Numerical methods for the Heston model

School of Applied Mathematics
A Thesis
Presented to the
Getulio Vargas Foundation

In Partial Fulfillment
of the Requirements for the Degree
M.Sc. of Applied Mathematics

Fernando O. Teixeira

September 19, 2017

Approved for the Division
(Mathematics)

_____

Hugo Alexander de la Cruz Cancino

Acknowledgements

You get pseudo-order when you seek order; you only get a measure of order and control when you embrace randomness. — Nassim Nicholas Taleb

Table of Contents

List of Tables

List of Figures

Abstract

In this thesis we revisit some of the numerical methods for solving the Heston model's European call. Specifically, we approach Euler's, the Kahl-Jackel an two versions of the exact algorithm schemes. To perform this task, firstly we present a literature review which brings stochastic calculus, the Black-Scholes (BS) model and its limitations, the stochastic volatility methods and why they resolve the issues of the BS model, and the peculiarities of the numerical methods - convergence, discretization and stability. Since it is impossible to have a deep approach to all these topics, we provide recommendations when we acknowledge that the reader might need more specifics. We introduce the methods previously cited providing all our implementations in R language. Also, we deliver an R package with these functions and others.

Keywords: Heston, Stochastic, Volatility, Black-Scholes, European call, R

Introduction

The french mathematician Louis Bachelier was the trail-blazer that brought Brownian motion, previously restricted to the field of botanics where it was firstly observed, to the financial framework. He modeled the stock prices as a Brownian motion with drift. In 1973, Black and Scholes [1] designed a model based on the Geometric Brownian Motion to price options.

Options are derivatives that give their bearers the rights to buy or sell a specific asset in a future date and with a predetermined price. They are, by design, affected by small variations in the underlying assets' components, for example, the variance.

The Black-Scholes model was once the standard way of option pricing, but was replaced by more modern models that are now prevalent. One of the main drawbacks of the Black-Scholes model is the strong assumption that the stock returns' volatility is constant. Thus, the implied model's volatility results in a flat surface when plotted against the option's strike price and maturity. Real world implied volatility varies with the strike price and maturity, forming what is called the 'volatility smile'.

The Heston model is an extension of the Black-Scholes model that tackles this volatility issue replacing the constant volatility with a stochastic process. There are many models that approach volatility stochastically, but the Heston has valuable characteristics such as presenting an analytical solution to the option pricing and also having a computationally simple implementation when compared to more sophisticated competitors.

We aim with this thesis to better understand the Heston model, its quirkinesses, associated numerical methods and their implementations, and eventually discuss its pros and cons. For such, we will not only discuss the challenges in a vaguely way, but show code snippets of our implementations in R programming language [2] to help clarify possible associated difficulties. Furthermore, since we didn't find code for the exact algorithm implementation we intend to release a CRAN package with the codes as a didactic tool.

This thesis is divided into five chapters, the first being this introduction. Following, we have a literature review that mainly addresses stochastic calculus, the Black-Scholes models and the Heston model. Thereafter we present the different Heston model implementations. Chapter 4 brings the results of the tests we perform and finally, chapter 5 presents the conclusion and work limitations.

Chapter 1

Theoretical Framework

This chapter presents the concepts of stochastic calculus, from the historic conception of how it first arose through the basic principles and applications in finance. We address with more care the classical Black-Scholes model and its limitations and the Heston model. This model is also well known, it brings the concept of stochastic volatility in it, which brings its results closer to reality.

1.1 Stochastic Calculus

Stochastic calculus arises from stochastic processes and allows the creation of a theory of integration where both the integrand and integrator terms are stochastic processes. Stochastic calculus was created by the Japanese mathematician Kiyosi Itô [1] in the 1940s and 1950s and is used for modeling financial options and in another wide variety of fields [3]. In this chapter we present the historical contexts in which the tools and models are used, but our focus is introducing the concepts and notations that will be further used in our work.

