Quantum computing in Finance: Pricing of European and American options

Alessio Espa

June 27, 2023

1 Introduction

This report aims to discuss the problems dealt with and the results obtained within the first two months of the internship carried out at Paris Saclay as part of the group MATHRISK, under the supervision of Prof. Goudenège and Prof. Girolami. The structure of the report is outlined in the following.

First, an overview of the subject matter is provided, including some background information needed for later sections, such as the financial concepts underlying the algorithms being examined. Then the problem is presented, along with an outline of some usual classical approaches that have been used to tackle it.

After that, the report turns to some notions of quantum computing deemed necessary to understand the description of the quantum algorithms. Indeed, the following section explores the codes written to price two common types of financial instruments, and is followed by an analysis of the results that they provide.

Finally, a section on the conclusions drawn from the results is reported, together with some comments on what the next steps could be and a mention of the future work being carried out in the remaining time of the internship (this report contains only the results of the first two months of work).

2 Overview

This section provides a quick overview of the financial concepts underlying the present work. [1] [8]

The main focus of the work is on option pricing. An option is a financial instrument that allows the buyer to buy or sell a specific asset, subject to certain conditions, within a specified timespan.

Several types of options exist. The main distinction on its nature is whether it is a call or a put option, i.e., whether it gives the investor the right to buy or to sell the option at a later time at a fixed price ("strike price"). The last day on which the option can be exercised is called "maturity". Given a strike and an asset price, it is possible to compute the "payoff", i.e. how much money could be made at a certain time under certain conditions. Of course, the value of the option is dependent on that of the asset it is connected to, and investors will clearly choose the kind of options based on whether they believe the price of the asset will increase or decrease in the future.

The other difference depends on when the option may be exercised. The main options taken into consideration are European, American and Asian. The first can only be exercised at a set time ("maturity"), the second one at any time before maturity and the third one depends on an average of set past values. Another kind of option is the Bermudan, which can only be exercised at set dates before the maturity (and that from now on will be used interchangeably with American).

Given the importance that such financial instruments play in the market, it is no surprise that banks are keen to know the expected return of a certain option in the future and thus estimate its fair price today. The focus of this report will be to lay out methods used to price such options. Throughout this article, European and American call options will be discussed (the generalisation to put options is usually straightforward; the key differences will be pointed out). Three main classical methods will be quickly presented, and one will be chosen as a starting model upon which the quantum circuits will be built; a comparison with its classical equivalent will also be later discussed.

3 Classical methods

The main classical methods that will be briefly discussed are the following: Black-Scholes, Binomial tree, Montecarlo simulations.

3.1 Black-Scholes

This subsection is based on [1]. It is one of the best-known and most widely used methods to price European options, even nowadays.

One of the main assumptions of this model is that the underlying asset prices have a log-normal distribution. In addition, the model posits that the option price can be assessed by knowing a few constants specifying the instance of the problem: these inputs are volatility, the price of the underlying asset, the strike price of the option, the time until maturity and the risk-free interest rate, which concerns the return of the option which is not subject to any risk. In the following, the differential equation used in the model (1) and its solution (2,3,4,5) are provided together with the legend.

$$\frac{\partial \mathrm{C}}{\partial \mathrm{t}} + \frac{1}{2}\sigma^2 \mathrm{S}^2 \frac{\partial^2 \mathrm{C}}{\partial \mathrm{C}^2} + \mathrm{rS}\frac{\partial \mathrm{C}}{\partial \mathrm{S}} = \mathrm{rC} \tag{1}$$

$$\mathrm{C}(\mathrm{S}, \mathrm{t}) = \mathrm{N}(\mathrm{d}_1)\mathrm{S} - \mathrm{N}(\mathrm{d}_2)\mathrm{K}\mathrm{e}^{-rt} \tag{2}$$

$$\mathrm{d}_1 = \frac{1}{\sigma\sqrt{\mathrm{t}}}\left[\ln\left(\frac{S}{K}\right) + t\left(r + \frac{\sigma^2}{2}\right)\right] \tag{3}$$

$$\mathrm{d}_2 = \frac{1}{\sigma\sqrt{\mathrm{t}}}\left[\ln\left(\frac{S}{K}\right) + t\left(r - \frac{\sigma^2}{2}\right)\right] \tag{4}$$

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \mathrm{e}^{-\frac{1}{2}z^2} dz \tag{5}$$

S: The spot price of the underlying asset, K: The strike price of the option, r: The risk-free interest rate, σ^2: The variance of the underlying asset, t: Time in years to the date of maturity

This approach may be used to price European options as long as the above assumptions are met; otherwise, the method of choice is usually Montecarlo simulations.

