Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 7

Matrix Operations

Submitted by:
Chipongian, John Patrick Ryan J.

Instructor:
Engr. Dylan Josh D. Lopez

December 22, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of matrix operations. It aims to familiarize matrix operation, its dot products and its properties. By the end of this laboratory activity, application of matrix operations to different engineering solutions should be possible.

II. Methods

In this laboratory activity, it discusses matrix operations and its properties. Firstly, the methods that were used in completing this laboratory activity are the numpy libraries and built-in functions of python. [1] The np.array() is used to create the matrices that would be used in the operations. [2] The np.eye() to create an identity matrix that would be used in completing one of the properties of the matrix. And lastly, [3] np.zeros to create a zero matrix that would also be used in completing one of the properties of the matrix.

```
Mat1 = np.array([
    [2,3,1],
    [6,7,4],
    [8,9,7]
])
Mat2 = np.array([
    [1,3,2],
    [4,7,6],
    [7,9,8]
])
Mat3 = np.array([
    [7,0,2],
    [-1,7,-5],
    [-3,1,-7]
])
```

Figure 1 Matrices that would be used in the activity

Figure 1 are the matrices that would be used in performing the properties of matrices in this laboratory activity. Other specific matrices, such as identity and zeros matrix, would be added to perform other properties.

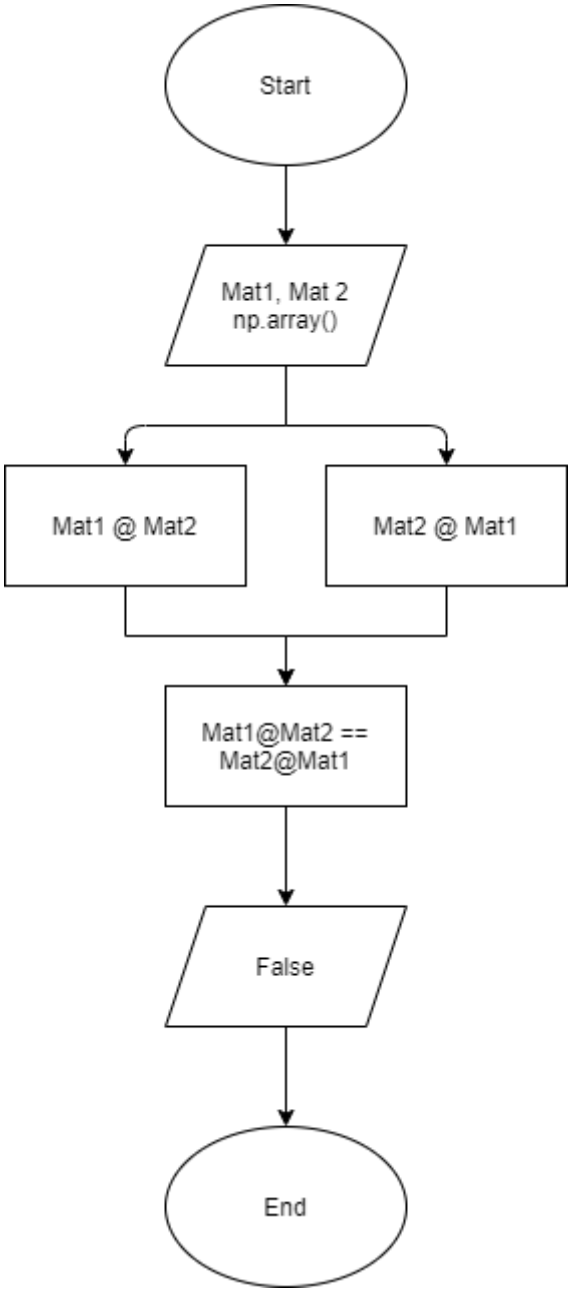

Figure 2 Flowchart for the first property

Figure 2 shows the flowchart of the first property of the matrix, the commutative property. This shows that the dot product of matrix 1 and matrix 2 are not equal to the dot product of matrix 2 and matrix 1.

