Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 7

Matrix Operations

Submitted by:
Reyes, Carl Vincent G.

Instructor:
Engr. Dylan Josh D. Lopez

December 19, 2020

I. Objectives

This laboratory activity aims to teach the students to be familiar with the fundamentals of matrix operations, and to enable them to apply different operations to solve intermediate equations given to them. This will also teach the students to apply matrix algebra in engineering solutions, enhancing their knowledge with coding and being better in using jupyter notebook.

II. Methods

In this laboratory, the deliveribales are to perform the given multiplication matrix properties which includes associative property, commutitative property, multiplicative identity, dimension property and the zero property of multiplication using the NumPy libraries such as the one we are most familiar with is the np.array(), then we have the np.eye(), np.zeros, np.shape(), and the np.array_equal/equiv(). This libraries we used all have different applications that were used in order to come up with the right answers. The use of np.array() is for data manipulation whereas you can determine the shape of your vector [1] which was done earlier , and from its name it was used to create an array of numbers, then we have np.eye() which returns an array where all all elements are equals to zero [2], as for np.zeros() it was used to create a new array of any given shape and type filled with zeros [3]. The use of np.array_equal is to prove if what you are comparing is equal, It is similar to the mathematical symbol equals (=) hence, its name.

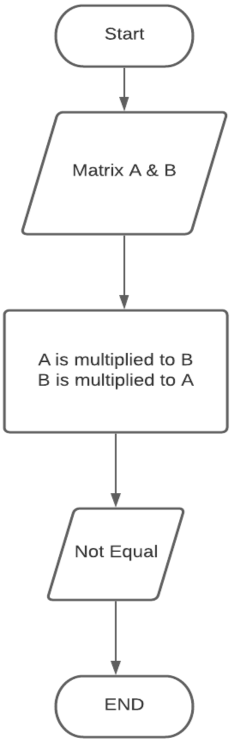

Figure 1: Flowchart for 1st task

As seen in figure 1, it shows the commutative part of the activity given, a step-by-step process whereas it starts then creates the Matrix that will be multiplied and then process it, if it is proven that it is not equal then the program will end.

```
In [35]: A = np.array([
             [2,3,4],
             [5,1,2],
             [4,4,4],

         ])
         B = np.array([
             [1,2,3],
             [4,5,6],
             [1,4,8],

         ])
         A.dot(B)

Out[35]: array([[18, 35, 56],
                [11, 23, 37],
                [24, 44, 68]])

In [36]: B.dot(A)

Out[36]: array([[24, 17, 20],
                [57, 41, 50],
                [54, 39, 44]])

In [37]: np.array_equiv(A,B)
```

Figure 2:Codes used for the 1st task

These is the codes used for the commutative, in order to demonstrate fully, the programmer first multiplied a and b then ran the code then multiplying B and A, then using the np.array_equiv in order to know if the results of the commutative property would be equal or not.

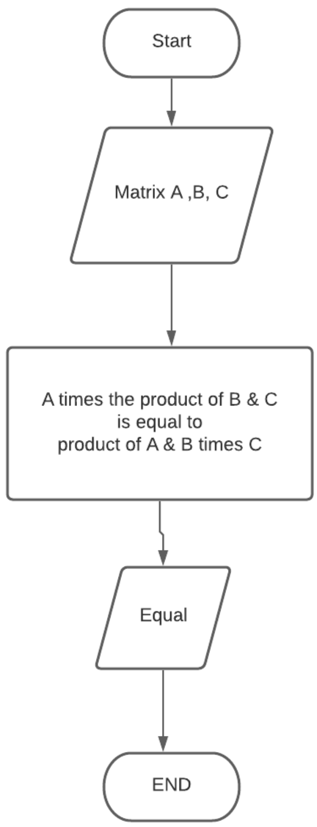

Figure 3: flowchart for 2nd number

The flowchart for number two asks the matrix for the A, B, and C then it multiplies the product of B and C to A, then multiplies the product of A and B to C and checks if it is equal, and if it is, it ends the process.

```
In [42]: A @ (B @ C)

Out[42]: array([[14, 20, 26, 34],
                [32, 47, 62, 81],
                [38, 54, 70, 94],
                [ 6,  8, 10, 12]])

In [43]: (A @ B) @ C

Out[43]: array([[14, 20, 26, 34],
                [32, 47, 62, 81],
                [38, 54, 70, 94],
                [ 6,  8, 10, 12]])

In [46]: np.array_equal(A @ (B @ C), (A @ B) @ C)
```

Figure 4: Codes used for number 2

This is the code used for the question 2 where the programmer again, mutlpied the first part of the equation and then the one on the right side and checks using np.array_equal if it is equal.