3.2 Montecarlo simulations

The usual classical choice is running many Montecarlo simulations to evolve the price of the asset in time based on some model, then compute the payoff function for each terminal step and eventually estimate the fair option price today through some backward recursion [7]

One of the advantages of this method, which partially accounts for its popularity, is that unlike the previous method it can also be used to price American options by means of some optimisation strategy. While effective, this method lacks efficiency, especially when faced with multidimensional problems (i.e. baskets of options) where the complexity becomes quickly overwhelming.

Such issues are the reason behind the growing appetite for faster and more efficient algorithms that can better handle highly complicated simulations. Quantum computing may provide the answer to that.

3.3 Binomial tree method

This method is popular due to its simplicity and efficacy. [8] It is essentially a model to evolve the stock price through some simple "up" and "down" movements along a tree, which may be further simplified if the tree is chosen to be recombining (i.e., the same price values will be present in the following layers of the tree). Similarly to Black-Scholes, in order to evolve the price some parameters are needed:

- S_0: Initial stock price
- r: Risk-free interest rate
- T: Time to expiration
- N: Number of time steps

from which the following paramenters are inferred:

- $u = e^{(\sigma\sqrt{T/N})}$: Up factor
- $d = e^{(-\sigma\sqrt{T/N})}$, Down factor
- $p = \frac{e^{(rT/N)}-d}{u-d}$, Probability of going up

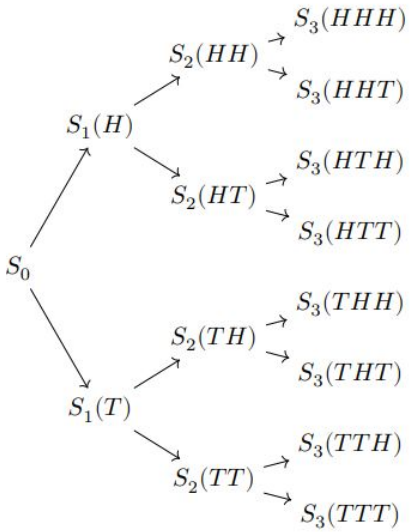

Figure 1: The binomial tree model. Heads(H) or tails (T) stand for up or down movements.

After generating the price tree, similarly to the other methods in this case as well the strategy is to compute the payoff function at maturity and, if the option is American, make use of backward recursion to determine the fair option price.

This concludes the brief outline of commonly used classical strategies. Given the versatility of the binomial tree method and its ability to price both European and American options, it will be chosen to experiment with quantum computation techniques in the following section.

4 Quantum computing

4.1 Key notions: qubits, gates and circuits

The most basic difference between quantum and classical computing lies in the way information is stored. Unlike classical bits, quantum bits ("qubits") can take both binary values 1 and 0 at the same time through a quantum mechanics principle called "superposition".

The core of a quantum code is a quantum circuit. That is, after initialising a certain number of qubits, a set of operations (possibly connecting them and thus acting on multiple qubits) is put in place through "quantum gates", which enforce a range of operations, some of which have their classical counterparts. [2]

Among the most common gates are the following, all unitary gates (it is quite important that they are to be able to run the quantum circuit):

- Hadamard gate H, which creates the superposition state :

$$|\Phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle) \tag{6}$$

  depending on whether the state acted on is $|0\rangle$ or $|1\rangle$.

- Pauli Gates X,Y,Z, which enforce the Pauli matrices. In particular, X swaps the qubit value.

$$\sigma_x = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7}$$

- Controlled gate CNOT, which swaps the value of the second qubit only if the first qubit is in $|1\rangle$ state.

- Rotation gates around x,y or z axis of an angle θ.

$$R_x = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}, R_y = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix},$$

$$\mathrm{R}_z = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix} \tag{8}$$

Once the quantum circuit executing the desired actions has been coded, it is usually required to end the algorithm with a measurement in classical bits of the information encoded in the purely quantum part of the code. Indeed, in order to do so, a simulator is used and the quantum state is collapsed into a classical state that has one specific value the moment its measurement is carried out. This step is normally the final one in a quantum code.

