Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 2

Plotting Vector using NumPy and MatPlotLib

Submitted by:
Sustento, Myke Alvin

Instructor:
Engr. Dylan Josh D. Lopez

September 30, 2000

I. Objectives

This laboratory activity aims to implement the principles and techniques of Python as a programming language and be familiar with its libraries such as matplotlib and NumPy for numerical and scientific programming. It also aims to visualize vectors and perform simple vector operations through Python programming.

II. Methods

The practices of the activity consist of using the libraries of matplotlib and NumPy. The libraries are used to plot vectors and graphs using values given on arrays. The deliverables of the activity are to use given formula in finding vector quantities and create graphs using the libraries given by the matplotlib and NumPy in order the create a visualization of the computations.

III. Results

This activity intends to apply the libraries of matplotlib and NumPy. The reason for using the libraries of matplotlib and NumPy is to create and visualize graphs created by vector operations.

```
%matplotlib inline
def track_eagle(make_figs=True):
    long = np.random.randint(-10,10, size=3) ## since we don't actually have eagle tracking data we will use
    lat =  np.random.randint(-10,10, size=3) ## random integers for this activity. These two lines will produce
                                             ## a vector with shape of (3,) describing the distances for the
                                             ## eagle's flight.

    dist1 = np.array([long[0],lat[0]])                      ## setups the array for each vector for the eagle's flight.
    dist2 = np.array([long[1],lat[1]])
    dist3 = np.array([long[2],lat[2]])

    dist_total = dist1 + dist2 + dist3                        ## formula for the resultant vector for the eagle's flight.
    disp  = np.linalg.norm(dist_total)                        ## formula for the magnitude of the displacement
    alpha = 10**-6
    theta = np.arctan((dist_total[1])/(dist_total[0] + alpha))     ## computes for the angle of the displacement.
    theta = np.degrees(theta)                                      ## converts theta from rad to deg.

    ## 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


    ### Self-discovery code block (Refer to guide question 2 for the task)

    # parameters of quiver([X, Y], U, V, [C], **kw)
        # [X, Y] defines the arrow location
        # U, V defines the arrow direction
        # C sets the color
        # units - To plot vectors in the x-y plane, with u and v having the same units as x and y,
            ## use ``angles='xy', scale_units='xy', scale=1``.


    #red arrow (trajectory 1)
    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)))
    #blue arrow (trajectory 2)
    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)))
    #green arrow (trajectory 3)
    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)))
    #orange arrow (total displacement)
    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()    #plt.legend() place the legend on the axes


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

    plt.show()   #plt.show() displays all figures

## END OF FUNCTION
```

Figure 1 Code for Graded Cell 1

Graded cell 1 is about tracking the flight of an eagle. Since there are no actual data provided for the eagle flight, random integers were used to produce vectors describing the

distances of the eagle's flight. The given for graded cell 1 was the randomly generated longitudinal distance and latitudinal distance of the eagle's flight. To set up the arrays for each vector, np.array() function was used to create arrays [1] for the long and lat variables. Use $Distance\ total = distance_1 + distance_2 + distance_3$ to get the resultant vector. Afterward, use np.linalg.norm()on the total distance to get the magnitude of the displacement. Using np.arctan() to get the trigonometric inverse tangent[2] and the formula $\theta = (\frac{Y}{X+\alpha})$, the value of theta can be attained. To remove the possibility of division by zero on the theta, set alpha as 10^{-6}. Using np.degrees(), theta could be converted from rad to deg [3].

Plt.quiver() function, as the name suggests, is the arrow that plots the 2D field of arrows[4]. There is 4 plt.quiver() function given in graded cell 1 which are trajectory 1 (red arrow), trajectory 2 (blue arrow), trajectory 3 (green arrow), and total displacement (orange arrow) in that order. Based on the documentation given by Jupyter notebook, the parameters that were used in graded cell 1 was quiver ([X, Y], U, V, [C], and the units). X, Y serves to define the starting point of the arrow location. The x,y in trajectory 1, 2, 3, and total displacement is [0,0],[dist1[0], dist1[1]], [(np.add(dist1[0],dist2[0]),np.add(dist1[1],dist2[1])], and [0,0] in that order. Np.add() is used to add arguments element-wise[4]. The reasoning for these x,y values can be seen on the graph which can be seen that the trajectory 1, 2, and 3 are interconnected with 0,0 as the origin. These values will manifest 3 interconnected vectors that will result in a total displacement from the origin to the end of trajectory 3. U, V is defined as the location of the arrow. U, V will serve as the direction in which the trajectory 1, 2, 3, and the total displacement will point towards. The u,v in trajectory 1, 2, 3 and total displacement are [dist1[0], dist1[1]], [dist2[0], dist2[1]], [dist3[0], dist3[1]], and [dist_total[0], dist_total[1]] in that order. The u,v values are created by the randomly generated values in long and lat variables. C or, in this case, color is self-explanatory. It sets the color of the trajectories for easier analysis. According to the documentation given by jupyter notebook, to plot vectors in the x-y plane, with u and v having the same units as x and y, use ``angles='xy', scale_units='xy', scale=1``. plt.legend() serves to place the legends on the axis given by the label and lastly, plt.show() displays all the figures as said by jupyter notebook.

