Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 3

Linear Combination and Vector Spaces

Submitted by:
Reyes, Carl Vincent G.

Instructor:
Engr. Dylan Josh D. Lopez

October 27, 2020

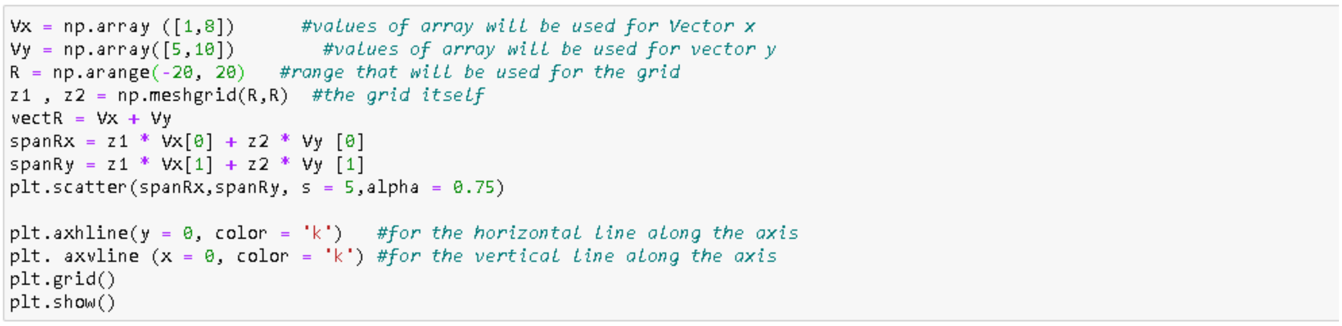
I. Objectives

This laboratory activity aims to implement the principles and techniques of python using different combinations of linear and vector spaces and making use of our fundamental knowledge about linear combination and using it for scientific programming

II. Methods

- What are the practices of the activity?
  - Applying our fundamental knowledge in linear combination from the past lessons and applying it for scientific programming.
- What do they imply and teach?
  - These laboratory teaches the use of 2- dimensional planes and visualizing spans with vector fields in python, thus, being able to do it and use it in different occasion.
- What are the deliverables of the activity?
  - The deliverables are the using vector fields, span, 2-dimensional planes, and the linear combination we learned during the past lessons.
- How did you achieve them?
  - Achieving or doing this laboratory must include, knowledge and a well understanding of the functions, the codes, the libraries used in the past lesson, such as the matplotlib which we have used in ploitting vectors, also plays a big part in this laboratory, another one is the numpy library for the np.arrays(), np.arange(), and np.meshgrid(), without a knowledge on this kinds of function, this lkaboratory will not be doable.

III. Results



```
Vx = np.array ([1,8])       #values of array will be used for Vector x
Vy = np.array([5,10])         #values of array will be used for vector y
R = np.arange(-20, 20)    #range that will be used for the grid
z1 , z2 = np.meshgrid(R,R)  #the grid itself
vectR = Vx + Vy
spanRx = z1 * Vx[0] + z2 * Vy [0]
spanRy = z1 * Vx[1] + z2 * Vy [1]
plt.scatter(spanRx,spanRy, s = 5,alpha = 0.75)

plt.axhline(y = 0, color = 'k')   #for the horizontal line along the axis
plt. axvline (x = 0, color = 'k') #for the vertical line along the axis
plt.grid()
plt.show()
```

