UCLouvain

LOUVAIN
School of Management

Louvain School of Management

Extension of a pedagogical tool in Python to price and hedge options

Author(s): Valentin Paquay
Supervisor(s): Prof. Frédéric Vrins
Academic year 2022.-2023.
Dissertation for the master 120 of Business Engineering
Focus in Financial Engineering
September 2023

Abstract :

Calculating the price of an arithmetic Asian option is not possible using an analytical formula. To obtain its price, it is possible to use the Monte-Carlo simulation method.

This master's thesis develops the methodology applied using the Monte-Carlo method to obtain the price of a derivative. In addition, this thesis aims to demonstrate the advantages of variance reduction methods relative to this simulation.

This work is the support for an application which is intended to illustrate graphically the objective stated above. This app is the continuation of the development of a platform designed to bring together tools that hedge or price derivatives and is the result of another thesis.

The hub of these applications is available here: https://central-hub-app.onrender.com

The application itself is available here: https://mc-asian-app.onrender.com

The application has been designed to be used in a pedagogical context, either by a teacher wishing to illustrate these concepts, or by a user curious to learn more about the use of Monte-Carlo simulation to calculate a derivative. The goal of this thesis is not to reinvent the theory, but to provide support for any curious user or student wishing to extend the web app.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management
Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve
Boulevard Emile Devreux 6, 6000 Charleroi, Belgique
Chaussée de Binche 151, 7000 Mons, Belgique
www.uclouvain.be/lsm

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Frédéric Vrins for his help with this thesis. I would like to thank him sincerely for his responsiveness to my questions and his always very precise explanations to help me accomplish this thesis. I would also like to take this opportunity to thank him for the 2 courses he gave me during my Master's degree, in which I was able to take part and benefit from his transmission of knowledge.

I would also like to thank Anaëlle De Busyer for supporting throughout the making of this thesis. She allowed me to always stay motivated, even in times when motivation was less present. I am also very grateful to her for all the constructive comments she was able to make to allow me to always improve my work.

Finally, I am grateful to my parents for always asking me the right question to help my work evolve. They were able to provide the perspective I did not always have, and I would like to thank them sincerely.

Contents

1 Application purpose

1.1 Objectives

In the world of derivatives, Asian options are widely traded. However, pricing certain versions of this option has proved a challenge for mathematicians due to its characteristics. (Horvath and Medvegyev, 2016)

Monte-Carlo simulation is one method of calculating this price. However, it is sometimes complex for students to understand how it provides a price, as it does not give a closed-form formula.

This thesis aims to provide support, whether for the student seeking to understand how a price can be approximated using the Monte-Carlo method, or for the teacher seeking to illustrate this mathematical concept.

This Master's thesis continues the work of Michel Vanderhulst (Vanderhulst, 2021). He has created a platform that brings together web based applications to price or hedge derivatives and is intended to be extended by other students. In particular, he has implemented several models illustrating the replication of derivative prices by investment strategies, and proving the absence of arbitrage opportunities.

Section 1.2 will briefly summarize the structure of the application and the models already implemented.

The intention of this thesis is to extend the application and provide the documentation needed to understand it. It therefore has 2 clear objectives:

- Implement a new model in the application
- Provide support through this master's thesis for understanding the application and the models implemented, as well as to provide the necessary documentation for anyone wishing to continue development.

As Michel Vanderhulst's dissertation already deals in depth with design choices, this thesis will focus on the methodology used to develop a new model to price Asian options and implement it, as well as the modifications made to the application.

The application hub can be accessed via this link: https://central-hub-app.onrender.com/

Note that it takes approximately 30 seconds for the application to start up. As the application is quite heavy, it sometimes breaks down. Update it by changing one of the inputs.

1.2 State of the app V1

To set the context for the application, this section will give a brief summary of what is already present in the first version of the application.

This section is not intended to be exhaustive of all the work already carried out by Michel Vanderhulst, but to give the basis on which this work will be added.

1.2.1 Hosting service

Initially, the application was hosted on Heroku, a hosting platform for deploying and running modern applications (Heroku, n.d.).

The platform uses "dyno's" which are, in simple terms, the equivalent of mini-servers. Previously, the most basic "dyno" option was free, enabling the deployment of a small app for intermittent use.However, this service has now become **chargeable**.

The application is therefore unavailable from the Heroku link in Michel Vanderhulst's thesis. Section 4.3 will return to the migration of the app and the choice of the new hosting service.

1.2.2 Application functioning

The app is developed in the Python programming language. It uses a special library called Dash, which is a package for web-based data visualization.

Dash uses a system of callback functions, which are functions automatically called when the value of an input changes. They enable interaction between the user and the application.

Dash uses a 2-part system to display its graphs:

- Data: it determines the type of graph and the data to be displayed
- Layout: it manages all graphical options

For each page of the application, a central file named `app.py` manages the application's operation. It controls the initialization of the Dash object that generates the application, the code for the graphics that are displayed and the functions for user-app interactions.

The graphical user interface (GUI) is separated into three files. `appHeader.py` contains the information for the application's banner. `appBody.py` manages all page content: layout, elements to be displayed and inputs. An `inputDescr.py` file contains the texts detailing the inputs, for the sake of clarity.

The mathematical model for calculating the derivative price is stored in `model.py`. It interacts in both directions with `app.py`, as it is recalled each time the value of an input changes.

The figure 1 shows the dependency between files more clearly.

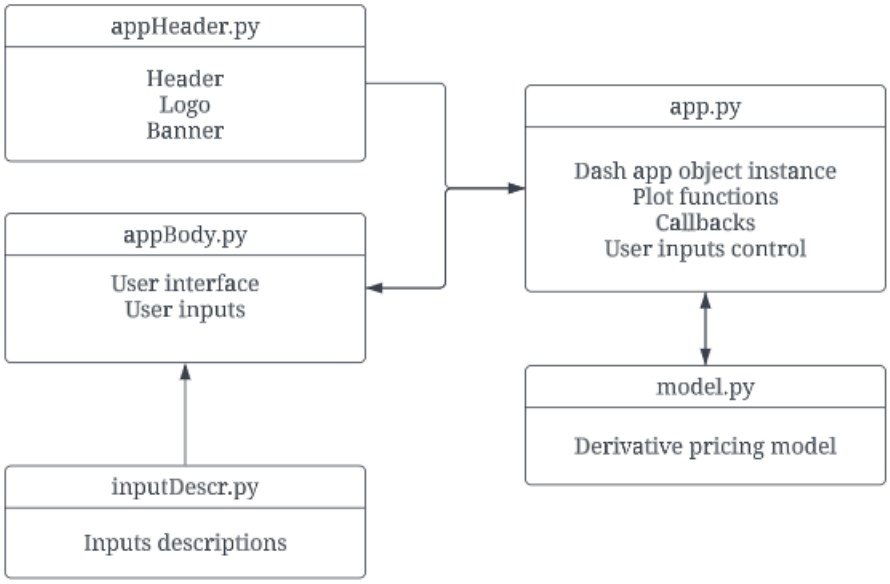


Figure 1: Code architecture (Adapted from Vanderhulst, 2021)

1.2.3 Models

2 types of model are already implemented on the app. Firstly, the Cox-Ross Rubinstein or binomial model. It uses a tree structure to obtain the price of an option (Cox et al., 1979). 2 types of options have been implemented:

- European option
- Arithmetic Asian option

The second model uses the Black-Scholes Merton framework to obtain a closed-form formula for the price of an option (Hull, 2008). 2 types of options have also been implemented:

- European option
- Exchange option

For each model, different graphs are presented in the app to illustrate the theory behind them. In addition, the data produced by the model can be downloaded for curious users.

