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。
https://qiskit.org/documentation/finance/tutorials/07_asian_barrier_spread_pricing.html

本教程包含：
1. 亚式障碍期权(Asian barriers)
2. 量子算法 - 通过QAE求解问题
3. 代码实例
 * TODO：完善算法的详细解读

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/07_asian_barrier_spread_pricing.py
https://gitee.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/07_asian_barrier_spread_pricing.py

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]
```

亚式障碍期权 (Asian barriers)

亚式障碍期权 (Asian barriers) 套利包含了3个期权类型：

- **亚式期权（Asian option）**：又称为平均价格期权，是期权的衍生，是在总结真实期权、虚拟期权和优先认股权等期权实施的经验教训基础上最早由美国银行家信托公司(Bankers Trust)在日本东京推出的。它是当今金融衍生品市场上交易最为活跃的奇异期权之一，与通常意义上股票期权的差别是对执行价格的限制，其执行价格为执行日的前半年二级市场股票价格的平均价格。

- **障碍期权（barrier option）**：是指在其生效过程中受到一定限制的期权，其目的是把投资者的收益或损失控制在一定范围之内。障碍期权一般归为两类，即敲出期权和敲入期权。

- **牛市套利（barrier option）**：是指个股期权投资者在买入一手执行价格较低的期权的同时，也卖出一手具有相同到期日，但执行价格较高的期权，并且期权要么都为看涨期权，要么都为看跌期权。

亚式障碍期权 (Asian barriers) -（简化模型）

假定$K_1 < K_2$，且时间区间为 t = 1,2，相应的现货市价为(S_1 S_2)，barrier阈值 B > 0，
损益函数定义为：

$$P(S_1, S_2) = \begin{cases} \min\left\{\max\left\{\frac{1}{2}(S_1 + S_2) - K_1, 0\right\}, K_2 - K_1\right\}, & \text{if } S_1, S_2 \leq B \\ 0, & \text{otherwise.} \end{cases}$$

- T：期权到期日
- S_1 , S_2：现货市价 (spot price)
- K：成交价 (strike price)

在后面的例子里，量子计算算法基于振幅估计实现，估算到期损益：

$$\mathbb{E}\left[P(S_1, S_2)\right].$$

Uncertainty Model

在后面的例子里，量子计算算法基于振幅估计实现，估算到期损益：

我们构造一个量子线路，加载多变量对数正态分布(Log-Normal Distribution)数据，初始化 n 量子位量子态。对每一个维度 j = 1,...d，数据分布区间[low_j,$high_j$]，使用 2^{nj} 个网格点离散化，n_j 表示用于维度 j 使用的量子位数，例如： $n_j + ... + n_d = n$ 。幺正变换算子如下：

$$|0\rangle_n \mapsto |\psi\rangle_n = \sum_{i_1,\ldots,i_d} \sqrt{p_{i_1\ldots i_d}} |i_1\rangle_{n_1} \cdots |i_d\rangle_{n_d},$$

$P_{i1\ldots id}$表示离散分布概率，i为对应正确区间的仿射映射：

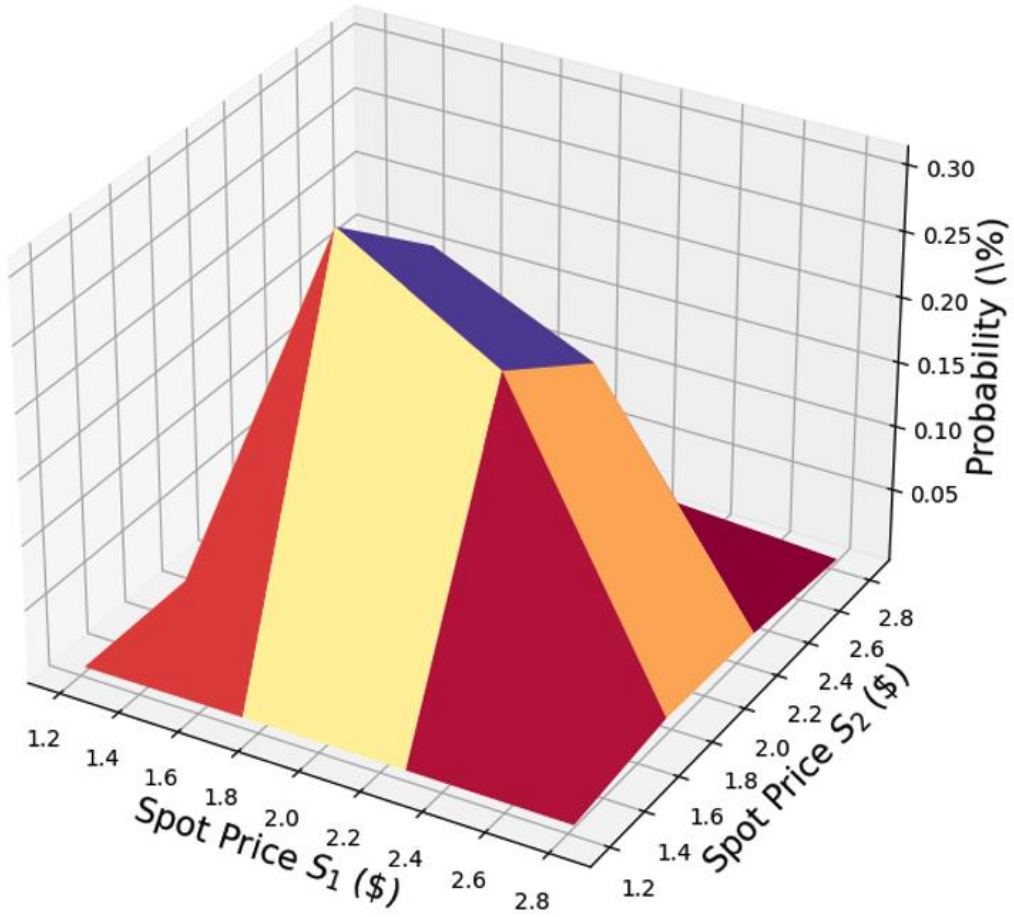
$$\{0,\ldots,2^{n_j}-1\} \ni i_j \mapsto \frac{\text{high}_j - \text{low}_j}{2^{n_j}-1} * i_j + \text{low}_j \in [\text{low}_j, \text{high}_j].$$

为了简化，我们假定两个股票价格是独立同分布的。当前实现最重要的假设是不同维度离散化网格具有相同的步长。

Uncertainty Model

