Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 7

Matrix Operations

Submitted by:
Gatchalian Paulo Valentin M.

Instructor:
Engr. Dylan Josh D. Lopez

April, 30, 2021

I. Objectives

This laboratory activity aims to give knowledge to the students to be familiar of the fundamentals of matrix operations through the application of python programming.

II. Methods

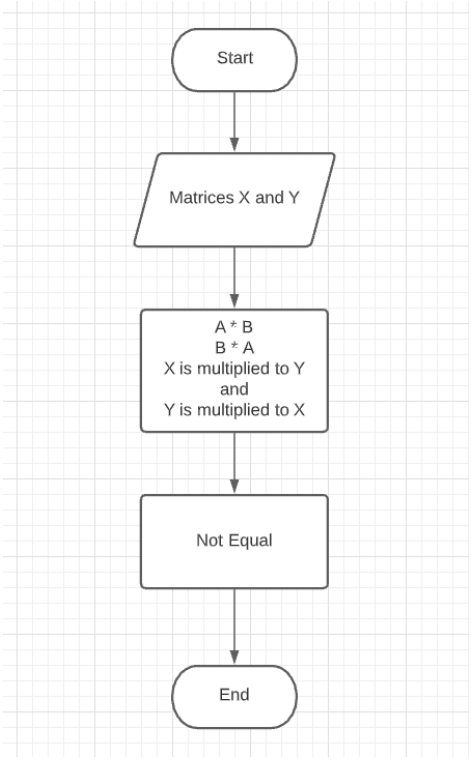

Figure 1: Flowchart for the 1st task

Figure 1 represents the commutative property in the given task wherein the flowchart represents the commutative flow on how the program was executed in a step by step process.

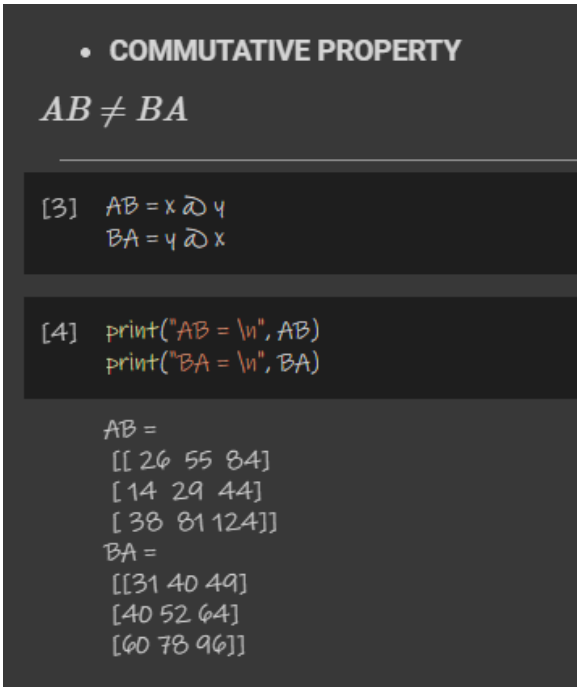

Figure 2: Codes for task 1 (commutative prorperty)

Figure 2 represents the commutative property through the application of python programming wherein A and B identifies the different values of X and Y in order to identify that the values of AB and BA are not equal to each other.

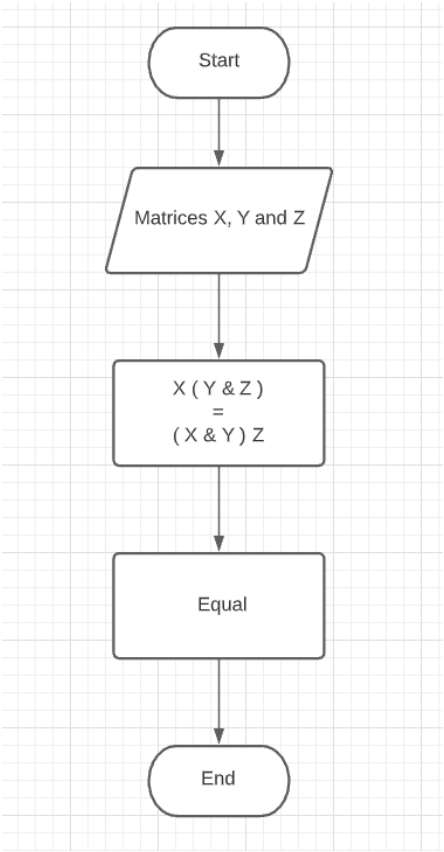

Figure 3: Flowchart of the 2nd property

The 2nd flowchart represents the step by step process of the 2nd property wherein it asks for the matrix X and Y and Z and multiplied to the product of Y and Z to X then multiplies the product of X and Y to the values of C in order to check if it is equal so that the program will end its process.