2 Design process

Software development is a long and methodical process. It requires organization to define objectives and how to achieve them.

The Waterfall methodology was used for this project. This section presents the different stages of this method, as well as its application for this thesis.

It is detailed in order to clarify the process followed for the development of the web app, but also in case another student would like to continue the development of the application and is looking for a methodology.

Other methodologies also exist in the literature, such as Agile, which offers greater flexibility (S. Sharma et al., 2012).

2.1 Waterfall methodology

Waterfall methodology allows you to organize your work into 5 phases, each of which represents a stage in the development and implementation of the model.

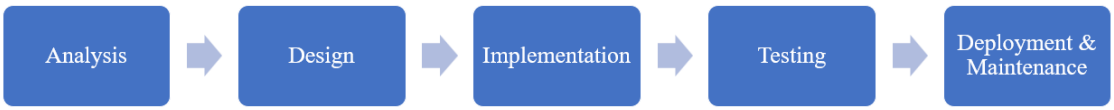


Figure 2: Waterfall methodology diagram

The principle is to define the objectives of these phases beforehand, and only move on to the next one if the previous one has been validated (Aroral, 2021).

2.1.1 Analysis

The first phase, often called Software Requirements Specifications (SRS), analyzes the requirements of the software to be developed. It is generally divided into functional and non-functional requirements. (Bassil, 2011)

Functional requirements detail everything the application needs to include in order to run. This includes the application's purpose, models and everything else that surrounds functionality.

On the other hand, non-functional requirements include everything that does not relate to actual functionality. This includes limitations, constraints and anything else that is imposed on app development. (Aroral, 2021)

2.1.2 Design

During the design phase, you need to imagine and synthesize what the application will actually do. You need to define the architecture of the software, how it will work and the structure of the code to be implemented for the various functionalities.

This part is not yet about the code, but about everything that needs to be done. It defines everything that needs to be written in the chosen programming language. This part is also important to give an idea of the timing to be respected.

2.1.3 Implementation

This phase transforms the theoretical design into a software program by writing the application code. It is at this stage that the application takes concrete form.

2.1.4 Testing

The testing phase checks that everything programmed is working as it should. The aim is to validate that everything listed in the first phase is present, and that limitations and constraints are respected.

It is also during this phase that bugs are identified and corrected.

2.1.5 Deployment and maintenance

Once the application is complete, it must be deployed for user access.

Certain external factors can always compromise its proper functioning, and this is part of the maintenance to be carried out. Errors may also appear even after the testing phase, and need to be corrected.

2.2 App example

The figure 1 describes the steps involved in developing the application that is the subject of this thesis, using the Waterfall methodology.

Phase	**Objectives**
Analysis	Defining thesis objectives Defining models to implement Users constraints
Design	Theoretical approach Graphs Defining features and inputs Hosting service choice
Implementation	Coding the app Code integration to app V1
Testing	Objectives validation Checking correct operation Bug fixing
Deployment & Maintenance	Hosting service deployment Bug fixing and maintenance over time

Table 1: App development phases

3 Theoretical approach

3.1 Derivatives

3.1.1 General overview

"Financial derivatives are financial instruments that are linked to a specific financial instrument or indicator or commodity, and through which specific financial risks can be traded in financial markets" (IMF, n.d.). In other words, a financial derivative binds 2 parties, who fix the terms of a contract based on an underlying asset, over a given period.

The nature of the underlying asset targeted by the derivative can be of different kinds, creating different types of options (Chiu, 2012). These may include foreign currencies, commodities, equities or equity indexes. Its main goal is to provide a protection against a fluctuation risk for a specific situation, in relation to this underlying asset.

There are 4 main families of derivatives. Forwards contracts and futures contracts form 2 families, but are similar in that they stipulate agreement to buy or sell a certain quantity of an asset at a certain price at a later date. The main difference is the ease of customization for a forward contract, whereas this is far more complex for a future contract. The third type is swaps, which are agreements between 2 counter-parties to exchange a stream of payments over a period of time. Finally, the last category is made up of options, which give the right but not the obligation to buy or sell a certain quantity of an asset at a certain predefined price at a certain point in time. (Chiu, 2012)

3.1.2 The option and its payoff

A call option is defined as the right but not the obligation to buy an underlying asset at a certain strike price, denoted K, for a certain maturity (or expiration date) denoted T. Conversely, a put option gives the right but not the obligation to sell.

At maturity, the holder receives a payoff $\psi(.)$ that depends on the type of option contracted. The quantity of the asset compared with the strike price depends on the type of option chosen. Mathematically, the payoff at maturity T takes the following form:

$$\psi^{Call}(X) = \max(0, X - K) \qquad \text{for a call option} \tag{1}$$

$$\psi^{Put}(X) = \max(0, K - X) \qquad \text{for a put option} \tag{2}$$

where X is the quantity of asset that depends on the type of option.

The mathematical expression of a call's payoff is represented by a maximum between 0 and the difference between the specific asset quantity and the strike price K. This represents the fact that if this quantity is lower than the strike price (i.e. $X < K$).

Conversely, the holder of a put option will only sell if the strike price is greater than the specific quantity asset at maturity.

The most common type of option is the European option. It pays the difference between the price of the underlying stock at maturity T and the strike price K. The payoff is therefore:

$$\psi^{Call,EU} = \psi^{Call}(S_T) = \max(0, S_T - K) \qquad \text{for a call option} \tag{3}$$

$$\psi^{Call,EU} = \psi^{Put}(S_T) = \max(0, K - S_T) \qquad \text{for a put option} \tag{4}$$

3.1.3 Asian Option

An Asian option is a type of option where the payoff depends on the different prices of the underlying asset between the start of the contract and its maturity T. This type of option is called path-dependent because it depends on the path taken by the asset during the contract period. For an Asian call option, the payoff is given by the difference between the average of the price of the underlying asset during its life and the strike price.

The mean can be either geometric or arithmetic.

Let $x = \{x_1, x_2, ..., x_n\}$ be a set of observations. The arithmetic mean $A(x)$ is such as:

$$A(x) = \frac{1}{n} \sum_{j=1}^{n} x_j \tag{5}$$

The geometric mean $G(x)$ is such as:

$$G(x) = \left[\prod_{j=1}^{n} x_j \right]^{\frac{1}{n}} \tag{6}$$

(5) and (6) become respectively $A(S)$ and $G(S)$ for an Asian option where S is the underlying stock, $S = \{S_t, t \in \{t_1, ..., t_n\}$ and $t_n = T\}$.

Mathematically, the payoff is the difference between this average and the strike price, and depends on the type of average chosen in the contract.

The payoff equation is:

$$\psi^{Call,Ari} = \psi^{Call}(A(S)) \qquad \text{for an arithmetic Asian call} \tag{7}$$

$$\psi^{Call,Geo} = \psi^{Call}(G(S)) \qquad \text{for a geometric Asian call} \tag{8}$$

The put formula can be simply found by using the ψ^{Put} formula in (2).

For simplicity's sake, the following theoretical explanations will use the call as an example.

3.1.4 Arbitrage-free & risk-neutral pricing

At this point, we know what the payoff of an option is, but we are missing one piece of information: **how do we calculate the price?**

The fair price V of a product is given by **the no-arbitrage requirement**. It coincides with the cost of launching a **self-financing strategy replicating the payoff**.

A self-financing portfolio Π is said self-financing because the portfolio value evolves without injecting or withdrawing cash (Vrins, 2021). We only move the money from one type of account to another.

