Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 7

Matrix Operations

Submitted by:
Sustento, Myke Alvin E.

Instructor:
Engr. Dylan Josh D. Lopez

December 23, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of matrix operations using the Numpy library. It aims to specifically perform matrix multiplication properties such as the commutative property of multiplication, associative property of multiplication, distributive properties, multiplicative identity property, the multiplicative property of zero, and dimension property.

II. Methods

The deliverables of the activity are to perform matrix multiplication properties such as the commutative property of multiplication, associative property of multiplication, distributive properties, multiplicative identity property, the multiplicative property of zero, and dimension property using the libraries in Numpy such as np.array(), np.eye(), np.zeros(), np.array_equal(), and np.shape(). In matrix multiplication, each entry in the product matrix is the dot product of a row in the first matrix and a column in the second matrix [6]. Np.array() was used to create an array [1] which would also represent the matrices, np.eye() was used to create an array with ones on the diagonal and zeros elsewhere [2] which would become the identity matrix that will be used in the multiplicative identity property, np.zeros() was used to create a new array filled with zeros [3] which would become the zero matrices that will be used in the multiplicative property of zero, np.array_equal() was used to show true if two arrays have the same shape and elements, false otherwise [4] which would be used to prove the matrix multiplication properties, np.shape() was used to get the shape of the matrices [5] which would be used to prove the dimension property.

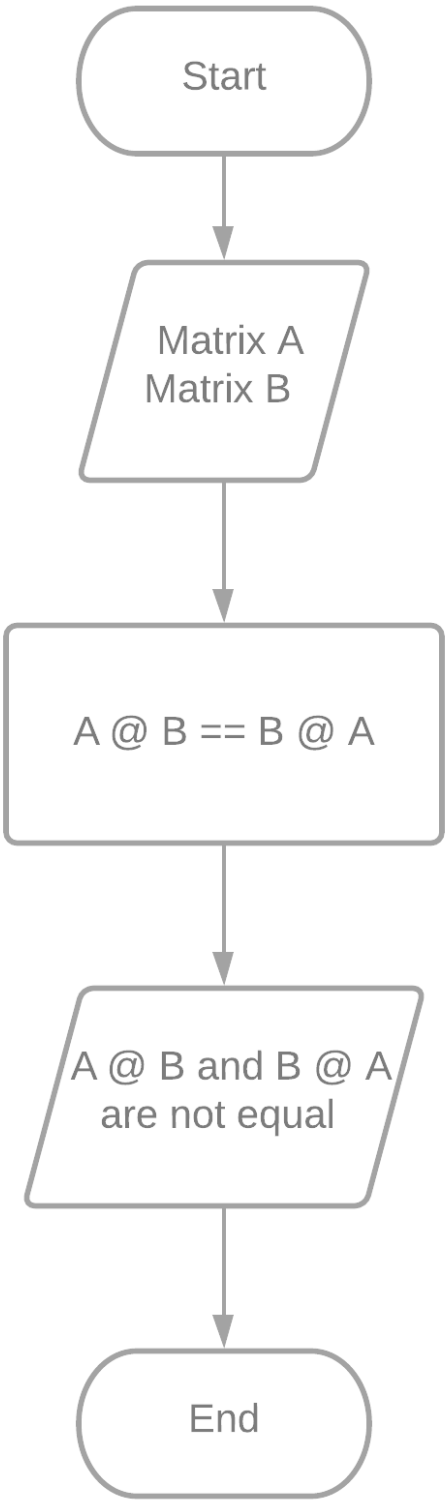

Figure 1 Flowchart for the Commutative Property

Figure 1 shows the flowchart for the commutative property. The flowchart starts by creating matrix A and matrix B. Afterwards, $A \times B == B \times A$ will be processed to check whether it is equal or not. Following the commutative property, the output should and would not be equal. Matrix multiplication is not commutative for the reason that the order in which two matrices are multiplied matters [6].

