Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 3

Linear Combination and Spans

Submitted by:
Chipongian, John Patrick Ryan J.

Instructor:
Engr. Dylan Josh D. Lopez

.

November, 04, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of linear combinations using vector fields and to perform vector fields operations using scientific programming.

II. Methods

The practices that were used in completing this activity are the numpy and matplotlib libraries. In the activity, the functions that were used from the numpy library are np.array(), np.arange() and np.meshgrid(). The practices that were used from matplotlib library are plt.scatter(), plt.xlim(), plt.ylim, plt.axhline(), plt.axvline(), plt.grid() and plt.show(). All of these functions from numpy and matplotlib libraries are used to help us in visualizing the vectors and linear combinations which are the deliverables of the activity.

$$General\,Linear\,Equation\,Form$$
$$Vect_P = c(7x - 4y)$$
$$Vect_Q = c(2x + 9y)$$

$$Vector\,Form$$
$$P = \begin{bmatrix} 7 \\ -4 \end{bmatrix}, Q = \begin{bmatrix} 2 \\ 9 \end{bmatrix}$$

Figure 1 Linear equation and vectors in task 1

The first activity consists of two vectors which has a linear equation of VectP=c(7x-4y) and the second vector has VectQ=c(2x+9y)

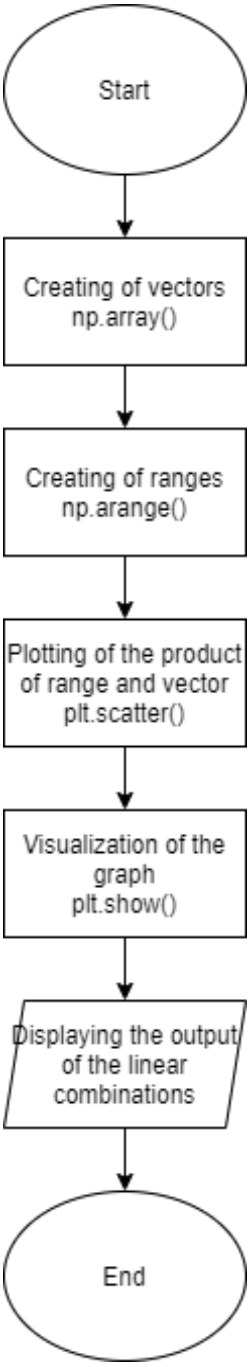

Figure 2 Flowchart of the codes in task 1

$$General\,Linear\,Equation\,Form$$
$$Vect_V = c_1(8x - 3y)$$
$$Vect_T = c_2(6x + 9y)$$

$$Vector\,Form$$
$$S = \left\{ c_1 \cdot \begin{bmatrix} 8 \\ -3 \end{bmatrix}, c_2 \cdot \begin{bmatrix} 6 \\ 9 \end{bmatrix} \right\}$$

Figure 3 First span of different linear combination in task 2

$$General\,Linear\,Equation\,Form$$
$$Vect_O = c_1(3x + 2y)$$
$$Vect_I = c_2(8x + y)$$

$$Vector\,Form$$
$$S = \left\{ c_1 \cdot \begin{bmatrix} 3 \\ 2 \end{bmatrix}, c_2 \cdot \begin{bmatrix} 8 \\ 1 \end{bmatrix} \right\}$$

Figure 4 Second span of different linear combination in task 2

$$General\,Linear\,Equation\,Form$$
$$Vect_X = c_1(-5x + 7y)$$
$$Vect_Z = c_2(-7x + 5y)$$

$$Vector\,Form$$
$$S = \left\{ c_1 \cdot \begin{bmatrix} -5 \\ 7 \end{bmatrix}, c_2 \cdot \begin{bmatrix} -7 \\ 5 \end{bmatrix} \right\}$$

Figure 5 Third span of different linear combination in task 2

In the second part of the activity, three unique spans were created using three different linear combinations. The first Linear combinations are $Vect_P = c_1(8x-3y)$ and $Vect_Q = c_2(6x+97)$, and the second linear combinations are $Vect_O = c_1(3x+2y)$ and $Vect_I = c_2(8x+y)$, and the third linear combinations are $Vect_X = c_1(-5x+7y)$ and $Vect_Z = c_2(-7x+5y)$.

