FM442 – Quantitative methods for finance & risk analysis

Monte Carlo pricing of path dependent and currency options

London School of Economics and Political Sciences

Candidate number: 15200

Contents

1 Introduction

A derivative is a contract, whose value *derives from* an underlying asset, e.g. stock, commodity or exchange rates. The most commonly traded derivatives are options, which give the holder the right but not the obligation to buy or sell the underlying at a predetermined strike price K at or prior to a future time T [1]. Options are one of the most versatile contracts in the financial world, since they offer traders very few restrictions when devising strategies, making them very prominent in capital markets.

Such is their relevance that in the late eighties, financial engineers started to fabricate new securities which would latter be given the name of *exotic options* [2]. Due to their complexity, they were rarely traded in exchanges, but instead were more frequent in over-the-counter (OTC) markets. Furthermore, this decade was heavily marked by the Jamaica agreement of 1976 that put an end to the Bretton Woods system. Combined with a fast growing economic globalisation, countries started relaxing their restrictions on foreign investors making the global markets more volatile [3]. This increased heavily the demand for foreign exchange derivatives, which developed rapidly and is the largest traded market in volume nowadays.

In this project, both exotic and foreign exchange options will be studied. Since they can be very convoluted contracts, often one has to resort to numerical methods and simulation techniques to price these options. Monte Carlo methods will be used to price these contracts, and variance reduction techniques will be employed to improve the precision of the estimation. In particular for this project, antithetic variables, moment matching and quasi-Monte Carlo methods will be implemented throughout.

2 Path dependent options: Barrier and Asian options.

The first traded *exotic* dates back to 1987, when *Bankers Trust* developed a pricing formula for an option which involved the average price of crude oil over a certain period [4]. Since they were in Tokyo at the time, this contract was given the name of *Asian option*. The main difference between exotic options compared to their *vanilla* counterparts, is that their option's price depends on the entire path of the underlying during the contract lifetime. For this reason, they are often also known as path-dependent options.

2.1 Choice of underlying asset

As previously mentioned, most exotic options are rarely traded in exchanges. Also, most such options are in fact written for crude oil or other commodities, and thus, it is hard to find publicly listed exotics with relatively high trading volumes. Therefore, only fictitious options will be analysed for this project, i.e. contracts that do not actually exists, where the Black-Scholes parameters are chosen arbitrarily.

However, to construct a more realistic security, historical data for the last 30 years of Brent crude prices was analysed [5]. The annualised historical volatility was found to be $\sigma = 0.41$ and used as the volatility parameter in the Black-Scholes model. The current price of Brent crude $S_0 = 54.84\$$ was also used as spot price. The risk-free rate was chose to be $r = 0.13\%$, since this is the last quoted rate in the United States by the international monetary fund (IMF) data [6]. The remaining parameters, namely the option maturity T and strike price K, were chosen arbitrarily to be $T = 0.25$ (3 months) and $K = 58\$$.

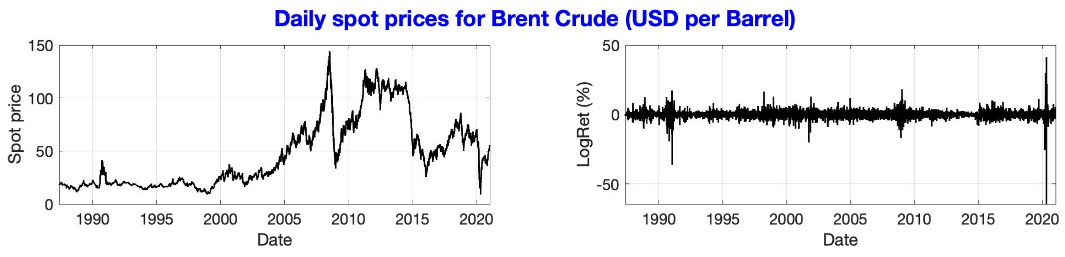


Figure 1: Daily prices and log returns for Brent crude oil (in USD per barrel) from 21/05/1987 - 07/01/2021.

2.2 Barrier options

Barrier options were first priced by Robert Merton in 1973. These type of contracts become activated (*knocked in*) or extinguished (*knocked out*) if the stock price exceeds or falls below a certain threshold (the barrier) at any point during the option's lifetime.

For example, in a down-and-out call option, the spot price starts above the barrier level and if the stock price falls below it, the option becomes worthless. These options have a payoff given by:

$$F(S_T) = \begin{cases} \max\{S_T - K, 0\} & S_t > B \\ 0 & S_t \leq B \end{cases} = (S_T - K)^+ \mathbb{1}_{\{S_t > B\}}$$

where $\mathbb{1}_{(\cdot)}$ is the indicator function. Pricing barrier options analytically is somewhat complicated and is done for example here [7]. A down-and-out barrier option on Brent crude will be considered with barrier level $B = 52$.

2.3 Asian options

Asian options are options whose value derives from the average underlying price over a certain period. There are two main ways in which the underlying's price can be averaged: arithmetically or geometrically. In the continuous case, the arithmetic average A_a and the geometric average A_g are given by:

$$A_a = \frac{1}{T}\int_0^T S_t dt \qquad A_g = \exp\left[\frac{1}{T}\int_0^T \log S_t dt\right]$$

A closed form solution exists for geometrically averaged Asian options [8], for example an Asian call with payoff:

$$F(S_T) = \max\left\{e^{\frac{1}{T}\int_0^T \log S_t dt} - K, 0\right\} = \left(e^{\frac{1}{T}\int_0^T \log S_t dt} - K\right)^+$$

has a price process:

$$V_t = S_t e^{(b-r)(T-t)}\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2) \tag{1}$$

where:

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

2.4 Miss-pricing errors in path-dependant options

One of the limitations of pricing path-dependant options is that the analytic solution assumes continuous monitoring of the underlying. Since Monte Carlo methods only allow us to observe the stock path discretely, notable miss-pricing errors arise. Therefore, there are two key parameters in the Monte Carlo estimation:

- N: The total number of stock paths simulated.
- Q: The number of times the stock is realised during the option lifetime. Most analytic solutions assume the stock is monitored continuously (i.e they are written for the limit $Q \to \infty$).

Q	**N**	**Stock observation frequency (min^{-1})**	**Option Price($)**	**Relative Error (%)**	**Run Time (s)**
1,000		130	1.973	11.2	0.002
10,000	10,000	13	1.839	3.62	0.020
100,000		1.3	1.750	1.39	0.342

Table 1: Down-and-out call option prices at different Q. The total number of paths N was held fixed. The analytic option price was calculated using the `barrierbybls` built-in function on Matlab to be: $1.775.

Increasing either one of the two parameters should make the approximation better: Increasing Q minimises the miss-prising errors due to the discretisation of the stock path, while increasing N reduces the variance of the Monte Carlo samples. This effect is studied in table 1, where N is held fixed and the option price is estimated at different Q.

2.5 Brownian bridge barrier option pricing

For barrier options, there exists an efficient way around the aforementioned miss-pricing error: the so-called **Brownian bridge method**. Brownian bridges describe the conditional probability distribution of a Brownian motion, so that one can then simply simulate the terminal stock price and multiply the option payoff by the probability that the underlying crosses the barrier.

For a down-and-out-call, assume the spot price S_0 and terminal price S_T of the underlying are known. If the terminal price is above the barrier level $S_T > B$, the probability that the minimum of stock price S_t along its entire path $t \in [0, T]$ exceeds the barrier B is:

$$P\left[\min_{0<t<T} S(t) > B\right] = 1 - \exp\left(\frac{-2\log(B/S_0)\log(B/S_T)}{\sigma^2 T}\right)$$

[9]. This equation gives us the probability that the stock price remains strictly above the barrier level B throughout the option lifetime. If $S_T \leq B$, by continuity of the geometric Brownian motion, the option must have become knocked out at some point $t < T$. Thus, we can define:

$$p = \begin{cases} 1 - \exp\left(\frac{-2\log(B/S_0)\log(B/S_T)}{\sigma^2 T}\right) & S_T > B \\ 0 & S_T \leq B \end{cases}$$

such that $p \cdot \max\{S_T - K, 0\}$ gives the correct payoff for the down-and-out European call. This method not only eliminates the time discretisiation error entirely, but is also much more computationally efficient, since only the terminal stock value needs to be simulated rather than the entire path.

2.6 Results

2.6.1 Brownian Bridge down-and-out call:

Figure 2 shows the 95% confidence level intervals, as well as the run time and variance for 3 samples: Crude Monte Carlo, quasi Monte Carlo and antithetic variables + moment matching sample.

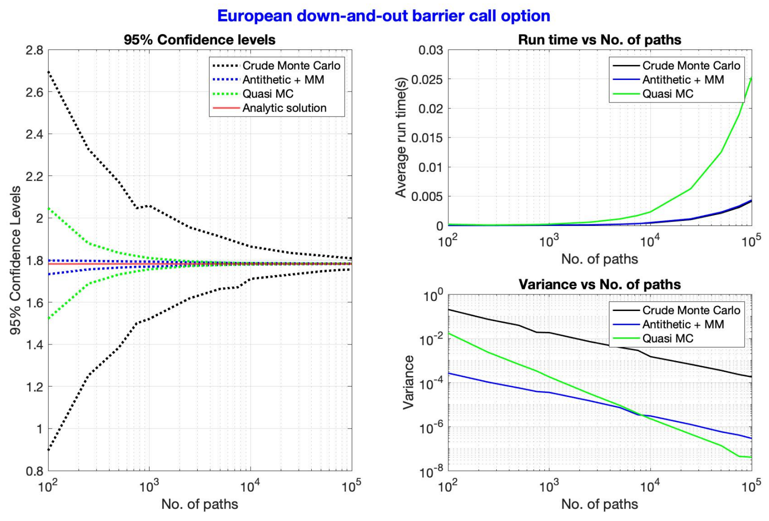


Figure 2: 95% Confidence level, run time and variance vs number of paths for three samples: crude Monte Carlo (black), quasi Monte Carlo (green) and antithetic variables + moment matching reduced (blue).