```
# resulting parameters for log-normal distribution
mu = (r - 0.5 * vol**2) * T + np.log(S)
sigma = vol * np.sqrt(T)
mean = np.exp(mu + sigma**2 / 2)
variance = (np.exp(sigma**2) - 1) * np.exp(2 * mu + sigma**2)
stddev = np.sqrt(variance)
# lowest and highest value considered for the spot price; in
between, an equidistant discretization is considered.
low = np.maximum(0, mean - 3 * stddev)
high = mean + 3 * stddev
# map to higher dimensional distribution
# for simplicity assuming dimensions are independent and
identically distributed)
dimension = 2
num_qubits = [num_uncertainty_qubits] * dimension
low = low * np.ones(dimension)
high = high * np.ones(dimension)
mu = mu * np.ones(dimension)
cov = sigma**2 * np.eye(dimension)
```

```
# construct circuit
u = LogNormalDistribution(num_qubits=num_qubits,
mu=mu, sigma=cov, bounds=(list(zip(low, high))))
```



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

损益函数 （Payoff Function）

$S_1 + S_2 < K_1$ 时损益函数等于 0，然后损益函数线性增加，直到 $S_1 + S_2 = K_2$为止。损益维持常量$K_2 - K_1$，
除非任何一个S超过阈值B，然后损益立即降为 0.
使用一个带权重的sum算子对现货市价 (spot prices)求和，保存于一个辅助寄存器，然后使用一个比较器，
当 $S_1 + S_2 >= K_1$ 时，会交换辅助量子比特: |0> → |1>, 同时另一个比较器/辅助量子比特用于$S_1 + S_2 >= K_2$
辅助量子比特用于控制损益函数的线性部分。额外，我们使用另一个辅助变量，用于检查S_1， S_2 是否超过阈值B。
损益函数只应用于 S_1， S_2 <= B 情形。
当 |y| 足够小时：

$$\sin^2(y + \pi/4) \approx y + 1/2$$

因此，对于一个给定的近似缩放因子： $c_{\text{approx}} \in [0, 1]$ and $x \in [0, 1]$

有：

$$\sin^2(\pi/2 * c_{\text{approx}} * (x - 1/2) + \pi/4) \approx \pi/2 * c_{\text{approx}} * (x - 1/2) + 1/2$$

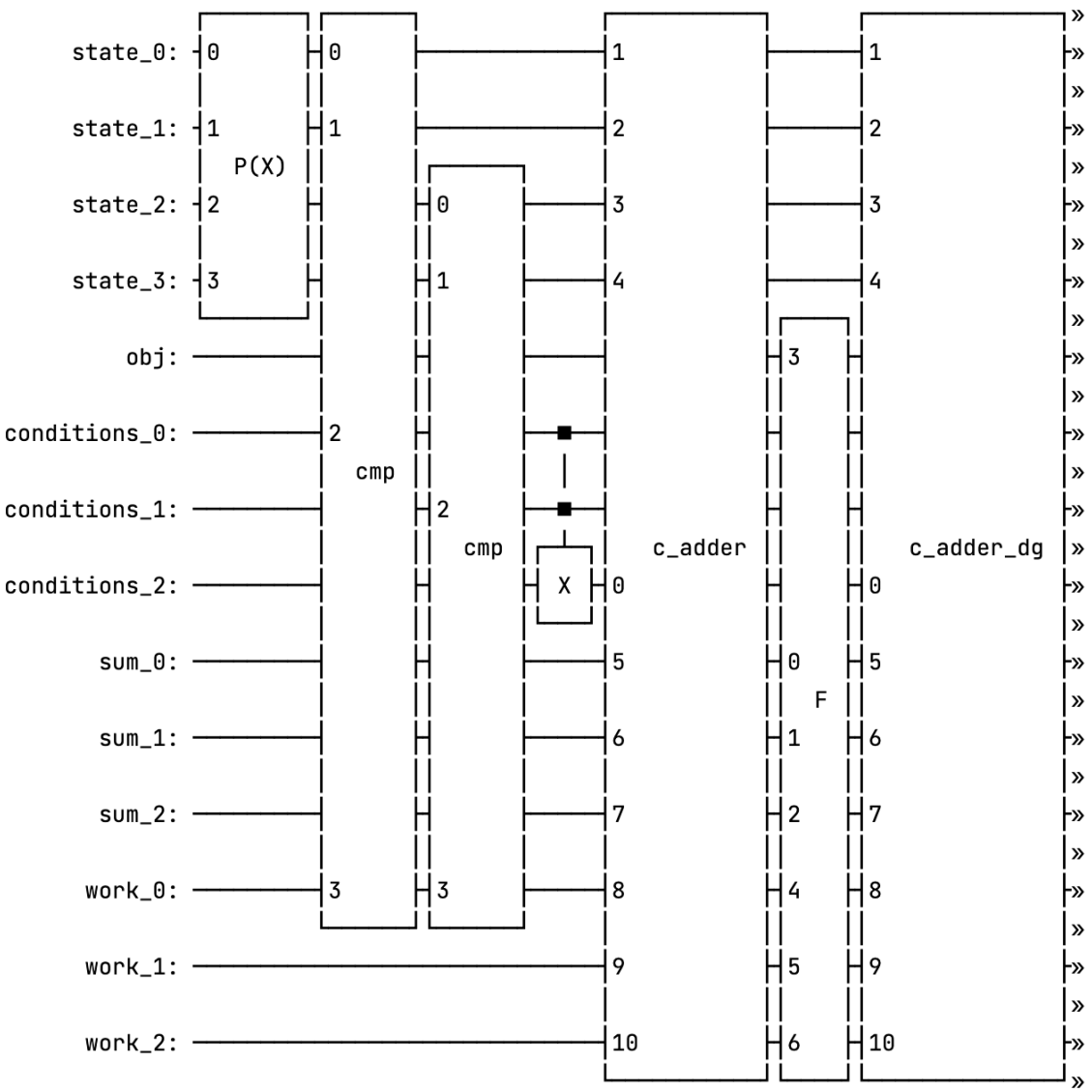
我们使用受控绕Y轴旋转，很容易构造一个算子：

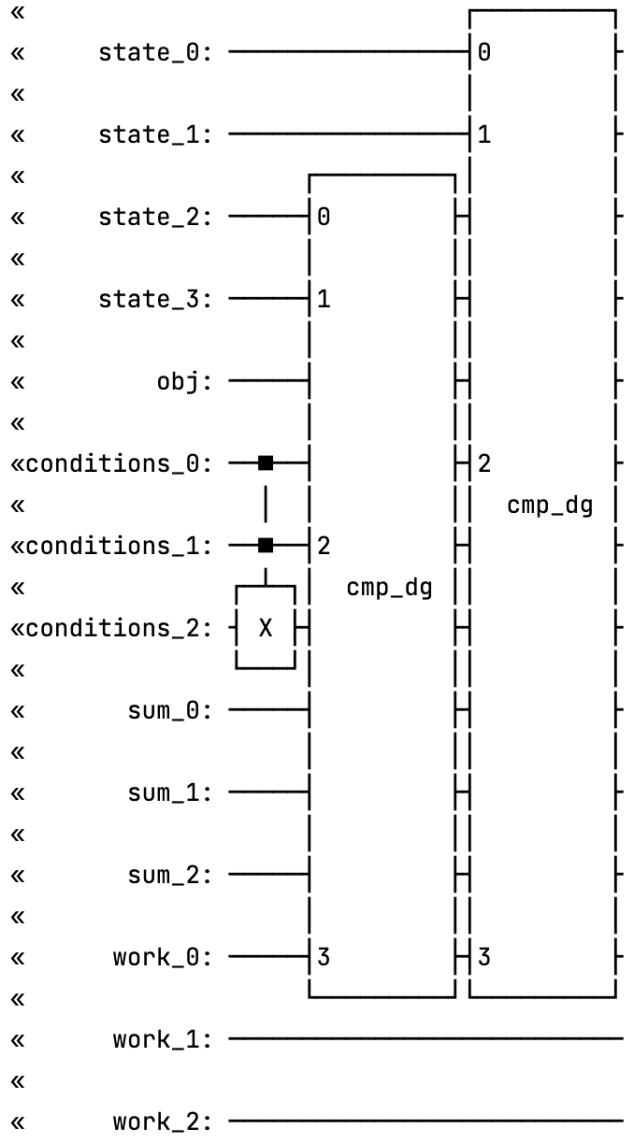
$$|x\rangle|0\rangle \mapsto |x\rangle \left(\cos(a * x + b)|0\rangle + \sin(a * x + b)|1\rangle\right)$$

最后，我们需要做的是测量|1>的概率振幅： $\sin^2(a * x + b)$

c_{approx} 越小越准，但是量子位m也需要相应调整。

量子线路

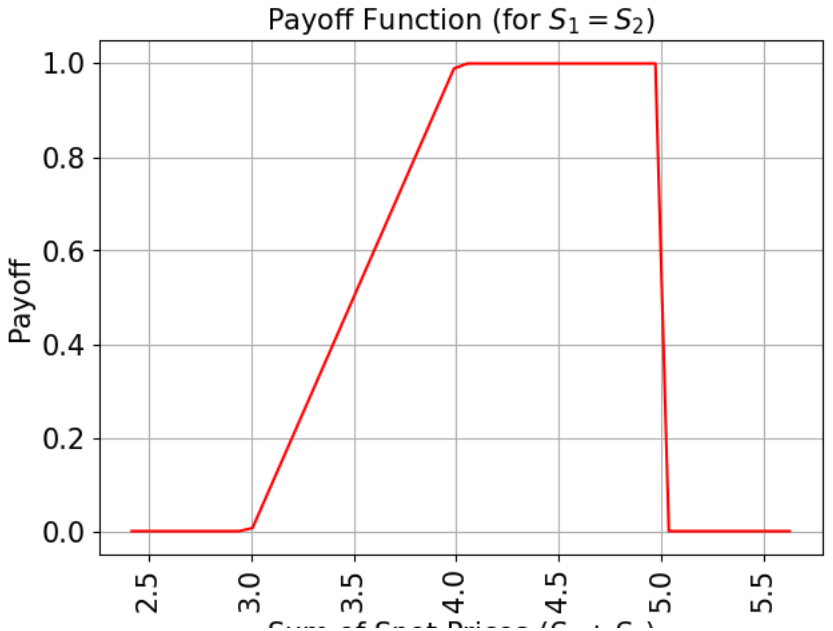




Calvin, QQ: 179209347 Mail: 179209347@qq.com

损益函数

