WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

IB9JH0
C++ for Quantitative Finace

Monte Carlo Method for One-Factor Option Pricing

Jón Sveinbjörn Halldórsson
1991391

March 27, 2020

Contents

1 Introduction

In this project, we will price some European and Asian options using two Monte Carlo methods and discuss how the framework can be extended to price other exotic options. The methods are coded in C++ with visualisations made from Python using the `MatPlotLib` and `Numpy` libraries. For reference, any CPU time stated in this report is measured with the `Stopwatch` class on a 2016 Dell Latitude E7470 with an Intel(R) Core(TM) i7-6600U CPU @ 2.60 GHz processor and 16.0 GB of RAM.

We will follow the standard notation and denote stock price with S, strike price with K, interest rates with r, volatility with σ and time to maturity with T.

2 Setup and assumptions

For this assignment, we will assume that the underlying asset is a geometric Brownian motion, i.e. S_t follows the stochastic differential equation

$$dS_t = S_t(r-d)dt + \sigma S_t dW_t$$

where $\sigma > 0$, r and d are constants and W_t is a Wiener process.

We will perform the following Monte Carlo methods and compare their accuracy to risk-neutral closed form results obtained by a Martingale pricing approach.

$$\text{Euler method: } \begin{cases} dS_t = S_t(r-d)dt + \sigma S_t dW_t \\ S_{t+\Delta t} = S_t + dS_t \end{cases}$$

$$\text{Exact simulation: } \begin{cases} X_0 = \log(S_0) \\ dX_t = \nu dt + \sigma dW_t; \quad \nu = r - d - \frac{\sigma^2}{2} \\ X_{t+\Delta t} = X_t + dX_t \\ S_t = \exp(X_t) \end{cases}$$

The option price, V_t, is calculated as the expected value of the discounted payoff

$$V_t = \exp(-rT)\,\mathbb{E}[h(S_T)] = \exp(-rT)\frac{1}{N}\sum_{i=1}^{M} h(S_{T_i})$$

where M is the number of simulations and h is the option's payoff function which can either depend on the final value of the asset price, S_T, or the entire path, e.g. Asian options or barrier options.

3 Numerical analysis

3.1 Standard deviations and variance reduction

The standard deviation and error of the option prices for both Monte Carlo methods is calculated with

$$\text{Standard deviation: } \quad \hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{M} h(S_{T_i})^2}{M} - \left\{\frac{\sum_{i=1}^{M} h(S_{T_i})}{M}\right\}^2}$$

$$\text{Standard error: } \quad \frac{\hat{\sigma}}{\sqrt{M}}$$

The standard deviation of the option prices was reduced by using the antithetic variates technique. Since the path of the Wiener process has normally distributed increments with mean zero and a standard deviation of $\sqrt{dt}$, we know that the same applies to the antithetic path, i.e. the negated values of W_t. We therefore simulate the following

$$\text{Euler method:}\ \begin{cases} dS_t^{\pm} = S_t(r-d)dt \pm \sigma S_t dW_t \\ S_{t+\Delta t} = S_t + dS_t^{\pm} \end{cases}$$

$$\text{Exact simulation:}\ \begin{cases} X_0 = \log(S_0) \\ dX_t^{\pm} = \nu dt \pm \sigma dW_t; \quad \nu = r - d - \frac{\sigma^2}{2} \\ X_{t+\Delta t} = X_t + dX_t^{\pm} \\ S_t = \exp(X_t) \end{cases}$$

and observe that the mean option price remains the same

$$\mathbb{E}[V_{t_i}] = \exp(-rT)\,\mathbb{E}\left[\frac{h(S_{T_i}^{+}) + h(S_{T_i}^{-})}{2}\right] = \exp(-rT)\,\mathbb{E}[h(S_T)]$$

whereas the variance is significantly reduced (at least halved) due to the negative covariance between $h(S_T^+)$ and $h(S_T^-)$

$$\begin{aligned} \text{Var}[V_{t_i}] &= \text{Var}\left[\frac{h(S_{T_i}^{+}) + h(S_{T_i}^{-})}{2}\right] \\ &= \frac{\text{Var}[h(S_{T_i}^{+})] + \text{Var}[h(S_{T_i}^{-})] + 2\text{Cov}[h(S_{T_i}^{+}, h(S_{T_i}^{-})]}{4} \\ &= \frac{\text{Var}[h(S_{T_i})]}{2} + \frac{\text{Cov}[h(S_{T_i}^{+}), h(S_{T_i}^{-})]}{2} \end{aligned}$$