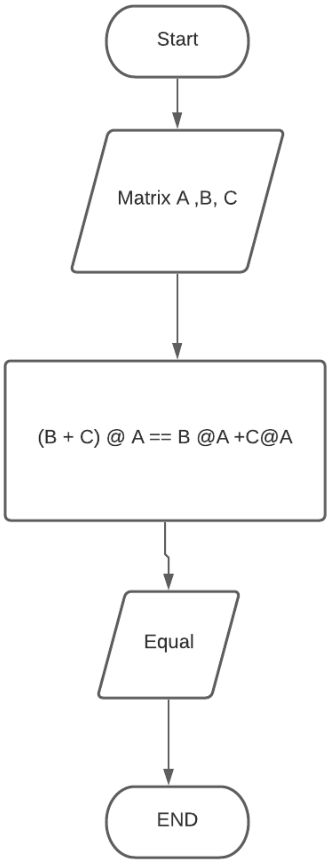

Figure 5: Flowchart for number 3

As for the figure 5, it again starts within the matrix then multiplies the sum of B and C to A and if it is equal to the sum of the products of B times A and C times A.

3. $A \cdot (B + C) = A \cdot B + A \cdot C$

```
: np.array_equal(A @ (B + C), A @ B + A @ C)
```

Figure 6: Codes used for number 3

These are the codes used for number 3 which focuses on distributive property of multiplication, seend from figure 6 is that np.array_equal was used in order to know if the equation from the left is equal to the equation from the right.

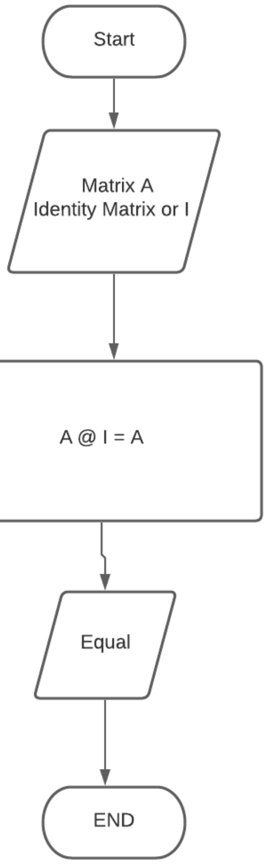

Figure 7: Flowchart for identity matrix

Figure 7 shows the flowchart for the 4^{th} number which is an identity matrix, where if A is multiplied by I it is still equal to to its original array.

$$5.\ A \cdot I = A$$

```
In [59]: I = np.eye(4)
         np.array_equal(A @ I, A)
```

Figure 8: Codes used for Identity Matrix

The codes used for figure 8 is np.eye which was discussed above creates a diagonal of 1’s and 0’s somewhere thus multiplying the A to itself rendering it equal.

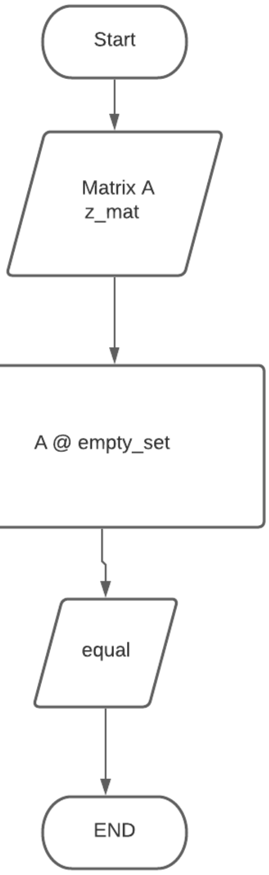

Figure 9: flowchart for zero property

As seen from figure 9, this is the flowchart of the 5th number which tackles zero property of multiplication where it first calls the matrix A and the z_mat or the zero matrix, then it would check if it is equal, if we are going to follow the zero property it would be equal if it is not, then there is something wrong with the code.

```
In [50]: z_mat = np.zeros(A.shape)
         z_mat

Out[50]: array([[0., 0., 0.],
                [0., 0., 0.],
                [0., 0., 0.]])

In [51]: a_dot_z = A.dot(np.zeros(A.shape))
         a_dot_z

Out[51]: array([[0., 0., 0.],
                [0., 0., 0.],
                [0., 0., 0.]])

In [52]: null_mat = np.empty(A.shape, dtype=float)
         null = np.array(null_mat,dtype=float)
         print(null)
         np.allclose(a_dot_z,null)
```

Figure 10: Codes used for zero property

These are the codes used by our professor, whereas an A matrix was created and the z_mat, then a function was created to determine the np.empty and then checks if they are equal.