As expected, the crude Monte Carlo sample has the largest variance overall. The antithetic + moment matching sample starts at very low variance ($\mathcal{O}(10^{-4})$ for $N = 100$ paths), and is only overtaken by the quasi Monte Carlo sample after $\sim$ 7500 paths, point at which the computational time for the quasi MC is already 4x larger that the antithetic + moment matching. Therefore, the additional computational cost does not compensate for the better variance reduction.

2.6.2 Geometrically averaged Asian option

Table 2 shows the option value and the variance through the different VRTs for a geometrically averaged Asian option for Brent crude. Since Brownian bridge methods cannot be used, the entire stock paths need to be simulated, leading to discretisation errors. Increasing N decreases the variance, while increasing Q gets the option price closer to the analytic solution: \$1.2866, calculated via equation 1.

Q	N	Option Value (\$)			Variance		
		Crude	**AV+MM**	**Quasi MC**	**Crude**	**AV+MM**	**Quasi MC**
50	100	1.2628	1.2756	1.2545	0.0663	0.0566	0.0771
	1,000	1.2445	1.2512	1.2482	0.0077	0.0091	0.0054
	10,000	1.2565	1.2517	1.2510	0.0010	0.0006	0.0006
200	100	1.2741	1.3103	1.2843	0.0902	0.0593	0.0918
	1,000	1.2908	1.2693	1.2740	0.0092	0.0068	0.0069
	10,000	1.2777	1.2745	1.2796	0.0009	0.0008	0.0007
500	100	1.2766	1.2813	1.2757	0.0707	0.0631	0.0797
	1,000	1.2834	1.2806	1.2666	0.0085	0.0076	0.0076
	10,000	1.2832	1.2883	1.2844	0.0011	0.0007	0.0007

Table 2: Option prices and variances for geometrically averaged Asian option on Brent crude through different variance reduction techniques.

We can already see from the table how the variance reduction through the different techniques "only" ranges from 10-40% reduction with respect to the crude MC samples. This proves that simulating the entire path, combined with the discretisation errors, truly hinders the power of these methods. Again, 95% confidence levels are drawn in figure 3, as well as the variance and run time vs N. On the other hand, because simulating the stock path is time more time consuming, the run time discrepancy between the 3 samples is less significant, specially for the quasi MC method.

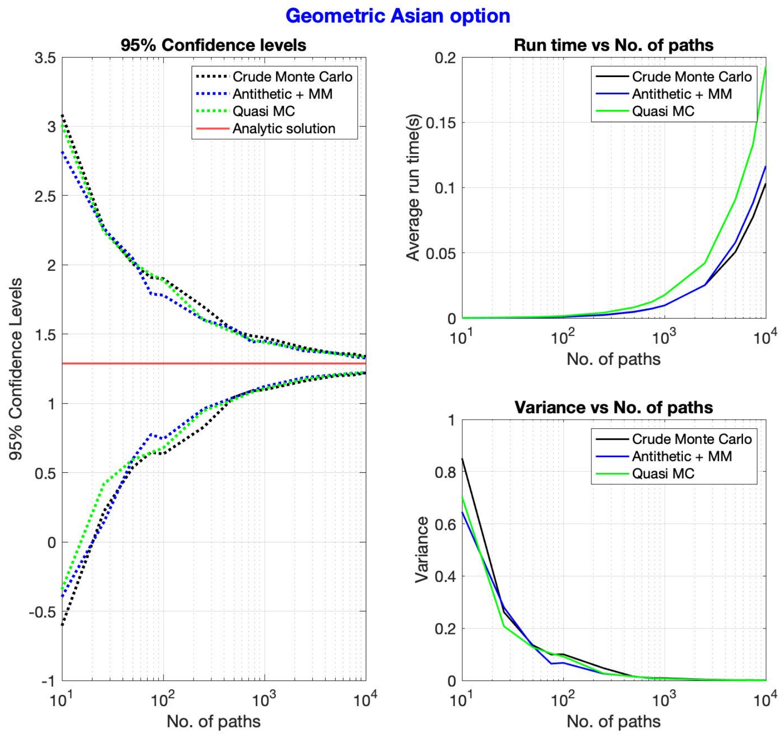


Figure 3: 95% confidence levels for the geometrically averaged Asian option realised on $Q = 200$ instances throughout it's lifetime.

3 Foreign exchange options

3.1 Modelling exchange rate

Similarly to equities, exchange rates can be modelled mathematically and used as an underlying in derivative contracts. These instruments are particularly relevant to multinational companies, when hedging against foreign exchange risk, but can also be used by speculators and arbitrageurs. If we denote by X_t the price at time t of one unit of the foreign currency in units of the domestic, then X_t follows the GBM stochastic differential equation:

$$dX_t = \mu_X X_t dt + X_t \sigma_X dW_t$$

where μ_X and σ_X are the exchange rate drift and volatility respectively. We can find a probability measure $\mathbb{Q}^d$ under which the discounted price process $e^{-r_d} X_t$ (where r_d is the domestic interest rate) is a local martingale [10]. Under this measure, the exchange rate process is governed by:

$$X_t = X_0 \exp\left\{ \left(r_d - rf - \frac{1}{2}\sigma_X^2 \right) t + \sigma_X W_t^{\theta_d} \right\} \tag{2}$$

where $W_t^{\theta_d}$ is a standard Brownian motion with respect to $\mathbb{Q}^d$ and r_f is the foreign interest rate. Using the above result, one can show that the dynamics for a foreign stock price S_t^f with volatility σ_f are given by:

$$S_t^f = S_0^f \exp\left\{ \left(r_f - \sigma_f \sigma_X - \frac{1}{2}\sigma_f^2 \right) t + \sigma_f W_t^{\theta_d} \right\}. \tag{3}$$

Similarly, the price of a foreign stock in units of the domestic currency is:

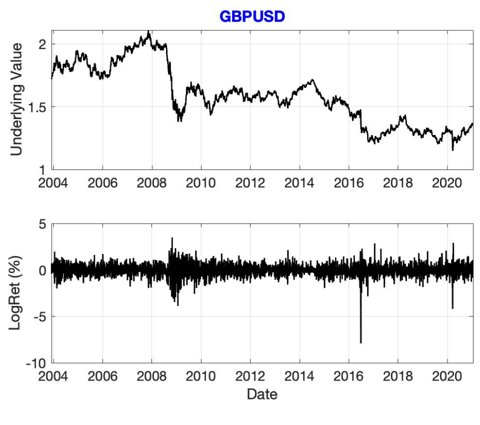
$$X_t S_t^f = X_0 S_0^f \exp\left\{ \left(r_d - \frac{1}{2}(\sigma_f^2 + \sigma_X^2) \right) t + \sqrt{\sigma_f^2 + \sigma_X^2}\, W_t^{\theta_d} \right\}. \tag{4}$$

These processes can be used to price currency options, options on foreign stocks, quantity-adjusting (quanto) options, etc. Examples of such securities will be priced by simulation, and compared to their analytical solutions.

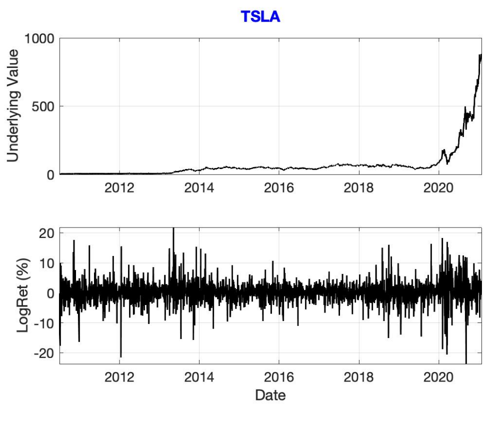
3.2 Choice of exchange rate and underlying asset

Again, only “fictitious” contracts will be considered since most of the options studied below are mathematical constructs which are rarely traded. The domestic and foreign currencies used are pound sterling and U.S. dollar respectively, with a spot rate of 1.3684 as of Jan. 15, 2021. The exchange rate volatility was estimated from the historical volatility form the past 18 years of trading data.

Foreign equity options will be considered for *Tesla Inc.* currently trading at $883.9 as of Jan. 26, 2021. The Black-Scholes volatility used was estimated through historical volatility from the past 10 years of trading data. Domestic and foreign risk-free rates were extracted once again from the international monetary fund (IMF) data.