3.2 Finite difference methods and path recycling

To evaluate the delta and the gamma of the options, we used the explicit Euler method

$$\begin{aligned} \Delta &= \frac{\partial V_t}{\partial S_t} = \frac{V_{j+1} - V_{j-1}}{2dS_t} + \mathcal{O}((dS_t)^2) \\ \Gamma &= \frac{\partial^2 V_t}{\partial S_t^2} = \frac{V_{j+1} - 2V_j + V_{j-1}}{dS_t^2} + \mathcal{O}((dS_t)^2) \end{aligned}$$

for $j = 1, \ldots, M-1$. The variance of the two greeks is calculated using

$$\begin{aligned} \text{Var}\left(\sum_{i=1}^{N} a_i X_i\right) &= \sum_{i=1}^{N} a_i^2 \text{Var}[X_i] + 2\sum_{1\le i<j\le N}^{N} a_i a_j \text{Cov}(X_i, X_j) \\ &= \sum_{i=1}^{N} a_i^2 \text{Var}[X_i] + 2\sum_{1\le i<j\le N}^{N} a_i a_j \sqrt{\text{Var}[V_i]\text{Var}[V_j]}\text{Corr}(X_i, X_j) \end{aligned}$$

and the following approximation: $\text{Var}[V_{j+1}] \approx \text{Var}[V_j] \approx \text{Var}[V_{j-1}]$.

By using the same M paths for the Wiener processes when calculating the price of an option for a given stock price, we ensure that the values of V_{j+1}, V_j, and V_{j-1} are not obtained by independent Monte Carlo simulations. Hence they will be dependent on one another and have a positive correlation, resulting in reduced variance.

The main problem with this method is that the values tend to blow up as dS_t approaches zero. A safer alternative could be to use the pathwise method instead of the finite difference method.

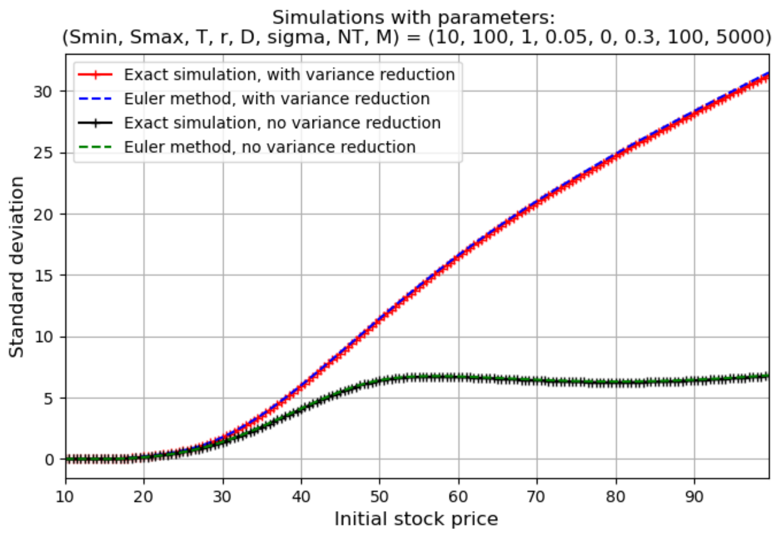

Figure 3.1: Demonstration of variance reduction for a European call option with strike $K = £50$

3.3 User defined accuracy

To calculate the minimum number of simulations, N, needed for some user defined accuracy ϵ and some user defined confidence level $1-\alpha$, it is first necessary to perform M Monte Carlo simulations and observe the maximum standard deviation over the range of stock prices, σ_{MC}. Afterwards, the number of simulations required is

$$||V_{\text{MC}} - V||_\infty = \epsilon \geq \Phi^{-1}\left(1-\frac{\alpha}{2}\right)\frac{\sigma_{\text{MC}}}{\sqrt{M}} \Rightarrow N \geq \left(\Phi^{-1}\left(1-\frac{\alpha}{2}\right)\frac{\sigma_{\text{MC}}}{\epsilon}\right)^2$$

where Φ is the normal cumulative distribution function. This ensures that the price of the option for all stock prices in a given range is $100(1-\alpha)\%$ accurate to within $£\epsilon$.

4 Option pricing

For both methods, the Wiener processes are generated and stored in a matrix for path recycling. They are then used to generate two matrices of stock price paths, S^+ and S^-, for initial stock prices ranging from $S_{\max}$ to $S_{\min}$ with increments dS_t.