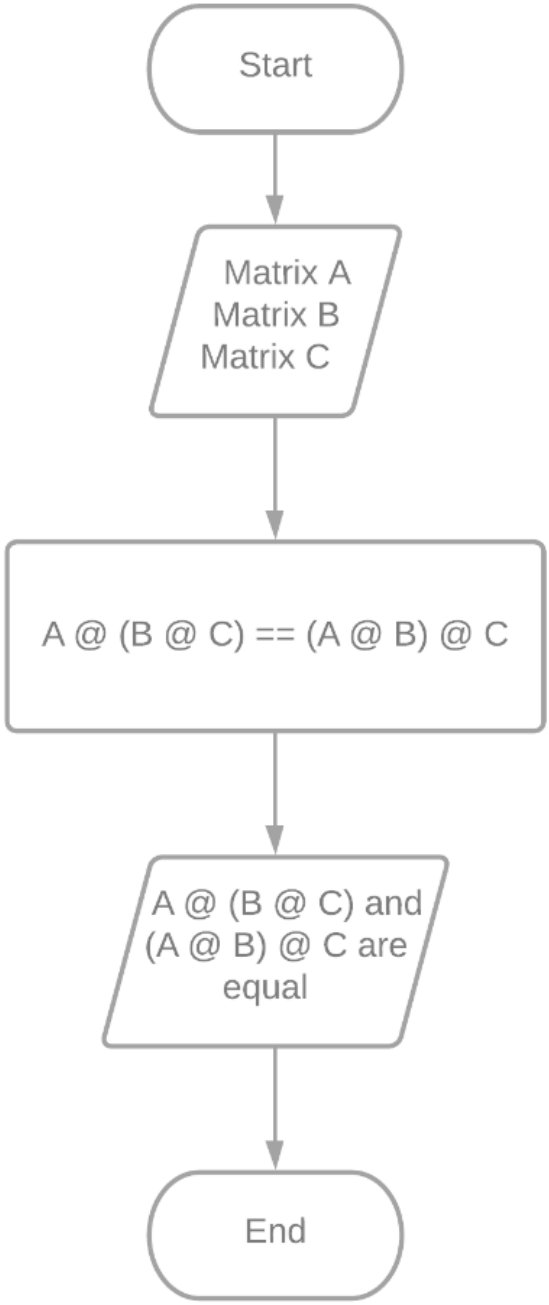

Figure 2 Flowchart for the Associative Property

Figure 2 shows the flowchart for the associative property. The flowchart starts by creating matrices A, B, and C. Afterwards, $A \times (B \times C) == (A \times B) \times C$ will be processed to check whether it is equal or not. Following the associative property, the output should and would be equal. This property states that you can change the grouping surrounding matrix multiplication [6].