The main factor that should make quantum computing more efficient than classical simulations is that using a relatively small number of qubits

and exploiting their ability to be manipulated through wisely chosen gates, results comparable to classical simulations should be obtained, according to some, with a quadratic speed-up, for instance, in Montecarlo simulations for option pricing tasks [2] [10]. The main challenge, though, is devising such circuits in reasonable times and with a low number of gates to avoid incurring in error sources such as decoherence, which may impact the correct quantum state. As will be pointed out in later sections, this is indeed a core problem to be reckoned with. [9]

5 Quantum option pricing

This report now turns to the actual work aimed at pricing European and American options using quantum computing. The software used for coding is Qiskit [5], IBM Quantum Computing [3]. The first focus has been the easiest option to price, the European one. As anticipated in a previous section, the model of choice for quantum computation has been the recombining binomial tree, known as the Cox-Ross-Rubinstein ("CRR") model.

5.1 European option

The steps used to price a European option using a quantum code are the following:

- 1. Load the probability distribution of the asset connected to the options. Usually a log-normal (Black-Scholes), normal or a binomial distribution.
- 2. Define a payoff function to compute the maximum profit at maturity.
- 3. Compute its expected value; for a European option the argument is simply the difference between price at maturity and strike:

$$E\left[\max\{S_T - K, 0\}\right] \tag{9}$$

Some algorithms are available within Qiskit Finance tutorials as inbuilt functions to load probability distributions common in finance, like log-normal, and other functions to price European options and estimate its value through algorithms like Amplitude Estimation. They will be used in

a later section to compare the computational cost with the algorithm that will be discussed in the following part.

5.1.1 Description of the algorithm

The first focus has been understanding how to build functions similar to those mentioned above tailored to specific purposes. Indeed, behind them lie quantum circuits of varying degree of complexity, and understanding the rationale behind them is of paramount importance in order to create novel circuitry.

Starting from loading the probability distribution, in [4] it is shown that in order to replicate a certain shape rotation gates and CNOT gates are the best choice. The number of qubits affects the speed and accuracy of the algorithm, while the rotation gates are used to change the probabilities of each quantum state and the CNOT gates are needed in order to symmetrise the distribuion so as to finally obtain the desired shape. The choice of gates is based on the principle of "universality", according to which any unitary operation on a quantum system can be constructed by combining CNOT gates and single-qubit gates.

The approach is more involved for asymmetric distributions. There does not seem to be a general intuition behind the construction of a specific distribution. That is, each distribution seems to be the result of specific choices (made easier if the distribution is symmetric) to obtain the target distribution. Nevertheless, as discussed in more detail in the paper, some conditions can be set on the rotation angles for symmetric distributions which make it easier to obtain their shape.

While this is an effective approach when a given distribution is needed, in the algorithm under examination it was chosen to carry out actual simulations and obtain the desired distribution through a quantum circuit. Indeed, as mentioned before, the chosen model is the recombining binomail tree, where of course a binomial distribution of prices is expected at maturity and which can be recovered by propagating the initial price throughout the tree layers down to the last one.

As can be seen in the first portion of the European option pricing section of the attached code, the basic idea to implement the CRR model is first to

encode the probability of going up and down into the root. Rotation gates are used to encode the probability amplitude of going up (the rotation is by a factor $\sin(\theta/2)$ so the angle is chosen accordingly) in the $|0\rangle$ state. The same principle is applied for probability of going down as the probability amplitude of the $|1\rangle$ state.

Indeed, the core idea of the code is that every layer ("timestep") of the tree should be encoded in one qubit, regardless of the number of nodes in that layer; indeed, amplitude encoding can account for all of them. Finally, once the probabilities have been obtained, the aforementioned payoff function needs to be computed.

Therefore, a CNOT gate is applied between adjacent qubits to propagate the probability, and the same two rotation gates are applied to each subsequent qubit for the reasons mentioned above. This completes the propagation of the price down to the leaves of the tree. If run many times, this procedure allows to produce a probability distribution for the spot prices (which, of course, is expected to be binomial due to the very nature of the tree), with an accuracy and complexity that depend on the number of qubits chosen to run the code.

The simulation function is then defined: given the number of time steps we would like our tree to span, the previous function is run and all qubits are measured to keep track of the paths chosen. The simulation is run using Qiskit's tools and the counts are extracted for each state. Every bit corresponds to an up or down movement (i.e.. $|01\rangle$ encodes down-up path).

The rest of the code is mainly concerned with extracting information from the output of the quantum circuit and converting the paths into spot prices at maturity according to the logic of the CRR model.

Next, the focus shifted to attempts at creating a working quantum circuit to simulate the payoff function and compute its expected value.

Given that the option being priced is European, having all the prices at maturity and their probability, the expected value of the fair option price can be obtained with the following formula:

$$option\ price = \exp(-r \cdot T) \cdot \sum_{i=1}^{N} \max\{S_{Ti} - K, 0\} \cdot prob_i \tag{10}$$

where the first exponential is the risk free part being discounted and the rest is the payoff defined in [9]. The results and their comparison with other techniques will be discussed in a later section.

