Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 3

Linear Combination and Vector Spaces

Submitted by:

Parco, Jovian Asher G.

Instructor:

Engr. Dylan Josh D. Lopez

October 25, 2020

I. Objectives

This laboratory activity aims to implement linear combinations in the 2-dimensional plane, visualize spans using vector fields, and perform vector fields operations using scientific programming.

II. Methods

In this laboratory activity, the practices consist of linear combinations w/o scalar, single-span vector plotting, and span of a linear combination of vectors. The researcher used MatPlotlib with a combination of NumPy for plotting and forming the matrices.

III. Results

The complete repository for the activity is available at Github(https://bit.ly/2HBpVNP). The lab activity consists of two parts to this lab activity; the first part is plotting different linear combinations with different scalar values. The second part is making three unique span using linear equations.

For part 1 of the lab activity, the researcher used this linear combination, "$\vec{z} = 3 * C_1$; $C_1 = \begin{bmatrix}1\\1\end{bmatrix}$" where "3" is a scalar value, and "C_1" is a matrix form, and its value is x = 1 & y = 1.

```
z = 3*np.array([1,1])

Range = np.arange(-10,10,.25)

plt.scatter(Range*z[0],Range*z[1],c="#ff1a75")

plt.xlim(-10,10)
plt.ylim(-10,10)

plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')

plt.grid()
plt.show()
```

Fig 1 Code for Part 1

For the code for part 1 (Fig 1), it is a simple linear combination that consists of different functions from Numpy & Matplotlib, where the first line is a set of an array with the values of

"[1,1]" that multiplies to a scalar value which is "3", next line uses the Numpy arrange() function where it returns evenly spaced values within a given interval [1], In the third line the research used the Numpy scatter() a scatter plot of y vs. x with varying marker size or color [2], while x/ylim() function sets the properties of the plot plane same as axh/vline() function that adds a horizontal and vertical line in the plot plane, the grid() function adds the grid line in the plane plot and last lastly show() function where it shows the plane plot with all the properties and the Numpy scatter function.

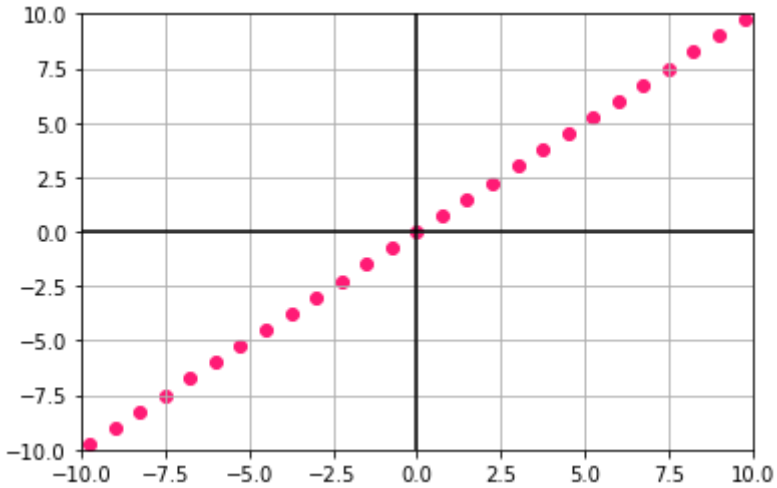

Fig 2 Output for part 1

Based on the linear combination of "$\vec{z} = 3 * C_1 ; C_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$" and input as a code (Fig 1), the researcher gets the outputted plot for part 1 (Fig 2), which shows a scattered dots along with the rise of three and the run of three where it the created a line.

