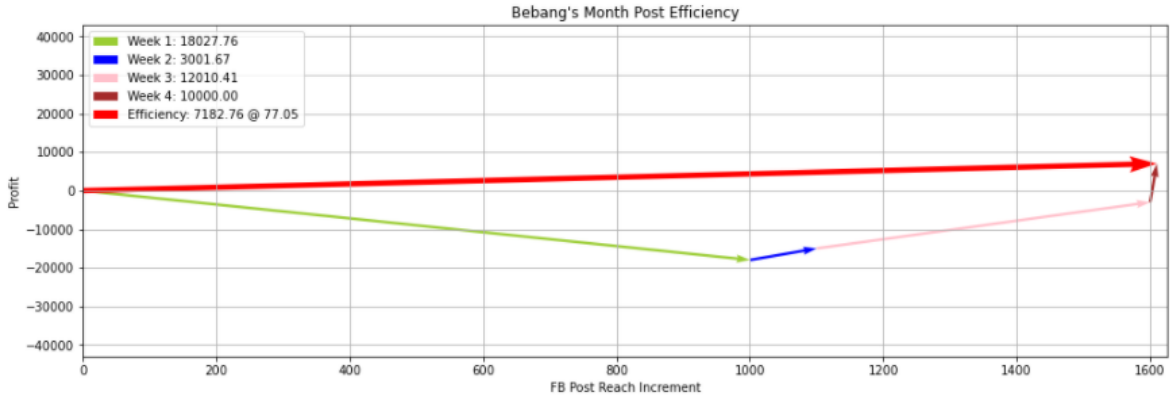

Figure 14 Plotted Figure for part 3

IV. Conclusion

This laboratory activity has been done with the use of the libraries of NumPy and matplotlib and all of the functions in it. In the first part of the activity, [2] one of the functions used is sum() which would return the sum of all the elements in array in the specified axis.

[3] A similar function is add() which would add two arrays element wise. [4] Another function used is sqrt() which is used to find the positive square root of an array, element-wise. [5] The next function used in this activity is the mathematical function arctan(), which helps in computing the inverse tangent of all array elements. [6] Another function is degree() which is used in determining the angle of the resultant vector in the activity as it converts the value from radian to degrees. All the functions that was presented was used for mathematical computation of the points. The other functions that was used are functions from the matplotlib libraries. The other functions that were inputted was used to show the figure. These functions were [7] figure(), to create a new figure, title(), xlim(), ylim(), xlabel(), ylabel(), [1] grid(), which sets the visibility of the grid in the figure.

Vectors can be useful in our lives. It can help us in determining what choices we should make. Example of this is in online shopping, when we want to buy something but do not trust its seller, we can check its sales by which would consist of vector of sales. Another example of this is when travelling and is using GPS, your current position and the place you want to go to would form a line as your guide while travelling.

References

[1] Tutorialspoint.com. 2020. *Matplotlib - Grids - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/matplotlib/matplotlib_grids.htm> [Accessed 8 October 2020].

[2] Gupta, M., 2020. *Numpy.Sum() In Python - Geeksforgeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/numpy-sum-in-python/> [Accessed 8 October 2020].

[3] Sayantan, J., 2018. *Numpy.Add() In Python - Geeksforgeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/numpy-add-in-python/#:~:text=add()%20function%20is%20used,not%20same%2C%20that%20is%20arr1.> [Accessed 8 October 2020].

[4] GeeksforGeeks. 2019. *Numpy.Sqrt() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/numpy-sqrt-in-python/> [Accessed 8 October 2020].

[5] Gupta, M., 2018. *Numpy.Arctan() In Python - Geeksforgeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/numpy-arctan-python/> [Accessed 8 October 2020].

[6] Numpy.org. 2020. *Numpy.Degrees — Numpy V1.19 Manual*. [online] Available at: <https://numpy.org/doc/stable/reference/generated/numpy.degrees.html> [Accessed 8 October 2020].

[7] Matplotlib.org. 2020. *Matplotlib.Pyplot.Figure — Matplotlib 3.3.2 Documentation*. [online] Available at: <https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.figure.html> [Accessed 8 October 2020].