1.1.1 The Stochastic differential equation - SDE

At first, before introducing stochastic differential equation, it is helpful to start with ordinary differential equation. Let $x(t) = x_t$ denote a population at time t so that the change in the population at time t is given by the following deterministic differential equation:

$$\begin{aligned} dx_t &= f(t, x_t)dt \\ x(0) &= x_0 \end{aligned} \tag{1.1}$$

[1] There is another important stochastic integral, called the *Stratonovich Integral* that unlike the Itô's integral, respects the conventional calculus chain rule. Also, the integral is evaluated at the interval's midpoint, instead of its left extreme. A Stratonovich integral can be expressed as an Itô integral and vice versa.

We now add a "noise" to this equation:

$$dx_t = \overbrace{\underbrace{f(t,x_t)}_{\text{drift}}\,dt}^{\text{deterministic}} + \overbrace{\underbrace{g(t,x_t)}_{\text{diffusion}}\,dW_t}^{\text{random shock}} \tag{1.2}$$
$$x(0) = x_0$$

This "noise" dW_t is a *random* Wiener process time derivative (which will be clarified below) and x_0 is our initial value.

The $g(t, x_t)$ part of the SDE is often referred as a *diffusion process*, these usually have a continuous paths. Before moving on, we must carefully define what the term *random* means and the best way to begin doing so is to precisely define a probability space:

Definition 1.1.1 (Probability Space)**.** A triple $(\Omega, \mathcal{U}, \mathcal{P})$ is called a *probability space* provided Ω is any set, $\mathcal{U}$ is a σ-algebra of subsets of Ω and $\mathcal{P}$ is a probability measure on $\mathcal{U}$.

1.1.2 Brownian Motion

The Brownian motion is the name given to the irregular motion observed in the motion of pollen particles suspended in fluid resulting from particle collision with atoms or molecules. It is named after Robert Brown, the first to have observed the movement in 1828. He noted two characteristic in the pollen movement [3]:

- the path of a given particle is very irregular, having a tangent at no point
- the motion of two distinct particles appear to be independent

The first quantitative works in Brownian motion come from an interest in stock price fluctuation by Bachelier in 1900. Albert Einstein also leaned over the subject and in 1905 derived the transition density for Brownian motion from molecular-kinetic theory of heat [3,4].

In 1923, the Wiener process was coined in honor of Norbert Wiener mathematical proof of existence of the Brownian motion and stating its properties.[2]

Definition 1.1.2 (Wiener Process)**.** Given a probability space $(\Omega, \mathcal{U}, \mathcal{P})$, a stochastic process W_t defined in this space is a *Wiener process* if it satisfies the following properties:

- $W_0 = 0$
- The change in W, given by $\Delta W = W_{t+1} - W_t$, is normally distributed with mean zero and standard deviation $\sqrt{\Delta t}$, meaning that $\Delta W = \epsilon\sqrt{\Delta t}$, where ϵ is $N(0, 1)$.

[2] More can be found on [5–7].

- If the increment Δt_1 does not overlap with the time increment Δt_2, then ΔW_1 and ΔW_2 are independent.
- The process is continuous, meaning that there are no jumps in the path.
- The process is Markovian. This means that the conditional expectation of W_{t+1} given its entire history is equal to the conditional expectation of W_{t+1} given today's information. This can be written as: $E[W_{t+1}|W_1, ..., W_t] = E[W_{t+1}|W_t]$.
- Consider the time interval $[0, t]$ with n equally spaced intervals given by $t_i = \frac{it}{n}$. Then the paths of the Brownian motion have unbounded variation, this means that they are not differentiable and go towards infinity as n increases. The quadratic variation is given by $\sum_{i=1}^{n} (Z_{t_i} - Z_{t_{i-1}})^2 \to t$, meaning that when n increases it stays constant at t.

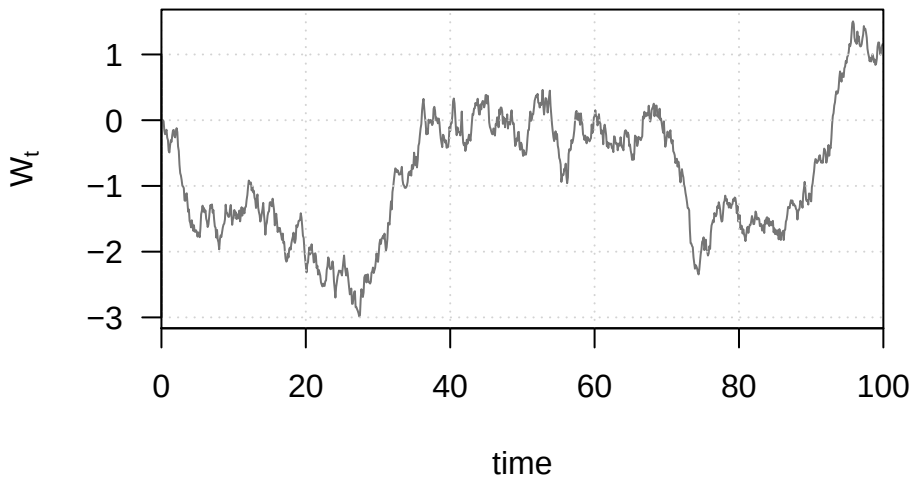

Figure 1.1: A Wiener process trajectory path example

Correlated Brownian Motions

Two independent Brownian motions that are correlated can describe a new process Z_t. Let W_1 and W_2 be these two *independent* Brownian motions and let $-1 \leq \rho \leq 1$ be a given number. For $0 \leq t \leq T$ define the new process Z_t as [3]:

$$Z_t = \rho W_{1,t} + \sqrt{1 - \rho^2} W_{2,t} \tag{1.3}$$

This equation is a linear combination of independent normals at each timestep t, so Z_t is normally distributed. It is proven that Z is a Brownian motion and that Z and $W_{1,t}$ have correlation ρ [3].

1.1.3 Itô's Integral

Formally, the SDE presented in equation (1.2) only exists because we can rewrite it in the form [6–11]:

$$x_t = x_0 + \int_0^t f(s, x_s)ds + \int_0^t g(s, x_s)dW_s \tag{1.4}$$

for some $f(s, x_s)$, $g(s, x_s)$ and $s \in [0, t]$.

The Itô integral can, as the Riemann integral, be approximated by a finite sum. Also, it has a definition as a certain limit. Itô's lemma 1.1.1 plays the same role as the fundamental theorem of calculus in allowing to evaluate integrals. It is the formal definition and presents an extra term not encountered in the conventional calculus theorem that is due to the non-smoothness characteristics of Brownian motion paths. It is possible, though, to define the integral in a less rigorous way:

$$Y_{\Delta t}(t) \approx \sum_{t_k < t} g(t_k)\Delta W_k \tag{1.5}$$

with the usual notions $t_k = k\Delta t$, and $\Delta W_k = W(t_{k+1}) - W(t_k)$. And in a more rigorous form, if the limit exists, then the Ito integral is:

$$Y(t) = \lim_{\Delta t \to 0} Y_{\Delta t}(t) \tag{1.6}$$

It is essential that the *forward difference* is used rather than the backward difference, which would be wrong.

Lemma 1.1.1 (Itô's Lemma)**.** *Assume that S_t has a stochastic differential given by:*

$$dS_t = \mu_t dt + \sigma_t dW_t \tag{1.7}$$

for μ_t, σ_t and $t \in [0, T]$. Assume $u : \mathbb{R} \times [0, T] \to \mathbb{R}$ is continuous and that $\frac{\partial u}{\partial t}$, $\frac{\partial u}{\partial x}$, $\frac{\partial^2 u}{\partial x^2}$ exist and are continuous.

$$Y_t := u(S_t, t)$$

Then Y has the following stochastic differential:

$$\begin{aligned} dY_t &= \frac{\partial u}{\partial t}dt + \frac{\partial u}{\partial x}dS_t + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}\sigma_t^2 dt \\ &= \left(\frac{\partial u}{\partial t} + \mu_t\frac{\partial u}{\partial x} + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}\sigma_t^2\right)dt + \sigma_t\frac{\partial u}{\partial x}dW_t \end{aligned} \tag{1.8}$$

where the argument of u, $\frac{\partial u}{\partial x}$ and $\frac{\partial^2 u}{\partial x^2}$ above is (S_t, t) .

