

LAB 2

Name: Heja Bibani
Student Number: 16301173

TASK 2

Use three matrices A, B, and C as an example, show that matrix multiplication satisfies distributivity on the right: $(A+B)C = AC + BC$. This should be verified by both manual calculation and Python code. The three matrices you pick should be different from the ones used in the Jupyter NB accompanying our lecture.

2.1 Use Markdown cells and Latex to include the manual calculation for the above.

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, B = \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}, C = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$$
$$A(B+C) = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \left(\begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \right) = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 4 & 2 \\ 3 & 8 \end{bmatrix}$$
$$= \begin{bmatrix} 2 \times 4 + 3 \times 3 & 2 \times 2 + 3 \times 8 \\ 4 \times 4 + 5 \times 3 & 4 \times 2 + 5 \times 8 \end{bmatrix} = \begin{bmatrix} 17 & 28 \\ 31 & 48 \end{bmatrix}$$

is equivalent to

$$AB + AC = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 2 \times 3 + 3 \times 1 & 2 \times 0 + 3 \times 5 \\ 4 \times 3 + 5 \times 1 & 4 \times 0 + 5 \times 5 \end{bmatrix} + \begin{bmatrix} 2 \times 1 + 3 \times 2 & 2 \times 2 + 3 \times 3 \\ 4 \times 1 + 5 \times 2 & 4 \times 2 + 5 \times 3 \end{bmatrix}$$
$$= \begin{bmatrix} 9 & 15 \\ 17 & 25 \end{bmatrix} + \begin{bmatrix} 8 & 13 \\ 14 & 23 \end{bmatrix} = \begin{bmatrix} 17 & 28 \\ 31 & 48 \end{bmatrix}$$

As we can see from the above two calculations are equivalent and thus it meets the distributive law. Thus $A(B+C) = AB+AC$.

2.2 Use Markdown cells and Code cells to include the Python code for the above.

In [73]:

```
import numpy as np
```

```
A = np.array([[2, 3], [4, 5]])
A
B = np.array([[3, 0], [1, 5]])
B
C = np.array([[1, 2], [2, 3]])
C
```

Out[73]:

```
array([[1, 2],
       [2, 3]])
```

The calculation for $A(B+C)$:

In [74]:

```
D1 = A.dot(B+C)
D1
```

Out[74]:

```
array([[17, 28],
       [31, 48]])
```

is equivalent to $AB+AC$:

In [75]:

```
D2 = A.dot(B) + A.dot(C)
D2
```

Out[75]:

```
array([[17, 28],
       [31, 48]])
```

We notice that the calculation are identical and produce the same results.

TASK 3

Use three matrices A, B, and C as an example, show that matrix multiplication satisfies associativity: $(AB)C = A(BC)$. This should be verified by both manual calculation and Python code. The three matrices you pick should be different from the ones used in the Jupyter NB accompanying our lecture.

3.1 Use Markdown cells and Latex to include the manual calculation for the above.

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, B = \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}, C = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$$
$$A(BC) = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \left(\begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \right) = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \left(\begin{bmatrix} 3 \times 1 + 0 \times 2 & 3 \times 2 + 0 \times 3 \\ 1 \times 1 + 5 \times 2 & 1 \times 2 + 5 \times 3 \end{bmatrix} \right)$$
$$= \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 3 & 6 \\ 11 & 17 \end{bmatrix} = \begin{bmatrix} 2 \times 3 + 3 \times 11 & 2 \times 6 + 3 \times 17 \\ 4 \times 3 + 5 \times 11 & 4 \times 6 + 5 \times 17 \end{bmatrix}$$
$$= \begin{bmatrix} 39 & 63 \\ 67 & 109 \end{bmatrix}$$

$$(AB)C = \left(\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix} \right) \times \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} = \left(\begin{bmatrix} 2 \times 3 + 3 \times 1 & 2 \times 0 + 3 \times 5 \\ 4 \times 3 + 5 \times 1 & 4 \times 0 + 5 \times 5 \end{bmatrix} \right) \times \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$$
$$= \begin{bmatrix} 9 & 15 \\ 17 & 25 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 9 \times 1 + 15 \times 2 & 9 \times 2 + 15 \times 3 \\ 17 \times 1 + 25 \times 2 & 17 \times 2 + 25 \times 3 \end{bmatrix}$$
$$= \begin{bmatrix} 39 & 63 \\ 67 & 109 \end{bmatrix}$$

As we can see from the above two calculations are equivalent and thus it meets the associative law. Thus $A(BC) = (AB)C$.