To understand the importance of the no-arbitrage requirement, let's illustrate what happens if $V_t \neq \Pi_t$. Suppose we have a positive constant k, and let's assume an initial condition where $V_0 = \Pi_0 + k$. The investor can sell the derivative at $\Pi_0 + k$ and invest in the self-financed replicating portfolio Π_0.

It means that he will make a profit of k, without taking any risk. This is possible because the self-financing replicating portfolio will deliver exactly the same payoff as the derivative, as the portfolio mimics the derivative's payoff (Example taken from Vanderhulst, 2021). The cost of the replication strategy **must** therefore be equal to the price of the derivative, **to avoid any arbitrage opportunities**.

The form of a derivative's price is given by the Fundamental Theorem of Asset Pricing (FTAP):

$$\Pi_0 = V_0 = \mathbb{E}^{\mathbb{Q}}\left[\frac{\psi(X)}{\beta_T}\right] \tag{9}$$

where Π_0 is the cost of the replicating strategy, V_0 the derivative price at time 0, $\psi(X)$ the derivative's payoff and β_T the discounting factor at maturity T.

It states that **the unique price that prevents arbitrage opportunities is found using the risk-neutral expectation and corresponds to the cost of the replication strategy by trading in the primary assets** (Vrins, 2021).

To properly understand this equation, we must explain the concept of **risk-neutral valuation**.

We first need to define the concept of martingale. A martingale is a process for which the expected value of the process for the next step is the current value of the process, given all the information that we have up to the current step (Shreve, 2004).

Intuitively, if we want to calculate the value of a cash flow in the future today, we discount it to its present value. However, as an option is linked to an underlying asset, there is also a risk associated with this asset and therefore with the option. Consequently, depending on his risk aversion, an investor will expect a risk premium, which will vary the price and make its calculation more complex.

To solve this problem, **we use the risk-neutral probability measure** $\mathbb{Q}$. This probability measure is not real, it is different from the real-world probability. It allows us to make the assumption that, under this risk-neutral probability, all assets follow a growth rate equal to the risk-free rate and, consequently, no risk premium will be charged.

This probability measure $\mathbb{Q}$ has a fundamental property. Under $\mathbb{Q}$, prices of non-dividend-paying assets discounted at the risk-free rate are martingales. This property ensures that no arbitrage opportunities can arise.

3.2 Monte-Carlo Simulation

In equation (9), we saw that the price of a derivative was the expectation of its payoff at maturity T, discounted at the risk-free rate. In the case of a geometric Asian option, this expectation accepts a closed-form formula. Indeed, the product of a lognormally distributed asset is also lognormally distributed, allowing us to apply the Black-Scholes Merton framework. (Horvath and Medvegyev, 2016)

However, in the case of an arithmetic Asian option, the distribution of the sum of lognormal correlated random variables is unknown, which prevents us from obtaining an explicit price formula. (Milevsky and Posner, 1998)

Used in many fields, such as biology and chemistry, the Monte-Carlo method can be used **to estimate the expectation of a random variable or a function of random variable**. It involves simulating a large number of samples of this random variable and calculating the mean to obtain an estimate of its expected value.

We can therefore see our equation as a function of random variables and use the Monte-Carlo method to approximate its expectation, which is necessary to calculate the option price.

3.2.1 Basic Monte-Carlo

Suppose that we want to estimate θ and $\theta = \mathbb{E}[g(X)]$ where $g(X)$ is an arbitrary function of the random variable X. The variance of $g(X)$ is denoted σ_g^2. Let $X_1, X_2, ..., X_m$ a sequence of m independent and identically distributed random variables generated from the density probability function $f(X)$. The estimator of θ is such as:

$$\hat{\theta} = \frac{1}{m} \sum_{i=1}^{m} g(X_i) \tag{10}$$

where $i \in \{1, 2, ..., m\}$

By the strong law of large number,

$$\frac{1}{m} \sum_{i=1}^{m} g(X_i) \xrightarrow{a.s.} \mathbb{E}\Big[g(X)\Big] \qquad \text{as} \quad m \to \infty \tag{11}$$

It can also be shown that the expectation of $\hat{\theta}$ is equal to θ. Since the expectation of a sum is equal to the sum of the expectations and since $g(X_i)$ has the same distribution as $g(X)$, the expectation of $g(X_i)$ is equal to θ.

$$\mathbb{E}[\hat{\theta}] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}[g(X_i)] = \frac{1}{m} m * \mathbb{E}[g(X)] = \mathbb{E}[g(X)] \tag{12}$$

The variance is an important measure in Monte-Carlo, as it will change with the number of simulations. **As m increases, the variance will decrease**.

$$\mathbb{V}ar[\hat{\theta}] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{V}ar[g(X_i)] = \frac{1}{m} \sigma_g^2 \tag{13}$$

Consequently, $\hat{\theta}$ is a variable with expectation θ and variance tending towards 0. The dispersion of $\hat{\theta}$ around its mean vanishes as $m \to \infty$ (Vrins, 2021).

3.2.2 Variance reduction method

The purpose of this thesis is to illustrate that, using Monte-Carlo simulation, we can estimate the expected payoff of a derivative, which is central to obtaining its price.

In practice, obtaining an estimate of this expectation is a good starting point, but we want to be sure that this estimate is as accurate as possible. This accuracy is represented by the variance of the simulation.

Intuitively, as we saw in the previous sub-subsection, we could increase the number of simulations m to reduce it. However, it is possible to further improve this accuracy, without changing the number of simulations. In other words, for a fixed number of simulations, we can obtain a more precise estimator, i.e. with a lower variance than that obtained with a classic MC simulation. (Hull, 2008)

We will therefore present 2 methods for reducing this variance, without having to increase the number of simulations.

- **Antithetic variate**

This method uses the negative dependency between 2 variables to reduce the variance (Glasserman, 2003). Mathematically, if we have 2 random variables A and B, the variance of the sum of these 2 random variables is such that:

$$\mathbb{V}ar[A+B] = \mathbb{V}ar[A] + \mathbb{V}ar[B] + 2Cov[A,B] \tag{14}$$

where $Cov[A,B] = \rho_{A,B}\sigma_A\sigma_B$ with $\rho_{A,B}$ being the correlation between A and B.

If the 2 variables are negatively correlated (i.e. $\rho_{A,B} < 0$), the covariance becomes negative and $\mathbb{V}ar[A+B] < \mathbb{V}ar[A] + \mathbb{V}ar[B]$.

To obtain a correlation as close as possible to -1, we need to take the opposite of a random variable (called the antithetic variable). Let's define $\mathrm{g}(Y) = g(-X)$.

The choice of our antithetic variable must respect a strict condition: the distribution of the random variable X must be symmetrical in 0 in order to use Y as the antithetic variable. This is necessary to ensure that $g(X)$ and $g(Y)$ have the same distribution and, consequently, the same expectation. This means that the sum of their 2 expectations divided by 2 is equal to their common expectation:

$$\mathbb{E}\Big[\frac{g(X)+g(Y)}{2}\Big] = \frac{\mathbb{E}[g(X)] + \mathbb{E}[g(Y)]}{2} = \mathbb{E}[g(X)] \tag{15}$$

Thanks to this property, assuming that $g(X)$ and $g(Y)$ have the same distribution, the Monte-Carlo estimator becomes:

$$\hat{\theta}^{AV} = \frac{1}{m}\sum_{i=1}^{m}\Big(\frac{g(X_i)+g(Y_i)}{2}\Big) = \frac{1}{2m}\sum_{i=1}^{m}\Big(g(X_i)+g(Y_i)\Big) \tag{16}$$

Intuitively, it means that very large outputs are compensated for by the generated antithetic variable. It shrinks extreme value closed to the mean.