4.1 Option payoffs

After the stock price paths have been generated, each vector is sent into the `OptionData` class to calculate the payoff, whether it is a path-dependent or not. The input of the payoff function is a vector and the output depends on the type of option.

$$\text{European options:} \begin{cases} \text{Call payoff:} & \max(S_T - K, 0) \\ \text{Put payoff:} & \max(K - S_T, 0) \end{cases}$$

$$\text{Asian options:} \begin{cases} \text{Arithmetic call payoff:} & \max\left(\frac{1}{N}\sum_{i=1}^{N} S_i - K, 0\right) \\ \text{Arithmetic put payoff:} & \max\left(K - \frac{1}{N}\sum_{i=1}^{N} S_i, 0\right) \\ \text{Geometric call payoff:} & \max\left(\sqrt[N]{\prod_{i=1}^{N} S_i} - K, 0\right) \\ \text{Geometric put payoff:} & \max\left(K - \sqrt[N]{\prod_{i=1}^{N} S_i}, 0\right) \end{cases}$$

The code is structured to allow for additional option types to be easily added. To this end, the option's payoff function should be added to the `OptionData::payoff` function, a closed form solution can be added for the option's price, delta and gamma in `OptionCommand` as derived classes (optional, only for accuracy comparison), and finally the `FairValue` constructor should reference the right pointers.

4.2 Comparison to closed form solutions

To compare the accuracy of our Monte Carlo simulations, we implemented the closed form solutions into a derived class in `OptionCommand`. For the European options, we used the Black Scholes formulas

$$\text{European call:} \begin{cases} \text{Price:} & C(S_t,t) = S_t \exp(-q(T-t))N(d_1) - K\exp(-r(T-t))N(d_2) \\ \text{Delta:} & \Delta_C = \exp(-q(T-t))N(d_1) \\ \text{Gamma:} & \Gamma_C = \exp(-q(T-t))N'(d_1) = \dfrac{\exp(-q(T-t))}{S_t\sigma\sqrt{T-t}\sqrt{2\pi}}\exp\left(\dfrac{-d_1^2}{2}\right) \end{cases}$$

$$\text{European put:} \begin{cases} \text{Price:} & P(S_t,t) = K\exp(-r(T-t))N(-d_2) - S_t\exp(-q(T-t))N(-d_1) \\ \text{Delta:} & \Delta_P = \exp(-q(T-t))(N(d_1)-1) \\ \text{Gamma:} & \Gamma_P = \exp(-q(T-t))N'(d_1) = \dfrac{\exp(-q(T-t))}{S_t\sigma\sqrt{T-t}\sqrt{2\pi}}\exp\left(\dfrac{-d_1^2}{2}\right) \end{cases}$$

where q is a continuously compounded dividend yield and d_1 and d_2 are

$$\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left(\log\left(\frac{S_t}{K}\right) + (r-q+\frac{\sigma^2}{2})(T-t)\right) \\ d_2 &= d_1 - \sigma\sqrt{T-t} \end{aligned}$$

Arithmetic Asian options were priced using the Turnbull and Wakeman approximation[1] whereas geometric Asian options can be shown[2] to have the following closed form solution

$$\text{Geometric Asian call:} \begin{cases} \text{Price:} & C_{\text{GAC}}(S_t,t) = S_t\exp((b-r)(T-t))N(d_1) - K\exp(-r(T-t))N(d_2) \\ \text{Delta:} & \Delta_{\text{GAC}} = \exp((b-r)(T-t))N(d_1) \\ \text{Gamma:} & \Gamma_{\text{GAC}} = \exp((b-r)(T-t))N'(d_1) = \dfrac{\exp((b-r)(T-t))}{S_t\sigma_G\sqrt{T-t}\sqrt{2\pi}}\exp\left(\dfrac{-d_1^2}{2}\right) \end{cases}$$

$$\text{Geometric Asian put:} \begin{cases} \text{Price:} & P_{\text{GAP}}(S_t,t) = K\exp(-r(T-t))N(-d_2) - S_t\exp((b-q)(T-t))N(-d_1) \\ \text{Delta:} & \Delta_P = \exp((b-r)(T-t))(N(d_1)-1) \\ \text{Gamma:} & \Gamma_P = \exp((b-q)(T-t))N'(d_1) = \dfrac{\exp((b-q)(T-t))}{S_t\sigma_G\sqrt{T-t}\sqrt{2\pi}}\exp\left(\dfrac{-d_1^2}{2}\right) \end{cases}$$

[1] 1997, Espen Gaarder Haug, The Complete Guide to Option Pricing formulas

[2] `https://en.wikipedia.org/wiki/Asian_option#European_Asian_call_and_put_options_with_geometric_averaging`

where

$$\sigma_G = \frac{\sigma}{\sqrt{3}}, \qquad b = \frac{1}{2}\left(r - q - \frac{1}{2}\sigma_G^2\right)$$
$$d_1 = \frac{1}{\sigma_G\sqrt{T-t}}\left(\log\left(\frac{S_t}{K}\right) + (b + \frac{\sigma^2}{2})(T-t)\right)$$
$$d_2 = d_1 - \sigma_G\sqrt{T-t}$$

The Turnbull-Wakeman approximation did not produce an accurate enough result so the arithmetic Asian options will not be measured and analysed in this report.