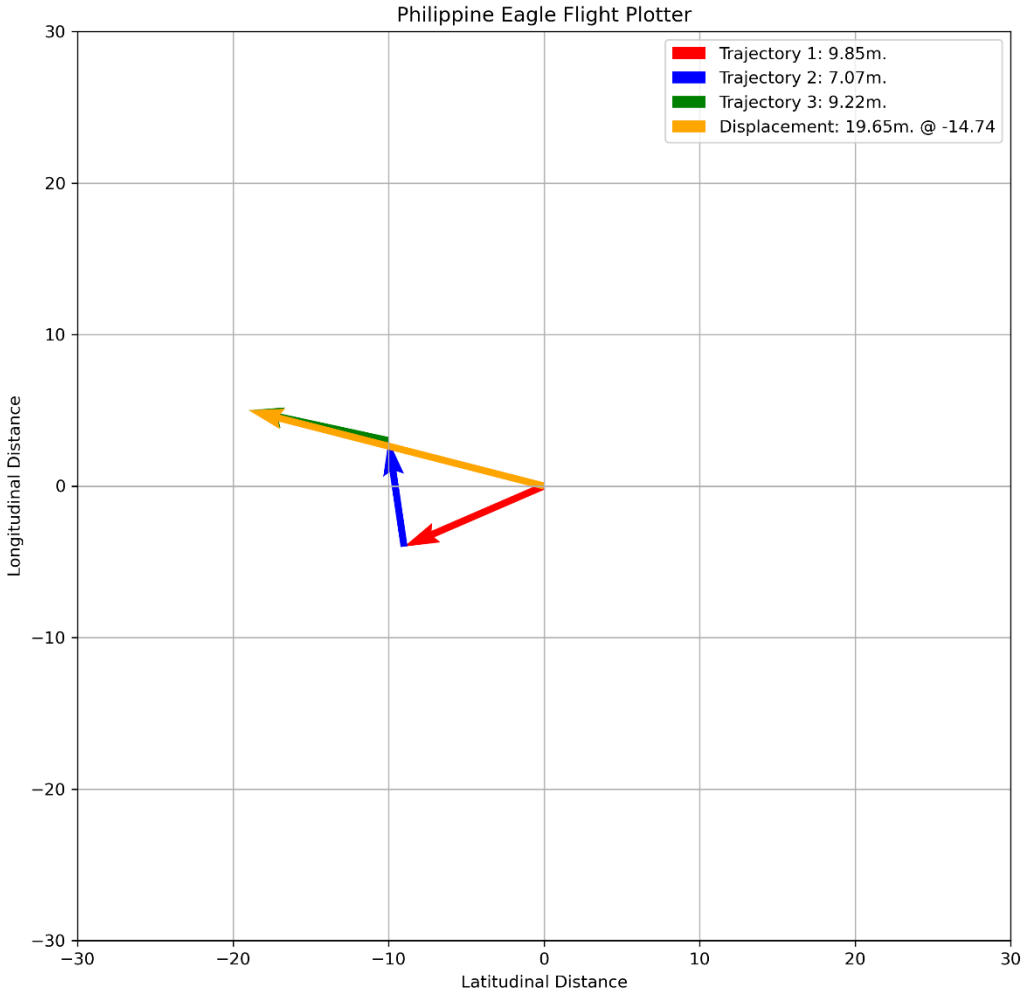

Figure 2 One of the Outputs of Graded Cell 1

Figure 2 shows one of the randomly generated outputs of graded cell 1. This shows the result of the code created in figure 1. The graph shows the trajectory 1, 2, 3, and the total displacement with its theta.

```
def eagle_kinematics(position, time):
    req_shape = 4
    velocity = np.zeros((req_shape-1,)) # array of zeroes [0,0,0]
    acceleration = np.zeros((req_shape-2,)) # array of zeroes [0,0]
    total_vector = np.array([t**3, t**2, t, 1]) # array [8,4,2,1]
    if position.shape == (req_shape,):
        velocity = np.array([3*position[0],2*position[1], position[2]]) # array [6,2,3]
        acceleration = np.array([2*velocity[0],velocity[1]]) # array [12,2]
        position_total = np.sum(np.multiply(position, total_vector)) #  np.multiply([8,4,2,1],[2,1,3,2]) = np.sum[16,4,6,2] = 28
        velocity_total = np.sum(np.multiply(velocity, total_vector[1:])) #  np.multiply([4,2,1],[6,2,3]) = np.sum[24,4,3] = 31
        acceleration_total = np.sum(np.multiply(acceleration, total_vector[2:])) #  np.multiply([2,1],[12,2]) = np.sum[24,2] = 26
    else:
        print(f'Input displacement vector is not valid. Make sure that the vector shape is ({req_shape},)')

    return position_total, velocity_total, acceleration_total
```

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

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

Figure 3 Graded Cell 3

Figure 3 shows the graded cell 3 of the laboratory activity 2. The given was a code that lacks documentation, comments, and has vague variable declarations. The function aims to create a kinematic for the movement of the eagle hence the name eagle_kinematics. The parameter of the function is position and time since the given input below was x and t. The return statements want to show the position_total, velocity_total, and acceleration_total. To get these values, use the np.multiply() on their corresponding parameters and then use np.sum() to get the sum of array elements over a given axis [21]. The result would be 28, 31, and 26.

```
## START OF FUNCTION
def month_profit_trace(profit, reach, make_figs=True):
    if (profit.shape == (4,)) and (reach.shape == (4,)): #checks if the number of profit and shape are 4 (4 because there are 4 weeks in a month)
        week1 = np.array((reach[0], profit[0])) #creates an array for the weekly with reach and profit as the elements
        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          #computes the resultant