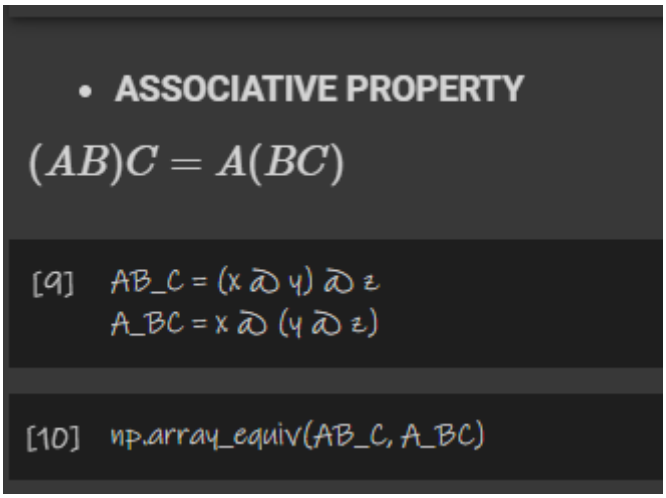

Figure 4: Codes used for the 2nd property (Associative Property)

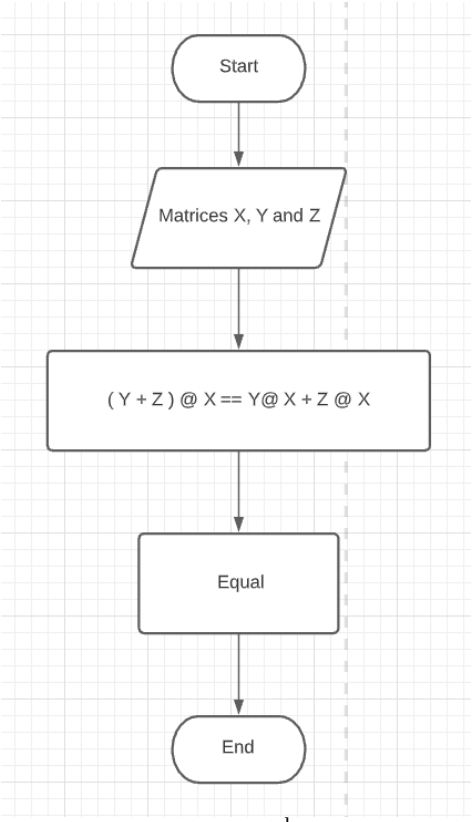

Figure 5: Flowchart of the 3rd Property (Distributive)

Figure 5 shows the process of the distributive property in a step by step process wherein the figure lets us know that the matrix multiplies the sum of Y and Z to X and if it is equal to the sum of the products of Y and X and Z and X.

```
[6]  A_BC = x @ (y+z)
     AB_AC = (x @ y) + (x @ z)

[17] print("A(B+C) = \n", A_BC)
     print("AB + AC = \n", AB_AC)
```

Figure 6: Codes used in the 3rd Property (Distributive)

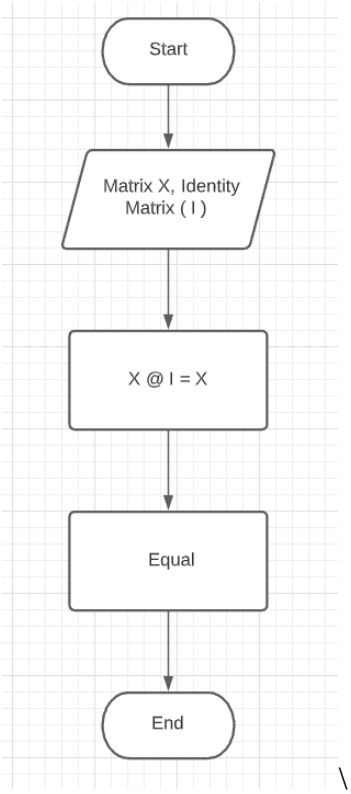

Figure 7: Represents the Flowchart in the Multiplicative Identity Property

Figure 7 shows that the visualization of the identity matrix wherein if X is multiplied by I and the result shows that it is equal to its array.

```
I = np.eye(3, dtype = int)
AI = x @ I

[12] print("AI = \n", AI)
     print("A = \n", x)
```