5 Software design

The code was written with the single responsibility principle in mind. Each class was given a single purpose and then the main functionality was implemented in `MonteCarlo`. When applicable, a Standard Template Library (STL) algorithm or data structure was used and the main option parameters were stored in `OptionData`, which uses`boostparameter.hpp`.

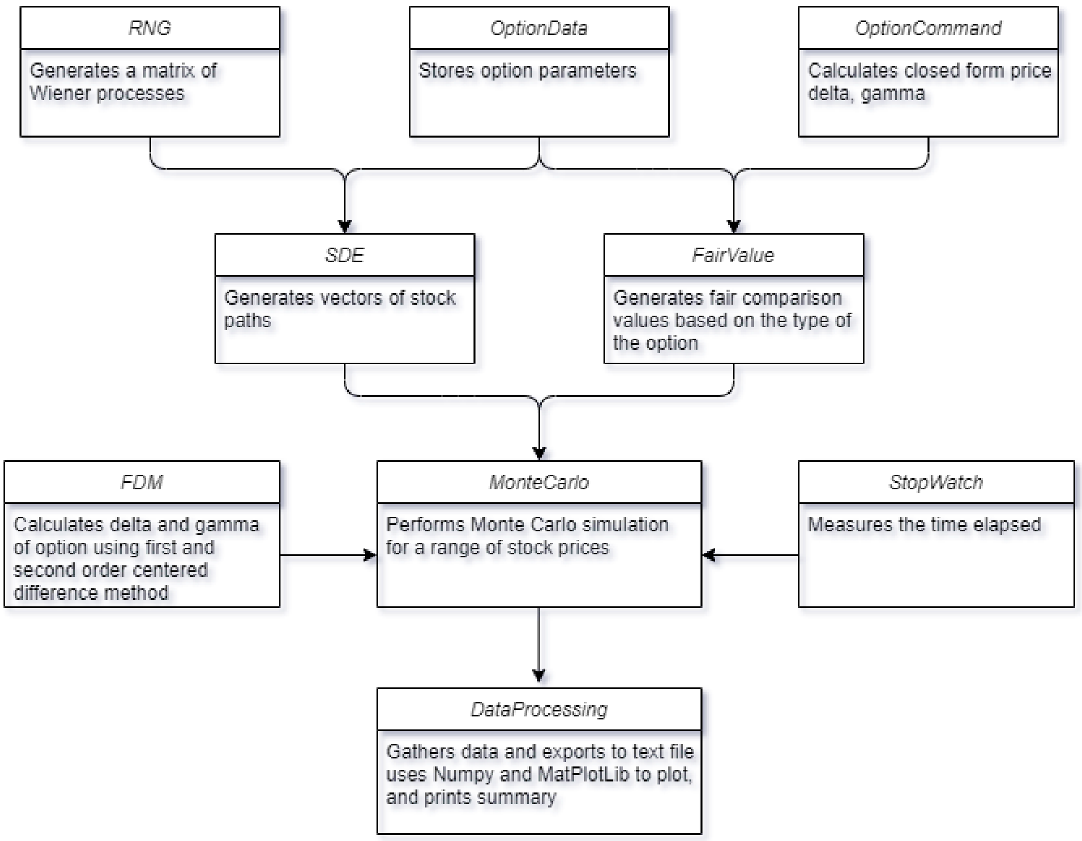
RNG
Generates a matrix of Wiener processes
OptionData
Stores option parameters
OptionCommand
Calculates closed form price delta, gamma
SDE
Generates vectors of stock paths
FairValue
Generates fair comparison values based on the type of the option
FDM
Calculates delta and gamma of option using first and second order centered difference method
MonteCarlo
Performs Monte Carlo simulation for a range of stock prices
StopWatch
Measures the time elapsed
DataProcessing
Gathers data and exports to text file uses Numpy and MatPlotLib to plot, and prints summary

Figure 5.1: Class diagram showing the collaboration among the elements of the projects and describing the functionalities performed by each system.