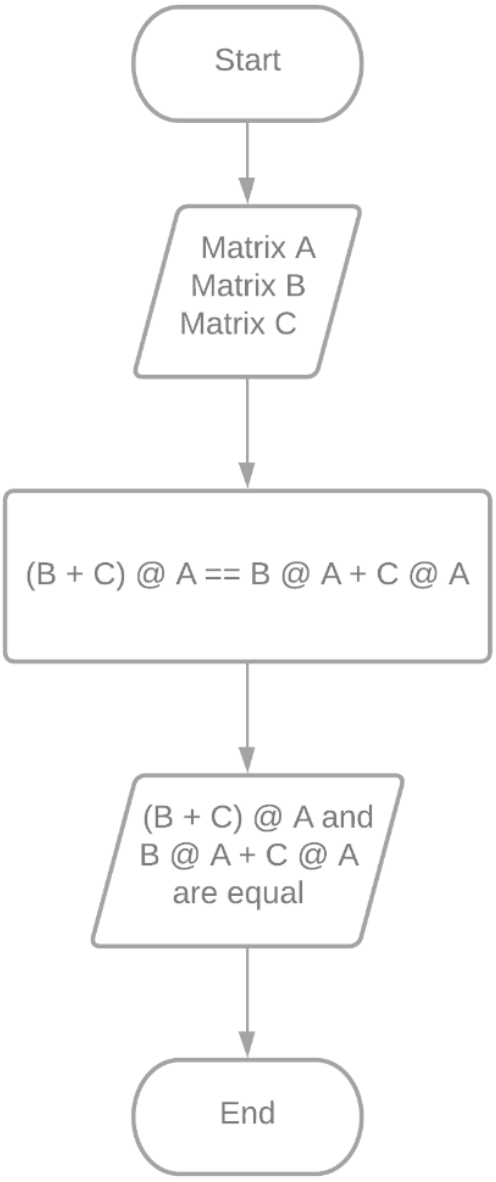

Figure 3 Flowchart for the Distributive Property

Figure 3 shows the flowchart for the distributive property. The flowchart starts by creating matrices A, B, and C. Afterwards, $(B + C) \times A == B \times A + C \times A$ will be processed to check whether it is equal or not. Following the distributive property, the output should and would be equal. It is possible to distribute matrices in much the same way we distribute real numbers [6].

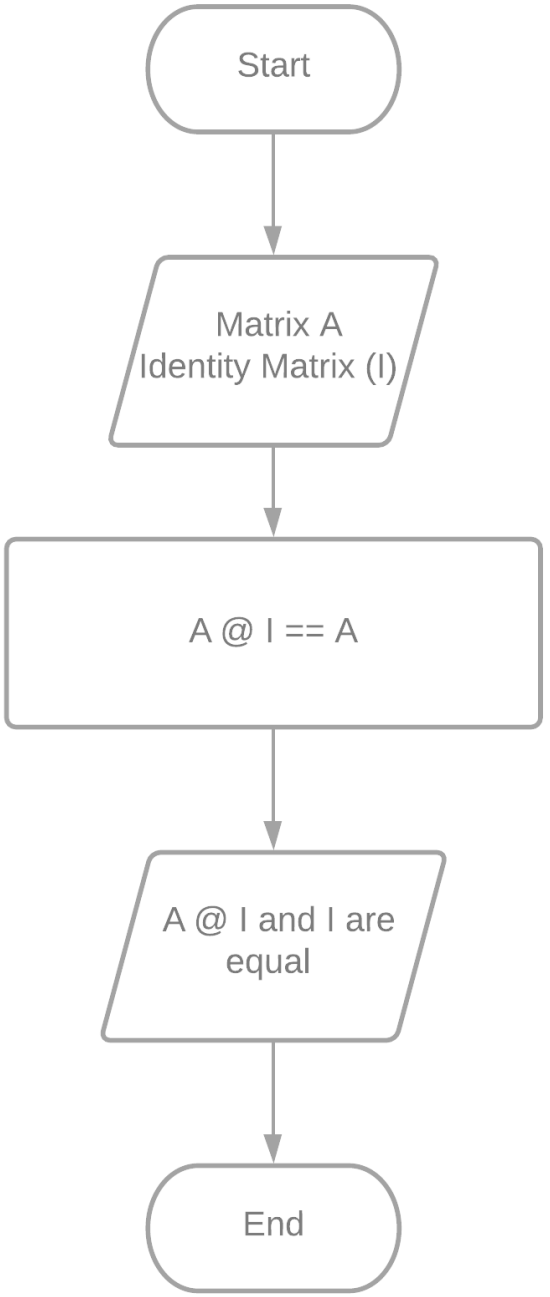

Figure 4 Flowchart for the Multiplicative Identity Property

Figure 4 shows the flowchart for the multiplicative identity property. The flowchart starts by creating matrix A and the identity matrix, also called I. Afterwards, $A \times I == I$ will be processed to check whether it is equal or not. Following the multiplicative identity property, the output should and would be equal. The multiplicative identity property states that, no matter the order in which the multiplication was performed, the product of any $n \times n$ matrix A and I_n is always A [6].

