Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 3

Linear Combination and Spans

Submitted by:

Surio, Rovil Jr., M.

Instructor:

Engr. Dylan Josh D. Lopez

October, 23, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of representing linear combinations in the 2-dimensional plane. This also aims to visualize spans using vector fields in Python and to perform vector fields operations using scientific programming.

II. Methods

- The practices of the activity include combining linear scaling and the addition of a vector.
    - The activity teaches and implies the techniques of representing linear combinations in the 2-dimensional plane and visualizing spans using vector fields in python. Some functions were used such as np.arange() function that returns evenly spaced values within a given interval [1], plt.scatter() function that makes the plotting of the x and y components of a vector into a plane faster [2], plt.axhline() and plt.axvline() functions that adds a horizontal and vertical line across the axes [3][4], np.meshgrid() function that returns coordinate matrices from coordinate vectors [5].
- The deliverables of the activity are to provide a linear equation and vector form of the linear combination.
    - It was achieved by providing random values to be used in the equations and the vector form of the linear combination. The linear equations used in activity 1 are $v_1 = c2x + c5y$, $v_2 = c_1 5x + c_1 4y$, and $v_3 = c_2 10x + c_2 14y$. The vector form of the linear combination used is $V_1 = \mathrm{c} \cdot \begin{bmatrix} 2 \\ 5 \end{bmatrix}$, $V_2 = \mathrm{c}_1 \cdot \begin{bmatrix} 5 \\ 4 \end{bmatrix}$, $V_3 = \mathrm{c}_2 \cdot \begin{bmatrix} 10 \\ 14 \end{bmatrix}$. For activity 2, the linear equations for 3 unique spans are $c_1 X = c_1 11x + c_1 5y \; and \; Y = c_2 2x + c_2 12y$ for first unique span and its vector form of span is $S = \left\{ c_1 \cdot \begin{bmatrix} 11 \\ 5 \end{bmatrix}, c_2 \cdot \begin{bmatrix} 2 \\ 12 \end{bmatrix} \right\}$. For the second unique span, the linear equations are $c_1 X = c_1 4x + c_1 9y \; and \; c_2 Y = c_2 7x + c_2 1y$ and its vector form of span is $S = \left\{ c_1 \cdot \begin{bmatrix} 4 \\ 9 \end{bmatrix}, c_2 \cdot \begin{bmatrix} 7 \\ 1 \end{bmatrix} \right\}$. Lastly, the linear equations are $c_1 X = c_1 8x + c_1 8y \; and \; c_2 Y = c_2 2x + c_2 2y$ and its vector form of span is $S = \left\{ c_1 \cdot \begin{bmatrix} 8 \\ 8 \end{bmatrix}, c_2 \cdot \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\}$.

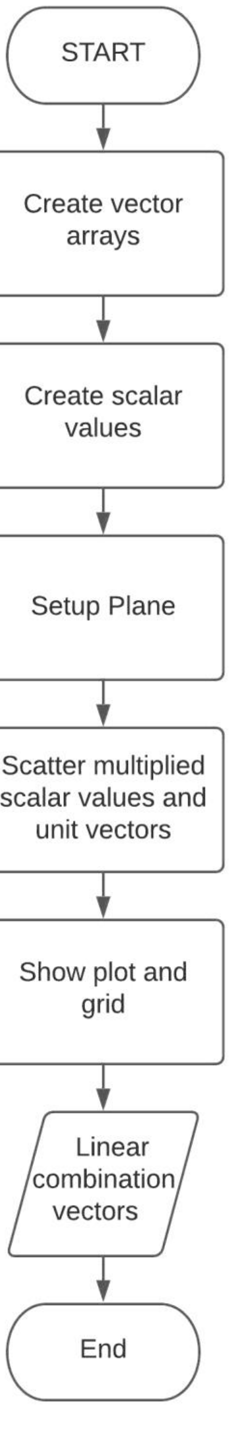


Figure 1 Activity 1 flowchart

Figure 1 shows the flowchart of the code created for activity 1.