6 Results

For the simulations, we will use options with the following parameters and take 100 time steps:

$$K = \pounds 25.0, \quad T = 1.0, \quad r = 3\%, \quad \sigma = 20\%, \quad q = 0.0\%$$

6.1 Time and error measurements

The time elapsed and maximum error was measured in `Test_measurements.cpp` for both a European call option and a Geometric Asian call option. The Turnbull-Wakeman approximation was not measured.

	European call option				Geometric Asian call option			
	Time		Max error		Time		Max error	
N	Euler	Exact	Euler	Exact	Euler	Exact	Euler	Exact
10	0.02	0.02	7.7127	7.7252	0.01	0.01	3.7328	3.7490
20	0.03	0.05	3.6936	3.7318	0.03	0.02	1.2695	1.2950
40	0.07	0.12	0.3240	0.3756	0.09	0.04	0.3338	0.3154
80	0.13	0.20	0.0897	0.0867	0.13	0.07	0.2815	0.2677
160	0.26	0.31	1.0019	1.0153	0.25	0.18	0.1677	0.1720
320	0.39	0.70	0.3480	0.3530	0.54	0.32	0.1411	0.1409
640	1.37	1.44	0.6743	0.6770	0.85	0.68	0.1081	0.1091
1,280	1.50	1.88	0.3857	0.3893	1.75	1.09	0.1359	0.1368
2,560	2.47	3.91	0.1989	0.2011	3.90	2.31	0.0865	0.0873
5,120	5.24	7.38	0.1636	0.1638	7.24	4.21	0.1105	0.1098
10,240	10.19	15.30	0.0213	0.0201	14.43	7.89	0.0379	0.0379
20,480	15.19	31.87	0.0272	0.0279	30.33	16.37	0.0262	0.0258
40,960	45.29	62.91	0.0232	0.0236	65.57	42.92	0.0231	0.0225
81,920	87.35	120.77	0.0332	0.0328	108.59	67.58	0.0299	0.0301
163,840	117.25	181.45	0.0062	0.0063	159.16	229.55	0.0071	0.0072
327,680	314.52	524.78	0.0051	0.0051	329.12	587.69	0.0061	0.0061

Table 1: Time and maximum error measurements for the two methods measured in seconds and pounds respectively.

6.2 Time complexity analysis

We assume that the relationship between the number of simulations for a fixed number of time steps, N, and time elapsed is $T(N) = \alpha N^{\beta}$. We then take the natural logarithm of both sides and fit a best line through it to find α and β. These calculations were carried out in `time_and_error_analysis.py`. The results were as follows:

$$\begin{aligned} \text{European Euler:} \quad & T(N) = 0.002279 N^{0.9171} \\ \text{European exact:} \quad & T(N) = 0.002757 N^{0.9404} \\ \text{Asian Euler:} \quad & T(N) = 0.001956 N^{0.9599} \\ \text{Asian exact:} \quad & T(N) = 0.000916 N^{1.0119} \end{aligned}$$

6.3 Comparison

The Euler method and the exact simulation produced similarly accurate results but the Euler method was faster on average, as was expected since it required more steps to calculate.