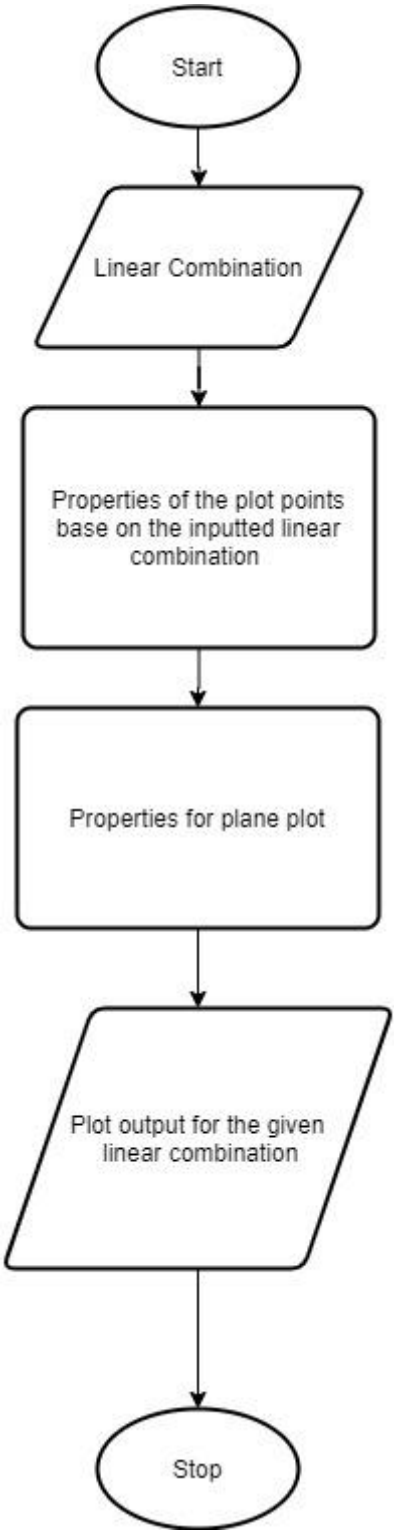

Fig 3 Flow chart for part 1

The flowchart of part 1 (Fig 1) shows all the processes based on the reachers code (Fig 1), where properties of the plane plot and the plot point itself are processed separately.

For part 2 of the lab activity, the researcher used three unique span using linear equations for the input; for the first linear equation, "$S = C_1 * V_x + C_2 * V_y$; $V_x = \begin{bmatrix}1\\0\end{bmatrix}, V_Y = \begin{bmatrix}0\\1\end{bmatrix}$" "C1/C2" are unit vectors or basis vector shows the axis relative to the plane, and the Vx/Vy are vectors in array form.

```
vectx = np.array([1,0])
vecty = np.array([0,1])

R = np.arange(-10,10,1)
c1, c2 = np.meshgrid(R,R)
spanRx = c1*vectx[0] + c2*vecty[0]
spanRy = c1*vectx[1] + c2*vecty[1]

plt.scatter(spanRx,spanRy, c="#ff1a75", s=5, alpha=0.75)

plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.grid()
plt.show()
```

Fig 4 Code for Part 2a

For the code of part 2 1st equation, the researcher groups them into four parts; the first part is the declaration of the array or inputting the equation as a code, the second part uses the Numpy meshgrid() function where it gives all the number in the arrange function in a grid-like mesh, The third part is the properties of the dot but in the code it is the span of the equation. While the fourth is the properties of the plane plot.