3.2 Use Markdown cells and Code cells to include the Python code for the above.

The calculation for $A(BC)$:

In [76]:

```
E1 = A.dot(B.dot(C))
E1
```

Out[76]:

```
array([[ 39,  63],
       [ 67, 109]])
```

We see in the previous that the calculation matches our answer in task 3.1

The calculation for $(AB)C$:

In [77]:

```
E2 = (A.dot(B)).dot(C)
E2
```

Out[77]:

```
array([[ 39,  63],
       [ 67, 109]])
```

We see in the previous that the calculation matches our answer in task 3.1 And also matches the answer in the previous calculation. Thus $A(BC) = (AB)C$.

TASK 4

Use two matrices A and B as an example, show that matrix multiplication may not satisfy commutativity: $AB \neq BA$. This should be shown by both manual calculation and Python code. The three matrices you pick should be different from the ones used in the Jupyter NB accompanying our lecture.

4.1 Use Markdown cells and Latex to include the manual calculation for the above.

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, B = \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}$$
$$AB = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix} = \begin{bmatrix} 2 \times 3 + 3 \times 1 & 2 \times 0 + 3 \times 5 \\ 4 \times 3 + 5 \times 1 & 4 \times 0 + 5 \times 5 \end{bmatrix}$$
$$= \begin{bmatrix} 9 & 15 \\ 17 & 25 \end{bmatrix}$$
$$BA = \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix} \times \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 3 \times 2 + 0 \times 4 & 3 \times 3 + 0 \times 5 \\ 1 \times 2 + 5 \times 4 & 1 \times 3 + 5 \times 5 \end{bmatrix}$$
$$= \begin{bmatrix} 6 & 9 \\ 22 & 28 \end{bmatrix}$$

Therefore:

$$AB \neq BA$$

As we can see from the above two calculations are not equivalent and thus it does not meet the commutative law. Thus $AB \neq AB$.

4.2 Use Markdown cells and Code cells to include the Python code for the above.

The calculation for AB :

In [78]:

```
F1 = A.dot(B)
F1
```

Out[78]:

```
array([[ 9, 15],
       [17, 25]])
```

We notice that this value equals the calculation in section 4.1

The calculation for BA :

In [79]:

```
F2 = B.dot(A)
F2
```

Out[79]:

```
array([[ 6,  9],
       [22, 28]])
```

We notice that this equals the calculation in section 4.1. And thus demonstrates that it does not meet the commutative law: $AB \neq AB$.

TASK 5

Use Python code to find the inverse of the matrix $\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$ and print the result.

In [80]:

```
A
```

Out[80]:

```
array([[2, 3],
       [4, 5]])
```

In [81]:

```
# Avoid inaccurate floating point values during inverse calculation
# See https://stackoverflow.com/questions/24537791/numpy-matrix-inversion-rounding-errors
np.set_printoptions(suppress=True)
```

We calculate the inverse in this section.

In [82]:

```
A_inv = np.linalg.inv(A)
A_inv
```

Out[82]:

```
array([[ -2.5,  1.5],
       [  2., -1. ]])
```

We check to see if multiplying the inverse brings the identity element with $A^{-1} \times A$.

In [83]:

```
A_inv.dot(A)
```

Out[83]:

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

We check to see if the identity element can be generated by doing the calculation with $A \times A^{-1}$.

In [84]:

```
A.dot(A_inv)
```

Out[84]:

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

We have noted in the above that we have found the inverse of A and it meets the criteria that if multiplied with the inverse and vice versa, the identity element will be generated.

TASK 6

Create a QuantumCircuit with a 1-qubit input and no bit output. Initiaize it's qubit $|0\rangle$ to $\frac{1}{3}|0\rangle + \frac{2\sqrt{2}}{3}|1\rangle$. Then, apply X-gate to qubit $|0\rangle$.