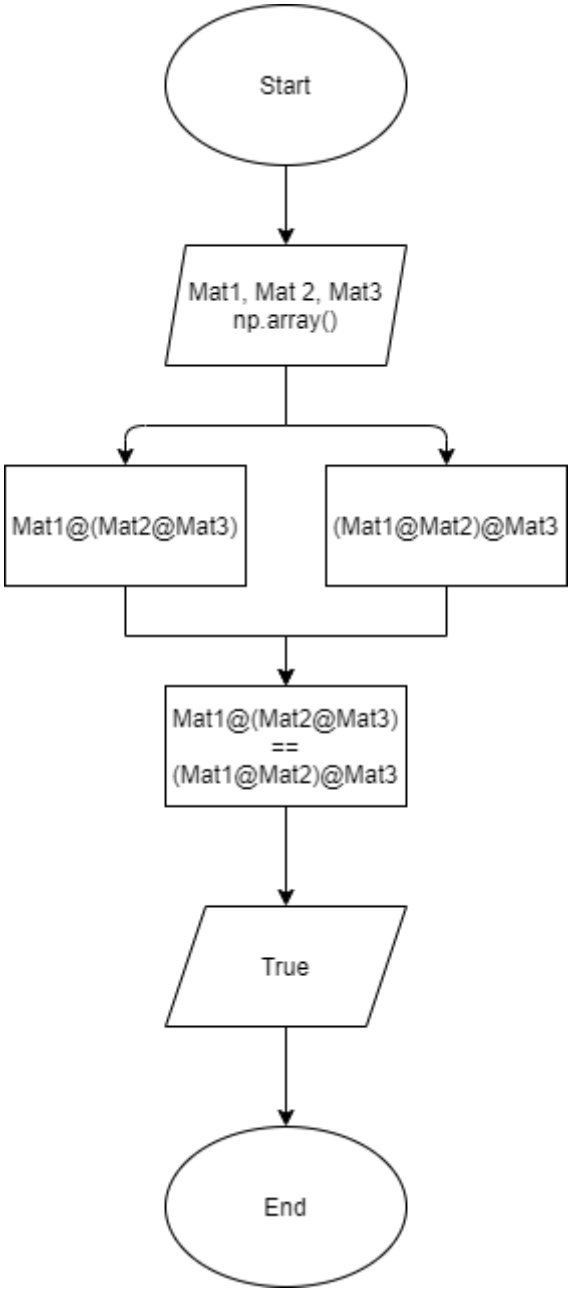

Figure 3 Flowchart for the second property

Figure 3 shows the flowchart of the second property of the matrix, the associative property. This shows that Mat1 @ (Mat2 @ Mat3) are equal to (Mat1 @ Mat2) @ Mat3.