What needs to be borne in mind, though, is that this approach works due to the simplicity of the European option mechanism. When it comes to estimating the option price for the American option, it is indeed much more challenging due to the additional degree of freedom allowing for early exercise of the option, which requires to consider all the paths that lead to a certain option price and to decide at each step whether exercising is profitable or not.

The next step will be to focus on how to overcome such hurdles and whether it is indeed possible to exploit the quantum advantage to create a working code capable of outperforming a classical simulation.

5.2 American option

The code used to price American options involves using the classical idea of the CRR model and implementing through quantum circuits. The core idea is breaking down the binomial tree into entities of three nodes (parent and its children), and compute the maximum in each entity starting from the last layer at maturity and then feed it into the previous node, recursively. While not completely quantum, this code is a quantum implementation of the classical approach.

While no American pricing codes are available in the Qiskit tutorials, a comparison will still be carried out in a later section with the purely classical implementation of the CRR model.

5.2.1 Description of the algorithm

Some preparatory work is needed before defining the actual function used to run the code.

First, given that the code is based on computation of a maximum function, a function is defined to be used whenever a conversion from decimal to binary is needed and to encode the binary number into a quantum circuit by using Hadamard gates (the qubit is flipped depending on whether the digit whose position it corresponds to is 1 or 0).

Next, the core of the code is defined. The function "compare" shown in the attached code takes as input the 3 numbers to be compared (which correspond to the 3 nodes, parent and its 2 children, that make up the binomial tree that is being simulated here).

Again, the appropriate conversion from decimal to binary is carried out using the function previously defined, and special conditions are set in case the layer is the maturity set (in which case the payoff function is European) and to avoid a 0-dimensional quantum circuit.

Next, after determining the number of qubits needed, the quantum circuit comparing the parent node price and the weighted average of the 2 children based on their probability (so the future expected price) is defined and added to the binary encoding of the parent node price.

The tree is then built by propagating the price from the root to the leaves according to the CRR model. Finally, the simulation is carried out and the information extracted to infer whether exercising (if parent node is bigger) or waiting (if average is bigger) would be a better choice to maximise profit.

In order to create the comparison function, the IntegerComparator function from the Qiskit circuit library was used. This function essentially compares the input numbers in their binary representation qubit by qubit using controlled operations and updates the result in ancilla qubits; eventually, the latter encode the desired piece of information which can be extracted.

Having defined all the necessary elements, the simulation is then run to compute the fair option price by using the spot prices previously computed. Starting from maturity, where the European payoff function is applied, the code works its way backwards till the root of the tree, making sure that every time the max between the current spot price (parent) and the expected future value (average of its children) is computed. Finally, the option price is returned, along with the time necessary to carry out the computation.

6 Results

This section discusses the results obtained with the two algorithms explored in Section 5. For both algorithms, an online tool [6] was used to have a reference against which the results may be compared.

Starting from the European pricing code, first the fair option price was computed for some set values of the parameters and for a variable value of the number of steps to check for discrepancies. In the following, the choices have been reported, together with the results. The choice of the parameters has been made to match the values used in the Qiskit tutorial, so as to facilitate a comparison.

- $S_0 = 2.0$ (initial spot price)
- $r = 0.05$ (annual interest rate of 5%)
- $\sigma = 0.4$ (volatility of 40%)
- $T = \frac{40}{365}$ (40 days to maturity)
- $K = 1.896$ (strike price)
- N (number of time steps in the binomial tree)

N	**Tutorial**	**Online: BS,Bin**	**Quantum**	**Classical**
5	0.171 ±0.005	0.170 0.175	0.173 ±0.006	0.167
7	0.171 ±0.005	0.170 0.174	0.172 ±0.005	0.167
11	0.171 ±0.005	0.170 0.171	0.173±0.006	0.170

Table 1: The results obtained for the various European pricing tools: the Qiskit tutorial, the online tool (both Black-Scholes and Binomial tree), the Quantum code and the classical code discussed above. The same amount of digits has been shown for values with no error as well.

A few things may be noticed from the values in table[1]. The values are the results of the average over ten runs, and the error is the discrepancy between largest and smallest values (standard deviation was deemed too small to be significant).

First of all, what stands out is that only quantum algorithms have been shown with their uncertainty. This is due to the fact that while the Quantum algorithms involve to some extent some quantum simulations and probability distributions (which slightly affect the output of the circuits), the classical

algorithm and the classical online tool used as benchmarks are based on exact code, and thus the output is deterministic.