The X-gate is represented by the following matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

To see the effect a gate has on a qubit, we simply multiply the qubit's statevector by the gate. We can see that the X-gate switches the two amplitudes of the state vector:

$$X \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} b \\ a \end{bmatrix}$$

6.1 Draw the above circuit.

We import the necessary libraries to ensure everything works.

In [85]:

```
import numpy as np
from qiskit import *
from qiskit.providers.aer import StatevectorSimulator
from math import sqrt
# We need this to embed the image produced into the notebook
%matplotlib inline
```

We initialize the quantum circuit to the state $\frac{1}{3}|0\rangle + \frac{2\sqrt{2}}{3}|1\rangle$. Then, apply X-gate to qubit $|0\rangle$. The circuit is then consequently drawn.

In [86]:

```
qc = QuantumCircuit(1)
# Create an initial state
initial_state = [1/3, sqrt(8)/3]
# Initialize qubit 0 with this state
qc.initialize(initial_state, 0)
qc.x(0)
qc.draw()
```

Out[86]: 

6.2 Use a StatevectorSimulator to run the above circuit with one shot only. After the run, obtain the state vector of qubit $|0\rangle$ and print it.

The X-gate is represented by the following matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The initial qubit was in state:

$$\begin{bmatrix} \frac{1}{3} \\ \frac{2\sqrt{2}}{3} \end{bmatrix}$$

Applying the gate:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} \frac{1}{3} \\ \frac{2\sqrt{2}}{3} \end{bmatrix} = \begin{bmatrix} \frac{2\sqrt{2}}{3} \\ \frac{1}{3} \end{bmatrix}$$

This is demonstrated in the output of the statevector below, the qubit will be in the following state after applying the X-gate:

$$\frac{2\sqrt{2}}{3}|0\rangle + \frac{1}{3}|1\rangle$$

In [87]:

```
# Alternatively: simulator = Aer.get_backend('statevector_simulator')
simulator = StatevectorSimulator()

# Print zero instead of tiny floating point values
np.set_printoptions(suppress=True)
job = execute(qc, backend=simulator, shots=1)
v = job.result().get_statevector()
v
```

Statevector([0.94280904+0.j, 0.33333333+0.j],
 dims=(2,))

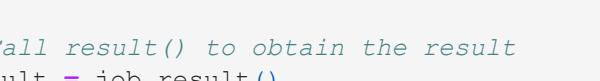
TASK 7

Create a QuantumCircuit with a 1-qubit input and a 1-bit output. Initialize its qubit 0 with the state of $-\frac{1}{3}|0\rangle + \frac{2\sqrt{2}}{3}|1\rangle$. Then, apply X-gate, Z-gate, and H-gate to qubit $|0\rangle$ (NB: Follow this order exactly). Finally measure qubit $|0\rangle$.

7.1 Draw the above circuit.

In [88]:

```
qc2 = QuantumCircuit(1,1)
# Create an initial state
initial_state = [-1/3, sqrt(8)/3]
# Initialize qubit 0 with this state
qc2.initialize(initial_state, 0)
qc2.x(0)
qc2.z(0)
qc2.h(0)
qc2.measure(0,0)
qc2.draw()
```

Out[88]: 