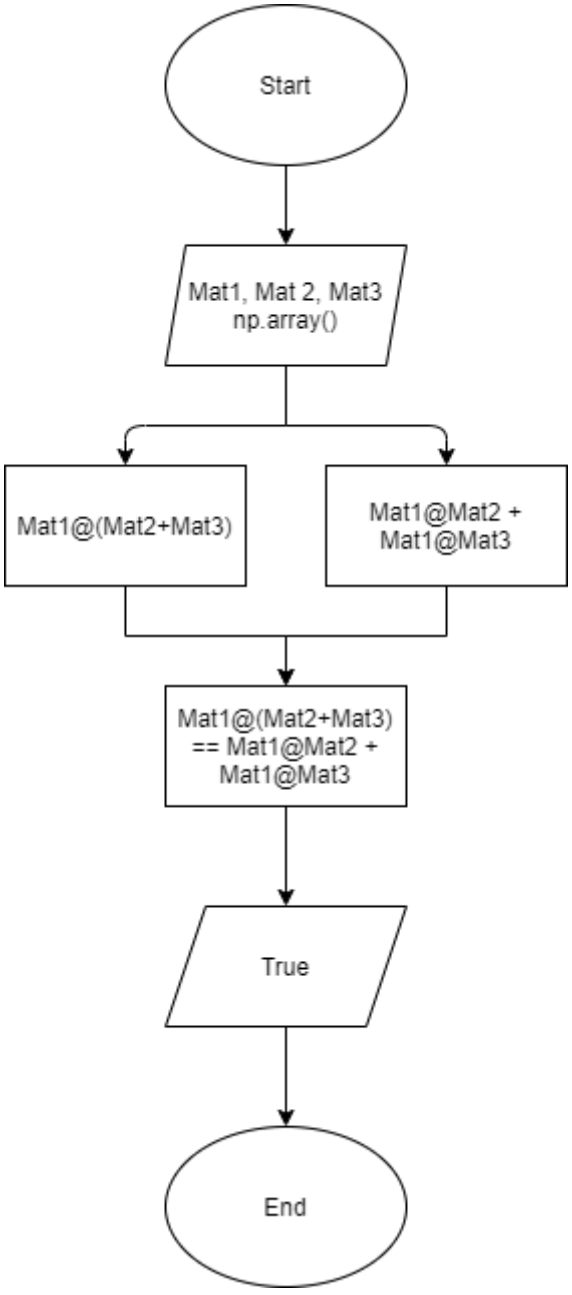

Figure 4 Flowchart for the third property

Figure 4 shows the flowchart of the third property of the matrix, the distributive property. This shows that Mat1 @ (Mat2 + Mat3) are equal to Mat1 @ Mat2 + Mat1 @ Mat3.

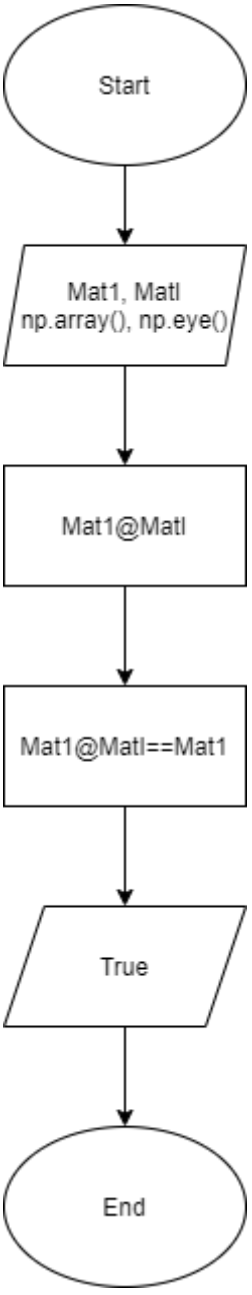

Figure 5 Flowchart for the fourth property

Figure 5 shows the flowchart of the property of matrix, multiplicative identity property. The dot product of Mat1 and the identity matrix MatI is first computed then it is proven equal.

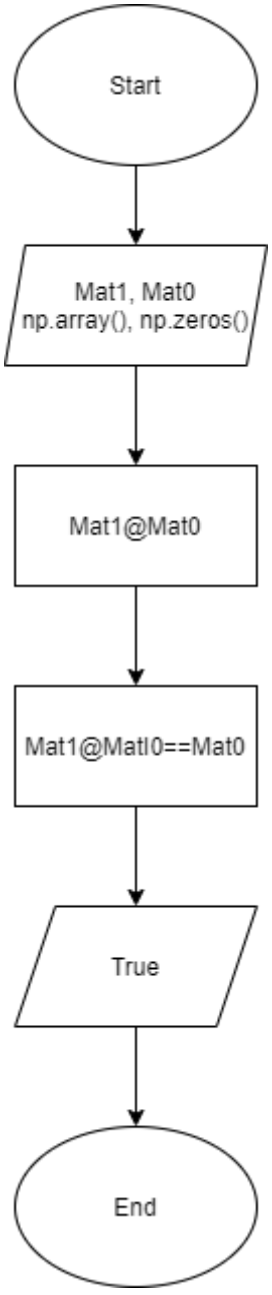

Figure 6 Flowchart for the fifth property

Figure 6 shows the flowchart of the last property of the matrix, the multiplicative property of zero. A matrix made of zeros, which is also called a zeros matrix, is firstly done. Then the dot product of Mat1 and Mat0 is computed. After that, it would be proven that Mat1 @ Mat0 is equal to Mat0.