        week_performance = np.linalg.norm(week_total)       #computes the magnitude
        alpha = 10**-6
        reach_gradient = np.arctan((week_total[1])/(week_total[0] + alpha))    ## computes for the angle of the displacement.
        reach_gradient = np.degrees(reach_gradient)                            ## converts theta from rad to deg.



        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()

        #Week 1
        plt.quiver(0,0, week1[0], week1[1],
                   angles='xy', scale_units='xy',scale=1, color='red', width=0.0025,
                   label='Week 1: {:.2f}'.format(np.linalg.norm(week1)))
        #Week 2
        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)))
        #Week 3
        plt.quiver((week1[0] + week2[0]), (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)))
        #Week 4
        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='yellowgreen', width=0.0025,
                   label='Week 4: {:.2f}'.format(np.linalg.norm(week4)))


        #Efficiency
        plt.quiver(0,0, week_total[0], week_total[1],
                   angles='xy', scale_units='xy',scale=1, color='orange', 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('Wait for the month to finish to calculation of the Monthly Post Efficiency')

## END OF FUNCTION
```

```
profit= np.array([-18000, 3000, 12000, 10000])        #Profit of Bebang's Online Business
## Original Values [-18000, 3000, 12000, 10000]
reach = np.array([1000, 100, 500, 10])           #Reach of Bebang's FB Posts
## Original Values [1000, 100, 500, 10]

month_profit_trace(profit, reach, make_figs=False) ## Toggle 'make_figs' during debugging or making documentations
```

Figure 4 Code for Graded Cell 3

Since the formulas and codes used in graded cells 1 and 3 are similar, the process of how the plot is similar. The parameters used in graded cell 3 are quiver ([X, Y], U, V, [C], and the units). The x,y represents the reach and profit in that order. The u,v defines as the week passed. The C still represents the color and the units used are angles, scale_units, scale, and width. Graded cells 1 and 3 have different representations of values but the process is the same.

```
profit= np.array([-18000, 3000, 12000, 10000])        #Profit of Bebang's Online Business
## Original Values [-18000, 3000, 12000, 10000]
reach = np.array([1000, 100, 500, 10])           #Reach of Bebang's FB Posts
## Original Values [1000, 100, 500, 10]

month_profit_trace(profit, reach, make_figs=False) ## Toggle 'make_figs' during debugging or making documentations
```