Figure 1: Codes used for task 1

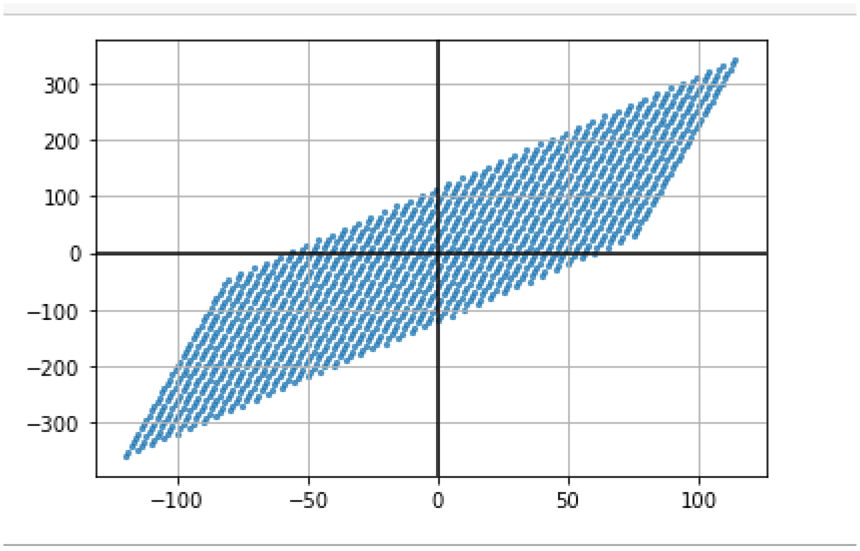


Figure 2: result for task 1

This are the codes used for the first task and as well as its result. The codes used are similar to the example made by the professor but tweaked in order to gain more knowledge through experimenting and understanding the scope of the lesson and opening more doors for learning. Looking at the result, there are more ways to tweak it such as the first thing I tried was adding more numbers and looking at the different results such as for R = np.arange(-20,20), adding a third number on this code actually helps make the points scatter or give more spaces on each other, np.arange() are values generated within the half-interval, in other terms [1], the values placed inside the parenthesis is the start and stop values, thus:

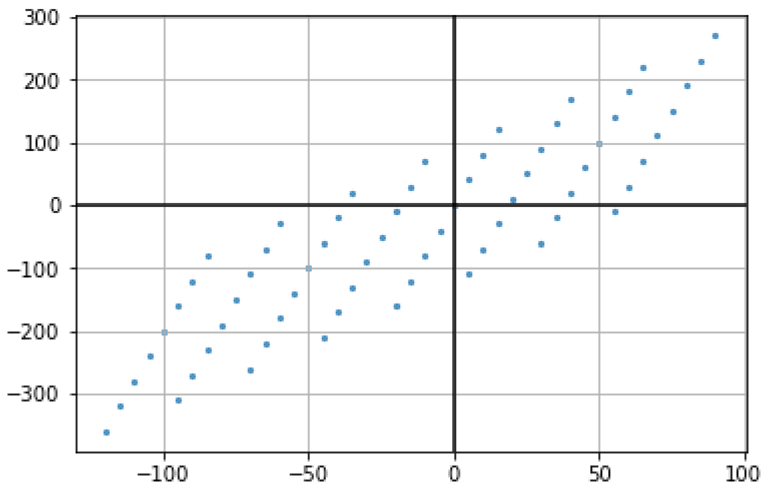
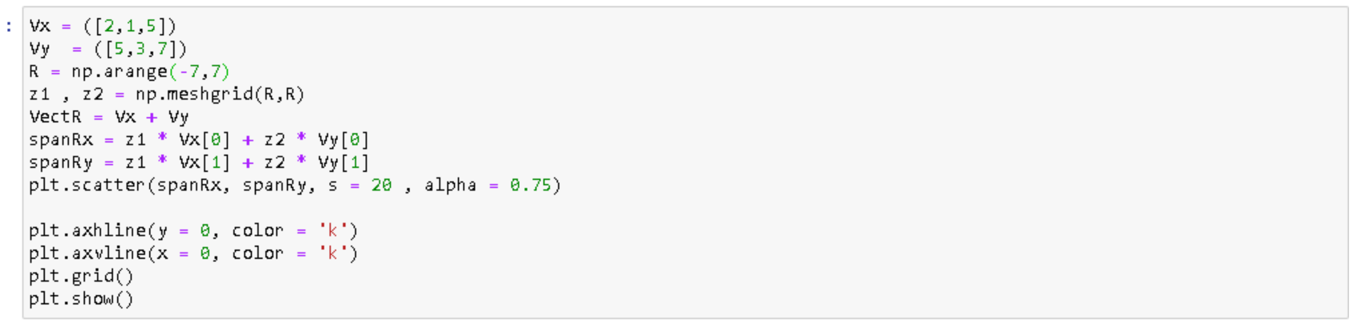