III. Results

```
Mat1 @ Mat2
```

```
array([[ 21,  36,  30],
       [ 62, 103,  86],
       [ 93, 150, 126]])
```

```
Mat2 @ Mat1
```

```
array([[ 36,  42,  27],
       [ 98, 115,  74],
       [132, 156,  99]])
```

```
Mat1 @ Mat2 == Mat2 @ Mat1
```

```
array([[False, False, False],
       [False, False, False],
       [False, False, False]])
```

Figure 7 Codes and output for the first property

Figure 7 shows the codes of the commutative property of matrix together with its outputs. The dot product of Mat1 and Mat2 and the dot product of Mat2 and Mat1 are firstly computed. These two dot products would show different values. And thus, it would prove to be not equal.

```
Mat1 @ (Mat2 @ Mat3)
```

```
array([[   21,   282,  -348],
       [   73,   807,  -993],
       [  123,  1176, -1446]])
```

```
(Mat1 @ Mat2) @ Mat3
```

```
array([[   21,   282,  -348],
       [   73,   807,  -993],
       [  123,  1176, -1446]])
```

```
Mat1 @ (Mat2 @ Mat3) == ( Mat1 @ Mat2) @ Mat3
```

```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

Figure 8 Codes and output for the second property

Figure 8 shows codes of the associative property of matrix together with its outputs. The dot product of Mat2 and Mat3 is first computed then the answer in it would be dot

producted to Mat1. In the other matrix, the dot product of Mat1 and Mat2 is first computed then the answer in it would be dot producted to Mat3. Lastly, the values of the first matrix and the last matrix would be compared. This would prove that the matrices are equal.

```
(Mat2 + Mat3) @ Mat1
```

```
array([[ 66,  81,  48],
       [ 98, 116,  66],
       [ 76,  91,  51]])
```

```
Mat2 @ Mat1 + Mat3 @ Mat1
```

```
array([[ 66,  81,  48],
       [ 98, 116,  66],
       [ 76,  91,  51]])
```

```
(Mat2 + Mat3) @ Mat1 == Mat2 @ Mat1 + Mat3 @ Mat1
```

```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

Figure 9 Codes and output for the third property

Figure 9 shows the codes for the distributive property with the outputs. The Mat1 is distributed to the sum of Mat2 and Mat3 by dot product. The second part is the dot product of Mat2 and Mat1 and the dot product of Mat3 and Mat1 is summed. Then, the first and second part is compared and proved to be equal.

```
MatI = np.eye(3)
MatI
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
Mat1 @ MatI
```

```
array([[2., 3., 1.],
       [6., 7., 4.],
       [8., 9., 7.]])
```

```
Mat1 @ MatI == Mat1
```

```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

Figure 10 Codes and output for the fourth property

Figure 9 shows codes of the multiplicative identity property of matrix together with its outputs. An identity matrix is firstly made. Then, the dot product of Mat1 and the identity matrix is computed. Then, it would be compared to Mat1 which would prove to be equal.

```
Mat0 = np.zeros((3,3))
Mat0
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
Mat1 @ Mat0
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
Mat1 @ Mat0 == Mat0
```

```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

Figure 11 Codes and output for the fifth property

Figure 10 shows codes of the multiplicative property of zero of the matrix together with its outputs. Firstly, a matrix made of zeros is made. Then, the dot product of the zero matrix and Mat is computed. After that, the value of the dot product would be compared to the zero matrix which would prove to be equal.

IV. Conclusion

Matrix operations applications in healthcare would help in computing the different tallies that are done in healthcare. A sample of this is when computing the total number of cases of covid 19 in the whole country. With the help of matrix operations, computation would be much easier. This can be done by putting all the cases in each region of Luzon, Visayas and Mindanao. Now you should have three matrices. To find the total cases of Covid 19, simply compute the dot product of all the matrices which are the Luzon, Visayas, and Mindanao.

References

[1] “numpy.array¶,” *numpy.array - NumPy v1.19 Manual*, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.array.html. [Accessed: 21-Dec-2020]

[2] Numpy.org. 2020. *Numpy.Eye — Numpy V1.19 Manual*. [online] Available at: <https://numpy.org/doc/stable/reference/generated/numpy.eye.html> [Accessed 22 December 2020].

[3] Numpy.org. 2020. *Numpy.Zeros — Numpy V1.19 Manual*. [online] Available at: <https://numpy.org/doc/stable/reference/generated/numpy.zeros.html> [Accessed 22 December 2020].