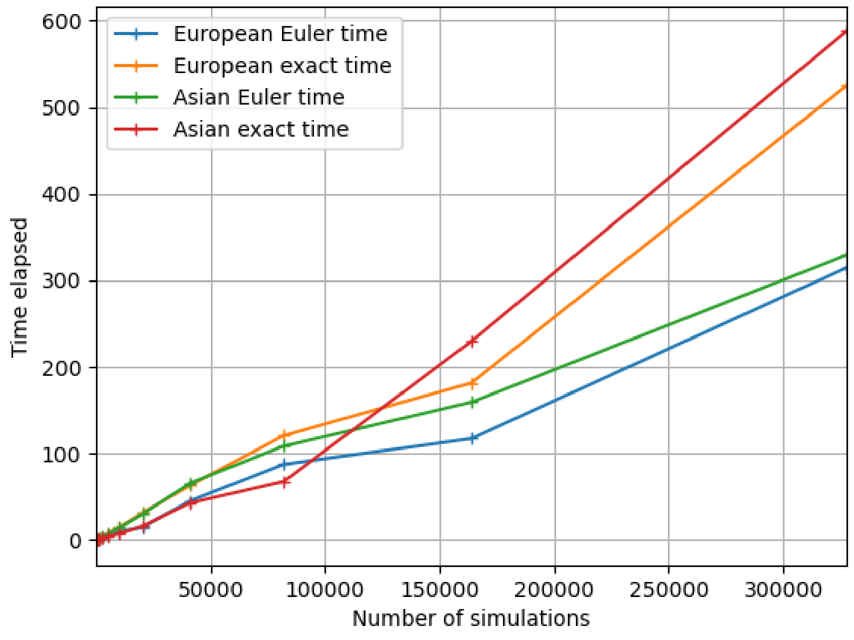

Figure 6.1: todo

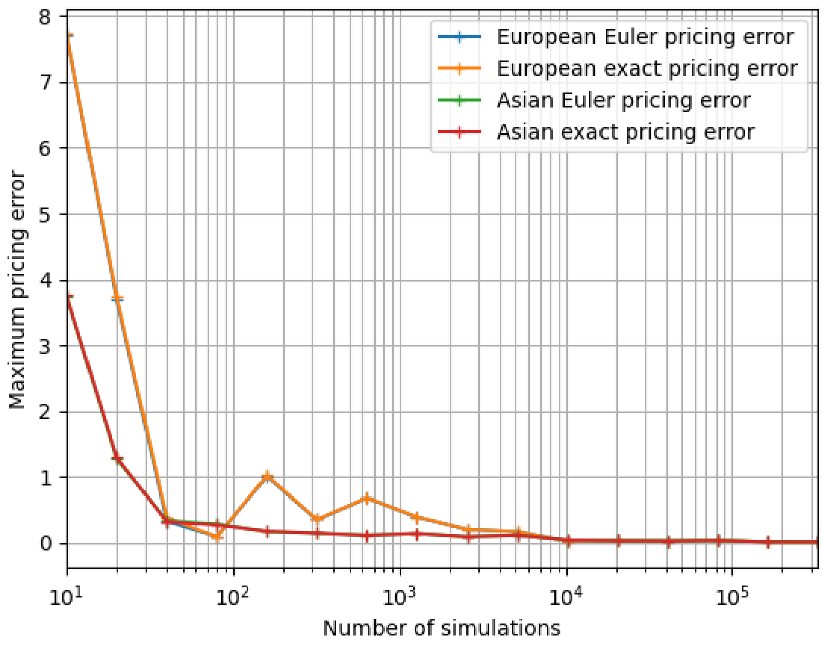

Figure 6.2: todo

A European call option

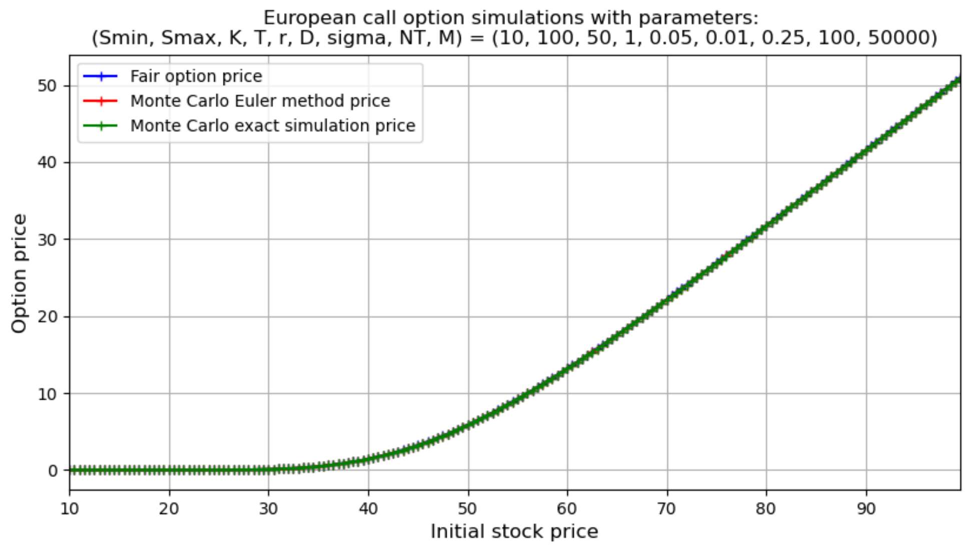

Figure A.1: Price of a European call option with initial stock price ranging from £10 to £100.