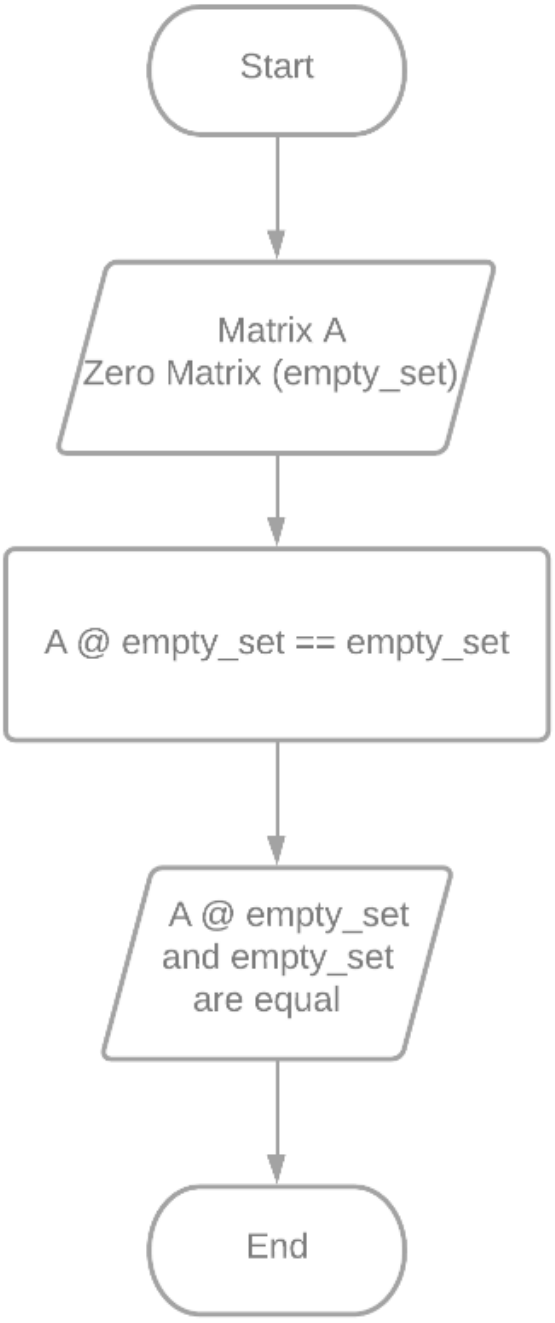

Figure 5 Flowchart for the Multiplicative Property of Zero

Figure 5 shows the flowchart for the multiplicative property of zero. The flowchart starts by creating matrix A and the zero matrix, also called empty_set. Afterward, $A \times empty_{set} == empty_{set}$ will be processed to check whether it is equal or not. Following the multiplicative property of zero, the output should and would be equal. The multiplicative property of zero states that the product of any $n \times n$ matrix and the $n \times n$ zero matrix is the $n \times n$ zero matrix [6].

