Financial Options Pricing with Quantum Computing

Dawid Kopczyk

Warsaw Quantum Computing meeting

25.05.2020

Outline

Introduction

Motivation

- The US stock and bond markets are capitalized at more than **$ 70 trillion**.
- Any optimization and speed-up accomplished in the area of financial risk analysis and pricing derivatives can have **a big impact on the success of financial institutions** and their customers
- Quantum computing can be used to **speed-up Monte Carlo method quadratically**.

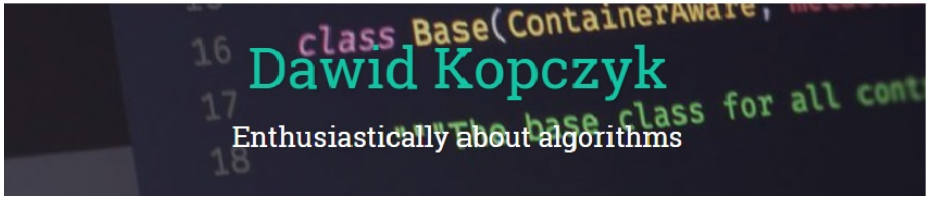
Introduction

Assumptions

- No knowledge about Monte Carlo and financial engineering.
- Basic knowledge of quantum computing
- A little bit of Python

Introduction

About me



Option Pricing: Definitions

Intro

This section will introduce you to the world of **financial engineering** defined as the application of technical methods, especially from mathematical finance and computational finance, in the practice of finance.

Financial Option Definition

A financial option is a contract that gives the buyer the **right** to buy or sell an asset at a particular price, **at a point** in the future.

Option Pricing: Definitions

Vanilla European call option

A **European** style option can only be used once the option has reached a predetermined **maturity date** in the future T. A **call** option gives the holder of the option the right to **buy** at a known price K.

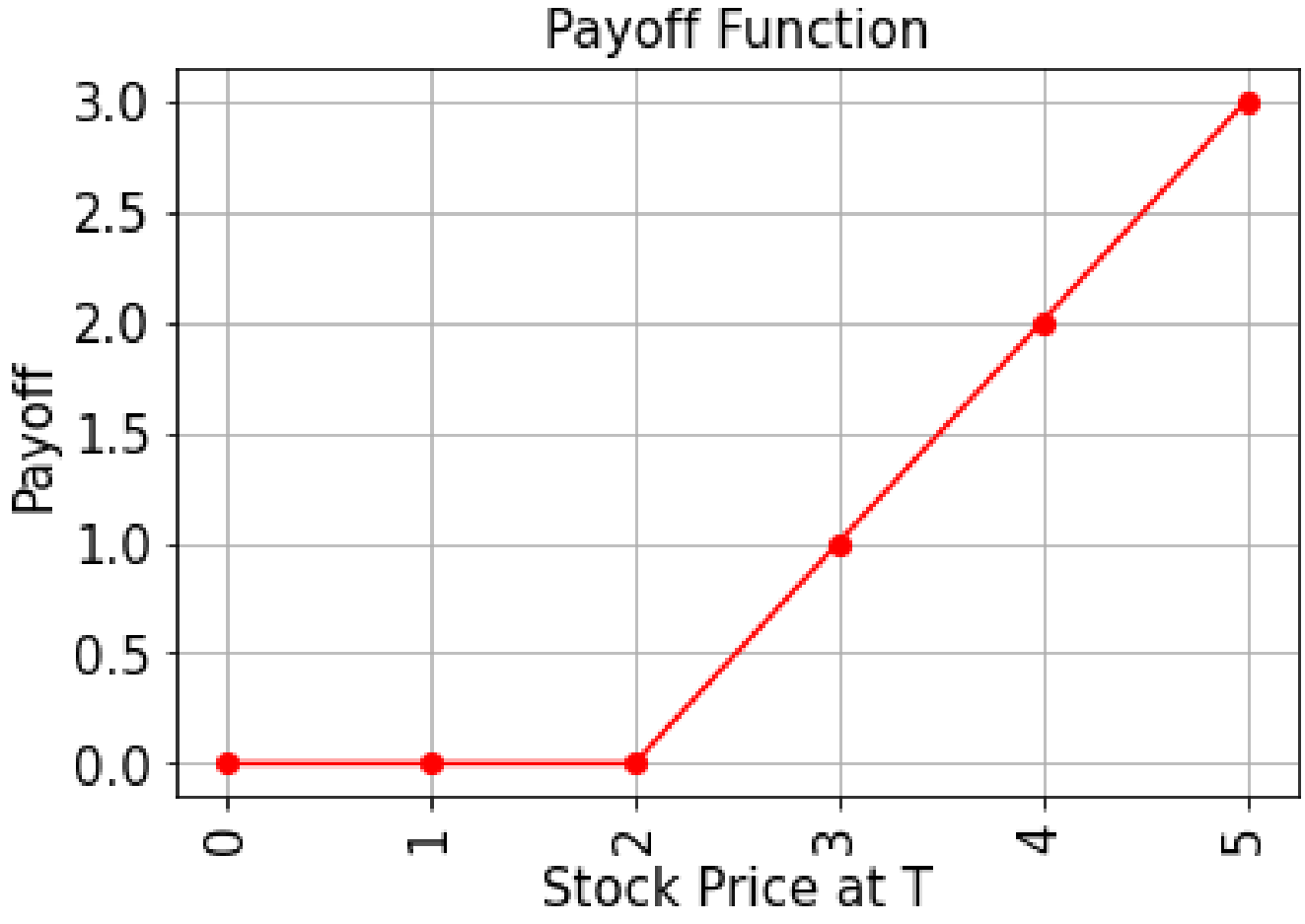
Payoff

A call option makes money if the price of the asset at maturity date, denoted by S_T, is above the strike price K, otherwise it's worth nothing:

$$C_T = \max(S_T - K, 0) \tag{1}$$

Option Pricing: Definitions

The price of stock S_T has a stochastic nature. Thus, C_T is also a random variable. Let's vary S_T and see how it impacts payoff C_T.



Option Pricing: Definitions

The expected payoff at expiry date T is defined as:

$$\mathbb{E}(C_T) = \mathbb{E}(\max(S_T - K, 0)) \tag{2}$$

The expected price of the option at any time $t < T$ is a present value PV of expected payoff:

$$\begin{aligned} \mathbb{E}(C_t) &= PV\left(\mathbb{E}(\max(S_T - K, 0))\right) \\ &= e^{r(T-t)}\mathbb{E}(\max(S_T - K, 0)) \end{aligned} \tag{3}$$

where r is the risk-free rate used for discounting.

Option Pricing: Black-Scholes Model

Black-Scholes

We can price an European call option using **Black-Scholes formula** which is a closed-form solution.

$$C_t = S_t\mathcal{N}(d_1) - Ke^{-r(T-t)}\mathcal{N}(d_2) \tag{4}$$

where

- $\mathcal{N}$ is the normal distribution
- σ is the volatility (standard deviation) of the stock
- $d_1 = \frac{\ln\left(\frac{S_t}{K}\right)+\left(r+\frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}$
- $d_2 = d_1 - \sigma\sqrt{T-t}$

Option Pricing: Black-Scholes Model

Black-Scholes Assumptions

- the stock price follows a geometric Brownian motion, i.e. in this case $\log(S_t) \sim \mathcal{N}$
- the volatility σ is constant
- the risk-free interest rate r is constant
- the stock does not pay dividend

Option Pricing: Monte-Carlo method

Motivation

There is no closed-form solution of option price (or is extremely difficult to find it) if:

- the underlying asset follows an arbitrary probability distribution for example hyperbolic, variance-gamma, etc. and/or ...
- ... the volatility σ is non-constant and is a random variable and/or...
- ... the risk-free interest rate r is non-constant and is a random variable and/or ...
- ... we deal with exotic options for example insurance guarantee products which can be treated as complex put options depending on additional risk factors such as inflation, mortality, etc.

Option Pricing: Monte-Carlo method

Code Example

The notebook is available here: `https://github.com/dawidkopczyk/training/blob/master/WQCG_2020_Quantum_Monte_Carlo_Tutorial/Quantum_Monte_Carlo.ipynb`

Option Pricing: Monte Carlo method

Problem Statement

The classical Monte Carlo method is good enough, however for most realistic examples in which stock price, interest rates and stock volatility follow arbitrary probability distribution with correlations between them, we require a huge amount of simulations to estimate the fair price correctly.

Option Pricing: Monte Carlo method

Convergence

Using Chebyshev's inequality with λ^2 as upper bound of variance

$$P[|\mathbb{E}(C) - C_{true}| \geq \epsilon] \leq \frac{\lambda^2}{N\epsilon^2} \tag{5}$$

Thus, for a constant success probability we require

$$N = \mathcal{O}\left(\frac{\lambda^2}{\epsilon^2}\right) \tag{6}$$

scenarios to reach given ϵ. In other words, Monte-Carlo method converges as $\mathcal{O}(\frac{1}{\sqrt{N}})$.

Quantum Algorithm for Monte Carlo

Goal

Use quantum phase estimation to estimate $\mathbb{E}(C)$ so that the algorithm converges as $\mathcal{O}(\frac{1}{N})$, which is a quadratic speed-up over classical Monte Carlo.

Quantum Algorithm for Monte Carlo: Overview

1. **Distribution Loading:** Encoding stock price probability distribution (for example $S_t \sim log\mathcal{N}$) into a quantum state using operator $\mathcal{A}$.

2. **Controlled Y-rotations** Express payoff function $max(S_T - K, 0)$ as quantum circuit and use rotation $\mathcal{R}$ of an ancilla qubit so that we can retrieve expectation value $\mathbb{E}(C)$ from probability of measuring ancilla qubit in state $|1\rangle$.

3. **Quantum Phase Estimation**: Allows the efficient estimation of the probability of measuring ancilla qubit in state $|1\rangle$.

Quantum Algorithm for Monte Carlo: Overview

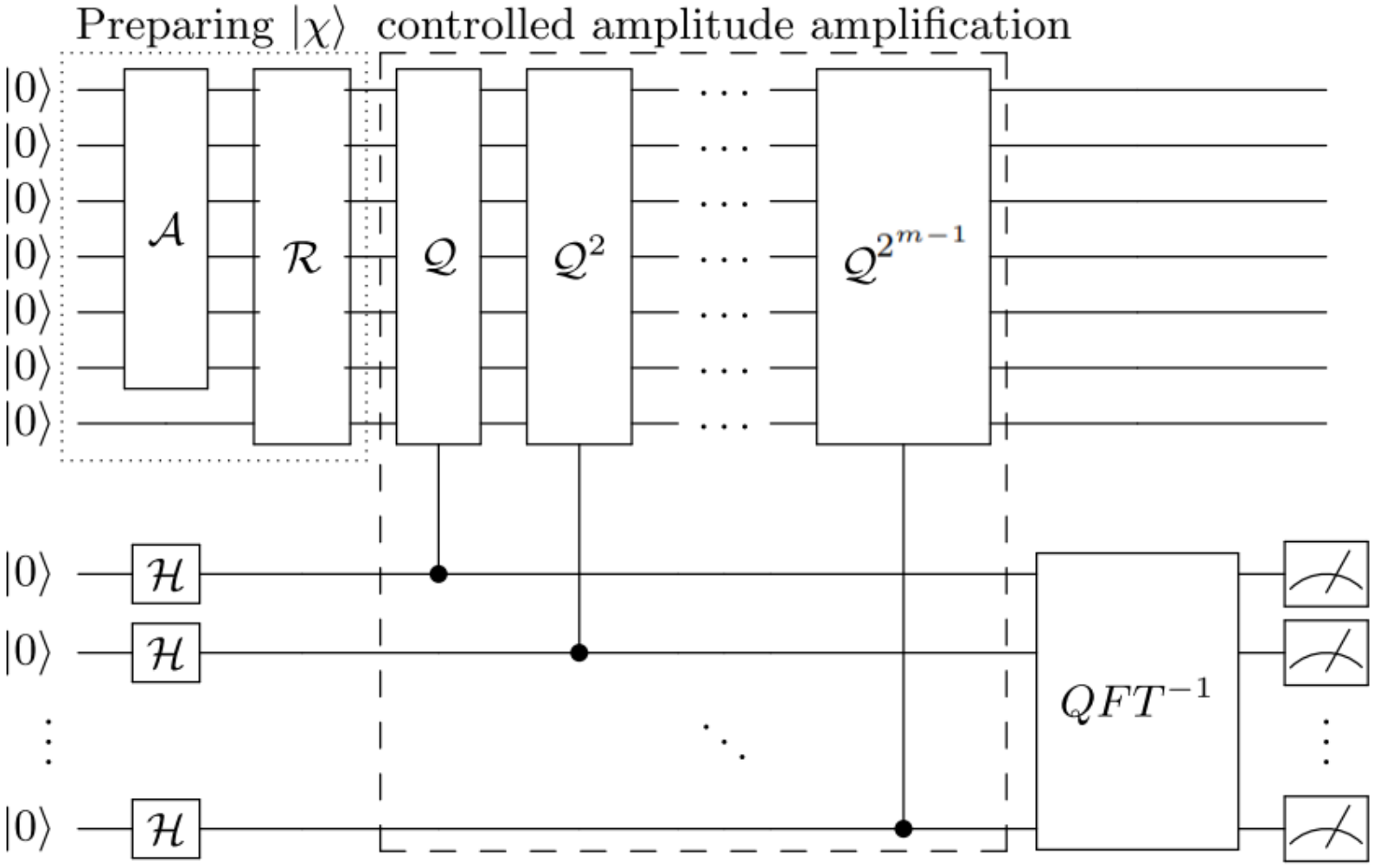


Figure 1: Circuit of quantum algorithm for Monte Carlo method.

Quantum Algorithm for Monte Carlo: Distribution Loading

The expectation value of variable x is defined as:

$$\mathbb{E}(x) = \sum_{i=0}^{N-1} p_i x_i \tag{7}$$

where p_i is the probability of realisation x_i and $\sum_{i=0}^{N-1} p_i = 1$

Quantum Algorithm for Monte Carlo: Distribution Loading

Suppose you can prepare the following quantum state with operator $\mathcal{A}$:

$$\mathcal{A}\,|0\rangle^n = \sum_{i=0}^{2^n-1} \sqrt{p_i}\,|i\rangle \tag{8}$$

The state $|i\rangle$ is one of the $N = 2^n$ possible realizations of a bounded discrete random variable x. Example: for $n = 2$, we have $|0\rangle = |00\rangle$, $|1\rangle = |01\rangle$, $|3\rangle = |11\rangle$. For each $|i\rangle$ there can be a mapping to express it as a real value.

In other words, we can encode a random probability distributions defined by probabilities p_i into a quantum state.

Quantum Algorithm for Monte Carlo: Expectation Value

Now, let's consider an arbitrary function
$f : \{0, \cdots, N-1\} \to \{0, 1\}$ and a rotation of an ancilla qubit:

$$\mathcal{R} |i\rangle |0\rangle = |i\rangle \left(\sqrt{1 - f(i)} |0\rangle + \sqrt{f(i)} |1\rangle \right) \tag{9}$$

When we apply $\mathcal{R}$ on $\mathcal{A} |0\rangle^n |0\rangle$ we would transform state into:

$$\sum_{i=0}^{2^n-1} \sqrt{p_i} |i\rangle \to \sum_{i=0}^{2^n-1} \sqrt{p_i} |i\rangle \left(\sqrt{1 - f(i)} |0\rangle + \sqrt{f(i)} |1\rangle \right) = |\chi\rangle \tag{10}$$

Quantum Algorithm for Monte Carlo: Expectation Value

Measurement of ancilla qubit in state $|1\rangle$ yields expectation value $\mathbb{E}(f(i))$ as success probability:

$$\sum_{i=0}^{2^n-1} |\sqrt{p_i f(i)}|^2 = \sum_{i=0}^{2^n-1} p_i f(i) = \mathbb{E}(f(i)) \tag{11}$$

Quantum Algorithm for Monte Carlo: Expectation Value

The success probability can be obtained by repeating the procedure M times, however for given accuracy ϵ the experiment has to be repeated $M = \mathcal{O}(\frac{1}{\epsilon^2})$, which is the same as for classical Monte Carlo.

The task of the Quantum Phase Estimation algorithm will be to improve the ϵ dependence from ϵ^2 to ϵ providing quadratic speed-up.

Quantum Algorithm for Monte Carlo: Preparation of $\mathcal{Q}$

Quantum Phase Estimation for given unitary G finds its eigenvalues. Let's define $\mathcal{F}$ as:

$$\mathcal{F} := \mathcal{R}(\mathcal{A} \otimes \mathcal{I}_2) \tag{12}$$

so that

$$\mathcal{F}\,|0\rangle^{n+1} := |\chi\rangle \tag{13}$$

We have to somehow change the operator $\mathcal{F}$ to an operator $\mathcal{Q}$, so that the eigenvalues of $\mathcal{Q}$ are linked to quantity of interest i.e. $\sum_{i=0}^{2^n-1} |\sqrt{p_i f(i)}|^2 = \mathbb{E}(f(i))$.

Quantum Algorithm for Monte Carlo: Preparation of $\mathcal{Q}$

Let's define $\mathcal{Q}$ as:

$$\mathcal{Q} := \mathcal{U}\mathcal{V}\mathcal{U}\mathcal{V} \tag{14}$$

where:

- $\mathcal{V} = \mathcal{I}_{2^{n+1}} - 2\mathcal{I}_{2^n} \otimes |1\rangle\langle 1|$
- $\mathcal{U} = \mathcal{F}\mathcal{Z}\mathcal{F}^\dagger$ and $\mathcal{Z} = \mathcal{I}_{2^{n+1}} - 2\,|0\rangle^{n+1}\langle 0|^{n+1}$

Quantum Algorithm for Monte Carlo: Preparation of $\mathcal{Q}$

The eigenvalues of $\mathcal{Q}$ are $e^{\pm i\theta}$ where θ is linked to $\mathbb{E}(f(i))$ via:

$$1 - 2\mathbb{E}(f(i)) = \cos(\theta/2) \tag{15}$$

In summary, we transformed $\mathcal{F} \to \mathcal{Q}$ so that we are able to use Quantum Phase Estimation algorithm. After we find the θ, we recover back the expectation value of $\mathbb{E}(f(i))$.

Quantum Algorithm for Monte Carlo: Quantum Phase Estimation

The detailed description of Quantum Phase Estimation algorithm can be found `http://dkopczyk.quantee.co.uk/qpe/`.

In short, phase estimation of θ relies on conditional application of $\mathcal{Q}$ $M = 2^m$ times and then inverse Quantum Fourier Transform is applied to peak the probability of finding correct value $\hat{\theta}$.

Phase estimation uses m additional sampling qubits to represent the result and $M = 2^m$ applications of $\mathcal{Q}$, i.e., M quantum samples.

Quantum Algorithm for Monte Carlo: Speed-up

To achieve $|\mathbb{E}(C) - C_{true}| \geq \epsilon$ with probability $1 - \delta$ it suffices to take $M = \mathcal{O}(\frac{1}{\epsilon})$ quantum samples which provides a quadratic speed-up in comparison to ϵ^2 dependence!

In other, words Quantum Algorithm for Monte Carlo converges as $\mathcal{O}(\frac{1}{M})$. An example: if classical Monte Carlo reaches $\epsilon = 10^{-4}$ in $N = 1000000$ samples, quantum algorithm should reach the same level of accuracy in $M = \sqrt{1000000} = 1000$ quantum samples.

Quantum Algorithm for Monte Carlo: Speed-up

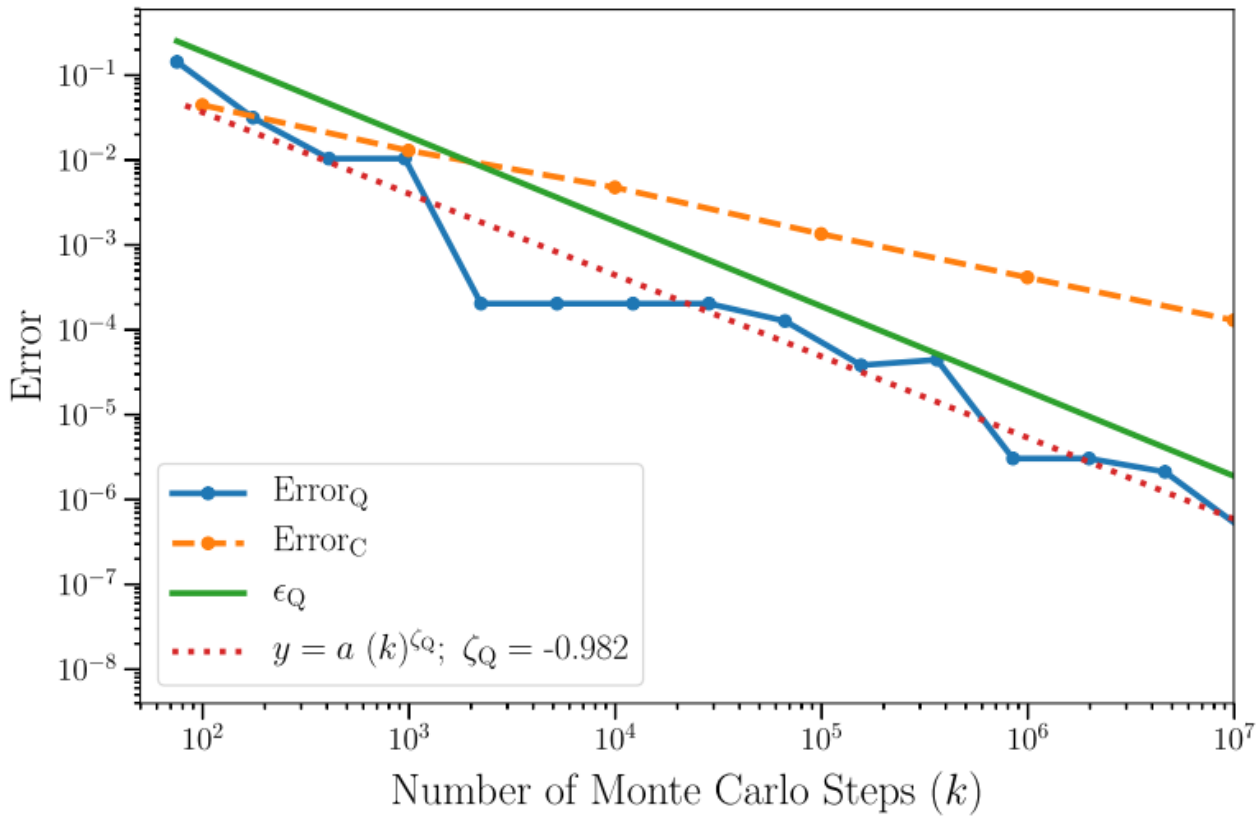


Figure 2: Monte Carlo convergence for European call option from paper `https://arxiv.org/abs/1805.00109`.

Quantum Algorithm for Monte Carlo: Code Example

Code Example

The notebook is available here: `https://github.com/dawidkopczyk/training/blob/master/WQCG_2020_Quantum_Monte_Carlo_Tutorial/Quantum_Monte_Carlo.ipynb`

Summary: Extensions

- In theory, Quantum Algortihm for Monte Carlo can be used not only for option pricing, but also bonds or other financial derivatives (swaps, etc.)
- We only considered expectation value (price) of a financial asset. However, in paper Quantum Risk Analysis, there is a procedure to calcualate other statistcs such as variance, value-at-risk (VaR), etc.

Summary: Challenges

1. The operation of the algorithm is much limited due to **quantum noise** which is a reality in today's quantum computers.

2. How to construct **complex payoff functions**? Would the time required to construct such circuits be overwhelming relatively to possible benefits?

3. What to do with non log-concave probability distributions (possible solution: qGANs)?

Summary: Further read

- P. Rebentrost, B. Gupt, T. R. Bromley (2018). *Quantum computational finance: Monte Carlo pricing of financial derivatives* `https://arxiv.org/abs/1805.00109`
- S. Woerner, D. J. Egger (2019). *Quantum Risk Analysis* `https://arxiv.org/abs/1806.06893`
- N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner (2020). *Option Pricing using Quantum Computers* `https://arxiv.org/pdf/1905.02666.pdf`
- `https://github.com/dawidkopczyk/training/blob/master/WQCG/grover.ipynb`
- Tutorial on Quantum Phase Estimation algorithm `http://dkopczyk.quantee.co.uk/qpe/`.