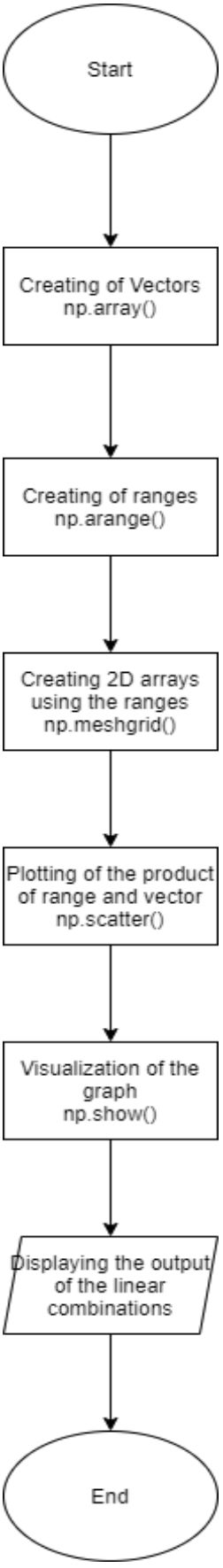

Figure 6 Flowchart of the program in Task 2

III. Results

```
vectP =  np.array([7,-4])
vectQ =  np.array([2,9])

c1 = np.arange(-11,11,0.25)
c2 = np.arange(-2,8,0.25)

plt.scatter(c1*vectP[0],c1*vectP[1], color = 'b')
plt.scatter(c2*vectQ[0],c2*vectQ[1], color = 'g')

## Visualization of the plots
plt.xlim(-10,10)
plt.ylim(-10,10)
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.grid()
plt.show()
```

Figure 7 Codes from task 1

Figure 7 are the codes that are used in implementing the task given in the activity. To implement it, the numpy and matplotlib libraries were used. The functions that were used are np.array() which creates an array, [1] np.arange() which creates an array with evenly spaced elements base on the interval, [2] plt.scatter() which is used to draw the plots to the grid, [3] plt.xlim() and [4] plt.ylim sets the x-limits and y-limits of the axes with similar function as [5] plt.axhline() and [6] plt.axvline() which adds horizontal and vertical lines across the axes. The last two functions are [7] plt.grid(), which makes the grid visible and is able to configure it, and [8] plt.show(), which is used to display all the figures.

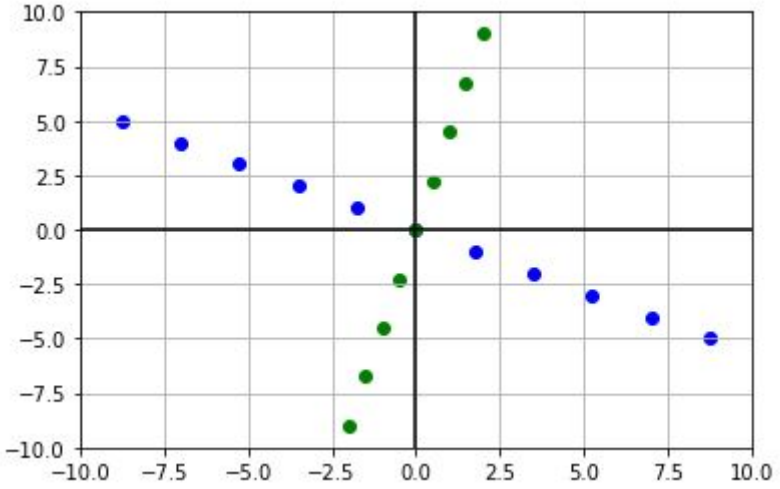

Figure 8 Output of the code in task 1

The codes from figure 7 would show the output that is shown in figure 8.

```
vectV = np.array([8,-3])
vectT = np.array([6,9])
R = np.arange(-15,15,2)
c1, c2 = np.meshgrid(R,R)
spanRx = c1*vectV[0] + c2*vectT[0]
spanRy = c1*vectV[1] + c2*vectT[1]
plt.scatter(spanRx,spanRy, s=5, alpha=0.75)

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

Figure 9 Codes of the 1st span of linear combination in task 2

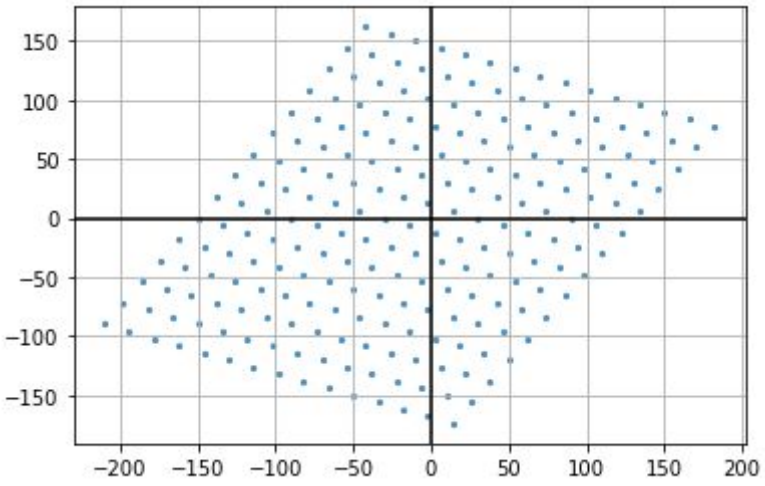

Figure 10

```
vectO = np.array([3,2])
vectI = np.array([8,1])
R = np.arange(-15,15,2)
c1, c2 = np.meshgrid(R,R)
spanRx = c1*vectO[0] + c2*vectI[0]
spanRy = c1*vectO[1] + c2*vectI[1]
plt.scatter(spanRx,spanRy, s=5, alpha=0.75)

