QWorld Project

Shivalee RK Shah

December 2021

1 Class QuantumProgram

1.1 Class variables

1. Identity Matrix: starting point for read_unitary()

2. zero vector: starting point for read_state()

1.2 Methods to perform different operations

1.2.1 __init__ method

1. the number of qubits to start with are stored in no_of_qubits

2. the basis state according to number of qubits is stored in basis

3. all the qubits start with $|0\rangle$

1.2.2 basis state

this takes in decimal as parameter
this function returns a string of basis
e.g for 2 qubits it will return

```
00 for decimal 0
01 for decimal 1
10 for decimal 2
11 for decimal 3
```

1.2.3 vector_mat_mul

this function returns a product of vector and matrix

1.2.4 mat_mat_mul

this function returns a product of matrix and matrix

1.2.5   Tensor_vector

this function returns a tensor product of two vectors

1.2.6   Tensor_matrix

this function returns a tensor product of two matrices

1.3   Methods for single qubit gates

1.3.1   h-gate

- Syntax= .h(qubit_index)

- Parameter= qubit index

- this function tensors Identity matrix for all indices which does not match with parameter and tensors h gate matrix with the one that matches

1.3.2   x-gate

- Syntax= .x(qubit_index)

- Parameter= qubit index

- this function tensors Identity matrix for all indices which does not match with parameter and tensors x gate matrix with the one that matches

1.3.3   z-gate

- Syntax= .z(qubit_index)

- Parameter= qubit index

- this function tensors Identity matrix for all indices which does not match with parameter and tensors z gate matrix with the one that matches

1.3.4   rotation-gate

- Syntax= .rotation(angle,qubit_index)

- Parameter= angle in radians and qubit index

- this function tensors Identity matrix for all indices which does not match with parameter and tensors rotation matrix with the one that matches

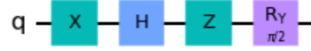- rotation matrix: $\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \cos\alpha & \sin\alpha \end{bmatrix}$

Figure 1: Circuit


```
0.0 1.0
1.0 0.0
unitary matrix




|0>0.0 + |1>1.0000000000000002

state vector


execute
{'1': 1024}
```

(a) IBM simulator

```
0.0 1.0
1.0 0.0
unitary matrix




|0>0.0 + |1>1.0

state vector


execute
{'1': 1024}
```
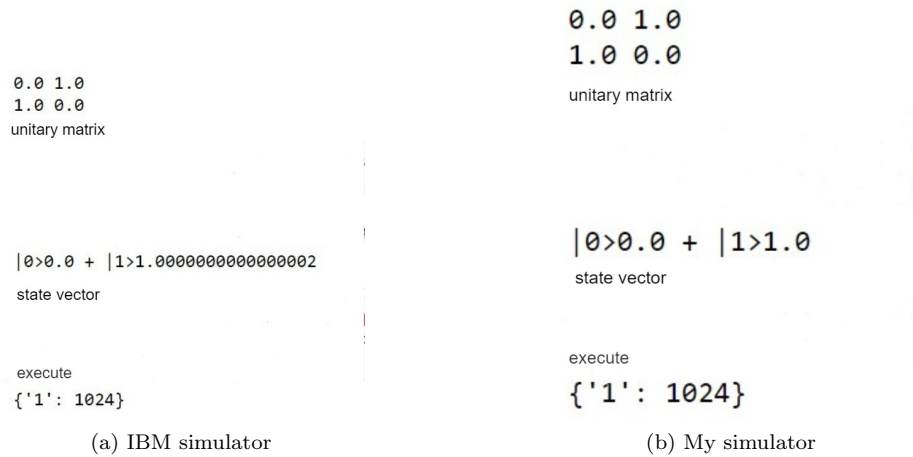
(b) My simulator

Figure 2: Comparison

1.4 Methods for 2 qubit gates

1.4.1 cx-gate

- Syntax= .cx(control, target)