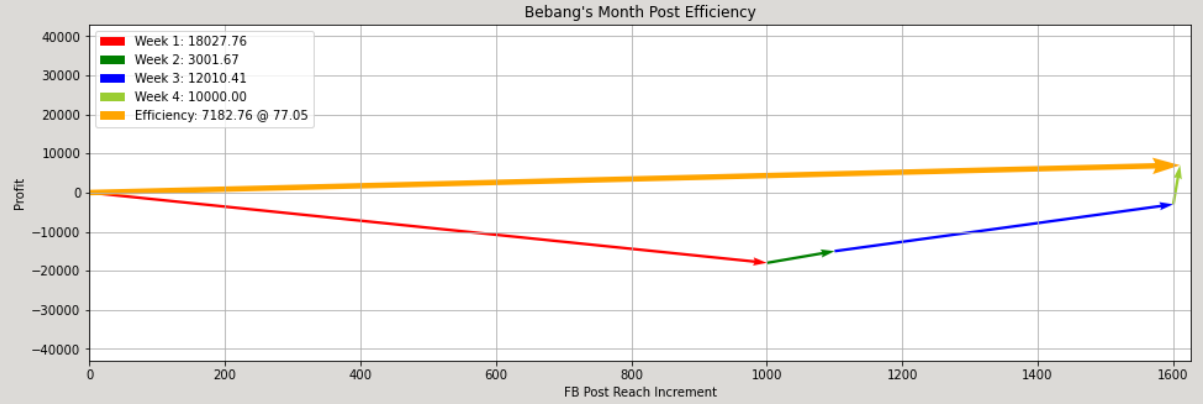

Figure 5 Output with the original values

Figure 5 shows Bebang's Month Post Efficiency with the original values given on the file for this activity. This will serve as the basis for the other results.

```
profit= np.array([-15000, 3000, 12000, 10000])        #Profit of Bebang's Online Business
## Original Values [-18000, 3000, 12000, 10000]
reach = np.array([ 1000 , 100 , 500  , 10])           #Reach of Bebang's FB Posts
## Original Values [1000, 100, 500, 10]

month_profit_trace(profit, reach, make_figs=True) ## Toggle 'make_figs' during debugging or making documentations
```

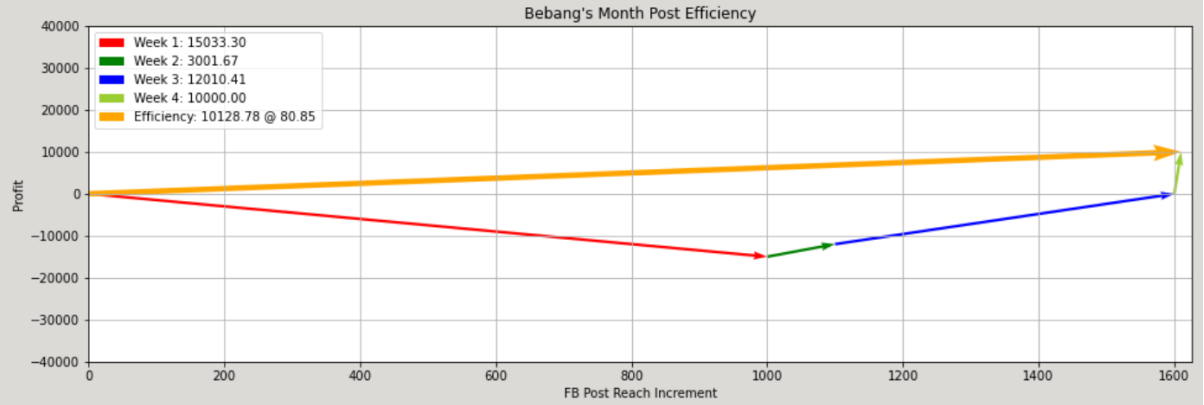

Figure 6 Scenario 1 with higher profit

Comparing figure 6 to figure 5, the magnitude of efficiency got higher because the profit got higher.

```
profit= np.array([-15000, 3000, 8000, 10000])        #Profit of Bebang's Online Business
## Original Values [-18000, 3000, 12000, 10000]
reach = np.array([ 1000 , 100 , 50  , 10])           #Reach of Bebang's FB Posts
## Original Values [1000, 100, 500, 10]

month_profit_trace(profit, reach, make_figs=True) ## Toggle 'make_figs' during debugging or making documentations
```