(a) GBP-USD exchange rate data and returns from 01/12/2003 - 15/01/2021. Historical volatility estimate for the period: 9.56%



(b) TSLA stock price (in USD) and returns from 29/06/2010 - 26/01/2021. Historical volatility estimate for the period: 56.1%

3.3 Foreign equity options struck on domestic currency

These are options written on a foreign stock (i.e. price of TSLA in USD) whose strike price is expressed in terms of the domestic currency (i.e. GBP). For a call option, the payoff is given by:

$$F = \max\{X_t S_T^f - K_d, 0\} = (X_t S_T^f - K_d)^+$$

where X_t is the GBP-USD exchange rate, K_d is the strike price in GBP and S_T^f is the stock value at maturity in USD. Equation 4 can be used to price the option analytically, such that:

$$C(t) = X_0 S_0^f \cdot \Phi(d_1) - K_d e^{-r_d(T-t)} \cdot \Phi(d_2)$$

and:

$$d_1 = \frac{\log(X_0 S_0^f / K_d) + (r_d + 0.5(\sigma_X^2 + \sigma_{Sf}^2))(T-t)}{\sqrt{(\sigma_X^2 + \sigma_{Sf}^2)(T-t)}} \qquad d_2 = d_1 - \sqrt{(\sigma_X^2 + \sigma_{Sf}^2)(T-t)}$$

where σ_X and σ_{Sf} denote the volatility of the exchange and the foreign stock respectively.

For an arbitrarily chosen strike of £850, Monte Carlo solutions were implemented and the results are given in figure 5 (a).

3.4 Quantity adjusting (Quanto) options

In a quantity adjusting option, also known as guaranteed exchange rate option, the payoff in the foreign currency is converted back to the domestic by a pre-specified exchange rate $\overline{X}$. These options are relevant for investors who speculate a foreign asset will perform well, but fear that country's currency will not [11]. A quanto call yields the payoff:

$$F = \overline{X} \max\{S_T^f - K_f, 0\} = \overline{X}(S_T^f - K_f)^+.$$

Using the dynamics from equation 3, one can find that the option value is:

$$C(t) = \overline{X} S_0^f e^{(r_f - r_d - \sigma_{Sf}\sigma_X)(T-t)} \Phi(d_1) - \overline{X} K_f e^{-r_d(T-t)} \Phi(d_2)$$

and:

$$d_1 = \frac{\log(S_0^f / K_f) + (r_f - \sigma_{Sf}\sigma_X + 0.5\sigma_{Sf}^2)(T-t)}{\sigma_{Sf}\sqrt{T-t}} \qquad d_2 = d_1 - \sigma_{Sf}\sqrt{T-t}$$

For an arbitrarily chosen strike of \$1,163.00 and $\overline{X}_{\$£} = 1.38$, Monte Carlo solutions were implemented and the results are given in figure 5 (b).

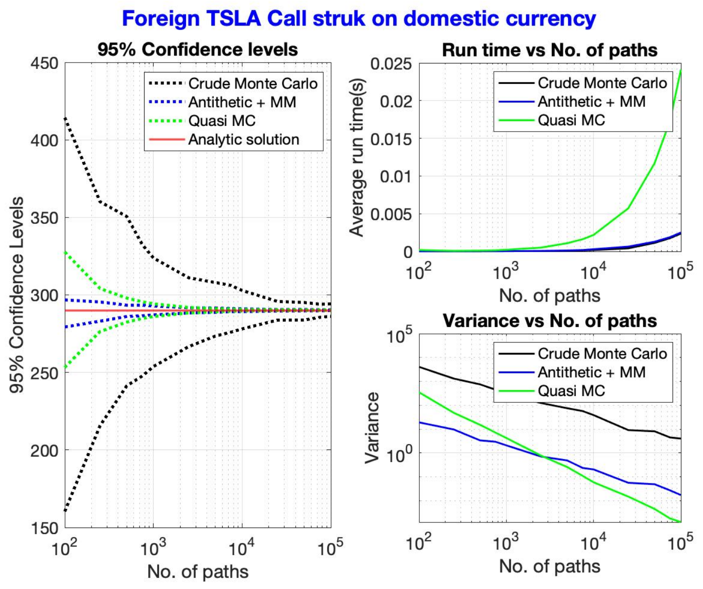
3.5 Results

Both plots show similar behaviour to the barrier option valuation through Brownian bridges. All three samples converge towards the analytic solution, with the quasi Monte Carlo method converging the fastest. However, the moment matching + antithetic variables sample starts at the lowest variance, and is only overtaken by the quasi MC sample at high N, rendering it the most efficient when it comes to computational cost.

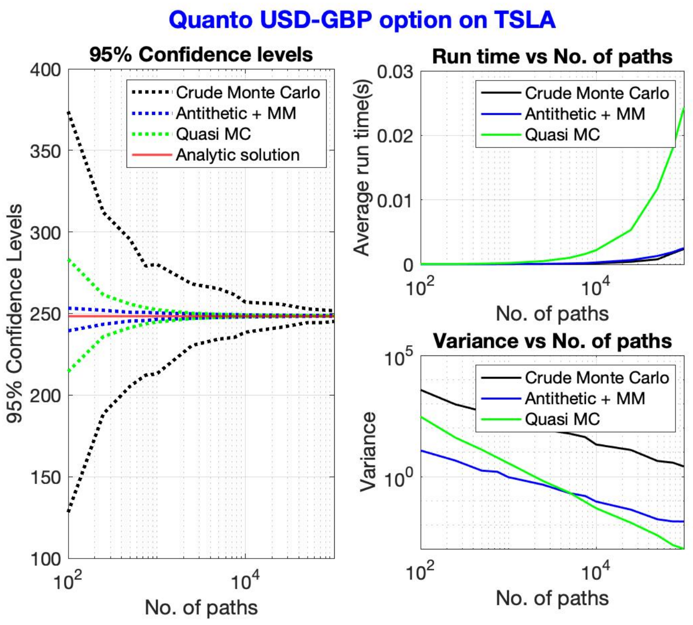
The discussion of foreign exchange options in 3.1. sets up a framework to price more advanced currency options. Note that equations 2 and 4 provide a model to simulate the time series for the exchange rate and the domestic price of a foreign stock respectively. This allows pricing options which may depend on the rate's path, or the stock path in the domestic currency.

4 Conclusion

In this project, we have valued more advanced options, namely path-dependant and currency options through Monte Carlo methods and used an array of variance reduction techniques. For options where only the terminal stock price is required, a high level of variance reduction was observed through all methods. The combination of antithetic variables and moment matching reduced the variance significantly, only surpassed by quasi MC methods at high N, point at which the computational cost of the latter method is far superior. Whenever the entire stock path is required, for example for Asian options, miss-pricing errors due to observing the stock path discretely hinder the power of variance reduction methods.



(a) European call option on foreign stock, struck on domestic currency.



(b) European quantity adjusting call option for *Tesla Inc.* with a fixed rate $\overline{X}_{\$£} = 1.38$.

Figure 5: Analysis of currency option valuation by Monte Carlo across different variance reduction techniques.

Word count: 1972.

5 References

[1] Wilmott, P., et al (1995). *The mathematics of financial derivatives.* 1st ed. Cambridge: Cambridge University Press, pp.4-7.

[2] Bouzoubaa, Mohamed., and Osseiran, Adel (2010). *Exotic Options and Hybrids : A Guide to Structuring, Pricing and Trading.* Chichester, West Sussex, U.K.: Wiley. Print. Wiley Finance Ser.

[3] McKinnon, Ronald I. *The Unloved Dollar Standard: From Bretton Woods to the Rise of China.* New York: Oxford UP, 2012. Web.

[4] Derivativepricing.com. n.d. Resolution : The authority on derivative pricing. [online] Available at: http://www.derivativepricing.com/blogpage.asp?id=23 [Accessed 8 January 2021].

[5] U.S. Energy Information Administration (EIA). n.d. *Europe Brent Spot Price FOB (Dollars per Barrel).* [online] Available at: https://www.eia.gov/dnav/pet/hist/RBRTED.htm [Accessed 15 January 2021].

[6] IMF Data. 2021. *Interest rates and selected indicators: Financial, Interest rates, Monetary Policy-related interest rate, Percent per annum.* [online] Available at: https://data.imf.org/regular.aspx?key=61545867 [Accessed 12 January 2021].

[7] Schurman, G., 2012. *Pricing A Down-And-Out Call Option.* [online] Applied Business Economics: UC Santa Barbara. Available at: http://www.appliedbusinesseconomics.com/files/gvsbar03.pdf [Accessed 1 February 2021].

[8] Krekel, M., 2003. *The Pricing of Asian Options on Average Spot with Average Strike.* SSRN Electronic Journal,.

[9] Lyuu, Y., 2015. *Brownian Bridge Approach to Pricing Barrier Options.* [online] National Taiwan University. Available at: https://www.csie.ntu.edu.tw/~lyuu/finance1/2015/20150520.pdf [Accessed 11 December 2020].

[10] Zervos, M., 2020, *Chapter 6: Foreign Exchange (FX) markets*, lecture notes, MA415: The Mathematics of the Black and Scholes Theory, London School of Economics.

[11] Chen, J., 2019. *Quantity-Adjusting Option (Quanto Option).* [online] Investopedia. Available at: https://www.investopedia.com/terms/q/quantooption.asp [Accessed 5 January 2021].