- Parameter = control, target

- we know that in controlled not gate if control is 1 then x-gate is applied on target qubit

- we first create a 0 matrix of size $2^{no\_of\_qubits}$

- then according to working of cnot gate we construct the matrix

$$|00\rangle \rightarrow |00\rangle$$
$$|01\rangle \rightarrow |01\rangle$$
$$|10\rangle \rightarrow |11\rangle$$
$$|11\rangle \rightarrow |10\rangle$$

1.4.2   cz-gate

- Syntax= .cz(control, target)

- Parameter = control, target

- in this we apply H gate then cx gate and then H gate

1.4.3   cr-gate

- Syntax= .cr(angle,control, target)

- Parameter = angle in radians, control, target

- here if control is 1 then rotation gate will be applied to the target bit

- then accordingly we construct the matrix

$$|00\rangle \to |00\rangle$$
$$|01\rangle \to |01\rangle$$
$$|10\rangle \to \cos\alpha|10\rangle + \sin\alpha|11\rangle$$
$$|11\rangle \to -\sin\alpha|10\rangle + \cos\alpha|11\rangle$$

- then multiplies the matrix with previous unitary matrix

Figure 3: Circuit

Unitary matrix from my simulator:
0.707 0.707 0.0 0.0
0.707 -0.707 0.0 0.0
0.0 0.0 0.707 -0.707
0.0 0.0 -0.707 -0.707

Unitary matrix from IBM simulator:
0.707 0.707 0.0 0.0
0.707 -0.707 0.0 0.0
0.0 0.0 0.707 -0.707
-0.0 -0.0 -0.707 -0.707
Statevector simulator from IBM Simulator:
[0.70710678 0.70710678 0.   -0.   ]

Statevector simulator from my Simulator:

[[0.7071067811865474], [0.7071067811865474], [0.0], [0.0]]

Execution from my Simulator:

{'00': 511, '01': 511}

Execution from IBM Simulator:
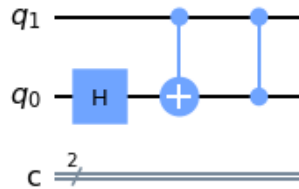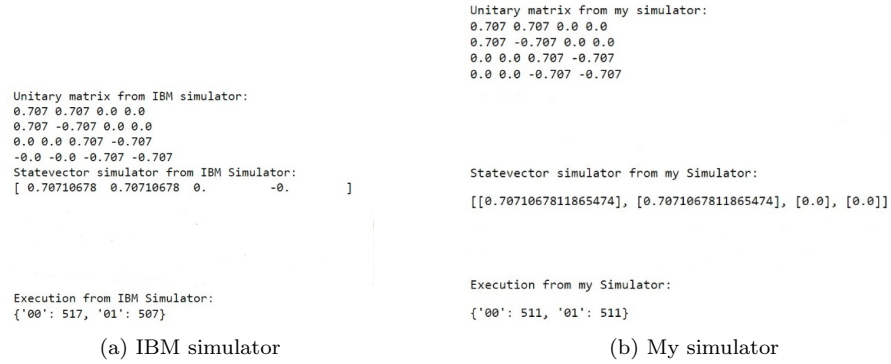{'00': 517, '01': 507}

(a) IBM simulator

(b) My simulator

Figure 4: Comparison

1.5 Method for 3 qubit gate

ccx gate

- Syntax= .ccx(control1, control2, target)

