浙江大學

《量子计算理论基础与软件系统》实验报告

实验名称	：	Lab2 QFT and QPE Algorithm
姓　　名	：	王晓宇
学　　号	：	3220104364
电子邮箱	：	3220104364@zju.edu.cn
联系电话	：	19550222634
授课教师	：	卢丽强/尹建伟
助　　教	：	储天尧

2024 年 11 月 5 日

Lab2 QFT and QPE Algorithm

Lab2 QFT and QPE Algorithm

1　实验简介

本次实验中，我们使用 `qiskit` 模拟运行量子电路以进一步理解 QFT 和 QPE 算法的原理。

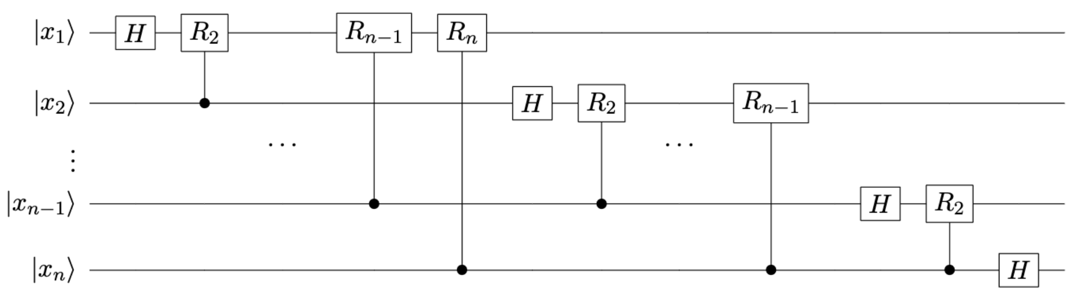
1.1　QFT 算法

量子傅里叶变换（Quantum Fourier Transform，QFT）是量子计算中的一种核心算法，是经典傅里叶变换的量子版本，能够将一个量子态在计算基底上的表示转换为频率基底上的表示。

对于一个 n 量子比特的输入态 $|x\rangle$，QFT 定义为：

$$\text{QFT}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i}{2^n}xk}|k\rangle$$

QFT 的电路实现可以通过一系列受控旋转门和 Hadamard 门实现，如下图所示：



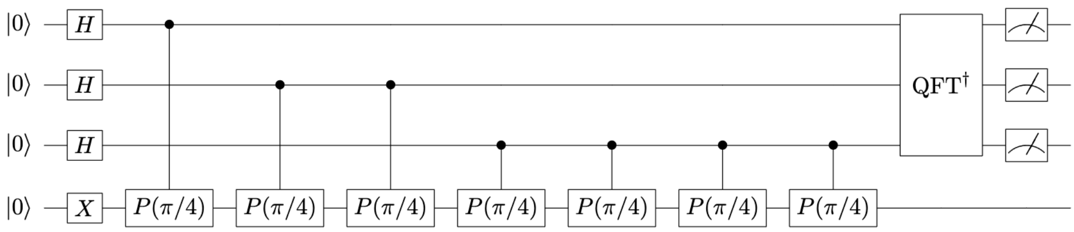
1.2　QPE 算法

量子相位估计（Quantum Phase Estimation，QPE）是量子计算中的一种重要算法，主要用于估计酉矩阵本征态对应本征值的相位，是许多量子算法（如 Shor 算法和量子模拟）的关键部分。

对于酉矩阵 U 的本征态 $|\psi\rangle$ 对应本征值的相位 ϕ，即：

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$$

QPE 算法使用两部分量子寄存器实现相位估计。下图展示了使用三个量子比特作为计数寄存器和一个量子比特作为特征寄存器的 QPE 电路：



其中 $P(\theta)$ 为相位旋转门:

$$P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

因此，该电路用于估计 $P(\pi/4)$ 门的本征态 $|1\rangle$ 对应本征值的相位，即 $\phi = 1/8$ 。

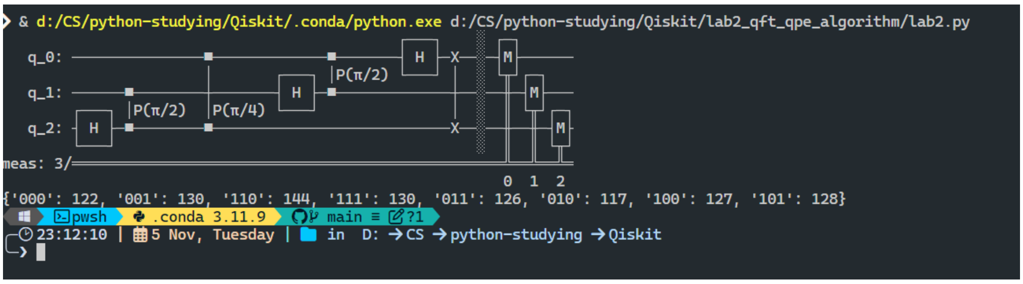
2 实验要求

2.1 QFT 算法

```python
from numpy import pi
from qiskit import QuantumCircuit, transpile
from qiskit.providers.basic_provider import BasicSimulator


def qft(qc:QuantumCircuit) -> QuantumCircuit:
    for i in range(qc.num_qubits - 1, -1, -1):
        qc.h(i)
        for j in range(i - 1, -1, -1):
            qc.cp(pi / 2 ** (i - j), j, i)
    for i in range(qc.num_qubits // 2):
        qc.swap(i, qc.num_qubits - i - 1)
    return qc


# TODO: change the number of qubits
# n_qubits = 3 ## Q1.1
n_qubits = 2 ## Q1.2

qc = QuantumCircuit(n_qubits)

```

```python
# TODO: add quantum gate to set the initial state
qc.x(1)

qc = qft(qc)
qc.measure_all()
print(qc)


backend = BasicSimulator()
tqc = transpile(qc, backend)
result = backend.run(tqc).result()
counts = result.get_counts()
print(counts)
```

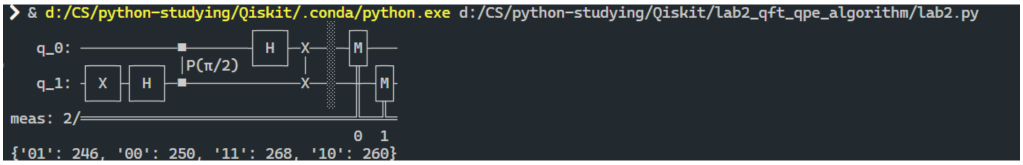
- 以上代码实现了一个三量子比特的 QFT 算法，运行代码查看结果态采样的频率分布。

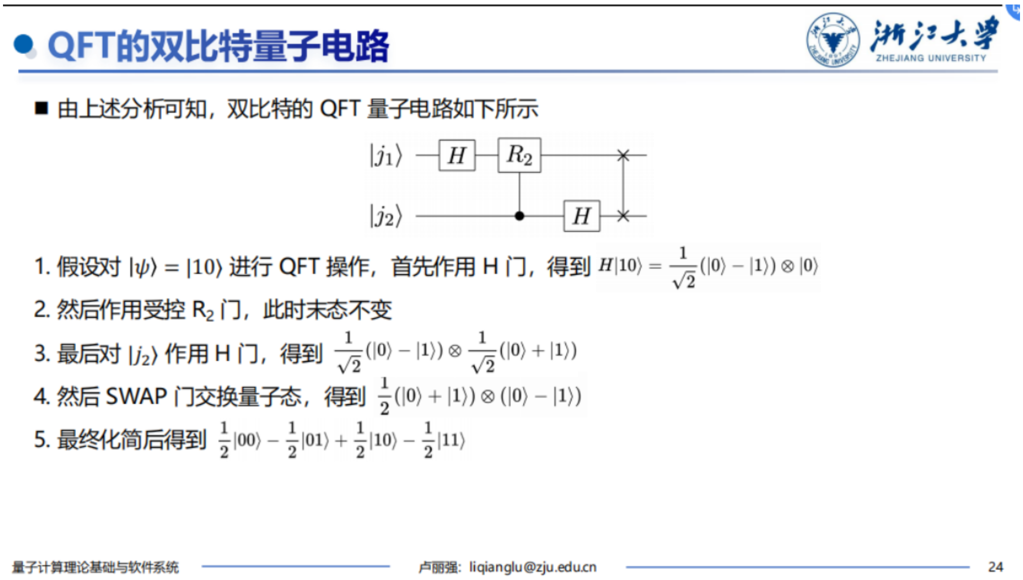


- 将 `n_qubits` 改为 2，初态设置为 $|10\rangle$，获得双量子比特的 QFT 电路，验证输出与理论结果一致。（注意：`qiskit` 中的量子比特按小端排列，即低索引的量子比特对应量子态比特串中的低位，如 $|q_1q_0\rangle$，这与课程 PPT 的表示方式相反）

```python
## Begin
n_qubits = 2
## Middle
qc.x(1)
## End
```

实验结果如下，各状态的概率基本接近，且都接近于$1/4$，这与理论结果预期一致。





> 图源PPT，实验结果和理论推导中各系数占比符合**相同**的分布，均为1/4

- 通过观察 `qft` 函数或者理论推导，计算 QFT 算法量子门层数（在所有量子比特上并行操作的一组量子门视为一层）关于量子比特数的复杂度。

  1. 观察qft函数：

     记`num_qubits`为N

     ```python
     def qft(qc:QuantumCircuit) -> QuantumCircuit:
         for i in range(qc.num_qubits - 1, -1, -1):
             qc.h(i)
             for j in range(i - 1, -1, -1):
                 qc.cp(pi / 2 ** (i - j), j, i)
         for i in range(qc.num_qubits // 2):
             qc.swap(i, qc.num_qubits - i - 1)
         return qc
     ```