6 Appendices

6.1 Black Scholes option pricing

The theory of option pricing revolves around the celebrated Black-Scholes model, first postulated by Fisher Black, ..., 1973. Under this model, the underlying asset is modelled by a geometric Brownian motion (GBM), parametrised by two parameters: the volatility σ and the drift μ. In particular, if we denote the underling's price at time t by S_t, the following stochastic differential equation holds:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

where W_t is a standard Brownian motion. This equation can be solved through Itô's lemma to find that the underlying's price process follows a log-normal distribution:

$$S_t = S_0 \exp\left\{ \left(\mu - \frac{\sigma^2}{2} \right) t + \sigma W_t \right\}.$$

For a *risk premium* $\theta = \frac{\mu - r}{\sigma}$, one defines the exponential martingale L_t by:

$$L_t = \exp\left\{ -\frac{\theta^2}{2} t - \theta W_t \right\}$$

such that $\mathbb{Q}(A) = \mathbb{E}[L_T \mathbb{1}_A]$ constitutes a probability measure known as the **risk neutral** measure. Furthermore, in the view of Girsanov's theorem, the process $W_t^\theta = \theta t + W_t$ is a Brownian motion with respect to the measure $\mathbb{Q}$. In the "risk neutral world", it may be shown that discounted stock price $e^{-rt} S_t$ is a martingale with respect to the filtration. Thus, under the absence of arbitrage opportunities, for a European contingent claim that yields the payoff $H_T \geq 0$ at it's maturity $T > 0$, there exists a replicating portfolio strategy such that the no-arbitrage price of this claim is given by:

$$P_t^{H_T} = \mathbb{E}^{\mathbb{Q}}[e^{-r(T-t)} H_T \mid \mathcal{F}_t].$$

In other words, the no-arbitrage price process of a European contingent claim that yields the payoff $F(S_T)$ at its maturity time $T > 0$ is given by the expectation of the payoff's discounted price with respect to the risk neutral measure. For a European call option with payoff $F(S_T) = \max\{S_T - K, 0\}$, it may be shown that the no-arbitrage price of the claim is:

$$C_t(S_t) = \mathbb{E}^{\mathbb{Q}}[\max\{S_t - K, 0\} \mid \mathcal{F}_t] = S_t \Phi(d_1) - e^{-r(T-t)} K \Phi(d_2)$$

where:

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

6.2 Monte Carlo methods & Variance Reduction

A Monte Carlo (MC) method is a non-deterministic algorithm used to estimate complex mathematical problems numerically with high precision. The underlying mechanism is based on random sampling: At each iteration, one (or more) random numbers are generated to solve the problem fully. The average over all such results then provides an estimator for the true solution.

These methods are used broadly in the physical sciences, engineering and many other technical disciplines. One of their main applications is in the financial industry, where one can estimate the temporal evolution of a certain stock or financial contract. Under the risk neutral measure, the price dynamics for an asset S_t at time t is given by:

$$S_{t+\Delta t} = S_0 \exp\left((r - \frac{1}{2}\sigma^2)\Delta t + \sigma\epsilon\sqrt{\Delta t}\right) \tag{5}$$

where $\epsilon \sim \mathcal{N}(0,1)$. This result can be used to price different financial contracts such as those exposed in sections 2 & 3.

Intuitively, increasing the number of simulations should increase the precision of our Monte Carlo estimator. In fact, if we simulate N realisations of a random variable X, the (biased) estimator for the variance is given by:

$$S^2 = \frac{1}{N}\sum_{n=1}^{N}(X_i - \overline{X})^2$$

where $\overline{X}$ is the sample mean. Therefore, we can see that the variance is inversely proportional to the sample size $S^2 \sim \mathcal{O}(N^{-1})$. However, there are a few techniques that can be employed to reduce the variance of a Monte Carlo estimation, known as **variance reduction techniques** (VRT). For the different techniques employed throughout the project: Antithetic variables, moment matching and quasi Monte Carlo methods, a brief summary of each technique is briefly provided below:

- **Antithetic variates:** is a method that attempts to exploit the negative correlation between random variables to reduce the variance. Given random variables (X, Y) which are **antithetic pairs** (i.e. with the same distribution), the antithetic variates estimator of $\mathbb{E}[X] = \mathbb{E}[Y]$ is given by the average of the two estimators:

$$\frac{\overline{X}_n + \overline{Y}_n}{2}$$

  where $\overline{X}_n = \sum_{n=1}^{N} X_i$ and $\overline{Y}_n = \sum_{n=1}^{N} Y_i$.

  For a standard normal, $X \sim \mathcal{N}(\mu, \sigma) \iff -X \sim \mathcal{N}(\mu, \sigma)$. Therefore, $(X, -X)$ are an antithetic pair such that, for each random draw for the shocks ϵ, it's negative will also be considered. This correction comes at no additional computational cost, but can significantly reduce the variance if the random variables are strongly negatively correlated.

- **Moment matching:** Is a technique which attempts to match the moments of random sample with those of the desired distribution. For a standard normal, antithetic variates ensure that the first moment (the mean) and the skewness match. The second moment (the variance) can be matched through a simple correction to the number generating process: by dividing each random draw by the sample's standard deviation, we can ensure that the shocks ϵ have the desired variance.

- **Quasi Monte Carlo:** Most random number generators are based on congruential random number generators, which prioritise drawing numbers close to those which was previously drawn. This causes clustering (and hence bias) in our results. Instead, non-random number generators, such as those provided by low discrepancy sequences, cover the desired domain of interest quickly and evenly. Using these sequences as a number generating mechanism can yield a $\mathcal{O}(N^{-2})$ variance reduction, but usually comes at a high computational cost.

7 Source Code

main.m file:

- Set up directories to data file, and each of the option types.
- For each option type, calculate the analytic solution and call `analyse_OptionType` to run different MC solutions.
- Makes use of the `HistVolatility` function to estimate the historical volatility of a stock / FX rate, and `QueryInterestRates` to consult the IMF data and recover the most recent rate at the specified countries.

```
%% FM443 Individual Project.
%   Monte Carlo pricing of path dependent and currency options.
%   Candidate No: 15200
%   London School of Economics.
%   MT 2020-21.
%% Set up directories to folders:
Dir             = cd;
DataDir         = strcat(Dir, '/data');
BarrierDir      = strcat(Dir, '/barrier');
AsianDir        = strcat(Dir, '/asian');
DomStrikeDir    = strcat(Dir, '/domestic_strike');
QuantoDir       = strcat(Dir, '/quanto');
%% Barrier Option:
cd(DataDir) % Change working directory to data folder
% Option Parameters:
[vol spot] = OilVolatility("oilprices");
B          = 52;     % Barrier level.
K          = 58;     % Strike.
T          = 0.25;   % Maturity = 3 month.
r          = QueryInterestRates("Interest_Rates", "United States", "United Kingdom");

% Analytic Solution:
Settle   = '01-Jan-2021';
Maturity = '01-Apr-2021';    % Aritrary 3 month period.
Compounding = -1;            % For continuous compounding.
Basis = 1;                   % Day count basis.

% Define a RateSpec structure.
RateSpec = intenvset('ValuationDate', Settle, 'StartDates', Settle, 'EndDates', ...
Maturity, 'Rates', r, 'Compounding', Compounding, 'Basis', Basis);

% Define a StockSpec structure.
StockSpec = stockspec(vol, spot);

% Down and out call:
V = barrierbybls(RateSpec, StockSpec, 'call', K, Settle, Maturity,  'DO', B)

% Simulation analysis:
cd(BarrierDir)  % Change working directory to barrier folder
analyse_barrier(vol, spot, B, K, T, r, V)
cd(Dir)         % Go back to main directory.

%% Geometric Asian Option:

% Analytic Solution:
```

```
sigG        = vol/sqrt(3);
b           = 0.5*(r-0.5*sigG^2);
d1          = (log(spot/K)+(b+0.5*sigG^2)*T)/(sigG*sqrt(T));
d2          = d1-sigG*sqrt(T);

disp('Analytic␣Solution:')
A = spot*exp((b-r)*T)*normcdf(d1)-K*exp(-r*T)*normcdf(d2)

% Simulation analysis:
cd(AsianDir)
analyse_asian(spot, vol, r, T, K, Q, A)
cd(Dir)

%% Quanto Option:

% Market parameters:
cd(DataDir)
% Domestic and foreign interest rates:
[rd, rf]  = QueryInterestRates("Interest_Rates", "United Kingdom", "United States")
% GBP-USD exchange rate volatility and spot price:
[sigX X0] = HistVolatility("GBPUSD",3);

% Foreign stock parameters:
[sig_Sf Sf_0] = HistVolatility("TSLA",4);

% Option parameters:
Xbar   = 1.38; % Pre-specified exchange rate.
Kf     = 850;  % Strike price in foreign currency.
T      = 1;    % Maturity.

% Analytic Solution
d1  = 1/(sig_Sf*sqrt(T))*( log(Sf_0/Kf) + (rf-sig_Sf*sigX + 0.5*sig_Sf^2)*T );
d2  = 1/(sig_Sf*sqrt(T))*( log(Sf_0/Kf) + (rf-sig_Sf*sigX - 0.5*sig_Sf^2)*T );
V   = exp((rf-rd-sig_Sf*sigX)*T)*Xbar*Sf_0*normcdf(d1)-exp(-rd*T)*Xbar*Kf*normcdf(d2)

% Simulation analysis:
cd(QuantoDir)
analyse_quanto(X0, Sf_0,  Kf, sigX, sig_Sf, rd, rf, T, Xbar, V)
cd(Dir)
%% Foreign Equity call struck on domestic currency:

Kd  = Kf*X0;
sig = sqrt(sigX^2+sig_Sf^2);

% Analytic Solution
d1  = 1/(sig*sqrt(T))*( log(Sf_0*X0/Kd) + (rd+0.5*sig^2)*T );
d2  = 1/(sig*sqrt(T))*( log(Sf_0*X0/Kd) + (rd-0.5*sig^2)*T );
V   = X0*Sf_0*normcdf(d1)-exp(-rd*T)*Kd*normcdf(d2)

% Simulation analysis:
cd(DomStrikeDir)
analyse_domstrike(X0, Sf_0,  Kd, sig, rd, rf, T, V);
cd(Dir)
```