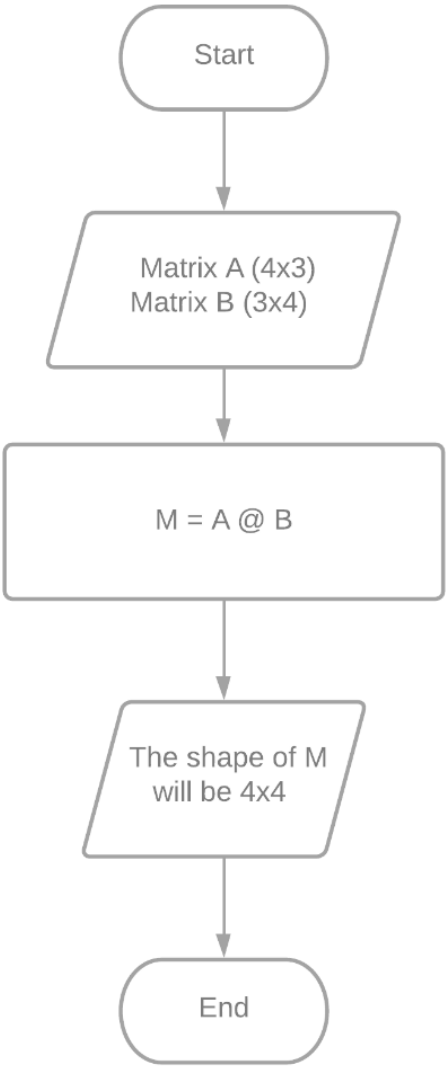

Figure 6 Flowchart for the Dimension Property

Figure 6 shows the flowchart for the dimension property. The flowchart starts by creating matrix A, matrix B, and the zero matrix which is also called empty_set. Afterward, $A \ @ \ empty_{set} == empty_{set}$ will be processed to check whether it is equal or not. Following the multiplicative property of zero, the output should and would be equal. Dimension property states that the product of two matrices will be defined if the number of columns in the first matrix is equal to the number of rows in the second matrix and if the product is defined, the resulting matrix will have the same number of rows as the first matrix and the same number of columns as the second matrix [6].

III. Results

```
# Matrices
A = np.array([
    [1,3,5],
    [0,3,1],
    [3,1,8],
])
B = np.array([
    [1,3,5],
    [9,7,5],
    [2,4,6]
])
C = np.array([
    [1,2,3],
    [4,3,2],
    [5,6,7],
])
```

Figure 7 Matrices A, B, and C

Figure 7 shows the matrices that will be used for the matrix multiplication properties to prove them.

1. $A \cdot B \neq B \cdot A$

```
# commutative property
# A @ B == B @ A
np.array_equal(A @ B, B @ A)
```

```
False
```

Figure 8 Commutative Property Proven Using Numpy

Figure 8 shows the commutative property that used the matrices provided in figure 7. Using the Numpy function, np.array_equal, the commutative property $A \cdot B \neq \mathrm{B} \cdot \mathrm{A}$ was proven through the output which was false.

2. $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

```
# associative property
# A @ (B @ C) == (A @ B) @ C
np.array_equal(A @ (B @ C), (A @ B) @ C)
```

```
True
```

Figure 9 Associative Property Proven Using Numpy

Figure 9 shows the commutative property that used the matrices provided in figure 7. Using the Numpy function, np.array_equal, the associative property $A \cdot (B \cdot \mathrm{C}) = (\mathrm{A} \cdot \mathrm{B}) \cdot \mathrm{C}$ was proven through the output which was true.

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

```
# distributive property
# (B + C) @ A == B @ A + C @ A
np.array_equal((B + C) @ A, B @ A + C @ A)
```

```
True
```

Figure 10 Distributive Property Proven Using Numpy

Figure 10 shows the distributive property that used the matrices provided in figure 7. Using the Numpy function, np.array_equal, distributive property $(B + C) \cdot A = B \cdot \mathrm{A} + \mathrm{C} \cdot \mathrm{A}$ was proven through the output which was true.

5. $A \cdot I = A$

```
# multiplicative identity property
I = np.eye(3)
# A @ I == A
np.array_equal(A @ I, A)
```

```
True
```

Figure 11 Multiplicative Identity Property Proven Using Numpy

Figure 11 shows the multiplicative identity property that used the matrix A provided in figure 7 and an identity matrix created in figure 10. Using the Numpy function, np.array_equal, the multiplicative identity property $A \cdot \mathrm{I} = A$ was proven through the output which was true.

6. $A \cdot \varnothing = \varnothing$

```
# multiplicative property of zero
empty_set = np.zeros((3,3))
# A @ empty_set == empty_set
np.array_equal(A @ empty_set, empty_set)
```

```
True
```

Figure 12 Multiplicative Property of Zero Proven Using Numpy

Figure 12 shows the multiplicative property of zero that used the matrix A provided in figure 7 and a zero matrix created in figure 12. Using the Numpy function, np.array_equal, the multiplicative property of zero $A \cdot \mathrm{empty}_{\mathrm{set}}(\varnothing) = \mathrm{empty}_{\mathrm{set}}(\varnothing)$ was proven through the output which was true.