Turning now to the results, the value obtained using the Qiskit tutorial is the same as it is computed using Black Scholes uncertainty model rather than a binomial tree. Assuming that the reference value is the one given by the online tool [6], the values obtained using the Quantum algorithm discussed in Section 5.1 are in line with the reference values for binomial model and always largely within uncertainty; what can be noticed is that the values reported tend to oscillate slighlty, both for quantum and classical algorithm, although the latter converges to the refernce value in the long run as N is increased. For the quantum code the value of N is limited by the number of qubits necessary to run the simulation.

The reason could be that as the number of layers increases, the number of gates is also increased and the errors might also accumulate, leading to diminishing accuracy. The value obtained with the tutorial is instead in line with the one predicted by the BS model of the reference.

The next step is to produce a plot that shows the fair option price as a function of the initial spot price, while keeping the other parameters of the model fixed. Again, the parameters chosen are shown below. The main difference with the parameters previously used is the strike price, changed so as to allow to cover a wide range of starting prices, necessary to produce a the plots discussed below. Parameters like annual interest rate and volatility have only been slightly changed in order to check for major differences, which did not arise.

- $r = 0.2$ (annual interest rate of 20%)
- $\sigma = 0.5$ (volatility of 50%)
- $T = \frac{40}{365}$ (40 days to maturity)
- $K = 60$ (strike price)
- $N = 10$ (number of time steps in the binomial tree)

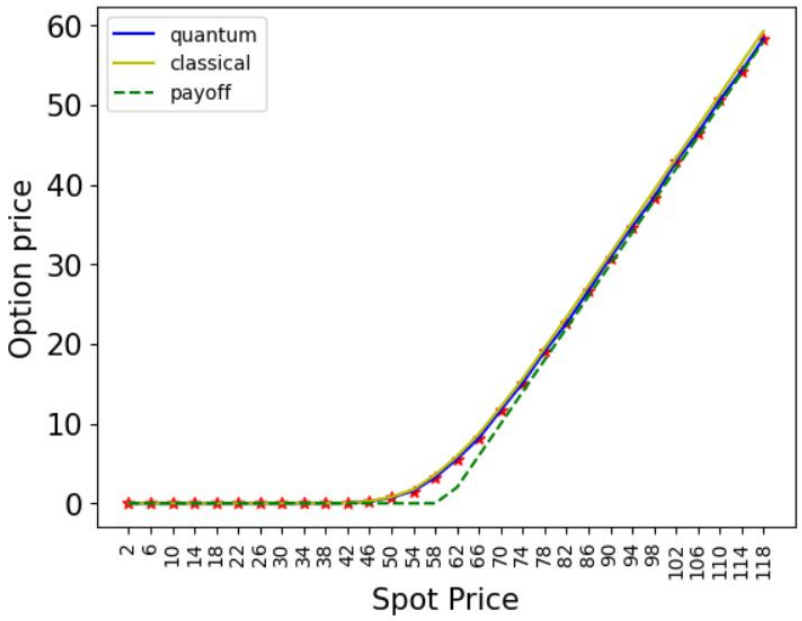

Figure 2: The value of the European call option price as a function of starting spot price, shown for both quantum and classical code for comparison. The payoff function is also reported in dashed line.

In figure 2 and 3, it is shown the option price of the European option (both call and put) as a function of the initial asset price, computed both with the classical and the quantum algorithm; the payoff function is also shown for reference. As can be seen from the figures, the quantum algorithm is quite close to the prediction of the classical one, and the options prices are always above the intrinsic payoff function for call options, while for put they cross the function at some point due to the very definition of its payoff (K and S swap roles in equation 9).

The analysis then turns to the American option pricing. The procedure chosen to show and comment the results is similar to the one used for the European case; the main difference is that no Qiskit tutorial algorithm was available for comparison. The parameters chosen to run the simulation are the same as those shown in Table 1. The results are shown below in Table 2.