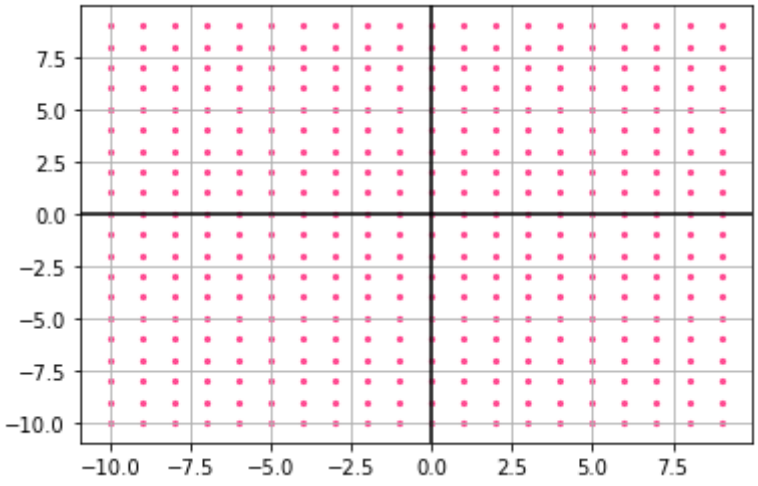

Fig 5 Output for part 2a

For the output of part 2a, the researcher confirmed that the equation "$S = C_1 * V_x + C_2 * V_y$; $V_x = \begin{bmatrix}1\\0\end{bmatrix}, V_Y = \begin{bmatrix}0\\1\end{bmatrix}$" produce a grid-like span its because all the possible horizontal line is (1,0) while all the vertical line is (0,1) and the only limit is "-10 to 10" because of the arrange() function.

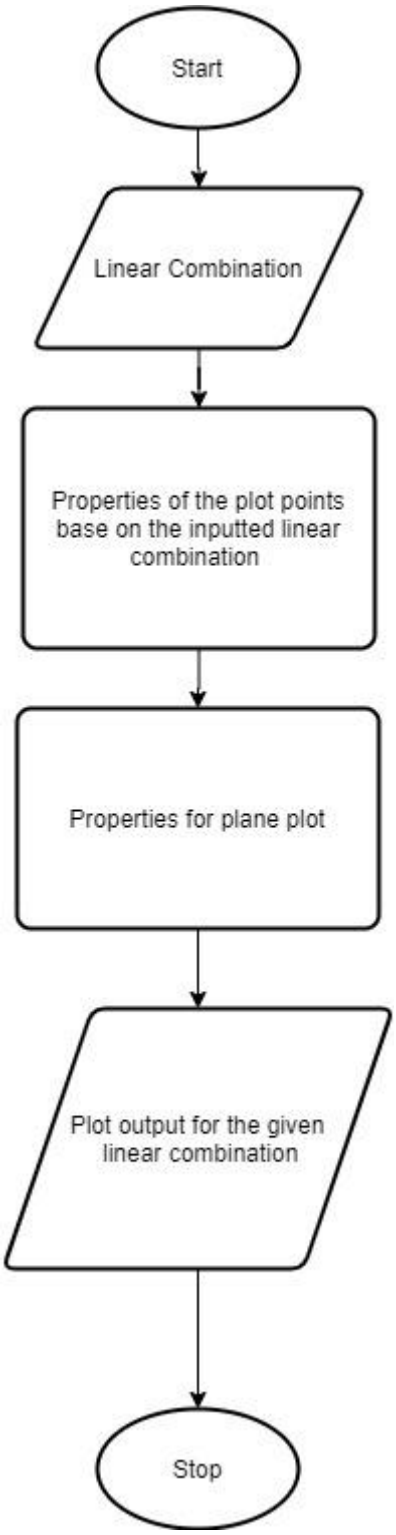

Fig 6 Flow chart for part 2

For the flow chart of part 2 (Fig 6), the chart flow is the same as the flow chart in part 1 lab activity.

Part 2b of the lab activity, it is the same as the part 2a, but the researcher changes the equation to create a unique span, which is "$S = C_1 * V_x + C_2 * V_y\ ;\ V_x = \begin{bmatrix}2\\0\end{bmatrix}, V_Y = \begin{bmatrix}3\\2\end{bmatrix}$" it is the most typical linear combination that will produce (Fig 7) a typical span.

