题目描述

2. （本题30分）对于线性系统$Ax = b$，系数矩阵A和右端b分别为
$A = [\frac{1}{\frac{\sqrt{2}}{2}} \quad \frac{1}{-\frac{\sqrt{2}}{2}}], b = [\frac{\frac{1}{2}}{-\frac{\sqrt{2}}{2}}]$
请基于pyqpanda给出HHL算法求解该线性系统的量子线路，并返回该方程的解x。
答题要求：
（1）选手需在IDE中编写完整代码，完整代码包含经典处理模块和量子线路模块，线路宽度限定为4比特；
（2）代码输出包含两个部分，第一部分为方程的解（List[double]，长度为2），第二部分为包含量子线路信息的OriginIR信息（string格式）（该信息可直接convert_qprog_to_originir函数获取）；
（3）选手应在本地运行代码，获取量子线路运行结果，并储存线路OriginIR信息；
（4）选手应在IDE的"OriginIR.txt"文件中黏贴对应的OriginIR字符串，注意字符串不包含前后引号，没有末尾换行；
（5）选手需提交一份.pdf格式说明文档，详细介绍求解过程，其中必须包含完整的线路解释，量子线路运行结果，以及将对应运行结果处理为方程的解的过程说明；
（6） 选手应仅根据pyqpanda中的基本量子逻辑门构建HHL算法量子线路，不允许使用pyqpanda自带的HHL、QPE、QFT等相关模块。

预处理

由于 HHL 算法在求解线性方程组时，要求系数矩阵 A 是一个厄米矩阵，即 A 的转置共轭等于它本身，其次要求输入 b 是一个单位向量。于是我们需要对原方程做等价变形：

1. 对等式两边同时左乘变换矩阵 $M = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 \end{bmatrix}$，使得系数矩阵变为厄米矩阵：

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

2. 再对等式两边同时乘以 $\frac{2\sqrt{2}}{\sqrt{5}}$，从而对 b 进行归一化，使其成为单位向量：

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} (\frac{2\sqrt{2}}{\sqrt{5}} x) = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{bmatrix}$$

3. 令 $A' = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$，$x' = \frac{2\sqrt{2}}{\sqrt{5}} x$，$b' = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{bmatrix}$，则可满足 HHL 算法的要求，因此求解原线性方程组，可转化为先求解 $A'x' = b'$，得到 x'后，只需对其除以 $\frac{2\sqrt{2}}{\sqrt{5}}$ 即可得到原方程组的解 x

HHL算法概述

HHL 算法主要包含以下三大步骤，并需要使用右端项比特、存储比特和辅助比特总共三个寄存器：

1. 相位估计，将矩阵 A' 的整数形式特征值全部转移到存储比特的基向量中。
2. 受控旋转，利用受控旋转门将特征值 λ_j 从存储比特的基向量转移到振幅上
3. 逆相位估计，对特征存储比特及右端项比特进行逆相位估计，将存储比特振幅上的特征值合并到右端项比特上，当辅助比特测量得到特定状态时，在右端项比特上可得到解的量子态。

