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/08_fixed_income_pricing.html

本教程包含：
1. 固定收益定价
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/08_fixed_income_pricing.py
https://gitee.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/08_fixed_income_pricing.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]
```

固定收益资产定价

固定收益资产价格是由未来预期现金流决定的，因此使用多期复利现值对其进行定价：

$$V = \sum_{t=1}^{T} \frac{c_t}{(1+r_t)^t}$$

- T：日期
- C_t：现金流
- r_t：利率

案例

面值100 元，票面利率为10 %，市场年利率为 9%，期限 3 年，每年支付一次利息，计算价格：

根据利息和本金的折现计算

$$V = \frac{10}{1+9\%} + \frac{10}{(1+9\%)^2} + \frac{110}{(1+9\%)^2}$$

Uncertainty Model

我们构造一个量子线路，加载多变量正态随机分布(Normal Distribution) d 维数据，初始化量子态。数据分布：

$$\otimes_{i=1}^{d}[low_i, high_i]$$

使用 2^{n_i} 个网格点离散化， n_i 表示用于维度 i = 1,…,d 使用的量子位数。么正变换算子如下：

$$|0\rangle_{n_1}\ldots|0\rangle_{n_d}\mapsto|\psi\rangle=\sum_{i_1=0}^{2^{n}-1}\cdots\sum_{i_d=0}^{2^{n}-1}\sqrt{p_{i_1,\ldots,i_d}}|i_1\rangle_{n_1}\ldots|i_d\rangle_{n_d},$$

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

$$\{0,\ldots,2^{n_j}-1\}\ni i_j\mapsto\frac{high_j-low_j}{2^{n_j}-1}*i_j+low_j\in[low_j,high_j].$$

另外，我们应用一个仿射映射，利率表示为：

$$\vec{r}=A*\vec{x}+b,\quad \vec{x}\in\otimes_{i=1}^{d}[low_i,high_i]$$

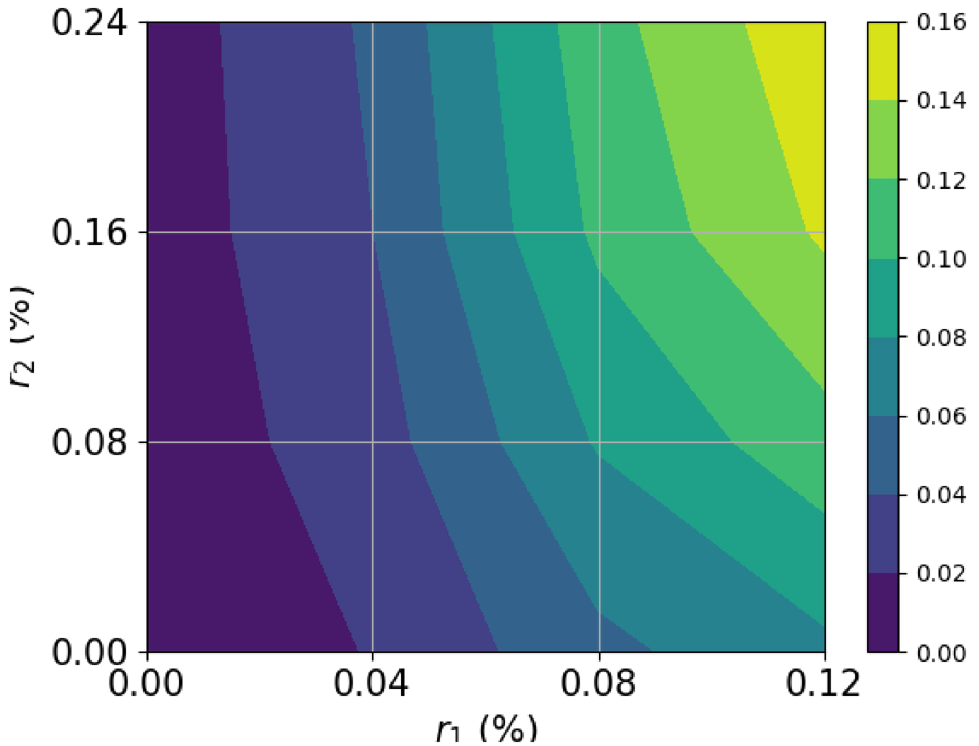
数据分布