- **Control Variate**

The control variate method (CV) uses information about the estimation errors of known quantities to reduce the estimation error of an unknown quantity. Recall that our classical estimator of θ is $\hat{\theta} = \frac{1}{m}\sum_{i=1}^{m} g(X_i)$.

Suppose now that, from the same data set $\{X_i, 1 \leq i \leq m\}$, we can also estimate a known quantity $\mathbb{E}[h(X)]$ where $h(X)$ is another arbitrary function of X whose expectation is known. We define $\hat{\gamma} = \frac{1}{m}\sum_{i=1}^{m} h(X_i)$ as the MC estimator of this known quantity.

Therefore, we can compute the error on this estimation and use it to correct $\hat{\theta}$. The estimation of $\hat{\theta}$ using a control variate is given by:

$$\hat{\theta}^{CV} = \hat{\theta} - \alpha * (\hat{\gamma} - \mathbb{E}[h(X)]) \tag{17}$$

where α is the coefficient used to minimize the variance.

Its optimal value α^* is given by:

$$\alpha^* = \frac{Cov[g(X), h(X)]}{\mathbb{V}ar[h(X)]} \tag{18}$$

However, to calculate the covariance between $g(X)$ and $h(X)$, we need $\mathbb{E}[g(X)]$, which is precisely what we are trying to estimate. Hence, we cannot compute α^* in the applications where θ is unknown and need to be estimated.

An alternative is to calculate α^* directly in the simulation and still get most of the benefit of this method.

$$\hat{\alpha}^* = \frac{\sum_{i=1}^{m} \Big((g(X_i) - \hat{\theta})(h(X_i) - \hat{\gamma}) \Big)}{\sum_{i=1}^{m} (h(X_i) - \hat{\gamma})^2} \tag{19}$$

3.3 Stock Dynamic

We now have several methods for calculating our payoff expectation, but we still need to assess how our underlying asset behaves.

Indeed, as an Asian option is path-dependent, we need to know the different values of our asset between the start of the contract and its expiration date T.

The stochastic differential equation (SDE) of a stock is the following geometric Brownian motion (GBM).

$$dS_t = \mu S_t dt + \sigma S_t W_t \Leftrightarrow \frac{dS_t}{S_t} = \mu dt + \sigma W_t \tag{20}$$

where $W_t \sim \sqrt{t}Z$ and $Z \sim \mathcal{N}(0,1)$ under $\mathbb{P}$

W is a Brownian motion (or Wiener process) under $\mathbb{P}$. It has the following properties (Vrins, 2021):

- It is grounded: $W_0 = 0$
- It has independent increments: for any $q \leq r \leq s \leq t$, $(W_r - W_q) \perp (W_t - W_s)$
- The increments are normally distributed: $W_t - W_s \sim W_{t-s} \sim \mathcal{N}(0, t-s)$

W is the driver of the dynamic. It captures **the uncertain nature of the process**, i.e. the daily movements of a stock on the market.

The dynamic is expressed under the historical probability measure $\mathbb{P}$. However, we are looking for the payoff expectation under the risk-neutral measure $\mathbb{Q}$, which is different.

Girsanov's theorem establishes how a process changes if we change its probability measure. By Girsanov, we know that S becomes a GBM with parameters (r, σ) under $\mathbb{Q}$:

$$\frac{dS_t}{S_t} = rdt + \sigma \widetilde{W}_t \tag{21}$$

where $\widetilde{W}$ is a $\mathbb{Q}$-BM, r is the risk-free rate and σ is the volatility.

Therefor, $\widetilde{W}_t \sim \sqrt{t}\widetilde{Z}$ where $\widetilde{Z} \sim \mathcal{N}(0,1)$ under $\mathbb{Q}$.

By applying Itô's lemma, we can find the solution of the SDE in (21):

$$S_t = S_0 e^{(r-\sigma^2/2)t + \sigma\sqrt{t}Z} \tag{22}$$

where $Z \sim \mathcal{N}(0,1)$ under $\mathbb{Q}$.

To replicate the conditions of a stock on a stock market, we can discretize this equation in n small time step Δ where $\Delta_j = t_{j+1} - t_j$ and $j \in \{1, ..., n\}, t_n = T$.

The equation (22) becomes:

$$S_{t_{j+1}} = S_{t_j} e^{(r-\sigma^2/2)\Delta_j + \sigma\sqrt{\Delta_j} Z_j} \tag{23}$$

where $Z_1, ..., Z_n$ are i.i.d. and $\sim \mathcal{N}(0,1)$ under $\mathbb{Q}$.

An example of a stock path is provided in Appendix 8.1.

3.4 Models implementation

3.4.1 Theory adapting

Let's recap: the fair price of a derivative takes the form of an expectation of a payoff under the risk-neutral measure $\mathbb{Q}$, discounted at the risk-free rate.

This payoff is determined as a function of an underlying asset, for which we have described the solution of its differential stochastic equation under the risk-neutral measure. We can then estimate the expectation of this payoff using Monte-Carlo simulation.

In theoretical notation, θ refers to the expectation of our payoff $\mathbb{E}^{\mathbb{Q}}[\psi(.)]$, $g(X)$ represents the payoff $\psi(X)$ and X represents our stock S.

Note that in the case of the antithetic variable, we stipulated that we should choose the opposite of the random variable X to minimize variance. This would be equivalent to taking $-S$, which would be incorrect as S does not have a symmetrical distribution.

Therefore, when we use the antithetical method, we are working on Z because $Z \sim \mathcal{N}(0,1)$, which means it is symmetric in 0 and $-Z$ will have the same distribution as Z.

For simplicity of notation, we will keep the notation $\psi(A(S))$ even if the modifications are made on Z and not S.

3.4.2 List of implemented models

A generic form of the arithmetic Asian option price estimator can be written as:

$$\hat{V}_0^{Asian} = e^{-rT} \underbrace{\hat{\psi}(A(S))}_{\hat{\theta}^{Cl} \text{ or } \hat{\theta}^{AV}} - \overbrace{\hat{\alpha} * \left[\hat{\gamma} - \mathbb{E}[h(X)]\right]}^{\text{Control variate}} \tag{24}$$

where $\hat{\psi}(A(S))$ can be estimated by the classical MC method (we represent this case by $\hat{\theta}^{Cl}$) or by the antithetic variate method (we represent this case by $\hat{\theta}^{AV}$).

The right-hand side of the equation is calculated using the control variate method. $\hat{\gamma}$ is the MC estimator of an arbitrary function of X denoted $h(X)$ and $\mathbb{E}[h(X)]$ is its analytically calculable expectation. The function $h(X)$ depends on the chosen control variable.

The following table summarizes the different models implemented in the app. The "/" sign means that the method is not used.

Name	$\hat{\psi}(A(S))$	$h(X)$	$\mathbb{E}[h(X)]$
Classical MC	$\hat{\theta}^{Cl}$	/	/
MC with antithetic	$\hat{\theta}^{AV}$	/	/
MC with european as CV	$\hat{\theta}^{Cl}$	$\psi(S_T)e^{-rT}$	V_0^{Euro}
MC with antithetic and european as CV	$\hat{\theta}^{AV}$	$\psi(S_T)e^{-rT}$	V_0^{Euro}
MC with average of european as CV	$\hat{\theta}^{Cl}$	$A(\hat{\psi}(S_t)e^{-rt})$	$A(V_{[0,t]}^{Euro})$
MC with geometric as CV	$\hat{\theta}^{Cl}$	$\psi(G(S))e^{-rT}$	V_0^{Geo}

Table 2: Implemented models table

V_0^{Euro} and V_0^{Geo} correspond respectively to the price of a European with maturity T and an Asian geometric option with maturity T calculated using the Black-Scholes Merton analytical formula. The exact formulas are available in Appendix 8.2.