具体线路图如下所示：

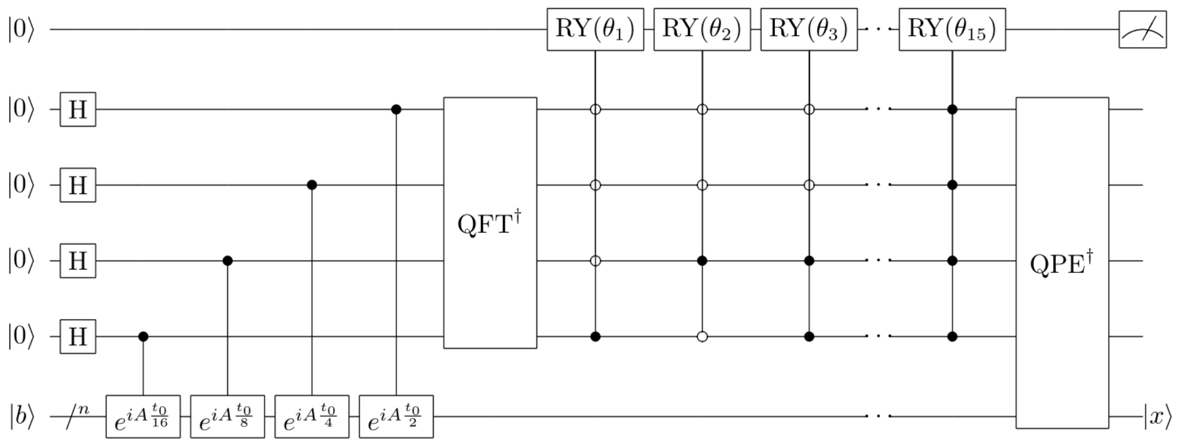


图1.HHL线路图

初态制备

$|b'\rangle$ 可以用一个量子比特表示，只需对 $|0\rangle$ 作用一个 $RY(\theta)$ 门即可得到，其中 θ 需要满足 $sin(\frac{\theta}{2}) = -\frac{2}{\sqrt{5}}$

```
# init the circuit
b=[1/sqrt(5),-2/sqrt(5)]
theta = np.arcsin(b[1])*2
HHL_circuit << RY(qubit[0], theta)
```

相位估计

注意到我们使用的变换后的矩阵 A' 满足性质 $A'^2 = I$，因此有：

$$U = e^{iA't_0} = cos(t_0)I + isin(t_0)A'$$

令 $t_0 = \frac{\pi}{2}$，可得 $U = iA' = \begin{bmatrix} \frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} & -\frac{i}{\sqrt{2}} \end{bmatrix}$，$U^2 = -A'^2 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