OilVolatility.m, HistVolatility.m and QueryInterestRates.m:

```
% OilVolatility.m calculates the historical volatility of oil prices.

% Inputs:
% - filename: File name of data file (from EIA).

% Outputs:
% - vol:  Calculated historical volatility (annualised).
% - spot: Underlying spot price (most recent quoted price in data).
% - Price time series + Log returns plots.

function [vol spot] = OilVolatility(filename)
```

```
    warning('off')
    file = strcat(filename,".xls");
    T = readtable(file, 'Sheet', 'Data␣1');
    head(T);

    disp("Succesfully loaded oil historical data.")
    disp("Calculating historical volatilities...")

    Dates         = T.Date;
    Prices        = T.SpotPrices;

    % Search for and delete NaNs
    [row]         = find(isnan(Prices));
    Dates(row)    = [];
    Prices(row)   = [];

    % Historical volatility calculations
    LogRet        = diff(log(Prices));
    DailyVol      = std(LogRet);
    fprintf('Daily␣volatility:␣%.2f␣percent.\n', DailyVol*100);

    AnnualVol     = DailyVol*sqrt(252);
    fprintf('Anualised␣volatility:␣%.2f␣percent.\n', AnnualVol*100);

    vol           = AnnualVol;
    spot          = Prices(end);

    figure(1)
    subplot(1,2,1)
    plot(Dates, Prices, 'k-', 'LineWidth', 1.5)
    sgtitle("Daily spot prices for Brent Crude (USD per Barrel)",...
        'FontSize', 20, 'Color', 'b', 'FontWeight', 'bold');
    ylabel('Spot␣price'); grid('on'); xlabel('Date');
    set(gcf,'color','w'); set(gca, 'FontSize', 14);

    subplot(1,2,2)
    plot(Dates(2:end), 100*LogRet, 'k-', 'LineWidth', 1.5)
    xlabel('Date'); ylabel('LogRet␣(%)'); grid('on');
    set(gcf,'color','w'); set(gca, 'FontSize', 14);

    disp(" ")

    warning('on')

end

% HistVolatility.m calculates the historical volatility of an underlying.

% Inputs:
% - filename: File name of data file (from Yahoo finance).
% - fig_no:   Figure Number.

% Outputs:
% - vol:  Calculated historical volatility (annulaised).
% - spot: Underlying spot price (most recent quoted price in data).
% - Price time series + Log returns plots.

function [vol spot] = HistVolatility(filename, fig_no)
    warning('off')
    file = strcat(filename,".csv");
    T = readtable(file);
    head(T);

    disp("Succesfully loaded exchange rate data.")
    disp("Calculating historical volatilities...")
    Dates          = T.Date;
    Prices         = T.AdjClose;

    % Search for and delete NaNs
    [row]          = find(isnan(Prices));
    Dates(row)     = [];
    Prices(row)    = [];

     % Historical volatility calculations
    LogRet         = diff(log(Prices));
    DailyVol       = std(LogRet);
    fprintf('Daily␣volatility:␣%.2f␣percent.\n', DailyVol*100);

    AnnualVol      = DailyVol*sqrt(252);
    fprintf('Anualised␣volatility:␣%.2f␣percent.\n', AnnualVol*100);

    vol            = AnnualVol;
    spot           = Prices(end);

    figure(fig_no)
    subplot(2,1,1)
    plot(Dates, Prices, 'k-', 'LineWidth', 1.5)
    sgtitle(filename,...
        'FontSize', 16, 'Color', 'b', 'FontWeight', 'bold');
    ylabel('Underlying␣Value'); grid('on');
    set(gcf,'color','w'); set(gca, 'FontSize', 14);

    subplot(2,1,2)
    plot(Dates(2:end), 100*LogRet, 'k-', 'LineWidth', 1.5)
    xlabel('Date'); ylabel('LogRet␣(%)'); grid('on');
    set(gcf,'color','w'); set(gca, 'FontSize', 14);

    disp(" ")
    warning('on')

end

% QueryInterestRates.m query IMF data to look for most recently quoted
% risk-free rates of two specified countries.

% Inputs:
% - filename: File name of data file (from IMF).
% - domestic: Domestic country name (e.g. United Kingdom).
% - foreign:  Foreign country name (e.g. United States).

% Outputs:
% - rd: Domestic rate.
% - rf: Foreign rate.

function [rd, rf] = QueryInterestRates(filename, domestic, foreign)
    warning('off')
    file = strcat(filename,".xlsx");
    T    = readtable(file);

    % Look for row of rates for domestic and foreign countries:
    domesticRates  = T(T.Country == domestic,:);
    foreignRates   = T(T.Country == foreign,:);

    l = width(domesticRates);
    LastDomesticRate = 0;
    LastForeignRate  = 0;

    % Iterate over row of rates, store the last quoted rate (Unless it is
    % not quoted or NaN).
    for i = 4:l
        if (isnumeric(domesticRates{1,i}) & isnan(domesticRates{1,i}) == 0)
```

```
            LastDomesticRate = domesticRates{1,i};
        end
        if (isnumeric(foreignRates{1,i}) & isnan(foreignRates{1,i}) == 0)
            LastForeignRate = foreignRates{1,i};
        end
    end
    rd = LastDomesticRate/100;
    rf = LastForeignRate/100;
    warning('on')
end
```

Halton.m and Sobol.m sequence generators for quasi MC methods:

```
% Sobol.m generates low-discrepancy Sobol sequence
% Inputs:
    % N   = Number of simulations per time instant
    % Q = Number of steps (dimensions) between current and maturity
% Output:
    % N*Q matrix of N(0,1) numbers from Sobol sequence.

function z_RandMat=Sobol(N,Q)
    % Construct quasi-random number stream
    q = qrandstream('sobol',Q,'Skip',1e3,'Leap',1e2);

    % Generate quasi-random points from stream
    RandMat = qrand(q,N);

    % Generate a N*Q matrix which values are normally inveresed of the Sobol sequence points
    z_RandMat = norminv(RandMat,0,1);
end

% Halton.m generates low-discrepancy Halton sequence
% Inputs:
    % N   = Number of simulations per time instant
    % Q   = Number of steps (dimensions) between current time and maturity
% Output:
    % N*Q matrix of N(0,1) numbers from Halton sequence.

function RandNums = GenerateHalton(N, Q)
   % Generate set of Halton numbers.
   HaltonSet = haltonset(2,'Skip',1e3,'Leap',1e2);
   % Extract the first N/2*Q elements and store in two vectors.
   U0         = net(HaltonSet,N/2*Q);
   U1         = U0(:,1); U2 = U0(:,2);

    % Box Muller Method:  takes any uniformly distributed variables and turns
    %    them into normally distributed ones.
    y1 = cos(2*pi.*U2).*sqrt(-2*log(U1));
    y2 = sin(2*pi.*U1).*sqrt(-2*log(U2));
    rng = [y1; y2];

    % Return N*Q matrix.
    RandNums = reshape(rng,N,Q);

end
```