```python
# specify the number of qubits that are used to represent the different dimenions of the uncertainty model
num_qubits = [2, 2]

# specify the lower and upper bounds for the different dimension
low = [0, 0]
high = [0.12, 0.24]
mu = [0.12, 0.24]
sigma = 0.01 * np.eye(2)
# construct corresponding distribution
bounds = list(zip(low, high))
u = NormalDistribution(num_qubits, mu, sigma, bounds)

# plot contour of probability density function
x = np.linspace(low[0], high[0], 2 ** num_qubits[0])
y = np.linspace(low[1], high[1], 2 ** num_qubits[1])
z = u.probabilities.reshape(2 ** num_qubits[0], 2 ** num_qubits[1])
```

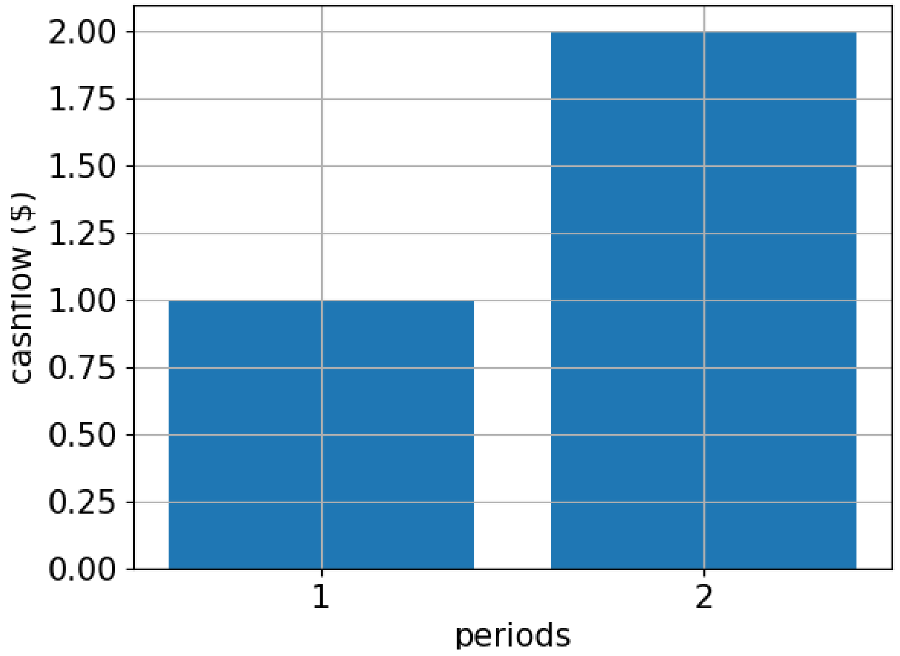
```python
plt.contourf(x, y, z)
plt.xticks(x, size=15)
plt.yticks(y, size=15)
...
plt.colorbar()
plt.show()
```



指定现金流

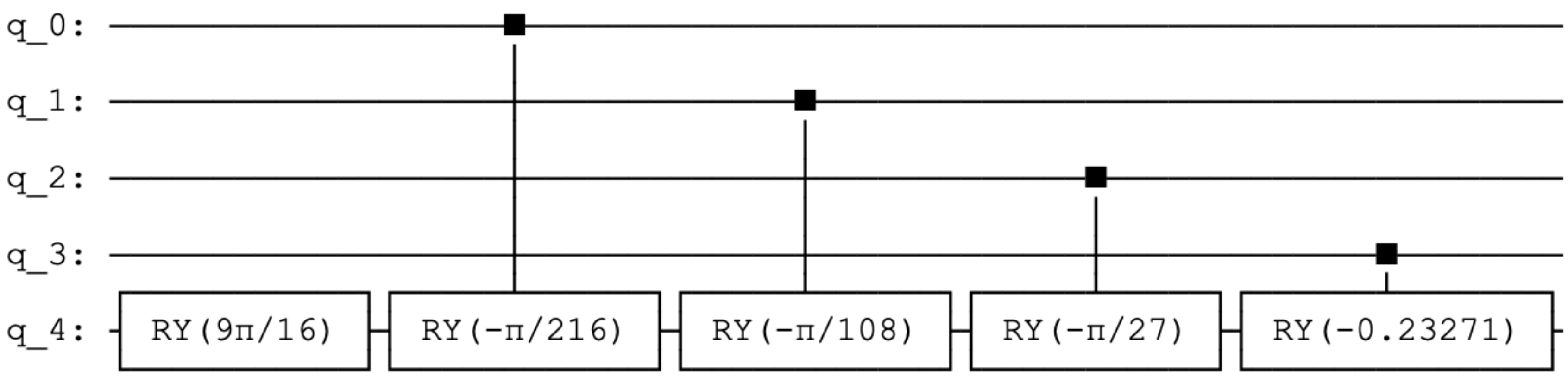
```
# specify cash flow
cf = [1.0, 2.0]
periods = range(1, len(cf) + 1)