Equation (1.8) is the stochastic equivalent to the chain rule, also known as Itô's formula or Itô's chain rule. The proof to this theorem is based on the Taylor expansion of the function $f(S_t, t)$ [6,8]. For practical use you should write out a second-order Taylor expansion for the function to be analyzed and apply the multiplication table [3] presented in Table 1.1.

	dt	dW_t
dt	0	0
dW_t	0	dt

Table 1.1: Box calculus

Itô's Integral Properties

Let f, $g \in \mathcal{V}$ and let $0 \leq t_0 < u < T$. Then

(i) $\int_{t_0}^{T} f dB_t = \int_{t_0}^{u} f dB_t + \int_{u}^{T} f dB_t$

(ii) $\int_{t_0}^{T} (\alpha f + \beta g) dB_t = \alpha \int_{t_0}^{T} f dB_t + \int_{t_0}^{T} \beta g dB_t$

(iii) $\mathbb{E}\left[\int_{t_0}^{T} f dB_t\right] = 0$

(iv) $\mathbb{E}\left[\left(\int_0^t H_s \, dB_s\right)^2\right] = \mathbb{E}\left[\int_0^t H_s^2 \, ds\right]$ (Isometry)

(v) $\mathbb{E}\left[\int_{t_0}^{T} f dB_t \mid \mathcal{F}_s\right] = \int_{t_0}^{s} f dB_t, \qquad for\ s < T.$ (Martingale[3])

1.2 Black-Scholes Model

1.2.1 Basics

The Black-Scholes (B-S) model arises from the need to price european options in the derivative markets. Derivatives are financial instruments traded in the market, stock exchange or over-the-counter (OTC) market, whose values depend on the values of an underlying asset. [1,12,13]

- A call option is a derivative that gives its bearer the right, but not the obligation, to purchase a specific asset by a fixed price before or on a given date.
- A put option is a derivative that gives its bearer the right, but not the obligation, to sell a specific asset by a fixed price before or on a given date.

The trading price of the option is called the option *premium* and the asset from which the option derives is called the *underlying asset*. This asset may be interest rates, exchange rates, stock exchanges indices, commodities or stocks. The fixed price in

[3]A martingale is a stochastic process with certain characteristics. The main one is that the expected value in time $t+1$ for X is the X value in t. This means there are no winning strategies when we are dealing with martingales (unlike when we play poker, for example). A Wiener process is a martingale.

contract in which the underlying asset might to be bought or sold is the *strick price*. The option expiration date is called the *maturity*. [1,13]

There are two major different option types: European and American. The difference between these two is that the bearer of the first may exercise it only at the end of its life, at its maturity while the latter can be exercised at any given time until its maturity. [1,14]

Definition 1.2.1 (Intrinsic value)**.** The intrinsic value of a call is the difference between the underlying asset price and the strike price. The put's intrinsic value operates the other way around, being the difference between the strike and the underling asset prices.

Geometric Brownian Motion

A stochastic process S_t is a Geometric Brownian Motion[4] if it is described by the solution of the following stochastic differential equation [3,8,15].

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{1.9}$$

for given constants $\mu \in \mathbb{R}$ and $\sigma > 0$. Also, the assumed initial value is positive, $S_0 > 0$.

Figure 1.2 shows the GBM,[5] which is quite often applied to model the dynamics of some financial assets because of its properties [16]. Equation (1.10) shows the formula to generate a GBM and we provide proof of this solution in appendix A[6]

$$S_t = S_0 \times exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right), \quad t > 0 \tag{1.10}$$

[4]There is an Arithmetic Brownian Motion: $dS_t = \mu dt + \sigma dB_t$. More information can be obtained at [3].

