Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 7

Matrix Operations

Submitted by:

Parco, Jovian Asher G.

Instructor:

Engr. Dylan Josh D. Lopez

November 21, 2020

I. Objectives

This laboratory activity aims to implement matrix operations and to solve intermediate engineering problems using matrix operation & matrix algebra using Python's Numpy module/

II. Methods

In this laboratory activity, the practices consist of creating a sample of different matrix multiplication properties using NumPy's matrix operation functions such as the dot() function, which is used to a dot product problem; another sample is linalg.det() function that solves determinant which is similar to scalar value but it is derived from a square matrix.

III. Results

The complete repository for the activity is available at Github (). The lab activity consists of one part but six different subparts, where the researcher must show six matrix multiplication properties with the proofs of it.

```
A = np.array([
              [1,2,3],
              [4,5,6],
              [7,8,9]
])
B = np.array([
              [1,2,3,4],
              [5,6,7,8],
              [9,10,11,12]
])
print(np.shape(A))
print(np.shape(B))
print("Dot product of (3,3)(3,4)")
print(A @ B)
print("Error If inner shape is not same like (3,3)(4,3)")
print(B @ A)
```

Figure 1 Code for Commutative property

For the first matrix properties, the researcher demonstrates the commutative property where $AB \neq BA$ because matrix multiplication is not commutative where the two matrices can only be solved in proper order, for the codes (Figure 1), the researcher set up two-variable A & B where A is an array that has a shape of (3,3). In contrast, B has a shape of (3,4), using the Numpy dot() function A & B run to while B & A also run to the function, but the researcher predicts that it will cause an error because the inner variable is not the same.

```
(3, 3)
(3, 4)
Dot product of (3,3)(3,4)
[[ 38  44  50  56]
 [ 83  98 113 128]
 [128 152 176 200]]
Error If inner shape is not same like (3,3)(4,3)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-1db23d5c6954> in <module>
     14 print(A @ B)
     15 print("Error If inner shape is not same like (3,3)(4,3)")
---> 16 print(B @ A)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is different from 4)
```

Figure 2 Output for Commutative property code

For the Output (Figure 2), The dot product of A & B shows the output while the dot product of B & A causes an error because the array shape's inner values are not the same as shown in the error description (Figure 2).

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

B = np.array([
              [10,11,12],
              [13,14,15],
              [16,17,18]
])

C = np.array([
              [19,20,21],
              [22,23,24],
              [25,26,27]
])
A @ (B @ C) == (A @ B) @ C
```

Figure 3 Code for Associative property

The next matrix property is the associative property of multiplication $(AB)C = A(BC)$ where you can use the dot product regardless of how they are grouped; to prove the associative matrix property, the researcher coded (Figure 3) used three "A, B, C" arrays where the array uses the Numpy dot() function to solve them together but B & C is a group and compared to the group of A & B.

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

Figure 4 Output for Associative property code

For the output for the associative property (Figure 4), because the researcher used operator "==" that compares the two arrays if the values are the same, it will output to "True" while "False" if the value is not the same, which proves the associative property.

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

B = np.array([
              [10,11,12],
              [13,14,15],
              [16,17,18]
])

C = np.array([
              [19,20,21],
              [22,23,24],
              [25,26,27]
])
A @ (B + C) == A @ B + A @ C
```

Figure 5 Code for Distributive properties

For the third matrix multiplication property a distributive property, where the equation $A(B + C) = AB + AC$ like algebra where A is distributed to B & C, for the python implementation, the researcher used the same array from the associative property (Figure 5) where it was implemented by using Numpy dot() function A from the addition of B & C and compared to the same distributive property.

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

Figure 6 Output for Distributive property code

The result (Figure 6)is the same as the code for the associative property, where it uses the operator to compare the matrix where it proves the matrix property.

```
A = np.array([
              [1,2,6,],
              [5,4,3],
              [9,10,11]
])

ident = (A @ np.linalg.inv(A))
ident @ A
```

```
array([[ 1.,  2.,  6.],
       [ 5.,  4.,  3.],
       [ 9., 10., 11.]])
```

Figure 7 Code & Output for Multiplicative identity property

For the next matrix multiplication property, a multiplicative identity property where $A \cdot I = A$ where the matrix is multiplied to one it will out the same matrix that has been inputted, the researcher implement it to python (Figure 7) using one array and using Numpy dot() function and the Numpy linalg.inv() function to create the identity and use it the again with the dot product of the matrix. The result is the array, which is the same as the inputted array that proves multiplicative identity property is real.

```
A = np.array([
              [1,2,6,],
              [5,4,3],
              [9,10,11]
])
null = np.zeros(shape=(3,3))
A @ null == null
```

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

Figure 8 Code & Output for Multiplicative property of zero

For the next matrix property, the Multiplicative property of zero is the same as the multiplicative identity property $A \cdot 0 = 0$, but this time it is multiplied to zero and is equal to zero; for the python implementation (Figure 8), the researcher used Numpy zeros with 3,3 function where it outputs 3x3 zeros and uses the dot product to compare to the function, the result is an array where the values are zeros that prove the multiplicative property of zero.

```
A = np.array([
              [1,2,3]
])

B = np.array([
              [10,11,12],
              [13,14,15],
              [16,17,18]
])

print(np.shape(A))
print(np.shape(B))
print(np.shape(A@B))
```

```
(1, 3)
(3, 3)
(1, 3)
```

Figure 9 Code & Output for Dimension property

For the final matrix property, the dimension property where $(m \times n)(n \times \mathrm{k}) = (m \times k)$ where the inner values are equal while the second matrix is the row; for the python implementation, the researcher created two matrices with the shape of 1,3 and the other one is 3,3 and uses the dot product with the combination of shape function to output 1,3 that proves the dimension property.

IV. Conclusion

The researcher learned different types of multiplication matrix property and concluded that the operation is similar to algebra except for the dimension property and uses different techniques to prove the matrix property. The researcher created a scenario to implement matrix operation in real life where,

Matrix operation can help health care facilities by seeing the trend of diseases in their for example:

100 patients in age 1-18 : 30 Heart Problems, 0 Stds, 0 cancer

100 patients in age 18-45 : 50 Heart Problems, 50 Stds, 16 cancer

100 patients in age 45-100 : 80 Heart Problems, 50 Stds, 32 cancer

All 300 patients have 160 Heart Problems, 100 Stds, 100 Stds, and 48 cancer.

Using Matrix operation, adding all the patient's disease can be simplified and optimized; also if there is an outbreak of disease that can infect two people a day with different total infected people by area, we can use the dot product method to easily approximate how many people will be infected in that day.

References

[1] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[2] S. van der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy Array: A Structure for Efficient Numerical Computation,” *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011, doi: 10.1109/MCSE.2011.37.