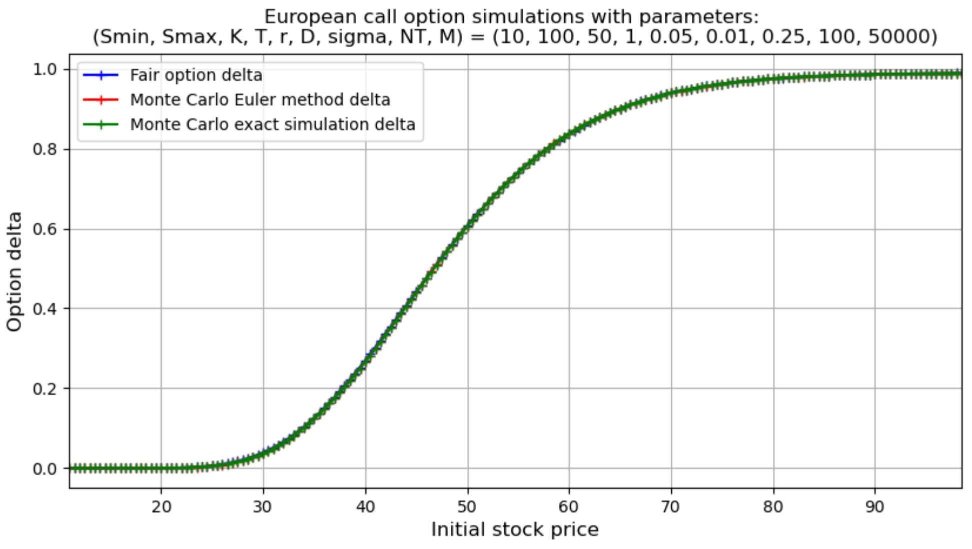

Figure A.2: Delta of a European call option with initial stock price ranging from £10 to £100.

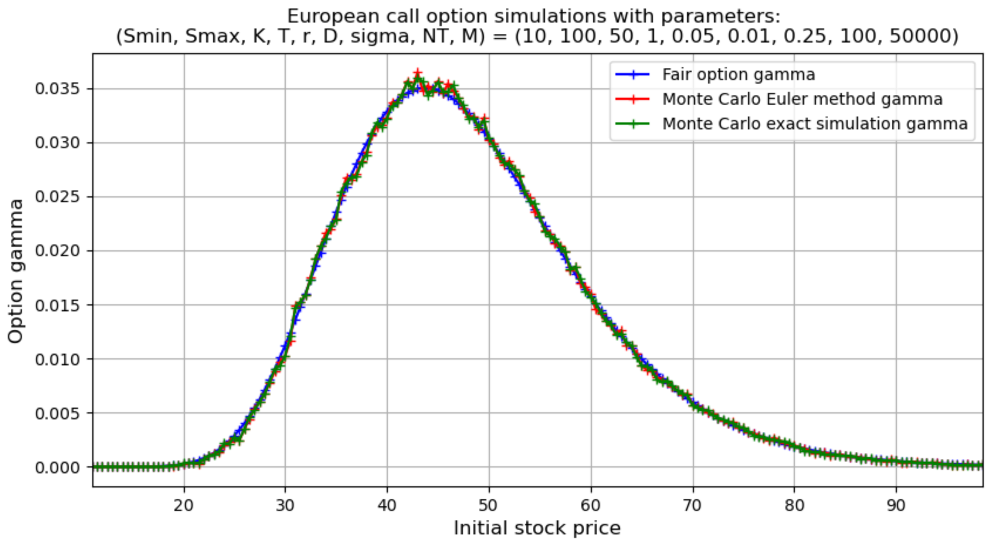

Figure A.3: Gamma of a European call option with initial stock price ranging from £10 to £100.