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

Figure 11

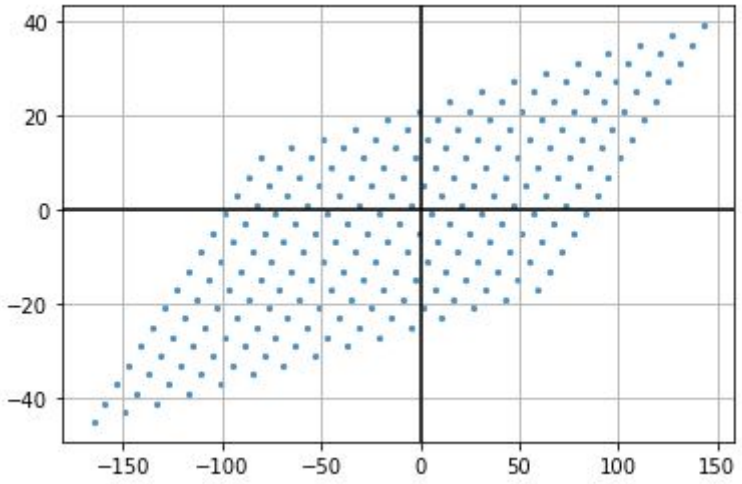

Figure 12

```
vectX = np.array([-5,7])
vectZ = np.array([-7,5])
R = np.arange(-15,15,2)
c1, c2 = np.meshgrid(R,R)
spanRx = c1*vectX[0] + c2*vectZ[0]
spanRy = c1*vectX[1] + c2*vectZ[1]
plt.scatter(spanRx,spanRy, s=5, alpha=0.75)

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

Figure 13

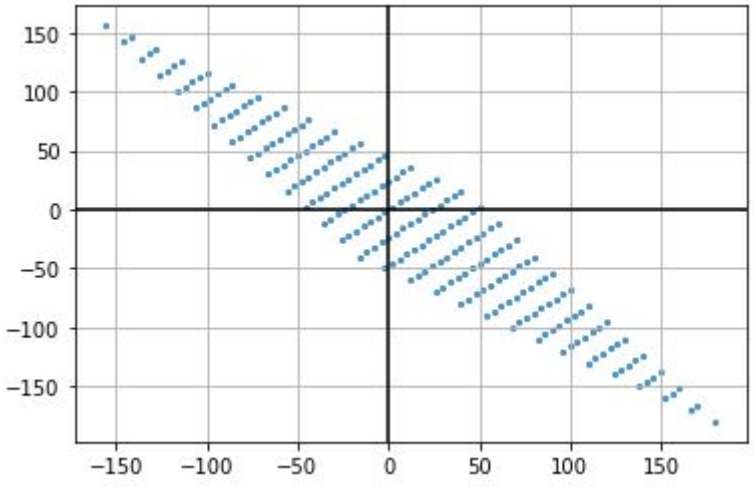

Figure 14

In the second task of the activity, all the functions that were used in the first task were also used and certain functions were added to accomplish the second task. All of these can be seen in the figures 9, 11, 13 with its own output in figures 10, 12, 14. The added function in this task is the function [9] np.meshgrid() which turns a one-dimensional array into matrices.

Having a ℝ=1 as represented as a line and ℝ=2 be represented as a plane, ℝ=3 would result to a three-dimensional plot and ℝ=4 a four-dimensional plot. This is because the rank is what tells what is the dimension of the plot.

The role of the unit vectors in relation with the linear combination is that it dictates the direction of it. It can be observed in the task given above that the linear direction is towards the unit vector given.

IV. Conclusion

Doing this laboratory activity proved that a one-dimensional line can be turned into a two-dimensional plane. This was proven using the numpy and matplotlib libraries which means that it can visualize using these libraries. By using these functions, it would be possible to change the rank of dimensions of an object.

Using linear combinations would help us in our job as a computer engineer mostly it would help us in making blueprints. This would help us visualize the figures and on how to make it all fit. Another application is by using it in visualizing economic graphs. By using this, it would make visualization and conceptualization of what the graphs mean.

References

[1] GeeksforGeeks. 2020. *Numpy.Arange() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/numpy-arange-python/> [Accessed 4 November 2020].

[2] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Scatter() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-scatter-in-python/> [Accessed 4 November 2020].

[3] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Xlim() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-xlim-in-python/#:~:text=The%20xlim()%20function%20in,limits%20of%20the%20current%20axes.&text=Parameters%3A%20This%20method%20accept%20the,set%20the%20xlim%20to%20right.> [Accessed 4 November 2020].

[4] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Ylim() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-ylim-in-python/> [Accessed 4 November 2020].

[5] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Axhline() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-axhline-in-python/> [Accessed 4 November 2020].

[6] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Axvline() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-axvline-in-python/> [Accessed 4 November 2020].

[7] Tutorialspoint.com. 2020. *Matplotlib - Grids - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/matplotlib/matplotlib_grids.htm> [Accessed 4 November 2020].

[8] GeeksforGeeks. 2020. *Matplotlib.Pyplot.Show() In Python - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/matplotlib-pyplot-show-in-python/> [Accessed 4 November 2020].

[9] Numpy.org. 2020. *Numpy.Meshgrid — Numpy V1.19 Manual*. [online] Available at: <https://numpy.org/doc/stable/reference/generated/numpy.meshgrid.html> [Accessed 4 November 2020].