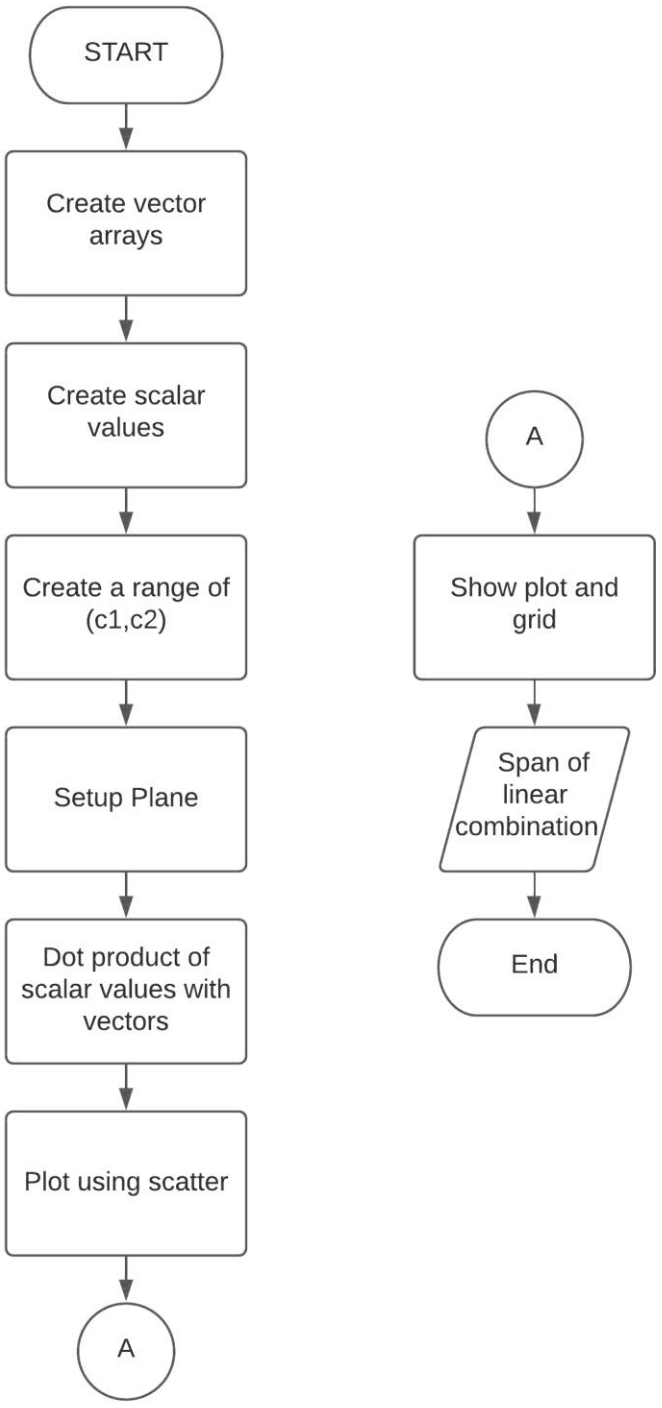
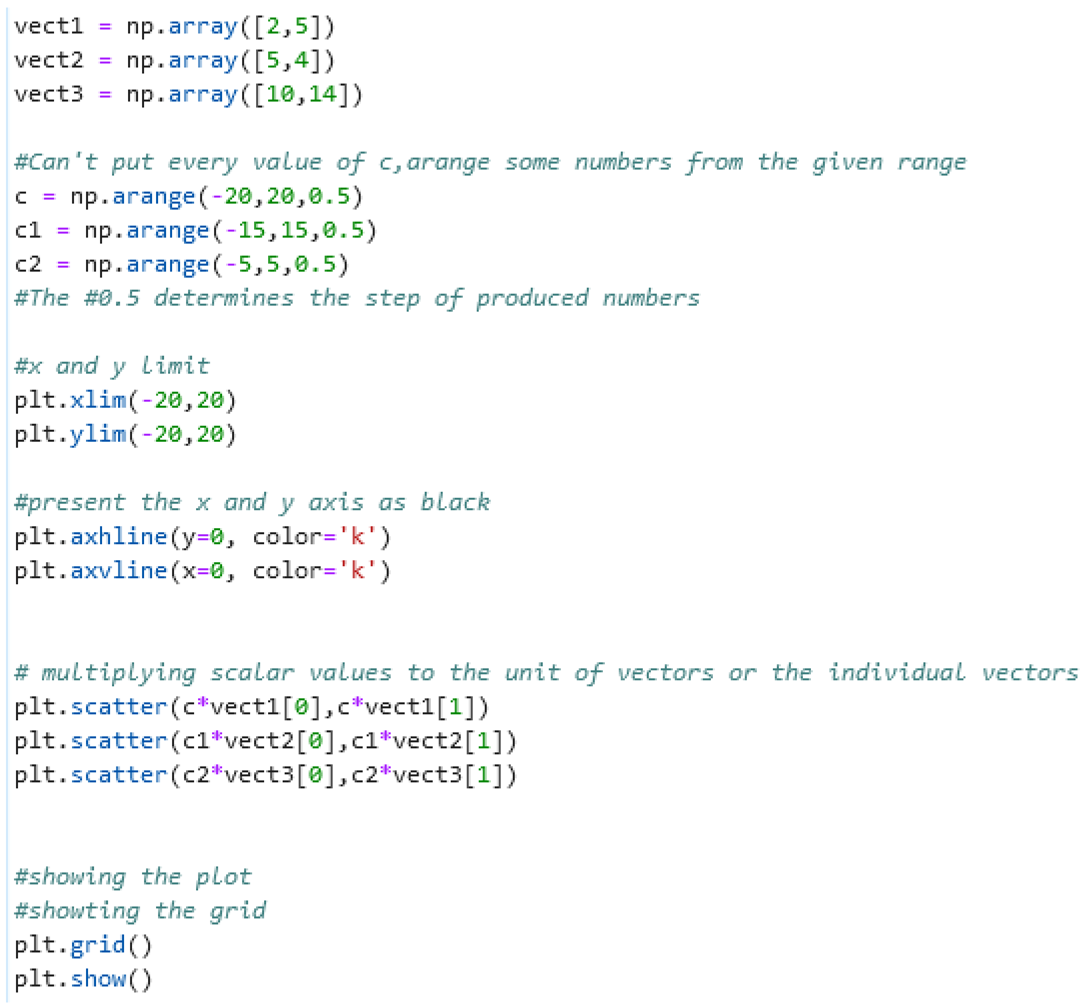


Figure 2 Activity 2 flowchart

Figure 2 shows the flowchart of the code created for activity 2.

III. Results

The result of the two activities done by implementing the principles and techniques of representing combinations in the 2-dimensional plane and visualizing spans using vector fields by performing operations using scientific programming was presented in this section.



```
vect1 = np.array([2,5])
vect2 = np.array([5,4])
vect3 = np.array([10,14])

#Can't put every value of c,arange some numbers from the given range
c = np.arange(-20,20,0.5)
c1 = np.arange(-15,15,0.5)
c2 = np.arange(-5,5,0.5)
#The #0.5 determines the step of produced numbers

#x and y limit
plt.xlim(-20,20)
plt.ylim(-20,20)

#present the x and y axis as black
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')


# multiplying scalar values to the unit of vectors or the individual vectors
plt.scatter(c*vect1[0],c*vect1[1])
plt.scatter(c1*vect2[0],c1*vect2[1])
plt.scatter(c2*vect3[0],c2*vect3[1])


#showing the plot
#showting the grid
plt.grid()
plt.show()
```

Figure 3 Activity 1 code

Figure 3 shows the code of block created to show the linear combination visualization. It was achieved by first creating 3 different arrays of vectors and arranging 3 different scalar values by using the np.array() function and np.arrange() function. The step used in 3 different scalars is 0.5. The linear combination in the first vector was created using the equation $cv_1 = \text{c} * v_1[0], \text{c} * v_1[1]$. The second linear combination was created by using the equation $c_1v_2 = c_1 * v_2[0], c_1 * v_2[1]$ and the third linear combination was also created by using the equation $c_2v_3 = c_2 * v_3[0], c_2 * v_3[1]$. These linear combinations in 1 vector were plotted into a 2-dimensional plane using the plt.scatter() function.