B Geometric Asian put option

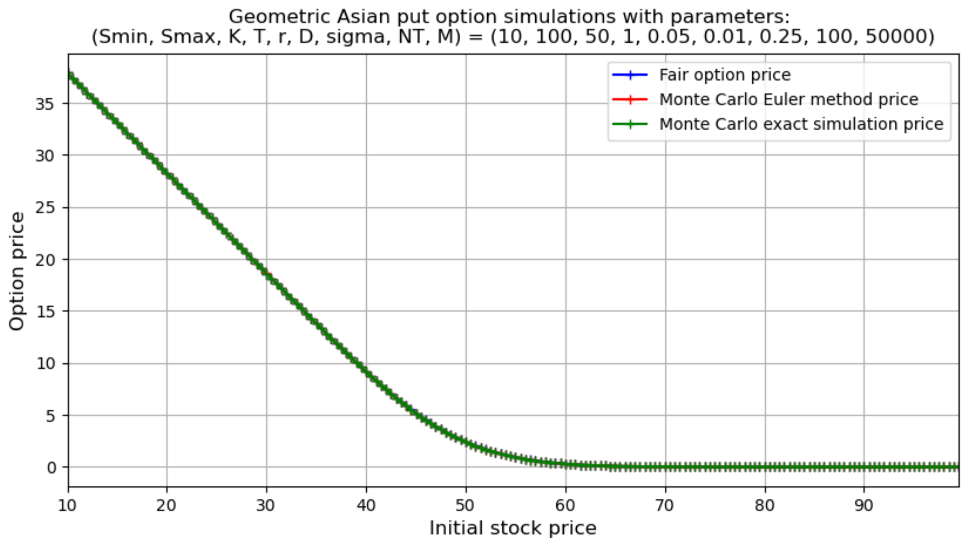

Figure B.1: Price of a Geometric Asian call option with initial stock price ranging from £10 to £100.

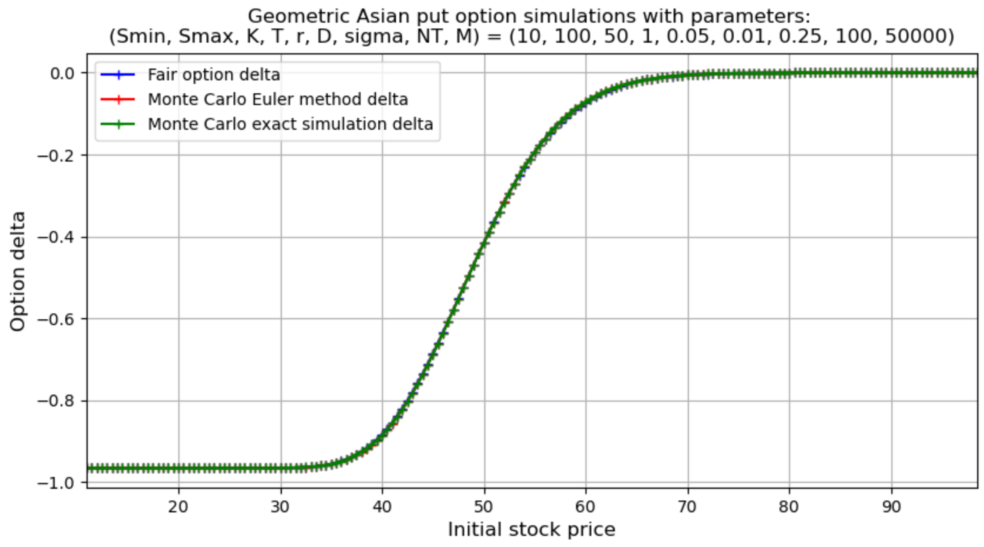

Figure B.2: Delta of a Geometric Asian call option with initial stock price ranging from £10 to £100.

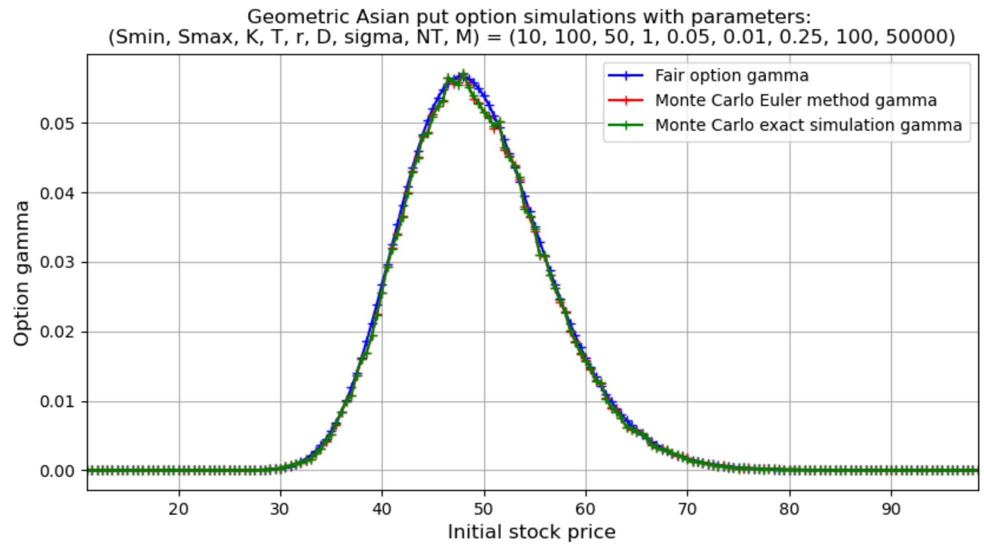

Figure B.3: Gamma of a Geometric Asian call option with initial stock price ranging from £10 to £100.

List of Tables

List of Figures