# 量子计算

## —量子金融

# Quantum Finance

网址：www.qubits.top

作者：Calvin Tang

邮箱：179209347@qq.com

# 介绍

本教程基于 IBM 的 **Qiskit，Qiskit[finance]** 编写。

**本教程包含：**

1. 量子振幅估计
2. 量子线路
3. 代码实例

Qiskit：
https://qiskit.org/documentation/getting_started.html
Qiskit finance：
https://qiskit.org/documentation/finance/tutorials/index.html

Github & Gitee 代码地址：
https://github.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/00_amplitude_estimation.py
https://gitee.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/00_amplitude_estimation.py

Calvin，QQ: 179209347 Mail: 179209347@qq.com

# Qiskit，Qiskit[finance] 配置和安装

## 虚拟环境

```
# 创建虚拟环境
conda create -n ENV_NAME python=3.8.0
# 切换虚拟环境
conda activate ENV_NAME
# 退出虚拟环境
conda deactivate ENV_NAME
# 查看现有虚拟环境
conda env list
# 删除现有虚拟环境
conda remove -n ENV_NAME --all
```

## 安装 Qiskit

```
pip install qiskit
```

```
# install extra visualization support
# For zsh user (newer versions of macOS)
# pip install 'qiskit[visualization]'

pip install qiskit[visualization]
```

## 安装 Qiskit[finance]

```
# For zsh user (newer versions of macOS)
# pip install 'qiskit[finance]’

pip install qiskit[finance]
```

Calvin，QQ: 179209347  Mail: 179209347@qq.com

# 量子振幅估计 QAE （Quantum Amplitude Estimation）

给定一个算子 A, 作用于量子基态 $|0\rangle$ :

$$\mathrm{A}|0\rangle = \sqrt{1-a}|\psi_0\rangle + \sqrt{a}\,|\psi_1\rangle$$

量子振幅估计的作用是获得量子态 $|\psi_1\rangle$ 的振幅 $a$ :

$$a = |\langle\psi_1|\psi_1\rangle|^2$$

这个算法由Brassard et al. [1] 2000年给出，使用Grover算子的组合实现。

$$Q = \mathrm{AS_0\,A^{\dagger}\,S_{\psi 1}}$$

$S_0$ , $S_{\psi 1}$ 是 $|0\rangle$ 和 $|\psi_1\rangle$ 的翻转。算法需要复杂的量子线路，计算开销大。
因此，其它本案例中引入了其它改进的QAE算法。

在本案例中，算子 A 描述了一个Bernoulli 随机变量（假定未知），成功概率是$p$：

$$\mathrm{U}|0\rangle = \sqrt{1-p}|0\rangle + \sqrt{p}\,|1\rangle$$

[1] Quantum Amplitude Amplification and Estimation. Brassard et al (2000). https://arxiv.org/abs/quant-ph/0005055

# 量子振幅估计 QAE （Quantum Amplitude Estimation）

在量子计算机中，我们可以将算子A 表示为单比特绕Y轴的旋转：

$$\mathrm{A} = \mathrm{R_y}(\theta_p), \quad \theta_p = 2\sin^{-1}(\sqrt{p})$$

Grover算子：

$$\boldsymbol{Q} = \mathrm{R_y}(2\theta_p)$$

其k次幂：

$$\boldsymbol{Q}^{\mathrm{k}} = \mathrm{R_y}(2\mathrm{k}\theta_p)$$

* 详细的推导过程，参考量子计算【算法篇】第2章 振幅放大。

Calvin，QQ: 179209347 Mail: 179209347@qq.com

# Qiskit 振幅估计流程

在量子计算机中，Qiskit 实现了多个QAE算法，都实现了AmplitudeEstimator接口。
输入参数：
EstimationProblem
返回参数：
AmplitudeEstimationResult

1. 首先，创建EstimationProblem，其包含了A 算子和 Q算子， 在这个例子里，|Ψ1⟩设定为|1⟩：

```
problem = EstimationProblem(
    state_preparation=A,  # A operator
    grover_operator=Q,    # Q operator
    objective_qubits=[0], # the "good" state Psi1 is identified as measuring |1> in qubit 0
)
```

2. 实例化算法：

```
# Amplitude Estimation
ae = XXXXXXEstimation(
    num_eval_qubits=3,  # the number of evaluation qubits specifies circuit width and accuracy
    quantum_instance=quantum_instance,
)
```