     作用Hardmand门数量：N

     - 但是这步可以同步作用到量子比特，时间复杂度可以视为1

     作用P门数量：$\sum_{i=1}^{N-1} i = \frac{N(N-1)}{2}$

     - 这步不可以同步进行，只能按序进行门操作

     作用Swap门数量：$\lfloor \frac{N}{2} \rfloor \approx \frac{N}{2}$

- 同理这步可以同步作用到量子比特，时间复杂度可以视为1

$$\therefore Time = 1 + \frac{N(N-1)}{2} + 1 = O(N^2)$$

2.2 QPE 算法

- 根据 QFT 算法的实现代码，构建实验简介中的 QPE 电路，运行代码查看结果态采样的频率分布。

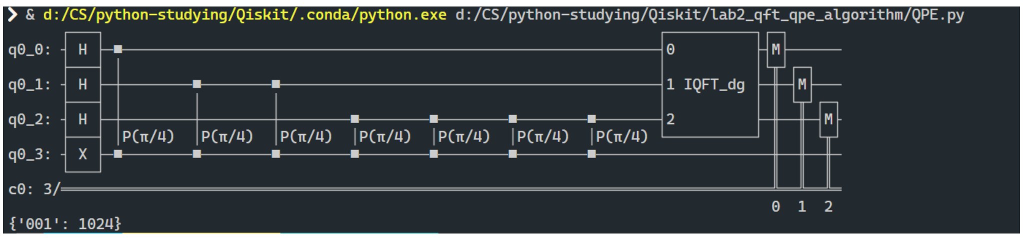
> 其中，可以通过如下代码将一个三量子比特逆 QFT 电路拼接至四量子比特电路中：
>
> ```
> from qiskit.circuit.library import QFT
>
> qc = QuantumCircuit(4)
> qc = qc.compose(QFT(3, inverse=True), [0, 1, 2])
> ```
>
> 此外，量子比特的部分测量需要通过声明量子和经典比特寄存器实现，代码示例如下：
>
> ```
> from qiskit import QuantumRegister, ClassicalRegister
>
> qr = QuantumRegister(4)
> cr = ClassicalRegister(3)
> qc = QuantumCircuit(qr, cr)
> qc.measure([0, 1, 2], [0, 1, 2])
> ```
>
> 这里将索引为 [0, 1, 2] 的量子比特的测量结果对应至索引为 [0, 1, 2] 的经典比特上。

我们修改代码如下

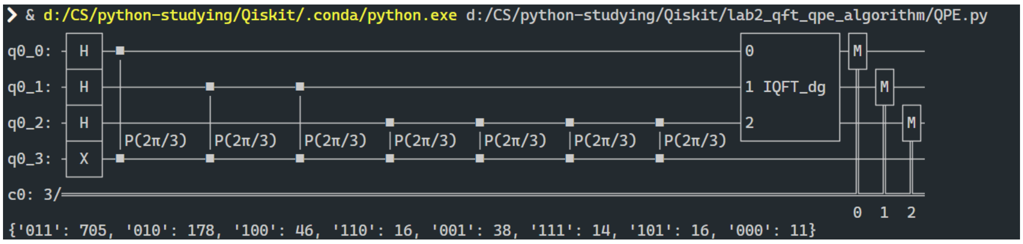
```
from numpy import pi
from qiskit import QuantumCircuit, transpile
from qiskit.providers.basic_provider import BasicSimulator
from qiskit.circuit.library import QFT
from qiskit import QuantumRegister, ClassicalRegister

n_qubits = 4
qr = QuantumRegister(4)
```

```python
cr = ClassicalRegister(3)
qc = QuantumCircuit(qr, cr)

## change the value of theta to test the algorithm
theta = pi / 4
# theta = 2 * pi / 3 ## Q2.23

## initialize the state
qc.initialize([3/5,4/5], 3)
# qc.x(3)
for i in range(0, qc.num_qubits-1, 1):
    qc.h(i)
for i in range(0, qc.num_qubits-1, 1):
    for j in range(0, 2**i, 1):
        qc.cp(theta, i, 3)

qc = qc.compose(QFT(3, inverse=True), [0, 1, 2])

## measure qubits 0, 1, 2 to classical bits 0, 1, 2
qc.measure([0, 1, 2], [0, 1, 2])
print(qc)
backend = BasicSimulator()
tqc = transpile(qc, backend)
result = backend.run(tqc).result()
counts = result.get_counts()
print(counts)
```