7.2 Use a QasmSimulator to run the above circuit with 2000 shots, and then plot the results with a histogram.

In [93]:

```
from qiskit.providers.aer import QasmSimulator
from qiskit.visualization import plot_histogram
from qiskit.tools import job_monitor

simulator1 = QasmSimulator()
job = execute(qc2, backend=simulator1, shots=2000)

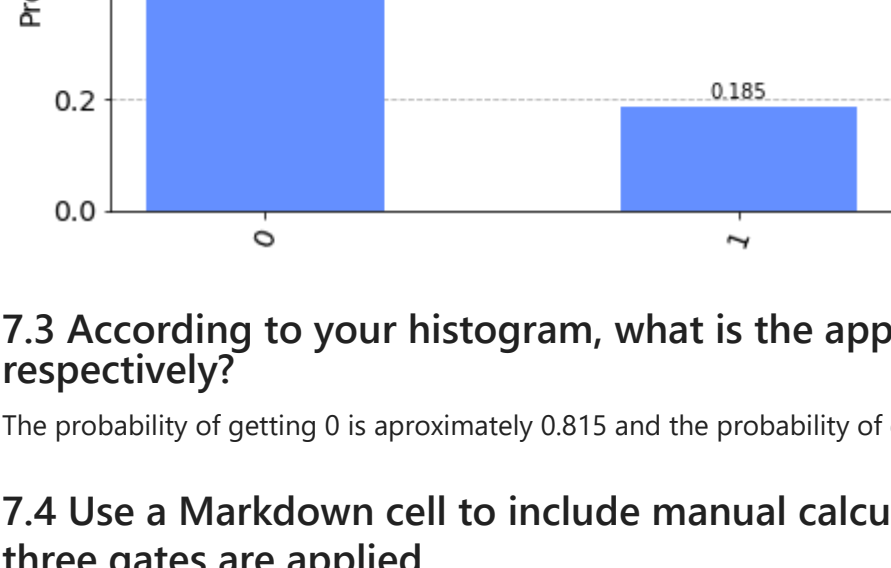
# Call result() to obtain the result
result = job.result()
# Call get_counts() to obtain the counts of different outputs
counts = result.get_counts(qc2)

print(type(counts))

# Print the counts of different outputs
print("\n The counts for different outputs are:", counts)
# Plot the histogram
# Need: from qiskit.visualization import plot_histogram
plot_histogram(counts)
```

<class 'qiskit.result.counts.Counts'>

The counts for different outputs are: {'0': 1629, '1': 371}



7.3 According to your histogram, what is the approximate probabilities of obtaining 0 and 1 respectively?

The probability of getting 0 is approximately 0.815 and the probability of getting 1 is 0.185.

7.4 Use a Markdown cell to include manual calculation on the state of qubit $|0\rangle$ after those three gates are applied.

The X-gate, Z and H gates are represented by the following matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The initial qubit was in state:

$$\begin{bmatrix} -\frac{1}{3} \\ \frac{2\sqrt{2}}{3} \end{bmatrix}$$

The total calculation would be of the equivalent:

$$H \times (Z \times (X \times |0\rangle)) = H(Z(X|0)) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -\frac{1}{3} \\ \frac{2\sqrt{2}}{3} \end{bmatrix} \right) \right)$$

Applying the X-gate:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} -\frac{1}{3} \\ \frac{2\sqrt{2}}{3} \end{bmatrix} = \begin{bmatrix} \frac{2\sqrt{2}}{3} \\ -\frac{1}{3} \end{bmatrix}$$

Applying the Z-gate:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \times \begin{bmatrix} \frac{2\sqrt{2}}{3} \\ -\frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{2\sqrt{2}}{3} \\ \frac{1}{3} \end{bmatrix}$$

Applying the H-Gate

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} \frac{2\sqrt{2}}{3} \\ \frac{1}{3} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{2\sqrt{2}+1}{3} \\ \frac{2\sqrt{2}-1}{3} \end{bmatrix} = \begin{bmatrix} \frac{2\sqrt{2}+1}{3\sqrt{2}} \\ \frac{2\sqrt{2}-1}{3\sqrt{2}} \end{bmatrix}$$

This tells us that the state of qubit is in the following form:

$$\frac{2\sqrt{2}+1}{3\sqrt{2}}|0\rangle + \frac{2\sqrt{2}-1}{3\sqrt{2}}|1\rangle$$

7.5 Verify that the approximate probability distribution obtained in 7.3 conforms with the state of qubit $|0\rangle$ obtained in 7.4. This should be done in a Markdown cell with the calculations based on probability amplitudes.

We know that the state of qubit is in this state:

$$\frac{2\sqrt{2}+1}{3\sqrt{2}}|0\rangle + \frac{2\sqrt{2}-1}{3\sqrt{2}}|1\rangle$$

To get the probability amplitudes we simply do the following:

$$P(|0\rangle) = \left(\frac{2\sqrt{2}+1}{3\sqrt{2}} \right)^2 \approx 0.814269681,$$

$$P(|1\rangle) = \left(\frac{2\sqrt{2}-1}{3\sqrt{2}} \right)^2 \approx 0.185730319$$

This probability calculation conforms to the one stipulated in section 7.2. As these show the probability amplitude for 0 and 1 to be 0.815 and 0.185 respectively.

In []: