BACHELOR'S DEGREE PROJECT IN MATHEMATICS

Stochastic Runge–Kutta Lawson Schemes for European and Asian Call Options Under the Heston Model

by

Nicolas Kuiper and Martin Westberg

MAA043 — Bachelor's Degree Project in Mathematics

DIVISION OF MATHEMATICS AND PHYSICS
MÄLARDALEN UNIVERSITY
SE-721 23 VÄSTERÅS, SWEDEN

MAA043 — Bachelor's Degree Project in Mathematics

Date of presentation:
31st May 2023

Project name:
Stochastic Runge–Kutta Lawson Schemes for European and Asian Options Call Under the Heston Model

Author(s):
Nicolas Kuiper & Martin Westberg

Version:
4th July 2023

Supervisor(s):
Anatoliy Malyarenko

Opponent:
Tarek Hasna

Examiner:
Linus Carlsson

Comprising:
15 ECTS Credits

Abstract

This thesis investigated Stochastic Runge–Kutta Lawson (SRKL) schemes and their application to the Heston model. Two distinct SRKL discretization methods were used to simulate a single asset's dynamics under the Heston model, notably the Euler–Maruyama and Midpoint schemes. Additionally, standard Monte Carlo and variance reduction techniques were implemented. European and Asian option prices were estimated and compared with a benchmark value regarding accuracy, effectiveness, and computational complexity. Findings showed that the SRKL Euler–Maruyama schemes exhibited promise in enhancing the price for simple and path-dependent options. Consequently, integrating SRKL numerical methods into option valuation provides notable advantages by addressing challenges posed by the Heston model's SDEs. Given the limited scope of this research topic, it is imperative to conduct further studies to understand the use of SRKL schemes within other models.

Keywords: Runge–Kutta Lawson scheme; Heston model; Black–Scholes model; Stochastic Differential Equation; Euler–Maruyama scheme; Midpoint scheme; Monte Carlo; European Options; Asian Options; Option pricing.

Acknowledgements

We express our deepest gratitude to our supervisor, Prof. Anatoliy Malyarenko, whose support and guidance have profoundly impacted our research. His insightful feedback, expertise, patience, and determination have played a pivotal role in shaping our ideas and refining our methodology.

We acknowledge the contribution of the authors whose works we have consulted and cited. Their work has provided us with a solid foundation and served as a guiding light in our research. In particular, we would like to extend our most profound appreciation to Kristian Debrabant, whose remarkable work in [5, 6, 7], and invaluable insights shared through our email correspondence, have enriched our understanding of the topic and added significant depth and nuance to our work.

We want to dedicate this thesis to our beloved families and friends as a sincere token of appreciation. Their constant encouragement and belief in our abilities have been a tremendous source of inspiration throughout our life, and their love and support have given us the strength to overcome every challenge and achieve new heights. Thank you.

Contents

List of Tables

List of Figures

List of Symbols

V	asset variance
S	asset price
S_t	asset price at time t
T	option contract maturity
r	risk-free interest rate
σ	volatility, standard error or volatility of volatility for the Heston model
κ	rate of mean reversion
ρ	correlation
θ	long-term average volatility
τ	time elapsed
$\sigma^2\tau$	efficiency
C_A, C_{EU}	call option price, arithmetic Asian and European, respectively

Acronyms

ATM At-the-money. 15, 38, 43, 53, 54

CIR Cox, Ingersoll, and Ross. 14, 18, 19

DSL Drift Stochastic Lawson. 30, 31, 49, 57, 59

FSL Full Stochastic Lawson. 30, 31, 49, 56, 57, 59

GBM Geometric Brownian Motion. 11, 17, 21, 26

ITM In-the-money. 15, 38, 43, 52–54

ODE Ordinary Differential Equation. 10, 27

OTM Out-the-money. 15, 19, 38, 43, 52–55

PDE Partial Differential Equation. 10, 17

RK Runge–Kutta. 11, 27

SDE Stochastic Differential Equation. 1, 10–12, 18, 19, 27, 29, 32, 33, 59

SRK Stochastic Runge–Kutta. 27

SRKL Stochastic Runge–Kutta Lawson. 1, 10–14, 27–30, 33, 37–45, 50, 57–59, 110

Chapter 1

Introduction

1.1 Motivation

Various option pricing methods have been proposed in previous research, such as Finite Difference methods, Binomial Trees, and Monte Carlo methods, to name a few. Mathematical tools such as Ordinary Differential Equations (ODE) and Partial Differential Equations (PDE) are often used for pricing options and modeling the behavior of financial assets; yet, these tools fail to consider the unpredictable nature of financial markets. On the contrary, Stochastic Differential Equations (SDE) incorporate this uncertainty through stochastic terms [21, pp. 278–279].

Some of the methods mentioned above make use of SDEs by modeling the dynamics of the underlying asset's price, volatility, and other relevant factors, such as market forces or company-specific factors. Therefore, understanding these dynamics becomes crucial since the value of options fluctuates in response to them. While simpler SDEs have closed-form solutions, such as the one found in the Black–Scholes model [1], more intricate models such as the Heston Stochastic Volatility model requires numerical methods to obtain solutions [27, pp. 10–12].

In light of these considerations, the motivation behind this thesis is to delve into the study and implementation of numerical methods for solving SDEs in the context of pricing European and Asian options under the Heston Stochastic Volatility model. Among the various numerical methods proposed for solving SDEs, our emphasis will be on Monte Carlo methods and the Stochastic Runge–Kutta Lawson (SRKL) schemes proposed in [6, p. 382], due to their high order of convergence and numerical consistency. In particular, we will perform Monte Carlo simulations by the Euler–Maruyama SRKL and Midpoint SRKL schemes suggested in [6] to approximate the solutions of the Heston SDEs.

1.2 Literature Review

A general summary of the concepts employed throughout the thesis surrounding options, option pricing and their use in the financial markets can be found in [12, 23, 27]. An introduction to the idea of stochastic processes and Itô Integrals is given in [13]. A supplementary overview of such integrals and stochastic calculus applied to finance is further contributed in [10, 16].

In their paper, authors of [1] introduced the classical Black–Scholes model for pricing European options, involving the SDE known as Geometric Brownian Motion (GBM) to simulate price paths of the underlying asset. In response to the limitations of the Black–Scholes model, the author in [11] proposed a model which considers the addition of stochastic volatility, namely, the Heston Stochastic Volatility model. For thorough details on the Heston model, its applications and limitations, see [8, 27],

Many researchers have investigated numerical methods for solving SDEs in the context of pricing options and other financial assets. A thorough outline of Monte Carlo methods, other numerical methods for solving SDEs, and variance reduction techniques, can be found in [9, 21]. In addition, for insightful overviews of numerical solutions to SDEs, see [15, 19, 22, 28].

Furthermore, numerous studies have specifically applied numerical methods to price options within the Heston model. For example, [20] presented a new numerical method to solve Heston's SDEs based on a higher-order Runge–Kutta (RK) weak approximation scheme, and showed results of quicker computational time. On the other hand, [3] implemented and calibrated the Heston model in MATLAB to real market data, showing difficulties in accurately approximating the Heston model coefficients. The author of [24, 25, 26] focused instead on order conditions for rooted tree analysis as well as Stratonovich and Itô SDEs.

Lastly, the authors of [6] proposed an SRKL scheme that is stable, accurate, and efficient in capturing the stochastic volatility process. In [7], they also investigated highly oscillatory problems in SDEs and proposed a method to improve the accuracy of numerical solutions. In [5] they offer the MATLAB functions employed in their papers mentioned previously. These papers will be at the core of this thesis.

As a whole, these studies exemplify the diverse approaches taken to analyse and implement stochastic models in the realm of financial mathematics.

1.3 Research Question

Option pricing techniques possess distinct advantages and disadvantages that can influence their purpose within specific problem characteristics. Therefore, our research aims to comprehensively analyse selected algorithms, with particular emphasis on the SRKL schemes proposed in [6]. Through these efforts, we aim to enhance the understanding of SRKL numerical methods by studying their strengths, weaknesses, and performance when implemented for solving option pricing problems.

In summary, the question that we intend to address throughout the entire thesis is:

- **Can the valuation of financial options be enhanced by integrating SRKL numerical methods to solve the SDEs of the Heston model?**

To address this question, we formulated the following sub-questions:

- *How do SRKL schemes differ from other methods commonly used for pricing options?*
- *How do the SDEs of the Heston model pose a challenge for option pricing?*
- *Can variance reduction techniques be implemented in SRKL schemes?*
- *Can integrating SRKL numerical methods lead to improvements in pricing specific financial options, such as exotic or path-dependent options?*

1.4 Research Delimitation

This research is focused on studying pricing algorithms for option contracts, specifically European and Asian options under the Heston model. The primary methods being considered for solving the option pricing challenges are the SRKL schemes proposed in [6, 7], with the more traditional Black–Scholes formula and Monte Carlo method as benchmarks for European and Asian options, respectively.

The following considerations define the delimitation of this study:

- **Limited scope:** The research will concentrate on option pricing methods using SRKL schemes and Monte Carlo techniques. We will not focus on other methods.
- **Option contract types:** The thesis will only focus on European and Asian call options to study and compare at least one path-dependent option class. We will not study other option types.
- **Underlying models:** The research will only consider the SDEs associated with the Heston model. We will not contemplate other models.
- **Variance reduction techniques:** The study will explore the potential for optimization of the schemes by implementing variance reduction techniques. In particular, antithetic variate and control variate techniques. We will not consider other optimization approaches.

1.5 Research Outline

Chapter 1 provides an extensive literature review and the general motivation for this thesis. The research question is presented together with the delimitation of our work. Henceforth, the thesis will be structured as follows: Chapter 2 lays down the theoretical frameworks for the thesis. More specifically, the concept of option pricing, the relevant models utilised throughout the study, the principles of Monte Carlo simulations, and the construction and application of SRKL schemes. Chapter 3 portrays the methodology employed in the research. The results obtained throughout the numerical implementation of the models are later presented in Chapter 4. In Chapter 5, the results are thoroughly discussed, and, lastly, concluding remarks are made in Chapter 6.

Chapter 2

Theoretical Background

This section deals with the principles that form the foundation for understanding and valuing financial derivatives known as options. The determination of option prices involves a complex interplay of factors which can be influenced by market dynamics, asset characteristics, and risk considerations, among others. These principles are deeply rooted in various mathematical models, the most prominent being the Black–Scholes model and its extensions, such as the Heston model.

Next, we explore the derivation of the Heston model. This model extends from the Black–Scholes model by introducing a stochastic process for the volatility. In particular, we adapt the Cox, Ingersoll, and Ross (CIR) model, proposed in [2], to capture the dynamics of stochastic volatility. The subsequent section deals with the concept of Monte Carlo simulations and highlights some variance reduction techniques, which in theory, allow for more accurate option pricing results. Lastly, we introduce the SRKL schemes, elaborating on their construction and practical application.

2.1 Principles of Option Pricing

Stocks, futures, commodities and even cryptocurrencies are some possible underlying assets for option contracts. As their name suggests, options contracts are a class of financial derivative where two parties enter an agreement that provides the holder of the option with the right, but not the obligation, to buy (call) or sell (put) a fixed quantity of the underlying for a fixed price, namely strike price K, within a set time frame called maturity, denoted by T.

Option prices are influenced by several factors, such as the current value of the underlying, S_0, or the contract's specifications, like the strike price, and remaining time until expiration. Naturally, market- or company-specific factors such as volatility, interest rates and dividends also need to be considered when determining an option's fair value.

A particular concept of interest when studying options is *moneyness*. In brief, moneyness describes the relationship between K and S_0, which helps determine potential profitability and risk associated with a particular option contract. The moneyness for call and put options can be expressed by [12]:

- In-the-money (ITM) — $S_T > K$ for calls. $S_T < K$ for puts.
- Out-the-money (OTM) — $S_T < K$ for calls. $S_T > K$ for puts.
- At-the-money (ATM) — $S_T = K$ for both.

Thus, these expressions help determine whether an option has intrinsic value and is favourable for exercising based on the relative prices of the underlying asset and the strike price.

Subject to the options class, exercising may be possible only at maturity (e.g. European and Asian), at particular exercise opportunities (e.g. Bermuda), or at any time throughout the life of the option (e.g. American). Furthermore, the value of each option type is determined based on its respective payoff function. As aforementioned, this study will only address the pricing of European and Asian options, which are described in detail later in this chapter.

2.1.1 Risk–Neutral Pricing

A risk–neutral measure, commonly denoted by $\mathbb{Q}$, is a particular choice of probability measure that lies at the core of derivatives pricing, particularly in option pricing by simulation. Under the risk–neutral probability, the return on investment is equivalent to the risk–free interest rate r, as discussed in [1, 12, 14].

In terms of option pricing, a risk–neutral probability measure suggests that the current price of the option is expressed as the expected present value of the payoff, discounted at the risk–free rate. Symbolically,

$$V_0 = e^{-rT}\mathbb{E}_{\mathbb{Q}}[V_T], \tag{2.1}$$

where $\mathbb{E}_{\mathbb{Q}}[\cdot]$ is the expectation function under the probability measure $\mathbb{Q}$; and V_0 and V_T are the value options at time 0 and T, respectively.

Under the $\mathbb{Q}$–measure, it is assumed that investors are indifferent to risk and only consider the expected returns of investments. This assumption greatly simplifies the modelling since the asset price dynamics are more easily described, and pricing of derivatives like options is then possible by Equation (2.1). In contrast, an objective probability measure, often denoted by $\mathbb{P}$, reflects the actual probabilities observed in the market, which introduces complexities in modelling the asset price dynamics.

2.1.2 European Options Pricing

European–type derivative securities are amongst the simplest type of derivatives. As mentioned before, they can only be exercised at maturity, so pricing them tends to be straightforward.

Let S_t be the underlying price at time t and K be the strike price of a European option with maturity T. The payoff functions are defined as

$$h_t^C = \max\left(S_t - K, 0\right) \equiv \left(S_t - K\right)^+, \tag{2.2a}$$

$$h_t^P = \max\left(K - S_t, 0\right) \equiv \left(K - S_t\right)^+, \tag{2.2b}$$

for a European call and put, respectively. The positive sign indicates that the payoff function cannot be negative, but may take zero–values.

Because European options are only exercised at maturity, we are interested in h_T^C and h_T^P. At maturity, the value of the call is given by $V_T^C \equiv h_T^C$ and the value of the put is given by $V_T^P \equiv h_T^P$. We evaluate in Equation (2.1) to obtain the present option values by

$$V_0^C = e^{-rT}\mathbb{E}\left[V_t^C\right], \tag{2.3a}$$

$$V_0^P = e^{-rT}\mathbb{E}\left[V_t^P\right]. \tag{2.3b}$$

2.1.3 Asian Options Pricing

Asian–type derivative securities are similar to their European counterpart in terms of the exercise opportunity being only at maturity. However, Asian options are path dependent, and their payoff function considers the average price of the underlying asset over the option lifespan. Additionally, arithmetic Asian options consider the actual prices at each observation point, while geometric Asian options consider the relative returns or growth rates of the prices.

Let K and T be the strike price and maturity, respectively. The payoff functions for an *arithmetic* Asian call and put are defined as follows:

$$h_t^C = \max\left(A - K, 0\right) \equiv \left(A - K\right)^+, \tag{2.4a}$$

$$h_t^P = \max\left(K - A, 0\right) \equiv \left(K - A\right)^+, \tag{2.4b}$$

where A represents the arithmetic average of the underlying asset prices over the averaging period $[t_i = 0, t_n = T]$, and is given by

$$A = \frac{1}{n}\sum_{i=1}^{n} S_{t_i}.$$

For a *geometric* Asian option, the payoff functions are defined as:

$$h_t^C = \max\left(G - K, 0\right) \equiv \left(G - K\right)^+, \tag{2.5a}$$

$$h_t^P = \max\left(K - G, 0\right) \equiv \left(K - G\right)^+, \tag{2.5b}$$

with G representing the geometric average of the underlying asset prices over the averaging period $[t_i = 0, t_n = T]$, and is given by

$$G = \left(\prod_{i=1}^{n} S_{t_i} \right)^{n^{-1}} .$$

Furthermore, S_{t_i} are the individual prices observations. Once A and G are estimated, the pricing of Asian options is analogous to that of European options, i.e. calculating the payoff at maturity and discounting at the risk–free rate.

In general, Asian options are popular instruments because they tend to be cheaper than their European counterparts. They can also provide investors with exposure to the underlying asset's average performance rather than focusing on a single point in time. Asian options can be useful in markets with high volatility or when investors want to hedge against average price movements rather than specific prices at a particular time [4].

2.2 Extending Black–Scholes: Derivation of the Heston Model

This section transitions from the Black–Scholes model to the Heston model in options pricing. The Heston model introduces stochastic volatility, addressing the limitations of constant volatility assumptions.

2.2.1 The Black–Scholes Model

The Black–Scholes equation, also known as the Black–Scholes PDE, was developed by Fisher Black and Myron Scholes in 1973 [1], with significant later contributions from Robert Merton in the paper [18]. Together, they were able to derive the closed-form formula for pricing European options by utilizing continuous-time and no-arbitrage reasoning. The main assumptions needed for the derivation of the model are:

- There exists a risk-free asset that earns r.
- Both sides of trades are permitted (i.e. going long or short the asset).
- Transaction costs are omitted.
- The underlying asset does not issue dividends.

Further, the Black–Scholes model assumes that the underlying asset follows a GBM:

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t, \tag{2.6}$$

where μ is the interest rate r, σ is the constant stock price volatility, and W_t is a standard Brownian motion or Wiener process [1, 12, 18]. We now present the following definitions and theorems that may be found in any standard textbook in financial mathematics:

Definition 1. A stochastic process W_t defined on the time interval $[0, \infty)$ is called a *Wiener process* or a *Brownian motion* if it satisfies the following requirements.

1. It has independent increments,
2. the increment $(W_{t+s} - W_t) \sim \mathcal{N}(0, s)$,
3. it has continuous sample paths and $W_0 = 0$.

Theorem 1. *The solution to Equation* (2.6) *takes the form*

$$S_t = S_0 \exp((r - \sigma^2/2)t + \sigma W_t),$$

where S_t and S_0 denote asset price at time t and $t = 0$ respectively.

Theorem 2. *The prices for European call- and put options are expressed as*

$$C(S_0, t) = S_0 N(d_1) - Ke^{-r(T-t)} N(d_2),$$
$$P(S_0, t) = Ke^{-r(T-t)} N(-d_2) - S_0 N(-d_1),$$

where

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

The functions $C(S_0, t)$ and $P(S_0, t)$ in Theorem 2 represent the prices of European call and put option contracts, respectively. In particular, S_0 denotes the current underlying asset price, K the strike price, r the risk-free rate, T the time to expiration, and $N(\cdot)$ the cumulative standard normal distribution function.

2.2.2 Cox–Ingersoll–Ross Model

The Cox, Ingersoll, and Ross (CIR) process is an SDE model describing the dynamics of the short-term interest rate under the assumption of mean-reversion. In 1985, John Cox, Jonathan Ingersoll and Stephen Ross introduced the idea, suggesting that interest rates tend to return gradually to the long-term average rates at a reversion rate relying on the current degree of deviation [2].

The CIR model is described by:

$$dr_t = \kappa(\theta - r_t)dt + \sigma\sqrt{r_t}dW_t, \tag{2.7}$$

with the interest rate at time t denoted by r_t, κ is the mean reversion rate, describing the speed at which the interest rate reverts to the long-term value, and $\theta > 0$ is the long-term value or mean of the interest rate. Further, σ is the volatility, and dW_t is the increment of a standard Wiener process [2].

An advantage of the model is its capacity to mimic important sequences such as volatility clustering and interest rate structures found in financial markets data. In practice, it is possible to incorporate stochastic volatility by introducing a volatility process that follows CIR-like dynamics.

2.2.3 The Heston Model

When pricing options under the Black–Scholes model, presuming volatility to be constant turns out to be ineffective because it can over- or under-price them, specifically in cases such as deep-OTM options [12, p. 653]. To improve the model, and thereby also the quality of option pricing, the Heston model was introduced in 1993 [11]

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sqrt{V_t}S_t\mathrm{d}W_t^1, \tag{2.8a}$$
$$\mathrm{d}V_t = \kappa(\theta - V_t)\mathrm{d}t + \sigma\sqrt{V_t}\mathrm{d}W_t^2. \tag{2.8b}$$

The spot price and the volatility of the asset at time t are S_t and V_t, respectively. Moreover, θ is the mean of the variance, σ is the volatility of volatility and κ is the rate of mean reversion. The Wiener processes, W_t^1 and W_t^2 are correlated, and in particular

$$\mathbb{E}\left[W_s^1 W_t^2\right] = \rho \min(s,t), \quad \rho \in [-1,1].$$

It is crucial to highlight that the SDEs of the Heston model combine the Black–Scholes SDE, governing the asset's price, with an adapted CIR model to capture the asset's volatility. This adaptation allows for mean reversion in the variance process, a common characteristic observed in financial time series [11]. Although the adapted CIR model may not capture the full complexity volatility dynamics observed in real markets, the consideration of stochastic volatility is a key component in pricing financial derivatives such as options [8, pp. 14–24].

An alternative representation of the Heston model with uncorrelated Wiener processes follows:

Theorem 3. *The system (2.8) is equivalent to the system*

$$\begin{aligned}
\mathrm{d}S_t &= rS_t\,\mathrm{d}t + \sqrt{V_t}S_t\,\mathrm{d}W_t^1,\\
\mathrm{d}V_t &= \kappa(\theta - V_t)\,\mathrm{d}t + \sigma\sqrt{V_t}\left[\rho\,\mathrm{d}W_t^1 + \sqrt{1-\rho^2}\,\mathrm{d}W_t^2\right],
\end{aligned}$$

where the Wiener processes W_t^1 and W_t^2 are uncorrelated.

Theorem 3 is of major importance, as it allows us to simulate the asset's price and volatility paths by simulating two uncorrelated Wiener processes. We apply Definition 1 to prove this theorem.

Proof of Theorem 3. Let $\tilde{W}_t^1$ and $\tilde{W}_t^2$ be two independent Wiener processes. Define the stochastic processes W_t^1 and W_t^2 by

$$W_t^1 = \tilde{W}_t^1, \qquad W_t^2 = \rho\tilde{W}_t^1 + \sqrt{1-\rho^2}\tilde{W}_t^2. \tag{2.9}$$

By equivalence, the process W_t^1 is a Brownian motion. To prove that W_t^2 is also one, we have to check conditions 1–3 of Definition 1.

Let (t_1, t_2) and (t_3, t_4) be two non-intersecting intervals. The increment $W_{t_2}^2 - W_{t_1}^2$ has the form

$$W_{t_2}^2 - W_{t_1}^2 = \rho\left[\tilde{W}_{t_2}^1 - \tilde{W}_{t_1}^1\right] + \sqrt{1-\rho^2}\left[\tilde{W}_{t_2}^2 - \tilde{W}_{t_1}^2\right]. \tag{2.10}$$

Similarly,

$$W_{t_4}^2 - W_{t_3}^2 = \rho\left[\tilde{W}_{t_4}^1 - \tilde{W}_{t_3}^1\right] + \sqrt{1-\rho^2}\left[\tilde{W}_{t_4}^2 - \tilde{W}_{t_3}^2\right]. \tag{2.11}$$

We observe that each of the two terms on the right-hand side of Equation (2.10) is independent of both terms in Equation (2.11); satisfying condition 1.

Consider the increment (2.10). To show that the distribution is centred (i.e. its mean is zero), we can calculate the expectation of $W_{t_2}^2 - W_{t_1}^2$. Since $\tilde{W}_t^1$ and $\tilde{W}_t^2$ are Wiener processes, the expectation of each term inside the square brackets on the right side is zero:

$$\mathbb{E}\left[W_{t_2}^2 - W_{t_1}^2\right] = \rho\mathbb{E}\left[\tilde{W}_{t_2}^1 - \tilde{W}_{t_1}^1\right] + \sqrt{1-\rho^2}\mathbb{E}\left[\tilde{W}_{t_2}^2 - \tilde{W}_{t_1}^2\right] = 0.$$

Therefore, the increment $W_{t_2}^2 - W_{t_1}^2$ has a centered normal distribution. We now calculate its variance:

$$\begin{aligned}\mathrm{Var}\left[W_{t_2}^2 - W_{t_1}^2\right] &= \rho^2\mathrm{Var}\left[\tilde{W}_{t_2}^1 - \tilde{W}_{t_1}^1\right] + \left(1-\rho^2\right)\mathrm{Var}\left[\tilde{W}_{t_2}^2 - \tilde{W}_{t_1}^2\right] \\ &= \rho^2(t_2 - t_1) + (1-\rho)^2(t_2 - t_1) = t_2 - t_1,\end{aligned}$$

If $t = t_1$ and $t + s = t_2$, then the increment $\left(W_{t+s=t_2} - W_{t=t_1}\right) \sim \mathcal{N}(0, s = t_2 - t_1)$; satisfying condition 2.

Finally, the process W_t^2 is a linear combination of two processes with continuous sample paths and has a continuous sample path itself; satisfying condition 3. We calculate the correlation between the processes W_s^1 and W_t^2 and obtain

$$\begin{aligned}\mathbb{E}\left[W_s^1 W_t^2\right] &= \mathbb{E}\left[\tilde{W}_s^1\left(\rho\tilde{W}_t^1 + \sqrt{1-\rho^2}\tilde{W}_t^2\right)\right] \\ &= \rho\mathbb{E}\left[\tilde{W}_s^1\tilde{W}_t^1\right] + \sqrt{1-\rho^2}\mathbb{E}\left[\tilde{W}_s^1\tilde{W}_t^2\right] \\ &= \rho\min\{s,t\} + \sqrt{1-\rho^2}\cdot 0 \\ &= \rho\min\{s,t\}.\end{aligned}$$

Therefore, we have obtained the correlation between W_t^1 and W_t^2 as $\mathbb{E}\left[W_s^1 W_t^2\right] = \rho\min\{s,t\}$, which aligns with the correlation in the Heston model. QED

2.2.4 Discretization of the Heston Model

The Heston model assumes that movements in the asset price follow a continuous-time process. However, the measurement of asset prices occurs in discrete times, which means that the continuous-time Heston model needs to be discretized to simulate the asset's dynamics [11]. For a simple discretization scheme, we can use the GBM described in Theorem 1 to discretize the asset's price. Thus, at each time step Δt, we update the asset price by

$$S_{t+\Delta t} = S_t \times \exp\left(\left(r - \frac{1}{2}V_t\right)\Delta t + \sqrt{V_t}\Delta W_t\right), \tag{2.12}$$

where $\Delta W_t = z_t\sqrt{\Delta t}$ and $z_t \sim \mathcal{N}(0,1)$, i.e. z_t is normally distributed with mean zero, and variance one. Given that the volatility (as well as the price) is a stochastic process of the form

$$\mathrm{d}X_t = \mu(X_t, t)\mathrm{d}t + \sigma(X_t, t)\mathrm{d}W_t,$$

we can use the well-known Euler–Maruyama method. In particular, X_t represents the stochastic process; $\mu(X_t, t)$ and $\sigma(X_t, t)$ are the drift and diffusion terms, respectively; $\mathrm{d}t$ is the differential time increment, and $\mathrm{d}W_t$ is the differential Wiener process. The Euler–Maruyama method approximates the solution of the stochastic process by

$$X_{t+\Delta t} \approx X_t + \mu(X_t, t)\Delta t + \sigma(X_t, t)\Delta W_t. \tag{2.13}$$

In terms of volatility for the Heston model, this means that we can update the asset volatility at each time step by

$$V_{t+\Delta t} = V_t + \kappa(\theta - V_t)\Delta t + \sigma\sqrt{V_t}\Delta W_t, \tag{2.14}$$

with $\Delta W_t = z_t\sqrt{\Delta t} \sim \mathcal{N}\left(0, \sqrt{\Delta t}\right)$. Combining Equations (2.12) and (2.14), we obtain the following discretized system for the Heston model:

$$\begin{aligned} S_{t+\Delta t} &= S_t \times \exp\left(\left(r - \frac{1}{2}V_t\right)\Delta t + \sqrt{V_t}\Delta W_t\right), \\ V_{t+\Delta t} &= V_t + \kappa(\theta - V_t)\Delta t + \sigma\sqrt{V_t}\Delta W_t. \end{aligned} \tag{2.15}$$

As previously mentioned, the system represented in (2.15) serves as a simple discretization scheme for the Heston model. In practice, one could discretize the asset's price *and* volatility by using either something resembling Equation (2.12) or the Euler–Maruyama method from Equation (2.13). Our choice is, therefore, arbitrary, specifically under the assumption that the price has a closed-form solution under the Black–Scholes model, which could lead to a better approximation. In subsequent analyses, this scheme will be a benchmark for comparing option pricing through Monte Carlo approximation. At a later stage, Subsections 2.4.4–2.4.5 address further discretization schemes.

2.3 Monte Carlo Methods

The Monte Carlo Method is a simulation technique that implements statistical principles. It leverages the relationship between the probability of an event and its likelihood of occurrence. This process involves randomly sampling from a domain of possible outcomes, and considering the random observations that fall within a specific set to approximate the set's volume. According to the law of large numbers, the estimate approaches the true value as the sample size increases [9].

To further understand this concept, let us consider the following integral

$$\alpha = \int_{\mathbb{A}} f(x) g(x)\, \mathrm{d}x, \quad \text{with} \quad \int_{\mathbb{A}} g(x)\, \mathrm{d}x = 1.$$

More specifically, $g(x)$ is a probability density function and $f(x)$ is a function. The idea is to sample x_i $(i = 1, \ldots, n)$ values from the density $g(x)$. By evaluating the n sampled points at $f(x)$, and averaging them, we can obtain the unbiased estimator

$$\hat{\alpha}_n = \frac{1}{n} \sum_{i=1}^{n} f(x_i),$$

with sample standard deviation

$$s_f = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(f(x_i) - \hat{\alpha}_n \right)^2}. \tag{2.16}$$

Because f is integrable over the domain $\mathbb{A}$, the strong law of large numbers guarantees that

$$\mathbb{P}\left(\lim_{n \to \infty} \hat{\alpha}_n = \alpha \right) = 1.$$

Additionally, the error $|\hat{\alpha}_n - \alpha| \sim \mathcal{N}\left(0, s_f/\sqrt{n}\right)$ implies a rather slow square-root convergence rate for the method. Because of this, Monte Carlo methods are generally unsuitable for problems with small time steps, where faster convergence is desired. However, their convergence rate remains consistent for d-dimensional models, which makes Monte Carlo methods popular and well-suited for handling high-dimensional problems.

Monte Carlo simulations are widely-used tools for estimating option prices, particularly in complex models lacking closed-form solutions, as seen in the Heston model. For option pricing under the Heston model, we use Monte Carlo simulation to generate a large number of paths for the underlying asset and volatility, using a discretization scheme of our choice. By estimating the prices at each point in the simulation, we can calculate the corresponding payoff and discount it at the risk-free interest rate to determine the option's value.

2.3.1 Efficiency Improvements

The *efficiency* of Monte Carlo simulations (and any other numerical method) can be evaluated through the product of variance and computation time per run, as proposed in [9]

$$\sigma_1^2\tau_1 < \sigma_2^2\tau_2,$$

with σ_i and τ_i, $i = 1, 2$, being the standard error and the time needed to generate the ith replication, respectively. This inequality depicts the criterion for favouring estimator 1 over estimator 2. While we do not delve deeper into the derivation of this inequality, it will be employed to compare the models.

2.3.2 Variance Reductions Techniques

In efforts to improve the output obtained from Monte Carlo simulations, various variance reduction techniques are available. Two of them, control variate and antithetic variate are described below.

Control Variate

The idea behind the control variate method works in the following way: suppose that when collecting samples of identically distributed replications $\alpha_1, ..., \alpha_n$ for the Monte Carlo estimator $\hat{\alpha} = (\alpha_1 + ... + \alpha_n)/n$, we are also capable of generating auxiliary outputs X_i in the same fashion. Assume that $E[X]$ is known. We can then evaluate $\alpha_i(b) = \alpha_i - b(X_i - E[X])$ for the $i = 1, \dots, n$ replications, and obtain the following control variate estimator

$$\bar{\alpha}_{CV}(b) = \hat{\alpha} - b(\hat{X} - E[X]) = \frac{1}{n}\sum_{i=1}^{n}(\alpha_i - b(X_i - E[X])), \tag{2.17}$$

which is unbiased and consistent almost certainly [9]. We can reduce the variance by utilizing

$$b^* = \frac{\sigma_\alpha}{\sigma_X}\rho_{X\alpha} = \frac{Cov[X, \alpha]}{Var[X]}. \tag{2.18}$$

Should $E[\alpha]$ be unknown, b^* can then be estimated by

$$\hat{b}_n = \frac{\sum_{i=1}^{n}(X_i - \hat{X})(\alpha_i - \hat{\alpha})}{\sum_{i=1}^{n}(X_i - \hat{X})^2}, \tag{2.19}$$

although this estimator is not free of bias.

The estimator's variance is effectively reduced by incorporating this control variate into the estimation process. The key idea behind this technique is to find a suitable control variate correlated with the primary random variable of interest. Therefore, the reduction in variance is achieved through the negative correlation between the control variate and the primary random variable. When the control variate is negatively correlated with the primary variable, the fluctuations of the control variate tend to offset the primary variable's fluctuations, reducing the estimator's overall variability [9].

Antithetic Variate

The antithetic variate method is straightforward to implement. The general idea is to reduce the variance of the estimator by introducing a negative counterpart for each pair of replications.

The fundamental idea follows from the consideration that given a uniformly distributed random variable $U \sim \mathcal{U}(0,1)$, then the random variable $(1-U)$ is also uniformly distributed on $(0,1)$. In this sense, generating random paths using U_i $(i = 1, \ldots, n)$ as inputs, we can also generate random paths using $(1-U_i)$ $(i = 1, \ldots, n)$ as inputs without affecting the probability law of simulated processes. The variables $(U_i, 1-U_i)$ account for an antithetic pair.

This concept is further extended to other distributions by the inverse transform method [9]. In simulation by Monte Carlo, we are interested in generating antithetic pairs of independent and identically distributed (i.i.d.) random variables $Z_i \sim \mathcal{N}(0,1)$ and i.i.d. random variables $-Z_i \sim \mathcal{N}(0,1)$. We can then use the Z_is to simulate the Brownian motion paths and the $-Z_i$s to simulate the reflection of these paths about the origin.

In terms of variance reduction, imagine we want to find $E[X]$ for some random variable X. Initially, we make a sequence of observations in pairs $(X_i, \tilde{X}_i)$ for $i = 1, ..., n$. Note that the pairs are i.i.d., though the elements X_i and $\tilde{X}_i$ are not independent despite having the same distribution. The antithetic variate estimator is as follows:

$$\hat{X}_{AV} = \frac{1}{2n} + \left(\sum_{i=1}^{n} X_i + \sum_{i=1}^{n} \hat{X}_i \right) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{X_i + \hat{X}_i}{2} \right). \quad (2.20)$$

As $n \to \infty$, by the Central Limit Theorem

$$\frac{\hat{X}_{AV} - E[X]}{\sigma_{AV}/\sqrt{n}} \sim \mathcal{N}(0,1), \quad \text{with} \quad \sigma_{AV}^2 = Var\left[\frac{X_i + \hat{X}_i}{2}\right],$$

and a typical $1-\delta$ confidence interval can be constructed as

$$\hat{X}_{AV} \pm z_{\delta/2}\, s_{AV}/\sqrt{n}, \quad (2.21)$$

where s_{AV} is the sample standard deviation and is calculated from Equation (2.16).

In general, antithetic variate methods can reduce variance if the following condition is met

$$Cov[X_i, \tilde{X}_i] < 0.$$

This means that the negative dependence of the input random variables must produce negative covariance between the estimates of two paired replications [9].

2.3.3 Examples of Monte Carlo Algorithm Implementation

This section presents examples of Monte Carlo simulation algorithms. The following algorithm demonstrates the Monte Carlo simulation of the Heston model by discretizing the asset's price and variance paths using the system from Equation (2.15).

Algorithm 1: Monte Carlo Simulation of the Heston Model

Inputs : $S_0, V_0, r, \kappa, \theta, \sigma, \rho, T, N, M$
Outputs: S, V
$\mathrm{d}t \leftarrow T/N$;
generate two $M \times N$ standard normal random variables: $z_1, z_2 \sim \mathcal{N}(0, 1)$;
simulate the $M \times N$ Brownian motions;
$\mathrm{d}W^1 \leftarrow \sqrt{\mathrm{d}t} \times z_1$;
$\mathrm{d}W^2 \leftarrow \rho \times \mathrm{d}W^1 + \sqrt{\mathrm{d}t} \times \sqrt{1-\rho^2} \times z_2$;
initiate two zero $M \times N$ matrices to hold the values of S and V;
assign starting price and volatility;
$S(:, 1) \leftarrow S_0$;
$V(:, 1) \leftarrow V_0$;
calculate price and volatility at each time step;
for $i \leftarrow 1$ **to** N **do**
  $S(:, i+1) \leftarrow S(:, i) \times \exp\left(\left(r - \frac{1}{2}V(:, i)\right)\mathrm{d}t + \sqrt{V(:, i)}\mathrm{d}W^1\right)$;
  $V(:, i+1) \leftarrow V(:, i) + \kappa\left(\theta - V(:, i)\right)\mathrm{d}t + \sigma\sqrt{V(:, i)}\mathrm{d}W^2$;
end
return S, V;

In practice, particularly for European and Asian options, one could take the values generated for S, check the value of the payoff function at maturity for all generated M–paths (European option), or calculate the average along every generated path to calculate the Asian payoff function of every path. After obtaining the respective payoff values, we can perform the Monte Carlo estimator by averaging these values and discounting the result at the risk–free rate.

Next, we present an algorithm that incorporates the antithetic variate variance reduction technique discussed in the previous section. The algorithm deals with an asset whose price follows a GBM for simplicity and illustration purposes.

Algorithm 2: Monte Carlo Simulation with antithetic variate

Inputs : S_0, σ, r, T, N, M
Outputs: S
$\mathrm{d}t \leftarrow T/N$;
Generate a $M \times N$ standard normal random variable: $z \sim \mathcal{N}(0, 1)$;
Simulate a $M \times N$ Brownian motion and its negative for the antithetic path;
$\mathrm{d}W^1 \leftarrow \sqrt{\mathrm{d}t} \times z_1$;
$\mathrm{d}W^2 \leftarrow -\mathrm{d}W^1$;
Initiate two zero $M \times N$ matrices to hold the values of S_1 and S_2;
Assign starting price;
$S_1(:, 1) \leftarrow S_0$;
$S_2(:, 1) \leftarrow S_0$;
Generate stock price path and antithetic path;
for $i \leftarrow 1$ **to** N **do**
$\quad S_1(:, i+1) \leftarrow S(:, i) \times \exp\left(\left(r - \frac{1}{2}V(:, i)\right)\mathrm{d}t + \sqrt{V(:, i)}\mathrm{d}W^1\right)$;
$\quad S_2(:, i+1) \leftarrow S(:, i) \times \exp\left(\left(r - \frac{1}{2}V(:, i)\right)\mathrm{d}t + \sqrt{V(:, i)}\mathrm{d}W^2\right)$;
end
Calculate the expectations of each path;
$S_1 \leftarrow \sum_{i=1}^{M} S_1(:, N)$;
$S_2 \leftarrow \sum_{i=1}^{M} S_2(:, N)$;
Calculate the antithetic estimate;
$S \leftarrow \frac{1}{2}(S_1 + S_2)$;
return S;

2.4 Stochastic Runge–Kutta Lawson Schemes for SDEs

This section features the construction of the general SRKL scheme proposed in [6]. The implementation of SRKL schemes to the Heston model is outlined, with specific emphasis placed on the SRKL Euler-Maruyama and SRKL Midpoint schemes.

2.4.1 Construction of SRKL Schemes

ODEs are frequently solved with numerical methods such as Runge–Kutta (RK), which considers derivatives at multiple points of an interval and yields dependable estimates of the solution. Compared to more straightforward methods such as Euler–Maruyama, the generalized RK method addresses a broader range of ODEs, including stiff systems, by offering greater flexibility in selecting intermediate steps and corresponding weights. Although this method requires additional iterations, it achieves better accuracy at the expense of higher computational costs [26].

In contrast, Stochastic Runge–Kutta (SRK) schemes address these challenges by incorporating techniques such as Itô calculus to appropriately account for the stochastic terms and maintain the desired properties of SDEs.

The SRKL schemes expand upon the SRK schemes by integrating Lawson methods. Lawson methods are fractional step methods that split the original problem into multiple sub-problems, which are easier to solve independently, resulting in a superior integration technique [17]. Particularly, SRKL schemes are applicable to a system of stochastic differential equations of the form [6]

$$\mathrm{d}\mathbf{X}(t) = \sum_{m=0}^{M} (A_m \mathbf{X}(t) + \mathbf{g}_m(t, \mathbf{X}(t)))\,\mathrm{d}W_m(t), \qquad \mathbf{X}(0) = \mathbf{x}_0, \tag{2.22}$$

where $W(0) = t$, $\mathbf{X}(t) \in \mathbb{R}^d$, $W_1(t), \ldots, W_M(t)$ are independent Brownian motions, and the subsequent matrices A_k and A_l are constant and commute, i.e. $\mathbf{A}_k\mathbf{A}_l - \mathbf{A}_l\mathbf{A}_k = 0$.

A numerical method for solving SDEs having strong convergence ensures accurate singular trajectories of a stochastic process. If there is weak convergence, it instead ensures accurate results for the expected value of many trajectories. In finance, both types of convergence are relevant in the sense that an asset can show long-term trends, but also be volatile in the short-term. The former relates to weak convergence, and the latter to strong convergence. Since SRKL includes both weak and strong convergence properties, the scheme is relevant to the field.

In the case of the Black–Scholes equation, it is evident that

$$\begin{aligned} &d = M = 1, \quad && A_0 = r, \\ &A_1 = \sigma, && g_0(t, \mathbf{X}(t)) = g_1(t, \mathbf{X}(t)) = 0. \end{aligned}$$

Furthermore, it is clear that the matrices A_0 and A_1 commute. In the Heston model we have

$$d = M = 2, \qquad A_0 = \begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix},$$
$$A_1 = A_2 = \mathbf{0}_{2\times 2}, \qquad g_0(t, \mathbf{X}(t)) = \begin{bmatrix} 0 \\ \kappa\theta \end{bmatrix},$$
$$g_1(t, \mathbf{X}(t)) = \begin{bmatrix} \sqrt{V(t)}S(t) \\ \sigma\rho\sqrt{V(t)} \end{bmatrix}, \quad g_2(t, \mathbf{X}(t)) = \begin{bmatrix} 0 \\ \sigma\sqrt{1-\rho^2}\sqrt{V(t)} \end{bmatrix}.$$

Clearly, the matrices A_k pairwise also commute.

To implement the SRKL scheme, the procedure is as follows: Consider an option with a maturity $T > t_0$, where $t_0 < t_1 < \cdots < t_N = T$. Let $h_n = t_{n+1} - t_n$ for $0 \leq n \leq N-1$. Next, define the following for each n:

$$L^n(t) = \left(A_0 - \frac{1}{2}\sum_{m=1}^{M} A_m^2\right)(t - t_n) + \sum_{m=1}^{M} A_m(W_m(t) - W_m(t_n)), \qquad t_n < t \leq t_{n+1}. \tag{2.23}$$

Specifically, for the Black–Scholes model, we obtain

$$L^n(t) = (r - \frac{1}{2}\sigma^2)(t - t_n) + \sigma(W(t) - W(t_n)),$$

while for the Heston model, we have

$$L^n(t) = \begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix}(t - t_n).$$

We define $\tilde{g}_0(t, \mathbf{x}(t))$ by subtracting the sum of $A_m g_m(t, \mathbf{x}(t))$ from $g_0(t, \mathbf{x}(t))$, resulting in

$$\tilde{g}_0(t, \mathbf{x}(t)) = g_0(t, \mathbf{x}(t)) - \sum_{m=1}^{M} A_m g_m(t, \mathbf{x}(t)).$$

For the Black–Scholes model, we have

$$\tilde{g}_0(t, \mathbf{x}) = 0,$$

while for the Heston model,

$$\tilde{\mathbf{g}}_0(t, \mathbf{x}) = \begin{bmatrix} 0 \\ \kappa\theta \end{bmatrix}.$$

Let $\tilde{g}_m(t, \mathbf{x}) = g_m(t, \mathbf{x})$ for $m > 0$. According to Debrabant et al. [6, Lemma 2.1], the function

$$\mathbf{V}_n(t) = \exp(-L^n(t))\mathbf{X}(t),$$

satisfies the system

$$\mathrm{d}\mathbf{V}_n(t) = \sum_{m=0}^{M} \exp(-L^n(t))\tilde{\mathbf{g}}_m\left(t, \exp\left(L^n(t)\right)\mathbf{x}(\mathbf{t})\right) \mathrm{d}W_m(t), \qquad \mathbf{V}_n(t_n) = \mathbf{X}(t_n), \tag{2.24}$$

on the interval $[t_n, t_{n+1}]$.

Thus, for the Black–Scholes and the Heston models, Equation (2.24) takes the form:

$$\mathrm{d}\mathbf{V}_n(t) = \exp\left(-(r - \frac{1}{2}\sigma^2)(t - t_n) + \sigma(W(t) - W(t_n))\right)\mathrm{d}t, \tag{2.25}$$

and

$$\mathrm{d}\mathbf{V}_n^1(t) = \exp\left(-r(t - t_n)\sqrt{V(t)}S(t)\,\mathrm{d}W_1(t)\right), \tag{2.26a}$$

$$\mathrm{d}\mathbf{V}_n^2(t) = \exp(\kappa(t - t_n))\left(\kappa\theta\mathrm{d}t + \sigma\sqrt{V(t)}\left(\rho\mathrm{d}W_1(t) + \sqrt{1 - \rho^2}\mathrm{d}W_2(t)\right)\right), \tag{2.26b}$$

respectively.

We observe that the equation for the Black–Scholes equation is trivial. For the Heston equation, we apply [6, Subsection 2.2]. Let s be a positive integer, $Z_{ij}^{m,n}$ and $z_i^{m,n}$, $1 \leq i \leq s$, $1 \leq j \leq s$, $0 \leq m \leq 2$ be random variables, let

$$c_m^{n,i} = \sum_{j=1}^{s} Z_{ij}^{m,n}, \qquad \Delta W^{n,m} = c_m^n = \sum_{i=1}^{s} z_i^{m,n},$$
$$\Delta L_i^n = \begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix} c_0^{n,i}, \qquad \Delta L^n = \begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix} c_0^n.$$

Lastly, the SRKL scheme takes the form of [6, Equation (2.12)]:

$$\begin{aligned}
\mathbf{H}_i &= \mathbf{Y}_n + \sum_{j=1}^{s} \exp(-\Delta L_j^n) \sum_{m=0}^{2} Z_{ij}^{m,n} \tilde{g}_m(t_n + c_0^{n,j}, \exp(\Delta L_j^n)\mathbf{H}_j), \\
\mathbf{V}_n^{n+1} &= \mathbf{Y}_n + \sum_{i=1}^{s} \exp(-\Delta L_i^n) \sum_{m=0}^{2} z_i^{m,n} \tilde{g}_m(t_n + c_0^{n,i}, \exp(\Delta L_i^n)\mathbf{H}_i), \\
\mathbf{Y}_{n+1} &= \exp(\Delta L^n)\mathbf{V}_n^{n+1}.
\end{aligned} \tag{2.27}$$

2.4.2 Application of SRKL Schemes to the Heston Model

Given that the Black–Scholes SDE has a closed-form solution, eliminating the need for approximation, numerical methods such as SRKL are not necessary. In the case of the Heston model, we can employ the general SRKL discretization scheme described by Equations (2.27) as follows:

$$
\begin{aligned}
\boldsymbol{H}_i &= \boldsymbol{Y}_n + \sum_{j=1}^{s} e^{-\begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix} c_0^{n,j}} \sum_{m=0}^{2} Z_{ij}^{m,n} \tilde{\boldsymbol{g}}_m (t_n + c_0^{n,j}, e^{-\begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix} c_0^{n,j}} \boldsymbol{H}_j), \\
\boldsymbol{V}_{n+1}^{n} &= \boldsymbol{Y}_n + \sum_{i=1}^{s} e^{-\begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix} c_0^{n,i}} \sum_{m=0}^{2} z_i^{m,n} \tilde{\boldsymbol{g}}_m (t_n + c_0^{n,i}, e^{-\begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix} c_0^{n,i}} \boldsymbol{H}_i), \\
\boldsymbol{Y}_{n+1} &= e^{\begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix} c_0^{n}} \boldsymbol{V}_{n+1}^{n}.
\end{aligned}
$$

The derivation of this result is postponed to Appendix B.

2.4.3 Drift– and Full–Stochastic Schemes

Numerical schemes which are formulated by considering constant step sizes and setting the matrices $A_m = 0$ for $m > 0$, allow for calculating the exponential term $\exp(L^n(t))$ only once, which significantly reduces computational costs. These are known as Drift Stochastic Lawson (DSL) schemes.

On the contrary, Full Stochastic Lawson (FSL) schemes contain at least one non-zero linear diffusion term in the operator $L^n(t)$ and the matrix exponentials $\exp(L^n(t))$ must be determined again at every time step. Because the solution of the implicit equation is solved with a single iteration of Newton's method, the FSL neglects any computational advantage from DSL schemes [6]. Thorough examples on this topic can be found in [7].

In the next sections we provide the formulation for two specific cases of SRKL schemes as presented in [6]: the Euler–Maruyama DSL and the Midpoint FSL schemes. These schemes will then serve as the primary models for comparison with traditional models. We also mention their weak and strong order of convergences, and comment on how they work.

The Euler–Maruyama DSL scheme is an explicit numerical method that performs iterations following a solution of the current time step. In contrast, implicit methods, such as the Midpoint scheme FSL, require iterative numerical methods and computer processing because the solutions at the following stage have an implicit dependency which is challenging to solve analytically. The solution approach often involves starting with an initial solution guess, and refining it iteratively until certain desired accuracy is obtained [15].

Implicit approaches are often preferred for models with highly nonlinear or irregular dynamics due to their ability to provide greater accuracy and stability. In the context of the SRKL scheme and the Heston model, the choice between explicit and implicit formulations depends on the specific coefficients and parameters involved.

2.4.4 Euler–Maruyama Drift Stochastic Lawson Scheme

The coefficients for the Euler–Maruyama DSL scheme, as found in [6, p. 387], are $s = 1, Z_{11}^{m,n} = 0, z_1^{m,n} = \Delta W_m^n$. Recall that for the Heston model, $A_0 = \begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix}$, $\Delta L^n = A_0 c_0^n$, and $c_0^n = \Delta W_0^n$. Thus, together with (2.27) we have:

$$
\begin{aligned}
H_i &= Y_n \\
V_{n+1}^n &= Y_n + [g_0 \Delta W_0^n + g_1 \Delta W_1^n + g_2 \Delta W_2^n] \\
Y_{n+1} &= e^{A_0 \Delta W_0^n} \cdot V_{n+1}^n.
\end{aligned}
$$

Through discretization into small steps and the use of the most recent value with the addition of a random term driven by a Wiener process, the Euler–Maruyama scheme includes both deterministic and stochastic structures. The popularity of this scheme comes from its simplicity and explicit nature. However, the results are potentially less reliable than those of other schemes. Nevertheless, due to having only first-order convergence, errors converge to zero proportionally to the step sizes h.

2.4.5 Midpoint Full Stochastic Lawson Scheme

The coefficients for the Midpoint FSL scheme are given by [6, p. 388], $s = 1, Z_{11}^{m,n} = 1/2 \Delta W_m^n, z_1^{m,n} = \Delta W_m^n$, this scheme for the Heston model is given by:

$$
\begin{aligned}
H_1 &= Y_n + \frac{e^{-\frac{1}{2}\Delta L^n}}{2} [g_0(t_n + \frac{h_n}{2}, e^{\frac{1}{2}\Delta L^n} H_1) \Delta W_0^n \\
&\quad + g_1(t_n + \frac{h_n}{2}, e^{\frac{1}{2}\Delta L^n} H_1) \Delta W_1^n \\
&\quad + g_2(t_n + \frac{h_n}{2}, e^{\frac{1}{2}\Delta L^n} H_1) \Delta W_2^n], \\
V_{n+1}^n &= Y_n + e^{-\frac{1}{2}\Delta L^n} [g_0(t_n + \frac{h_n}{2}, e^{\frac{1}{2}\Delta L^n} H_1) \Delta W_0^n \\
&\quad + g_1(t_n + \frac{h_n}{2}, e^{\frac{1}{2}\Delta L^n} H_1) \Delta W_1^n \\
&\quad + g_2(t_n + \frac{h_n}{2}, e^{\frac{1}{2}\Delta L^n} H_1) \Delta W_2^n], \\
Y_{n+1} &= e^{\Delta L^n} V_{n+1}^n.
\end{aligned}
$$

Using $H_1 = \frac{1}{2}(V_{n+1}^n + Y_n)$, we can simplify further:

$$
\begin{aligned}
Y_{n+1} = e^{\Delta L^n} Y_n + e^{\frac{1}{2}\Delta L^n} &\cdot [g_0(t_n + \frac{h_n}{2}, \frac{1}{2}(e^{\frac{1}{2}\Delta L^n} Y_n + e^{-\frac{1}{2}\Delta L^n} Y_{n+1})) \Delta W_0^n \\
&+ g_1(t_n + \frac{h_n}{2}, \frac{1}{2}(e^{\frac{1}{2}\Delta L^n} Y_n + e^{-\frac{1}{2}\Delta L^n} Y_{n+1})) \Delta W_1^n \\
&+ g_2(t_n + \frac{h_n}{2}, \frac{1}{2}(e^{\frac{1}{2}\Delta L^n} Y_n + e^{-\frac{1}{2}\Delta L^n} Y_{n+1})) \Delta W_2^n].
\end{aligned}
$$

If the random process associated with the SDE is commutative, the scheme exhibits a mean-square (strong) convergence of order 1; otherwise, the convergence order is 0.5. Additionally, the scheme has a weak convergence of order 1, and is a computationally efficient implicit scheme. It uses the middle value of current and subsequent time steps together with the slope at the current point to update the solution.

Chapter 3

Methodology

This section presents the methodology employed in this study to price European and Asian call options under the Heston model by applying selected numerical methods. The main objectives were to enhance options pricing accuracy and efficiency by implementing the SRKL Midpoint, and Euler–Maruyama schemes proposed in [6], and compare with the more traditional Monte Carlo method.

Two variance reduction techniques, antithetic and control variate, were utilized to improve the efficiency of each method. For the European option, the control variate was chosen by estimating another European option with a different strike price. In contrast, for the arithmetic Asian option we utilized a geometric Asian call price as the control variate. We opted for these variance reduction techniques due to their straightforward implementation [9].

Moreover, we devised custom MATLAB codes to discretize and simulate the SDEs of the Heston model within each numerical method, applying the theoretical framework formulated in Chapters 2 and 3. To ensure accurate implementation of the SRKL schemes to the Heston model, we adapted the MATLAB functions `MidpointFSLVectorized.m` and `EulerDSLVectorized.m` provided in [5], chosen for their accuracy and suitability, as explained in Section 2.4. The adaptation of these functions can be found in Appendix A. For transparency, we performed the computations using MATLAB R2022 on a PC with an AMD Ryzen 5 4600H processor, with Radeon Graphics @ 3.00 GHz and 16 GB of RAM.

An accuracy analysis for every method was carried out by establishing benchmark values for both option types. European options were benchmarked against the Black–Scholes formula (Theorem 2), while Asian options utilized the price obtained from a standard Monte Carlo method as a benchmark. Efficiency metrics were calculated by multiplying the squared standard error by the computational time, as shown in Section 2.3.1, to enable another comparison metric between the methods.

Furthermore, a sensitivity analysis was performed for every method. The sensitivity analyses involved simultaneous variations of two variables to observe their impact on option prices. In particular, this concerned varying the initial price (S_0) and the initial volatility

(V_0), varying the initial price and the strike price (K), and varying the initial volatility and the correlation (ρ). The selection of pairwise varying variables was made to observe the effects of stochastic volatility in option prices, determine the effects of moneyness in each method, and identify the importance of the correlation between the price and volatility in the Heston model, respectively. Additionally, we conducted a convergence analysis by varying the number of step sizes and simulations to assess the stability of the methods.

Inputs for the pricing models encompassed initial asset price (S_0), strike price (K), time to expiration (T), risk-free interest rate (r), initial volatility (V_0), volatility of volatility (σ), mean of variance (θ), rate of mean reversion (κ), and correlation between the asset price and volatility (ρ). The particular choice of inputs was arbitrary, and is presented in the next section.

Chapter 4

Results

This chapter presents the results obtained from the MATLAB codes in Appendix A using suitable tables and figures. The parameter inputs for the Heston model were as follows:

- $S_0 = 80$
- $K = 85$
- $\theta = 0.05$
- $V_0 = 0.04$
- $T = 1$
- $\sigma = 0.2$
- $r = 0.05$
- $\kappa = 1$
- $\rho = -0.7$

Specified values vary for the heat maps presented later in Section 4.2.

4.1 Accuracy and Efficiency

4.1.1 European Call Option

Method	τ (ms)	C_{EU}	σ	$\sigma^2\tau$	Abs. Err. (%)
Std. MC	180.99	5.9984	9.3013×10^{-2}	1.5658×10^{-3}	1.7011×10^{-1}
A. MC	363.06	5.9492	6.5527×10^{-2}	1.5589×10^{-3}	6.5134×10^{-1}
C. MC	201.88	6.0946	9.5178×10^{-2}	1.8288×10^{-3}	1.7757
Midpoint	1674.30	5.3426	9.0087×10^{-2}	1.3588×10^{-2}	10.783
A. Midpoint	3279.80	5.5293	6.4428×10^{-2}	1.3614×10^{-2}	7.6639
C. Midpoint	1771.70	5.5824	9.1433×10^{-2}	1.4812×10^{-2}	6.7778
Euler	200.92	6.0327	9.3468×10^{-2}	1.7553×10^{-3}	7.4266×10^{-1}
A. Euler	448.07	5.9857	6.5878×10^{-2}	1.9446×10^{-3}	4.2861×10^{-2}
C. Euler	176.52	6.1247	9.5760×10^{-3}	1.6187×10^{-3}	2.278
Black–Scholes	–	5.9882	–	–	–

Table 1: The table presents performance metrics for a European call option under the Heston model, including computational time (τ), call price (C_{EU}), standard error (σ), and efficiency ($\sigma^2\tau$). The calculations are based on different computational schemes and the application of variance reduction techniques. Furthermore, the table includes the absolute error, which was determined by comparing the computed price with the Black-Scholes price for a European call option as a benchmark.

4.1.2 Asian Call Option

Method	τ (ms)	C_A	σ_A	C_G	σ_G
Std. MC	254.63	2.3546	4.4002×10^{-2}	2.2368	4.2281×10^{-2}
A. MC	566.21	2.3559	3.0927×10^{-2}	2.2385	2.9722×10^{-2}
C. MC	280.23	2.3918	4.4169×10^{-2}	2.2715	4.2420×10^{-2}
Midpoint	1761.60	2.0612	4.1099×10^{-2}	1.9528	3.9390×10^{-2}
A. Midpoint	3530.90	2.1591	2.9791×10^{-2}	2.0455	2.8571×10^{-2}
C. Midpoint	1809.80	2.1923	3.7948×10^{-3}	2.0762	4.0817×10^{-2}
Euler	294.75	2.2748	4.2968×10^{-2}	2.1611	4.1274×10^{-2}
A. Euler	551.55	2.3739	3.1113×10^{-2}	2.2554	2.9898×10^{-2}
C. Euler	295.95	2.4098	3.7672×10^{-3}	2.2884	4.2667×10^{-2}
		Arithmetic		Geometric	

Table 2: The table presents performance metrics for an arithmetic and geometric Asian call option under the Heston model, including computational time (τ), call price ($C_{\cdot}$), and standard error ($\sigma_{\cdot}$). The calculations are based on different computational schemes and the application of variance reduction techniques. Furthermore, the computational time corresponds to a simultaneous calculation of the arithmetic and the geometric call.

Method	$\sigma_A^2\tau$	Abs. Err. (%)	$\sigma_G^2\tau$	Abs. Err. (%)
A. MC	5.4157×10^{-4}	5.5479×10^{-2}	5.0019×10^{-4}	7.4495×10^{-2}
C. MC	5.4672×10^{-4}	1.5789	5.0427×10^{-4}	1.5503
Midpoint	2.9755×10^{-3}	12.461	2.7331×10^{-4}	12.695
A. Midpoint	3.1337×10^{-3}	8.3024	2.8822×10^{-4}	8.5506
C. Midpoint	2.6063×10^{-5}	6.8924	3.0152×10^{-4}	7.1798
Euler	5.4419×10^{-4}	3.3902	5.0214×10^{-4}	3.3843
A. Euler	5.3390×10^{-4}	8.2085×10^{-1}	4.9301×10^{-4}	8.3082×10^{-1}
C. Euler	4.2000×10^{-6}	2.3422	5.3878×10^{-4}	2.3076
Std. MC	4.930×10^{-4}	–	4.2281×10^{-2}	–
	Arithmetic		Geometric	

Table 3: The table presents performance metrics for an arithmetic and geometric Asian call option under the Heston model, including efficiency ($\sigma_{\cdot}^2\tau$), and absolute error. The calculations are based on Table 2 results. Furthermore, the absolute errors were determined by comparing the computed price with the standard Monte Carlo price for an arithmetic and geometric call option as benchmark values.

4.2 Sensitivity Analysis – European Call Option

4.2.1 Volatility Vs. Underlying

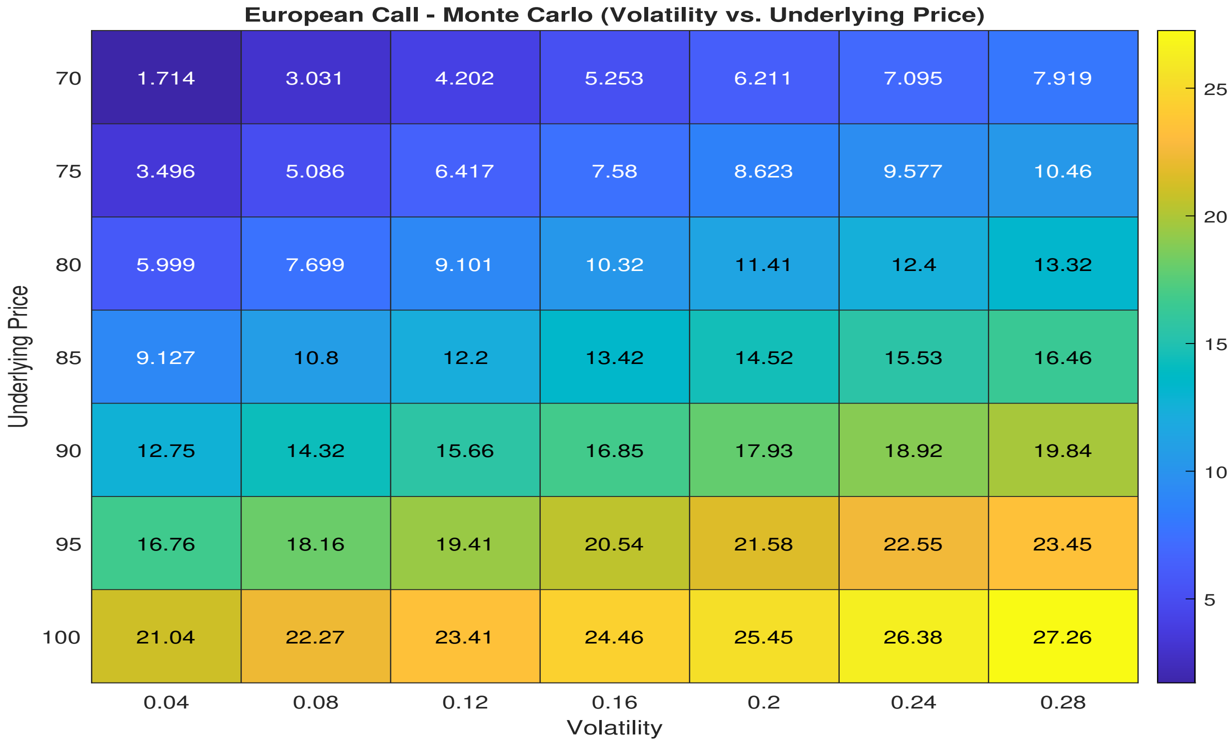

Figure 1: The figure shows a heat-map of European call option prices with Monte Carlo methods and variable initial price (S_0) and volatility (V_0). The values for V_0 are shown on the x-axis and the values for S_0 on the y-axis. Darker shades indicate lower prices and lighter shades represent higher prices.

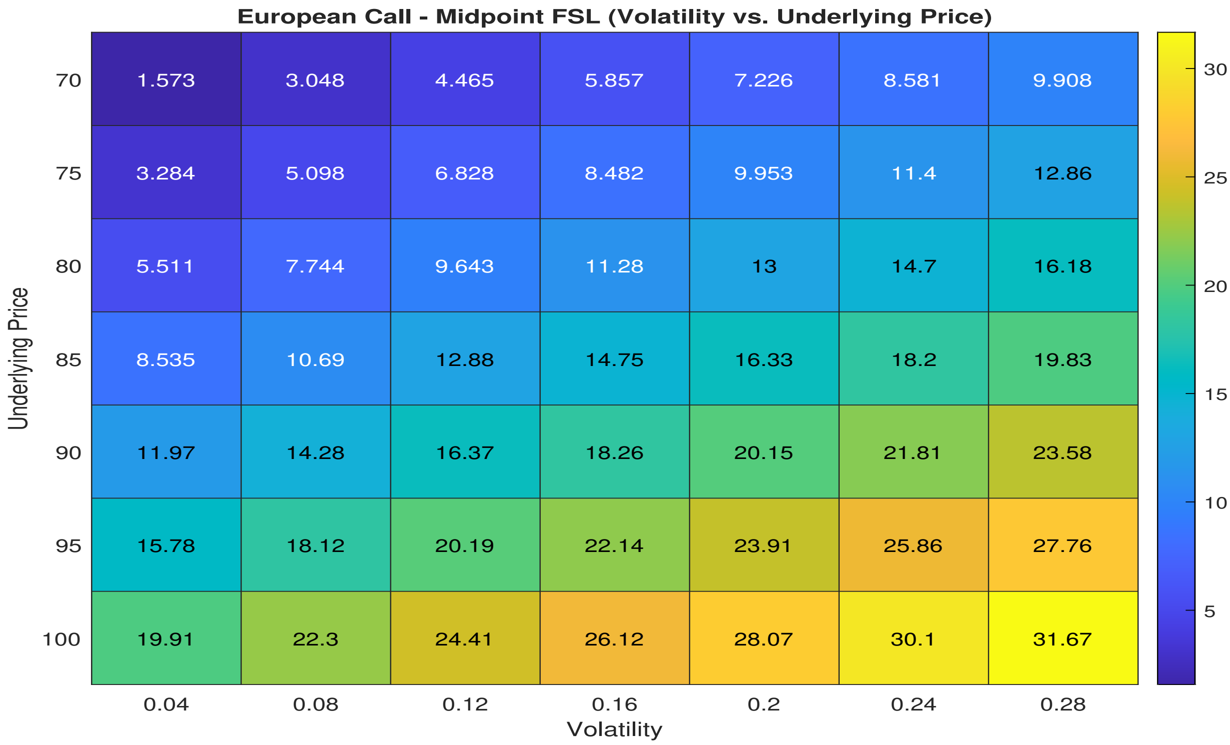

Figure 2: The figure shows a heat-map of European call option prices with SRKL Midpoint schemes and variable initial price (S_0) and volatility (V_0).

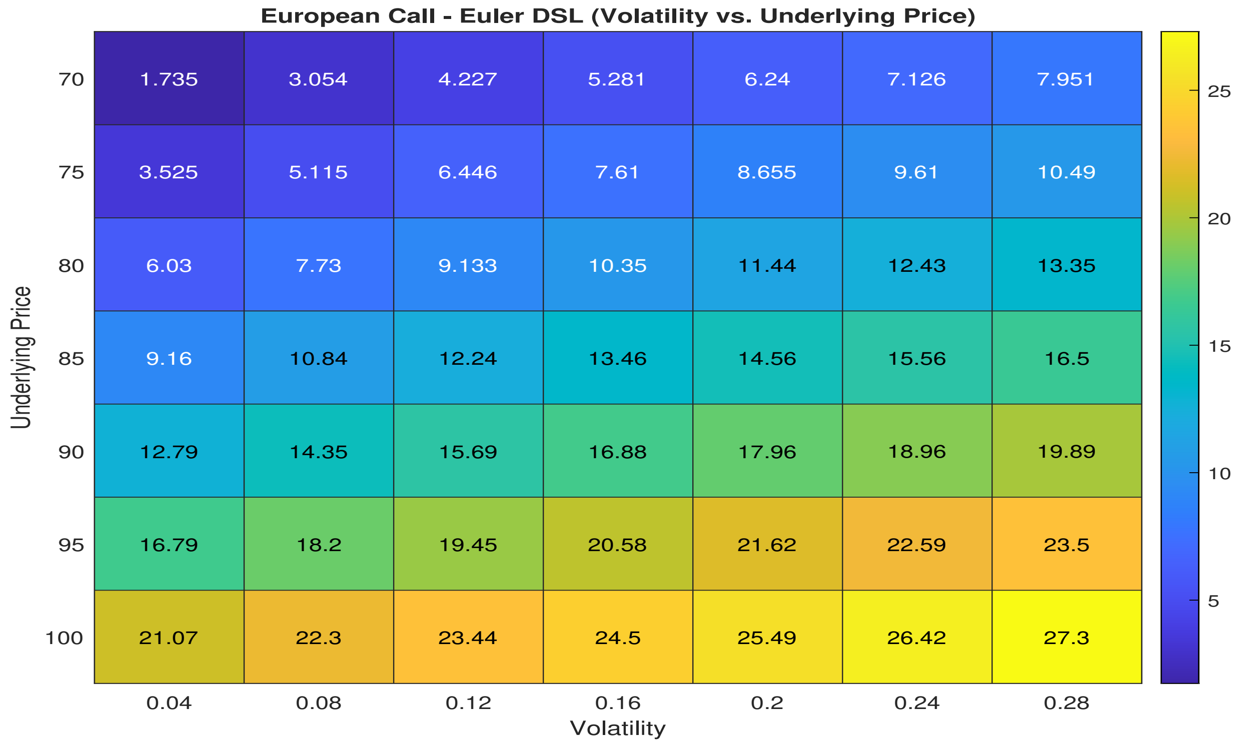

Figure 3: The figure shows a heat-map of European call option prices with SRKL Euler schemes and variable initial price (S_0) and volatility (V_0).

4.2.2 Strike Vs. Underlying

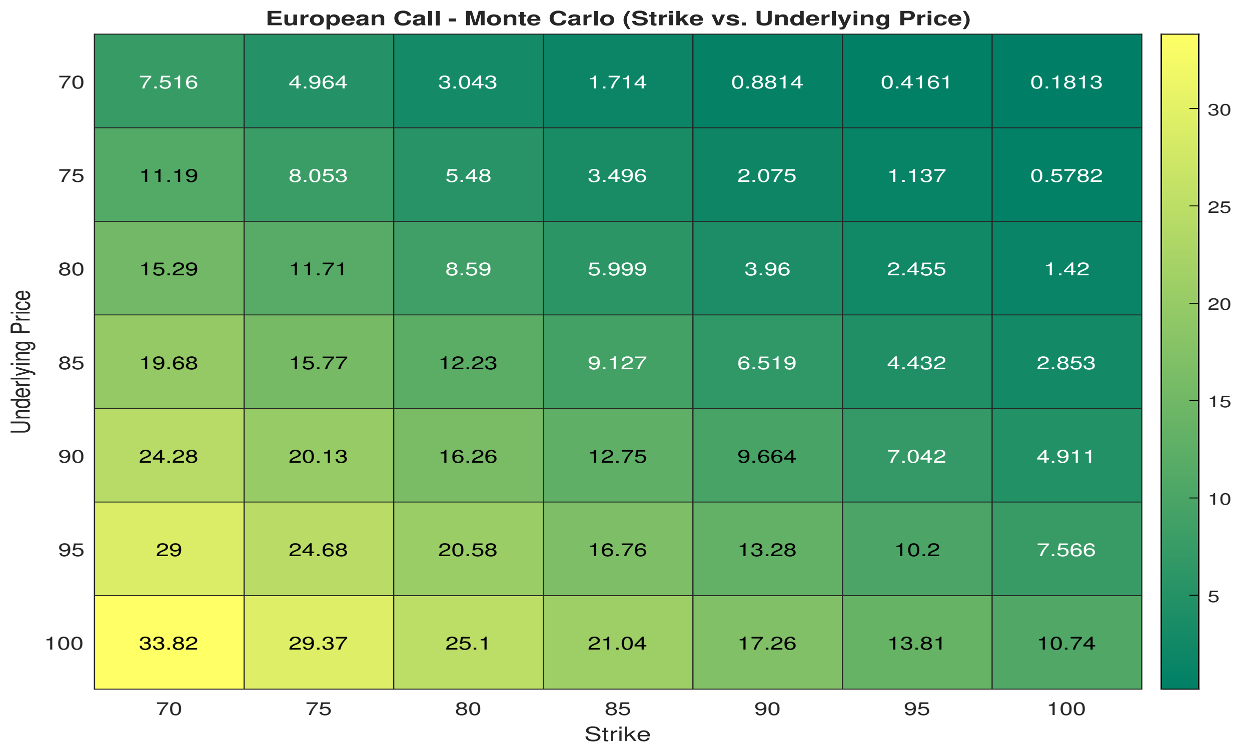

Figure 4: The figure shows a heat-map of European call option prices with Monte Carlo method and variable initial price (S_0) and strike price (K). Prices on the diagonal from top-left to bottom-right represent ATM options, and prices over and under the diagonal represent OTM and ITM options, respectively. Darker shades indicate lower prices and lighter shades represent higher prices.

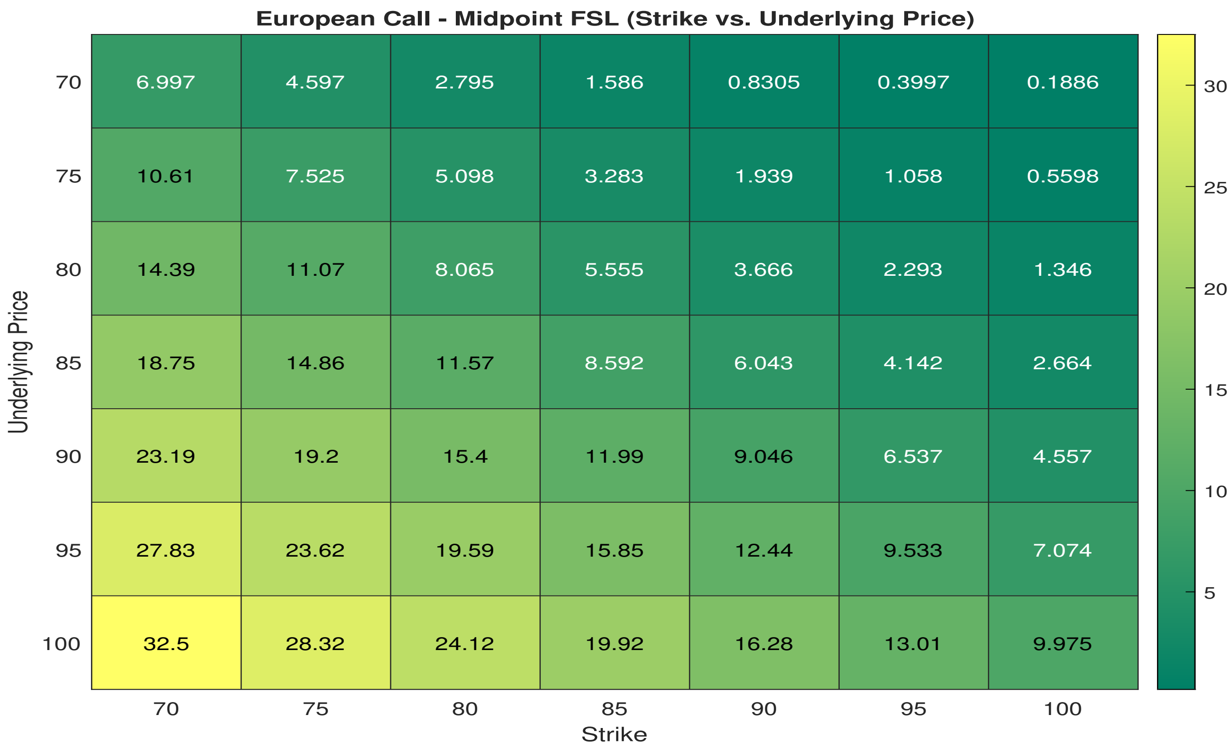

Figure 5: The figure shows a heat-map of European call option prices with SRKL Midpoint schemes and variable initial price (S_0) and strike price (K).

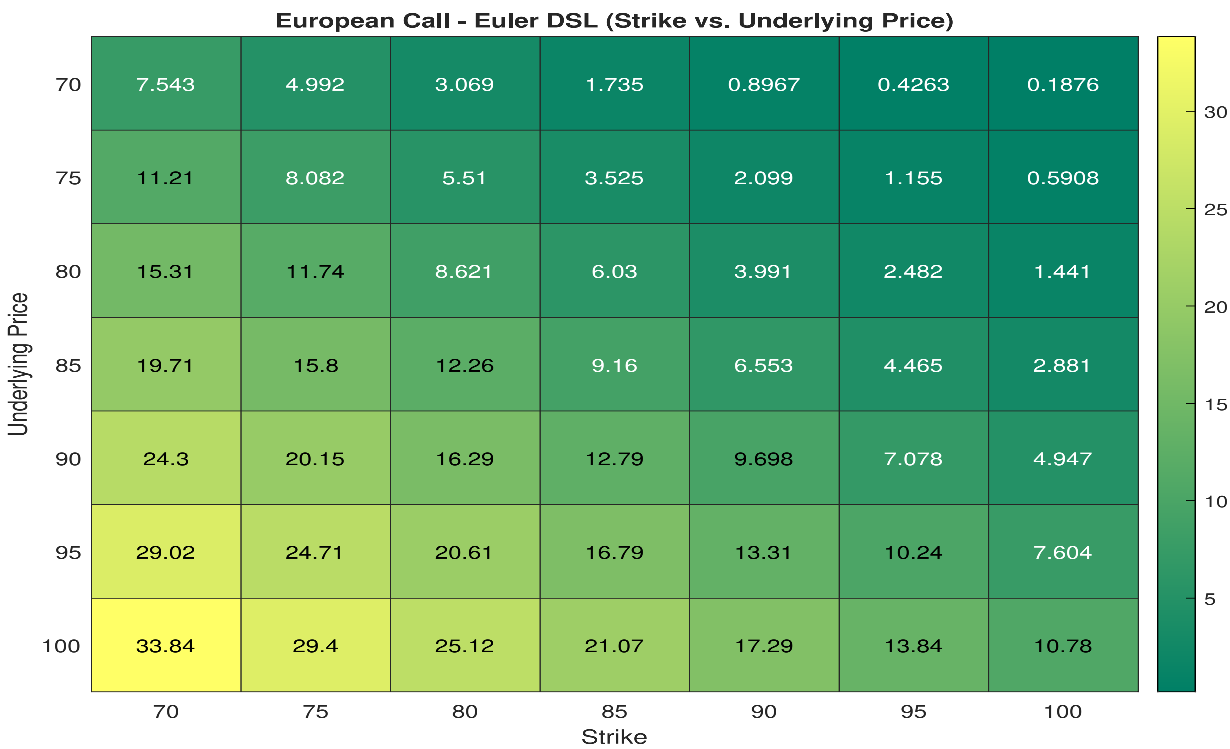

Figure 6: The figure shows a heat-map of European call option prices with SRKL Euler schemes and variable initial price (S_0) and strike price (K).

4.2.3 Correlation Vs. Volatility

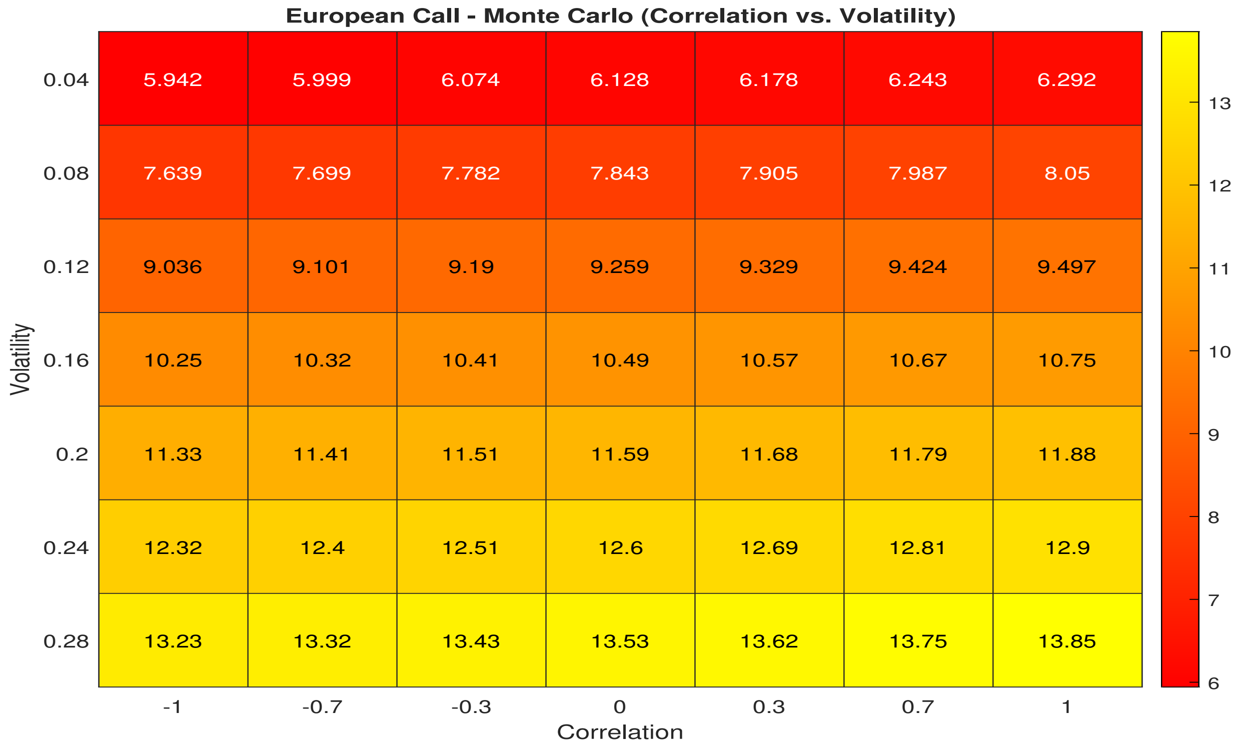

Figure 7: The figure shows a heat-map of European call option prices with Monte Carlo method and variable initial volatility (V_0) and correlation (ρ). The values for ρ are shown on the x-axis, and the values for V_0 are shown on the y-axis. Darker shades indicate lower prices and lighter shades represent higher prices.

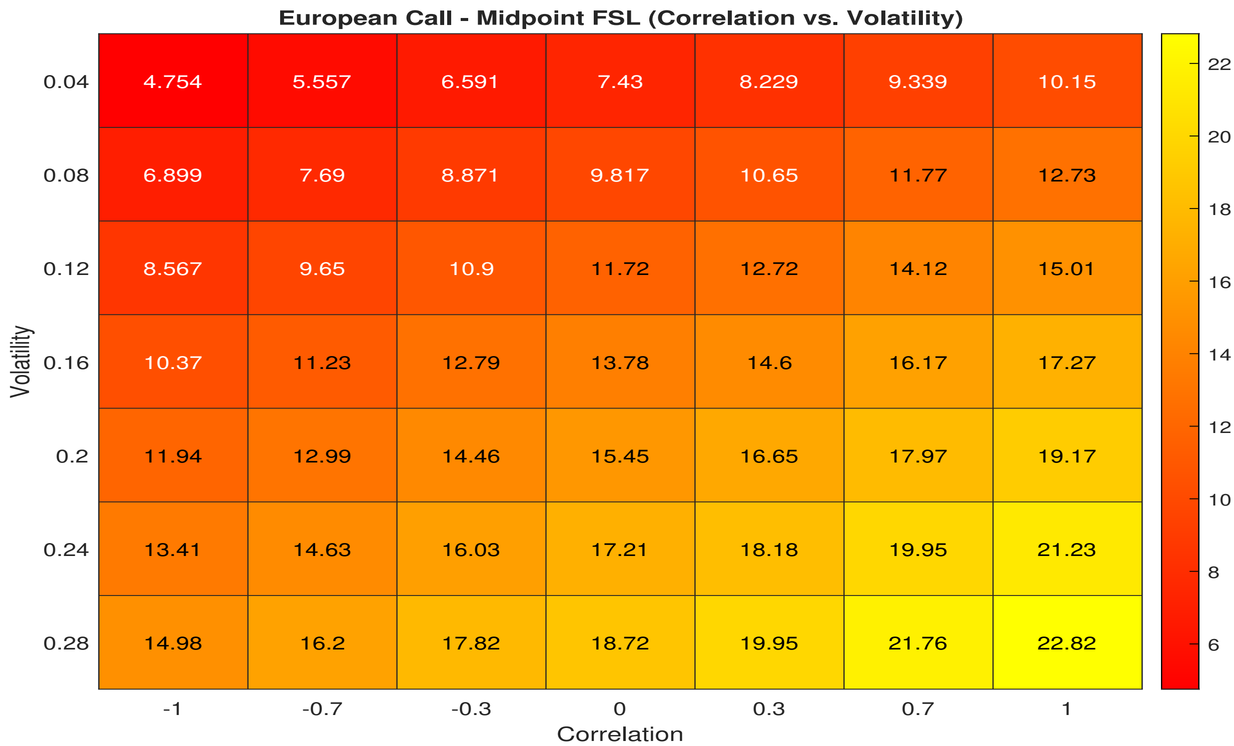

Figure 8: The figure shows a heat-map of European call option prices with SRKL Midpoint schemes as a function of initial volatility (V_0) and correlation (ρ).

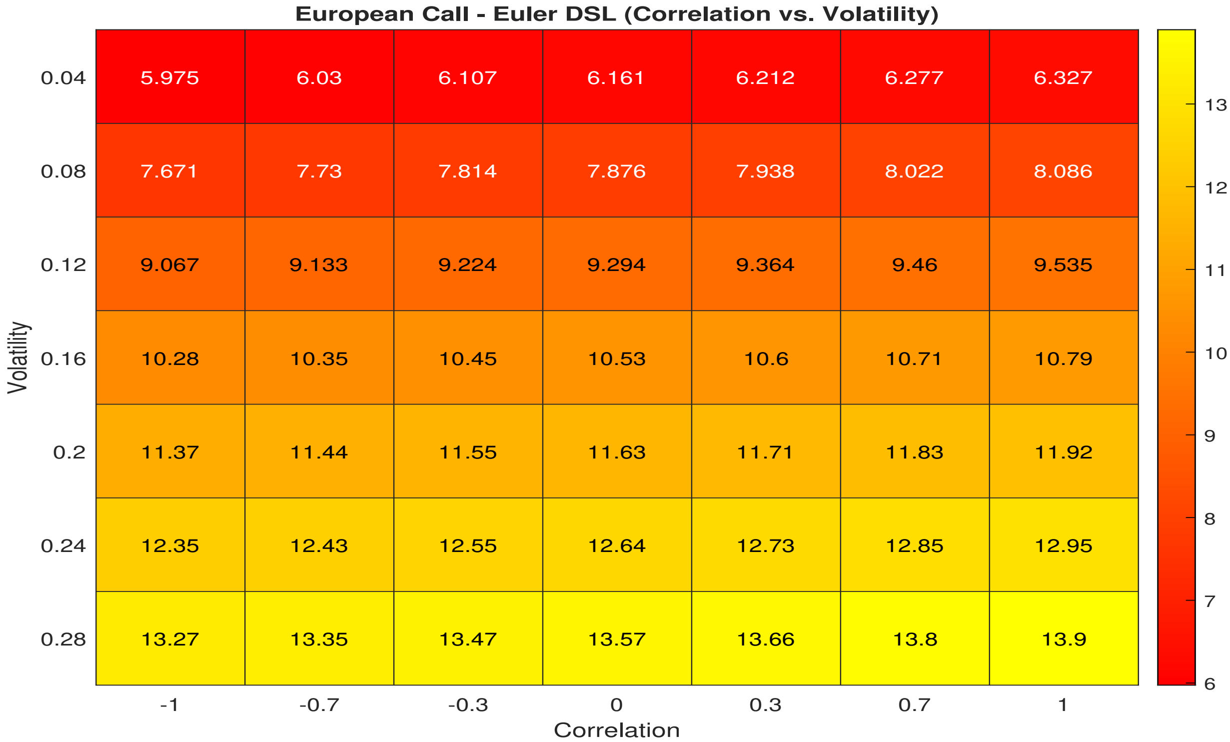

Figure 9: The figure shows a heat-map of European call option prices with SRKL Euler schemes and variable initial volatility (V_0) and correlation (ρ).

4.3 Sensitivity Analysis – Arithmetic Asian Call Option

4.3.1 Volatility Vs. Underlying

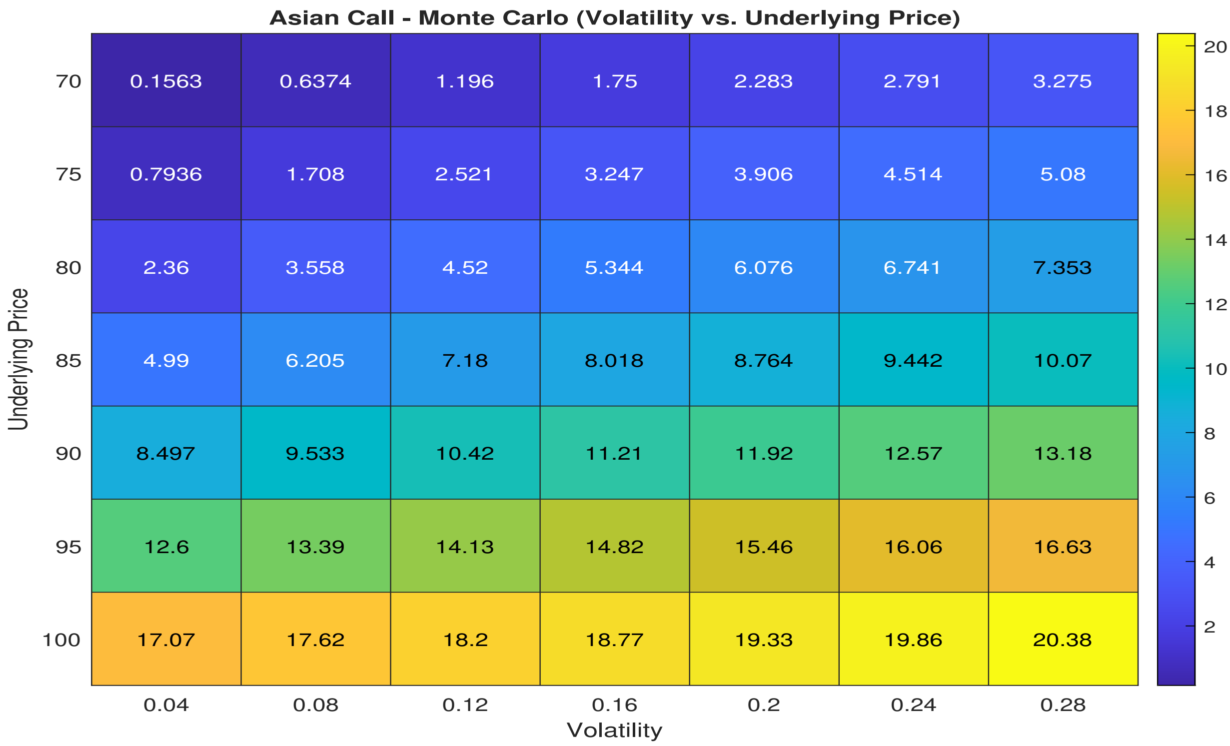

Figure 10: The figure shows a heat-map of arithmetic Asian call option prices with Monte Carlo methods and variable initial price (S_0) and volatility (V_0). The values for V_0 are shown on the x-axis, and the values for S_0 on the y-axis. Darker shades indicate lower prices and lighter shades represent higher prices.

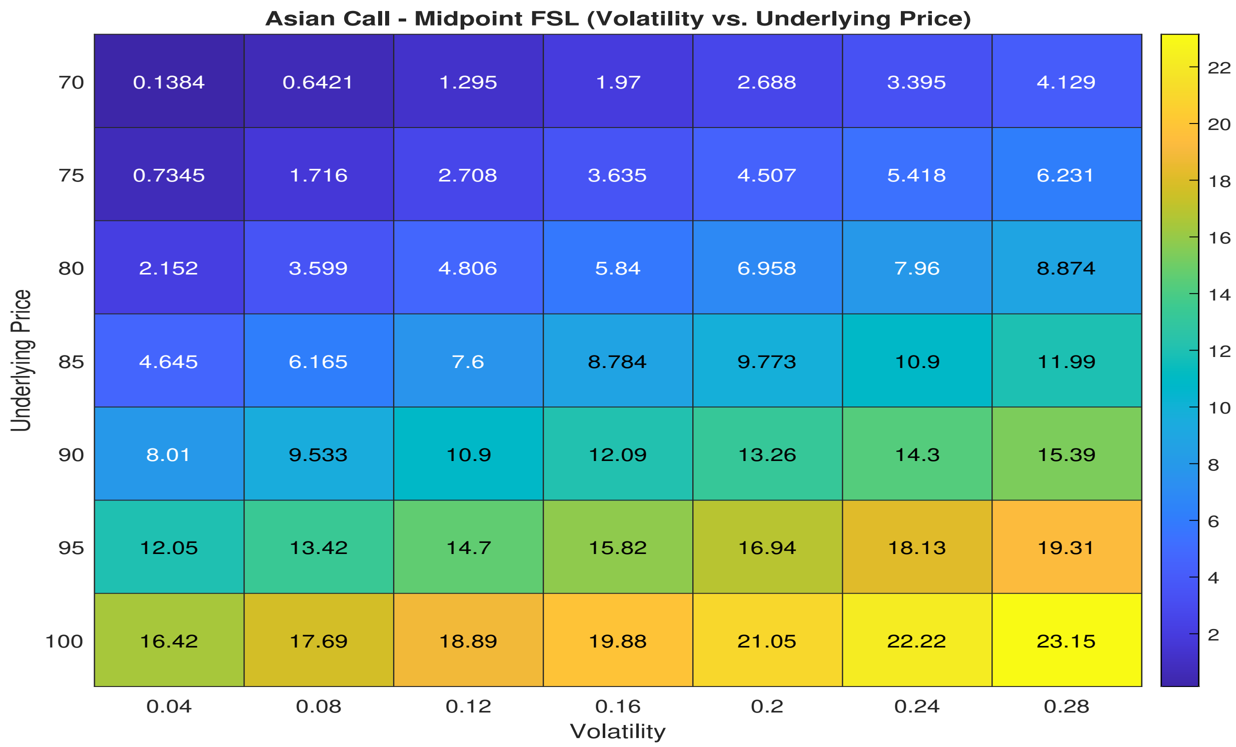

Figure 11: The figure shows a heat-map of arithmetic Asian call option prices with SRKL Midpoint schemes and variable initial price (S_0) and volatility (V_0).

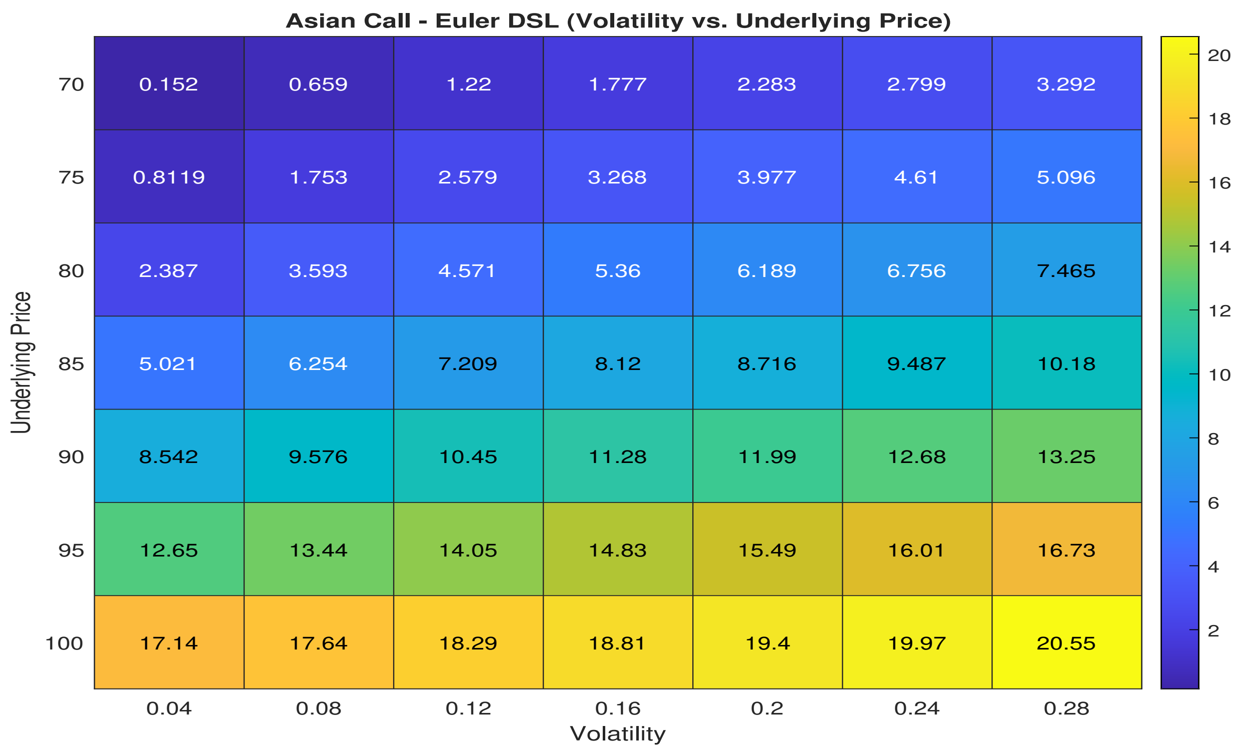

Figure 12: The figure shows a heat-map of arithmetic Asian call option prices with SRKL Euler schemes and variable initial price (S_0) and volatility (V_0).

4.3.2 Strike Vs. Underlying

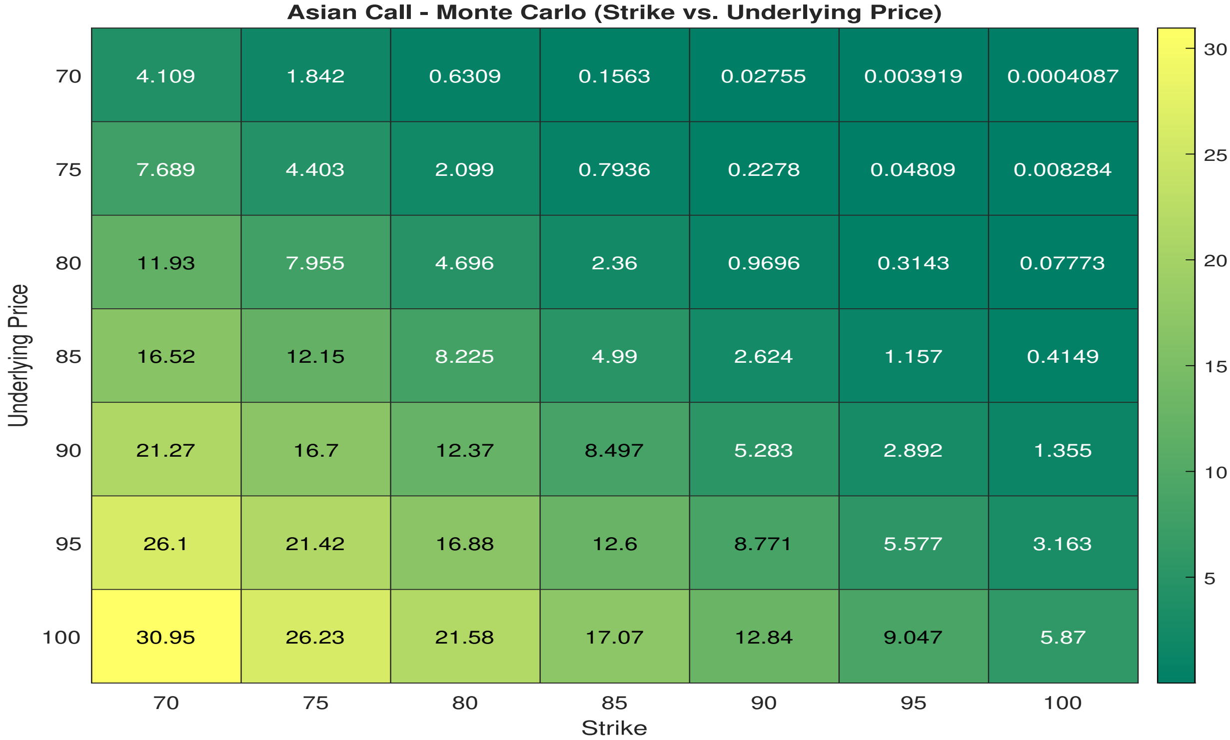

Figure 13: The figure shows a heat-map of arithmetic Asian call option prices with Monte Carlo method and variable initial price (S_0) and strike price (K). Prices on the diagonal from top-left to bottom-right represent ATM options, and prices over and under the diagonal represent OTM and ITM options, respectively. Darker shades indicate lower prices and lighter shades represent higher prices.

Figure 14: The figure shows a heat-map of arithmetic Asian call option prices with SRKL Midpoint schemes and variable initial price (S_0) and strike price (K).

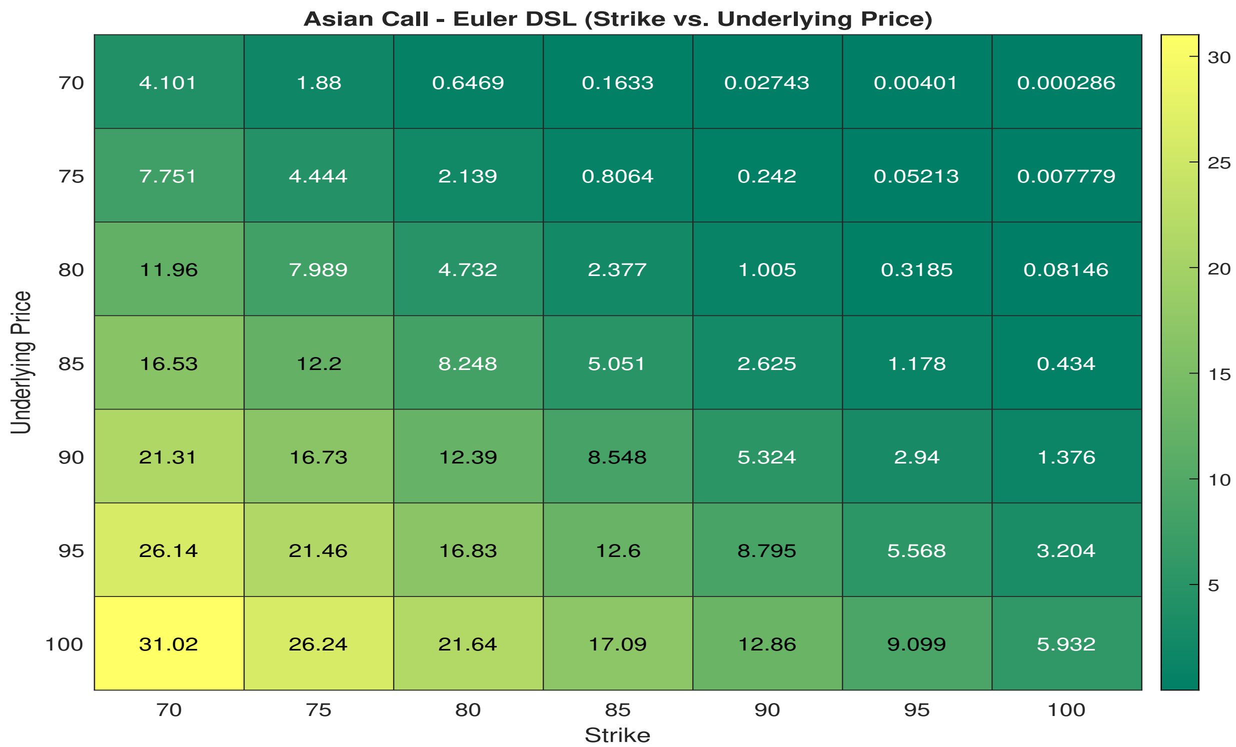

Figure 15: The figure shows a heat-map of arithmetic Asian call option prices with SRKL Euler schemes and variable initial price (S_0) and strike price (K).

4.3.3 Correlation Vs. Volatility

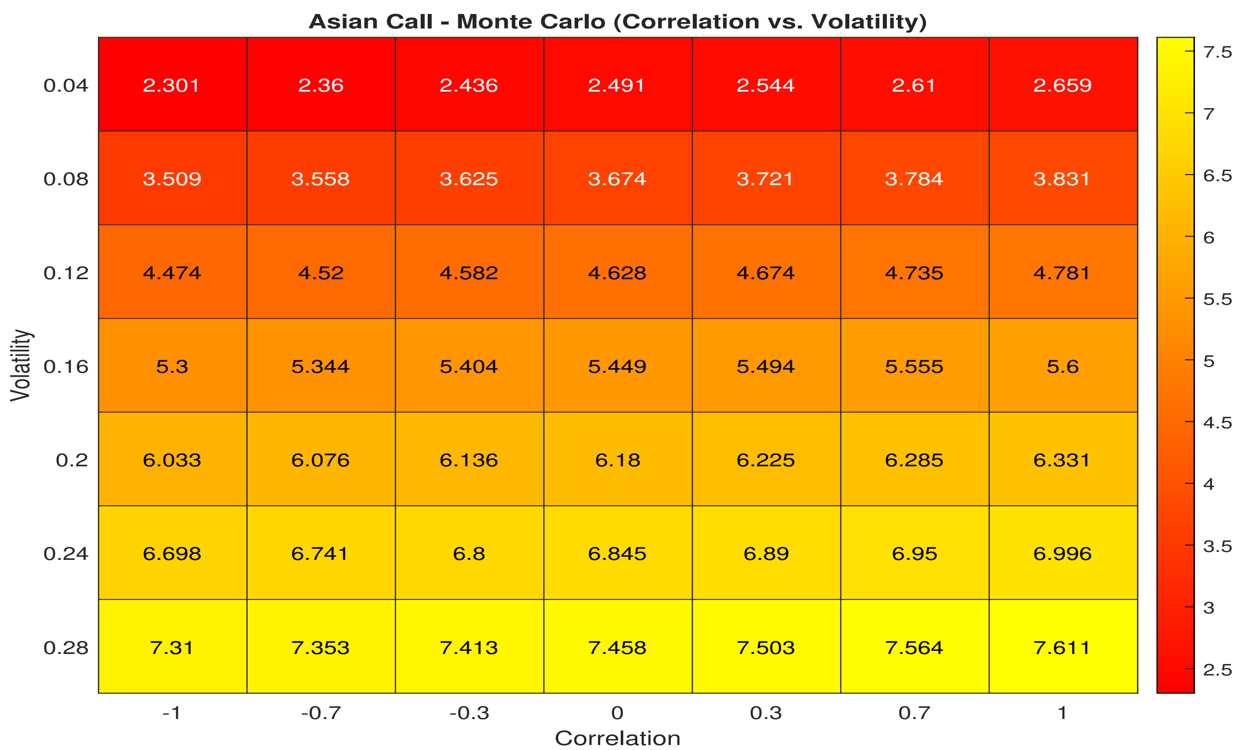

Figure 16: The figure shows a heat-map of arithmetic Asian call option prices with Monte Carlo method and variable initial volatility (V_0) and correlation (ρ). The values for ρ are shown on the x-axis, and the values for V_0 are shown on the y-axis. Darker shades indicate lower prices and lighter shades represent higher prices.

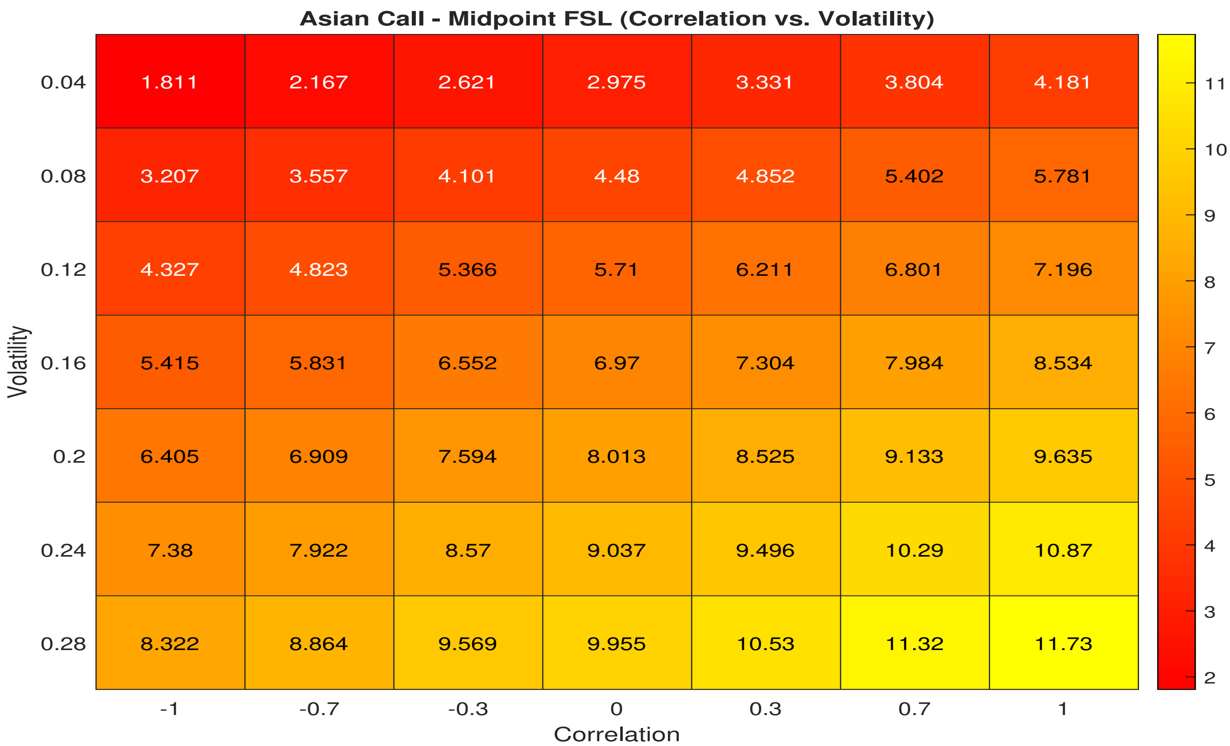

Figure 17: The figure shows a heat-map of arithmetic Asian call option prices with SRKL Midpoint schemes as a function of initial volatility (V_0) and correlation (ρ).

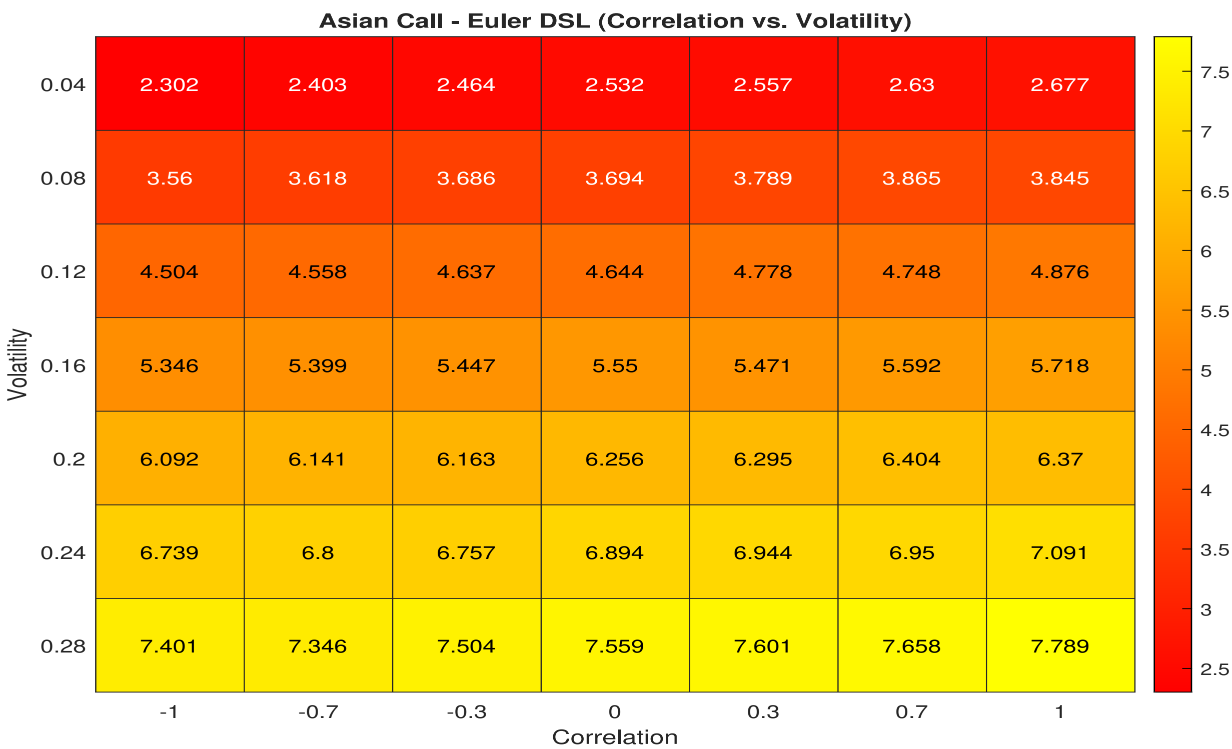

Figure 18: The figure shows a heat-map of arithmetic Asian call option prices with SRKL Euler schemes and variable initial volatility (V_0) and correlation (ρ).

4.4 Convergence Analysis

4.4.1 Price Convergence – Variable Step–Size

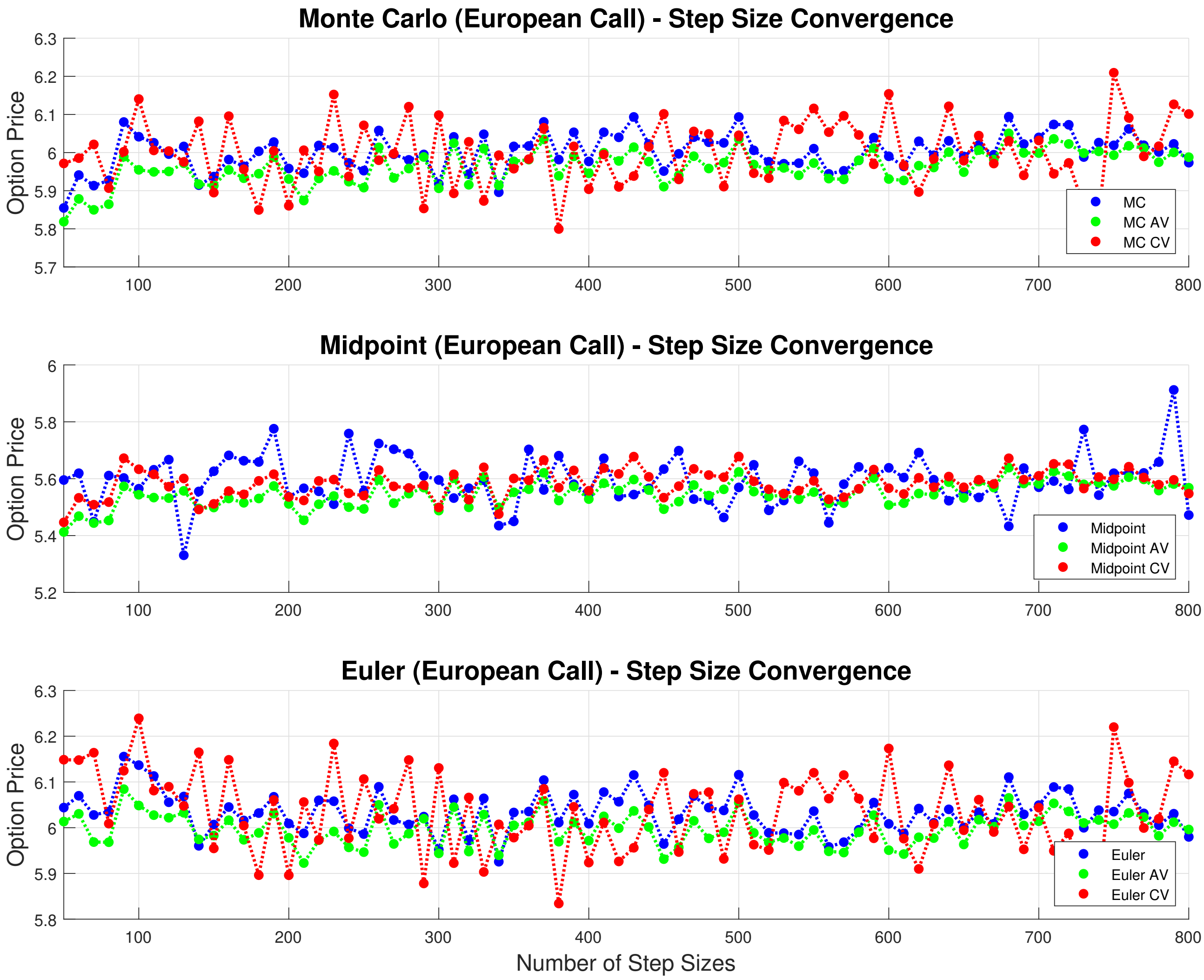

Figure 19: The figure illustrates the price convergence for a European call option with an actual price of $5.9882. Option prices are shown on the vertical axis and the number of step sizes on the horizontal axis. The Monte Carlo, Midpoint, and Euler methods with antithetic variate and control variate variance reduction techniques were used with variable step sizes. The number of simulations was 1000.

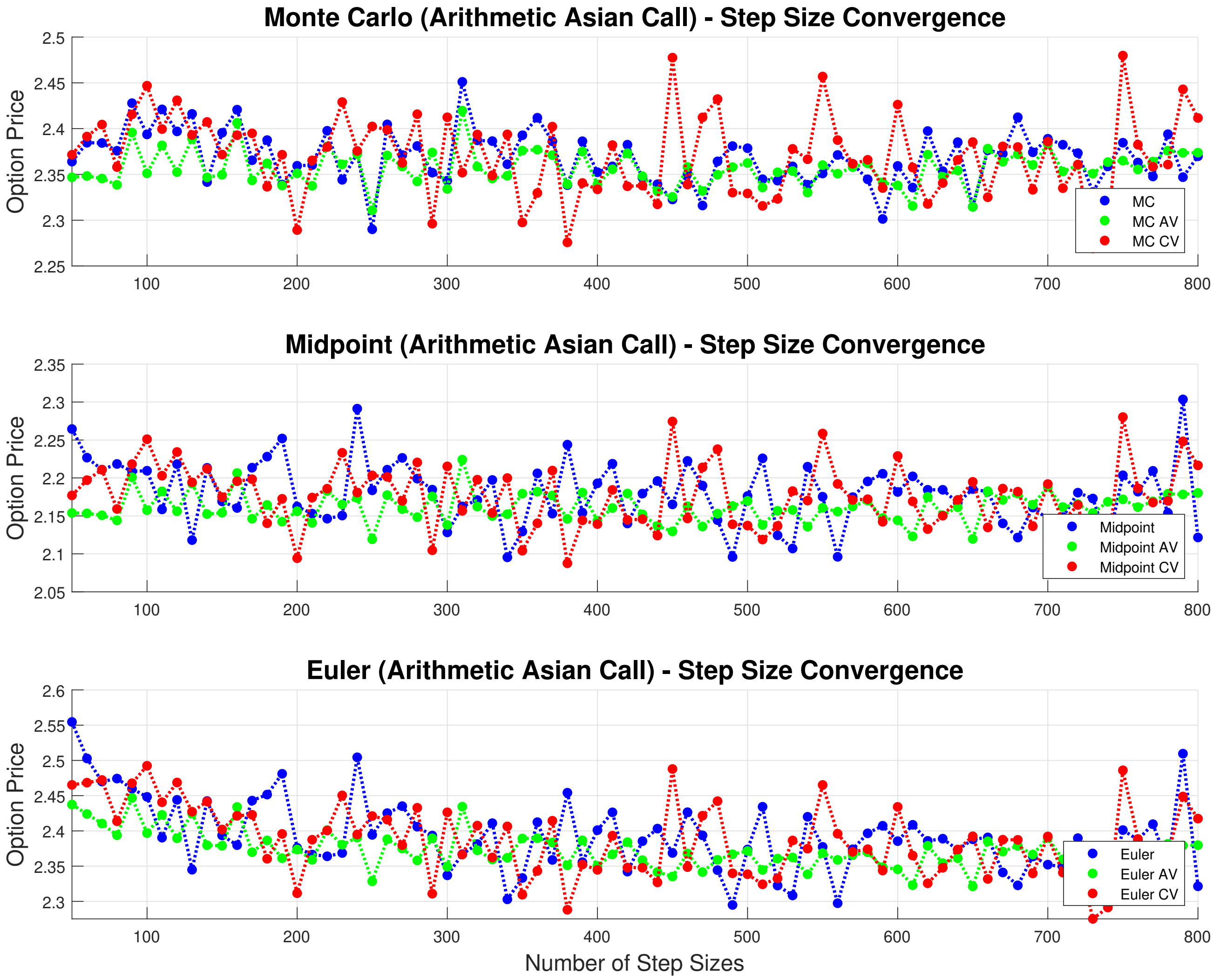

Figure 20: The figure illustrates the price convergence for an arithmetic Asian call option with an actual price of \$2.3546. Option prices are shown on the vertical axis and the number of step sizes on the horizontal axis. The Monte Carlo, Midpoint, and Euler methods with antithetic variate and control variate variance reduction techniques were used with variable step sizes. The number of simulations was 1000.

4.4.2 Price Convergence – Variable Number of Simulations

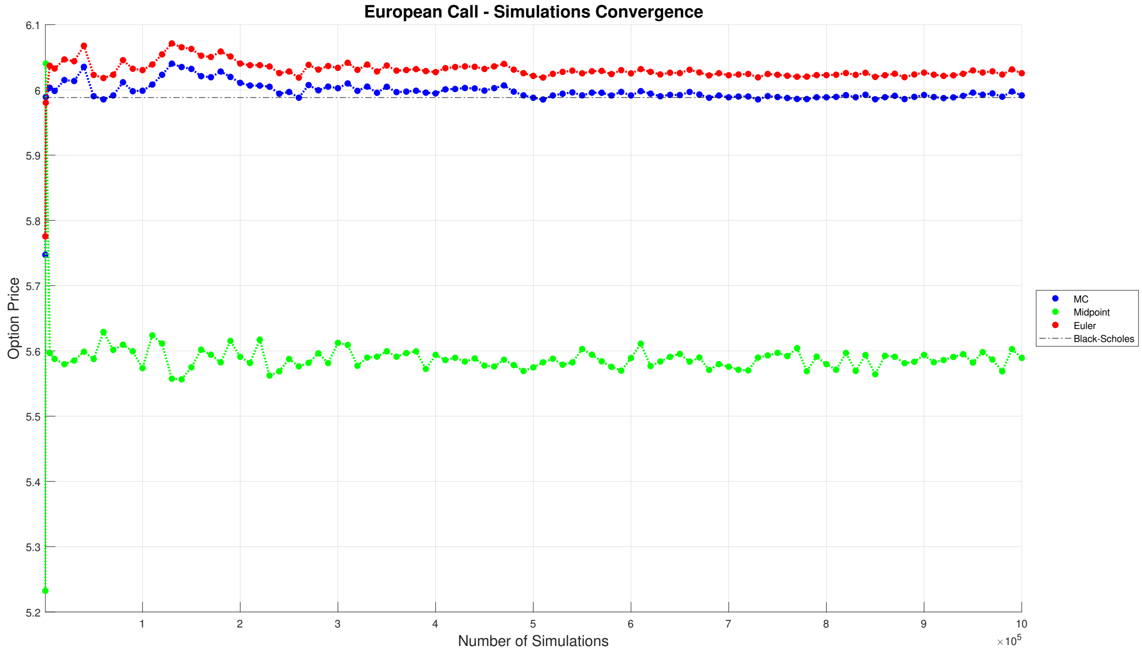

Figure 21: The figure illustrates the price convergence for a European call option with an actual price of \$5.9882. Option prices are shown on the vertical axis and the number of simulations on the horizontal axis. The calculations are based on different schemes, along without the application of variance reduction techniques.

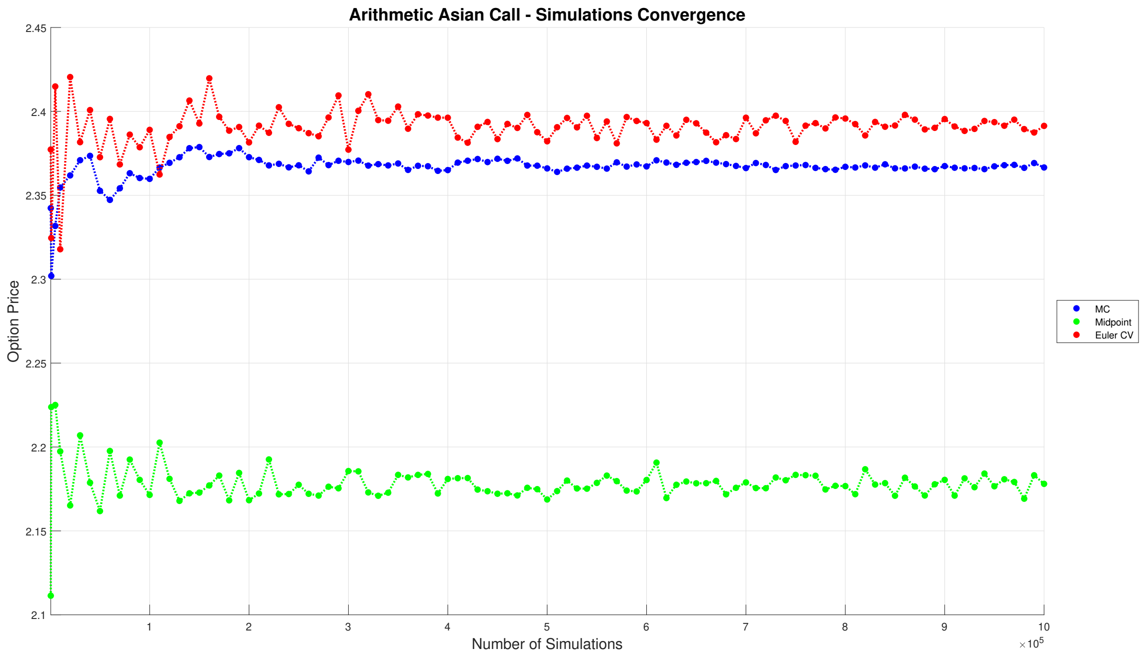

Figure 22: The figure illustrates the price convergence for an arithmetic Asian call option with an actual price of \$2.3546. Option prices are shown on the vertical axis and the number of simulations on the horizontal axis. The calculations are based on different schemes, along without the application of variance reduction techniques.

Chapter 5

Discussion

5.1 Accuracy and Efficiency Analysis

5.1.1 European Option

Table 1 shows that the differences between the Monte Carlo and Euler methods are relatively small regarding the computed call price. The Monte Carlo method yielded a call price of \$5.9984, followed closely by the Euler DSL scheme with \$5.9805, indicating a slight discrepancy between the two methods. On the contrary, for the Midpoint FSL scheme, the computed value was unexpectedly underpriced at \$5.3426. This surprising result may warrant further investigation to determine the cause of this deviation from the expected price.

Regarding the variance reduction methods, the antithetic variates technique consistently outperformed (in terms of price accuracy) the control variates technique across all employed methods, except for the Midpoint method. The call price values obtained using antithetic variates exhibited closer proximity to the benchmark value (\$5.9882, obtained from the Black–Scholes equation), suggesting that either the antithetic variate technique demonstrated higher effectiveness in reducing variance compared to the control variate method, or that the selection of the control variate was poorly made.

We used the absolute error to further study the accuracy of the methods. For the most part, we considered the absolute errors to be within an acceptable range of less than 1% for the Monte Carlo end Euler methods, suggesting reasonably accurate results. On the other hand, the implementation of control variate techniques exhibited lower performance in both the Monte Carlo and Euler methods, although the absolute errors remained acceptable at 1.78% and 2.28%, respectively. Furthermore, the Midpoint scheme performed significantly worse than the other methods, with absolute errors ranging from 6.77% to 10.78%, the former belonging to the control variate version and the latter to the standard method. These values indicate poor performance of the method in accurately valuing options under the Heston model, suggesting a possible need for adjustments when implementing the method.

In terms of computational time, the Midpoint scheme exhibits the highest time, with a value of 1674.30 milliseconds, while the Euler scheme took 209.20 milliseconds. The Monte Carlo method falls even lower, with a computational time of 180.99 milliseconds. In particular, the antithetic variate variance reduction technique shows the highest computational time across all methods, with a time of 363.06 milliseconds for Monte Carlo, 3279.80 for Midpoint, and 448.07 for Euler. These higher times are somewhat as expected since this variance reduction technique simulates two simultaneous price paths for the asset, which explains the roughly double-required time. On the contrary, the control variate reduction technique resulted in times that fall somewhere in between except for Euler, where the control variate computational time was lower than that of the standard method. These results are consistent with existent theory.

Efficiency, calculated as in Section 2.3.1, provides an overall measure of the computational performance. The Midpoint scheme stands out with lower efficiencies, characterized by values of magnitude 10^{-2}. In contrast, the other methods generally displayed efficiencies within the magnitude of 10^{-3}. In terms of efficiency, the standard methods and the antithetic variate approach performed similarly within the Monte Carlo and Midpoint methods individually. However, when considering the Euler scheme, the control variate technique exhibited better performance compared to both the standard methods and the antithetic variate approach.

The Midpoint scheme demonstrates the least preferable trade-off between computational time, accuracy, and efficiency. However, the fact that the control variate reduction technique performed the best within this scheme is a stimulating result that requires further observation. For the Euler scheme, the use of antithetic variate provided the most accurate result in terms of absolute error, outperforming the traditional Monte Carlo methods. These results indicate that, at least for European options, SRKL schemes can be subject to variance reduction techniques to improve the methods further.

5.1.2 Arithmetic Asian Option

Table 2 shows that the Midpoint scheme consistently exhibits the highest computational times among all methods for Asian options as well. These can be attributed to the nature of the Midpoint scheme, which involves more complex calculations than the other methods. In general, the Midpoint scheme requires more steps to compute the option price accurately, resulting in longer computational times.

The Euler scheme follows closely behind the Midpoint scheme regarding computational time. Although the Euler scheme is generally faster than the Midpoint scheme, it still takes longer than the standard Monte Carlo method. The increased computational time in the Euler scheme is due to the discretization approach, which introduces additional calculations and iterations.

On the other hand, the standard Monte Carlo method exhibits the lowest computational times. This is because it employs a more straightforward approach by directly simulating the asset price paths. Once again, it should be noted that the computational times for the antithetic variate technique are higher across all methods due to simulating two simultaneous price paths for the asset.

Moving on to the call prices, we observe that the arithmetic and geometric Asian call prices vary across the different methods. We also observe that geometric prices are higher than arithmetic prices. This result aligns with the underlying theory of Asian options. According to this theory, the geometric Asian option's payoff relies on the geometric mean of the underlying asset's prices over a specific period. Consequently, significant price fluctuations have a reduced impact on the average price, leading to a lower payout for the option holder.

We can also observe that the Midpoint schemes yields lower call prices than the other methods; a similar result as obtained for European options. This result can be attributed to the nature of the Midpoint scheme, which introduces discretization errors that affect the accuracy of the computed prices. The discrepancy between these values, suggests that the Midpoint scheme may also require adjustments to improve its performance in accurately valuing Asian options under the Heston model.

In terms of efficiency, in Table 3 we observe that the arithmetic Asian call options generally exhibit higher efficiency values compared to the geometric Asian call options. This can be explained by the fact that arithmetic options require more calculations, which involve averaging the underlying asset prices. For both options, the Midpoint scheme control variate technique exhibits better efficiency within the Midpoint scheme than the standard method. This indicates that the control variate technique helps improve the efficiency of the Midpoint scheme, despite its poorer performance in other metrics, such as computational time and accuracy.

Considering the absolute errors presented in Table 3, we find that the antithetic variate technique generally provides far better accuracy than the control variate technique for arithmetic and geometric prices. However, the control variate technique yields a lower absolute error for the arithmetic Midpoint method. The variations in absolute errors can be attributed to the specific characteristics of each method. The Midpoint scheme exhibits higher absolute errors, suggesting a poorer performance in accurately valuing Asian options, similarly as with European options. This may be due to the inherent limitations of the Midpoint scheme in capturing the complex dynamics of the underlying model, or an erroneous implementation of the method.

5.2 Sensitivity Analysis

5.2.1 European Option

Volatility Vs. Underlying

The heatmap figures illustrated in Chapter 4 are valuable tools for visually representing option prices and analysing their behaviour based on varying parameters. In particular, Figures 1–3 show the price change for different starting prices and volatilities.

Price patterns and trends become evident through observation of the colours in the heatmap. For instance, higher volatility values indicate an increase in the option's value for fixed values of the initial stock price. This remark is under basic options mechanics since the fluctuating nature of volatility can result in changing levels of uncertainty, which affects the pricing of options. Higher volatility leads to higher option prices due to the increased likelihood of more significant price swings and potential profit opportunities. Similarly, by fixing the volatility and varying the initial stock price, we see that the option price increases for higher values of the initial price. This result is because we have a fixed strike price, so taking the initial price lower or higher than the strike leaves us in OTM or ITM territory, thus resulting in lower or higher prices due to the intrinsic value of the contract. These results are also consistent with standard options theory.

In particular, we observe that for the Monte Carlo and Euler methods, the gradient (i.e. the rate of change) for the option prices is lower than that of the Midpoint scheme in both directions (i.e. change in price and change in volatility). This observation indicates that the Midpoint scheme is more sensitive to changes in the input parameters. Notably, the Midpoint scheme overprices the option for higher volatility values, leading to opposite results from previous discussions.

Another interesting observation is that option values seem consistent across all the models for a fixed volatility of 8%, even for varying initial prices. This probably indicates that a better calibration of the input parameters for every model could lead to more consistent results, so further studies should be performed to test this hypothesis.

Strike Vs. Underlying

Figures 4–6 present an alternative sensitivity analysis by varying the initial price and strike price parameters. Specifically, we observe that the prices along the diagonal from top-left to bottom-right represent ATM options. In contrast, prices over and under the diagonal represent OTM and ITM options, respectively. Similar to what we previously discussed, the intrinsic value of ATM, OTM and ITM plays an essential role in the valuation of the option.

Notably, the values for Monte Carlo and Euler methods are similar in every combination of strike and starting prices. The gradient for these methods is lower than that of the Midpoint scheme. We see that for ATM option prices, Midpoint underprices the option significantly. An interesting remark is that for deep–OTM and deep–ITM options, the Midpoint method approximates the Monte Carlo values, while for deep–ITM options, the Euler method underprices the option value. This observation indicates that the methods are sensitive to input parameter changes. However, the Midpoint method is more effective at pricing deep–OTM options than the Euler method.

Correlation Vs. Volatility

Figures 7–9 display heatmaps depicting option prices by considering variations in the correlation and initial volatility parameters. This analysis allows us to explore the relationship between these factors and their impact on option valuation.

The heatmaps show that both the Monte Carlo and Euler methods exhibit similar price patterns across different combinations of correlations and starting volatilities. However, these methods' gradients are lower than those of the Midpoint scheme. In particular, we observe in Figure 8 that the Midpoint method is susceptible to the volatility values and the correlation between the volatility and the stock price.

For example, for a volatility value of 20% and correlation values of ± 1 (i.e. perfect negative and perfect positive correlation), the Monte Carlo method prices the option by \$11.33 and \$11.88, respectively. On the other hand, the Euler method yields option prices of \$11.37 and \$11.92, which are relatively close in value. Finally, the Midpoint scheme yields prices of \$11.94 and \$19.17, which are significantly higher.

When the absolute values of the correlation are lower, the prices exhibit a higher level of stability. However, their accuracy remains suboptimal as they continue to underperform. These results indicate that the correlation between the price and volatility in the Heston model can cause significant impacts when utilizing the Midpoint scheme and thus make the method unsuitable for this model.

5.2.2 Arithmetic Asian Option

Volatility Vs. Underlying

The analysis of volatility and underlying parameters for Asian options follows a similar pattern to European options. Figures 10–12 show that higher volatility values still lead to increased option prices. This is because the fluctuating nature of volatility introduces more uncertainty into the market. This increased uncertainty provides greater potential for price swings and profit opportunities, resulting in higher option prices.

By fixing the volatility and varying the initial stock price, we observe that the option price increases for higher values of the initial price. This is because a higher initial price brings the option closer to being ITM, which increases its intrinsic value. Conversely, a lower initial price places the option further OTM, resulting in a lower intrinsic value and hence a lower option price. These observations align with the basic principles of options pricing.

Comparing the different pricing methods, we find that the gradient of option prices is lower for the Monte Carlo and Euler methods than for the Midpoint scheme. However, the price gradients for the Asian contracts are significantly larger than their European counterpart. This suggests that the Midpoint scheme is more sensitive to changes in the input parameters for Asian options as well. However, the Midpoint scheme tends to overprice the option for higher volatility values, similar to the behaviour observed in European options.

Strike Vs. Underlying

When examining the sensitivity of Asian option prices to variations in the strike and underlying prices, similar patterns emerge as observed in European options, as depicted in Figures 13–15.

For Asian options, the values produced by the Monte Carlo and Euler methods are similar across different strike and underlying price combinations. The gradient of these methods remains lower compared to the Midpoint scheme. The Midpoint scheme significantly underprices ATM options.

Moreover, the Midpoint method approximates the Monte Carlo values for deep OTM and deep ITM options, while the Euler method underprices deep ITM options. This indicates that both methods are sensitive to changes in input parameters. However, the Midpoint scheme performs better in pricing deep OTM options than the Euler method for Asian options.

Correlation Vs. Volatility

Analyzing correlation and volatility in Asian options reveals essential insights into option pricing. We can observe from Figures 16–18 similar price patterns among the Monte Carlo and Euler methods, exhibiting lower gradients than the Midpoint scheme. However, it is crucial to consider the impact of correlation and volatility on option valuation.

A high correlation between price and volatility indicates a strong connection between the two factors. In such cases, increasing volatility introduces higher uncertainty into the market. This increased uncertainty provides greater potential for the underlying asset to experience significant movements, which can lead to profitable outcomes for OTM options. As a result, option prices tend to be higher in scenarios of high correlation, reflecting the increased profit potential due to the elevated uncertainty.

Conversely, a low correlation implies a weaker connection between volatility and the underlying asset. In this situation, whether the underlying asset will exhibit sufficient movement to bring OTM options into potential profit territory becomes uncertain. Consequently, option valuation is lower when the correlation is low, indicating the reduced likelihood of achieving profits due to the weakened correlation between volatility and the underlying asset.

It is important to note that the Midpoint scheme tends to underperform in accurately pricing Asian options because of its susceptibility to volatility values and the correlation between volatility and the stock price. Even with lower absolute correlation values that tend to stabilize prices, the Midpoint scheme's accuracy remains suboptimal, as with European options. Thus, the Midpoint scheme's limitations in capturing the relationship between volatility and the underlying asset make it unsuitable for accurately valuing Asian options in the Heston model.

5.3 Convergence Analysis

Figure 19 provides insights into the convergence of European call options by examining step sizes. It allows us to observe the behaviour of different variance reduction techniques while keeping the number of simulations fixed.

The control variate technique exhibits erratic behaviour for the Monte Carlo and Euler methods, deviating more from the standard methods than the antithetic variate technique. These patterns indicate that the choice of control variate needs to be improved. On the other hand, when using the antithetic variate technique, the price variability for each method tends to stay closer to the standard approach. However, it generally remains slightly below it. This suggests that the antithetic variate underestimates the option price compared to the other approaches.

For the Midpoint scheme, the standard method shows a volatile trajectory, while the variance reduction techniques help dampen the amplitude of fluctuations. Like the previous case, the antithetic variate approach underestimates the option price.

Moving on to Figure 20, which focuses on arithmetic Asian call options, we observe similar patterns but with more erratic trajectories than Figure 19. Despite the increased volatility, the antithetic variate approach exhibits the least erratic behaviour. Additionally, the slight smile-like pattern in the Euler trajectories for lower values of the number of step sizes may be attributable to optimization-related issues.

Figure 21 demonstrates the convergence of the Monte Carlo method to the Black-Scholes benchmark value for European call options. In contrast, the Euler scheme tends to overestimate the option price, while the Midpoint scheme drastically underestimates it. In Figure 22, where an explicit benchmark value is unavailable, still suggests that the Monte Carlo method converges based on the observations. However, it is essential to note that the Monte Carlo method is the benchmark for Asian call options.

In particular, although the methods are not accurate, they converge to a specific value for European options after a relatively low number of simulations. For Asian options, the convergence is more unstable for the Euler and Midpoint schemes, indicating that these methods are less reliable in producing consistent results.

5.4 Thesis Limitations and Further Research

This section discusses the limitations encountered during the research and proposes areas for further investigation. The limitations include convergence patterns, accuracy analysis, calibration of schemes, and complex-valued prices arising from the Heston model. The proposed further research focuses on exploring different simulation sizes, optimization techniques, and additional research directions.

5.4.1 Limitations

Several limitations were encountered in the research that requires attention for future investigations. Firstly, the Heston model exhibited complex-valued prices due to the model's assumptions made about the volatility process. The mean-reverting behaviour and the presence of a square root term in the volatility process lead to negative volatility values, resulting in complex-valued stock prices for some simulations.

Secondly, the calibration of schemes proved challenging, particularly in the Midpoint FSL scheme when dealing with complex numbers arising from the simulations of the Heston model. Implementing a "forced" code to remove the complex component produced stable results but introduced bias. Future research could focus on refining calibration techniques for more accurate and unbiased results.

Furthermore, variations in hardware and software configurations can yield varying results, including differences in efficiency, absolute errors, and option prices. While the impact of individual configurations is considered minor due to the use of large sample sizes for analysis, optimization techniques such as parallel computing can further enhance running time and mitigate the effects of individual configurations.

5.4.2 Further Research Directions

While combining the Heston model with SRKL schemes was one possible avenue for exploration, it is crucial to determine whether it was the optimal choice. Evaluating alternative pricing models other than the Heston model could provide insights into the suitability, and potential advantages of SRKL schemes regarding the accuracy and efficiency of options pricing.

Additionally, expanding the scope of the thesis to include other option types or financial products, such as swaps or futures, would allow for a broader analysis. This expansion would involve exploring different market factors and underlying assets with diverse dynamics, going beyond the current focus. Moreover, considering alternative models and variations, such as jump-diffusion or hybrid models, could provide a more sophisticated representation of volatility dynamics and contribute further to understanding SRKL schemes. Notably, there was a striking disparity in performance between the FSL and DSL schemes, highlighting the need for a more comprehensive investigation into the distinct dynamics of these models. Conducting a deeper study in this area has the potential to yield enhanced insights and a better understanding of the results obtained.

Further research could include an accuracy analysis that compares each value in the presented heatmaps with their corresponding benchmark values. This analysis would provide valuable insights into the impact of moneyness. Furthermore, it is important to consider alternative accuracy measurements beyond the absolute error. Treating price overestimation and underestimation as the same may not be appropriate, particularly in options trading, where these errors have different implications for profit and loss. Therefore, implementing other statistical measures, like mean squared error, can provide a more subtle insight.

Once we establish confidence in a pricing model's capabilities, the next step is evaluating portfolio strategies using financial products such as options for hedging or risk management. This evaluation should consider market factors such as volatility, liquidity, and transaction costs to provide a comprehensive analysis. Calibrating a selected model to market data is crucial for real-time option pricing. This process involves estimating valid parameter values, reverse-engineering the model, and utilizing available market data. A potential avenue for further research is automating the model fitting process using machine learning techniques.

Further investigation is warranted to assess whether MATLAB was the most suitable software for the research objective. It would be worth considering alternative programming languages or software packages that offer advantages regarding computational efficiency, or specific features relevant to option pricing and analysis.

5.5 Thesis Contributions

This thesis was a collaborative effort among the authors, with few individual contributions that influenced the making of the thesis. In particular, Kuiper, N. focused on the writing, adaptation, and implementation of the MATLAB scripts found in Appendix A at the end of this paper. On the other hand, Westberg, M. contributed to presenting and building the theoretical frameworks used throughout this study. In other respects, both authors shared foundational knowledge of statistics, financial mathematics, and programming necessary to generate and interpret the results of this research. Overall, both authors comprehend and agree upon the collection of ideas presented in the thesis.

To highlight where credits are due, proof of Theorem 3 was proposed by our supervisor Prof. Anatoliy Malyarenko. The MATLAB functions `MidpointFSLVectorized.m` and `EulerDSLVectorized.m` in Appendix A were adapted by the author of [5], Kristian Debrabant. And the derivation of the SRKL for the Heston model found in Appendix B was performed by the authors of this paper. Other contents are supported by the literature.

Chapter 6

Conclusion

In conclusion, this thesis focused on investigating the potential benefits of integrating SRKL numerical schemes into the Heston model's SDEs for the valuation of financial options. Examining sub-questions provided valuable insights into the numerical schemes and their applicability. The computational complexity of the Heston model's SDEs required the use of efficient numerical approximation techniques for obtaining solutions. Implementing these methods required careful consideration and exploring variance reduction techniques in SRKL schemes aimed to enhance computational efficiency.

Combining SRKL schemes with variance reduction techniques proved to be feasible and effective. However, despite the high accuracy and stability of the selected SRKL schemes in solving SDEs, our findings revealed discrepancies in option pricing compared to benchmark values. Remarkably, the Monte Carlo method outperformed the SRKL schemes regarding accuracy, efficiency, and consistency. Among the SRKL schemes, the Euler DSL scheme exhibited significant superiority over the Midpoint FSL scheme in all measurements, indicating the advantages of DSL schemes for option pricing problems.

The integration of SRKL Euler numerical methods exhibited promise in enhancing the price for simple and path-dependent options. Consequently, integrating SRKL numerical methods into option valuation provides notable advantages by addressing challenges posed by the Heston model's SDEs and enhancing pricing accuracy for a diverse range of options.

Bibliography

[1] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973. ISSN 0022-3808. doi: 10.1086/260062.

[2] John C Cox, Jonathan E. Ingersoll Jr, and Stephen A. Ross. A theory of the term structure of interest rates. *Econometrica: Journal of the Econometric Society*, 53(2):385–407, 1985. ISSN 1468-0262. doi: 10.2307/1911242.

[3] Ricardo Crisóstomo. An Analysis of the Heston Stochastic Volatility Model: Implementation and Calibration Using Matlab. *SSRN Electronic Journal*, 2014. ISSN 2172-7147. doi: 10.2139/ssrn.2527818.

[4] Michael Curran. Valuing Asian and portfolio options by conditioning on the geometric mean price. *Management Science*, 40(12):1705–1711, 1994. ISSN 00251909. doi: 10.1287/mnsc.40.12.1705.

[5] Kristian Debrabant, Anne Kværnø, and Nicky Cordua Mattsson. Matlab code: Runge–Kutta Lawson schemes for stochastic differential equations, 2020. doi:https://doi.org/10.5281/zenodo.4062482.

[6] Kristian Debrabant, Anne Kværnø, and Nicky Cordua Mattsson. Runge–Kutta Lawson schemes for stochastic differential equations. *BIT*, 61(2):381–409, 2021. ISSN 0006-3835. doi: 10.1007/s10543-020-00839-8.

[7] Kristian Debrabant, Anne Kværnø, and Nicky Cordua Mattsson. Lawson schemes for highly oscillatory stochastic differential equations and conservation of invariants. *BIT*, 62(4):1121–1147, 2022. ISSN 0006-3835. doi: 10.1007/s10543-021-00906-8.

[8] Jim Gatheral. *The Volatility Surface: A Practitioner's Guide*. John Wiley & Sons, 2006. ISBN: 978-0471792512. doi: 10.1002/9781119202073.

[9] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer Science+Business Media, 2003. ISBN: 978-0-387-00451-8.

[10] W. A. Heinrich and L. Arnold. *Stochastic differential equations: Theory and applications*. Wiley, New York, 1974. ISBN: 978-0471033592. doi: 10.1002/zamm.19770570413.

[11] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, 6(2):327–343, 1993. ISSN 0893-9454. doi: 10.1093/rfs/6.2.327.

[12] John C. Hull. *Options, Futures, and Other Derivatives*. Pearson, 9th edition, 2014. ISBN: 978-0133456318.

[13] K Itô. Stochastic integral. *Proceedings of the Imperial Academy*, 20(8):519–524, 1944. doi: 10.3792/pia/1195572786.

[14] Mark S. Joshi. *The concepts and practice of mathematical finance*. Mathematics, Finance and Risk. Cambridge University Press, Cambridge, 2nd edition, 2008. ISBN: 978-0521514088.

[15] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, New York, 1995. ISBN: 0-387-54062-8.

[16] D. Lamberton and B. Lapeyre. *Introduction to Stochastic Calculus Applied to Finance*. CRC Press, New York, 2nd edition, 1996. ISBN: 978-1584886266.

[17] J. Douglas Lawson. Generalized Runge–Kutta processes for stable systems with large lipschitz constants. *SIAM Journal on Numerical Analysis*, 4(3):372–380, 1967.

[18] Robert C. Merton. Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, 4(1):141–183, 1973. ISSN 00058556. doi: 10.2307/3003143.

[19] G. N. Milstein. *Numerical integration of stochastic differential equations*. Springer, Berlin, 1995. ISBN: 978-0-7923-3213-8. doi: 10.1007/978-94-015-8455-5.

[20] Mariko Ninomiya and Syoiti Ninomiya. A new higher-order weak approximation scheme for stochastic differential equations and the Runge–Kutta method. *Finance and Stochastics*, 13(3):415–443, may 2009. ISSN 1432-1122. doi: 10.1007/s00780-009-0101-4.

[21] Arlie O. Petters and Xiaoying Dong. *An Introduction to Mathematical Finance with Applications: Understanding and Building Financial Intuition*. Springer, 2016. ISBN: 978-1-4939-3781-3. doi: 10.1007/978-1-4939-3783-7.

[22] Eckhard Platen and Nicola Bruti-Liberati. *Numerical solution of stochastic differential equations in finance*. Springer-Verlag, Berlin, 2010. ISBN: 978-3-642-12057-2. doi: 10.1007/978-3-642-13694-8.

[23] Jan R. M. Röman. *Analytical Finance: Volume I*. Springer International Publishing, 2016. ISBN: 978-3-319-34026-5. doi: 10.1007/978-3-319-34027-2.

[24] Andreas Rößler. Rooted tree analysis for order conditions of stochastic Runge-–Kutta methods for the weak approximation of stochastic differential equations. *Stochastic Analysis and Applications*, 24(1):97–134, March 2006. ISSN 0736-2994. doi: 10.1080/07362990500397699.

[25] Andreas Rößler. Second order Runge–Kutta methods for Stratonovich stochastic differential equations. *BIT Numerical Mathematics*, 47(3):657–680, May 2007. ISSN 0006-3835. doi: 10.1007/s10543-007-0130-3.

[26] Andreas Rößler. Second order Runge–Kutta methods for Itô stochastic differential equations. *SIAM Journal on Numerical Analysis*, 47(3):1713–1738, January 2009. ISSN 1095-7170. doi: 10.1137/060673308.

[27] Fabrice D. Rouah and Steven L. Heston. *The Heston Model and its Extensions in Matlab and C#*. Wiley, 2013. ISBN: 9781118548257. doi: 10.1002/9781118656471.

[28] Bernt Øksendal. *Stochastic Differential Equations*. Springer-Verlag, Berlin, 2003. ISBN: 978-3-540-04758-2. doi: 10.1007/978-3-642-14394-6.

Appendices

A European and Asian Call Option Prices

This appendix features the MATLAB scripts used to simulate and price European and Asian options with Monte Carlo and Stochastic Runge–Kutta Lawson methods, with the application of variance reduction techniques.

A.1 Function: MidpointFSLVectorized.m

```
function [t,S,X] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,
    dW,g0Jac,gJac,Bexp)
% Midpoint FSL scheme of strong order 1 to approximate the
    solution of dX=[AX+g_0(X)]dt + [BX+g_1(X)]dW on the time
    interval tspan with step size h, Wiener increments dW of
    dimension [n-1,P] where n-1 is the number of time steps and
     P the number of paths to simulate, and initial value X0.
    g0Jac and g1Jac are handles to a function implementing the
    Jacobian matrix of g0 and g1, their output should be of
    dimension [d,d,P] where d is the dimension of X0 and P is
    the number of different paths as described above. The
    solution of the implicit equation is solved with a single
    iteration of Newton's method (cmp. K. Debrabant and A. Kv{\
    ae}rn{\o}, B-series analysis of stochastic Runge-Kutta
    methods that use an iterative scheme to compute their
    internal stage values, SIAM J. Numer.Anal. 47, no.1
    (2008/09), pp.181-203.

%number of stochastic integrals
M = length(g);
for m=1:M
    if norm(A*B{m}-B{m}*A)>1e-10
        error("A and B have to commute")
    end
end
% create temporal grid
```

```
t = tspan(1):h:tspan(2);
n = length(t);
%initialise solution
P = size(dW{1},2);
X = repmat(X0,[1,P]);
% adapted variable to track the paths of the asset
S = zeros(n-1,P);
S(1,:) = X0(1,:);
%prepare for vectorized solution
nw=length(X0);
zw=repmat((1:nw)',[nw,1]);
sw=kron((1:nw)',ones(nw,1));
Zws=kron((0:nw:(P-1)*nw)',ones(nw^2,1));
Sws=Zws+repmat(sw,[P,1]);
Zws=Zws+repmat(zw,[P,1]);
clear zw sw
EshpA=sparse(Zws,Sws,reshape(repmat(expm(A*h/2),1,1,P),nw^2*P
    ,1),nw*P,nw*P,nw^2*P);
EshpAinv=inv(EshpA);
for i = 2:n-1
    %calculate matrix exponentials
    Eshp = EshpA;
    Eshm=EshpAinv;
    for m=1:M
        [ExpBdWh,InvExpBdWh]=Bexp{m}(dW{m}(i-1,:)/2);
        Eshp = Eshp * sparse(Zws,Sws,reshape(ExpBdWh,nw^2*P,1)
            ,nw*P,nw*P,nw^2*P);
        Eshm = Eshm * sparse(Zws,Sws,reshape(InvExpBdWh,nw^2*P
            ,1),nw*P,nw*P,nw^2*P);
    end
    Eh = Eshp*Eshp;
    %Update V
    Zx=calculateZx(X);
    VmX = - Zx\f(X);
    X=X+reshape(VmX,[nw,P]);
    X = reshape(Eh*X(:),[nw,P]);
    X(2,i) = max(X(2,i),0);
    S(i,:) = X(1,:);
end
    function erg=f(Vnew)
        argument=reshape(Eshp*(X(:)+Vnew(:))/2,[nw,P]);
        temp=h*g0(argument);
        for mind=1:M
            temp=temp+g{mind}(argument).*dW{mind}(i-1,:);
```

```
        end
        erg=Vnew(:)-X(:) -Eshm * reshape(temp,nw*P,1);
    end
    function erg=calculateZx(XEvalPoint)
        EvalPoint=reshape(Eshp*XEvalPoint(:),[nw,P]);
        temp=h*feval(g0Jac,EvalPoint);
        for mind=1:M
            temp=temp+bsxfun(@times,feval(gJac{mind},EvalPoint),reshape(dW{mind}(i-1,:),1,1,P));
            %temp=temp+feval(gJac{mind},EvalPoint)*reshape(dW{mind}(i-1,:),1,1,P);
        end
        erg=speye(nw*P)-Eshm*sparse(Zws,Sws,reshape(temp,nw^2*P,1),nw*P,nw*P,nw^2*P)*Eshp/2;
    end
end
```

A.2 Function: EulerDSLVectorized.m

```
function [t,S,X] = EulerDSLVectorized(X0,A,g0,g,tspan,h,dW)
%Euler-Maruyame DSL scheme of strong order 0.5 and weak order 1 to approximate the solution of
% dX=[AX+g_0(X)]dt + sum_{m=1}^Mg{m}(X)dW{m} on the time interval tspan with step
%size h, Wiener increments dW of dimension [n-1,P] where n-1 is the number
%of time steps and P the number of paths to simulate, and initial value X0.

% Number of stochastic integrals
M = length(g);

% create temporal grid
t = tspan(1):h:tspan(2);
n = length(t);

% Initialize solution
X = repmat(X0, [1, size(dW{1}, 2)]);
S = zeros(n-1,size(dW{1}, 2));
S(1,:) = X0(1,:);

% Do time stepping
for i = 2:n
```

```
    % calculate exponential matrix for each time step
    Ep = expm(A * h);

    % Update X using the Euler-Maruyama DSL scheme
    X = X + h * g0(X);
    for m = 1:M
        X = X + g{m}(X) .* dW{m}(i - 1, :);
        X(2,:) = max(X(2,:),0);
    end

    % Apply the exponential matrix
    X = Ep * X;
    X(2,i) = max(X(2,i),0);
    S(i,:) = X(1,:);
end
end
```

A.3 European – Standard Monte Carlo

```
% File: European_Heston_MC.m
%
% Purpose: Standard Monte Carlo simulations for pricing
%          European Options under the Heston model
%
% Algorithm: Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
    European_Heston_MC(S0,r,V0,K,type,kappa,theta,sigma,rho,Nt,
    Nsim,h,R)
% set random number generator seed for reproducibility
rng('default');
tic
% generate correlated Brownian motions
% generate two matrices of standard normal numbers
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
% calculate first Brownian motion matrix
dW1{1} = sqrt(h)*Z1;
% calculate correlated Brownian motion
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% pre-allocate memory for price and variance paths
X1 = cell(2,1);
```

```
X1{1} = zeros(Nt,Nsim);
X1{2} = zeros(Nt,Nsim);
% initiate asset price and variance at time 0
X1{1}(1,:) = S0;
X1{2}(1,:) = V0;
% simulate asset paths
for i = 1:Nt-1
    % generate asset price paths
    X1{1}(i+1,:) = X1{1}(i,:).*exp((r - 0.5*X1{2}(i,:))*h +
       sqrt(X1{2}(i,:)).*dW1{1}(i,:));
    % generate asset volatility paths
    X1{2}(i+1,:) = X1{2}(i,:) + kappa*(theta - X1{2}(i,:))*h +
        sigma*sqrt(X1{2}(i,:)).*dW1{2}(i,:);
    X1{2}(i+1,:) = max(X1{2}(i+1,:), 0);
end
% define payoff function for option type
if strcmp(type, 'call')
    payoff = max(X1{1}(end,:) - K,0);
elseif strcmp(type,'put')
    payoff = max(K - X1{1}(end,:),0);
end
% calculate option price and time used
payoff_mean = mean(payoff);
option_price = R*payoff_mean;
std_deviation = std(payoff);
elapsed_time = toc;
end
```

A.4 European – Control Variate Monte Carlo

```
% File: European_Heston_MC_CV.m
%
% Purpose: Control Variate variance reduction Monte Carlo
%          simulations for pricing European Options under
%          the Heston model
%
% Algorithm: Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
   European_Heston_MC_CV(S0,r,V0,K,K_cv,type,kappa,theta,sigma
   ,rho,Nt,Nsim,h,R)
% set random number generator seed for reproducibility
rng('default');
```

```
tic
% generate correlated Brownian motions
% generate two matrices of standard normal numbers
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
% calculate first Brownian motion matrix
dW1{1} = sqrt(h)*Z1;
% calculate correlated Brownian motion
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% pre-allocate memory for price and variance paths
X1 = cell(2,1);
X1{1} = zeros(Nt,Nsim);
X1{2} = zeros(Nt,Nsim);
% initiate asset price and variance at time 0
X1{1}(1,:) = S0;
X1{2}(1,:) = V0;
% simulate asset paths
for i = 1:Nt-1
    % generate asset price paths
    X1{1}(i+1,:) = X1{1}(i,:).*exp((r - 0.5*X1{2}(i,:))*h +
        sqrt(X1{2}(i,:)).*dW1{1}(i,:));
    % generate asset volatility paths
    X1{2}(i+1,:) = X1{2}(i,:) + kappa*(theta - X1{2}(i,:))*h +
         sigma*sqrt(X1{2}(i,:)).*dW1{2}(i,:);
    X1{2}(i+1,:) = max(X1{2}(i+1,:), 0);
end
% define payoff function for option type
if strcmp(type, 'call')
    payoff = max(X1{1}(end,:) - K,0);
    payoff_CV = max(X1{1}(end,:) - K_cv, 0);
elseif strcmp(type,'put')
    payoff = max(K - X1{1}(end,:),0);
    payoff_CV = max(K_cv - X1{1}(end,:), 0);
end
% estimate the control variate coefficient
CV = payoff_CV;
v = var(payoff);
C = cov(payoff, CV);
b = C(1,2)/v;
% calculate option price and time used
adjusted_payoff = payoff - b*(CV - mean(CV));
option_price = R*mean(adjusted_payoff);
std_deviation = std(payoff);
```

```
elapsed_time = toc;
end
```

A.5 European – Antithetic Variate Monte Carlo

```
% File: European_Heston_MC_AV.m
%
% Purpose: Antithetic Variate variance reduction Monte Carlo
%          simulations for pricing European Options under the
%          Heston model
%
% Algorithm: Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
   European_Heston_MC_AV(S0,r,V0,K,type,kappa,theta,sigma,rho,
   Nt,Nsim,h,R)
% set random number generator seed for reproducibility
rng('default');
tic
% generate correlated Brownian motion
% generate two matrices of standard normal numbers
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
% calculate first Brownian motion matrix
dW1{1} = sqrt(h)*Z1;
% calculate correlated Brownian motion
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% generate Brownian motions for antithetics path
dW2 = cell(2,1);
dW2{1} = -dW1{1};
dW2{2} = rho*dW2{1} + sqrt(h)*sqrt(1 - rho^2)*(-Z2);
% pre-allocate memory for price and variance paths
X1 = cell(2,1);
X1{1} = zeros(Nt,Nsim);
X1{2} = zeros(Nt,Nsim)
% antithetics
X2 = cell(2,1);
X2{1} = zeros(Nt,Nsim);
X2{2} = zeros(Nt,Nsim);
% initiate asset price and variance at time 0
X1{1}(1,:) = S0;
X1{2}(1,:) = V0;
```

```
% antithetics
X2{1}(1,:) = S0;
X2{2}(1,:) = V0;
% simulate asset paths (and antithetic paths)
for i = 1:Nt-1
    % generate asset price paths
    X1{1}(i+1,:) = X1{1}(i,:).*exp((r - 0.5*X1{2}(i,:))*h + sqrt(X1{2}(i,:)).*dW1{1}(i,:));
    % generate asset volatility paths
    X1{2}(i+1,:) = X1{2}(i,:) + kappa*(theta - X1{2}(i,:))*h + sigma*sqrt(X1{2}(i,:)).*dW1{2}(i,:);
    % ensure volatility is non-negative
    X1{2}(i+1,:) = max(X1{2}(i+1,:), 0);
    % generate antithetic price paths
    X2{1}(i+1,:) = X2{1}(i,:).*exp((r - 0.5*X2{2}(i,:))*h + sqrt(X2{2}(i,:)).*dW2{1}(i,:));
    % generate antithetic volatility paths
    X2{2}(i+1,:) = X2{2}(i,:) + kappa*(theta - X2{2}(i,:))*h + sigma*sqrt(X2{2}(i,:)).*dW2{2}(i,:);
    % ensure volatility is non-negative
    X2{2}(i+1,:) = max(X2{2}(i+1,:), 0);
end
% define payoff function for option type
if strcmp(type, 'call')
    payoff = max(X1{1}(end,:) - K, 0);
    antithetic_payoff = max(X2{1}(end,:) - K, 0);
elseif strcmp(type,'put')
    payoff = max(K - X1{1}(end,:), 0);
    antithetic_payoff = max(K - X2{1}(end,:), 0);
end
% calculate option price
payoff_mean = mean(payoff);
price = R*payoff_mean;
payoff_std = std(payoff);
% calculate antithetic option price
antithetic_payoff_mean = mean(antithetic_payoff);
antithetic_price = R*antithetic_payoff_mean;
antithetic_payoff_std = std(antithetic_payoff);
% calculate option price average and time used
option_price = 0.5*(price + antithetic_price);
std_deviation = 0.5*sqrt(payoff_std^2+antithetic_payoff_std^2);
elapsed_time = toc;
end
```

A.6 European – Midpoint FSL

```
% File: European_Heston_Midpoint_FSL.m
%
% Purpose: Monte Carlo simulation of the Heston model by a
%          Midpoint Full Stochastic Lawson scheme to price
%          European Options
%
% Algorithm: Kristian Debrabant, Anne Kv{\ae}rn{\o}, Nicky
%            Gordua Matsson. Runge-Kutta Lawson schemes for
%            stochastic differential equations. BIT Numerical
%            Matematics 61 (2021), 381-409.
%
% Implementation: Kristian Debrabant, Anne Kv{\ae}rn{\o}{\o},
%                 Nicky Gordua Matsson.
% Matlab code: Runge-Kutta Lawson schemes for stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
   European_Heston_Midpoint_FSL(S0,r,V0,K,T,type,kappa,theta,
   sigma,rho,Nt,Nsim,h,R)
% set random number generator seed for reproducibility
rng('default');
tic
% prepare input parameters to call Matlab funxction
   MidpointFSLVectorized
tspan=[0,T];
X0=[S0;V0];
ExpMatrixB=cell(2);
ExpMatrixB{1}=@(p,dW) RotMatExpdW(p,dW);
ExpMatrixB{2}=@(p,dW) RotMatExpdW(p,dW);
% calculate g1 and g2 and Jacobians
g{1}=@(x)[sqrt(x(2,:)).*x(1,:);zeros(1,size(x,2))];
g{2}=@(x)[zeros(1,size(x,2));sigma*sqrt(x(2,:))];
gJac{1}=@(x) getgJac1(x);
gJac{2}=@(x) getgJac2(x);
% matrices A1,A2
B=cell(2);
Bexp=cell(2);
B{1}=zeros(2);
B{2}=zeros(2);
```

```
Bexp{1} = @(W) ExpMatrixB{1}(0,W);
Bexp{2} = @(W) ExpMatrixB{1}(0,W);
% generate Brownian motions
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW = cell(2,1);
dW{1} = sqrt(h)*Z1;
dW{2} = rho*dW{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% A0
A=[r,0;0,-kappa];
% g0 and Jacobian
g0=@(x) getg0(x);
g0Jac=@(x) getgJac0(x);
% generate asset price at maturity
[~,~,X] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW,g0Jac,
   gJac,Bexp);    % returns price at expiration, and price and
    volatility paths
X = real(X);
% define payoff function for option type
if strcmp(type,'call')
    payoff = max(X(1,:)-K,0);
elseif strcmp(type,'put')
    payoff = max(K - X(1,:),0);
end
% calculate the price of the option and time used
payoff_mean = mean(payoff);
option_price = R*payoff_mean;
std_deviation = std(payoff);
elapsed_time = toc;
%% Functions for calling Midpoint FSL
function [erg,inverg]=RotMatExpdW(lambda,dW)
%calculate Matrix exponentials expm([0 -lambda;lambda 0]*dW(i)
   ) and their inverses and save them in erg(:,:,i) and inverg
   (:,:,i), respectively.
    erg=zeros(2,2,length(dW));
    inverg=zeros(size(erg));
    temp1=cos(lambda*dW);
    temp2=sin(lambda*dW);
    erg(1,1,:)=temp1;
    erg(2,2,:)=temp1;
    erg(1,2,:)=-temp2;
    erg(2,1,:)=temp2;
    inverg(1,1,:)=temp1;
    inverg(2,2,:)=temp1;
```

```
    inverg(1,2,:)=temp2;
    inverg(2,1,:)=-temp2;
end
function result=getg0(x)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end
function result=getgJac0(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
end
function result=getgJac1(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(1,1,:)=sqrt(x(2,:));
    result(1,2,:)=x(1,:)./(2*sqrt(x(2,:)));
end
function result=getgJac2(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(2,2,:)=sigma./(2*sqrt(x(2,:)));
end
end
```

A.7 European – Control Variate Midpoint FSL

```
% File: European_Heston_Midpoint_FSL_CV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a
%          Full Stochastic Lawson scheme with Control Variate
%          variance reduction technique to price European
%          Options
%
% Implementation: Kristian Debrabant, Anne Kv{\ae}rn{\o}{\o},
%                 Nicky Gordua Matsson.
% Matlab code: Runge-Kutta Lawson schemes for stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
```

```
% Adapted by Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
   European_Heston_Midpoint_FSL_CV(S0,r,V0,K,K_cv,T,type,kappa
   ,theta,sigma,rho,Nt,Nsim,h,R)
% set random number generator seed for reproducibility
rng('default');
tic
% prepare input parameters to call Matlab funxction
   MidpointFSLVectorized
tspan=[0,T];
X0=[S0;V0];
ExpMatrixB=cell(2);
ExpMatrixB{1}=@(p,dW) RotMatExpdW(p,dW);
ExpMatrixB{2}=@(p,dW) RotMatExpdW(p,dW);
% calculate g1 and g2 and Jacobians
g{1}=@(x)[sqrt(x(2,:)).*x(1,:);zeros(1,size(x,2))];
g{2}=@(x)[zeros(1,size(x,2));sigma*sqrt(x(2,:))];
gJac{1}=@(x) getgJac1(x);
gJac{2}=@(x) getgJac2(x);
% matrices A1,A2
B=cell(2);
Bexp=cell(2);
B{1}=zeros(2);
B{2}=zeros(2);
Bexp{1} = @(W) ExpMatrixB{1}(0,W);
Bexp{2} = @(W) ExpMatrixB{1}(0,W);
% generate Brownian motions
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW = cell(2,1);
dW{1} = sqrt(h)*Z1;
dW{2} = rho*dW{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% A0
A=[r,0;0,-kappa];
% g0 and Jacobian
g0=@(x) getg0(x);
g0Jac=@(x) getgJac0(x);
% generate asset price at maturity
[~,~,X] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW,g0Jac,
   gJac,Bexp);    % returns price at expiration, and price and
    volatility paths
X = real(X);
% define payoff function for option type
```

```
if strcmp(type,'call')
    payoff = max(X(1,:) - K,0);
    payoff_CV = max(X(1,:) - K_cv,0);
elseif strcmp(type,'put')
    payoff = max(K - X(1,:),0);
    payoff_CV = max(K_cv - X(1,:),0);
end
% estimate the control variate coefficient
CV = payoff_CV;
v = var(payoff);
C = cov(payoff, CV);
b = C(1,2)/v;
% calcualte the option price and time taken
adjusted_payoff = payoff - b*(CV - mean(CV));
option_price = R*mean(adjusted_payoff);
std_deviation = std(payoff);
elapsed_time = toc;
%% Functions for calling Midpoint FSL
function [erg,inverg]=RotMatExpdW(lambda,dW)
%calculate Matrix exponentials expm([0 -lambda;lambda 0]*dW(i)
   ) and their inverses and save them in erg(:,:,i) and inverg
   (:,:,i), respectively.
    erg=zeros(2,2,length(dW));
    inverg=zeros(size(erg));
    temp1=cos(lambda*dW);
    temp2=sin(lambda*dW);
    erg(1,1,:)=temp1;
    erg(2,2,:)=temp1;
    erg(1,2,:)=-temp2;
    erg(2,1,:)=temp2;
    inverg(1,1,:)=temp1;
    inverg(2,2,:)=temp1;
    inverg(1,2,:)=temp2;
    inverg(2,1,:)=-temp2;
end
function result=getg0(x)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end
function result=getgJac0(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
```

```
end
function result=getgJac1(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(1,1,:)=sqrt(x(2,:));
    result(1,2,:)=x(1,:)./(2*sqrt(x(2,:)));
end
function result=getgJac2(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(2,2,:)=sigma./(2*sqrt(x(2,:)));
end
end
```

A.8 European – Midpoint Antithetic FSL

```
% File: European_Heston_Midpoint_FSL_AV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a
%          Full Stochastic Lawson scheme with Antitethic
%          Variate variance reduction technique to price
%          European Options
%
% Implementation: Kristian Debrabant, Anne Kv{\ae}rn{\o}{\o},
%                 Nicky Gordua Matsson.
% Matlab code: Runge-Kutta Lawson schemes for stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] = European_Heston_Midpoint_FSL_AV(S0,r,V0,K,T,type,kappa,theta,sigma,rho,Nt,Nsim,h,R)
% set random number generator seed for reproducibility
rng('default');
tic
% prepare input parameters to call Matlab funxction MidpointFSLVectorized
tspan=[0,T];
X0=[S0;V0];
```

```
ExpMatrixB=cell(2);
ExpMatrixB{1}=@(p,dW) RotMatExpdW(p,dW);
ExpMatrixB{2}=@(p,dW) RotMatExpdW(p,dW);
% calculate g1 and g2
g{1}=@(x)[sqrt(x(2,:)).*x(1,:);zeros(1,size(x,2))];
g{2}=@(x)[zeros(1,size(x,2));sigma*sqrt(x(2,:))];
gJac{1}=@(x) getgJac1(x);
gJac{2}=@(x) getgJac2(x);
% A1 and A2
B=cell(2);
Bexp=cell(2);
B{1}=zeros(2);
B{2}=zeros(2);
Bexp{1} = @(W) ExpMatrixB{1}(0,W);
Bexp{2} = @(W) ExpMatrixB{1}(0,W);
% generate Brownian motions
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
dW1{1} = sqrt(h)*Z1;
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% generate Brownian motions for antithetic paths
dW2 = cell(2,1);
dW2{1} = -dW1{1};
dW2{2} = rho*dW2{1} + sqrt(h)*sqrt(1 - rho^2)*-Z2;
A=[r,0;0,-kappa];
g0=@(x) getg0(x);
g0Jac=@(x) getgJac0(x);
% generate asset price at maturity
[~,~,X1] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW1,g0Jac
   ,gJac,Bexp);    % returns price at expiration, and price
   and volatility paths
X1 = real(X1);
% generate antithetic paths price at maturity
[~,~,X2] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW2,g0Jac
   ,gJac,Bexp);    % returns antithetic price at expiration,
   and price and volatility paths
X2 = real(X2);
% calculate the price of the European option
if strcmp(type,'call')
    payoff = max(X1(1,:)-K,0);
    antithetic_payoff = max(X2(1,:)-K,0);
elseif strcmp(type,'put')
    payoff = max(K - X1(1,:),0);
```

```
    antithetic_payoff = max(K - X2(1,:),0);
end
payoff_mean = mean(payoff);
payoff_std = std(payoff);
antithetic_payoff_mean = mean(antithetic_payoff);
antithetic_payoff_std = std(antithetic_payoff);
option_price = R*0.5*(payoff_mean + antithetic_payoff_mean);
std_deviation = 0.5*sqrt(payoff_std^2+antithetic_payoff_std^2)
   ;
elapsed_time = toc;
%% Functions for calling Midpoint FSL
function [erg,inverg]=RotMatExpdW(lambda,dW)
%calculate Matrix exponentials expm([0 -lambda;lambda 0]*dW(i)
   ) and their inverses and save them in erg(:,:,i) and inverg
   (:,:,i), respectively.
    erg=zeros(2,2,length(dW));
    inverg=zeros(size(erg));
    temp1=cos(lambda*dW);
    temp2=sin(lambda*dW);
    erg(1,1,:)=temp1;
    erg(2,2,:)=temp1;
    erg(1,2,:)=-temp2;
    erg(2,1,:)=temp2;
    inverg(1,1,:)=temp1;
    inverg(2,2,:)=temp1;
    inverg(1,2,:)=temp2;
    inverg(2,1,:)=-temp2;
end
function result=getg0(x)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end
function result=getgJac0(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
end
function result=getgJac1(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(1,1,:)=sqrt(x(2,:));
    result(1,2,:)=x(1,:)./(2*sqrt(x(2,:)));
```

```
end
function result=getgJac2(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(2,2,:)=sigma./(2*sqrt(x(2,:)));
end
end
```

A.9 European – Euler DSL

```
% File: European_Heston_Euler_DSL.m
%
% Purpose: Monte Carlo simulation of the Heston model by a
   Euler-
%          Maruyama Drift Stochastic Lawson scheme to price
   European
%          Options
%
% Algorithm: Kristian Debrabant, Anne Kv{\ae}rn{\o}, Nicky
   Gordua Matsson.
%            Runge-Kutta Lawson schemes for stochastic
   differential
%            equations. BIT Numerical Matematics 61 (2021),
   381-409.
%
% Implementation: Kristian Debrabant, Anne Kv{\ae}rn{\o},
   Nicky Gordua Matsson.
%                 Matlab code: Runge-Kutta Lawson schemes for
   stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
   European_Heston_Euler_DSL(S0,r,V0,K,T,type,kappa,theta,
   sigma,rho,Nt,Nsim,R)
tic
X0 = [S0; V0];
A = [r, 0; 0, -kappa];
g0 = @(x) getg0(x,kappa,theta);
```

```
g = cell(2, 1);
g{1} = @(x) [sqrt(x(2, :)).*x(1, :); zeros(1, size(x, 2))];
g{2} = @(x) [zeros(1, size(x, 2)); sigma * sqrt(x(2, :))];

tspan = [0, T];
h = T / Nt;
rng('default');
Z1 = randn(Nt, Nsim);
Z2 = randn(Nt, Nsim);
dW = cell(2, 1);
dW{1} = sqrt(h) * Z1;
dW{2} = rho * dW{1} + sqrt(h) * sqrt(1 - rho^2) * Z2;

[~,~, X] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW);

% % calculate the price of the European option
if strcmp(type,'call')
    payoff = max(X(1,:)-K,0);
elseif strcmp(type,'put')
    payoff = max(K - X(1,:),0);
end
payoff_mean = mean(payoff);
option_price = R*payoff_mean;
std_deviation = std(payoff);
elapsed_time = toc;

function result=getg0(x,kappa,theta)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

end
```

A.10 European – Antithetic Variate Euler DSL

```
% File: European_Heston_Euler_DSL_AV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a Euler-
%          Maruyama Drift Stochastic Lawson scheme with Antithetic Variate
```

```
%            variance reduction technique to price European
   Options
%
% Algorithm: Kristian Debrabant, Anne Kv{\ae}rn{\o}, Nicky
   Gordua Matsson.
%              Runge-Kutta Lawson schemes for stochastic
   differential
%              equations. BIT Numerical Matematics 61 (2021),
   381-409.
%
% Implementation: Kristian Debrabant, Anne Kv{\ae}rn{\o},
   Nicky Gordua Matsson.
%                   Matlab code: Runge-Kutta Lawson schemes for
   stochastic
%                   differential equations (2020).
%                   https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
   European_Heston_Euler_DSL_AV(S0,r,V0,K,T,type,kappa,theta,
   sigma,rho,Nt,Nsim,R)
tic
X0 = [S0; V0];
A = [r, 0; 0, -kappa];
g0 = @(x) getg0(x,kappa,theta);

g = cell(2, 1);
g{1} = @(x) [sqrt(x(2, :)).*x(1, :); zeros(1, size(x, 2))];
g{2} = @(x) [zeros(1, size(x, 2)); sigma * sqrt(x(2, :))];

tspan = [0, T];
h = T / Nt;
rng('default');
Z1 = randn(Nt, Nsim);
Z2 = randn(Nt, Nsim);
dW1 = cell(2, 1);
dW1{1} = sqrt(h) * Z1;
dW1{2} = rho * dW1{1} + sqrt(h) * sqrt(1 - rho^2) * Z2;

% generate Brownian motions for antithetic paths
dW2 = cell(2,1);
dW2{1} = -dW1{1};
```

```
dW2{2} = rho*dW2{1} + sqrt(h)*sqrt(1 - rho^2)*-Z2;

[~,~, X1] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW1);
[~,~, X2] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW2);

% calculate the price of the European option
if strcmp(type,'call')
    payoff = max(X1(1,:)-K,0);
    antithetic_payoff = max(X2(1,:)-K,0);
elseif strcmp(type,'put')
    payoff = max(K - X1(1,:),0);
    antithetic_payoff = max(K - X2(1,:),0);
end
payoff_mean = mean(payoff);
payoff_std = std(payoff);

antithetic_payoff_mean = mean(antithetic_payoff);
antithetic_payoff_std = std(antithetic_payoff);

option_price = R*0.5*(payoff_mean + antithetic_payoff_mean);
std_deviation = 0.5*sqrt(payoff_std^2+antithetic_payoff_std^2)
    ;
elapsed_time = toc;

function result=getg0(x,kappa,theta)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

end
```

A.11 European – Control Variate Euler DSL

```
% File: European_Heston_Euler_DSL_CV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a
    Euler-
%          Maruyama Drift Stochastic Lawson scheme with
    Control Variate
%          variance reduction technique to price European
    Options
%
```

```
% Algorithm: Kristian Debrabant, Anne Kv{\ae}rn{\o}, Nicky
    Gordua Matsson.
%             Runge-Kutta Lawson schemes for stochastic
    differential
%             equations. BIT Numerical Matematics 61 (2021),
    381-409.
%
% Implementation: Kristian Debrabant, Anne Kv{\ae}rn{\o},
    Nicky Gordua Matsson.
%                 Matlab code: Runge-Kutta Lawson schemes for
    stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, option_price, std_deviation, elapsed_time] =
    European_Heston_Euler_DSL_CV(S0,r,V0,K,K_cv,T,type,kappa,
    theta,sigma,rho,Nt,Nsim,R)
tic
X0 = [S0; V0];
A = [r, 0; 0, -kappa];
g0 = @(x) getg0(x,kappa,theta);

g = cell(2, 1);
g{1} = @(x) [sqrt(x(2, :)).*x(1, :); zeros(1, size(x, 2))];
g{2} = @(x) [zeros(1, size(x, 2)); sigma * sqrt(x(2, :))];

tspan = [0, T];
h = T / Nt;
Nt = round(Nt);
Nsim = round(Nsim);
rng('default');

Z1 = randn(Nt, Nsim);
Z2 = randn(Nt, Nsim);
dW = cell(2);
dW{1} = sqrt(h)*Z1;
dW{2} = rho * dW{1} + sqrt(h) * sqrt(1 - rho^2) *Z2;

[~,~,X] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW);

% calculate the price of the European option
```

```
if strcmp(type,'call')
    payoff = max(X(1,:) - K,0);
    payoff_CV = max(X(1,:) - K_cv,0);
elseif strcmp(type,'put')
    payoff = max(K - X(1,:),0);
    payoff_CV = max(K_cv - X(1,:),0);
end
CV = payoff_CV;
% estimate the control variate coefficient
v = var(payoff);
C = cov(payoff, CV);
b = C(1,2)/v;
adjusted_payoff = payoff - b*(CV - mean(CV));
option_price = R*mean(adjusted_payoff);
std_deviation = std(payoff);
elapsed_time = toc;

function result=getg0(x,kappa,theta)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

end
```

A.12 Asian – Standard Monte Carlo

```
% File: Asian_Heston_MC.m
%
% Purpose: Standard Monte Carlo simulations for pricing
%          Asian Options under the Heston model
%
% Algorithm: Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price,geometric_price,
   arithmetic_std, geometric_std, elapsed_time] =
   Asian_Heston_MC(S0, r, V0, K, type, kappa, theta, sigma,
   rho, Nt, Nsim, T, R)
% set random number generator seed for reproducibility
rng('default');
tic
h = T/Nt;
% Generate correlated Brownian Motion
```

```
% generate two matrices of standard normal numbers
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
% calculate first Brownian motion matrix
dW1{1} = sqrt(h)*Z1;
% calculate correlated Brownian motion
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% pre-allocate memory for price and variance paths
X1 = cell(2,1);
X1{1} = zeros(Nt,Nsim);
X1{2} = zeros(Nt,Nsim);
% initiate asset price and variance at time 0
X1{1}(1,:) = S0;
X1{2}(1,:) = V0;
% simulate asset paths
for i = 1:Nt-1
    % generate asset price paths
    X1{1}(i+1,:) = X1{1}(i,:).*exp((r - 0.5*X1{2}(i,:))*h +
       sqrt(X1{2}(i,:)).*dW1{1}(i,:));
    % generate asset volatility paths
    X1{2}(i+1,:) = X1{2}(i,:) + kappa*(theta - X1{2}(i,:))*h +
        sigma*sqrt(X1{2}(i,:)).*dW1{2}(i,:);
    % ensure volatility is non-negative
    X1{2}(i+1,:) = max(X1{2}(i+1,:), 0);
end
% initiate asian averages matrices
arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);
% calculate paths averages
for i = 1:Nsim
    arithmetic_mean(i) = mean(X1{1}(2:end,i));
    geometric_mean(i) = geomean(X1{1}(2:end,i));
end
% calculate payoffs for each path
if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    geometric_payoff = max(geometric_mean - K,0);
else
    arithmetic_payoff = max(K - arithmetic_mean,0);
    geometric_payoff = max(K - geometric_mean,0);
end
% calculate option prices and price elapsed
arithmetic_price = R*mean(arithmetic_payoff);
```

```
geometric_price = R*mean(geometric_payoff);
arithmetic_std = std(arithmetic_payoff);
geometric_std = std(geometric_payoff);
elapsed_time = toc;
end
```

A.13 Asian – Control Variate Monte Carlo

```
% File: Asian_Heston_MC_CV.m
%
% Purpose: Monte Carlo simulations with control variate
   variance
%          reduction technique for pricing Asian Options under
    the
%          Heston model
%
% Algorithm: Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price,geometric_price,
   arithmetic_std, geometric_std, elapsed_time] =
   Asian_Heston_MC_CV(S0, r, V0, K, type, kappa, theta, sigma,
    rho, Nt, Nsim, T, R)
% set random number generator seed for reproducibility
%rng('default');
tic
h = T/Nt;
% generate correlated Brownian Motion
% generate two matrices of standard normal numbers
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
% calculate first Brownian motion matrix
dW1{1} = sqrt(h)*Z1;
% calculate correlated Brownian motion
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% pre-allocate memory for price and variance paths
X1 = cell(2,1);
X1{1} = zeros(Nt,Nsim);
X1{2} = zeros(Nt,Nsim);
% Initiate asset price and variance at time 0
X1{1}(1,:) = S0;
X1{2}(1,:) = V0;
% simulate asset paths
```

```
for i = 1:Nt-1
    % generate asset price paths
    X1{1}(i+1,:) = X1{1}(i,:).*exp((r - 0.5*X1{2}(i,:))*h +
       sqrt(X1{2}(i,:)).*dW1{1}(i,:));
    % generate asset volatility paths
    X1{2}(i+1,:) = X1{2}(i,:) + kappa*(theta - X1{2}(i,:))*h +
        sigma*sqrt(X1{2}(i,:)).*dW1{2}(i,:);
    % ensure volatility is non-negative
    X1{2}(i+1,:) = max(X1{2}(i+1,:), 0);
end
% allocate memory for averages
arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);
% calculate the averages for each path
for i = 1:Nsim
    arithmetic_mean(i) = mean(X1{1}(2:end,i));
    geometric_mean(i) = geomean(X1{1}(2:end,i));
end
% define payoff functions and
if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K ,0);
    geometric_payoff = max(geometric_mean - K,0);
else
    arithmetic_payoff = max(K - arithmetic_mean,0);
    geometric_payoff = max(K - geometric_mean,0);
end
% set the geometric payoff as control variate
CV = geometric_payoff;
payoff = arithmetic_payoff;
% estimate the control variate coefficient
v = var(payoff);
C = cov(CV, payoff);
b = C(1,2)/v;
% calculate option price
adjusted_payoff = payoff - b*(CV - mean(CV));
arithmetic_price = R*mean(adjusted_payoff);
geometric_price = R*mean(geometric_payoff);
arithmetic_std = std(arithmetic_payoff);
geometric_std = std(geometric_payoff);
elapsed_time = toc;
end
```

A.14 Asian – Antithetic Variate Monte Carlo

```
% File: Asian_Heston_MC_AV.m
%
% Purpose: Monte Carlo simulations with antithetic variate
   variance
%          reduction technique for pricing Asian Options under
    the Heston
%          model
%
% Algorithm: Nicolas Kuiper and Martin Westberg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price,geometric_price,
   arithmetic_std, geometric_std, elapsed_time] =
   Asian_Heston_MC_AV(S0, r, V0, K, type, kappa, theta, sigma,
    rho, Nt, Nsim, T, R)
% set random number generator seed for reproducibility
rng('default');
tic
h = T/Nt;
% generate correlated Brownian Motion
% generate two matrices of standard normal numbers
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
% calculate first Brownian motion matrix
dW1{1} = sqrt(h)*Z1;
% calculate correlated Brownian motion
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;
% generate Brownian motions for antithetics path
dW2 = cell(2,1);
dW2{1} = -dW1{1};
dW2{2} = rho*dW2{1} + sqrt(h)*sqrt(1 - rho^2)*(-Z2);
% pre-allocate memory for paths
X1 = cell(2,1);
X1{1} = zeros(Nt,Nsim);
X1{2} = zeros(Nt,Nsim);
% pre-allocate memory for price and variance paths
X2 = cell(2,1);
X2{1} = zeros(Nt,Nsim);          % antithetics price
X2{2} = zeros(Nt,Nsim);          % antithetics variance
% Initiate asset price and variance at time
X1{1}(1,:) = S0;
X1{2}(1,:) = V0;
```

```
X2{1}(1,:) = S0;          % antithetics initial price
X2{2}(1,:) = V0;          % antithetics initial variance
% simulate asset paths under geometric Brownian Motion
for i = 1:Nt-1
    % generate asset price paths
    X1{1}(i+1,:) = X1{1}(i,:).*exp((r - 0.5*X1{2}(i,:))*h + ...
        sqrt(X1{2}(i,:)).*dW1{1}(i,:));
    % generate asset volatility paths
    X1{2}(i+1,:) = X1{2}(i,:) + kappa*(theta - X1{2}(i,:))*h + ...
        sigma*sqrt(X1{2}(i,:)).*dW1{2}(i,:);
    % ensure volatility is non-negative
    X1{2}(i+1,:) = max(X1{2}(i+1,:), 0);
    % generate asset price paths for antithetics variate
    X2{1}(i+1,:) = X2{1}(i,:).*exp((r - 0.5*X2{2}(i,:))*h + ...
        sqrt(X2{2}(i,:)).*dW2{1}(i,:));
    % generate asset volatility paths for antithetics variate
    X2{2}(i+1,:) = X2{2}(i,:) + kappa*(theta - X2{2}(i,:))*h + ...
        sigma*sqrt(X2{2}(i,:)).*dW2{2}(i,:);
    % ensure volatility is non-negative
    X2{2}(i+1,:) = max(X2{2}(i+1,:), 0);
end
% pre-allocate memory for averages
arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);
antithetic_arithmetic_mean = zeros(1,Nsim);
antithetic_geometric_mean = zeros(1,Nsim);
% calculate the averages for each path
for i = 1:Nsim
    arithmetic_mean(i) = mean(X1{1}(2:end,i));
    geometric_mean(i) = geomean(X1{1}(2:end,i));
    antithetic_arithmetic_mean(i) = mean(X2{1}(2:end,i));
    antithetic_geometric_mean(i) = geomean(X2{1}(2:end,i));
end
% calculate payoffs for each path
if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    geometric_payoff = max(geometric_mean - K,0);
    antithetic_arithmetic_payoff = max(
       antithetic_arithmetic_mean - K,0);
```

```
    antithetic_geometric_payoff = max(antithetic_geometric_mean - K,0);
else
    arithmetic_payoff = max(K - arithmetic_mean,0);
    geometric_payoff = max(K - geometric_mean,0);
    antithetic_arithmetic_payoff = max(K - antithetic_arithmetic_mean,0);
    antithetic_geometric_payoff = max(K - antithetic_geometric_mean,0);
end
% calculate option prices
arithmetic_price = R*mean(arithmetic_payoff);
geometric_price = R*mean(geometric_payoff);
antithetic_arithmetic_price = R*mean(antithetic_arithmetic_payoff);
antithetic_geometric_price = R*mean(antithetic_geometric_payoff);
% average the option prices
arithmetic_price = 0.5*(arithmetic_price + antithetic_arithmetic_price);
geometric_price = 0.5*(geometric_price + antithetic_geometric_price);
arithmetic_std = 0.5*sqrt(std(arithmetic_payoff)^2 + std(antithetic_arithmetic_payoff)^2);
geometric_std = 0.5*sqrt(std(geometric_payoff)^2 + std(antithetic_geometric_payoff)^2);
elapsed_time = toc;
end
```

A.15 Asian – Midpoint FSL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% File: Asian_Heston_Midpoint_FSL.m
%
% Purpose: Monte Carlo simulation of the Heston model by a Midpoint Full
%          Stochastic Lawson scheme to price Asian Options
%
% Algorithm: Kristian Debrabant, Anne Kv rn , Nicky Gordua Matsson.
%            Runge-Kutta Lawson schemes for stochastic differential
```

```
%             equations. BIT Numerical Matematics 61 (2021), 381-409.
%
% Implementation: Kristian Debrabant, Anne Kv rn  , Nicky Gordua Matsson.
%                  Matlab code: Runge-Kutta Lawson schemes for stochastic
%                  differential equations (2020).
%                  https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [S, arithmetic_price, geometric_price, arithmetic_std,...
     geometric_std, elapsed_time] = Asian_Heston_Midpoint_FSL(S0,r,...
     V0,K,T,type,kappa,theta,sigma,rho,Nt,Nsim,R)
tic
% Prepare input parameters to call Matlab function MidpointFSLVectorized
tspan=[0,T];
X0=[S0;V0];
ExpMatrixB=cell(2);
ExpMatrixB{1}=@(p,dW) RotMatExpdW(p,dW);
ExpMatrixB{2}=@(p,dW) RotMatExpdW(p,dW);

% Calculate g1 and g2
g{1}=@(x)[sqrt(x(2,:)).*x(1,:);zeros(1,size(x,2))];
g{2}=@(x)[zeros(1,size(x,2));sigma*sqrt(x(2,:))];

% g{1} = @(x)[sqrt(real(x(2,:))).*x(1,:);sigma*rho*sqrt(real(x(2,:)))];
% g{2} = @(x)[zeros(1,size(x,2));sigma*sqrt((1-(rho^2))*real(x(2,:)))];

gJac{1}=@(x) getgJac1(x);
gJac{2}=@(x) getgJac2(x);
B=cell(2);
Bexp=cell(2);
B{1}=zeros(2);
B{2}=zeros(2);
Bexp{1} = @(W) ExpMatrixB{1}(0,W);
Bexp{2} = @(W) ExpMatrixB{1}(0,W);
```

```

h = T/Nt;
% Generate Brownian Motions
%rng('default');
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW = cell(2,1);
dW{1} = sqrt(h)*Z1;
dW{2} = rho*dW{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;

A=[r,0;0,-kappa];
g0=@(x) getg0(x);
g0Jac=@(x) getgJac0(x);

% Generate asset price at maturity
[~,S,~] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW,g0Jac,
   gJac,Bexp);    % returns price at expiration, and price and
    volatility paths

S = real(S);

% Calculate the price of the Asian option
arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);

for i = 1:Nsim
    arithmetic_mean(i) = mean(S(2:end,i));
    geometric_mean(i) = geomean(S(2:end,i));
end

if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    geometric_payoff = max(geometric_mean - K,0);
elseif strcmp(type,'put')
    arithmetic_payoff = max(K - arithmetic_mean,0);
    geometric_payoff = max(K - geometric_mean,0);
end

arithmetic_price = R*mean(arithmetic_payoff);
arithmetic_std = std(arithmetic_payoff);
geometric_price = R*mean(geometric_payoff);
geometric_std = std(geometric_payoff);
elapsed_time = toc;

```

```
%% Functions for calling Midpoint FSL
function [erg,inverg]=RotMatExpdW(lambda,dW)
%Calculate Matrix exponentials expm( [0 -lambda;lambda 0]*dW(i)) and their
%inverses and save them in erg(:,:,i) and inverg(:,:,i), respectively.
    erg=zeros(2,2,length(dW));
    inverg=zeros(size(erg));
    temp1=cos(lambda*dW);
    temp2=sin(lambda*dW);
    erg(1,1,:)=temp1;
    erg(2,2,:)=temp1;
    erg(1,2,:)=-temp2;
    erg(2,1,:)=temp2;
    inverg(1,1,:)=temp1;
    inverg(2,2,:)=temp1;
    inverg(1,2,:)=temp2;
    inverg(2,1,:)=-temp2;
end

function result=getg0(x)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

function result=getgJac0(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
end

function result=getgJac1(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(1,1,:)=sqrt(x(2,:));
    result(1,2,:)=x(1,:)./(2*sqrt(x(2,:)));
end

function result=getgJac2(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
```

```
    result(2,2,:)=sigma./(2*sqrt(x(2,:)));
end

end
```

A.16 Asian – Control Variate Midpoint FSL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% File: Asian_Heston_Midpoint_FSL_CV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a
   Midpoint Full
%          Stochastic Lawson scheme with control variance
   reduction
%          technique to price Asian Options
%
% Algorithm: Kristian Debrabant, Anne Kv rn  , Nicky Gordua
   Matsson.
%            Runge-Kutta Lawson schemes for stochastic
   differential
%            equations. BIT Numerical Matematics 61 (2021),
   381-409.
%
% Implementation: Kristian Debrabant, Anne Kv rn  , Nicky
   Gordua Matsson.
%                 Matlab code: Runge-Kutta Lawson schemes for
   stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price, geometric_price,
   arithmetic_std, geometric_std, elapsed_time] =
   Asian_Heston_Midpoint_FSL_CV(S0,r,V0,K,T,type,kappa,theta,
   sigma,rho,Nt,Nsim,R)
tic
% Prepare input parameters to call Matlab function
   MidpointFSLVectorized
tspan=[0,T];
X0=[S0;V0];
```

```
ExpMatrixB=cell(2);
ExpMatrixB{1}=@(p,dW) RotMatExpdW(p,dW);
ExpMatrixB{2}=@(p,dW) RotMatExpdW(p,dW);

% Calculate g1 and g2
g{1}=@(x)[sqrt(x(2,:)).*x(1,:);zeros(1,size(x,2))];
g{2}=@(x)[zeros(1,size(x,2));sigma*sqrt(x(2,:))];

% g{1} = @(x)[sqrt(real(x(2,:))).*x(1,:);sigma*rho*sqrt(real(x
    (2,:)))];
% g{2} = @(x)[zeros(1,size(x,2));sigma*sqrt((1-(rho^2))*real(x
    (2,:)))];

gJac{1}=@(x) getgJac1(x);
gJac{2}=@(x) getgJac2(x);
B=cell(2);
Bexp=cell(2);
B{1}=zeros(2);
B{2}=zeros(2);
Bexp{1} = @(W) ExpMatrixB{1}(0,W);
Bexp{2} = @(W) ExpMatrixB{1}(0,W);

h = T/Nt;
% Generate Brownian Motions
%rng('default');
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW = cell(2,1);
dW{1} = sqrt(h)*Z1;
dW{2} = rho*dW{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;

A=[r,0;0,-kappa];
g0=@(x) getg0(x);
g0Jac=@(x) getgJac0(x);

% Generate asset price at maturity
[~,S,~] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW,g0Jac,
    gJac,Bexp);    % returns price at expiration, and price and
     volatility paths
S = real(S);

% Calculate the price of the Asian option
arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);
```

```

for i = 1:Nsim
    arithmetic_mean(i) = mean(S(2:end,i));
    geometric_mean(i) = geomean(S(2:end,i));
end

if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    geometric_payoff = max(geometric_mean - K,0);
else
    arithmetic_payoff = max(K - arithmetic_mean,0);
    geometric_payoff = max(K - geometric_mean,0);
end
% set the geometric payoff as control variate
CV = geometric_payoff;
payoff = arithmetic_payoff;
% estimate the control variate coefficient
v = var(payoff);
C = cov(CV, payoff);
b = C(1,2)/v;
adjusted_payoff = payoff - b*(CV - mean(CV));
arithmetic_price = R*mean(adjusted_payoff);
geometric_price = R*mean(geometric_payoff);

arithmetic_std = std(adjusted_payoff);
geometric_std = std(geometric_payoff);
elapsed_time = toc;

%% Functions for calling Midpoint FSL
function [erg,inverg]=RotMatExpdW(lambda,dW)
%Calculate Matrix exponentials expm( [0 -lambda;lambda 0]*dW(i
   )) and their
%inverses and save them in erg(:,:,i) and inverg(:,:,i),
   respectively.
    erg=zeros(2,2,length(dW));
    inverg=zeros(size(erg));
    temp1=cos(lambda*dW);
    temp2=sin(lambda*dW);
    erg(1,1,:)=temp1;
    erg(2,2,:)=temp1;
    erg(1,2,:)=-temp2;
    erg(2,1,:)=temp2;
    inverg(1,1,:)=temp1;
    inverg(2,2,:)=temp1;
```

```
    inverg(1,2,:)=temp2;
    inverg(2,1,:)=-temp2;
end

function result=getg0(x)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

function result=getgJac0(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
end

function result=getgJac1(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(1,1,:)=sqrt(x(2,:));
    result(1,2,:)=x(1,:)./(2*sqrt(x(2,:)));
end

function result=getgJac2(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(2,2,:)=sigma./(2*sqrt(x(2,:)));
end

end
```

A.17 Asian – Antithetic Variate Midpoint FSL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% File: Asian_Heston_Midpoint_FSL_AV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a
   Midpoint Full
%          Stochastic Lawson scheme with antithetic variance
   reduction
```

```
%           technique to price Asian Options
%
% Algorithm: Kristian Debrabant, Anne Kv rn  , Nicky Gordua Matsson.
%             Runge-Kutta Lawson schemes for stochastic differential
%             equations. BIT Numerical Matematics 61 (2021), 381-409.
%
% Implementation: Kristian Debrabant, Anne Kv rn  , Nicky Gordua Matsson.
%                  Matlab code: Runge-Kutta Lawson schemes for stochastic
%                  differential equations (2020).
%                  https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price, geometric_price, arithmetic_std, geometric_std, elapsed_time] = Asian_Heston_Midpoint_FSL_AV(S0,r,V0,K,T,type,kappa,theta,sigma,rho,Nt,Nsim,R)
tic
% Prepare input parameters to call Matlab function MidpointFSLVectorized
tspan=[0,T];
X0=[S0;V0];
ExpMatrixB=cell(2);
ExpMatrixB{1}=@(p,dW) RotMatExpdW(p,dW);
ExpMatrixB{2}=@(p,dW) RotMatExpdW(p,dW);

% Calculate g1 and g2
g{1}=@(x)[sqrt(x(2,:)).*x(1,:);zeros(1,size(x,2))];
g{2}=@(x)[zeros(1,size(x,2));sigma*sqrt(x(2,:))];

% g{1} = @(x)[sqrt(real(x(2,:))).*x(1,:);sigma*rho*sqrt(real(x(2,:)))];
% g{2} = @(x)[zeros(1,size(x,2));sigma*sqrt((1-(rho^2))*real(x(2,:)))];

gJac{1}=@(x) getgJac1(x);
gJac{2}=@(x) getgJac2(x);
B=cell(2);
```

```
Bexp=cell(2);
B{1}=zeros(2);
B{2}=zeros(2);
Bexp{1} = @(W) ExpMatrixB{1}(0,W);
Bexp{2} = @(W) ExpMatrixB{1}(0,W);

h = T/Nt;
% Generate Brownian Motions
rng('default');
Z1 = randn(Nt,Nsim);
Z2 = randn(Nt,Nsim);
dW1 = cell(2,1);
dW1{1} = sqrt(h)*Z1;
dW1{2} = rho*dW1{1} + sqrt(h)*sqrt(1 - rho^2)*Z2;

% Generate Brownian Motions for antithetic paths
dW2 = cell(2,1);
dW2{1} = -dW1{1};
dW2{2} = rho*dW2{1} + sqrt(h)*sqrt(1 - rho^2)*-Z2;

A=[r,0;0,-kappa];
g0=@(x) getg0(x);
g0Jac=@(x) getgJac0(x);

% Generate asset price at maturity
[~,S1,~] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW1,g0Jac
    ,gJac,Bexp);    % returns price and volatility paths
[~,S2,~] = MidpointFSLVectorized(X0,A,g0,B,g,tspan,h,dW2,g0Jac
    ,gJac,Bexp);    % returns price and volatility paths
S1 = real(S1);
S2 = real(S2);

% Calculate average price throughout option life
arithmetic_mean = zeros(1,Nsim);
antithetic_arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);
antithetic_geometric_mean = zeros(1,Nsim);
for i = 1:Nsim
    arithmetic_mean(i) = mean(S1(2:end,i));
    antithetic_arithmetic_mean(i) = mean(S2(2:end,i));

    geometric_mean(i) = geomean(S1(2:end,i));
    antithetic_geometric_mean(i) = geomean(S2(2:end,i));
end
```

```

% calculate payoffs for each path
if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    antithetic_arithmetic_payoff = max(antithetic_arithmetic_mean - K,0);

    geometric_payoff = max(geometric_mean - K,0);
    antithetic_geometric_payoff = max(antithetic_geometric_mean - K,0);
else
    arithmetic_payoff = max(k - arithmetic_mean,0);
    antithetic_arithmetic_payoff = max(K - antithetic_arithmetic_mean,0);

    geometric_payoff = max(K - geometric_mean,0);
    antithetic_geometric_payoff = max(k - antithetic_geometric_mean,0);
end
% calculate payoffs mean and discount
arithmetic_price = R*mean(arithmetic_payoff);
geometric_price = R*mean(geometric_payoff);

anithetic_arithmetic_price = R*mean(antithetic_arithmetic_payoff);
anithetic_geometric_price = R*mean(antithetic_geometric_payoff);

arithmetic_price = 0.5*(arithmetic_price + anithetic_arithmetic_price);
geometric_price = 0.5*(geometric_price + anithetic_geometric_price);

arithmetic_std = 0.5*sqrt(std(arithmetic_payoff)^2+std(antithetic_arithmetic_payoff)^2);
geometric_std = 0.5*sqrt(std(geometric_payoff)^2+std(antithetic_geometric_payoff)^2);

elapsed_time = toc;

%% Functions for calling Midpoint FSL
function [erg,inverg]=RotMatExpdW(lambda,dW)
%Calculate Matrix exponentials expm( [0 -lambda;lambda 0]*dW(i)) and their
```

```
%inverses and save them in erg(:,:,i) and inverg(:,:,i),
   respectively.
    erg=zeros(2,2,length(dW));
    inverg=zeros(size(erg));
    temp1=cos(lambda*dW);
    temp2=sin(lambda*dW);
    erg(1,1,:)=temp1;
    erg(2,2,:)=temp1;
    erg(1,2,:)=-temp2;
    erg(2,1,:)=temp2;
    inverg(1,1,:)=temp1;
    inverg(2,2,:)=temp1;
    inverg(1,2,:)=temp2;
    inverg(2,1,:)=-temp2;
end

function result=getg0(x)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

function result=getgJac0(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
end

function result=getgJac1(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(1,1,:)=sqrt(x(2,:));
    result(1,2,:)=x(1,:)./(2*sqrt(x(2,:)));
end

function result=getgJac2(x)
    nw=size(x,1);
    P=size(x,2);
    result=zeros(nw,nw,P);
    result(2,2,:)=sigma./(2*sqrt(x(2,:)));
end

end
```

A.18 Asian – Euler DSL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% File: European_Heston_Euler_DSL.m
%
% Purpose: Monte Carlo simulation of the Heston model by a Euler-
%          Maruyama Drift Stochastic Lawson scheme to price Asian
%          Options
%
% Algorithm: Kristian Debrabant, Anne K v r n  , Nicky Gordua Matsson.
%              Runge-Kutta Lawson schemes for stochastic differential
%              equations. BIT Numerical Matematics 61 (2021), 381-409.
%
% Implementation: Kristian Debrabant, Anne K v r n  , Nicky Gordua Matsson.
%                   Matlab code: Runge-Kutta Lawson schemes for stochastic
%                   differential equations (2020).
%                   https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price, geometric_price, arithmetic_std, geometric_std, elapsed_time] = Asian_Heston_Euler_DSL(S0,r,V0,K,T,type,kappa,theta,sigma,rho,Nt,Nsim,R)
tic
X0 = [S0; V0];
A = [r, 0; 0, -kappa];
g0 = @(x) getg0(x,kappa,theta);

g = cell(2, 1);
g{1} = @(x) [sqrt(x(2, :)).*x(1, :); zeros(1, size(x, 2))];
g{2} = @(x) [zeros(1, size(x, 2)); sigma * sqrt(x(2, :))];
```

```
tspan = [0, T];
h = T / Nt;
%rng('default');
Z1 = randn(Nt, Nsim);
Z2 = randn(Nt, Nsim);
dW = cell(2, 1);
dW{1} = sqrt(h) * Z1;
dW{2} = rho * dW{1} + sqrt(h) * sqrt(1 - rho^2) * Z2;

[~,S, ~] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW);

% Calculate the price of the Asian option
arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);

for i = 1:Nsim
    arithmetic_mean(i) = mean(S(2:end,i));
    geometric_mean(i) = geomean(S(2:end,i));
end

if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    geometric_payoff = max(geometric_mean - K,0);
elseif strcmp(type,'put')
    arithmetic_payoff = max(K - arithmetic_mean,0);
    geometric_payoff = max(K - geometric_mean,0);
end

arithmetic_price = R*mean(arithmetic_payoff);
arithmetic_std = std(arithmetic_payoff);
geometric_price = R*mean(geometric_payoff);
geometric_std = std(geometric_payoff);
elapsed_time = toc;

function result=getg0(x,kappa,theta)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

end
```

A.19 Asian – Control Variate Euler DSL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% File: Asian_Heston_Euler_DSL_CV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a Euler-
%          Maruyama Drift Stochastic Lawson scheme with control
%          variate variance reduction technique to price Asian
%          Options
%
% Algorithm: Kristian Debrabant, Anne Kv rn , Nicky Gordua Matsson.
%            Runge-Kutta Lawson schemes for stochastic differential
%            equations. BIT Numerical Matematics 61 (2021), 381-409.
%
% Implementation: Kristian Debrabant, Anne Kv rn , Nicky Gordua Matsson.
%                 Matlab code: Runge-Kutta Lawson schemes for stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price, geometric_price, arithmetic_std, geometric_std, elapsed_time] = Asian_Heston_Euler_DSL_CV(S0,r,V0,K,T,type,kappa,theta,sigma,rho,Nt,Nsim,R)
tic
X0 = [S0; V0];
A = [r, 0; 0, -kappa];
g0 = @(x) getg0(x,kappa,theta);

g = cell(2, 1);
g{1} = @(x) [sqrt(x(2, :)).*x(1, :); zeros(1, size(x, 2))];
g{2} = @(x) [zeros(1, size(x, 2)); sigma * sqrt(x(2, :))];

tspan = [0, T];
```

```
h = T / Nt;
%rng('default');
Z1 = randn(Nt, Nsim);
Z2 = randn(Nt, Nsim);
dW = cell(2, 1);
dW{1} = sqrt(h) * Z1;
dW{2} = rho * dW{1} + sqrt(h) * sqrt(1 - rho^2) * Z2;

[~,S, ~] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW);

% Calculate the price of the Asian option
arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);

for i = 1:Nsim
    arithmetic_mean(i) = mean(S(2:end,i));
    geometric_mean(i) = geomean(S(2:end,i));
end

if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    geometric_payoff = max(geometric_mean - K,0);
else
    arithmetic_payoff = max(K - arithmetic_mean,0);
    geometric_payoff = max(K - geometric_mean,0);
end
% set the geometric payoff as control variate
CV = geometric_payoff;
payoff = arithmetic_payoff;
% estimate the control variate coefficient
v = var(payoff);
C = cov(CV, payoff);
b = C(1,2)/v;
adjusted_payoff = payoff - b*(CV - mean(CV));
arithmetic_price = R*mean(adjusted_payoff);
geometric_price = R*mean(geometric_payoff);

arithmetic_std = std(adjusted_payoff);
geometric_std = std(geometric_payoff);
elapsed_time = toc;

function result=getg0(x,kappa,theta)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
```

```
    result(2,:)=result(2,:)*kappa*theta;
end

end
```

A.20 Asian – Antithetic Variate Euler DSL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% File: European_Heston_Euler_DSL_AV.m
%
% Purpose: Monte Carlo simulation of the Heston model by a Euler-
%          Maruyama Drift Stochastic Lawson scheme with antithetic
%          variate variance reduction technique to price Asian
%          Options
%
% Algorithm: Kristian Debrabant, Anne K v rn  , Nicky Gordua Matsson.
%            Runge-Kutta Lawson schemes for stochastic differential
%            equations. BIT Numerical Matematics 61 (2021), 381-409.
%
% Implementation: Kristian Debrabant, Anne K v rn  , Nicky Gordua Matsson.
%                 Matlab code: Runge-Kutta Lawson schemes for stochastic
%                 differential equations (2020).
%                 https://doi.org/10.5281/zenodo.4062482
%
% Adapted by Nicolas Kuiper and Martin Westberg
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [type, arithmetic_price, geometric_price, arithmetic_std, geometric_std, elapsed_time] = Asian_Heston_Euler_DSL_AV(S0,r,V0,K,T,type,kappa,theta,sigma,rho,Nt,Nsim,R)
tic
X0 = [S0; V0];
A = [r, 0; 0, -kappa];
g0 = @(x) getg0(x,kappa,theta);
```

```
g = cell(2, 1);
g{1} = @(x) [sqrt(x(2, :)).*x(1, :); zeros(1, size(x, 2))];
g{2} = @(x) [zeros(1, size(x, 2)); sigma * sqrt(x(2, :))];

tspan = [0, T];
h = T / Nt;
rng('default');
Z1 = randn(Nt, Nsim);
Z2 = randn(Nt, Nsim);
dW1 = cell(2, 1);
dW1{1} = sqrt(h) * Z1;
dW1{2} = rho * dW1{1} + sqrt(h) * sqrt(1 - rho^2) * Z2;

% Generate Brownian Motions for antithetic paths
dW2 = cell(2,1);
dW2{1} = -dW1{1};
dW2{2} = rho*dW2{1} + sqrt(h)*sqrt(1 - rho^2)*-Z2;

[~,S1, ~] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW1);
[~,S2, ~] = EulerDSLVectorized(X0, A, g0, g, tspan, h, dW2);

% Calculate average price throughout option life
arithmetic_mean = zeros(1,Nsim);
antithetic_arithmetic_mean = zeros(1,Nsim);
geometric_mean = zeros(1,Nsim);
antithetic_geometric_mean = zeros(1,Nsim);
for i = 1:Nsim
    arithmetic_mean(i) = mean(S1(2:end,i));
    antithetic_arithmetic_mean(i) = mean(S2(2:end,i));

    geometric_mean(i) = geomean(S1(2:end,i));
    antithetic_geometric_mean(i) = geomean(S2(2:end,i));
end

% calculate payoffs for each path
if strcmp(type,'call')
    arithmetic_payoff = max(arithmetic_mean - K,0);
    antithetic_arithmetic_payoff = max(
       antithetic_arithmetic_mean - K,0);

    geometric_payoff = max(geometric_mean - K,0);
    antithetic_geometric_payoff = max(
       antithetic_geometric_mean - K,0);
```

```
else
    arithmetic_payoff = max(k - arithmetic_mean,0);
    antithetic_arithmetic_payoff = max(K - antithetic_arithmetic_mean,0);

    geometric_payoff = max(K - geometric_mean,0);
    antithetic_geometric_payoff = max(k - antithetic_geometric_mean,0);
end
% calculate payoffs mean and discount
arithmetic_price = R*mean(arithmetic_payoff);
geometric_price = R*mean(geometric_payoff);

anithetic_arithmetic_price = R*mean(antithetic_arithmetic_payoff);
anithetic_geometric_price = R*mean(antithetic_geometric_payoff);

arithmetic_price = 0.5*(arithmetic_price + anithetic_arithmetic_price);
geometric_price = 0.5*(geometric_price + anithetic_geometric_price);

arithmetic_std = 0.5*sqrt(std(arithmetic_payoff)^2+std(antithetic_arithmetic_payoff)^2);
geometric_std = 0.5*sqrt(std(geometric_payoff)^2+std(antithetic_geometric_payoff)^2);

elapsed_time = toc;

function result=getg0(x,kappa,theta)
    result=ones(size(x));
    result(1,:)=0*result(1,:);
    result(2,:)=result(2,:)*kappa*theta;
end

end
```

B Derivation of Stochastic Runge–Kutta Lawson Schemes for the Heston model

In this appendix, we verify that the SRKL scheme with the suggested coefficients in subsection (2.4.1) indeed yields equivalence to the Heston model.

As per Theorem 3, the uncorrelated version of the Heston model takes the form of the system

$$\begin{cases} \mathrm{d}S(t) = rS(t)\,\mathrm{d}t + \sqrt{V(t)}S(t)\,\mathrm{d}W_1(t), \\ \mathrm{d}V(t) = \kappa(\theta - V(t))\,\mathrm{d}t + \sigma\sqrt{V(t)}\left(\rho\,\mathrm{d}W_1(t) + \sqrt{1-\rho^2}\,\mathrm{d}W_2(t)\right). \end{cases} \tag{1}$$

In particular, for the system (1), we have $\boldsymbol{X}(t) = \big(S(t), V(t)\big) \in \mathbb{R}^2$, and as shown in subsection (2.4.1), the Heston model parameters are:

$$\begin{aligned} &d = M = 2, && A_0 = \begin{bmatrix} r & 0 \\ 0 & -\kappa \end{bmatrix}, \\ &A_1 = A_2 = \mathbf{0}_{2\times 2}, && g_0(t, \mathbf{X}(t)) = \begin{bmatrix} 0 \\ \kappa\theta \end{bmatrix}, \\ &g_1(t, \mathbf{X}(t)) = \begin{bmatrix} \sqrt{V(t)}S(t) \\ \sigma\rho\sqrt{V(t)} \end{bmatrix}, && g_2(t, \mathbf{X}(t)) = \begin{bmatrix} 0 \\ \sigma\sqrt{1-\rho^2}\sqrt{V(t)} \end{bmatrix}. \end{aligned}$$

We can verify that these chosen parameters are correct by extending the right-hand side of Equation (2.22) with $d = M = 2$ to obtain:

$$\begin{aligned} \mathrm{d}\boldsymbol{X}(t) &= \sum_{m=0}^{2} \left[A_m \boldsymbol{X}(t) + \boldsymbol{g}_m(t, \boldsymbol{X}(t))\right] \mathrm{d}W_m(t), \\ \mathrm{d}\boldsymbol{X}(t) &= \left[\boldsymbol{A}_0 \boldsymbol{X}(t) + \boldsymbol{g}_0(t, \boldsymbol{X}(t))\right] \mathrm{d}W_0(t) \\ &\quad + \left[\boldsymbol{A}_1 \boldsymbol{X}(t) + \boldsymbol{g}_1(t, \boldsymbol{X}(t))\right] \mathrm{d}W_1(t) \\ &\quad + \left[\boldsymbol{A}_2 \boldsymbol{X}(t) + \boldsymbol{g}_2(t, \boldsymbol{X}(t))\right] \mathrm{d}W_2(t), \\ \boldsymbol{X}(t_0) &= \boldsymbol{x}_0. \end{aligned}$$

Now, by putting in the aforementioned parameters, together with the fact that

$$\boldsymbol{X}(t) = (S(t), V(t)), \quad W_0(t) = t,$$

we get:

$$
\begin{aligned}
\mathrm{d}\big(S(t),V(t)\big) &= \left[\begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix}\big(S(t),V(t)\big) + \begin{pmatrix} 0 \\ \kappa\theta \end{pmatrix}\right]\mathrm{d}t \\
&+ \left[\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}\big(S(t),V(t)\big) + \begin{pmatrix} \sqrt{V(t)}S(t) \\ \sigma\rho\sqrt{V(t)} \end{pmatrix}\right]\mathrm{d}W_1(t) \\
&+ \left[\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}\big(S(t),V(t)\big) + \begin{pmatrix} 0 \\ \sigma\sqrt{1-\rho^2}\sqrt{V(t)} \end{pmatrix}\right]\mathrm{d}W_2(t) \\
\mathrm{d}\big(S(t),V(t)\big) &= \left[\begin{pmatrix} r & 0 \\ 0 & -k \end{pmatrix}\big(S(t),V(t)\big) + \begin{pmatrix} 0 \\ k\theta \end{pmatrix}\right]\mathrm{d}t \\
&+ \begin{pmatrix} \sqrt{V(t)}S(t) \\ \sigma\rho\sqrt{V(t)} \end{pmatrix}\mathrm{d}W_1(t) \\
&+ \begin{pmatrix} 0 \\ \sigma\sqrt{1-\rho^2}\sqrt{V(t)} \end{pmatrix}\mathrm{d}W_2(t) \\
\big(S(t_0),V(t_0)\big) &= (s_0, v_o),
\end{aligned}
$$

where, after performing the corresponding vector multiplications, this yields:

$$
\begin{aligned}
\mathrm{d}\big(S(t),V(t)\big) &= \begin{pmatrix} rS(t) \\ \kappa\big(\theta-V(t)\big) \end{pmatrix}\mathrm{d}t + \begin{pmatrix} \sqrt{V(t)}S(t) \\ \sigma\rho\sqrt{V(t)} \end{pmatrix}\mathrm{d}W_1(t) + \begin{pmatrix} 0 \\ \sigma\sqrt{1-\rho^2}\sqrt{V(t)} \end{pmatrix}\mathrm{d}W_2(t) \\
\big(S(t_0),V(t_0)\big) &= (s_0, v_o),
\end{aligned}
$$

which is the same as Equation (1), as expected.

To construct the SRKL scheme, we let $T > t_0$ be the maturity of our option, so that $t_0 < t_1 < \cdots < t_N = T$, and let $h_n = t_{n+1} - t_n$ $(n = 0, 1, \cdots, N-1)$ denote the step size of the discretization. Under [6, Lemma 2.1], we let $\boldsymbol{X}$ be the solution of the SDE (2.22) and assume that the the matrices $\boldsymbol{A}_k$ and $\boldsymbol{A}_l$ are constant and commute. Then, the variable $\boldsymbol{V}^n$ defined by

$$
\boldsymbol{V}^n(t) = e^{-L^n(t)}\boldsymbol{X}(t) \tag{2}
$$

with

$$
L^n(t) = \left(\boldsymbol{A}_0 - \gamma^\star \sum_{m=1}^{M} \boldsymbol{A}_m^2\right)(t-t_n) + \sum_{m=1}^{M} \boldsymbol{A}_m\left(W_m(t) - W_m(t_n)\right) \tag{3}
$$

satisfies the SDE

$$
\mathrm{d}\boldsymbol{V}^n(t) = \sum_{m=0}^{M} e^{-L^n(t)}\tilde{\boldsymbol{g}}_m\left(t, e^{L^n(t)}\boldsymbol{x}\right)\mathrm{d}W_m(t), \quad \boldsymbol{V}^n(t_n) = \boldsymbol{X}(t_n) \tag{4}
$$

where

$$\tilde{\boldsymbol{g}}_m(t,\boldsymbol{x}) := \begin{cases} \boldsymbol{g}_0(t,\boldsymbol{x}) - 2\gamma^\star \sum_{m=1}^{M} \boldsymbol{A}_m \boldsymbol{g}_m(t,\boldsymbol{x}), & m = 0 \\ \boldsymbol{g}_m(t,\boldsymbol{x}), & m > 0, \end{cases} \tag{5}$$

where $\gamma\star = \frac{1}{2}$ in the case of Itô integrals and $\gamma\star = 0$ in the case of Stratonovich integrals.

Plugging in the Heston model parameters A_m together with $M = 2$ in equation (3), we get:

$$L^n(t) = \begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix}(t - t_n). \tag{6}$$

Also, inserting the Heston model parameters and $M = 2$ in (5), we obtain:

$$\tilde{\boldsymbol{g}}_m(t,\boldsymbol{x}) = \boldsymbol{g}_m\left(t, \boldsymbol{X}(t)\right) = \begin{cases} \begin{pmatrix} 0 \\ \kappa\theta \end{pmatrix}, & m = 0 \\ \begin{pmatrix} \sqrt{V(t)}S(t) \\ \sigma\rho\sqrt{V(t)} \end{pmatrix}, & m = 1 \\ \begin{pmatrix} 0 \\ \sigma\sqrt{1-\rho^2}\sqrt{V(t)} \end{pmatrix}, & m = 2. \end{cases} \tag{7}$$

We can now expand (4) with $M = 2$ and insert (6) and (7) to obtain:

$$\mathrm{d}\boldsymbol{V}^n(t) = \sum_{m=0}^{2} e^{-L^n(t)} \tilde{\boldsymbol{g}}_m\left(t, e^{L^n(t)}\boldsymbol{x}\right) \mathrm{d}W_m(t)$$

$$\begin{aligned} \mathrm{d}\boldsymbol{V}^n(t) = {} & e^{-L^n(t)} \tilde{\boldsymbol{g}}_0\left(t, e^{L^n(t)}\boldsymbol{x}\right) \mathrm{d}W_0(t) \\ & + e^{-L^n(t)} \tilde{\boldsymbol{g}}_1\left(t, e^{L^n(t)}\boldsymbol{x}\right) \mathrm{d}W_1(t) \\ & + e^{-L^n(t)} \tilde{\boldsymbol{g}}_2\left(t, e^{L^n(t)}\boldsymbol{x}\right) \mathrm{d}W_2(t) \end{aligned}$$

$$\begin{aligned} \mathrm{d}\boldsymbol{V}^n(t) = {} & e^{-\begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix}(t-t_n)} \begin{pmatrix} 0 \\ \kappa\theta \end{pmatrix} \mathrm{d}t \\ & + e^{-\begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix}(t-t_n)} \begin{pmatrix} \sqrt{V(t)}S(t) \\ \sigma\rho\sqrt{V(t)} \end{pmatrix} \mathrm{d}W_1(t) \\ & + e^{-\begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix}(t-t_n)} \begin{pmatrix} 0 \\ \sigma\sqrt{1-\rho^2}\sqrt{V(t)} \end{pmatrix} \mathrm{d}W_2(t). \end{aligned}$$

Lastly, if we define $\mathrm{d}\boldsymbol{V}_n^1(t)$ and $\mathrm{d}\boldsymbol{V}_n^2(t)$ as the results from the equation above after performing the calculation with rows **1** and **2** of each matrix, respectively, we obtain the following system:

$$\begin{cases} \mathrm{d}\boldsymbol{V}_n^1(t) = e^{-r(t-t_n)}\sqrt{V(t)}S(t)\,\mathrm{d}W_1(t), \\ d\boldsymbol{V}_n^2(t) = \kappa\theta e^{\kappa(t-t_n)}\,\mathrm{d}t + \sigma\sqrt{V(t)}e^{\kappa(t-t_n)}\left(\rho\,\mathrm{d}W_1(t) + \sqrt{1-\rho^2}\,\mathrm{d}W_2(t)\right). \end{cases} \tag{8}$$

We now apply [6, Subsection 2.2] and let s be a positive integer, $Z_{ij}^{m,n}$ and $z_i^{m,n}$, $1 \leq i \leq s$, $1 \leq j \leq s$, $0 \leq m \leq M$ be random variables such that:

$$c_m^{n,i} = \sum_{j=1}^{s} Z_{ij}^{m,n}, \qquad c_m^n = \sum_{i=1}^{s} z_i^{m,n}, \tag{8}$$

and we define the discrete updates:

$$\Delta W_m^n = W_m^{n+1} - W_m^n = c_m^n, \tag{9}$$

$$\Delta L_i^n = \left(\boldsymbol{A}_0 - \gamma^\star \sum_{m=1}^{M} \boldsymbol{A}_m^2 \right) \Delta c_m^{n,i} + \sum_{m=1}^{M} \boldsymbol{A}_m c_m^{n,i}, \tag{10}$$

$$\Delta L^n = \left(\boldsymbol{A}_0 - \gamma^\star \sum_{m=1}^{M} \boldsymbol{A}_m^2 \right) \Delta W_0^n + \sum_{m=1}^{M} \boldsymbol{A}_m W_m^n. \tag{11}$$

Plugging the Heston model parameters in both (10) and (11) we obtain:

$$\Delta L_i^n = \begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix} c_0^{n,i}, \qquad \Delta L^n = \begin{pmatrix} r & 0 \\ 0 & -\kappa \end{pmatrix} c_0^n. \tag{12}$$

The general stochastic Runge-–Kutta Lawson scheme takes the form of [6, equation (2.12)], given by:

$$\begin{aligned} \boldsymbol{H}_i &= \boldsymbol{Y}_n + \sum_{j=1}^{s} e^{-\Delta L_j^n} \sum_{m=0}^{M} Z_{ij}^{m,n} \tilde{\boldsymbol{g}}_m(t_n + c_0^{n,j}, e^{-\Delta L_j^n} \boldsymbol{H}_j), \\ \boldsymbol{V}_{n+1}^n &= \boldsymbol{Y}_n + \sum_{i=1}^{s} e^{-\Delta L_i^n} \sum_{m=0}^{M} z_i^{m,n} \tilde{\boldsymbol{g}}_m(t_n + c_0^{n,i}, e^{-\Delta L_i^n} \boldsymbol{H}_i), \\ \boldsymbol{Y}_{n+1} &= e^{\Delta L^n} \boldsymbol{V}_{n+1}^n. \end{aligned} \tag{13}$$

To obtain (13) in terms of the Heston model, we plug in (12) together with $M = 2$, which yields and verifies that:

$$\begin{aligned} \boldsymbol{H}_i &= \boldsymbol{Y}_n + \sum_{j=1}^{s} e^{-\left(\begin{smallmatrix} r & 0 \\ 0 & -\kappa \end{smallmatrix}\right) c_0^{n,j}} \sum_{m=0}^{2} Z_{ij}^{m,n} \tilde{\boldsymbol{g}}_m(t_n + c_0^{n,j}, e^{-\left(\begin{smallmatrix} r & 0 \\ 0 & -\kappa \end{smallmatrix}\right) c_0^{n,j}} \boldsymbol{H}_j), \\ \boldsymbol{V}_{n+1}^n &= \boldsymbol{Y}_n + \sum_{i=1}^{s} e^{-\left(\begin{smallmatrix} r & 0 \\ 0 & -\kappa \end{smallmatrix}\right) c_0^{n,i}} \sum_{m=0}^{2} z_i^{m,n} \tilde{\boldsymbol{g}}_m(t_n + c_0^{n,i}, e^{-\left(\begin{smallmatrix} r & 0 \\ 0 & -\kappa \end{smallmatrix}\right) c_0^{n,i}} \boldsymbol{H}_i), \\ \boldsymbol{Y}_{n+1} &= e^{\left(\begin{smallmatrix} r & 0 \\ 0 & -\kappa \end{smallmatrix}\right) c_0^n} \boldsymbol{V}_{n+1}^n. \end{aligned} \tag{14}$$