[5]Also known as exponential Brownian motion.

[6]An intuitive proof can be found at [17].

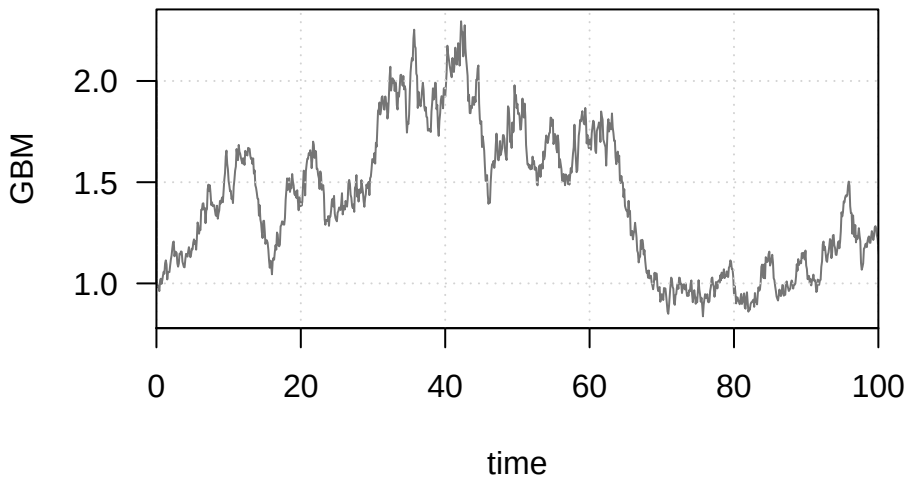

Figure 1.2: A GBM trajectory path example

1.2.2 The model

The Black-Scholes model provides analytical solution to the price of a European call at time t and can be described as follows [1,5,12]:

$$C(S_t, t) = N(d_1)S_t - N(d_2)Ke^{-r(T-t)} \tag{1.11}$$

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \tag{1.12}$$

$$d_2 = d_1 - \sigma\sqrt{T-t} \tag{1.13}$$

Where:

- S_t is the spot price of the underlying asset at time t
- r is the risk free rate (generally an annual rate)[7]
- σ is the volatility of returns of the underlying asset [8]
- $N(\cdot)$ is the cumulative distribution function of the standard Gaussian distribution
- K is the strike price
- $T - t$ is the time to maturity

Also, the stock price path is a Geometric Brownian Motion as previously stated, and is under the risk-neutral measure with the following dynamics [5,18]:

$$dS_t = (r - q)S_t d_t + \sigma S_t dW_t \tag{1.14}$$

[7]Assumed to be constant.
[8]See footnote 1.

Where dW_t is a Wiener process [1,18], r is the risk free rate and q is the dividend yield[9] and t denotes the current point in time.

1.2.3 Limitations

Although the Black-Scholes is very popular and the *de facto* standard in the market there are implications to the B-S model assumptions that affect the results and that are unrealistic. The main assumption that does not hold up is the deterministic (constant) volatility, that can more accurately be described as a stochastic process since we observe that small moves usually are followed by small moves and large moves by large moves. [5,12]

Other assumptions that are critical to the B-S model and are not always observed in practice refer to the asset's continuity through time (no jumps), being allowed to perform continuous hedge without transactions costs and normal (Gaussian) returns.

Most models focus on the volatility problem because transaction costs often translate to rises in volatility and fat-tails (abnormal) returns can be simulated by stochastic volatility and market or volatility jumps.

1.3 Stochastic Volatility models

Introducing stochastic volatility to models brings complexity, but enables modeling some features observed in reality that are crucial, like the randomic market volatility effects, skewness (market returns are more realistically modeled) and volatility smile[10] (see Figure 1.3). This kind of model is applied highly succesfully in foreign exchange and credit markets.

Definition 1.3.1 (Volatility Smile)**.** Volatility smiles are implied volatility patterns that arise in pricing financial options. In particular for a given expiration, options whose strike price differs substantially from the underlying asset's price command higher prices (and thus implied volatilities) than what is suggested by standard option pricing models. These options are said to be either deep in-the-money or out-of-the-money.