利用 pyQpanda 提供的 $U4$ 门接口可以方便的得到酉矩阵对应的量子门

再加一个量子傅里叶逆变换，即可完成相位估计电路的搭建，如下所示：

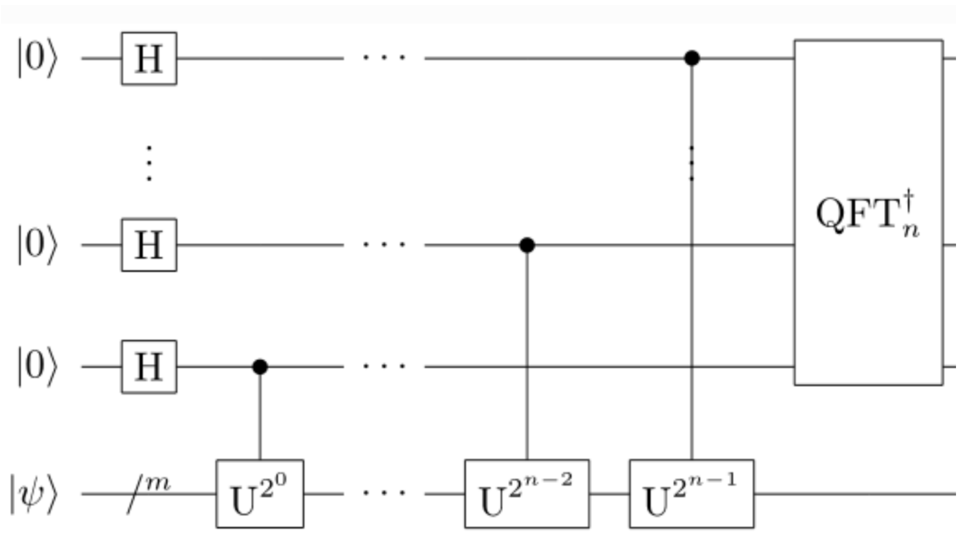


图2.相位估计线路图

其中量子傅里叶逆变换即是对量子傅里叶变换电路的反向搭建，本案例中为 $n=2$ 的情况，其对应的量子傅里叶逆变换线路如下图所示：

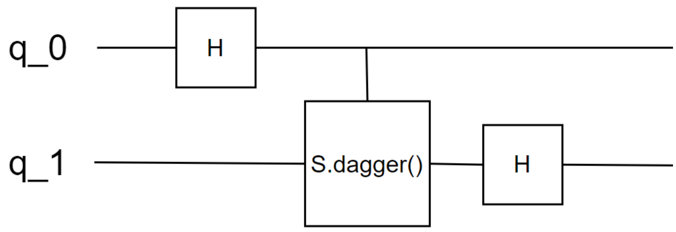


图3. 两量子比特傅里叶逆变换线路图

```
# prepare the QPE circuit
M1 = [1j/sqrt(2),1j/sqrt(2),1j/sqrt(2),-1j/sqrt(2)]
M2 = [-1,0,0,-1]
QPE_circuit = QCircuit()
QPE_circuit  << H(cbits) << U4(M1,qubit[0]).control(cbits[1]) << U4(M2,qubit[0]).control(cbits[0])\
             << H(cbits[0]) << S(cbits[1]).dagger().control(cbits[0]) << H(cbits[1])
```

令 A' 的本征值为 λ_j ，由于 $U=e^{iA't_0}$ ，因此有 U 的本征值为 $e^{i\lambda_j t_0}$ ，且 U 的本征向量与 A' 的本征向量相同

又由相位估计的定义可知，最终测量得到的量子态编码与 A' 的本征值之间的关系为：

1. 如果存在正整数 $2^n\in\mathbb{Z}$，则可以以概率 1 测量得到 $|c\rangle=|2^n\varphi\rangle$ ，其中 $|c\rangle$ 为测量后所得编码
2. 否则以至少概率 $\frac{4}{\pi^2}$ 得到最接近 $2^n\varphi$ 的整数$\tilde{\lambda}_j$ ，进而得到近似解

其中 φ 满足 $U|\psi\rangle=e^{i\lambda_j t_0}|\psi\rangle=e^{2\pi i\varphi}|\psi\rangle$,此处 $|\psi\rangle$ 为 U 和 A' 共同的本征向量

因此，我们可以得到如下关系：

$$\varphi=\frac{\lambda_j t_0}{2\pi}$$

最终，将 $t_0=\frac{\pi}{2},n=2$ 带入后得：$\varphi=\frac{\lambda_j}{4},|c\rangle=|\tilde{\lambda}_j\rangle$ ，即经过相位估计电路后存储比特对应的编码值即为 A' 的各个特征值的近似估计。

受控旋转

受控旋转门的作用为将特征值从存储比特的基向量转移到振幅，即要满足：

$$(\prod(CR(k)\otimes I))\sum_{j=0}^{N-1}b_j\,|0\rangle\left|\widetilde{\lambda_j}\right\rangle|u_j\rangle=\sum_{j=0}^{N-1}(\sqrt{1-\frac{C^2}{\widetilde{\lambda_j}^2}}|0\rangle+\frac{C}{\widetilde{\lambda_j}}|1\rangle)b_j\left|\widetilde{\lambda_j}\right\rangle|u_j\rangle.$$

由于 $n=2$，即精度为两位，故 $|\tilde{\lambda}_j\rangle$ 只能为 $|00\rangle$、$|01\rangle$、$|10\rangle$、$|11\rangle$，而根据线性代数知识，线性方程组若有唯一解，则系数矩阵对应的特征值一定不为零， 因此合法的 $|\tilde{\lambda}_j\rangle$ 只能为 $|01\rangle$、$|10\rangle$、$|11\rangle$ ，分别对应 $\tilde{\lambda}_j$ 为1、-2、-1，为满足 $C\le min_j|\tilde{\lambda}_j|=1$ ，我们可以选取 $C=\frac{1}{2}$，从而得到 RY 的旋转角 $\theta=2arcsin(\frac{C}{\tilde{\lambda}_j})$

具体来说，$|01\rangle$、$|10\rangle$、$|11\rangle$ 对应的 θ 角分别为 $2arcsin(\frac{1}{2})$、 $2arcsin(-\frac{1}{4})$、 $2arcsin(-\frac{1}{2})$

```python
# prepare the ROT circuit
ROT_circuit = QCircuit()
    # |01>
ROT_circuit << X(cbits[1]) << RY(tbit,2*np.arcsin(1/2)).control(cbits)\
    # |10>
    << X(cbits) << RY(tbit,2*np.arcsin(-1/4)).control(cbits)\
    # |11>
    << X(cbits[0]) << RY(tbit,2*np.arcsin(-1/2)).control(cbits)
```