# plot cash flow
plt.bar(periods, cf)
plt.xticks(periods, size=15)
plt.yticks(size=15)
plt.grid()
plt.xlabel("periods", size=15)
plt.ylabel("cashflow ($)", size=15)
plt.show()
```



量子线路

```python
# create fixed income pricing application
from qiskit_finance.applications.estimation import FixedIncomePricing
fixed_income = FixedIncomePricing(
    num_qubits=num_qubits,
    pca_matrix=A,
    initial_interests=b,
    cash_flow=cf,
    rescaling_factor=c_approx,
    bounds=bounds,
    uncertainty_model=u,
)
fixed_income._objective.draw()
```



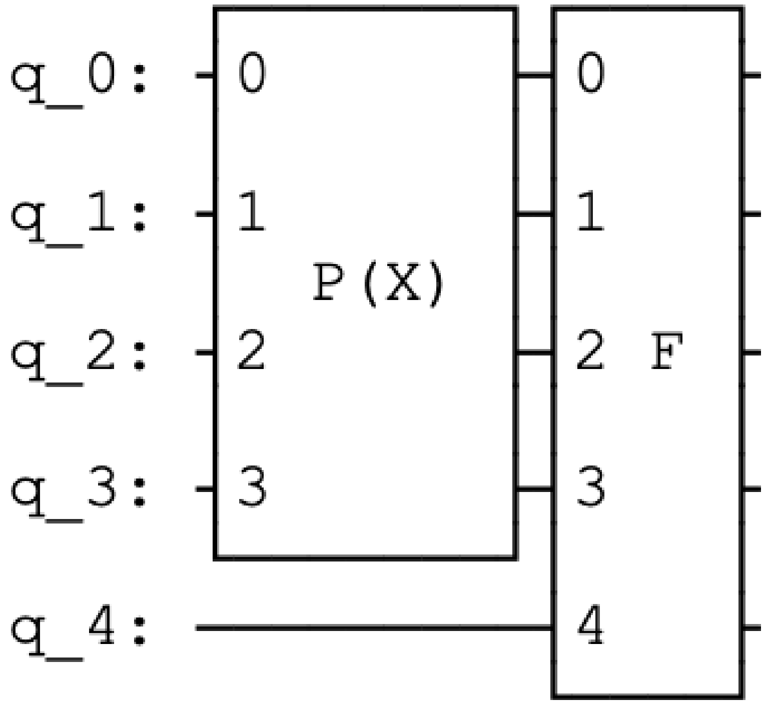
量子线路

```
fixed_income_circ = QuantumCircuit(fixed_income._objective.num_qubits)

# load probability distribution
fixed_income_circ.append(u, range(u.num_qubits))

# apply function
fixed_income_circ.append(fixed_income._objective,
range(fixed_income._objective.num_qubits))

fixed_income_circ.draw()
```



振幅估计

```
# set target precision and confidence level
epsilon = 0.01
alpha = 0.05

# construct amplitude estimation
qi = QuantumInstance(Aer.get_backend("aer_simulator"), shots=100)

problem = fixed_income.to_estimation_problem()

ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha, quantum_instance=qi)

result = ae.estimate(problem)

conf_int = np.array(result.confidence_interval_processed)
print("Exact value:        \t%.4f" % exact_value)
print("Estimated value:    \t%.4f" % (fixed_income.interpret(result)))
print("Confidence interval:\t[%.4f, %.4f]" % tuple(conf_int))
```

结果：

```
Exact value:
2.1942
Estimated value:
2.3442
Confidence interval:
[2.2932, 2.3951]
```

参考

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

Qubits
qubits.top

Thank You