Furthermore, stochastic volatility models use statistical methods as foundations to price and forecast options' behaviors and the underlying's security volatility is arbitrary. The Heston, the 3/2 and other models, like the GARCH[11] and SABR,[12] are considered standard smile models.

[9] r and q are assumed to be constant.

[10] The name derives from the concave shape of the graph, which resembles a smile.

[11] generalized autoregressive conditional heteroscedasticity.

[12] stochastic alpha, beta, rho.

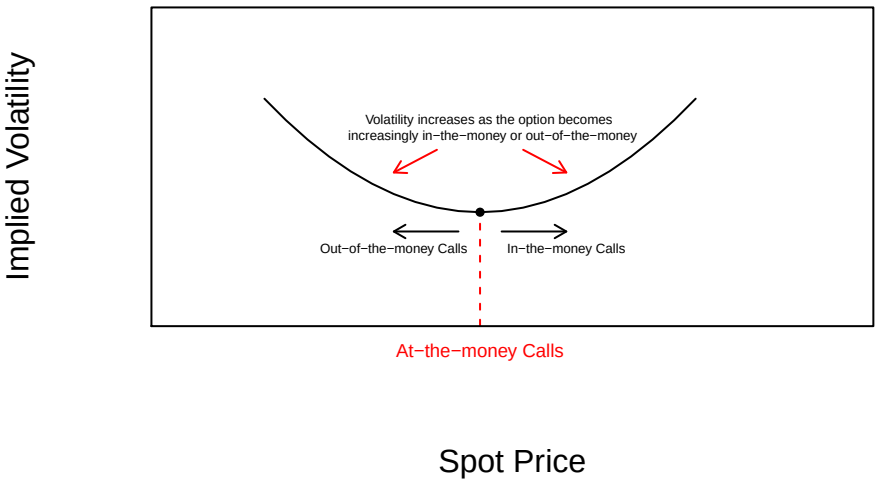

Figure 1.3: Volatility Smile

1.3.1 Cox-Ingersoll-Ross model

The Cox-Ingersoll-Ross (CIR) model is a well-known short-rate model that describes the interest rate movements driven by one source of market risk. The dynamics are described as follows[19,20]:

$$dr_t = k(\theta - r_t)dt + \sigma\sqrt{r_t}dB_t \tag{1.15}$$

Where, r_t is the short rate interest described by parameters κ the speed of mean reversion, θ the long-run mean variance and σ the volatility of the variance process.

This model has been widely used to describe the dynamics of the short rate interest because it has some fundamental features like intuitive parametrization, nonnegativity and pricing formulas. Besides, it takes account of anticipations, risk aversion, investment alternatives and preferences about consumption timing and allows for detailed predictions about how changes in a wide range of underlying variables affect the term structure[19]. Furthermore, this equation constitutes one of the two Heston model equations with the volatility taking the short rate interest place.

1.3.2 Heston Model

Heston model was introduced in 1993 by Steven Heston to solve the deterministic volatility problems. It was designed to analize bond and currency options and it introduced the following equations, which represent the dynamics of the stock price

and the variance processes under the risk-neutral measure [20,21]:

$$dS_t = \mu S_t dt + \sqrt{V_t} S_t dW_t^* \tag{1.16}$$

$$dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t} dB_t \tag{1.17}$$

The second equation, as described in Section 1.3.1, is the CIR model equation. The first equation states the asset price process. μ is the asset's rate of return, $dW_{t,1}$ and $dW_{t,2}$ are two correlated wiener processes with correlation coefficient of ρ.