Figure 8: Codes used in the Multiplicative Identiy Property

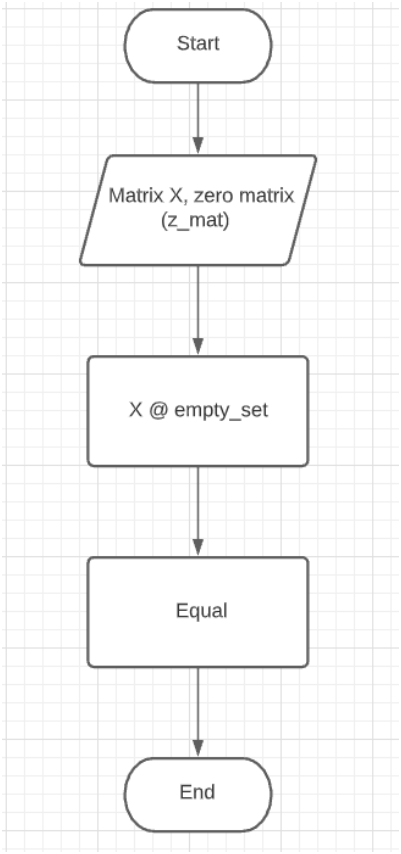

Figure 9: Flowchart of Multiplicative Property of Zero

Figure 9 tackles about the zero propery of multiplication where it first calls the matrix A and the zero matrix then it would check if its is equal, if we are going to follow the zero property it would be equal if it is not errors will occur during the process of the codes.

III. Results

```
AB =
[[ 26  55  84]
 [ 14  29  44]
 [ 38  81 124]]
BA =
[[31 40 49]
 [40 52 64]
 [60 78 96]]
```

Figure 10: Result of the Commutative property

Figure 10 Shows the result of the Commutative property wherein it identifies the process of the codes to represent the commutative property through the application of python programming.

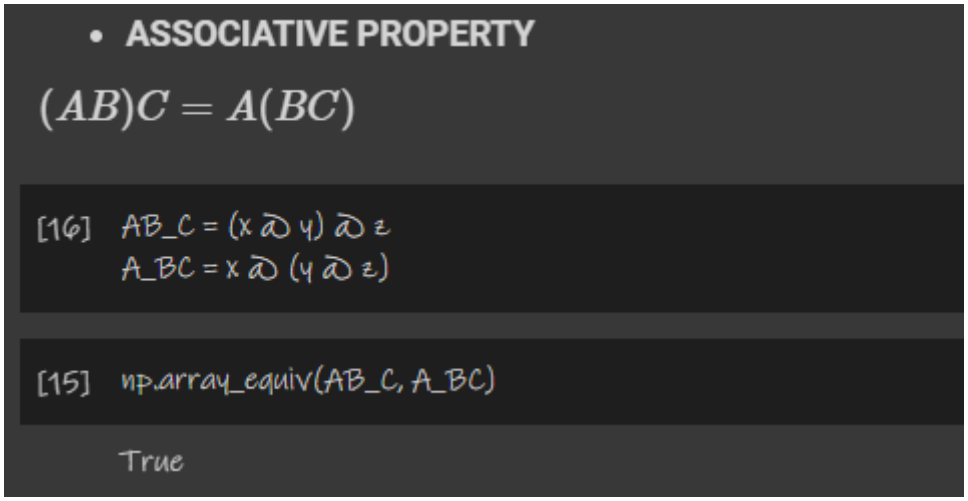

Figure 11: Result of the Associative property

Figure 11 is the result of the Associative property whereas it represents the application of pyhon programming using the Associative property application.