```
# plot exact payoff function
plt.figure(figsize=(7, 5))
x = np.linspace(sum(low), sum(high))
y = (x <= 5) * np.minimum(np.maximum(0, x - strike_price_1), strike_price_2 - strike_price_1)
plt.plot(x, y, "r-")
plt.grid()
plt.title("Payoff Function (for $S_1 = S_2$)", size=15)
plt.xlabel("Sum of Spot Prices ($S_1 + S_2)$", size=15)
plt.ylabel("Payoff", size=15)
plt.xticks(size=15, rotation=90)
plt.yticks(size=15)
plt.show()
```



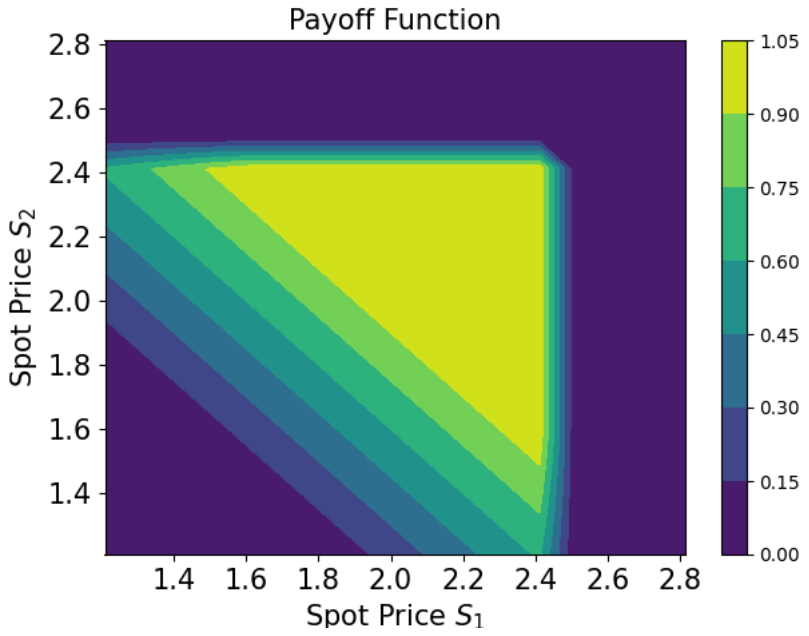
结果：