3. 执行算法： `ae_result = ae.estimate(problem)`

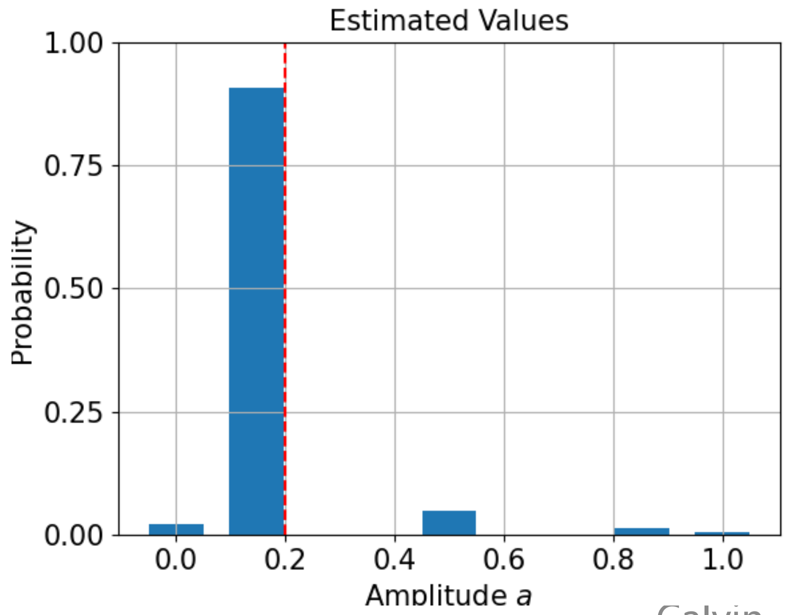
Calvin，QQ: 179209347  Mail: 179209347@qq.com

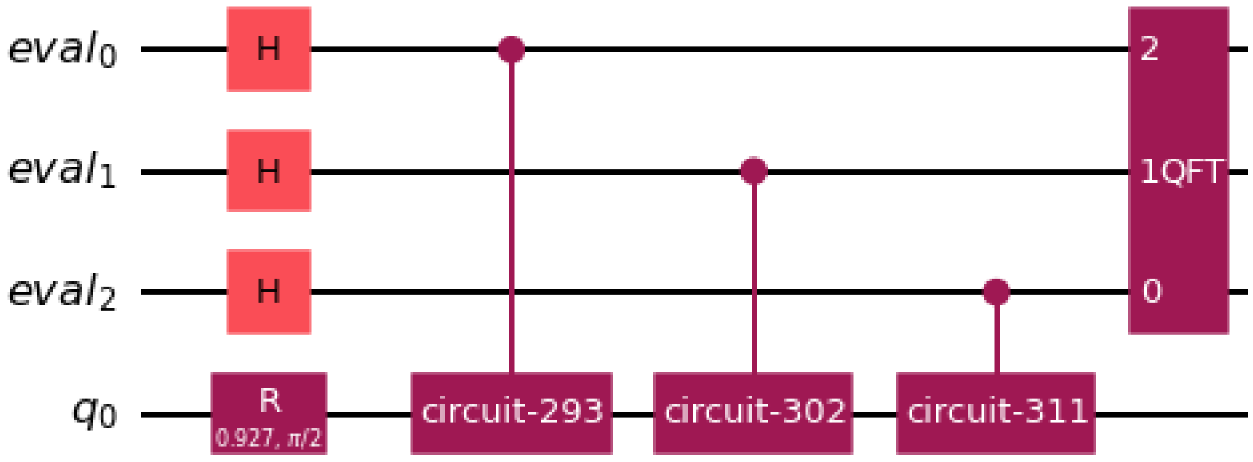
# 1. Canonical Amplitude Estimation

```
# Amplitude Estimation
ae = AmplitudeEstimation(
    num_eval_qubits=3,  # the number of evaluation qubits specifies circuit width and accuracy
    quantum_instance=quantum_instance,
)

ae_result = ae.estimate(problem)
print("Interpolated MLE estimator:", ae_result.mle)
```

Interpolated MLE estimator: 0.19999999390907777



# 2. Iterative Amplitude Estimation



```
from qiskit.algorithms import IterativeAmplitudeEstimation

iae = IterativeAmplitudeEstimation(
    epsilon_target=0.01,  # target accuracy
    alpha=0.05,  # width of the confidence interval
    quantum_instance=quantum_instance,
)

iae_result = iae.estimate(problem)

print("Estimate:", iae_result.estimation)
```

Estimate: 0.19999999999999998

# 3. Maximum Likelihood Amplitude Estimation

```
from qiskit.algorithms import MaximumLikelihoodAmplitudeEstimation

mlae = MaximumLikelihoodAmplitudeEstimation(
    evaluation_schedule=3, quantum_instance=quantum_instance  # log2 of the maximal Grover power
)

mlae_result = mlae.estimate(problem)

print("Estimate:", mlae_result.estimation)
```

Estimate: 0.20002237175368104

# 4. Faster Amplitude Estimation

```
from qiskit.algorithms import FasterAmplitudeEstimation

fae = FasterAmplitudeEstimation(
    delta=0.01,  # target accuracy
    maxiter=3,  # determines the maximal power of the Grover operator
    quantum_instance=quantum_instance,
)

fae_result = fae.estimate(problem)

print("Estimate:", fae_result.estimation)
```

```
Estimate: 0.20000000000000018
```

Qubits
qubits.top

# Thank You