Note also the "*MC with average of European as CV*" model equation. The control variable is the average price of European options for each maturity t between 0 and T, $0 \leq t \leq T$. $V_{[0,t]}^{Euro}$ therefore corresponds to the price of a European option of maturity t, calculated at time 0.

4 App implementation

As explained above, this thesis deals with the implementation of models in an existing app. The goal is not to review the entire code architecture.

This section will therefore focus on modifications to the platform or on points that proved challenging during the creation of the application, in order to give guidance to potential students wishing to develop the application further.

For the development of the new part of the app, we can focus on 2 key features:

- -Consistency: When it comes to adding a new part to existing software, it is important to keep the user interface consistent with what is already been created. This creates a sense of control and reliability for the user (De La Riva, 2021).

  This does not mean copying exactly what is already been done, but understanding the style and structure and using it to develop your own part of the app.

- -Futur-proof: Given that the goal of the app is to be further enhanced by other students, it is important to keep the design open to improvement. This includes clarity in the code and clear annotation of code specifics and steps (Molina, 2023).

4.1 App design

As we have seen in the theory, several methods can be implemented to modify the results of Monte-Carlo simulation. In practice, we want to show 2 results:

- We can use the Monte-Carlo method to calculate the price
- Different variance reduction technics provide a better accuracy

To demonstrate this, we first need to think about what the user is going to enter: his inputs. There are 3 categories of inputs:

Category	**Inputs**
Models selector	-Choose between models
Option parameters	-Call or put option -Model variables
Simulation parameters	-Simulation variables -Seed

Table 3: Inputs categories

Based on the inputs, the outputs must be determined. First of all, we want to demonstrate

the price obtained by selected method, and above all its accuracy. We will therefore display the price obtained, together with the standard error and the 95% confidence interval.

Note that we use the standard error σ instead of the variance σ^2, as the latter is generally very small. It can be simply calculated using $\sigma = \sqrt{\sigma^2}$.

A graphical representation will illustrate the differences between the two selected models. To achieve this, a box-plot for each model will be displayed on a single graph. For the sake of clarity, the application is limited to the comparison between 2 models.

It is important to note here that the graphs do not display the results of a single simulation, **but of several simulations** (which is also determined by user input). This was done to illustrate that, in the case of more precise methods, the prices obtained for each simulation are highly concentrated, in contrast to less precise method.

To observe the results independently, the distribution of the results obtained will be displayed in a bar graph and the density will be estimated using the Kernel method (see [Parzen, 1962] for more details).

4.2 Code writing

The application is written in Python and uses the Dash package which is a framework for web app development and visual interface construction (Castillo, 2023).

Python is one of the most widely used programming languages in finance (G. Sharma, 2023). It seemed obvious to stay with this language, especially as migrating the whole application to another language would take a lot of time for very few positive changes.

In software development, code is divided into 2 main parts (Simmons, 2023):

- The front end, for everything linked to user interaction and interfaces.
- The back end, which manages responses to interactions and application development "behind the scenes".

Dash is a framework for creating visuals and interactions in Python. It is specifically built to handle all front-end requests and is built on a higher level by using Flask. In simple terms, Flask is a Python package that can be used to develop other Python packages, in particular to create web apps. It manages the back-end of what is displayed by Dash.

Other packages, such as Django, manage both the back-end and front-end of the web app. However, they are much harder to master and are designed to build more complex applications.

It is therefore preferable to continue using Dash, as our app does not require any additional

complexity. However, using Django or another more complex framework could become an option for a student wishing to develop the application code in greater depth.

Our back-end language for everything to do with models is therefore python. When our app is used locally, it uses the resources of our computer, which are generally sufficient to run the app efficiently. However, since we have to host it on an online service, our resources will be limited.

The application's Monte-Carlo simulation code is by definition heavy, requiring a large number of simulations to be efficient. It was therefore important to pay close attention to certain points when developing the code, in order to be as efficient as possible

The following sub-subsections describe the methods used to try to optimize the application for this purpose.

4.2.1 PEP8 and best practices

PEP8 is a set of rules for annotating and building code. It does not advise on the development itself, but on the structure of how the code is written (Van Rossum et al., 2013).

To obtain future-proof code, it is important to annotate it well. The PEP8 recommendations provide a guide to do this correctly. In addition, they recommend keeping to a maximum of 79 characters per line, indenting correctly and separating `import` functions at the beginning of the code.

An exhaustive list of these tips can be found here: https://peps.python.org/pep-0008/

In addition, some specific tips have been used to improve the efficiency of the code (Hossain, n.d.):

- Minimize the number of loops (see next sub-subsection)
- Use list comprehension
- Do not use global variables
- Use library function (especially those from NumPy and sciPy)
- Do not import all function of a package

Using these tips is one of the best practices you can adopt to ensure that your code is optimized to the maximum. If another student wishes to add further models in the app, he or she is strongly advised to keep these practices in mind when coding.

4.2.2 Vectorization

The first point of the previous sub-section advised minimizing the number of loops. This is particularly true in the case of Monte-Carlo simulation.

Intuitively, when we want to simulate m paths of n time steps, we would like to make a code such that:

```
for i in range(0,m):
  for j in range(0,n):
    #your code (incl. random variables)
```

By doing this, we create a nested loop within another loop, which is resource-intensive for Python.

An alternative is to use vectors to calculate our simulations. NumPy allows the use of a variable type *numpy.ndarray* and functions around it.

In the case of Monte-Carlo simulation, it allows us to create our matrix of normally distributed random variables using a line of code and then apply the necessary transformations:

```
Z = numpy.random.normal(size=(n, n))
#Add operations to get all the stock paths by using NumPy functions
```

For curious readers, the implemented code is available in open access here: https://github.com/Vapaquay

4.3 Hosting service benchmark

4.3.1 Selection

As previously explained, Heroku now operates on a dyno system and there is no longer a free tier. The hosting service application must therefore be migrated.

Hosting platforms like Heroku are known as PaaS (Platform as a Service). They offer the possibility of hosting a web application on a runtime environment. In other words, instead of running the application locally on our computer, it runs on an external server and is accessible via a link (Azure, 2023).

This type of hosting comes at a price, as it uses resources on this server. In the case of Heroku, a "dyno" refers to a mini-server, which incurs costs.

Some platforms still have free parts. The following table summarizes 5 possible alternatives to Heroku (Ekete, 2023):

Site	Advantages	Disadvantages
Render	-Free Tier -Illimited number of app	-Slower than Heroku
Cyclic	-Free Tier -Fast restart	-Only 1 app at a time
Railway	-Powerful servers	-Free version limited to 500h
Deta Space	-Everything is free	-Beta version
Fly.io	-Powerfull free allowance	-No free tier, only free allowance

Table 4: Alternative to Heroku benchmark

To determine which service to use, we define 2 characteristics that should guide our choice:

1. Number of apps: We need to be able to run several applications at the same time, in order to host the different models.
2. Long-term free tier: The option must be free of charge in the long term, to avoid having to migrate from one platform to another.

Render seems to be a solid alternative, offering the possibility of hosting several apps and a usage limit of 500h per month. In addition, it offers precise documentation on how to migrate your app from Heroku (Render, 2023).

4.3.2 Installation

To help another student upload his or her own application to this new hosting service, this section details the steps to follow.

First, create an account on https://render.com/ and register.

Next, a package specially designed for uploading Dash projects on Render exists and is available here: https://pypi.org/project/dash-tools/

To install the package, simply write the following line of code in the command prompt (accessible by searching for cmd in the computer's search bar): `pip install dash-tools`.