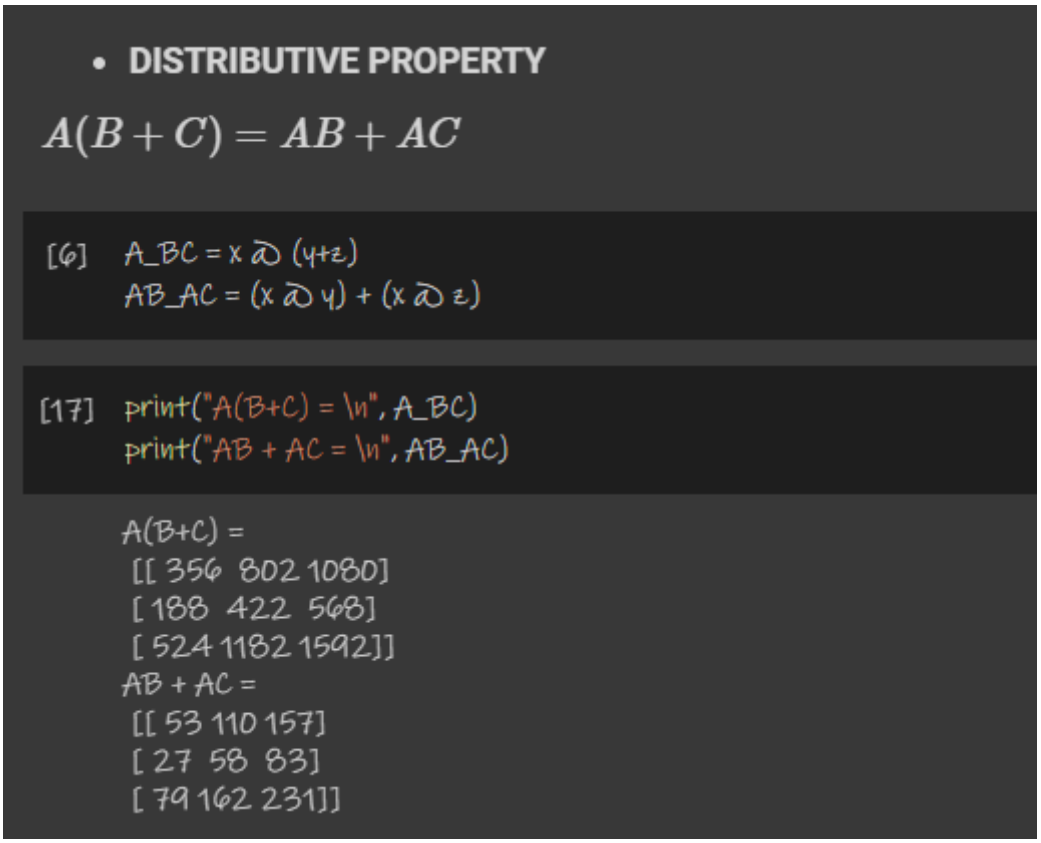

Figure 12: Result of the Distributive property

- **MULTIPLICATIVE IDENTITY PROPERTY**

$$AI = A$$

```
[11]  I = np.eye(3, dtype = int)
      AI = x @ I

[12]  print("AI = \n", AI)
      print("A = \n", x)

      AI =
       [[3 4 5]
       [1 2 3]
       [5 6 7]]
      A =
       [[3 4 5]
       [1 2 3]
       [5 6 7]]

[ ]   np.array_equiv(AI, x)

      True
```

Figure 13: Result of the Multiplicative Identity Property

- **MULTIPLICATIVE PROPERTY OF ZERO**

$$A0 = 0$$

```
[ ]   zero = np.zeros(x.shape)
      A_zero = x @ zero

[ ]   print("A0 = \n", A_zero)
      print("zero = \n", zero)

      A0 =
       [[0. 0. 0.]
       [0. 0. 0.]
       [0. 0. 0.]]
      zero =
       [[0. 0. 0.]
       [0. 0. 0.]
       [0. 0. 0.]]

[ ]   np.array_equiv(zero, A_zero)

      True
```

Figure 14: Result of the Multiplicative Identity Property

IV. Conclusion

In Conclusion, the activity gives knowledge about the Matrix Operations using python programming it also adds additional information in order to process Matrix Operaions in different applicaions.

Marix Operations can help through the use to Assess Patient canre in terms of Improvement and Core Competencies such as helping the improvement of different kinds of systematic process that will help the healthworkers provide faster response to each individual cases and as well as the process in improving the workflow of each healthcare environment.

References

[1] D.J.D. Lopez. “Adamson University Computer Engineering Department Honor Code,” AdU-CpE Departmental Policies, 2020.

[[Online]. Available; https://www.tutorialspoint.com/matrix-manipulation-in-python[AccessedMay 03, 2021]

[[Online]. Available; https://www.w3schools.com/python/numpy/numpy_creating_arrays.asp[Accessed May 03, 202]

Github repo: https://github.com/Gatchplease/LinAlg-Lab7