20211779 Sun Weixiang 孙尉翔

1. $\hat{H}\psi = E\psi$ eigenvalue E

$\hat{P}\psi = p\psi$ eigenvalue p

2. $[\hat{x}, \hat{p}] = \hat{x}\hat{p} - \hat{p}\hat{x}$

where $\begin{cases} \hat{x} = x \\ \hat{p} = -i\hbar \frac{d}{dx} \end{cases}$

therefore $[\hat{x}, \hat{p}] = x \cdot (-i\hbar \frac{d}{dx}) - (-i\hbar \frac{d}{dx}) x$

$= -i\hbar (x \frac{d}{dx} + \frac{d}{dx} \cdot x) = -2i\hbar x \neq 0$

Explain: It's not possible to simultaneously know the precise values of both position and momentum of a particle

3. $\langle\psi| = \cos(\frac{\theta}{2}) |0\rangle + e^{i\phi} \sin(\frac{\theta}{2}) |1\rangle$

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

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

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

In [2]:
```
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(1)

qc.x(0)

backend = Aer.get_backend('statevector_simulator')
result = execute(qc, backend).result()
statevector = result.get_statevector()

print("X|0⟩ = ", statevector)

qc.x(0)

result = execute(qc, backend).result()
statevector = result.get_statevector()

print("X|1⟩ = ", statevector)
```

```
X|0⟩ =  Statevector([0.+0.j, 1.+0.j],
            dims=(2,))
X|1⟩ =  Statevector([1.+0.j, 0.+0.j],
            dims=(2,))
```

5. Superposition: A quantum particle can be in a combination of different state, each with an associated probability amplitude. When measure, the system collapses into one of states with a probability

Example: Consider $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

Where α, β are complex probability amplitudes.

There the probability of state $|0\rangle$ is α^2

state $|1\rangle$ is β^2

Entanglement: When particles become entangled, their states are no longer indenpendent of each other. Measuring one of particles' state can determines others.

$$|\phi\rangle \neq |a\rangle \otimes |b\rangle$$

Example: $|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

if one is measured to $|0\rangle$
another too

if one is measured to $|1\rangle$
another too.

6. $\left(\frac{1}{\sqrt{5}}\right)^2 = \frac{1}{5}$ = probability of $|0\rangle$

7. CNOT gate $= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

In [6]:
```python
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2)

qc.cx(0, 1)

print(qc)

backend = Aer.get_backend('statevector_simulator')
result = execute(qc, backend).result()
statevector = result.get_statevector()

print("Output state: ", statevector)
```

```
q_0: ──■──
     ┌─┴─┐
q_1: ┤ X ├
     └───┘
Output state:  Statevector([1.+0.j, 0.+0.j, 0.+0.j, 0.+0.j],
            dims=(2, 2))
```

8.

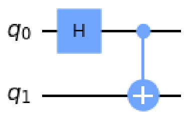
In [5]:
```python
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2)

qc.h(0)
qc.cx(0, 1)

qc.draw('mpl')
```

Out[5]:



Circuit 1

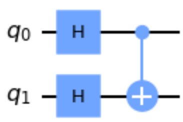
In [6]:
```python
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2)

qc.h(0)
qc.h(1)
qc.cx(0, 1)

qc.draw('mpl')
```

Out[6]:



Circuit 2

```
In [7]: from qiskit import QuantumCircuit, Aer, execute

# Circuit 1
qc1 = QuantumCircuit(2)
qc1.h(0)
qc1.cx(0, 1)

# Circuit 2
qc2 = QuantumCircuit(2)
qc2.h(0)
qc2.h(1)
qc2.cx(0, 1)

backend = Aer.get_backend('statevector_simulator')

result1 = execute(qc1, backend).result()
statevector1 = result1.get_statevector()

result2 = execute(qc2, backend).result()
statevector2 = result2.get_statevector()

print("Circuit 1 output state: ", statevector1)
print("Circuit 2 output state: ", statevector2)
```

```
Circuit 1 output state:  Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
Circuit 2 output state:  Statevector([0.5+0.j, 0.5+0.j, 0.5+0.j, 0.5+0.j],
            dims=(2, 2))
```

$[0.707\cdots, 0, 0, 0.707\cdots] = [\frac{1}{\sqrt{2}}, 0] \otimes [0, \frac{1}{\sqrt{2}}]$

So circ 1 is not entangled state

$[0.5, 0.5, 0.5, 0.5]$ cannot be written as $|\alpha\rangle \otimes |\beta\rangle$

So circ 2 is entangled state.

8.

```
In [11]: from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2)
qc.h(0)
qc.t(0)
qc.cx(0, 1)

backend = Aer.get_backend('unitary_simulator')
result = execute(qc, backend).result()
unitary_matrix = result.get_unitary()

print("Unitary matrix of the circuit:")
print(unitary_matrix)
```

```
Unitary matrix of the circuit:
Operator([[ 0.70710678+0.00000000e+00j,  0.70710678+8.65956056e-17j,
            0.        +0.00000000e+00j,  0.        +0.00000000e+00j],
          [ 0.        +0.00000000e+00j,  0.        +0.00000000e+00j,
            0.5       +5.00000000e-01j, -0.5       -5.00000000e-01j],
          [ 0.        +0.00000000e+00j,  0.        +0.00000000e+00j,
            0.70710678+0.00000000e+00j,  0.70710678+8.65956056e-17j],
          [ 0.5       +5.00000000e-01j, -0.5       -5.00000000e-01j,
            0.        +0.00000000e+00j,  0.        +0.00000000e+00j]],
         input_dims=(2, 2), output_dims=(2, 2))
```

the written ans is in next page.