Once installed, you can open the web page by writing the following instruction in the command prompt: `dashtools gui`.

Once this page is open, a 'Deploy' option is available on the left of the page. Once on this page, you need to copy the path of the folder in which the files are located.

The tool will then display the requirements needed to deploy the application (see figure 3):

- **File exists (*src/app.py*):** the main file of your application must be placed in a folder named `src`.
- **File exists (*render.yaml* and *requirements.txt*):** these files are required by Render to start the application. Good news: by clicking on the little arrow on the right, dashtools will generate them automatically.
- **Project Requirement (Pushed to GitHub):** the complete file must be uploaded on GitHub. Indeed, Render uploads the files directly from GitHub to launch the app. A full tutorial on how to create a GitHub repository is available here: https://docs.github.com/fr/get-started/quickstart/create-a-repo
- **Code exists:** The application's Python code must contain `server = app.server` where `app` is the name of the Dash instance created (i.e. an object of the Dash class that enables web app creation). The instruction must be placed after the object initialization command (i.e. `app = dash.Dash`).

You must also select the application name from the text bar. Once all these operations have been completed, the "ready" message appears, and the application can be deployed.

The web app will be automatically created and host on a server by using the Render account created beforehand. Note that it takes between 5 and 10 minutes for the link to be available.

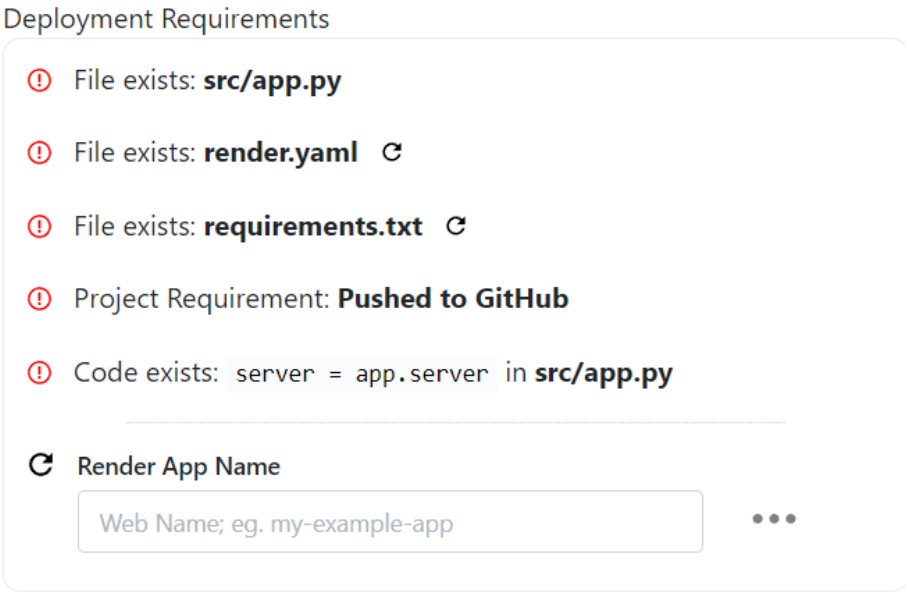


Figure 3: Dash-tools requirements

5 App presentation

5.1 Overview

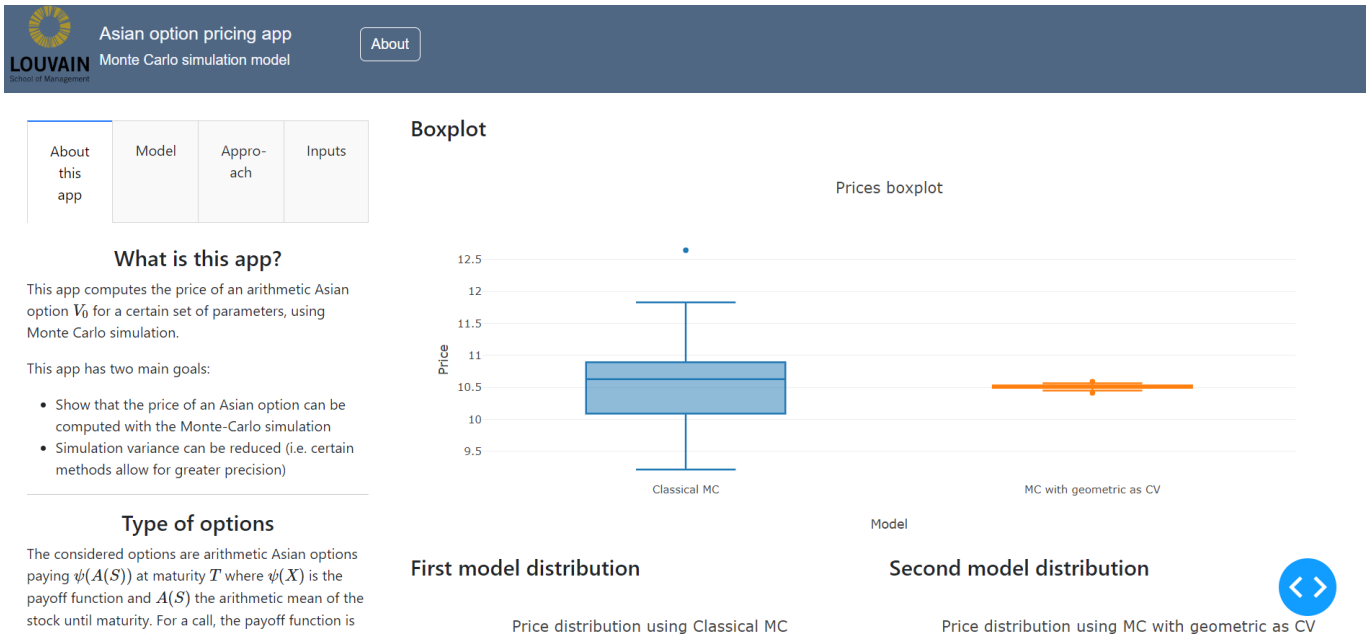


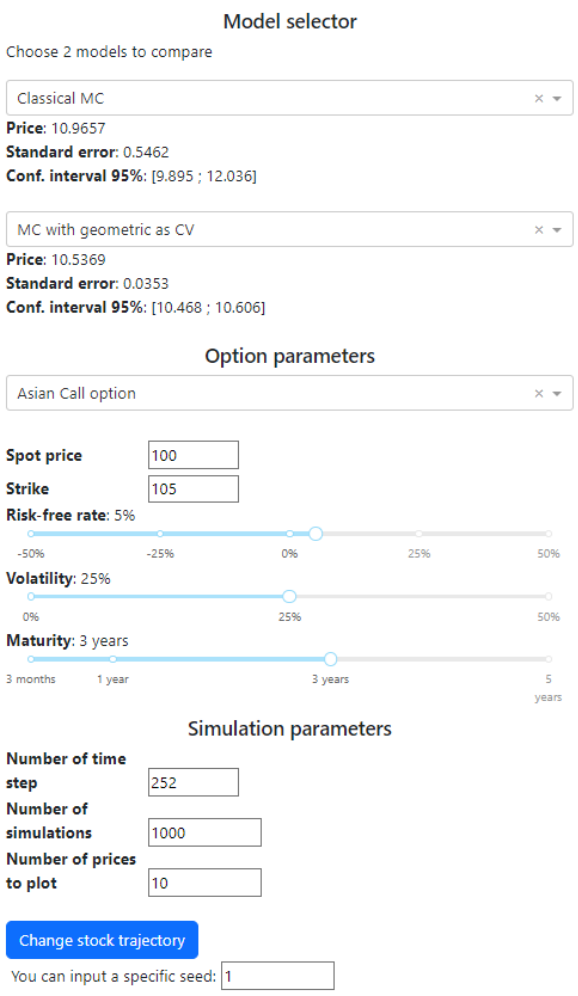
Figure 4: App overview

The figure 4 shows an overview of the app. To ensure consistency with the parts already created by Michel Vanderhulst, the application uses the same graphic characteristics. The top banner and section organization are identical to those of existing apps.