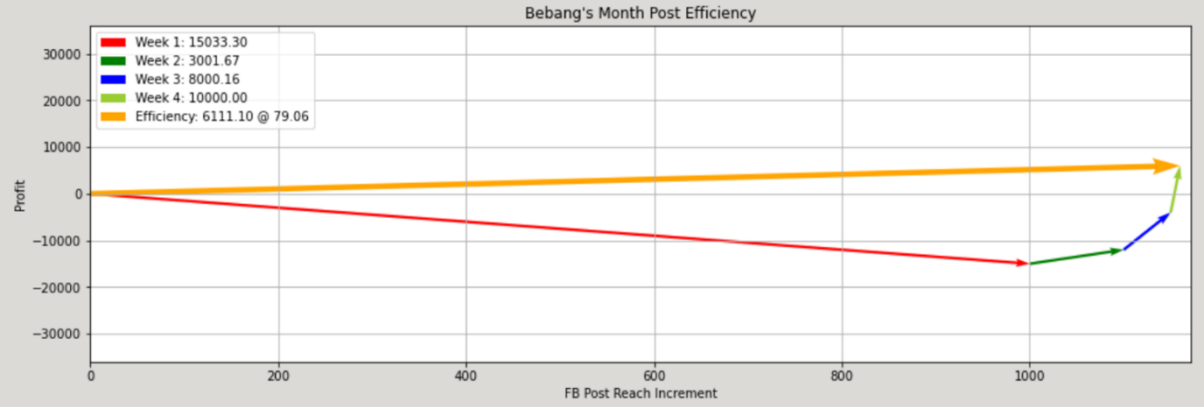

Figure 7 Scenario 2 with a decrease in reach

Comparing Figure 7 to figure 6, the magnitude of efficiency decreased because of the decrease in reach.

```
profit= np.array([-15000, 3000, 12000, 10000])        #Profit of Bebang's Online Business
## Original Values [-18000, 3000, 12000, 10000]
reach = np.array([ 1000 , 100 , 50  , 10])           #Reach of Bebang's FB Posts
## Original Values [1000, 100, 500, 10]

month_profit_trace(profit, reach, make_figs=True) ## Toggle 'make_figs' during debugging or making documentations
```

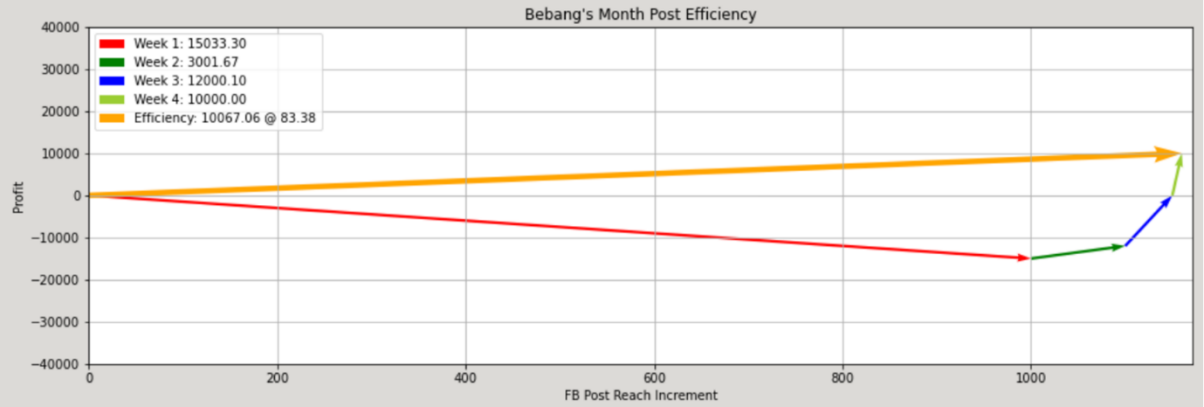

Figure 8 Scenario 3 with a decrease in profit

Comparing Figure 8 to figure 7, the magnitude of efficiency decreased because of the decrease in profit.