Figure 3: another result for task 1

Such as this one, the figure is still there and the grid itself did not change, but adding one number into the range can give spaces, it is less compact and more of a dots floating around. Plt.scatter() also helps in changing the size of the figure once edited, in this line of code, there many things that can be done and achieved, different figures, different sizes, just by adding different numbers.



```
: Vx = ([2,1,5])
  Vy  = ([5,3,7])
  R = np.arange(-7,7)
  z1 , z2 = np.meshgrid(R,R)
  VectR = Vx + Vy
  spanRx = z1 * Vx[0] + z2 * Vy[0]
  spanRy = z1 * Vx[1] + z2 * Vy[1]
  plt.scatter(spanRx, spanRy, s = 20 , alpha = 0.75)

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

Figure 4: codes used for task 2

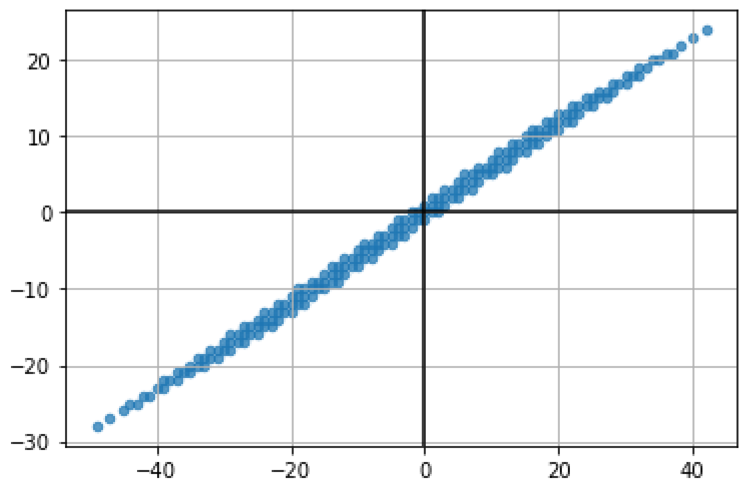


Figure 5: Result for task 2

For the second task, most of the codes are actually the same, but the only difference is that there are more variables in the vectors x and y, so instead of only using two numbers, there are now three, then the size and alpha were also changed, then all are the same. The figure now changed and is close to being somewhat like a line, but these image somehow remind me of seismographic images found during earthquakes, it resembles the main location of where the earthquake struck and the circles are the intensities of the quake.

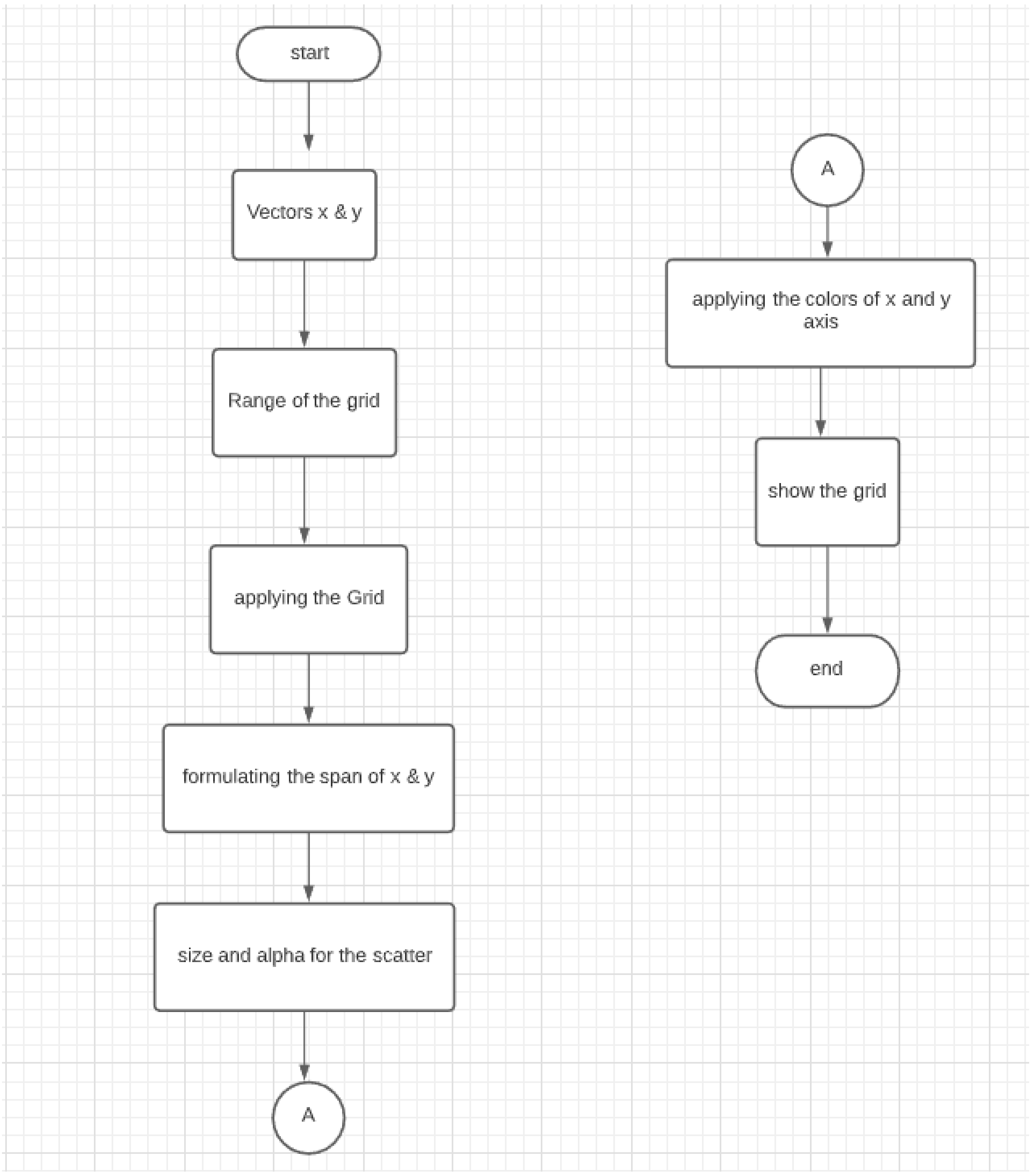

start
Vectors x & y
Range of the grid
applying the Grid
formulating the span of x & y
size and alpha for the scatter
A
A
applying the colors of x and y axis
show the grid
end

Figure 6: Flow chart of the tasks

The figure above is the flowchart for the both task since there is not much of a difference between their codes. The first process actually defines the first codes which is setting the variables for x and y, then the next process is the setting the range of the grid, then in applying the grid or the meshgrid itself for the plotting, then the next is formulating the span which in the code, it is needed to multiply z1 and z2 to the vectors x and y, then the size and alpha of the scatter for the visibility and how compact the points are going to be. In connection, the next one would be the colors of the x and y axis, which can also vary depending on the color you will place, then last is the show. where plt.show() will allow the graph to be processed and be seen after running the codes.