DownAndOutCall and analyse_barrier

- DownAndOutCall prices a down-and-out barrier call option through the Brownian bridge method.
- analyse_barrier computes the Monte Carlo solutions through the different variance reduction techniques.

```
% DownAndOutCall2.m Price a down-and-out call option via Brownian bridge MC
% methods:

% Inputs:
%  - spot: Underlying spot price.
%  - vol:  Underlying volatility.
%  - r:    Risk-free rate
%  - K:    Option strike price.
%  - B:    Barrier level.
%  - T:    Option maturity.
%  - N:    Number of simulations (paths).
%  - phi:  N-vector of N(0,1) random numbers.

% Outputs:
%  Option price.

function price = DownAndOutCall2(N, spot, vol, K, B, T, r, phi)
    C = 0;
    for n = 1:N
        ST = spot * exp((r-0.5*vol^2)*T + sqrt(T)*vol*phi(n,1));

        % Brownian bridge probability p:
        if B < ST
            p = 1 - exp(-2*log(B/spot)*log(B/ST)/(vol^2*T));
        else
            p = 0;
        end
        C  = C + p*max(ST-K,0);
    end
    price = C*exp(-r*T)/N;
end

% analyse_barrier.m computes the 95% CL intervals for the pricing of a European
% down-and-out barrier call option through three Monte Carlo samples:
%  1) Crude Monte Carlo (No VRTs)
%  2) Antithetic Variates + Moment Matching Monte Carlo
%  3) Quasi Monte Carlo.

% Inputs:
%  - spot: Underlying spot price.
%  - vol:  Underlying volatility.
%  - B:    Barrier Level
%  - r:    Risk-free rate
%  - T:    Option Maturity.
%  - K:    Option strike price.
%  - V:    Analytic solution of the option.

% Outputs:
%  Option prices for the 3 different samples.
%  95% CL plots + Variance reduction plots + Elapsed time plots.

function option_prices = analyse_barrier(vol, spot, B, K, T, r, V)
    M    = 100;
    N    = [100, 250, 500, 750, 1000, 2500, 5000, 7500 10000 25000 50000 75000 100000];
    lenN = length(N);

    % ------------------------------------------------------------------------- %
                                  % Crude Monte Carlo:
    % ------------------------------------------------------------------------- %
```

```
values_crude = NaN(M,lenN);
times_crude  = NaN(M,lenN);

for n = 1:lenN
    No_paths = N(n);

    for m = 1:M
        phi_crude          = randn(No_paths,1);
        tic;
        values_crude(m,n)    = DownAndOutCall2(No_paths, spot, vol, K, B, T, r, phi_crude);
        times_crude(m,n)     = toc;
    end
end

% Calculate variances, run times and 95% CL:
mean_values_crude     = mean(values_crude);
variances_crude       = var(values_crude);
run_times_crude       = mean(times_crude);
upperCL_crude    = mean_values_crude + 2.*sqrt(variances_crude);
lowerCL_crude    = mean_values_crude - 2.*sqrt(variances_crude);

% ------------------------------------------------------------------------ %
%              Antithetic variables + Moment Matching:
% ------------------------------------------------------------------------ %
values_anti = NaN(M,lenN);
times_anti  = NaN(M,lenN);

for n = 1:lenN
    No_paths = N(n);

    for m = 1:M
        phi_anti    = randn(No_paths/2,1);
        phi_anti    = [phi_anti; -phi_anti];
        phi_anti    = 1/std(phi_anti).*phi_anti;

        tic;
        values_anti(m,n)    = DownAndOutCall2(No_paths, spot, vol, K, B, T, r, phi_anti);
        times_anti(m,n)     = toc;
    end
end
% Calculate variances, run times and 95% CL:
mean_values_anti     = mean(values_anti);
variances_anti       = var(values_anti);
run_times_anti       = mean(times_anti);
upperCL_anti         = mean_values_anti + 2.*sqrt(variances_anti);
lowerCL_anti         = mean_values_anti - 2.*sqrt(variances_anti);

% ------------------------------------------------------------------------ %
%                      Quasi Monte Carlo:
% ------------------------------------------------------------------------ %
values_quasi = NaN(M,lenN);
times_quasi  = NaN(lenN,1);

for n = 1:lenN
    No_paths = N(n);
    tic;
    phi_quasi     = Halton(No_paths,M);

    for m = 1:M
        phi = phi_quasi(:,m);
        values_quasi(m,n)    = DownAndOutCall2(No_paths, spot, vol, K, B, T, r, phi);
    end
    times_quasi(n,1) = toc/M;
end

mean_values_quasi     = mean(values_quasi);
```

```
    variances_quasi       = var(values_quasi);
    run_times_quasi       = times_quasi;
    upperCL_quasi         = mean_values_quasi + 2.*sqrt(variances_quasi);
    lowerCL_quasi         = mean_values_quasi - 2.*sqrt(variances_quasi);

    % Plot:
    figure(3)
    sgtitle('European␣down-and-out␣barrier␣call␣option', 'FontSize', ...
            18, 'Color', 'b', 'FontWeight', 'bold');
    subplot(2,2,[1 3])
    semilogx(N,upperCL_crude, ':k', 'LineWidth', 2.5)
    hold on
    semilogx(N,upperCL_anti, ':b', 'LineWidth', 2.5)
    hold on
    semilogx(N,upperCL_quasi, ':g', 'LineWidth', 2.5)
    hold on
    yline(V, 'r', 'LineWidth', 1.75)
    hold on
    semilogx(N,lowerCL_crude, ':k', 'LineWidth', 2.5)
    hold on
    semilogx(N,lowerCL_anti, ':b', 'LineWidth', 2.5)
    hold on
    semilogx(N,lowerCL_quasi, ':g', 'LineWidth', 2.5)

    xlabel('No.␣of␣paths'); ylabel('95%␣Confidence␣Levels'); grid('on');
    set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC', 'Analytic␣solution')
    title('95%␣Confidence␣levels')

    subplot(2,2,2)
    semilogx(N, run_times_crude, 'k-', 'LineWidth', 1.5)
    hold on
    semilogx(N, run_times_anti, 'b-', 'LineWidth', 1.5)
    hold on
    semilogx(N, run_times_quasi, 'g-', 'LineWidth', 1.5)
    grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
    ylabel('Average␣run␣time(s)'); xlabel('No.␣of␣paths');
    title('Run␣time␣vs␣No.␣of␣paths')

    subplot(2,2,4)
    loglog(N, variances_crude, 'k-', 'LineWidth', 1.5)
    hold on
    loglog(N, variances_anti, 'b-', 'LineWidth', 1.5)
    hold on
    loglog(N, variances_quasi, 'g-', 'LineWidth', 1.5)
    grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
    xlabel('No.␣of␣paths'); ylabel('Variance');
    title('Variance␣vs␣No.␣of␣paths')

    option_prices = [mean_values_crude(end), mean_values_anti(end),...
                     mean_values_quasi(end)];

end
```

`geom_asian_pricer` and `analyse_asian`

- `geom_asian_pricer` prices a geometrically averaged Asian call option.
- `analyse_asian` computes the Monte Carlo solutions through the different variance reduction techniques.

```
% geom_asian_pricer.m Price a geometrically averaged Asian option via MC methods:

% Inputs:
%  - spot: Underlying spot price.
%  - vol:  Underlying volatility.
%  - r:    Risk-free rate
%  - K:    Option strike price.
%  - T:    Option maturity.
%  - N:    Number of simulations (paths).
%  - Q:    Number of times stock path is realised.
%  - phi:  N*Q matrix of N(0,1) random numbers.

% Outputs:
%  Option price.

function price = geom_asian_pricer(spot, vol, r, T, K, N, Q, phi)
    % N is the no. of simulations, Q the discretisation frequency.
    dt      = T/Q;       % Time increment.
    Value   = NaN(N,1);

    for n = 1:N
        St      = NaN(Q,1);  % Stock path.
        St(1,1) = spot;

        for q  = 2:Q
            St(q,1) = St(q-1,1) * exp((r-0.5*vol^2)*dt + sqrt(dt)*vol*phi(n,q));
        end

        geom_av = mean(log(St));
        Value(n,1) = max(exp(geom_av)-K,0);
    end
    price = exp(-r*T)*mean(Value);
end

% analyse_asian.m computes the 95% CL intervals for the pricing of a geometrically
% average Asian call option through three Monte Carlo samples:
%  1) Crude Monte Carlo (No VRTs)
%  2) Antithetic Variates + Moment Matching Monte Carlo
%  3) Quasi Monte Carlo.

% Inputs:
%  - spot: Underlying spot price.
%  - vol:  Underlying volatility.
%  - r:    Risk-free rate
%  - K:    Option strike price.
%  - Q:    Number of times stock path is realised.
%  - T:    Option Maturity.
%  - V:    Analytic solution of the option.

% Outputs:
%  Option prices for the 3 different samples.
%  95% CL plots + Variance reduction plots + Elapsed time plots.

function option_prices = analyse_asian(spot, vol, r, T, K, Q, V)
    M    = 100;
    N    = [10, 26, 50, 76, 100, 250, 500, 750, 1000, 2500, 5000, 7500 10000];
    lenN = length(N);

    h1 = waitbar(0,'Calculating␣Crude␣Monte␣Carlo.␣Please␣wait...');
```

```
% ------------------------------------------------------------------------ %
                            % Crude Monte Carlo:
% ------------------------------------------------------------------------ %
values_crude = NaN(M,lenN);
times_crude  = NaN(M,lenN);

for n = 1:lenN
    No_paths = N(n);
    for m = 1:M
        phi_crude           = randn(No_paths,Q);
        tic;
        values_crude(m,n)   = geom_asian_pricer(spot, vol, r, T, K, No_paths, Q, phi_crude)
        times_crude(m,n)    = toc;
    end
        waitbar(n/(3*lenN),h1)
end

% Calculate variances, run times and 95% CL:
mean_values_crude  = mean(values_crude);
variances_crude    = var(values_crude);
run_times_crude    = mean(times_crude);
upperCL_crude      = mean_values_crude + 2.*sqrt(variances_crude);
lowerCL_crude      = mean_values_crude - 2.*sqrt(variances_crude);

h2 = waitbar(0,'Applying␣Anithetic␣vars␣+␣MM.␣Please␣wait...');

% ------------------------------------------------------------------------ %
%               Antithetic variables + Moment Matching:
% ------------------------------------------------------------------------ %
values_anti = NaN(M,lenN);
times_anti  = NaN(M,lenN);

for n = 1:lenN
    No_paths = N(n);

    for m = 1:M
        phi_anti   = randn(No_paths/2,Q);
        phi_anti   = [phi_anti; -phi_anti];
        phi_anti   = 1./std(phi_anti).*phi_anti;

        tic;
        values_anti(m,n)   = geom_asian_pricer(spot, vol, r, T, K, No_paths, Q, phi_anti);
        times_anti(m,n)    = toc;
    end
    waitbar((lenN+n)/(lenN*3),h2)
end

% Calculate variances, run times and 95% CL:
mean_values_anti    = mean(values_anti);
variances_anti      = var(values_anti);
run_times_anti      = mean(times_anti);
upperCL_anti        = mean_values_anti + 2.*sqrt(variances_anti);
lowerCL_anti        = mean_values_anti - 2.*sqrt(variances_anti);

h3 = waitbar(0,'Applying␣Quasi␣MC␣methods.␣Please␣wait...');

% ------------------------------------------------------------------------ %
%                        Quasi Monte Carlo:
% ------------------------------------------------------------------------ %
values_quasi = NaN(M,lenN);
times_quasi  = NaN(lenN,1);

for n = 1:lenN
    tic;
    No_paths  = N(n);
    phi_large = Sobol(No_paths*M,Q);

    for m = 1:M
        % Create submatrix from larger matrix
        start_  = No_paths*(m-1)+1;
        end_    = No_paths*m;
        phi_sub = phi_large(start_:end_,:);

        values_quasi(m,n)   = geom_asian_pricer(spot, vol, r, T, K, No_paths, Q, phi_sub);
    end
    waitbar((2*lenN+n)/(lenN*3),h3)
    times_quasi(n,1) = toc/M;
end
mean_values_quasi    = mean(values_quasi);
variances_quasi      = var(values_quasi);
run_times_quasi      = times_quasi;
upperCL_quasi        = mean_values_quasi + 2.*sqrt(variances_quasi);
lowerCL_quasi        = mean_values_quasi - 2.*sqrt(variances_quasi);

close([h1 h2 h3]);

% ------------------------------------------------------------------------ %
%                                Plot:
% ------------------------------------------------------------------------ %
sgtitle('Geometric␣Asian␣option', 'FontSize', ...
        18, 'Color', 'b', 'FontWeight', 'bold');
subplot(2,2,[1 3])
semilogx(N,upperCL_crude, ':k', 'LineWidth', 2.5)
hold on
semilogx(N,upperCL_anti, ':b', 'LineWidth', 2.5)
hold on
semilogx(N,upperCL_quasi, ':g', 'LineWidth', 2.5)
hold on
yline(V, 'r', 'LineWidth', 1.75)
hold on
semilogx(N,lowerCL_crude, ':k', 'LineWidth', 2.5)
hold on
semilogx(N,lowerCL_anti, ':b', 'LineWidth', 2.5)
hold on
semilogx(N,lowerCL_quasi, ':g', 'LineWidth', 2.5)

xlabel('No.␣of␣paths'); ylabel('95%␣Confidence␣Levels'); grid('on');
set(gcf,'color','w'); set(gca, 'FontSize', 14);
legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC', 'Analytic␣solution')
title('95%␣Confidence␣levels')

subplot(2,2,2)
semilogx(N, run_times_crude, 'k-', 'LineWidth', 1.5)
hold on
semilogx(N, run_times_anti, 'b-', 'LineWidth', 1.5)
hold on
semilogx(N, run_times_quasi, 'g-', 'LineWidth', 1.5)
grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
ylabel('Average␣run␣time(s)'); xlabel('No.␣of␣paths');
title('Run␣time␣vs␣No.␣of␣paths')

subplot(2,2,4)
semilogx(N, variances_crude, 'k-', 'LineWidth', 1.5)
hold on
semilogx(N, variances_anti, 'b-', 'LineWidth', 1.5)
hold on
semilogx(N, variances_quasi, 'g-', 'LineWidth', 1.5)
grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
xlabel('No.␣of␣paths'); ylabel('Variance');
```

```
    title('Variance␣vs␣No.␣of␣paths')

    option_prices = [mean_values_crude(end), mean_values_anti(end),...
                     mean_values_quasi(end)];
end
```