IV. Conclusion

The functions that I used in this laboratory exercise was numpy.array(), numpy.arctan(), numpy.degree(), matplotlib.pyplot.quiver(), numpy.add(), and numpy.linalg.norm(). Numpy.array() was used to create an array [1], numpy.arctan() was used to get the trigonometric inverse tangent element-wise[2], numpy.degrees() was used to convert angles from radians to degree[3], matplotlib.pyplot.quiver() was used to plot a 2d field of arrows[4], numpy.add() was used to add arguments element-wise[5], numpy.linalg.norm() was used to return a matrix or vector norm [6]. For the given functions, there are np.random.randint(), plt.figure(), plt.title(), plt.xlim(), plt.ylim(), plt.xlabel(), plt.ylabel(), plt.grid(), plt.legend(), plt.show(), and plt.savefig(). np.random.randint() was used to return random integers with given parameters low(inclusive) to high(exclusive)[7], plt.figure() was used to create a new figure or activate an existing figure [8], plt.title() was used to set a title for the axes [9], plt.xlim() was used to get or set the x limits of the current axes[10], plt.ylim() was used to get or set the y-limits of the current axes[11], plt.xlabel() was used to set the label for the x-axis[12], plt.ylabel() was used to set the label for the y-axis[13], plt.grid() was used to configure the grid lines[14], plt.legend() was used to place a legend on the axes[15], plt.show() was used to display all figures[16], and plt.savefig() was used to save the current figure [17].

A vector is an object that has both a magnitude and a direction [18]. Vector quantities can be represented by a lot of things such as displacement, velocity, acceleration, force, etc. Vector quantities are just scalar quantities with direction. If we can represent vector quantities by the magnitude of something and direction as a pointer, then we can use it for real-life values such as weekly profit or reach to calculate the efficiency of your business using the given magnitude and direction.

Vectors are mostly used to know locations. An example I can give is GPS or Global Positioning System. GPS is a device that provides us with location and velocity [19] which makes it a device that gives a real-time vector quantity. Another example I can give is projectiles. When projectile motion is mentioned in physics, it usually finds the ideal condition in which you can throw an object at the maximum range [20]. We can model projectiles by finding or solving for the angle of the trajectory for the direction and knowing the velocity of the projectile for the magnitude.

References

[1]"numpy.array — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.array.html. [Accessed: 29- Sep- 2020].

[2]"numpy.arctan — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.arctan.html. [Accessed: 29- Sep- 2020].

[3]"numpy.degrees — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.degrees.html. [Accessed: 29- Sep- 2020].

[4]"matplotlib.pyplot.quiver — Matplotlib 3.3.2 documentation", Matplotlib.org, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.quiver.html. [Accessed: 29- Sep- 2020].

[5]"numpy.add — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.add.html. [Accessed: 29- Sep- 2020].

[6]"numpy.linalg.norm — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html. [Accessed: 29- Sep- 2020].

[7]"numpy.random.randint — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html. [Accessed: 29- Sep- 2020].

[8]"matplotlib.pyplot.figure — Matplotlib 3.3.2 documentation", Matplotlib.org, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.figure.html. [Accessed: 29- Sep- 2020].

[9]"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: 29- Sep- 2020].

[10]"matplotlib.pyplot.xlim — Matplotlib 3.3.2 documentation", Matplotlib.org, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.xlim.html. [Accessed: 29- Sep- 2020].

[11]"matplotlib.pyplot.ylim — Matplotlib 3.3.2 documentation", Matplotlib.org, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.ylim.html. [Accessed: 29- Sep- 2020].

[12]"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: 29- Sep- 2020].

[13]"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: 29- Sep- 2020].

[14]"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: 29- Sep- 2020].

[15]"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: 29- Sep- 2020].

[16]"matplotlib.pyplot.show — Matplotlib 3.3.2 documentation", Matplotlib.org, 2020. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.show.html. [Accessed: 29- Sep- 2020].

[17]"matplotlib.pyplot.savefig — Matplotlib 3.3.2 documentation", Matplotlib.org, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.savefig.html. [Accessed: 29- Sep- 2020].
[18]"An introduction to vectors - Math Insight", Mathinsight.org, 2020. [Online]. Available: https://mathinsight.org/vector_introduction#:~:text=A%20vector%20is%20an%20object,its%20tail%20to%20its%20head. [Accessed: 29- Sep- 2020].
[19]"What Is GPS? | Geotab", Geotab, 2020. [Online]. Available: https://www.geotab.com/blog/what-is-gps/. [Accessed: 29- Sep- 2020].
[20]C. Foundation, "Average Velocity", CK-12 Foundation, 2020. [Online]. Available: https://www.ck12.org/physics/average-velocity/rwa/Maximum-Range/. [Accessed: 29- Sep- 2020].
[21]"numpy.sum — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.sum.html. [Accessed: 30- Sep- 2020].

Appendix

Github Link - https://github.com/Sus102/LA_Lab/tree/master/LA_LAB_2