The "*About this app*" tab gives a quick overview of the app and its objectives. The "*Model*" tab presents the model's hypotheses and the theory of Monte-Carlo simulation. Explanations are kept succinct for the sake of clarity, and academic references are provided for curious users. The "*Approach*" tab details the exact method used to obtain our estimate of the price of an Asian option. Finally, the "*Input*" tab allows the user to modify the model's parameters and is detailed below.

5.2 Inputs

The inputs are separated into three parts:



- **Models selector:** the user can choose the 2 models to be compared. This section also displays the price, standard error and 95% confidence interval. They are obtained from a single run of the Monte-Carlo simulation.
- **Option parameters:** the user can select model parameters to analyze their impact on price or accuracy.
- **Simulation parameters:** the user can choose the parameters for the Monte-Carlo simulation. The number of time steps corresponds to n and the number of simulations to m. Finally, the user can choose the number of prices to be displayed. This corresponds to the number of times the Monte-Carlo method will be used to calculate a price.

Figure 5: Inputs

To make the last point very clear, let's look at a concrete example. If the number of time steps is 252, the number of simulations 1000 and the number of prices displayed is 10, the app will calculate 10 times an estimate of the option price with the first 2 parameters. There will therefore be 10 times 1000 simulations of 252 time steps and these 10 prices will be used to create the charts.

An interesting feature is also worth noting. If the user chooses the same model twice, **the calculated results will be different even if all the parameters are identical**. This illustrates the random nature of the model.

Finally, the seed is set by default to enable sensitivity analysis. Users can enter a value themselves, or ask the application to generate a new one using the button 'Change stock trajectory'.

5.3 Plots

The first 2 graphs are box-plots illustrating the differences in accuracy between the 2 selected models. The tighter the box-plot, the more accurate the model. As an example in the figure 6, we can see that the model using the geometric Asian option as a control variable is more accurate than the classical method, since all the results obtained are very close.

Figure 6: Boxplots in the app

The 2 graphs in figure 7 show the distribution of prices obtained for each use of the Monte-Carlo method. The probability density is estimated using the Kernel density estimation (KDE) method on the basis of these results. This allows the user to vary the parameters to analyze the impact on the distribution.

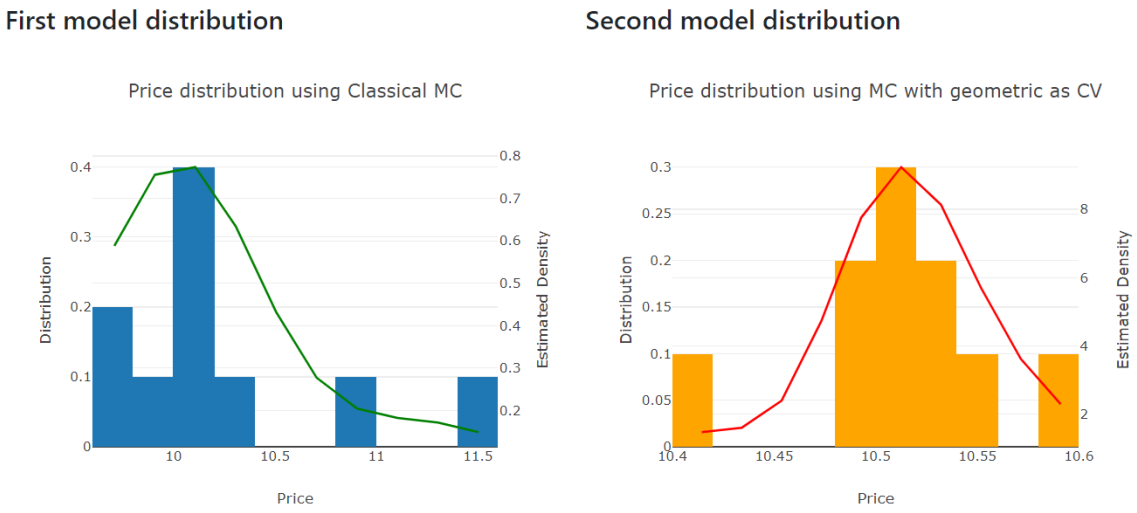


Figure 7: Distribution plot in the app

6 Conclusion

The objective of this master's thesis was to extend a web application that provides different methods for pricing derivatives. We contributed to this application by implementing the Monte-Carlo method to price arithmetic Asian options.

This work had two main objectives. Firstly, we wanted to demonstrate how the Monte-Carlo method could be used to price a derivative. Secondly, we wanted to illustrate how the accuracy of the method could be improved using variance reduction techniques.

These 2 objectives were achieved through the creation of an application illustrating these results. The heart of this web application lies in its educational interface, which allows the user to interact with complex financial concepts effortlessly. This application is documented thanks to this Master's thesis to enable other students to continue extending it.

We have therefore detailed the methodology used to build the software, for example using the Waterfall method. We have also detailed the various changes made to the application, such as the migration of the hosting service or the use of vectorization to improve loading speed.

I am convinced of the usefulness of this application and of the importance of its continued development. It enables other students of quantitative finance to understand certain complexities that are sometimes hard to grasp. In addition, it will also enable future students who develop it to acquire knowledge that will be useful in their future careers.

I therefore hope that other students will take an interest in the project and continue to feed the application.

7 References

Aroral, H. K. (2021). Waterfall process operations in the fast-paced world: Project management exploratory analysis. *International Journal of Applied Business and Management Studies*.

Azure. (2023). What is paas? https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-paas

Bassil, Y. (2011). A simulation model for the waterfall software development life cycle. *International Journal of Engineering 'I&' Technology*, *2*(5).

Castillo, D. (2023). *Develop data visualization interfaces in python with dash*. https://realpython.com/python-dash/#:~:text=Dash%20is%20an%20open%2Dsource,requiring%20advanced%20web%20development%20knowledge.

Chiu, M. (2012). Derivatives markets, products and participants: An overview. In B. f. I. Settlements (Ed.), *Proceedings of the workshop* (pp. 3–11). Bank for International Settlements. https://EconPapers.repec.org/RePEc:bis:bisifc:35-01

Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of Financial Economics*.

De La Riva, M. (2021). *Why consistency is so incredibly important in ui design*. https://careerfoundry.com/en/blog/ui-design/the-importance-of-consistency-in-ui-design/

Ekete, D. (2023). *5 heroku alternatives for free full stack hosting*. https://www.makeuseof.com/heroku-alternatives-free-full-stack-hosting/

Glasserman, P. (2003). *Monte carlo methods in financial engineering*. Springer.

Heroku. (n.d.). *The heroku platform* [Retrieved from: https://www.heroku.com/platform].

Horvath, A., & Medvegyev, P. (2016). Pricing asian options: A comparison of numerical and simulation approaches twenty years later. *Journal of Mathematical Finance*, (6), 810–841.

Hossain, T. (n.d.). *Speed up python code* [Retrieved from: https://www.loginradius.com/blog/engineering/speed-up-python-code/].

Hull, J. (2008). *Options, futures and other derivatives*. Pearson.