domstrike_pricer and analyse_domstrike

- `domstrike_pricer` prices a European call option on a foreign equity, struck in the domestic currency.
- `analyse_domstrike` computes the Monte Carlo solutions through the different variance reduction techniques.

```
% dom_strike_pricer.m Price a European call option on a foreign equity struck
% on domestic currency, via MC methods:

% Inputs:
%  - X0:   Exchange rate spot value.
%  - Sf_0: Equity spot price (foreign currency).
%  - Kd:   Strike price (domestic currency).
%  - sig:  sqrt((Exchange rate Volatility)^2+(Foreign Equity Volatility)^2);
%  - rd:   Domestic Risk-free rate.
%  - rf:   Foreign Risk-free rate.
%  - T:    Option Maturity.
%  - phi:  N-vector of N(0,1) random numbers.
%  - N:    Number of simulations (paths).

% Outputs:
%  Option price.

function price = quanto_pricer(X0, Sf_0,  Kd, sig, rd, rf, T, phi, N)
    Values = NaN(N,1);
    for n = 1:N
        XT_ST_f  = X0*Sf_0 * exp( (rd-0.5*sig^2)*T+sig*sqrt(T)*phi(n,1));
        Values(n,1) = exp(-rd*T)*max(XT_ST_f-Kd, 0);
    end
    price = mean(Values);
end

% analyse_dom_strike.m computes the 95% CL intervals for the pricing of a European
% call option on a foreign equity struck in the domestic currency through three
% Monte Carlo samples:
%  1) Crude Monte Carlo (No VRTs)
%  2) Antithetic Variates + Moment Matching Monte Carlo
%  3) Quasi Monte Carlo.

% Inputs:
%  - X0:   Exchange rate spot value.
%  - Sf_0: Equity spot price (foreign currency).
%  - sig:  sqrt((Exchange rate Volatility)^2+(Foreign Equity Volatility)^2);
%  - rd:   Domestic Risk-free rate.
%  - rf:   Foreign Risk-free rate.
%  - Kd:   Strike price (domestic currency).
%  - T:    Option Maturity.
%  - V:    Analytic solution of the option.

% Outputs:
%  Option prices for the 3 different samples.
%  95% CL plots + Variance reduction plots + Elapsed time plots.

function option_prices = analyse_domstrike(X0, Sf_0,  Kd, sig, rd, rf, T, V)
    M    = 100;
    N    = [100, 250, 500, 750, 1000, 2500, 5000, 7500 10000 25000 50000 75000 100000];
    lenN = length(N);
```

```
% --------------------------------------------------------------------------- %
                            % Crude Monte Carlo:
% --------------------------------------------------------------------------- %
values_crude = NaN(M,lenN);
times_crude  = NaN(M,lenN);

for n = 1:lenN
    No_paths = N(n);
    for m = 1:M
        phi_crude           = randn(No_paths,1);
        tic;
        values_crude(m,n)   = domstrike_pricer(X0, Sf_0,  Kd, sig, rd, rf, T, phi_crude, No
        times_crude(m,n)    = toc;
    end
end

% Calculate variances, run times and 95% CL:
mean_values_crude   = mean(values_crude);
variances_crude     = var(values_crude);
run_times_crude     = mean(times_crude);
upperCL_crude       = mean_values_crude + 2.*sqrt(variances_crude);
lowerCL_crude       = mean_values_crude - 2.*sqrt(variances_crude);

% --------------------------------------------------------------------------- %
%               Antithetic variables + Moment Matching:
% --------------------------------------------------------------------------- %
values_anti = NaN(M,lenN);
times_anti  = NaN(M,lenN);

for n = 1:lenN
    No_paths = N(n);

    for m = 1:M
        phi_anti    = randn(No_paths/2,1);
        phi_anti    = [phi_anti; -phi_anti];
        phi_anti    = 1/std(phi_anti).*phi_anti;

        tic;
        values_anti(m,n)    = domstrike_pricer(X0, Sf_0,  Kd, sig, rd, rf, T, phi_anti, No_p
        times_anti(m,n)     = toc;
    end
end
% Calculate variances, run times and 95% CL:
mean_values_anti     = mean(values_anti);
variances_anti       = var(values_anti);
run_times_anti       = mean(times_anti);
upperCL_anti         = mean_values_anti + 2.*sqrt(variances_anti);
lowerCL_anti         = mean_values_anti - 2.*sqrt(variances_anti);

% --------------------------------------------------------------------------- %
%                       Quasi Monte Carlo:
% --------------------------------------------------------------------------- %
values_quasi = NaN(M,lenN);
times_quasi  = NaN(lenN,1);

for n = 1:lenN
    No_paths = N(n);
    tic;
    phi_quasi     = GenerateHalton(No_paths,M);

    for m = 1:M
        phi = phi_quasi(:,m);
        values_quasi(m,n)   = domstrike_pricer(X0, Sf_0,  Kd, sig, rd, rf, T, phi, No_paths)
    end
    times_quasi(n,1) = toc/M;
end
```

```
    mean_values_quasi    = mean(values_quasi);
    variances_quasi      = var(values_quasi);
    run_times_quasi      = times_quasi;
    upperCL_quasi        = mean_values_quasi + 2.*sqrt(variances_quasi);
    lowerCL_quasi        = mean_values_quasi - 2.*sqrt(variances_quasi);

     % ----------------------------------------------------------------------- %
    %                                Plot:
    % ----------------------------------------------------------------------- %
    figure(6)
    sgtitle('Foreign␣TSLA␣Call␣struk␣on␣domestic␣currency', 'FontSize', ...
            18, 'Color', 'b', 'FontWeight', 'bold');
    subplot(2,2,[1 3])
    semilogx(N,upperCL_crude, ':k', 'LineWidth', 2.5)
    hold on
    semilogx(N,upperCL_anti, ':b', 'LineWidth', 2.5)
    hold on
    semilogx(N,upperCL_quasi, ':g', 'LineWidth', 2.5)
    hold on
    yline(V, 'r', 'LineWidth', 1.75)
    hold on
    semilogx(N,lowerCL_crude, ':k', 'LineWidth', 2.5)
    hold on
    semilogx(N,lowerCL_anti, ':b', 'LineWidth', 2.5)
    hold on
    semilogx(N,lowerCL_quasi, ':g', 'LineWidth', 2.5)

    xlabel('No.␣of␣paths'); ylabel('95%␣Confidence␣Levels'); grid('on');
    set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC', 'Analytic␣solution')
    title('95%␣Confidence␣levels')

    subplot(2,2,2)
    semilogx(N, run_times_crude, 'k-', 'LineWidth', 1.5)
    hold on
    semilogx(N, run_times_anti, 'b-', 'LineWidth', 1.5)
    hold on
    semilogx(N, run_times_quasi, 'g-', 'LineWidth', 1.5)
    grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
    ylabel('Average␣run␣time(s)'); xlabel('No.␣of␣paths');
    title('Run␣time␣vs␣No.␣of␣paths')

    subplot(2,2,4)
    loglog(N, variances_crude, 'k-', 'LineWidth', 1.5)
    hold on
    loglog(N, variances_anti, 'b-', 'LineWidth', 1.5)
    hold on
    loglog(N, variances_quasi, 'g-', 'LineWidth', 1.5)
    grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
    xlabel('No.␣of␣paths'); ylabel('Variance');
    title('Variance␣vs␣No.␣of␣paths')

    option_prices = [mean_values_crude(end), mean_values_anti(end),...
                     mean_values_quasi(end)];
end
```