You might have notices that the dimensions plot of linear combinations change according to its rank. If a vector is ℝ=1R=1 the plot of its linear combination is one-dimensional or a line, and if ℝ=2R=2 the plot is a plane. What will be the shape of the vector visualization if ℝ=3R=3 and if ℝ=4R=4?Based on what I have learned from this activity Python can work with the same logic as the C++ language does. But where they differ

IV. Conclusion

Overall, Laboratory 3: Linear Combinations and Vector Spaces showed us that there is a lot of possibilities that can happen if we use the right codes and units in vectors, we can create a 1-dimensional line, 2-dimensional planes, and many more. This laboratory also introduced us to different uses of the NumPy and the MatPlotLib which can visualize more than we had learned for the past weeks, and with these we are now able to realize new opportunities and scenarios in which we can apply this knowledge in real-time situations.

These real-time situations could be collisions, because from the graph and the lesson, those vectors are all lines pointing towards the movement of an object, meaning if there are two airplanes and are both going in a direction and there Is a possibility of them crashing together, and as an engineer or data analyst, you can use this knowledge to divert their courses and avoid them from crashing. Another example is game development where an AI would shoot beam of rays and see if there is an enemy within range, and if there is none, nothing changes. [2]

V. References

[1] "numpy.arange," The SciPy community, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.arange.html#:~:text=Return%20evenly%20spaced%20values%20within,including%20start%20but%20excluding%20stop)..

[2] "Parametric representations of lines," [Online]. Available: https://www.khanacademy.org/math/linear-algebra/vectors-and-spaces/vectors/v/linear-algebra-parametric-representations-of-lines?modal=1.. [Accessed 27 10 2020].

GITHUB REPO:

https://github.com/ReyesCarl/Linalg_Lab3