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

10.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The first-order QFT creates an equal superposition of $|0\rangle$ and $|1\rangle$ and the output will be $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

11. $$C_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$|0\rangle\langle 0| \otimes I = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$|1\rangle\langle 1| \otimes X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Therefore

$$C_X = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| X = \begin{bmatrix} 1 & 0 & & \\ 0 & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix} \quad \square.$$

Then we prove it by Qiskit

In [17]:
```python
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2)
qc.cx(0, 1)
print(qc)

backend = Aer.get_backend('unitary_simulator')
result = execute(qc, backend).result()
unitary_matrix_cx = result.get_unitary()

print("Unitary matrix of C_X:")
print(unitary_matrix_cx)
```

```
q_0: ──■──
     ┌─┴─┐
q_1: ┤ X ├
     └───┘
Unitary matrix of C_X:
Operator([[1.+0.j, 0.+0.j, 0.+0.j, 0.+0.j],
          [0.+0.j, 0.+0.j, 0.+0.j, 1.+0.j],
          [0.+0.j, 0.+0.j, 1.+0.j, 0.+0.j],
          [0.+0.j, 1.+0.j, 0.+0.j, 0.+0.j]],
         input_dims=(2, 2), output_dims=(2, 2))
```

In [19]:
```python
import numpy as np

# Define the individual matrices
zero_proj = np.array([[1, 0],
                      [0, 0]])

id_matrix = np.eye(2)

one_proj = np.array([[0, 0],
                     [0, 1]])

X_gate = np.array([[0, 1],
                   [1, 0]])

tensor_product_1 = np.kron(zero_proj, id_matrix)
tensor_product_2 = np.kron(one_proj, X_gate)
combined_unitary_matrix = tensor_product_1 + tensor_product_2

print("Unitary matrix of |0⟩⟨0|⊗I + |1⟩⟨1|⊗X:")
print(combined_unitary_matrix)
```

```
Unitary matrix of |0⟩⟨0|⊗I + |1⟩⟨1|⊗X:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]]
```

12.

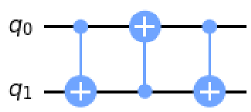
In [23]:
```python
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2)

qc.cx(0, 1)
qc.cx(1, 0)
qc.cx(0, 1)

qc.draw('mpl')
```

Out[23]:



13. $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

$$e^{-it(X+Z)} = I + (-it(X+Z)) + \frac{[-it(X+Z)]^2}{2!} + \frac{[-it(X+Z)]^3}{3!} + \cdots$$

Since $(X+Z)^2 = X^2 + Z^2 + XZ + ZX = 2I$

$(X+Z)^3 = 2I \cdot (X+Z) = 2(X+Z)$

$(X+Z)^4 = 4I$

$\vdots$

$(X+Z)^{2n} = 2n I$

$(X+Z)^{2n+1} = 2n(X+Z)$

and what's more $(-i)^2 = 1$

$(-i)^3 = -i$

$(-i)^4 = -1$

$\vdots$

So $e^{-it(X+Y)} = I - \frac{It^2}{2!} - \frac{it^3(X+Z)}{3!} + \frac{It^4}{4!} + \cdots$

$= \left(I - \frac{It^2}{2!} + \frac{It^4}{4!} + \cdots\right) + \left(it(X+Z) - \frac{it^3(X+Z)}{3!} + \cdots\right)$

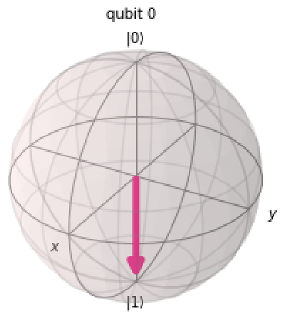
$= \cos(t) I - i\sin(t)(X+Z)$

14.

In [29]:
```python
def lab1_ex1():
    qc = QuantumCircuit(1)
    # FILL YOUR CODE IN HERE
    #
    qc.x(0)
    return qc

state = Statevector.from_instruction(lab1_ex1())
plot_bloch_multivector(state)
```

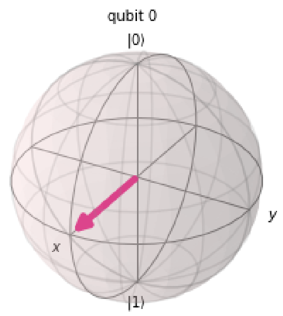
Out[29]:



In [32]:
```python
def lab1_ex2():
    qc = QuantumCircuit(1)
    # FILL YOUR CODE IN HERE
    #
    qc.h(0)
    return qc

state = Statevector.from_instruction(lab1_ex2())
plot_bloch_multivector(state)
```

Out[32]:



In [33]:
```python
from qiskit import QuantumCircuit, Aer, execute

def lab1_ex5():
    qc = QuantumCircuit(2, 2)
    # FILL YOUR CODE IN HERE
    #
    qc.h(0)
    qc.cx(0, 1)
    return qc

qc = lab1_ex5()
qc.draw()
```

Out[33]:

```
     ┌───┐     
q_0: ┤ H ├──■──
     └───┘┌─┴─┐
q_1: ─────┤ X ├
          └───┘
c: 2/══════════
```