Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 4

Vector Operations

Submitted by:
Surio, Rovil Jr., M.

Instructor:
Engr. Dylan Josh D. Lopez

October 25, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of doing the operations between the vectors using Python and to visualize the vector operations.

II. Methods

- The practices of the activity include being familiar with different operations that can be done between the array of vectors and to make functions that will solve operations following a certain formula.
    - The activity teaches and implies the techniques of doing operation between an array of vectors and visualizing the resulting vector using a 3D-plot.
- The deliverables of the activity are to provide a function that will solve the modulus of a vector using the Euclidean Norm formula, a function that will solve the inner product of two vectors using the inner product formula, and to code the operation of vectors given by the instructor also, explain its resulting vector using a 3D-plot.
    - It was achieved by implementing loops on the function created and provide specific operations to solve the modulus of vector and inner product of two vectors. The expected value in the last task was achieved by implementing the operation vectors introduced in this laboratory and it was visualize using a 3D-plot.

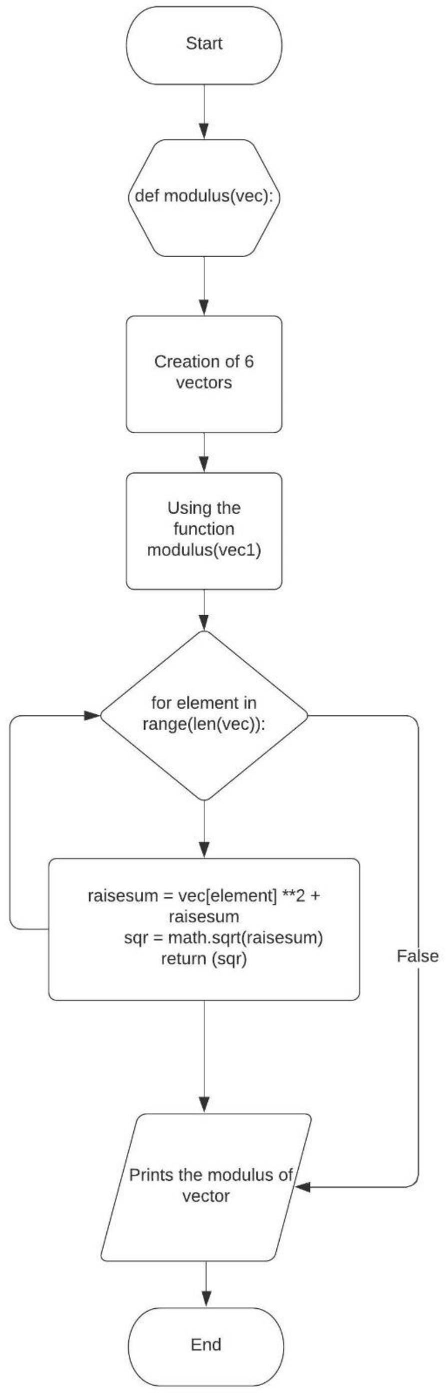

Start
def modulus(vec):
Creation of 6 vectors
Using the function modulus(vec1)
for element in range(len(vec)):
raisesum = vec[element] **2 + raisesum
sqr = math.sqrt(raisesum)
return (sqr)
False
Prints the modulus of vector
End

Figure 1 The flowchart of task 1

Figure 1 shows the flowchart of the function created that solves the modulus of a vector. A function named ‘modulus’ was defined first and it has a parameter ‘vec’ so that it can receive the value of the vector. When the values of vectors are created and were passed into the ‘vec’ of the ‘modulus’ function, the loop will be taking that values and it will perform the operations created inside the loop and the number of times it will execute the loop is based on the number of items in a created list of values or length of the vector. Afterward, the computed values will be returned and printed.