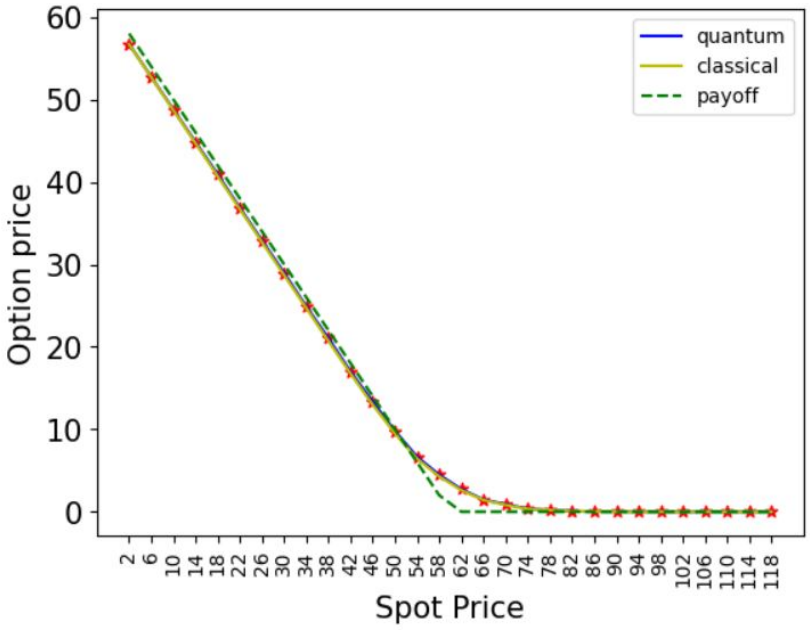

Figure 3: The value of the European put option price as a function of starting spot price, shown for both quantum and classical code for comparison. The payoff function is also reported in dashed line.

First of all, no uncertainty is present on the quantum algorithm unlike before due to the lack of a probability distribution being used in the quantum code which, therefore, lacks stochasticity. This could also be the reason behind the greater similarity between classical and quantum algorithm; both are in line with the online tool results that, as before, are taken as reference values.

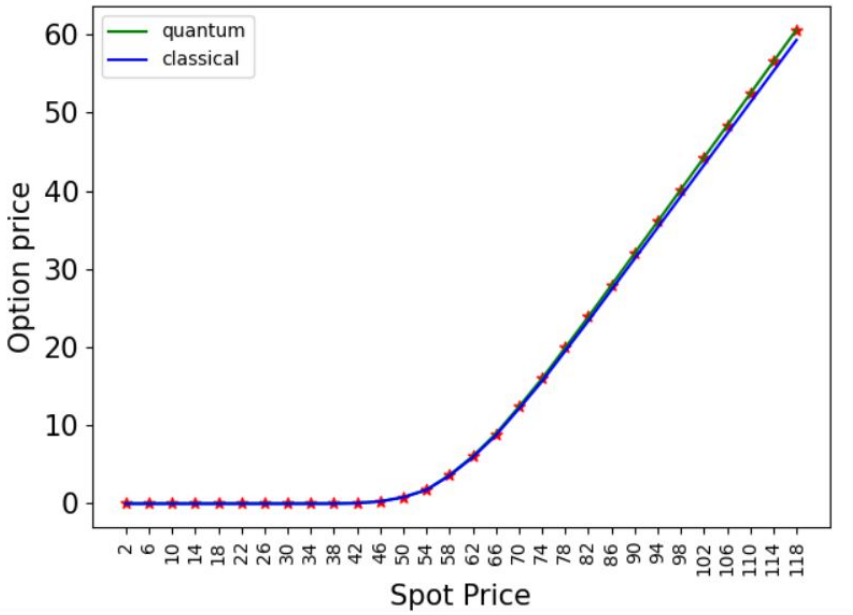

Figure 4: The value of the American call option price as a function of starting spot price, shown for both quantum and classical code for comparison.

N	**Online tool: Bin**	**Quantum algorithm**	**Classical algorithm**
5	0.175	0.168	0.167
7	0.174	0.168	0.167
11	0.171	0.170	0.170

Table 2: The results obtained for the various American pricing tools: the online tool (Binomial tree), the Quantum code and the classical code discussed above. There is no uncertainty as all codes are deterministic. The number of digits shown has been chosen to match those of the results shown in Table 1.

For ease of comparison, the same parameters as those used in figure 2 and 3 were chosen for figure 4 and 5 as well, which show the equivalent plots for American options pricing.