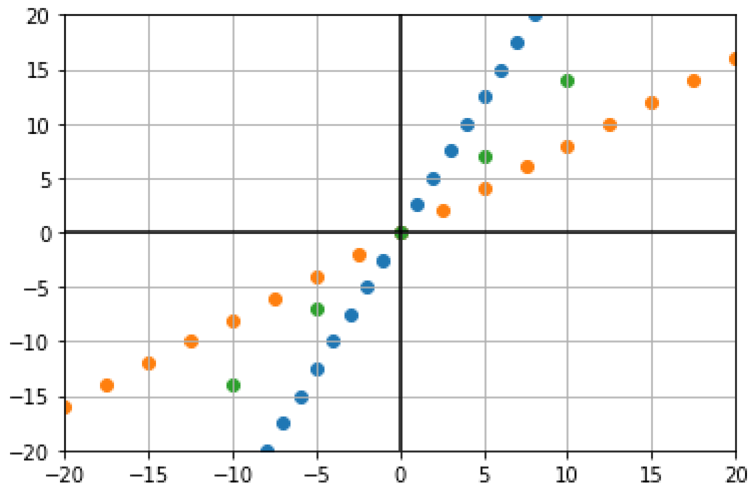


Figure 4 Output of activity 1

Figure 4 shows the output of 3 linear combinations in 1 vector. Each combination intersects in the origin and makes a line and it can be observed that they are not linearly dependent on each other.

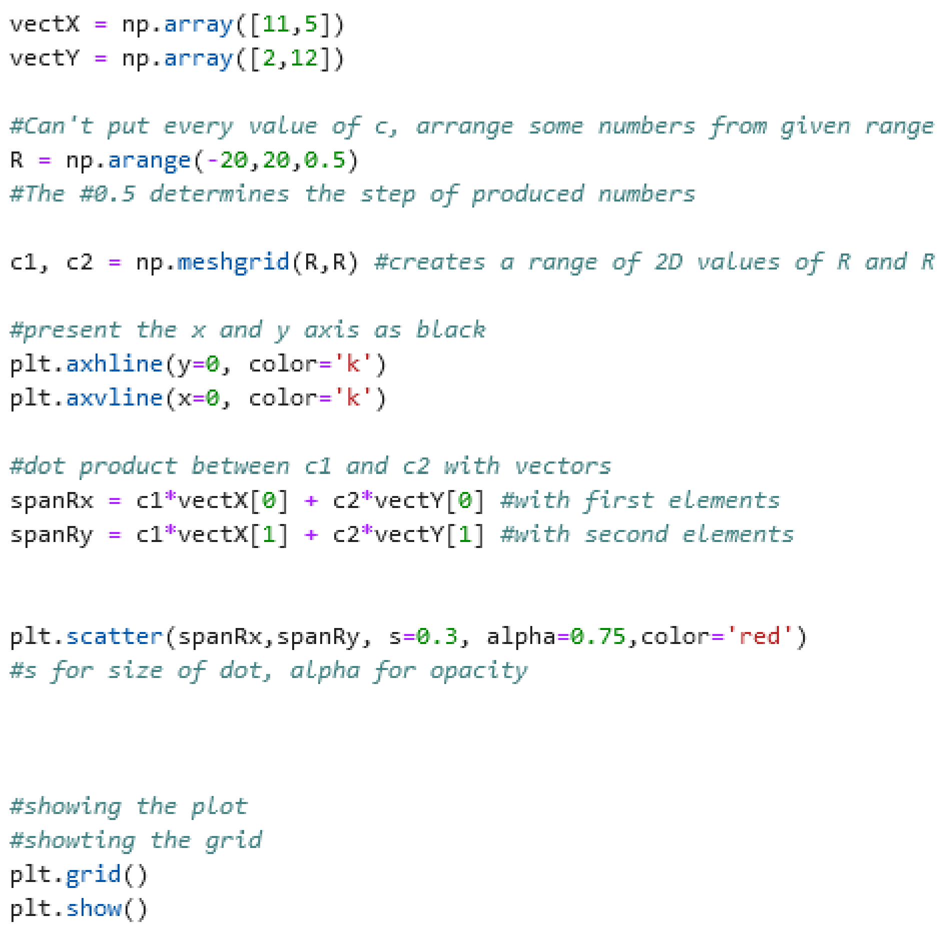
$$c_1X = c_1 11x + c_1 5y$$
$$c_2Y = c_2 2x + c_2 12y$$