`quanto_pricer` and `analyse_quanto`

- `quanto_pricer` prices a European quantity adjusting call option on a foreign equity.
- `analyse_quanto` computes the Monte Carlo solutions through the different variance reduction techniques.

```
% quanto_pricer.m Price a European quanto call option via MC methods:

% Inputs:
%  - X0:    Exchange rate spot value.
%  - Sf_0:  Equity spot price (foreign currency).
%  - Kf:    Strike price (foreign currency).
%  - sigX:  Exchange rate volatility.
%  - sigSf: Foreign equity volatility.
%  - rd:    Domestic Risk-free rate.
%  - rf:    Foreign Risk-free rate.
%  - T:     Option Maturity.
%  - Xbar:  Guaranteed exchange rate.
%  - phi:   N-vector of N(0,1) random numbers.
%  - N:     Number of simulations (paths).

% Outputs:
%  Option price.

function price = quanto_pricer(X0, Sf_0,  Kf, sigX, sigSf, rd, rf, T, Xbar, phi, N)
    Values = NaN(N,1);
    for n = 1:N
        ST_f  = Sf_0 * exp((rf-sigX*sigSf-0.5*sigSf^2)*T + sigSf*phi(n,1));
        Values(n,1) = Xbar*exp(-rd*T)*max(ST_f-Kf,0);
    end
    price = mean(Values);
end

% analyse_quanto.m computes the 95% CL intervals for the pricing of a European
% quantity adjusting (quanto) call option through three Monte Carlo samples:
%  1) Crude Monte Carlo (No VRTs)
%  2) Antithetic Variates + Moment Matching Monte Carlo
%  3) Quasi Monte Carlo.

% Inputs:
%  - X0:    Exchange rate spot value.
%  - Sf_0:  Equity spot price (foreign currency).
%  - Kf:    Strike price (foreign currency).
%  - sigX:  Exchange rate volatility.
%  - sigSf: Foreign equity volatility.
%  - rd:    Domestic Risk-free rate.
%  - rf:    Foreign Risk-free rate.
%  - T:     Option Maturity.
%  - Xbar:  Guaranteed exchange rate.
%  - V:     Analytic solution of the option.

% Outputs:
%  Option prices for the 3 different samples.
%  95% CL plots + Variance reduction plots + Elapsed time plots.

function option_prices = analyse_quanto(X0, Sf_0,  Kf, sigX, sigSf, rd, rf, T, Xbar, V)
    M    = 100;
    N    = [100, 250, 500, 750, 1000, 2500, 5000, 7500 10000 25000 50000 75000 100000];
    lenN = length(N);

    % ---------------------------------------------------------------------- %
                               % Crude Monte Carlo:
    % ---------------------------------------------------------------------- %
    values_crude = NaN(M,lenN);
    times_crude  = NaN(M,lenN);
```

```
for n = 1:lenN
    No_paths = N(n);
    for m = 1:M
        phi_crude           = randn(No_paths,1);
        tic;
        values_crude(m,n)   = quanto_pricer(X0, Sf_0,  Kf, sigX, sigSf, rd, rf, T, Xbar, phi
        times_crude(m,n)    = toc;
    end
end

% Calculate variances, run times and 95% CL:
mean_values_crude  = mean(values_crude);
variances_crude    = var(values_crude);
run_times_crude    = mean(times_crude);
upperCL_crude      = mean_values_crude + 2.*sqrt(variances_crude);
lowerCL_crude      = mean_values_crude - 2.*sqrt(variances_crude);

% ------------------------------------------------------------------------ %
%               Antithetic variables + Moment Matching:
% ------------------------------------------------------------------------ %
values_anti = NaN(M,lenN);
times_anti  = NaN(M,lenN);

for n = 1:lenN
    No_paths = N(n);

    for m = 1:M
        phi_anti   = randn(No_paths/2,1);
        phi_anti   = [phi_anti; -phi_anti];
        phi_anti   = 1/std(phi_anti).*phi_anti;

        tic;
        values_anti(m,n)   = quanto_pricer(X0, Sf_0,  Kf, sigX, sigSf, rd, rf, T, Xbar, phi_
        times_anti(m,n)    = toc;
    end
end
% Calculate variances, run times and 95% CL:
mean_values_anti    = mean(values_anti);
variances_anti      = var(values_anti);
run_times_anti      = mean(times_anti);
upperCL_anti        = mean_values_anti + 2.*sqrt(variances_anti);
lowerCL_anti        = mean_values_anti - 2.*sqrt(variances_anti);

% ------------------------------------------------------------------------ %
%                       Quasi Monte Carlo:
% ------------------------------------------------------------------------ %
values_quasi = NaN(M,lenN);
times_quasi  = NaN(lenN,1);

for n = 1:lenN
    No_paths = N(n);
    tic;
    phi_quasi    = GenerateHalton(No_paths,M);

    for m = 1:M
        phi = phi_quasi(:,m);
        values_quasi(m,n)   = quanto_pricer(X0, Sf_0,  Kf, sigX, sigSf, rd, rf, T, Xbar, phi
    end
    times_quasi(n,1) = toc/M;
end

mean_values_quasi    = mean(values_quasi);
variances_quasi      = var(values_quasi);
run_times_quasi      = times_quasi;
upperCL_quasi        = mean_values_quasi + 2.*sqrt(variances_quasi);
```

```
    lowerCL_quasi        = mean_values_quasi - 2.*sqrt(variances_quasi);

     % ------------------------------------------------------------------------ %
    %                               Plot:
    % ------------------------------------------------------------------------ %
    figure(5)
    sgtitle('Quanto␣USD-GBP␣option␣on␣TSLA', 'FontSize', ...
            18, 'Color', 'b', 'FontWeight', 'bold');
    subplot(2,2,[1 3])
    semilogx(N,upperCL_crude, ':k', 'LineWidth', 2.5)
    hold on
    semilogx(N,upperCL_anti, ':b', 'LineWidth', 2.5)
    hold on
    semilogx(N,upperCL_quasi, ':g', 'LineWidth', 2.5)
    hold on
    yline(V, 'r', 'LineWidth', 1.75)
    hold on
    semilogx(N,lowerCL_crude, ':k', 'LineWidth', 2.5)
    hold on
    semilogx(N,lowerCL_anti, ':b', 'LineWidth', 2.5)
    hold on
    semilogx(N,lowerCL_quasi, ':g', 'LineWidth', 2.5)

    xlabel('No.␣of␣paths'); ylabel('95%␣Confidence␣Levels'); grid('on');
    set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC', 'Analytic␣solution')
    title('95%␣Confidence␣levels')

    subplot(2,2,2)
    semilogx(N, run_times_crude, 'k-', 'LineWidth', 1.5)
    hold on
    semilogx(N, run_times_anti, 'b-', 'LineWidth', 1.5)
    hold on
    semilogx(N, run_times_quasi, 'g-', 'LineWidth', 1.5)
    grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
    ylabel('Average␣run␣time(s)'); xlabel('No.␣of␣paths');
    title('Run␣time␣vs␣No.␣of␣paths')

    subplot(2,2,4)
    loglog(N, variances_crude, 'k-', 'LineWidth', 1.5)
    hold on
    loglog(N, variances_anti, 'b-', 'LineWidth', 1.5)
    hold on
    loglog(N, variances_quasi, 'g-', 'LineWidth', 1.5)
    grid('on'); set(gcf,'color','w'); set(gca, 'FontSize', 14);
    legend('Crude␣Monte␣Carlo', 'Antithetic␣+␣MM', 'Quasi␣MC')
    xlabel('No.␣of␣paths'); ylabel('Variance');
    title('Variance␣vs␣No.␣of␣paths')

    option_prices = [mean_values_crude(end), mean_values_anti(end),...
                     mean_values_quasi(end)];
end
```