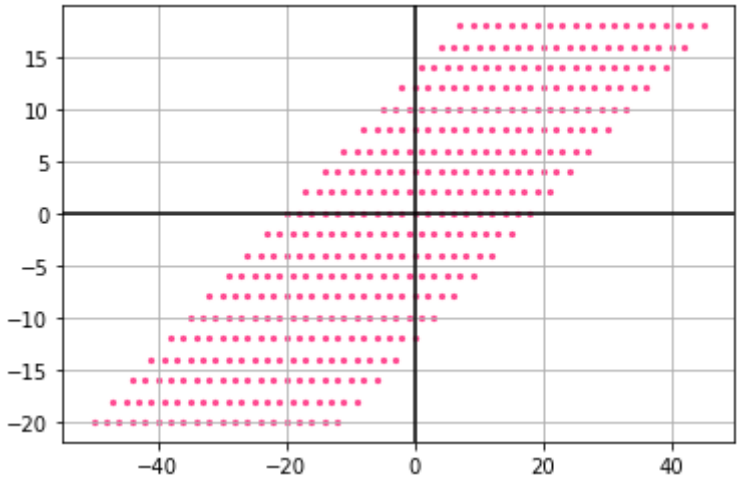

Fig 7 Output for part 2b

For part 2c of the lab activity, the researcher added linearly depend or co-linear equations where the span is overlapping, and the inputted equation is "$S = C_1 * V_x + C_2 * V_y\ ;\ V_x = \begin{bmatrix}3\\6\end{bmatrix}, V_Y = \begin{bmatrix}1\\2\end{bmatrix}$" that will produce a single line (Fig 8) like part 1.

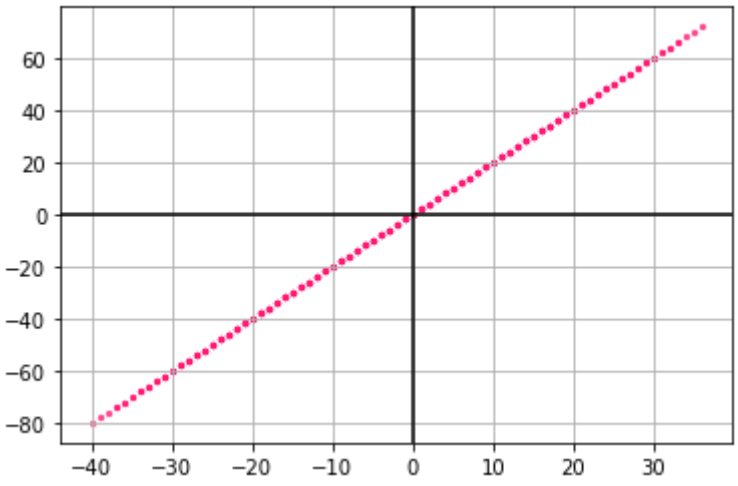

Fig 8 Output for part 2c

R It represents the number of dimensions in an array. R=1 it outputs a single line because it is on a one dimensional that only need one vector, another R=2 output two-dimensional lines. It is on a two-dimensional plane that needs 2 vectors, the same as R=3 that needs two 3 vectors in three dimensions, while R=4,5,6... it is possible. However, it must have vectors as the dimension goes on, and visualizing it for us living in three-dimensional space cannot comprehend four-dimensional space but for us possible.

Unit vectors represented as $\hat{x}$ which means a vector with one magnitude, also known as basis vector, while linear combinations are the combination of vectors and scalars to create different equations using unit vectors it represents the position of the vectors relative to the plane, example $\hat{x}$ it represents the x-axis while $\hat{y}$ represent y-axis another is $\hat{z}$ that requires a third dimension or three vectors.

IV. Conclusion

The researcher concludes that setting up a linear combination into a physical figure like a plot is straightforward. However, the only had part is reverse engineering because the researcher manually finds a specific combination to produce the plot. The researcher also finds out that every linear combination can produce a different span except for co-dependent equations that will produce a straight line. Linear combinations can be useful when calculating a trajectory that will plot all the possible points and predict the projectile path like a ball or a missile or something.

References

[1] “Numpy.arange.” https://numpy.org/doc/stable/reference/generated/numpy.arange.html (accessed Oct. 25, 2020).

[2] “Matplotlib.pyplot.scatter.” https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.scatter.html (accessed Oct. 25, 2020).