$$S = \left\{ c_1 \cdot \begin{bmatrix} 11 \\ 5 \end{bmatrix}, c_2 \cdot \begin{bmatrix} 2 \\ 12 \end{bmatrix} \right\}$$

$$Equations$$
$$Rx = c1 * vectX[0] + c2 * vectY[0]$$
$$Ry = c1 * vectX[1] + c2 * vectY[1]$$

Figure 5 Vector form and equations

Figure 5 shows the 2 vectors and the equations to be used in getting the span visualization of the first linear combination of two vectors.



```
vectX = np.array([11,5])
vectY = np.array([2,12])

#Can't put every value of c, arrange some numbers from given range
R = np.arange(-20,20,0.5)
#The #0.5 determines the step of produced numbers

c1, c2 = np.meshgrid(R,R) #creates a range of 2D values of R and R

#present the x and y axis as black
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')

#dot product between c1 and c2 with vectors
spanRx = c1*vectX[0] + c2*vectY[0] #with first elements
spanRy = c1*vectX[1] + c2*vectY[1] #with second elements


plt.scatter(spanRx,spanRy, s=0.3, alpha=0.75,color='red')
#s for size of dot, alpha for opacity



#showing the plot
#showting the grid
plt.grid()
plt.show()
```

Figure 6 Activity 2 code 1

Figure 6 shows the code of block created to present the span of the first linear combination of two vectors. Two arrays of the vectors were created and the creation of scalar values was done using the np.arrange() function and this time with the np.meshgrid() function to create a range of 2D values of R and R. A dot product was performed between c1 and c2 with the array of vectors using the equations $Rx = c1 * vectX[0] + c2 * vectY[0]\ and\ Ry = c1 * vectX[1] + c2 * vectY[1]$ to get the spanRx and spanRy. This was plotted into the plane using the plt.scatter().

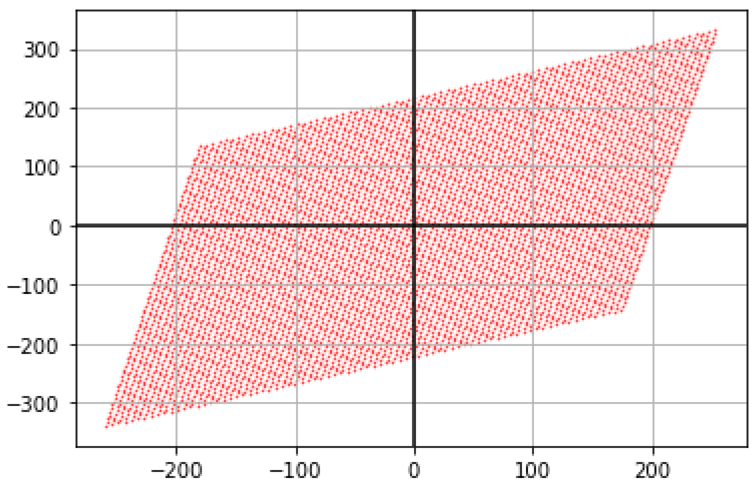


Figure 7 Activity 2 output 1

Figure 7 shows the output of the code executed to present the span of the first linear combination of two vectors. All the specific vectors inside the plane that was created using the equation are shown. The vectors are not a scale of each other therefore it is not linearly dependent and it has a rank of 2.