1.3.3 Other Models

Ornstein-Uhlenbeck

The Ornstein-Uhlenbeck is the earliest recorded SDE. Named after Leonard Ornstein and George Eugene Uhlenbeck, it is a stochastic process that describes the acceleration of a pollen particle in a liquid subject to bombardments by molecules [3]. As we can observe in equation (1.18), x_t represents the one dimension velocity of the particle, thus dx_t is the *change* in velocity, in other words, its acceleration. The $-\theta x_t$ component slows down the acceleration and is to be understood as frictional force. Besides, we add a noise W_t with intensity σ that models the random bombardment by the molecules.

$$dx_t = -\theta x_t dt + \sigma dW_t \tag{1.18}$$

With θ and σ being positive constants. Expressing in terms of x_t we get:

$$x_t = e^{-\theta t} \times \left[x_0 + \sigma \int_{t=0}^{T} e^{\theta t} dW_s \right] . \tag{1.19}$$

Langevin

The Langevin equation describes a system that consists of the molecular bombardment of a speck of dust on a water surface. We know that the intensity of the bombardement does not depend on the state variables [22,23].

$$m\frac{dv}{dt} = -\zeta v + \delta F(t) \tag{1.20}$$

m is the mass of the particle, v it's velocity, $-\zeta v$ is the frictional force, which is proportional to the velocity, and $\delta F(t)$ is a *fluctuating* force (random) to the frictional force.

1.4 Numerical Methods

Numerical methods are tools that are often applied to solve stochastic differential equations because most of these do not have explicit solution. This means that we are not able to solve these equations using symbolic computation. Although we are

unable to find an analitical solution, when facing real problems, the approximation given by a numerical method is often sufficient. Alongside the analytical issue, the need to calculate the SDE's trajectory through time is the main reason why studying numerical methods is so important. An implementation of a numerical method is called a numerical algorithm.

We will simulate sample paths of time discrete approximations implemented in the R programming language [2] that we base on a finite discretization of a time interval $[t_0, T]$. We shall generate approximate values of the sample path for each step contained in the discretized interval [22].

In the fixed step methods, the distance between two contiguous points is the distance $d_i = t_i - t_{i-1} = \frac{T-t_0}{N} \quad \forall i \mid 1 \leq i \leq N \in \mathbb{N}$. N being the time interval partition number.

According to Kloeden [22], in the stochastic realm, simulated sample paths can be statistically analysed to find how good an approximation is compared to an exact solution. Moreover, the computational costs such as time and memory increases polynomially with the problem's dimension, which is good, and it is possible to apply variance reduction methods that allow a considerable decrease in the required sample size.

1.4.1 Convergence

As soon as we talk about numerical methods we are required to approach the topic of approximations and how to handle them. Methods efficiency receive the name of *convergence order*. In the SDE domain there are two main methods of convergency, that are classified according to their criteria. Firstly, we present the *strong order of convergence*. A method is said to have strong convergence δ to Y if a time discretized Y_δ of a continous-time process Y, with δ being the maximum time increment of the discretization, and for any fixed time horizon T holds true that [16]:

$$\mathbb{E} \mid Y_\delta(T) - Y(T) \mid \leq C\delta^\gamma, \ \ \forall \delta < \delta_0$$

with $\delta_0 > 0$ and $\mathcal{C}$ a constant not depending on δ. Strong convergence addresses the problem of solutions' trajectories. For specific conditions, the Euler method has strong convergence order $\gamma = \frac{1}{2}$. Furthermore, there is the *weak order of convergence*. The weak convergence

$$\mid \mathbb{E}p(Y_n) - \mathbb{E}p(Y(\tau)) \mid \leq C\Delta t^\gamma$$

Strong and weak convergence are not mutually exclusive [16]. That means that a method with a given strong order of convergence might have a higher weak order of convergence too. This is the case for the Euler scheme, with a strong order of convergence of 1/2 and a weak order of 1 (under some conditions). For a more detailed and rigorous explanation of convergence we recommend consulting [24].

It is worth noting that, altough schemes have a given convergency order, it is not unusual that they behave better than their order for some SDEs specifications.

