Accuracy of Quantum Methods for Black-Scholes Option Pricing

Willó Corry

January 2023

Natural Science Specialisation Course: Future Horizons

Supervisors: Felicia Dinnétz & Per-Olof Freerks

Natural Science

Kungsholmens Gymnasium

NA20EC

Abstract

Financial markets are complex systems which are notoriously difficult to predict. This makes it especially difficult to price option contracts whose value depend on the future value of its underlying securities. Even so, there exist techniques for pricing these types of financial instruments. In this paper we investigated the accuracy of two different algorithms for pricing European call option contracts that don't pay dividends: Monte Carlo and Quantum Amplitude Estimation. The error was measured as the relative difference in percent between the estimated price and the valuation given by the Black-Scholes formula which is a popular mathematical model for pricing options. The algorithms were run on five different sets of test data. By applying a mapping onto the results and using a Pearson correlation test it was shown that the error of the Monte Carlo algorithm is significantly and positively correlated with the function $\frac{1}{\sqrt{n}}$ and hence scales as $O(\frac{1}{\sqrt{n}})$, where the significance level was set to 0.05. Monte Carlo also seemed to perform better on distributions with a lower standard deviation. On the other hand the test yielded p-values above 0.05 for the Quantum Amplitude Estimation algorithm. Hence there was not a statistically significant correlation with the function $\frac{1}{n}$. A possible explanation for this could have been due to an insufficient number of qubits available on hardware. All the code and test data is available on GitHub: https://github.com/FloppingCode/DP_BlackScholes.

Keywords: Quantum Computing, Option Pricing, Monte Carlo, Black-Scholes

Contents

1 Introduction

In the field of mathematical modeling, it is not uncommon that the process or phenomenon that is being modelled consists of a great number of microscopic actors. This creates impracticality for the usage of deterministic models, primarily due to two reasons. Firstly it is computationally taxing to model the interactions between the different actors of the system. Secondly the long-term behaviour of the model might not accurately represent reality since in practice it is near impossible to attain perfect information about all the actors in the system. This is what causes difficulties with predicting complex systems such as the weather and fluids flow. One approach to circumvent the issue is by using a non-deterministic model which utilizes uncertainty. This allows predictions of the aggregate of the system instead of modeling its individual constituents which might have seemingly random behaviour. This paper will be comparing two algorithms for pricing European options: a problem within the field of financial engineering. The two algorithms are Monte Carlo and the Quantum Amplitude Estimation (QAE), both of which utilize uncertainty in order to model the expected payoff of the options contract. The Monte Carlo method is a widely used technique and is commonly used for modelling in finance and nature. It relies on random sampling to estimate an unknown quantity. On the other hand, the Quantum Amplitude Estimation algorithm is a relatively new approach that utilizes the principles of quantum mechanics to perform calculations which are computationally taxing for classical computers. The performance of each method will be investigated to determine which algorithm is more efficient for accurately pricing European options.

1.1 Aim

Our aim is to investigate the accuracy of two different methods for pricing European option contracts, namely the methods of Quantum Amplitude Estimation and Monte Carlo Integration. The error will be measured as the relative error between the estimation and the Black-Scholes formula for European call options which is a widely used model for option pricing.

1.2 Research Question

How does the error of the pricing given by the Monte Carlo and Quantum Amplitude Estimation algorithms respectively depend on the number of samples taken?

1.2.1 Hypothesis

The error terms of the Monte-Carlo and the Quantum Amplitude Estimation Algorithms are in $O(\frac{1}{\sqrt{n}})$ and $O(\frac{1}{n})$ respectively, where n is the number of samples taken. Here the Big O-notation denotes that the error of the estimates decrease at the same speed as the functions $\frac{1}{\sqrt{n}}$ and $\frac{1}{n}$ up to a constant factor. These are the results that have been obtained in other investigations [1].

2 Background

In section we present the relevant concepts and information required to understand the undertaking of the investigation.

2.1 European options contracts

An options contract is a contract that enables the holder of the contract to buy or sell underlying assets at a predetermined price called the strike price at a specified time in the future. A traditional options contract can be classified as either a put or a call option. For an investor to profit from a call option, the price of the underlying asset must have had a sufficient increase in price, whereas to profit from a put option, a sufficient decrease in price is required to profit. A European options contract is a type of options contract that can only be executed at the date of expiry. Meaning that even if the underlying asset experiences a price change prior to maturity, a put or call action cannot be executed until the option has expired. For the purposes of this project, only European Call Options will be considered, but the general principles used are applicable to both types of contracts. [2]

2.2 Defining the Problem

Consider a European call option on an underlying asset with strike price K and a spot maturity price S_T, where T is the expiry date of the contract. Then the value V_T of the contract at time T is defined as

$$V_T = \max(S_T - K, 0).$$

Given that S_T is a random variable that follows a given probability distribution, the fair option price at time 0 (when the contract is formed) will be priced at [2]

$$V_0 = \mathbb{E}[V_T].$$

where $\mathbb{E}[X]$ is the expected value of a random variable X. For the purpose of this investigation, we will consider European call options for underlying assets that pay no dividends. This is done by using the Black-Scholes model.

2.3 Black-Scholes Call Option Pricing Model

The Black-Scholes option pricing model is a widely used method to price option contracts in financial markets. The model is based on the Stochastic Differential Equation and relies on a variety of assumptions, such as the options price following a log normal distribution [3]. The Black-Scholes formula for European call options on underlying assets paying no dividends is [4]

$$c_{BS} = SN(d_1) - Ke^{-r\tau}N(d_2)$$

where

$$d_1 = \frac{\log(S/K) + (r + \frac{\sigma^2}{2}\tau)}{\sigma\sqrt{\tau}}$$

and

$$d_2 = d_1 - \sigma\sqrt{\tau}.$$

The variables used are defined as

1. c_{BS} is the theoretical price of a European call option on a stock,
2. S is the stock price,
3. K is the striking price,
4. r is the continuously compounded rate of interest,
5. τ is the time to option expiration,
6. σ is the volatility (standard deviation of the instantaneous rate of return on the stock),
7. N$(\cdot)$ is the cumulative standard normal density function.

2.4 Introduction to Monte Carlo Methods

Monte Carlo algorithms is a class of algorithms that utilize random sampling to estimate a quantity. This is usually done by designing a probability distribution such that the expected value of a random variable is the desired quantity. Let us introduce a classic example of an application of the Monte Carlo, namely estimating the value of π. This is done by using the fact that the area of a circle is $A = \pi r^2$ where r and A is the radius and the area of a circle respectively.

1. A random sample p_i from points in $[-0.5,0.5]^2$ is generated by a random number generator with a uniform probability distribution.
2. We check if p_i is within a circle centered at $(0, 0)$ with diameter 1
3. We keep track of the number of points which are within the circle and outside the circle. On average the proportion of points that land inside the circle is the same as the ratio between the area of the circle and the square that inscribes it.
4. We use the fact that on average $\frac{PointsInsideCircle}{NumberOfSamples} = \pi \cdot 0.5^2$.
5. Thus we can print $4 \cdot \frac{PointsInsideCircle}{NumberOfSamples}$

Algorithm 1 Monte-Carlo for estimating π

We set the number of samples we want to take
$NumberOfSamples \leftarrow input()$.
We introduce a counter variable to keep track of the number of points land inside the circle.
$PointsInsideCircle \leftarrow 0$
for $i = 1$ to $NumberOfSamples$ **do**
    $x_{cor} \leftarrow random\_float(-0.5, 0.5)$
    $y_{cor} \leftarrow random\_float(-0.5, 0.5)$

8: $distance \leftarrow \sqrt{x_{cor}^2 + y_{cor}^2}$
9: **if** $distance > 0.5$ **then**
10: $PointsOutsideCircle \leftarrow PointsOutsideCircle + 1$
11: **end if**
12: **end for**
13: **Print** $4 \cdot \frac{PointsInsideCircle}{NumberOfSamples}$

Running this code for an increasing number of samples yield an increasingly accurate estimate for π at the expense of an increased runtime. The Monte Carlo algorithm used to model the option price uses the same underlying principles as the π estimation algorithm

2.4.1 Monte Carlo Integration Algorithm

We make use of Monte-Carlo integration in order estimate $\mathbb{E}[V_T]$. Monte-Carlo integration is a numerical technique for calculating integrals by sampling a set of random numbers from a probability distribution. The problem can be transformed into a problem involving solving an integral. This is because $\mathbb{E}[V_T] = \int_\Omega V_T \, d\mathbb{P}$ where Ω is the set of outcomes and $\mathbb{P}$ is the probability measure. Estimating this integral is equivalent to estimating the expected value of V_T, which can be done by Monte-Carlo integration. [5] For our purposes we will be integrating over a log normal distribution, namely

$$P(S_T) = \frac{1}{S_T \sigma \sqrt{2\pi T}} e^{-\frac{(\ln S_T - \mu)^2}{2\sigma^2 T}}$$

where $\mu = (r - 0.5\sigma^2) + \ln S_0$.

This distribution comes from the Black-Scholes model we mentioned earlier [1]. This function is non-trivial to integrate hence using numerical techniques might be a good approach.

Algorithm 2 Monte Carlo Integration for a Log Normal Distribution

1: Here we create a payoff function and take the relevant inputs
2: $k \leftarrow input()$ Strike price
3: **function** PAYOFF(x)
4: return $\max(x - K, 0)$
5: **end function**
6: $NumberOfSamples \leftarrow input()$.
7: $\sigma \leftarrow input()$ volatility
8: $Maturity \leftarrow input()$ expiration date
9: $r \leftarrow input()$
10: $S_0 \leftarrow input()$
11: $Mean \leftarrow (r - 0.5\sigma^2) + \ln S_0$,
12: $Low \leftarrow input()$, $High \leftarrow input()$
13: Here the Monte Carlo Algorithm starts
14: **for** $i = 1$ to $NumberOfSamples$ **do**
15: $V \leftarrow LogGaussian.random\_float(Low, High, Mean, Maturity, \sigma)$

16: $Sum \leftarrow Sum + \text{Payoff}(V) \cdot P(V)$
17: **end for**
18: **Print** $\frac{Sum \cdot (High - Low)}{NumberOfSamples}$

This algorithm yields increasingly accurate results for the value of the integral $\int_a^b \text{Payoff}(X)\, d\mathbb{P}$ which is equivalent to estimating the $\mathbb{E}[\text{Payoff}(X)]$ where X is a random variable that follows the Gaussian distribution.

2.5 A Quantum Approach to Option Pricing

In quantum physics, the position of a quantum particle is described by a wave function $\psi(x)$ given by the Schrödinger equation [6]. $\psi(x)$ is essentially a probability density function which describes the probability of the particle being measured having position x. Applying the so called position operator on $\psi(x)$ yields $\hat{x} = \mathbb{E}[\psi(x)]$. Here already we can see how this is relevant to the problem we are trying to solve. Quantum computing is a type of computing that makes use of these properties of quantum mechanics in order to perform computational tasks. Quantum computers are made up of quantum circuits which can perform operations on qubits; bits which can be either 1, 0 or a superposition of both according to probability distribution. For certain types of problems quantum computers can have an advantage over its classical counterpart due to the fact that they both operate according to separate principles. This is especially true for problems involving probabilities as quantum mechanics intrinsically operates according to statistical laws. In the following sections we lay out in further detail how quantum computers work and how they can be used to solve the problem of option pricing.

2.6 Basic Quantum Information Theory

As previously mentioned the main unit of computation for a quantum computer is a qubit. A qubit as opposed to a regular bit which can be in either an on or off state (typically represented by a 1 or 0) has a continuum of possible states each with an associated probability of being a 1 or 0. This can be represented by a so called statevector. First we can consider the simplest possible qubits, a qubit that is 0 (off) 100% of the time and a qubit that is 1 (on) 100% of the time. Mathematically these two qubits can be represented by two orthogonal ket vectors [7] $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$, $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$. Here each entry encodes the probability that the qubit is in either the on or off state. Since $|0\rangle$ and $|1\rangle$ form an orthogonal basis space any 2D vector can be expressed as a combination of the two, e.g. $|q\rangle_0 = \frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. To understand what this more complex qubit means we must apply the following rule. To find the probability of measuring the qubit $|\phi\rangle$ in state $|x\rangle$ we use that $p(|x\rangle) = \langle x|\phi\rangle^2$. Here $\langle\cdot|\cdot\rangle$ denotes the inner product of the bra and the ket vector. [8]

Another important property of qubits is the observer effect. We know that the amplitudes contains information about the probabilities that a qubit is in a certain state. But once it has been measured that uncertainty disappears and becomes a classical 0 or 1

bit. In order to make use of the quantum properties of qubits, measurements are usually made multiple times at the end of a program (more specifically the quantum circuit) in order to find information about the final statevector.

2.7 Quantum Circuits and Operators

In computer science a circuit is a computational model that can perform operations (usually through so called gates) on information flowing through wires. Classical computers traditionally make use of so called boolean circuits. A Boolean Circuit is a circuit which can only perform boolean operations, i.e operations on information in states 0 or 1. Some examples of these operators are the Not, And, Or operators (these are sometimes referred to as bitwise operations) displayed in the table below [9].

a	$\neg a$
0	1
1	0

ab	$a \wedge b$
00	0
01	0
10	0
11	1

ab	$a \vee b$
00	0
01	1
10	1
11	1

Figure 1: Tables displaying the Not, And, and Or bitwise operations.

There are many different types of circuits but for our purpose we will mainly focus on quantum circuits, as they are used in quantum computers. In a quantum circuit the wires are represented by qubits and the gates are operations that are applied to the qubits in the circuit. The operations are often represented in the form of matrix transformations since qubits are usually represented as vectors. The some common types of gates used in quantum circuits are Hadamara, CNOT and Pauli-gates. The individual gates perform an operation on the qubit going through the gate by modifying its statevector. [9] Although we cannot explicitly see these changes to the qubits, we can reasonably approximate the statevector by doing controlled measurements of the qubit and analysing the frequency of each possible collapsed state.

The specifics of each type of quantum gate is mostly beyond the scope of this project as it predominantly occurs on a lower level of the hardware. The important thing to keep in mind is that these logic gates can be combined in intricate ways in order to construct more complicated structures or operations such as an inverse quantum Fourier transform or loading a probability distribution.

2.8 Quantum Amplitude Estimation

As previously mentioned the amplitudes of qubits contain information about the probability of each of its collapsed states being measured. QAE is a method used to solve the following problem.

2.8.1 Amplitude Estimation

Given a unitary operator $\mathcal{A}$ such that $\mathcal{A}|0\rangle = \sqrt{a}\,|\phi_0\rangle_n\,|0\rangle + \sqrt{1-a}\,|\phi_1\rangle_n\,|1\rangle$ for normalised states $|\phi_0\rangle_n$ and $|\phi_1\rangle_n$ where $a \in [0,1]$ and is unknown. QAE is used to estimate the probability a which is done by repeated iterations of a certain oracle Q. The oracle Q is defined as $Q = \mathcal{A}S_0\mathcal{A}^\dagger S_{\phi_0}$ where $\mathcal{A}^\dagger$ denotes the complex transpose of $\mathcal{A}$, $S_0 = 1 - 2\,|0\rangle\,|0\rangle\,\langle 0|\,\langle 0|$ and $S_{\phi_0} = 1 - 2\,|\phi_0\rangle\,|0\rangle\,\langle\phi_0|\,\langle 0|$. This represents a $2\theta_a$ rotation around the two-dimensional space spanned by $|\phi_0\rangle_n\,|0\rangle$ and $|\phi_1\rangle_n\,|1\rangle$. From the definition of $\mathcal{A}$ we see that $a = \sin^2(\theta_a)$. In order to solve the problem we need to attain an approximation for θ_a which can be done by an algorithm such as Quantum Phase Estimation. Phase estimation approximates certain eigenvalues for Q which is then mapped to an estimator for a. To do this it uses m sampling qubits to represent results and $M = 2^m$ applications of Q. The m qubits are initialized by harmada gates which brings them to equal superpositions. This is done in order to make control variables for the different powers of Q. After this, a quantum inverse fourier transform is applied and a measurement is made to the qubit resulting in an integer $y \in \{1, 2, \ldots, M-1\}$. This integer is mapped to an estimator for $\hat{a} = \sin^2\left(\frac{\pi y}{M}\right)$. [5]

This estimator satisfies the relation

$$|a - \hat{a}| \leq \frac{\pi}{M} + \frac{\pi}{M^2}$$

with a probability of at least $\frac{8}{\pi^2}$. Thus we have that the error for the estimator $\hat{a}$ shrinks in $O(\frac{1}{n})$ which is a quadratic speed-up over the MonteCarlo method.

Another way of doing this without the use of quantum phase estimation is to make use of the fact that $Q^k\mathcal{A}|0\rangle = \cos((2k+1)\,\theta_a)\,|0\rangle + \sin((2k+1)\,\theta_a)\,|1\rangle$. Now a maximum likely hood estimation can be used in order to estimate θ_a which is done by simply measuring the superposition repetitively and doing a frequency analysis. [1]

2.8.2 Distribution loading

In order to price the options contract on a quantum computer we have to load the relevant Black-Scholes log normal distribution. This is done by discrediting the distribution into 2^n grid points. To load this into a qubit we apply a unitary operator that does the following mapping:

$$|0\rangle_n \mapsto |\phi\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i}\,|i\rangle_n$$

where each p_i is associated with the probability $|i\rangle_n$ being measured i.e. the probability associated with the i-th grid point in the discretized log normal distribution. i is mapped to the right interval by the following mapping:

$$i \in \{1, 2, \ldots, 2^n - 1\} \mapsto \frac{low - high}{2^n - 1} + low$$

Hence measuring the qubit $|\phi\rangle_n$ is equivalent to sampling the desired distribution which aligns with our aim of modeling the problem with quantum system.

2.8.3 Calculating the expected pay-off

Once we have an approximation for θ_a we need to calculate the expected value of the payoff f. To do this we define a new operator $\mathcal{A}$ such that $a = \mathbb{E}[f(S)]$ where S is a random variable taken from the probability distribution given by Black-Scholes that is loaded on the quantum registers. We define this new $\mathcal{A}$ as

$$\mathcal{A} = \left(\sum_{i=0}^{2^n-1} \sqrt{1 - f(S_i)}\sqrt{p_i}\,|S_i\rangle\right)|0\rangle + \left(\sum_{i=0}^{2^n-1} \sqrt{f(S_i)}\sqrt{p_i}\,|S_i\rangle\right)|1\rangle$$

so then $a = \sum_{i=0}^{2^n-1} f(S_i)p_i = \mathbb{E}[f(S)]$. Thus by QAE we get the expected value of the pay-off $f(s)$.

The issue is that applying the QAE requires the operator F (corresponding to the payoff function). In general, representing F is a costly endeavour, requiring $O(2^n)$ gates or many additional ancilla bits for pre-computation. To overcome this hurdle we make use of the method outlined in the article Quantum Risk Analysis. [1] This method makes use of the fact that an operator mapping $|x\rangle|0\rangle \longrightarrow |x\rangle(\cos\zeta(x)|0\rangle + \sin\zeta(x)|1\rangle)$ where $\zeta(x)$ is a polynomial of degree k, can be efficiently constructed using so called multi-controlled Y-rotations. These are single qubit operations which can be done with only $O(n)$ gates and $O(n)$ ancilla bits, both of which scale more sustainably than the naive approach for representing the pay-off function f. We now try to find a polynomial such that $\sin^2\zeta(y) = y$ so that we can set $y = f(x)$. Since $f(x)$ is a piecewise linear function we can try to estimate the linear part by estimating $\mathbb{E}\left[c\left(f(X) - \frac{1}{2}\right) - \frac{1}{2}\right]$ for $c \in [0, 1]$ instead of $\mathbb{E}[f(x)]$. The motivation behind this is that $\sin^2\left(y + \frac{\pi}{4}\right) = y + \frac{1}{2} + O(y^3)$. So we want to find a polynomial $\zeta(x)$ such that $c\left(y - \frac{1}{2}\right) + \frac{1}{2}$ is sufficiently well approximated by $\sin^2\left(c \cdot \zeta(x) + \frac{\pi}{4}\right)$. As mentioned in the article, setting the two terms equal we get that $\zeta(x) = \frac{1}{c}\left(\arcsin\left(\sqrt{c\left(y - \frac{1}{2}\right) + \frac{1}{2}}\right) - \frac{\pi}{4}\right)$. We can now apply a Taylor approximation around $y = \frac{1}{2}$ to approximate $\zeta(x)$.

3 Methodology

The methodology is split up into 2 parts.

1. **Data Generation:** An input data set will be generated by a python script under given restrictions. This is explained in further detail in the next subsection
2. **Data Processing:** The two algorithms will process the input data and output their respective estimations for that data. The output data will then be compared to the option pricing given by the analytic solution of the Black-Scholes model, allowing us to determine the accuracy of the two estimations.

3.1 Data Generation

Both algorithms accept 5 arguments from the user which are needed in order to use the Black-Scholes model. These arguments are:

1. S is the stock price,
2. k is the striking price,
3. r is the continuously compounded rate of interest (in percent),
4. τ is the time to option expiration (in days),
5. σ is the volatility (standard deviation of the instantaneous rate of return on the stock),

5 Groups of random test data of these arguments was generated by python script, each of which slightly modified the restrictions of each of the inputs. In general these groups have the following imposed restrictions:

1. $\tau \in [14, 365]$
2. $S \in [0.01, 100]$
3. $\sigma \in [0.1, 20]$
4. $r \in [0.01, 4]$
5. $k \in [0.01, S]$

and for each specific group we make the following alteration to the data generation.

1. **Short Maturity:** $\tau \in [1, 14]$, ceteris paribus.
2. **Low Interest Rate:** $r \in [0.01, 0.25]$, ceteris paribus.
3. **Low Volatility:** $\sigma \in [0.1, 5]$, ceteris paribus.
4. **High Interest Rate:** $r \in [3, 8]$, ceteris paribus.
5. **High Volatility:** $\sigma \in [20, 70]$, ceteris paribus.

3.2 Data Processing

After these parameters have been generated and imported into their respective CSV input file, we run the inputs through the two algorithms and generate a CSV output file for each input file. The output file contains the following:

1. the Black-Scholes theoretical pricing of the stock option
2. a list of estimates where the i-th estimate is the estimate attained after i samples
3. the corresponding list of differences between the estimates and the answer where the i-th difference is the difference between the answer provided by the Black-Scholes model and the i-th estimate
4. the list of percentage differences between the estimate and the theoretical pricing

Each set of test data was also run multiple times in order to get a better estimate of the expected error.

4 Results

After the data gathered from running the algorithms had been collected we calculated the Pearson correlation coefficient which determines the strength of the correlation on a scale from -1 to 1. We also calculated the t-value (which is dependent on the correlation coefficient) to use it and the degrees of freedom in order to calculate the p-value of the correlation. This was done for each data set for both algorithms, the results of which are displayed below in the following tables:

Table I: Displays the results attained from running 10000 samples 100 times over the different groups of test data. The error in the table is the average error after 10000 samples. The Pearson correlation was used to test whether the percentage error was correlated with the function $\frac{1}{\sqrt{(n)}}$.

Monte Carlo Performance					
	Short Maturity	**Low Interest Rate**	**Low Volatility**	**High Interest Rate**	**High Volatility**
Error (%)	2.51%	1.90%	1.72%	3.07%	17.6%
Pearson (r)	0.759	0.697	0.773	0.457	0.265
t-value	117	97.1	122	51.4	27.4
p-value	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$

Table II: Displays the results from running the amplitude estimation. Running the algorithm on the different groups of test data yielded 10 data points ranging from 25 to 250 samples taken and utilizing 5 qubits. The Pearson correlation was used to test whether the percentage error was correlated with the function $\frac{1}{n}$.

Quantum Amplitude Estimation Performance					
	Short Maturity	**Low Interest Rate**	**Low Volatility**	**High Interest Rate**	**High Volatility**
Error (%)	32.7%	11.1%	393%	2735%	263%
Pearson (r)	0.034	−0.086	0.062	−0.057	0.021
t-value	0.02	0.03	0.07	0.19	0.13
p-value	> 0.05	> 0.05	> 0.05	> 0.05	> 0.05

5 Discussion

In order to interpret the result we must first analyse how the statistical tools used in the results section actually work and what it tells us about the potential relationship between the error and the number of samples taken.

5.1 Statistical analysis

In order investigate the relationship between the error and the number samples for the respective algorithms we employ a Pearson correlation test. This test is used to measure the linear correlation between two different sets of data. But in our case we're not aiming to determine whether a linear correlation exists or not. We want to know if the error is inversely proportional to the square root of the number of samples, or simply inversely proportional for Monte/-Carlo and Quantum Amplitude Estimation respectively. To test for this type of correlation with a Pearson correlation test we apply a mapping on the error data in order to map the correlation function to a linear correlation function. To do this we let $M(n)$ and $Q(n)$ be functions for the expected error after n samples have been taken. Suppose that $M(n) \sim O(\frac{1}{\sqrt{n}})$ and $Q(n) \sim O(\frac{1}{\sqrt{n}})$. We can then apply the following mapping:

$$Q(n) \mapsto q(n) = Q(n)^{-1}$$

$$M(n) \mapsto m(n) = M(n)^{-2}$$

so $m(n) \sim O(n)$ and $q(n) \sim O(n)$. But this is a linear relationship which can be tested for with the Pearson correlation test. Which is what is done in the results section. By convention the significance level is a p-value of 0.05. So from our results we can infer that there is a significant correlation between the error of the estimates for the Monte Carlo algorithm across all test groups, whereas for Quantum Amplitude Estimation we cannot conclude that error correlate the number of samples with the function $\frac{1}{n}$.

5.2 Sources of Error

In this section we explore the possible sources of error in this investigation. Since these algorithms are based on different types of hardware and software they might also be impacted by different factors.

5.2.1 Monte Carlo

As seen in Figure 2, the algorithm quickly approaches a threshold value, whereafter the accuracy of the estimation seems to plateau. This is contrary to what was expected as theoretically the error should approach 0. This could be due to the accuracy of the calculations done by the program. In order to save memory and save runtime programming languages often approximate functions which otherwise can be difficult to compute. Some examples of these functions is the e^x, $\ln x$ and $\sqrt{x}$. None of these functions are easy to compute but getting an accurate approximation can be done quite easily.

All of these functions are present in the probability density function and could possibly contribute to increasing the error. This has a greater effect when the data is more spread out as smaller quantities account for a larger part of the expected value. This is also what we see from Table I, the data set with the greatest spread (high volatility) has the highest and the data set with the lowest spread (low volatility) has the lowest error. One possible way to remedy this is by using less significant figures in the inputs. It is also possible that using an non-builtin implementations of special functions such as $\ln x$ that has better accuracy might also decrease the error.

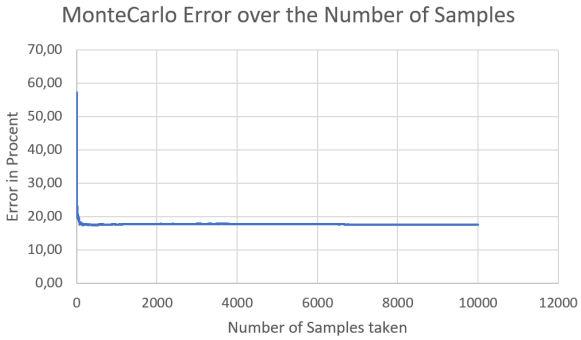

Figure 2: Graph displaying the average error (in percent, y-axis) of the estimates yielded after the Monte Carlo algorithm has taken some amount of samples (x-axis). This graph was made from the results from the high volatility test data.

5.2.2 Quantum Amplitude Estimation

Quantum Amplitude Estimation is partly affected by the same issues as the Monte Carlo algorithm as some parts of the code is run on classical hardware. But quantum computer are also affected by other factors that classical computers are not. One stark issue with the use of quantum computers is the accessibility. Quantum computers are currently in their infancy and they are quite costly and difficult to produce. In order to gain access to quantum hardware the IBM Qiskit python library was used. Qiskit is a library which can write code for quantum computers and send requests to IBM's quantum computers and perform the desired operations. Since there is a limited number of quantum computers IBM utilize a queue system which gives certain requests priorities over others. This makes it difficult for non-researches to use the get a lot of data points as it might take hours just to get a single data point which makes it harder to demonstrate a correlation. Furthermore, there is more demand for quantum computer of better quality i.e. with a larger number of qubits. In order to decrease the amount of queuing, we used a modest amount of qubits, namely 5. This discretized our probability distribution into 32 intervals. In some cases this can cause error in the pricing because the loaded distribution can differ significantly from the desired distribution. This could explain the obtained results as for some groups of test data the estimate after many samples. This could indicate that the algorithm isn't working as intended and doesn't approximate the expected payoff correctly. Having the wrong distribution loaded onto the quantum circuit would explain this behaviour. A way of investigating if this is a source of error is by running Monte Carlo on the discretized distribution and observing if it displays similar behaviour in its results.

Another thing to make note of is that qubits aren't perfectly reliable. Noise inherently account for some of the error but it isn't sufficient to explain the results [5] as they should have less than 10% error depending on the hardware that was used. To investigate how much the noise could have accounted for the error one would have to test the performance of the qubits of the hardware that was used. Of course there could also have been a software bug which yielded these unexpected results. But seeing as most of the code for the quantum hardware are built-in methods of the Qiskit library which is commonly used by other researchers, it would be unlikely for a bug to go unnoticed and not patched for such a long time.

6 Conclusion

In conclusion we have statistically significant evidence for the error of the Monte Carlo algorithm scaling as $\frac{1}{\sqrt{n}}$, although the error of the estimation doesn't seem to approach 0 as expected. This could be due to the way python computes the functions $\ln x$, e^x and $\sqrt{x}$. Furthermore, floating point errors could also account for some of the error especially when the probability distribution is more spread out. This is also what is observed in our results. On the other hand we cannot reject the null hypothesis for the error of the Quantum Estimation Algorithm, since there was no correlation between the error and the function $\frac{1}{n}$. This could be because the number of qubits couldn't load an accurate approximation of the probability distribution. A way to check whether or

not this affected the results is by running Monte Carlo on the loaded distribution and seeing if it yields similar results. lastly, testing the noisiness of the qubits would help check if other factors were involved. For future investigations these possible remedies should be considered being implemented in order to reduce sources of error and better evaluate the performance of the Quantum Amplitude Estimation algorithm.

References

[1] S. Woerner and D. J. Egger, "Quantum risk analysis," *npj Quantum Information*, vol. 5, no. 1, p. 15, Feb 2019. [Online]. Available: https://doi.org/10.1038/s41534-019-0130-6

[2] H. Föllmer and M. Schweizer, "Hedging by sequential regression: an introduction to the mathematics of option trading," *ASTIN Bulletin: The Journal of the IAA*, vol. 19, no. S1, p. 29–42, 1989.

[3] A. Shinde and K. Takale, "Study of black-scholes model and its applications," *Procedia Engineering*, vol. 38, p. 270–279, 2012.

[4] P. Pianca, "Simple formulas to option pricing and hedging in the black- scholes model," 12 2005.

[5] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner, "Option Pricing using Quantum Computers," *Quantum*, vol. 4, p. 291, Jul. 2020. [Online]. Available: https://doi.org/10.22331/q-2020-07-06-291

[6] R. Alphonce, L. Bergström, P. Gunnvald, E. Johansson, and R. Nilsson, *Heureka Fysik 3 Lärobok*. Natur Kultur Läromedel, 2013.

[7] "Single systems," https://learn.qiskit.org/course/basics/single-systems, accessed: 2023-05-20.

[8] "Multiple systems," https://learn.qiskit.org/course/basics/multiple-systems#multiple-systems-quantum-states, accessed: 2023-05-20.

[9] "Quantum circuits," https://learn.qiskit.org/course/basics/quantum-circuits, accessed: 2023-05-20.