```
objective qubit index 4
exact expected value:    0.8023
```

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

损益函数

```
# plot contour of payoff function with respect to both time steps, including
barrier
plt.figure(figsize=(7, 5))
z = np.zeros((17, 17))
x = np.linspace(low[0], high[0], 17)
y = np.linspace(low[1], high[1], 17)
for i, x_ in enumerate(x):
    for j, y_ in enumerate(y):
        z[i, j] = np.minimum(
            np.maximum(0, x_ + y_ - strike_price_1), strike_price_2 - strike_price_1
        )
        if x_ > barrier or y_ > barrier:
            z[i, j] = 0
plt.title("Payoff Function", size=15)
plt.contourf(x, y, z)
plt.colorbar()
plt.xlabel("Spot Price $S_1$", size=15)
plt.ylabel("Spot Price $S_2$", size=15)
plt.xticks(size=15)
plt.yticks(size=15)
plt.show()
```



结果：

```
exact expected value:  0.8023
```

振幅估计

```
problem = EstimationProblem(
    state_preparation=asian_barrier_spread,
    objective_qubits=[objective_index],
    post_processing=objective.post_processing,
)
# construct amplitude estimation
ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha, quantum_instance=qi)
result = ae.estimate(problem)
conf_int = (
    np.array(result.confidence_interval_processed)
    / (2**num_uncertainty_qubits - 1)
    * (high_ - low_)
)
print("Exact value:    \t%.4f" % exact_value)
print(
    "Estimated value:\t%.4f"
    % (result.estimation_processed / (2**num_uncertainty_qubits - 1) * (high_ - low_))
)
print("Confidence interval: \t[%.4f, %.4f]" % tuple(conf_int))
```

结果：

```
Exact Operator Value:
0.6303
Mapped Operator value:
0.8319
Exact Expected Payoff:
0.8023
```

参考

[1] Quantum Risk Analysis. Woerner, Egger. 2018.
https://www.nature.com/articles/s41534-019-0130-6

[2] Option Pricing using Quantum Computers. Stamatopoulos et al. 2019.
https://quantum-journal.org/papers/q-2020-07-06-291/

Thank You