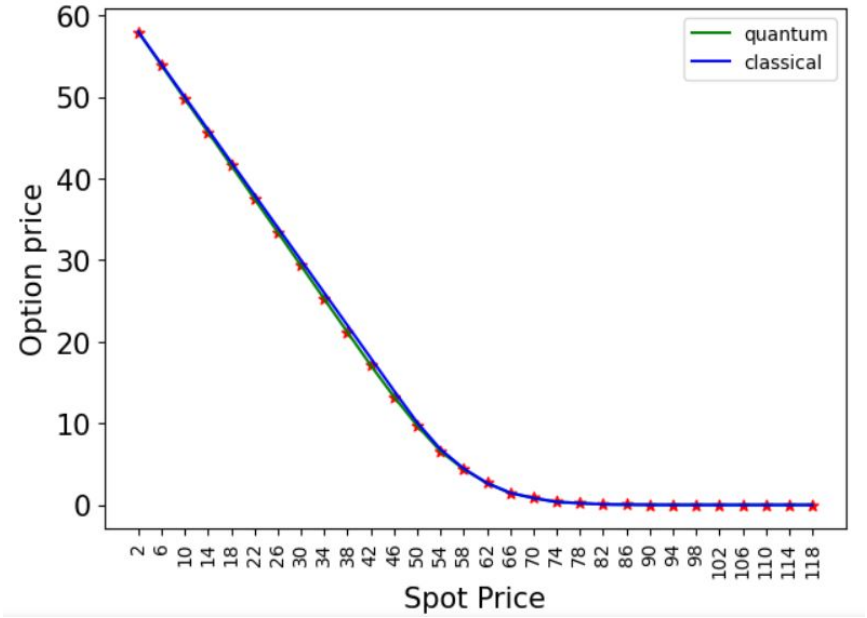

Figure 5: The value of the American put option price as a function of starting spot price, shown for both quantum and classical code for comparison.

As can be seen from the figures, for certain high value of the spot prices there are slight discrepancies between the two algorithms; this is most likely due to the binary encoding of the spot prices, which for big numbers requires many gates and thus affects the accuracy of the output.

Finally, the comparison between the option price for both American and European is shown in the same plot (figure 6 and 7), together with the payoff function, for both call and put.

As expected, the American price is always greater than the European price due to the fact that American options may be exercised early and increase the profit. For both kind of options, as expected, the values are always above (or below for put) the payoff function.

7 Discussion and Conclusions

This final section of the report aims to discuss the conclusions drawn from the results, point out what may be interesting ideas of future research and

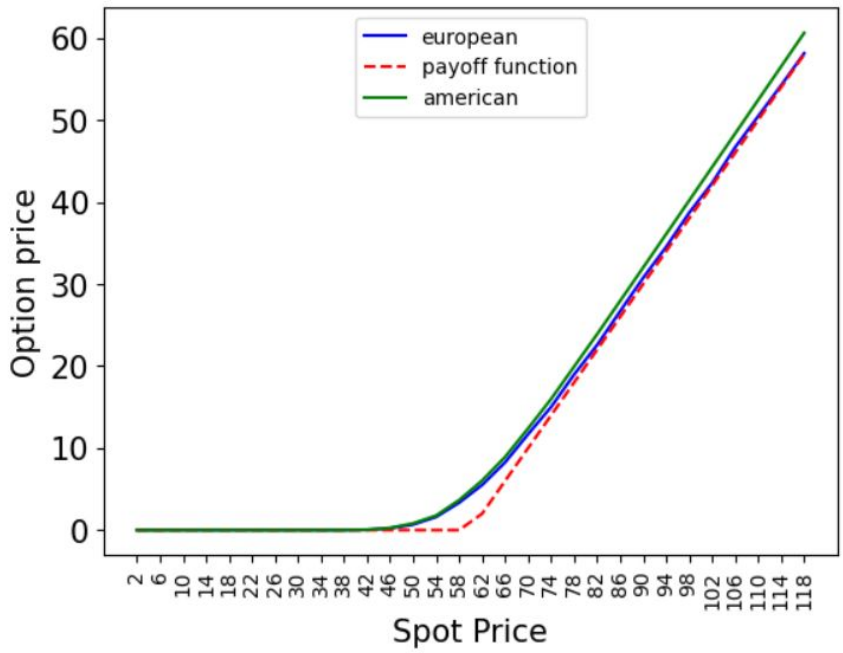

Figure 6: The value of the American and European call option price computed with the two quantum algorithms as a function of starting spot price, with the payoff function shown for comparison.

briefly discusses the content of the future work of the current internship.

In order to check the efficiency of the algorithms discussed in the report, the execution times were computed and reported in Table 3. To do so, the time function available in Python was used and the time recorded from start to end of each run of the code. The values in Table 3 are the results of the average over ten runs.