III. Results

```
[54, 59, 44]])

In [37]: np.array_equiv(A,B)

Out[37]: False
```

Figure 11: Result from number 1

Figure 11 shows the results of the code from the method above, whereas it is similar to the equation proving it correct, thus the commutative property of multiplication was shown clearly.

```
[38, 54, 70, 94],
[ 6,  8, 10, 12]])
```

```
In [46]: np.array_equal(A @ (B @ C), (A @ B) @ C)
Out[46]: True
```

Figure 12: Result from number 2

Figure 12 shows the result of number 2 which Is true and in using np.array_equal it proves that the equation really is equals, thus showing the associative property of multiplication clearly.

3. $\boldsymbol{A} \cdot (\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{A} \cdot \boldsymbol{B} + \boldsymbol{A} \cdot \boldsymbol{C}$

```
In [55]: np.array_equal(A @ (B + C), A @ B + A @ C)
Out[55]: True
```

Figure 13: results from number 3

Figure 13 shows the result for number 3 which turns out to be true where the commutative property of multiplication was seen. The code that was used in order to identify if they were equal or not is the np.array_equal.

4. $(\boldsymbol{B} + \boldsymbol{C}) \cdot \boldsymbol{A} = \boldsymbol{B} \cdot \boldsymbol{A} + \boldsymbol{C} \cdot \boldsymbol{A}$

```
In [57]: np.array_equal((B + C) @ A , B @ A + C @ A)
Out[57]: True
```

Figure 14: Results from number 4

Figure 14 shows that the result from the np.array_equal of the matrix Is true, thus clearly showing the distributive property of multiplication.

5. $A \cdot I = A$

```
[59]: I = np.eye(4)
      np.array_equal(A @ I, A)
t[59]: True
```

Figure 15:results from number 5

Figure 15 shows the result of the number 15 which clearly states the indentity property of multiplication, and the result was true which is correct coming from the equation.

```
[0. 0. 0.]]
Out[52]: True
```

Figure 16:results from number 6

Figure 16 shows that the result from number 6 is true clearly stating the zero property of multiplication.

IV. Conclusion

This activity enhances the knowledge of every student that was able to successfully answer the remaining questions, it also adds additional topic that they will need in order to go further in their engineering course.

Matrix operations can help through diagnosing patients more accurately, according to the research of Dra. Shakila Banu, she uses a a matrix model called a frequency distribution matrix whereas, it can show the accurate symptoms of the disease and as well as the appropriate therapeutic actions needed in order to combat such disease. [4] With this it can vastly increase chances of survival and recovery of every patient if the diseases were to know accurately.

Github Repository:

https://github.com/ReyesCarl/LinAlg_Lab7

V. References

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

[1] [Online]. Available: https://jakevdp.github.io/PythonDataScienceHandbook/02.02-the-basics-of-numpy-arrays.html#:~:text=Data%20manipulation%20in%20Python%20is,reshape%2C%20and%20join%20the%20arrays.. [Accessed 17 12 2020].

[2] [Online]. Available: https://www.pythonprogramming.in/numpy-eye-array-example.html#:~:text=in%20your%20browser.-,NumPy%20Eye%20array%20example,values%20are%20equal%20to%20one.. [Accessed 17 12 2020].

[3] [Online]. Available: https://www.w3resource.com/numpy/array-creation/zeros.php#:~:text=The%20zeros()%20function%20is,and%20type%2C%20filled%20with%20zeros.&text=Shape%20of%20the%20new%20array,2%2C%203)%20or%202.&text=The%20desired%20data%2Dtype%20for,int8.. [Accessed 17 12 2020].

[4] D. P. S. Banu. [Online]. Available: http://www.ijsrp.org/research-paper-0613/ijsrp-p18144.pdf. [Accessed 18 12 2020].