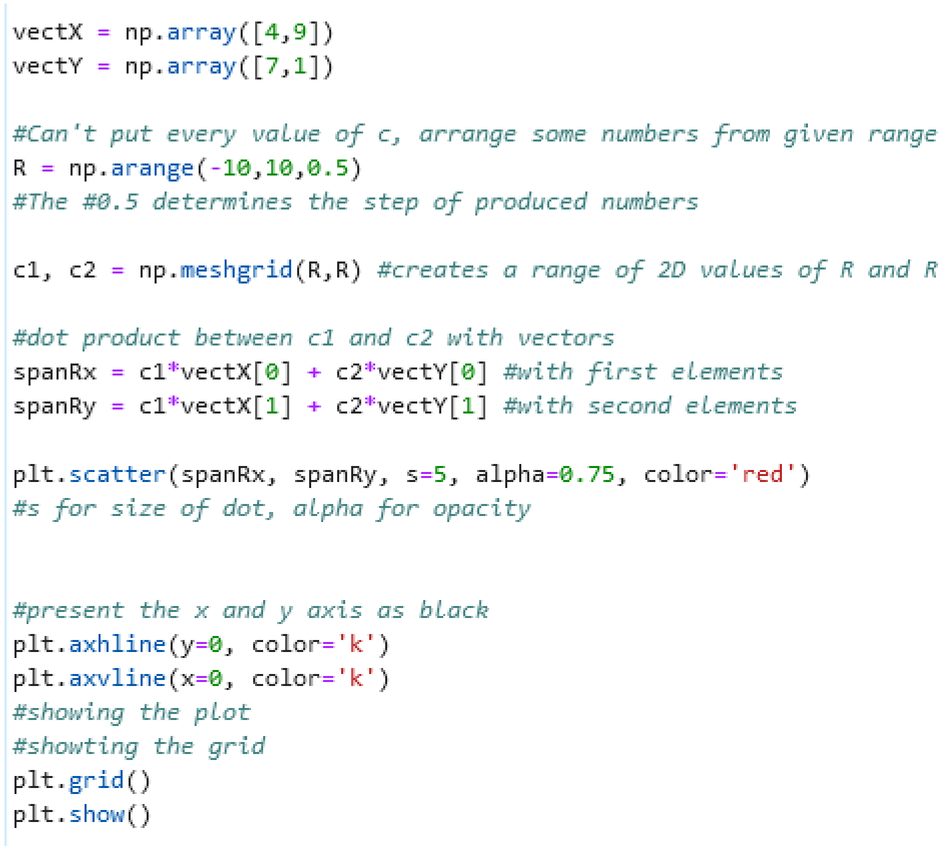
$$c_1X = c_1 4x + c_1 9y$$
$$c_2Y = c_2 7x + c_2 1y$$

$$S = \left\{ c_1 \cdot \begin{bmatrix} 4 \\ 9 \end{bmatrix}, c_2 \cdot \begin{bmatrix} 7 \\ 1 \end{bmatrix} \right\}$$

$$\textit{Equations}$$
$$Rx = c1 * vectX[0] + c2 * vectY[0]$$
$$Ry = c1 * vectX[1] + c2 * vectY[1]$$