IMF. (n.d.). *Financial derivatives*. https://www.imf.org/external/np/sta/fd/

Milevsky, M. A., & Posner, S. (1998). Asian options, the sum of lognormals, and the reciprocal gamma distribution. *Journal of financial and quantitative analysis*, *33*(3).

Molina, F. (2023). *5 tips for successful web application development*. https://www.bairesdev.com/blog/web-application-development-tips/

Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, *33*, 1065–1076.

Render. (2023). Migrate from heroku to render. https://render.com/docs/migrate-from-heroku

Sharma, G. (2023). *Best programming languages for finance 'i&' fintech in 2023*. https://www.bankersbyday.com/programming-languages-banking-finance-fintech/

Sharma, S., Sarkar, D., & Gupta, D. (2012). Agile processes and methodologies: A conceptual study. *International Journal on Computer Science and Engineering*.

Shreve, S. E. (2004). *Stochastic calculus for finance ii*. Springer.

Simmons, L. (2023). *Front-end vs. back-end: What's the difference?* https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/#:~:text=Front%2Dend%20development%20focuses%20on%20the%20visual%20aspects%20of%20a,create%20interactive%2C%20visually%20pleasing%20websites.

Van Rossum, G., Warsaw, B., & Coghlan, N. (2013). *Pep 8 – style guide for python code* [Retrieved from: https://peps.python.org/pep-0008/].

Vanderhulst, M. (2021). *Design of a pedagogical tool in python to price and hedge european options* (Master's thesis). Louvain School Of Management, Université Catholique de Louvain.

Vrins, F. (2021). *Derivatives pricing* [Retrieved from Louvain School of Management LLSM2225 Derivatives Pricing.].

Wilmott, P. (2006). *Paul wilmott in quantitative finance*. Wiley.

Zhang, H. (2009). Pricing asian options using monte carlo methods [Published for Department of Mathematics, Uppsala University].

8 Appendix

8.1 Stock path

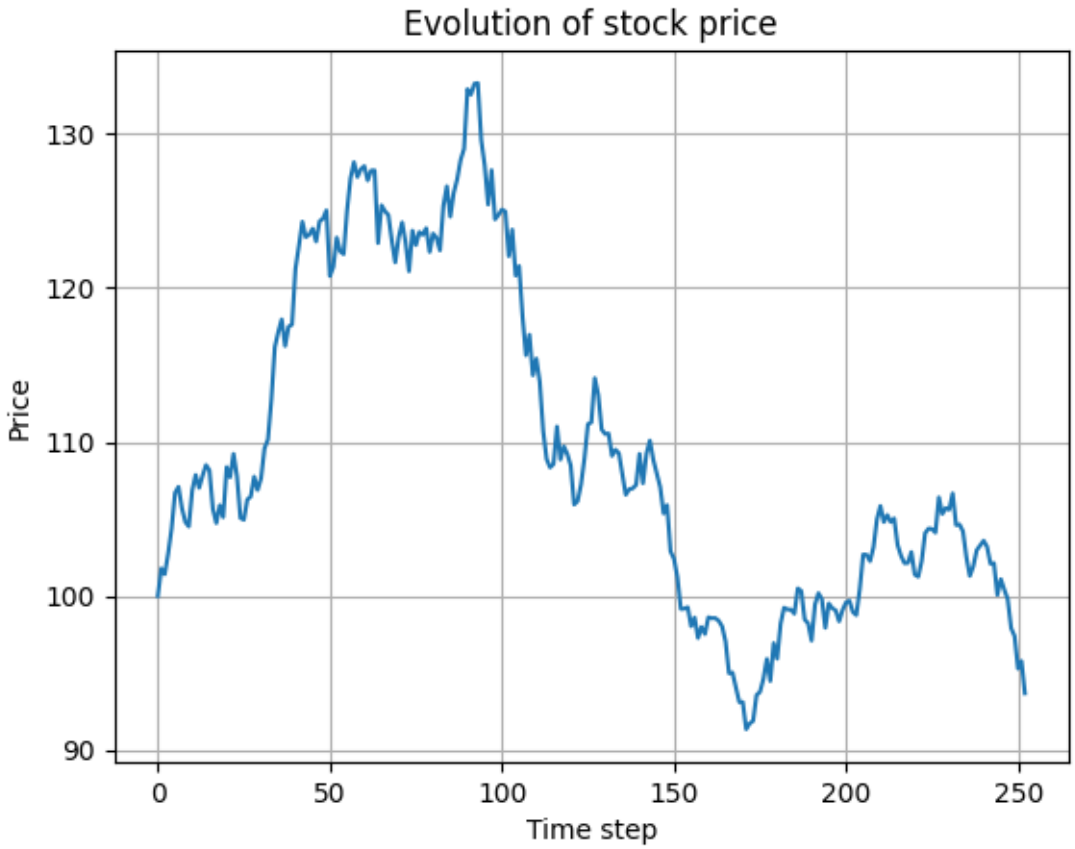


Figure 8: Example of one stock price path for $S_0 = 100$, $\sigma = 0.2$, $r = 0.05$, $T = 1$, $\Delta = \frac{1}{252}$

Click here to go back to section 3.3

8.2 BSM prices

8.2.1 European option

The price of an European call option with payoff $\psi(S_T) = (S_T - K)^+$ is given by the risk-neutral expectation:

$$V_0^{Euro} = \mathbb{E}^{\mathbb{Q}}\Big[\frac{(S_T - K)^+}{\beta_T}\Big] \tag{25}$$

$$= e^{-rT}\mathbb{E}^{\mathbb{Q}}(S_T - K)^+ \tag{26}$$

where $\beta_t = e^{rt}$ and r is the constant risk-free rate.

The closed-form solution of this equation is given by the Black-Scholes formula:

$$V_0^{Euro} = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2) \tag{27}$$

where:

$$d_1 = \frac{\ln S_0/K + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \qquad \text{and} \qquad d_2 = d_1 - \sigma\sqrt{T}$$

For a put option, we can find it using the put/call parity equation $P_t + S_t = C_t + Ke^{-r(T-t)}$ (Wilmott, 2006). Therefore:

$$V_0^{Euro,Put} = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1) \tag{28}$$

8.2.2 Geometric Asian option

Because the product of log-normally distributed random variables is also log-normally distributed, we can apply the Black-Scholes Merton framework to compute the closed-form equation for the price of a geometric Asian option with payoff $\psi(G(S)) = (G(S) - K)^+$.

Its price using the risk-neutral valuation is given by:

$$V_0^{Geo} = \mathbb{E}^{\mathbb{Q}}\Big[\frac{(G(S)-K)^+}{\beta_T}\Big] \tag{29}$$

$$= e^{-rT}\mathbb{E}^{\mathbb{Q}}(\Big[\prod_{j=1}^{n} S_{t_j}\Big]^{\frac{1}{n}} - K)^+ \tag{30}$$

for $j \in \{1, ..., n\}$ and $t_n = T$

The solution of this expectation is given by:

$$V_0^{Geo} = e^{-rT}\Big[S_0 e^{r^*}\Phi(d_1) - K\Phi(d_2)\Big] \tag{31}$$

where

$$d_1 = \frac{\ln S_0/K + (r^* + \sigma^{*2}/2)}{\sqrt{\sigma^{*2}}} \qquad \text{and} \qquad d_2 = d_1 - \sqrt{\sigma^{*2}}$$

with $\sigma^{*2} = \sigma^2\Big(\frac{2n+1}{6n+6}\Big)T$ and $r^* = \frac{\sigma^{*2}}{2} + (r - \frac{\sigma^2}{2})T$

The solution for the geometric Asian put option can be found using the call/put parity equation given in 8.2.1.

The full mathematical development can be found here: [Zhang, 2009].

Click here to go back to section 3.4.2