逆相位估计

只需要加入 $QPE\_circuit$ 的转置共轭即可。

线路实现

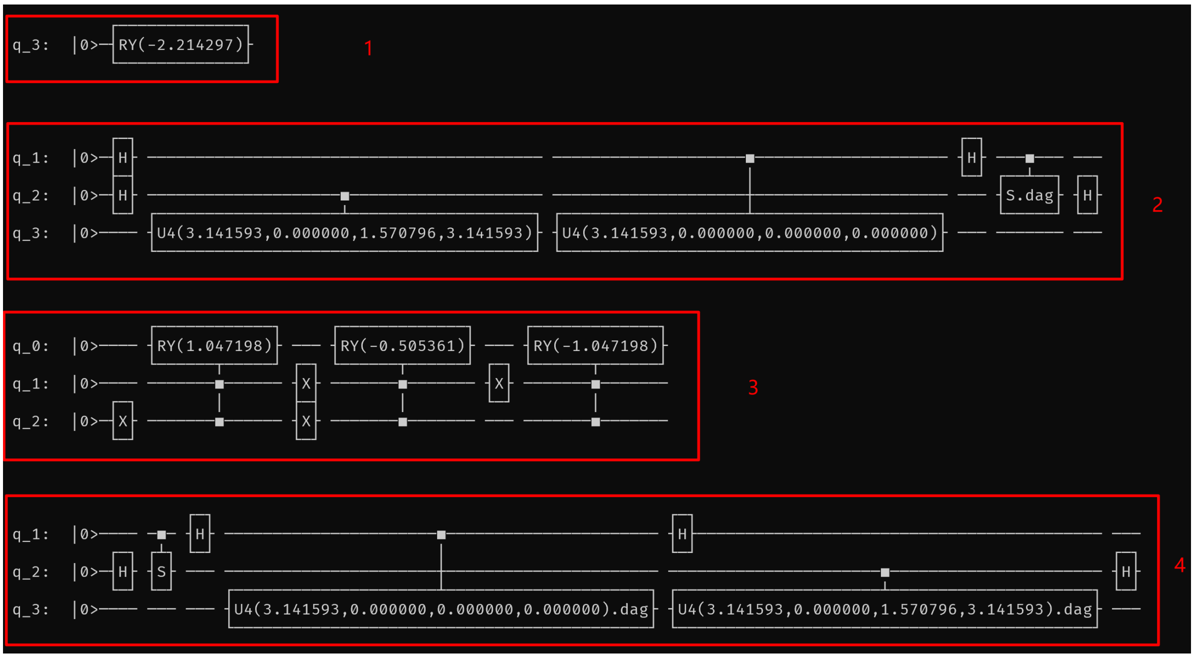
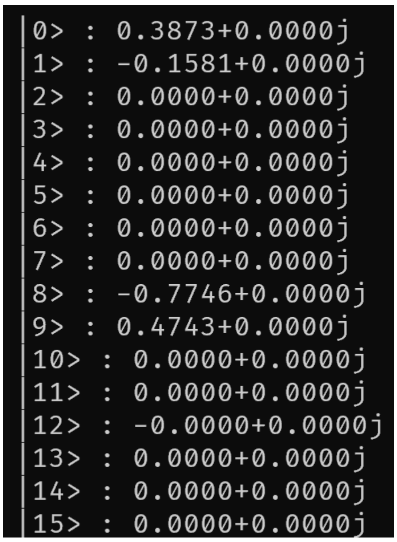


图4.由pyqpanda生成的量子线路

如上图所示，序号1,2,3,4分别为初态制备、相位估计、受控旋转以及逆相位估计电路。

运行后，最终得到的量子态为：



```
|0> : 0.3873+0.0000j
|1> : -0.1581+0.0000j
|2> : 0.0000+0.0000j
|3> : 0.0000+0.0000j
|4> : 0.0000+0.0000j
|5> : 0.0000+0.0000j
|6> : 0.0000+0.0000j
|7> : 0.0000+0.0000j
|8> : -0.7746+0.0000j
|9> : 0.4743+0.0000j
|10> : 0.0000+0.0000j
|11> : 0.0000+0.0000j
|12> : -0.0000+0.0000j
|13> : 0.0000+0.0000j
|14> : 0.0000+0.0000j
|15> : 0.0000+0.0000j
```