Figure 8 Vector form and equation

Figure 8 shows the 2 vectors created and the equations to be used in getting the span visualization of the second linear combination of two vectors.



```
vectX = np.array([4,9])
vectY = np.array([7,1])

#Can't put every value of c, arrange some numbers from given range
R = np.arange(-10,10,0.5)
#The #0.5 determines the step of produced numbers

c1, c2 = np.meshgrid(R,R) #creates a range of 2D values of R and R

#dot product between c1 and c2 with vectors
spanRx = c1*vectX[0] + c2*vectY[0] #with first elements
spanRy = c1*vectX[1] + c2*vectY[1] #with second elements

plt.scatter(spanRx, spanRy, s=5, alpha=0.75, color='red')
#s for size of dot, alpha for opacity


#present the x and y axis as black
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
#showing the plot
#showting the grid
plt.grid()
plt.show()
```

Figure 9 Activity 2 code 2

Figure 9 shows the code of block created to show span visualization of the second linear combination of two vectors that will be produced from the array of vectors and scale values. The code and functions used and how the plane and plot showed were the same as the first code presented in this activity. The only changes made are the values of the array of vectors.

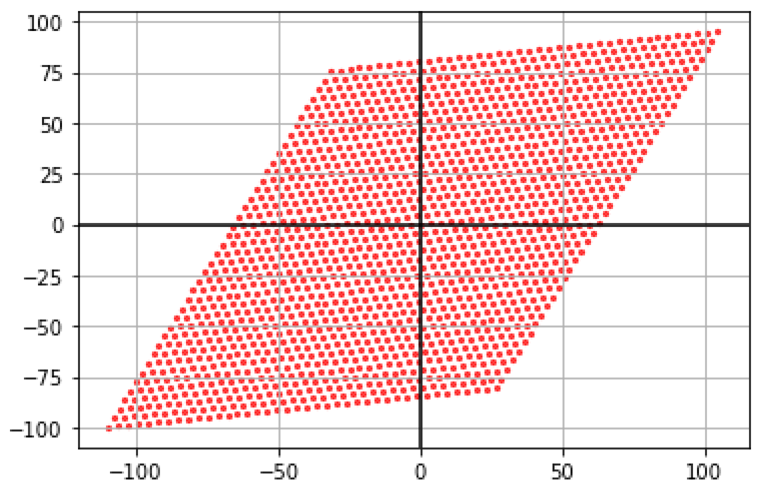


Figure 10 Activity 2 output 2

Figure 10 shows the output or the span of the second linear combination of two vectors. It can be observed that all the set of vectors were plotted and it was linearly independent since the vectors are not a scale of each other nor they aligned. It also has a rank of 2.

$$c_1 X = c_1 8x + c_1 8y$$
$$c_2 Y = c_2 2x + c_2 2y$$

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

$$\textit{Equations}$$
$$Rx = c1 * vectX[0] + c2 * vectY[0]$$
$$Ry = c1 * vectX[1] + c2 * vectY[1]$$

Figure 11 Vector form and equation

Figure 11 shows the vectors used and the equation for getting the span visualization of the third linear combination of two vectors.

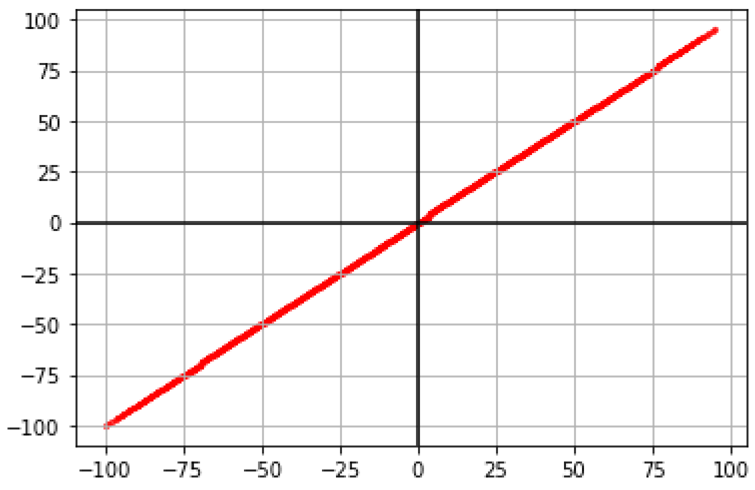


Figure 12 Activity 2 output 3

Figure 12 shows the output of the plotted values from the third linear combination of two vectors. It can be observed that the span is a colinear vector or aligned hence it is a linearly dependent vector and has a rank of 1.

It was noticeable that the dimensions plot of linear combination change according to its rank. For example, in Figure 12, it has a rank of 1 so it is one-dimensional and in Figure 10, it has a rank of 2 and a plot is a plane. The shape of the vector visualization if $\mathbb{R} = 3$ and if $\mathbb{R} = 4$ would be (3,3,3) and (4,4,4) because it will become three-dimensional and four-dimensional vectors if the shape was found out using the function from NumPy. Also, the number of dimensions can be extended into an arbitrary number [6]. In addition, the shape of vector visualization that has a $\mathbb{R} = 3$ and plottled using 3D plot, will be a horizontal plane or vertical plane [7].

Since the linear combination is about scaling two or one vectors and adding it, when no unit vectors are given then the scalar values don't have anything to scale. So, the role of unit vectors is to have a value if the purpose is to reach a specific span since when the scalar was multiplied to this unit vectors, it transforms that unit vector or stretches it out into a plane [6]. Also, the scalar values multiplied to the unit vectors tell where the given vectors will land in graphical representation or plane.

IV. Conclusion

The implementation of the principles and techniques of representing linear combinations in the 2-dimensional plane are learned through the discussion in this laboratory and the given activities by the instructor. Using the vector fields in Python and by performing vector field operations using scientific programmings such as doing a dot product between the scalar values and vectors, the spans were visualized. I have learned new functions in this laboratory such as the plt.scatter() function that makes the plotting of the x and y components of a vector into a plane faster [2]. The np.arange() functions that create numbers from the given interval [1]. I also learned that the span can be linearly dependent vector when it is a scale of one another otherwise it is an independent vector. Moreover, for every linear combination, there would be a distinct set of values or plotting unless it is a linearly dependent vector. If a span is linearly dependent, it has a rank of 1 and if the span is presented in a 2-dimensional plane then it has a rank of 2. To conclude what I have learned, the span of the two vectors represents all the possible vectors you can reach using only the two vector field operations which is vector addition and scalar multiplication. Also, the scalar values scale the unit vectors or tell where the vector will land in the plane.

The concept of linear combination can be applied in engineering especially computer engineering or in some programming-related courses in a way that some programs need to do the concept of linear combination because of the specific purpose of the program. For example, a program that represents data through vectors that plotted in a plane. This program can be about showing all the specific position of something like how is the movement of an eagle at a specific time. We can plot it using a linear combination and in this way, we can see the span of all possible location movement of the eagle during the specific time set.

References

[1]"numpy.arange — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.arange.html. [Accessed: 22- Oct- 2020].

[2]"matplotlib.pyplot.scatter — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.scatter.html. [Accessed: 22- Oct- 2020].

[3]"matplotlib.pyplot.axvline — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.axvline.html. [Accessed: 22- Oct- 2020].

[4]"matplotlib.pyplot.axhline — Matplotlib 3.3.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.axhline.html. [Accessed: 22- Oct- 2020].

[5]"numpy.meshgrid — NumPy v1.19 Manual", *Numpy.org*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.meshgrid.html. [Accessed: 22- Oct- 2020].

[6]"Vector and spaces | Linear algebra", 2020. [Online]. Available: https://www.khanacademy.org/math/linear-algebra/vectors-and-spaces/vectors/v/linear-algebra-parametric-representations-of-lines?modal=1. [Accessed: 23- Oct- 2020].

[7]"Calculus II - The 3-D Coordinate System", *Tutorial.math.lamar.edu*, 2020. [Online]. Available: https://tutorial.math.lamar.edu/classes/calcii/3dcoords.aspx. [Accessed: 25- Oct- 2020].

Appendix

This is the GitHub Repository https://github.com/RovilSurioJr/Laboratory-3