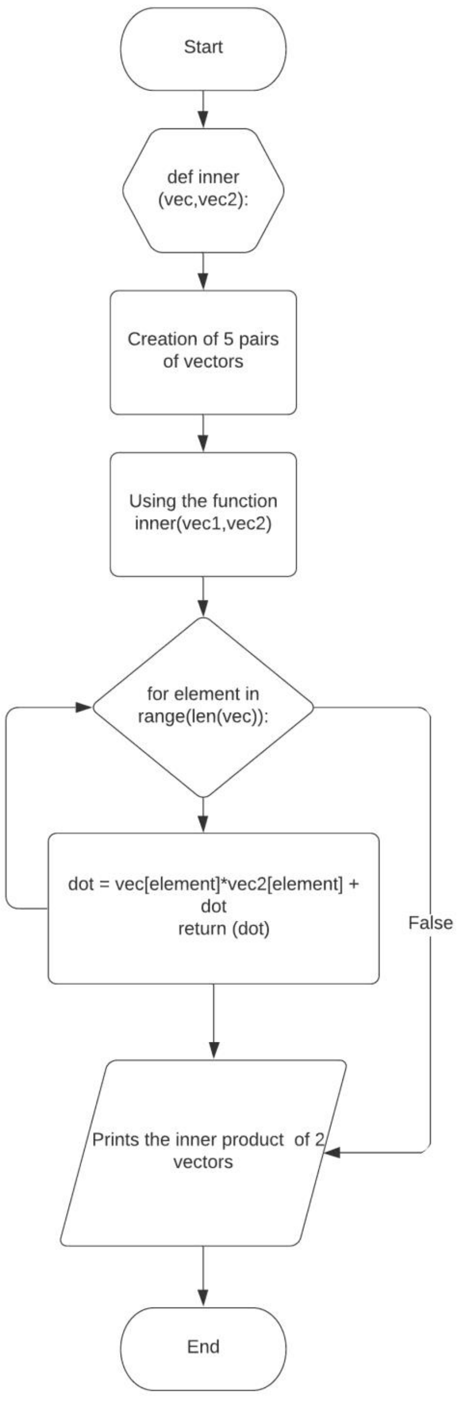

Start
def inner (vec,vec2):
Creation of 5 pairs of vectors
Using the function inner(vec1,vec2)
for element in range(len(vec)):
dot = vec[element]*vec2[element] + dot
return (dot)
False
Prints the inner product of 2 vectors
End

Figure 2 The flowchart of task 2

Figure 2 shows the flowchart of the code for the function created that solves the inner product of the vectors using the inner product formula. The function was defined first and when the values of the pair of vectors were passed to the parameter of this function, these values will be used in the loop created that contains the operations to solve the inner product of the 2 vectors. The loop will continue looping until the elements of vectors were used and the result can be printed after the result from the loop was returned.