运行代码如下，可以看到数据为`'001' = 1024`，代表分数$\frac{1}{8}$，精确估计出了相位，实验预期与理论结果一致。



- 更改 $P(\theta)$ 门中的角度参数为 $2\pi/3$，验证输出与理论结果一致。



首先要注意到**理论结果**$\varphi = 1/3 \approx 0.333$，不能由二进制精确表示。

- 共计运行电路**1000**次，测量得到某个值的概率可用于衡量与真实值的接近程度。所以在实验结果中`011`以最大概率得到，此时测量测到的相位即`011`的小数表示--$\frac{3}{8} = 0.375$，是最接近$1/3$的相位估计值了。
- QPE精度取决于qubit数目：n个qubit可以估计相位到$\frac{1}{2^n}$的精度。

- 基于$P(\theta)$门计算QPE 算法量子门层数关于相位估计精度（即计数寄存器中的量子比特数）的复杂度。

  只是基于$P(\theta)$门来计算复杂度的话，我们可以看到计数寄存器中第i个量子比特，将会对特征寄存器量子比特做2^i次$P(\theta)$，且不可以并行门操作，即：

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1 = O(2^n)$$

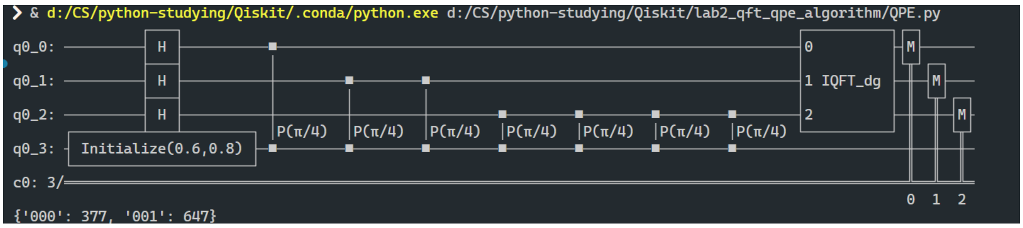
  逆量子傅里叶变换（IQFT）的时间复杂度是 $O(n^2)$，所以我们即使考虑了IQFT的存在，我们的量子门层数关于相位估计精度复杂度仍为$O(2^n)$

- （选做）更改特征寄存器的初态为非本征态 $|\varphi\rangle = [3/5, 4/5]^\top$，验证输出与理论结果一致。

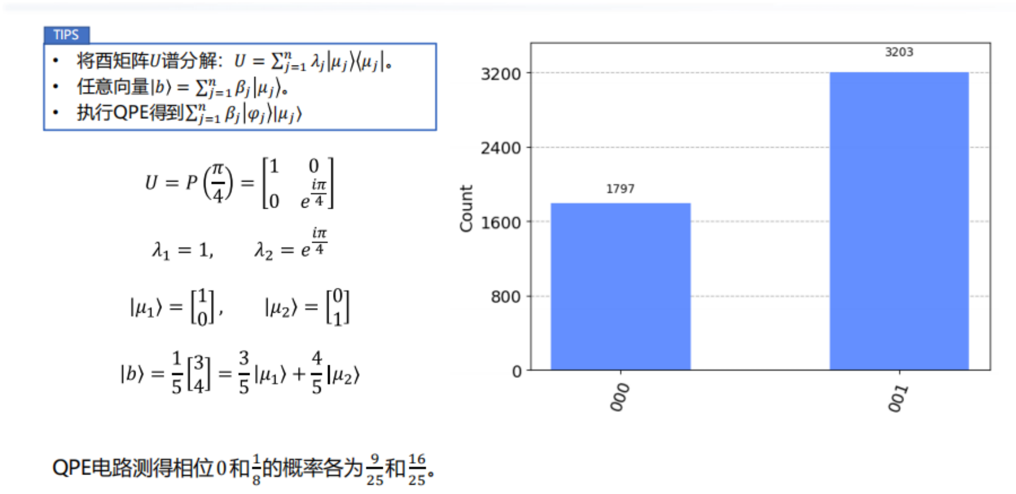
  利用如下命令对量子初态初始化

```python
qc.initialize([3/5,4/5], 3) ## Q2.4
```

  实验结果如下：QPE电路测得相位0和$1/8$的概率各为$\frac{377}{1000} = 0.377 \approx 0.36 = \frac{9}{25}$和$\frac{647}{1000} = 0.647 \approx 0.64 = \frac{16}{25}$。



  结果表明和以下理论结果一致：



> 理论结果来源PPT，当初态是非本征态 $|\varphi\rangle = [3/5, 4/5]^\top$时，我们测得的相位相位0和1/8的概率分别为9/25和16/25.