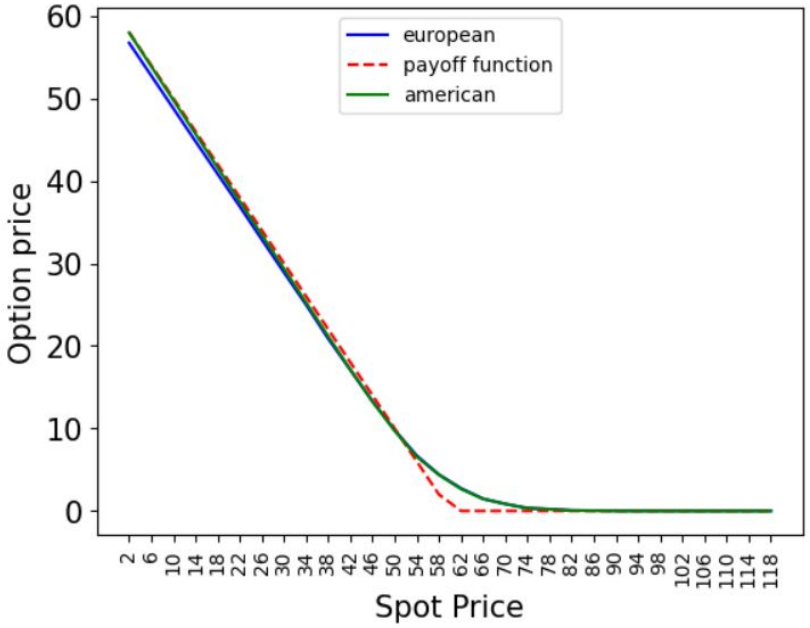

Figure 7: The value of the American and European put option price computed with the two quantum algorithms as a function of starting spot price, with the payoff function shown for comparison.

N	**Tut EUR**	**Class EUR**	**QM EUR**	**Class US**	**QM US**
5	6.2*	0.004	2.1	0.001	3.9
7	6.2*	0.006	2.6	0.002	4.0
11	6.2*	0.008	4.6	0.003	9.7

Table 3: The computation times in seconds for the Qiskit tutorial and the classical and quantum algorithms, both for European and American pricing. *value for just one run, so to be compared has to be multiplied by the number of spot prices used in the other codes, 30, obtaining around 180 seconds

As can be noticed, the classical algorithm for European pricing is approximately 500 times faster than the quantum one, while for American options the factor is of order 10^3 and increases with N as the number of gates increases and slows down the quantum code significantly.

The tutorial provided in Qiskit is considerably slower, about 200 times slower than the quantum pricing code. The reason is that the tutorial makes use of very expensive gates that greatly impact the efficiency of the computation.

As can be easily inferred from the aforementioned conclusions, the classical codes used to price European and American options appear to still be superior to a quantum implementation. One of the main hurdles is that gates are usually computationally expensive and many of them are required to carry out operations that in classical computation are straightforward.

One of the solutions could be to focus more on hybrid codes, i.e. codes where only certain calculations are carried out using quantum circuits while its weaknesses are addressed by classical code.

The main focus of the rest of the internship will be to try to address the shortcomings of the quantum codes created so far to improve the efficiency and make full use of the quantum advantage. One of the ideas could be to try to exploit parallelisation and use forward propagation in the binomial tree rather than the backward propagation used so far.

References

[1] Fischer Black and Myron Scholes. "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81.3 (1973), pp. 637–654.

[2] Daniel J. Egger and Nikitas Stamatopoulos et al. "Option Pricing using Quantum Computers". In: *Quantum 4, 291* (2020). DOI: https://doi.org/10.48550/arXiv.1905.02666.

[3] *IBM quantum computing*. https://quantum-computing.ibm.com/.

[4] Binoy Paine Kalyan Dasgupta. "Loading Probability Distributions in a Quantum circuit". In: (2022). DOI: https://doi.org/10.48550/arXiv.2208.13372.

[5] open source. *Qiskit*. https://qiskit.org/.

[6] *Options Calculator*. http://www.math.columbia.edu/~smirnov/options13.html.

[7] Pracht Rafał. "The Pricing of American Options on the Quantum Computer". In: *SSRN* (2023). DOI: http://dx.doi.org/10.2139/ssrn.4350641.

[8] Steven E. Shreve. *Stochastic calculus for Finance I, The Binomial Asset Pricing Model*. Springer Finance, 2004. ISBN: 9780387401003.

[9] Andrew Steane. "Quantum computing". In: *Reports on Progress in Physics 61 117* (1998). DOI: https://doi.org/10.1088/0034-4885/61/2/002.

[10] Daniel J. Egger Stefan Woerner. "Quantum risk analysis". In: *npj Quantum Information* (2019). DOI: https://doi.org/10.48550/arXiv.1806.06893.