- Parameter = control1, control2, target

- it applies x-gate to target only when both control1 and control2 are 1

- according to its function matrix is made

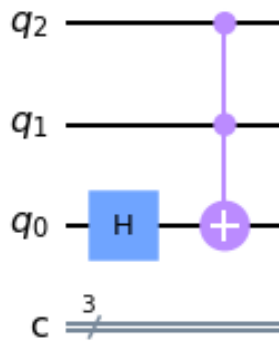- then multiplies the matrix with previous unitary matrix
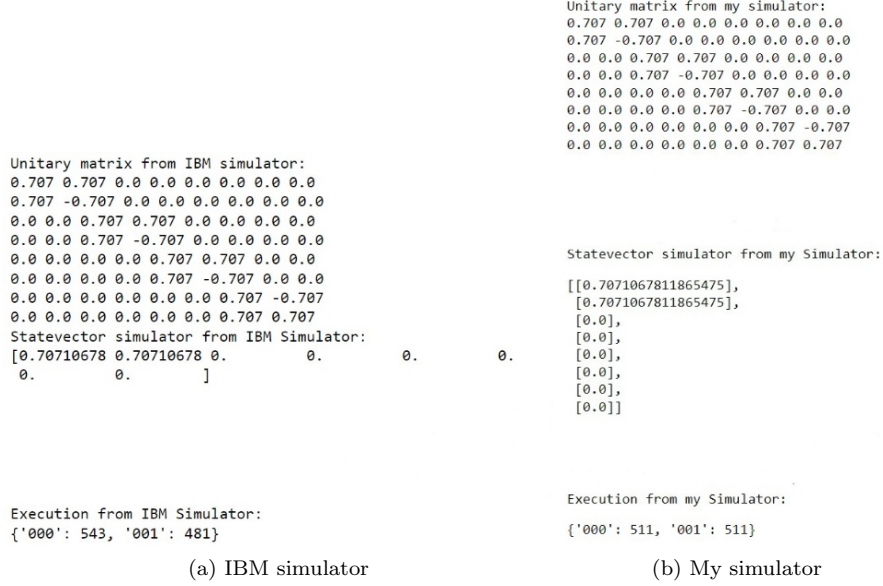

Figure 5: Circuit

```
Unitary matrix from my simulator:
0.707 0.707 0.0 0.0 0.0 0.0 0.0 0.0
0.707 -0.707 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.707 0.707 0.0 0.0 0.0 0.0
0.0 0.0 0.707 -0.707 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.707 0.707 0.0 0.0
0.0 0.0 0.0 0.0 0.707 -0.707 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.707 -0.707
0.0 0.0 0.0 0.0 0.0 0.0 0.707 0.707
```

```
Unitary matrix from IBM simulator:
0.707 0.707 0.0 0.0 0.0 0.0 0.0 0.0
0.707 -0.707 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.707 0.707 0.0 0.0 0.0 0.0
0.0 0.0 0.707 -0.707 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.707 0.707 0.0 0.0
0.0 0.0 0.0 0.0 0.707 -0.707 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.707 -0.707
0.0 0.0 0.0 0.0 0.0 0.0 0.707 0.707
Statevector simulator from IBM Simulator:
[0.70710678 0.70710678 0.     0.     0.     0.
 0.     0.     ]
```

```
Statevector simulator from my Simulator:

[[0.7071067811865475],
 [0.7071067811865475],
 [0.0],
 [0.0],
 [0.0],
 [0.0],
 [0.0],
 [0.0]]
```

```
Execution from IBM Simulator:
{'000': 543, '001': 481}
```

```
Execution from my Simulator:

{'000': 511, '001': 511}
```

(a) IBM simulator   (b) My simulator

Figure 6: Comparison

1.6 Methods for simulating circuit

1.6.1 read_unitary()

Returns a single unitary matrix (quantum operator) equivalent to all defined quantum operators until this point, i.e., the multiplication of all quantum operators in the defined order.
Returns in the form of a matrix

1.6.2 read_state()

Returns the current quantum state of circuit.
Returns in the form of a column vector

1.6.3 observing_probabilities()

Returns the probabilities of observing the basis states if all qubits are measured at this moment.
Returns a histogram showing basis state in x-axis and probabilities in y

1.6.4 execute()

Returns the observed outcomes with their frequencies.
Returns a dictionary

2 Other links

Link to Gitlab repository
Video link to video explanation of project