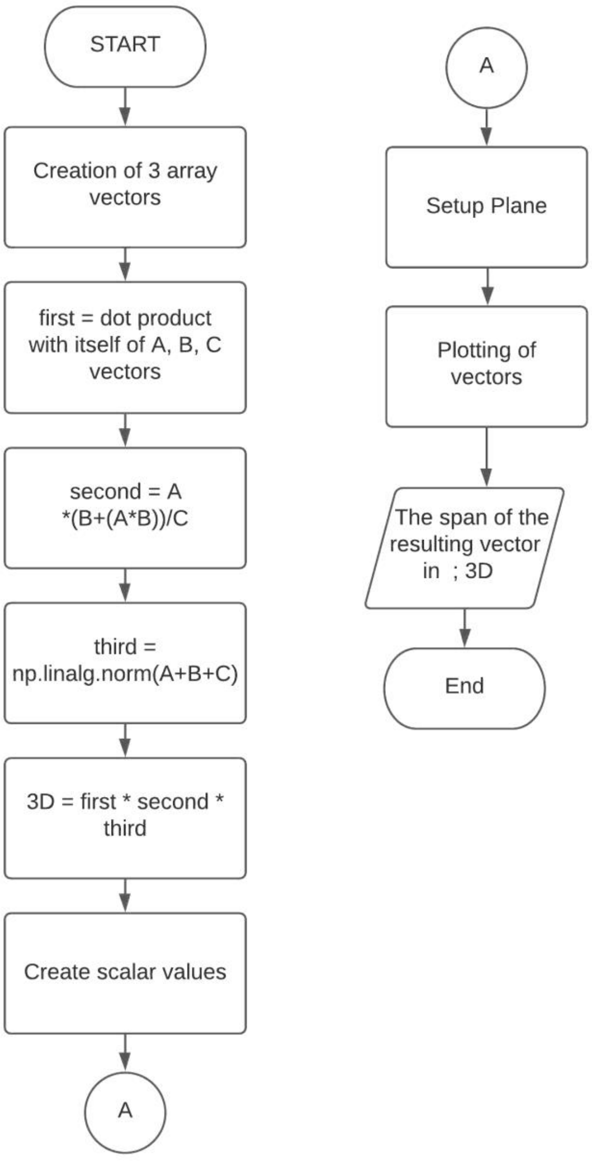
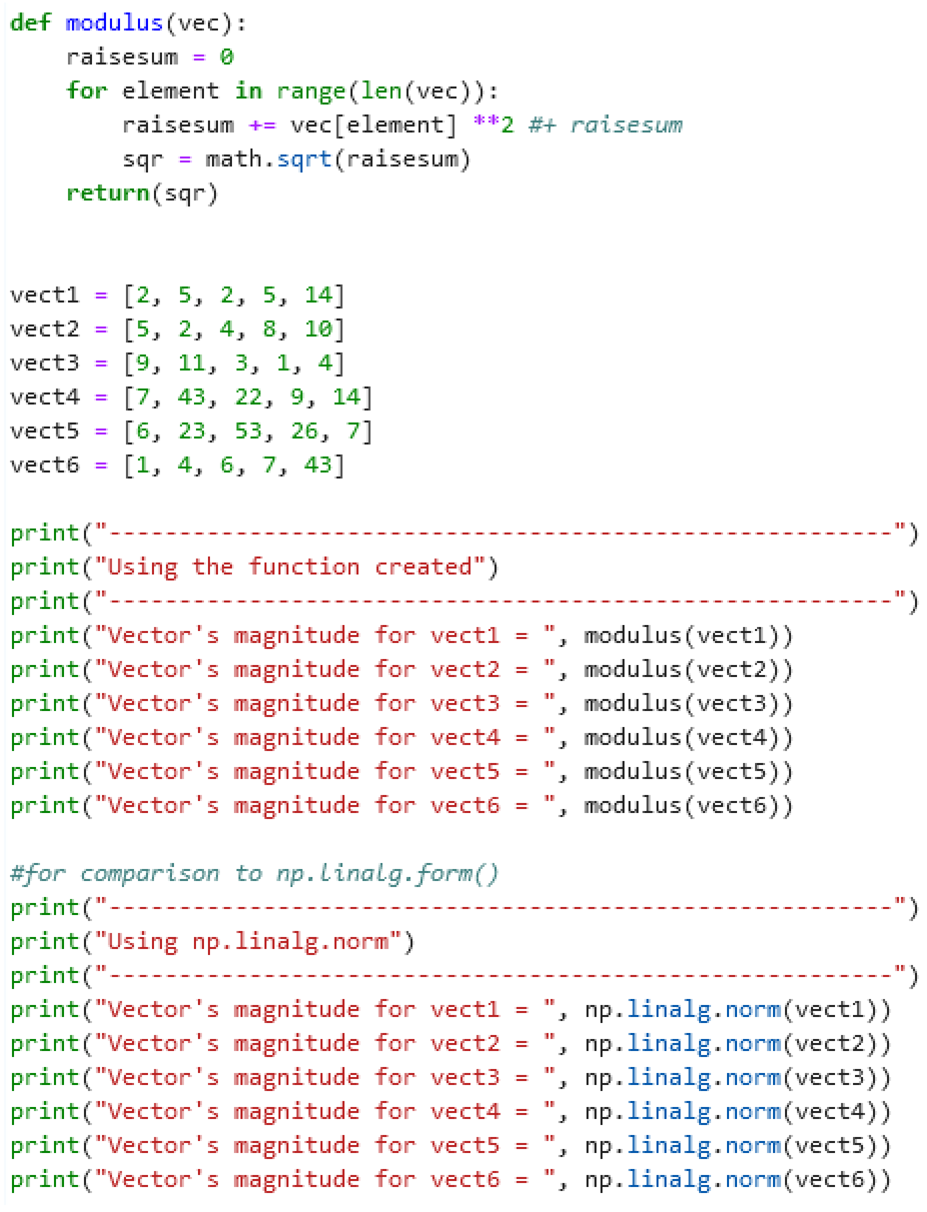

START
Creation of 3 array vectors
first = dot product with itself of A, B, C vectors
second = A *(B+(A*B))/C
third = np.linalg.norm(A+B+C)
3D = first * second * third
Create scalar values
A
A
Setup Plane
Plotting of vectors
The span of the resulting vector in ; 3D
End

Figure 3 The flowchart of task 3

Figure 3 shows the algorithm created to get the resulting vector of the 3 vectors by doing the operations on the given vectors. The creation of 3 array vectors given by the instructor was done first. The operations are sliced into 3 steps to easily visualize or to make the debugging on the program easier if the result doesn’t match the expected value of the resulting vector. Followed by arranging values to be used as scalar values and setting up the 3-dimensional plane. Lastly, it was plotted and shown at the same time with the resulting vector.

III. Results

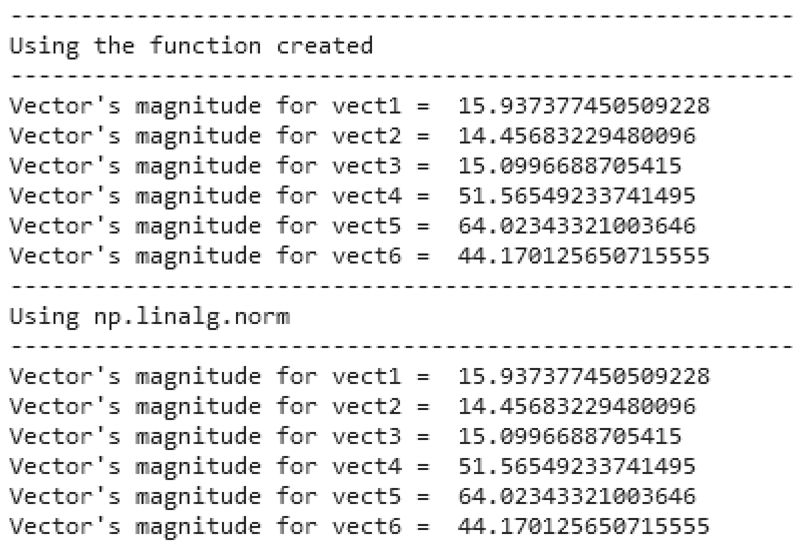
The result of the activities done by implementing the techniques and principles of doing the different operations between vectors to solve specific problems like getting the modulus of vectors or the inner product of two vectors are presented in this section.



```
def modulus(vec):
    raisesum = 0
    for element in range(len(vec)):
        raisesum += vec[element] **2 #+ raisesum
        sqr = math.sqrt(raisesum)
    return(sqr)


vect1 = [2, 5, 2, 5, 14]
vect2 = [5, 2, 4, 8, 10]
vect3 = [9, 11, 3, 1, 4]
vect4 = [7, 43, 22, 9, 14]
vect5 = [6, 23, 53, 26, 7]
vect6 = [1, 4, 6, 7, 43]

print("--------------------------------------------------------")
print("Using the function created")
print("--------------------------------------------------------")
print("Vector's magnitude for vect1 = ", modulus(vect1))
print("Vector's magnitude for vect2 = ", modulus(vect2))
print("Vector's magnitude for vect3 = ", modulus(vect3))
print("Vector's magnitude for vect4 = ", modulus(vect4))
print("Vector's magnitude for vect5 = ", modulus(vect5))
print("Vector's magnitude for vect6 = ", modulus(vect6))

#for comparison to np.linalg.form()
print("--------------------------------------------------------")
print("Using np.linalg.norm")
print("--------------------------------------------------------")
print("Vector's magnitude for vect1 = ", np.linalg.norm(vect1))
print("Vector's magnitude for vect2 = ", np.linalg.norm(vect2))
print("Vector's magnitude for vect3 = ", np.linalg.norm(vect3))
print("Vector's magnitude for vect4 = ", np.linalg.norm(vect4))
print("Vector's magnitude for vect5 = ", np.linalg.norm(vect5))
print("Vector's magnitude for vect6 = ", np.linalg.norm(vect6))
```

Figure 4 The code for modulus function

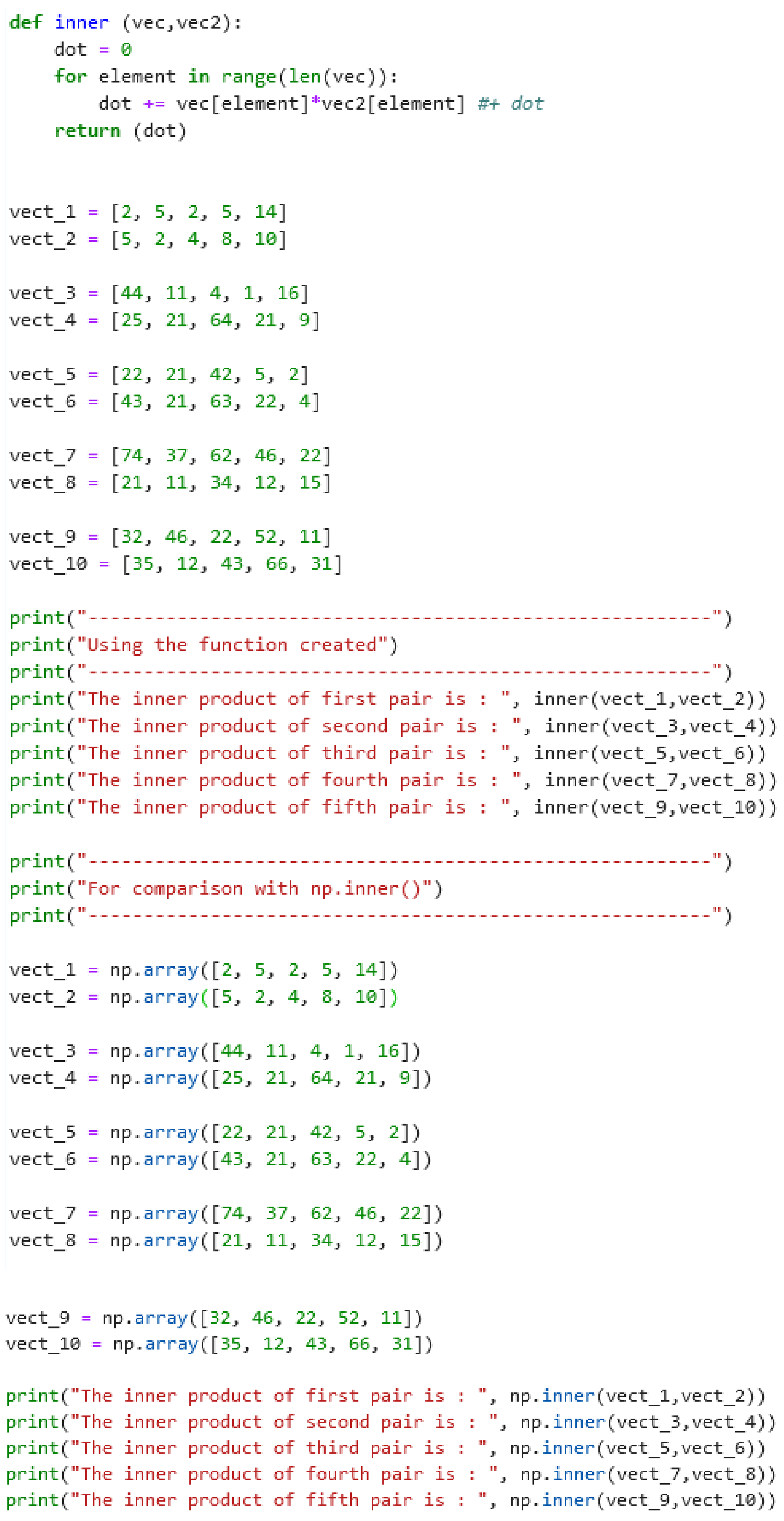
Figure 4 shows the code of block created to have a function that calculates the modulus without using the function from the NumPy library. The variable 'raisesum' was set to 0 so that the assignment of this variable is possible at the same time that it was used as referenced. A loop statement was created to access every element inside the vectors created and inside this loop statement are the operations that will compute the modulus of the 6 different vectors. The operations include the summation of the squared elements from the vectors. The sum of these elements was assigned to the ‘raisesum’ variable and the square root of this was assigned to ‘sqr’ which was returned afterward and printed. The second part of the code is for the comparison between the function created and the function from the NumPy library.



```
-----------------------------------------------------
Using the function created
-----------------------------------------------------
Vector's magnitude for vect1 =  15.937377450509228
Vector's magnitude for vect2 =  14.45683229480096
Vector's magnitude for vect3 =  15.0996688705415
Vector's magnitude for vect4 =  51.56549233741495
Vector's magnitude for vect5 =  64.02343321003646
Vector's magnitude for vect6 =  44.170125650715555
-----------------------------------------------------
Using np.linalg.norm
-----------------------------------------------------
Vector's magnitude for vect1 =  15.937377450509228
Vector's magnitude for vect2 =  14.45683229480096
Vector's magnitude for vect3 =  15.0996688705415
Vector's magnitude for vect4 =  51.56549233741495
Vector's magnitude for vect5 =  64.02343321003646
Vector's magnitude for vect6 =  44.170125650715555
```

Figure 5 The Outputs from two function

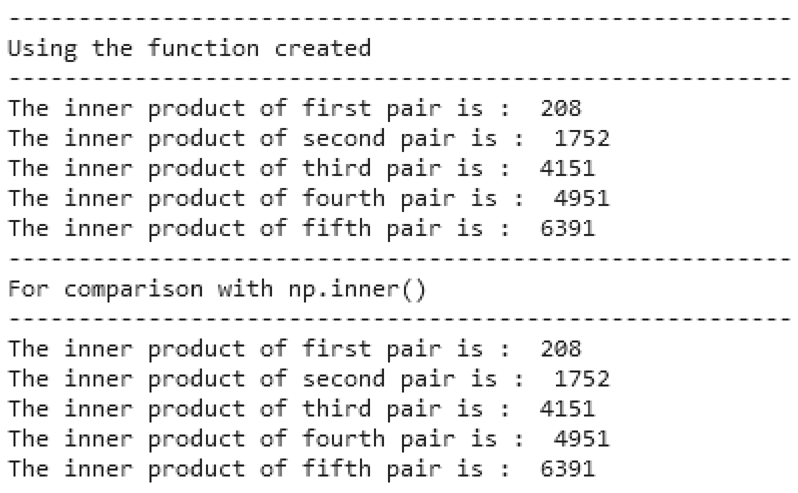
Figure 5 shows the output from using the modulus function created and the output from using the np.linalg.norm() function from NumPy library. It can be observed that using the basic operations in the modulus function created produced the same result with the np.linalg.norm() function.



```python
def inner (vec,vec2):
    dot = 0
    for element in range(len(vec)):
        dot += vec[element]*vec2[element] #+ dot
    return (dot)


vect_1 = [2, 5, 2, 5, 14]
vect_2 = [5, 2, 4, 8, 10]

vect_3 = [44, 11, 4, 1, 16]
vect_4 = [25, 21, 64, 21, 9]

vect_5 = [22, 21, 42, 5, 2]
vect_6 = [43, 21, 63, 22, 4]

vect_7 = [74, 37, 62, 46, 22]
vect_8 = [21, 11, 34, 12, 15]

vect_9 = [32, 46, 22, 52, 11]
vect_10 = [35, 12, 43, 66, 31]

print("--------------------------------------------------------")
print("Using the function created")
print("--------------------------------------------------------")
print("The inner product of first pair is : ", inner(vect_1,vect_2))
print("The inner product of second pair is : ", inner(vect_3,vect_4))
print("The inner product of third pair is : ", inner(vect_5,vect_6))
print("The inner product of fourth pair is : ", inner(vect_7,vect_8))
print("The inner product of fifth pair is : ", inner(vect_9,vect_10))

print("--------------------------------------------------------")
print("For comparison with np.inner()")
print("--------------------------------------------------------")

vect_1 = np.array([2, 5, 2, 5, 14])
vect_2 = np.array([5, 2, 4, 8, 10])

vect_3 = np.array([44, 11, 4, 1, 16])
vect_4 = np.array([25, 21, 64, 21, 9])

vect_5 = np.array([22, 21, 42, 5, 2])
vect_6 = np.array([43, 21, 63, 22, 4])

vect_7 = np.array([74, 37, 62, 46, 22])
vect_8 = np.array([21, 11, 34, 12, 15])


vect_9 = np.array([32, 46, 22, 52, 11])
vect_10 = np.array([35, 12, 43, 66, 31])

print("The inner product of first pair is : ", np.inner(vect_1,vect_2))
print("The inner product of second pair is : ", np.inner(vect_3,vect_4))
print("The inner product of third pair is : ", np.inner(vect_5,vect_6))
print("The inner product of fourth pair is : ", np.inner(vect_7,vect_8))
print("The inner product of fifth pair is : ", np.inner(vect_9,vect_10))
```

Figure 6 The code for inner product function

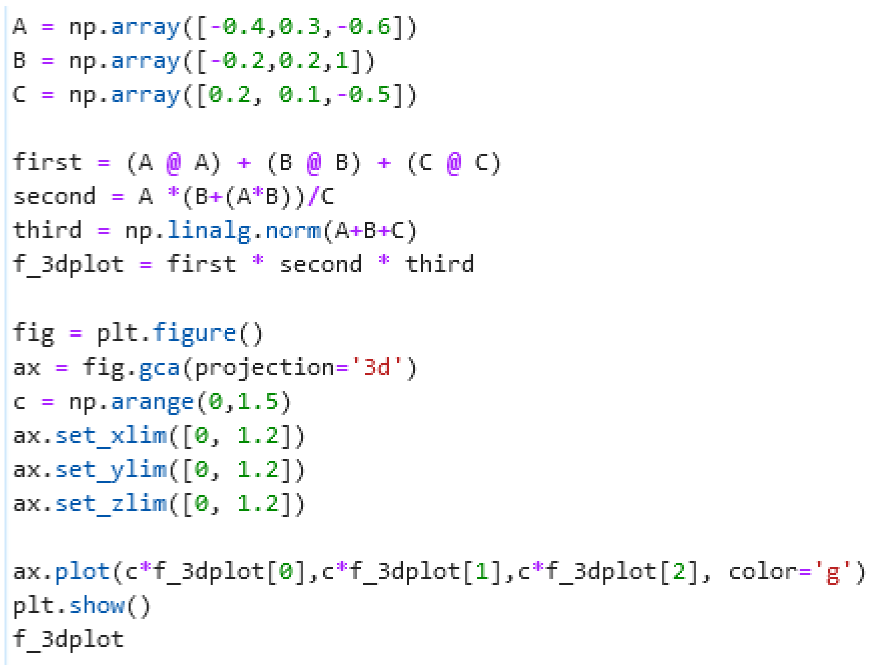
Figure 6 shows the code of block created to show the difference between the results from using the function from the NumPy and the function created that involves vector operations. The inner() function created contains a loop that will run based on the number of elements inside the list created to represent a vector. It gets the values from the parameters of inner() function which is the 'vec' and 'vec2' that holds the values being passed when the list of vectors used as an argument when calling the function or in code, 'inner(vect_1,vect_2)'. The operations inside the loop now have an access to the elements hence it can do the operations properly and return the result.



```
---------------------------------------------------------
Using the function created
---------------------------------------------------------
The inner product of first pair is :  208
The inner product of second pair is :  1752
The inner product of third pair is :  4151
The inner product of fourth pair is :  4951
The inner product of fifth pair is :  6391
---------------------------------------------------------
For comparison with np.inner()
---------------------------------------------------------
The inner product of first pair is :  208
The inner product of second pair is :  1752
The inner product of third pair is :  4151
The inner product of fourth pair is :  4951
The inner product of fifth pair is :  6391
```

Figure 7 The output from two functions

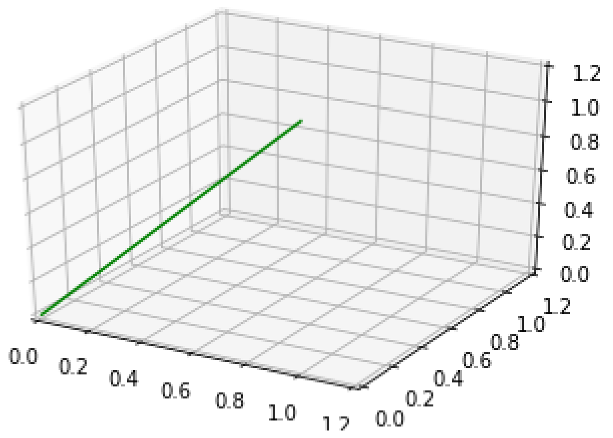
Figure 7 shows the output from using the function created and using the np.inner() function from the NumPy library. It can be observed that the resulting values from using the function created are identical with the resulting values from using the np.inner() function.



```
A = np.array([-0.4,0.3,-0.6])
B = np.array([-0.2,0.2,1])
C = np.array([0.2, 0.1,-0.5])

first = (A @ A) + (B @ B) + (C @ C)
second = A *(B+(A*B))/C
third = np.linalg.norm(A+B+C)
f_3dplot = first * second * third

fig = plt.figure()
ax = fig.gca(projection='3d')
c = np.arange(0,1.5)
ax.set_xlim([0, 1.2])
ax.set_ylim([0, 1.2])
ax.set_zlim([0, 1.2])

ax.plot(c*f_3dplot[0],c*f_3dplot[1],c*f_3dplot[2], color='g')
plt.show()
f_3dplot
```

Figure 8 The code for solving the vector operation

Figure 8 shows the block of code created to solve the vector operation using the given vector values. After analyzing the problem, the first part of the operation was coded first, and it can be observed that the square of the vector was done by getting the dot product of itself because according to [1], “The projection of a vector on to itself leaves its magnitude unchanged, the dot product of any vector with itself is the square of that vector's magnitude.” It was followed by calculating the second and third parts then the multiplication of each part to get the resulting vector. To visualize the resulting vector, the values were plotted into a 3-dimensional plane using the plot instead of the quiver to see how the continuous line of this behaves or to show all the possible vectors it can reach using the scalar values.



```
array([0.34769805, 1.13001866, 0.6953961 ])
```

Figure 9 The resulting vector in a 3D plane

Figure 9 shows the resulting vector and the visualization of it using a 3D plot. The line represents the transformed values from the resulting vector by scaling the values with the arbitrary constant or by multiplying each of these into the scalar values created to show the span of it. The values achieved are the same as the expected values by the instructor.

IV. Conclusion

This laboratory demonstrated the concepts behind the different principles and techniques that can be implemented when working on the operation between vectors. I got to familiarize myself with the different operations that can be used or the other forms of the specific operation. For example, when dividing, the function np.divide() can be used or simply the slash ‘/’. Moreover, the other operations too can be done just by using the Python built-in function but using the NumPy library functions can make our life easier for solving specific problems that require multiple operations such as getting the magnitude of a vector. It was shown in this laboratory through discussion and the activities how long the code will be if we try to make a function to solve using the formula compare by using the functions from NumPy library specifically the np.linalg.norm() for computing the magnitude of a vector and the np.inner() for computing the inner or dot product of two vectors. I also learned through the activity given that the dot product of any vector with itself is the square of that vector's magnitude [1]. For me, this is the greatest knowledge I achieved in this laboratory because I spent a lot of time trying to find out what is wrong when I am doing task 3 and upon searching, I found out that the square of a vector is the square of its magnitude and not just the square of its given values.

References

[1]"Vector Multiplication – The Physics Hypertextbook", *The Physics Hypertextbook*, 2020. [Online]. Available: https://physics.info/vector-multiplication/. [Accessed: 24- Oct- 2020].

Appendix

This is the GitHub Repository https://github.com/RovilSurioJr/Laboratory-4