7. $(m \times n) \cdot (n \times k) = (m \times k)$

```
# dimension property
A = np.array([
    [1,3,5],
    [0,3,1],
    [3,1,8],
    [3,2,4]
])
B = np.array([
    [1,3,5,7],
    [9,7,5,3],
    [4,7,3,2]
])

# A 4x3 matrix
# B 3x4 matrix
# A @ B will be a 4x4 matrix
# product AB is defined

M = A @ B
print("A:\n",A)
print("B:\n",B)
print("Shape of A: ",A.shape)
print("Shape of B: ",B.shape)
print("A @ B:\n",M)
print("Shape of A @ B: ",M.shape)
```

```
A:
 [[1 3 5]
 [0 3 1]
 [3 1 8]
 [3 2 4]]
B:
 [[1 3 5 7]
 [9 7 5 3]
 [4 7 3 2]]
Shape of A:  (4, 3)
Shape of B:  (3, 4)
A @ B:
 [[48 59 35 26]
 [31 28 18 11]
 [44 72 44 40]
 [37 51 37 35]]
Shape of A @ B:  (4, 4)
```

Figure 13 Dimension Property Proven Using Numpy

Figure 13 shows the dimension property that used the matrix A and matrix B provided in figure 7. The dimension property $(m \times n) \cdot (n \times k) = (m \times k)$ is proven true by displaying two matrices and showing their shapes using the np.ndarray.shape() function. It could be seen that the shape of matrix A is (4,3) while the shape of matrix B is (3,4). By following the dimension property, the result should be (4,4) which is proven true by the "shape of A@B: (4,4)" in the output.

IV. Conclusion

This laboratory activity aims to implement the principles and techniques of matrix operations: commutative property of multiplication, associative property of multiplication, distributive properties, multiplicative identity property, the multiplicative property of zero, and dimension property using the Numpy library. The laboratory has shown that NumPy can prove and implement the properties of matrix operations using specific functions such as np.array_equal() and np.ndarray.shape().

Matrix operations can solve problems in healthcare by using the System of Objectified Judgement Analysis (SOJA) matrix model and InforMatrix which are used for assessing medicines within a certain pharmaceutical class [7]. The SOJA matrix model defines several selection criteria for a group of medicines and scores the degree to which each medicine achieves the requirements for each criterion [7]. The InforMatrix model is a device that allows the users of the program to determine an order of merit for the various medicines accessible in a specific category on basis of agreed criteria [7].

References

[1]"numpy.array — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.array.html. [Accessed: 16- Dec- 2020].

[2]"numpy.eye — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.eye.html. [Accessed: 16- Dec- 2020].

[3]"numpy.zeros — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.zeros.html. [Accessed: 16- Dec- 2020].

[4]"numpy.array_equal — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.array_equal.html. [Accessed: 16- Dec- 2020].

[5]"numpy.ndarray.shape — NumPy v1.19 Manual", Numpy.org, 2020. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.ndarray.shape.html. [Accessed: 16- Dec- 2020].

[6]"Properties of matrix multiplication (article) | Khan Academy", Khan Academy, 2020. [Online]. Available: https://www.khanacademy.org/math/precalculus/x9e81a4f98389efdf:matrices/x9e81a4f98389efdf:properties-of-matrix-multiplication/a/properties-of-matrix-multiplication. [Accessed: 16- Dec- 2020].

[7]"How matrix models can support generic medicine prescribing - GaBI Journal", Gabi-journal.net, 2020. [Online]. Available: http://gabi-journal.net/how-matrix-models-can-support-generic-medicine-prescribing.html. [Accessed: 16- Dec- 2020].