图5.测量结果

后处理

在上述量子线路中，我们需要得到当q_0比特为$|1\rangle$时，q_3比特的量子态，即 $|x'\rangle$，因此我们只需要关注结果为 $|0001\rangle$ 和 $|1001\rangle$ 的量子态，并根据条件概率公式，对其进行归一化处理，就可得到 $|x'\rangle$ 。根据预处理第3步，对$|x'\rangle$进行伸缩变换最终得到 $|x\rangle$。

```
extr = np.array([stat[1],stat[9]])
cof = 2*np.sqrt(2)/np.sqrt(5)
ans = extr/np.linalg.norm(extr)/cof
```

运行结果

```
The solution of the system of linear equations is:
 [-0.25+0.j  0.75+0.j]
```

图6.程序最终输出

即 $x = \begin{bmatrix} -0.25 \\ 0.75 \end{bmatrix}$

源代码

```
from typing import Tuple, Any

from pyqpanda import *
import numpy as np

def question1() -> Tuple[list, str]:
    # prepare the parameters
    # x' = x * cof
    cof = 2*np.sqrt(2)/np.sqrt(5)
    A = [1/np.sqrt(2), 1/np.sqrt(2), 1/np.sqrt(2), -1/np.sqrt(2)]
    b = [1/np.sqrt(5), -2/np.sqrt(5)]
    theta = np.arcsin(b[1])*2

    qvm = CPUQVM()
    qvm.init_qvm()

    abit = qvm.qAlloc_many(1) # ancilla bit
    cbits = qvm.qAlloc_many(2) # control bits
    qubit = qvm.qAlloc_many(1) # qubit
    prog = QProg()

    HHL_circuit = QCircuit()
    # init the circuit
    HHL_circuit << RY(qubit[0], theta)
    # prepare the QPE circuit
    M1 = [1j/np.sqrt(2),1j/np.sqrt(2),1j/np.sqrt(2),-1j/np.sqrt(2)]
    M2 = [-1,0,0,-1]
    QFT_circuit = QCircuit()
    QFT_circuit << H(cbits[1]) << S(cbits[1]).control(cbits[0]) << H(cbits[0])
    QPE_circuit = QCircuit()
    QPE_circuit << H(cbits) \
```

```python
                << U4(M1,qubit[0]).control(cbits[1]) << U4(M2,qubit[0]).control(cbits[0])  \
                << QFT_circuit.dagger()
    # print(QPE_circuit)
    # prepare the ROT circuit
    ROT_circuit = QCircuit()
    ROT_circuit << X(cbits[1]) << RY(abit,2*np.arcsin(1/2)).control(cbits)\
                << X(cbits) << RY(abit,2*np.arcsin(-1/4)).control(cbits)\
                << X(cbits[0]) << RY(abit,-2*np.arcsin(1/2)).control(cbits)
    # print(ROT_circuit)

    HHL_circuit << QPE_circuit << ROT_circuit << QPE_circuit.dagger()
    # print(HHL_circuit)
    prog << HHL_circuit
    print(prog)
    result = qvm.prob_run_list(prog, qubit, -1)
    # print(result)
    stat = qvm.get_qstate()
    # print(stat)
    for i in range(len(stat)):
        print("|{}> : {:.4f}".format(i,stat[i]))

    # stat is a list of complex numbers
    # I want to get the stat of q_3 when q_0 is |1>,that's to say,get the stat of |0001> and |1001>
    extr = np.array([stat[1],stat[9]])
    ans = extr/np.linalg.norm(extr)/cof
    # print(ans)

    OriginIR = convert_qprog_to_originir(prog,qvm)
    # print(OriginIR)
    qvm.finalize()
    # return res:
    return (ans, OriginIR)

if __name__ == '__main__':
    ans, ir = question1()
    print("The solution of the system of linear equations is:\n",ans)
#     print(ir)
```

参考链接

[HHL算法与实现](#)

[Solving Linear Systems of Equations using HHL](#)