1.4.2 Discretization

We know that convergence is an important feature to a numerical method and studies have found not all time discrete possible approximations of an SDE converge in a useful sense to the solution process as the step size adopted tends toward zero [25,26]. Moreover, particularly for SDEs, some of the more rapidly convergent methods available for ordinary differential equations (ODE) do not work, such as higher order Runge-Kutta methods.[13]

One of the methods that do work for ODEs and SDEs is the Euler method, named after the Swiss mathematician Leonhard Euler. Figure 1.4 shows an example of an implementation for the Newton's cooling law with timestep of 2 seconds compared to its analytical solution. This method (*a.k.a.* forward Euler method) is a first-order numerical procedure. It is the most basic explicit method[14] for numerical integration.

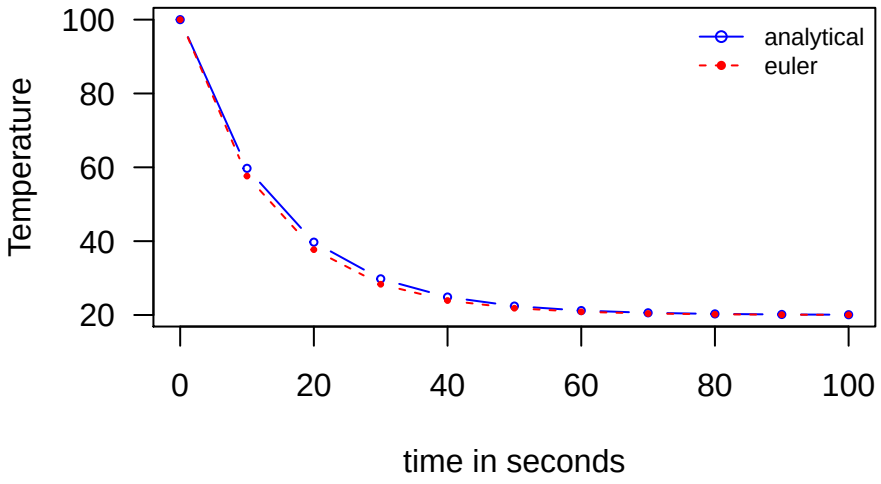

Figure 1.4: Analytical x Euler solutions

The method is first-order, as stated above, this means that the error in each step is a proportion of the square of the step size. Also, the global error at a given time is a function of the step size. We proceed to apply the Euler method to SDEs. Consider the equation:

$$dS_t = \mu(S_t, t)dt + \sigma(S_t, t)dW_t \tag{1.21}$$

dW_t is the Brownian motion, μ and σ are functions depending on S_t and t, over an

[13]The euler method is the simplest Runge-Kutta method.

[14]Explicit methods calculate the state of a system at a later time from the state of the system at the current time. Mathematically we have something like $Y(t + \Delta t) = F(Y(t))$.

interval $[0, T]$, and we want to discretize it as $0 = t_1 < t2 < \cdots < t_m = T$ with increments equally spaced d_t.

Integrating it from t to dt we have the starting point for our (and any) discretization scheme:

$$S_{t+dt} = S_t + \int_t^{dt} \mu(S_u, u)du + \int_t^{dt} \sigma(S_u, u)dW_u \tag{1.22}$$

To use the Euler discretization is the equivalent of approximating integrals using the left-point rule as in Figure 1.5[15], we then have:

$$\begin{aligned}
\int_t^{t+dt} \mu(S_u, u)dW_u &\approx \mu(S_t, t) \int_t^{t+dt} dW_u \\
&= \mu(S_t, t)(W_{t+dt} - W_t) \\
\int_t^{t+dt} \sigma(S_u, u)dW_u &\approx \sigma(S_t, t) \int_t^{t+dt} dW_u \\
&= \sigma(S_t, t)(W_{t+dt} - W_t) \\
&= \sigma(S_t, t)\sqrt{dt}Z
\end{aligned}$$

$W_{t+dt} - W_t$ and $\sqrt{dt}Z$ have identical distribution, Z being a standard gaussian variable. The Euler discretization of equation (1.22) is then:

$$S_{t+dt} = S_t + \mu(S_t, t)dt + \sigma(S_t, t)\sqrt{dt}Z \tag{1.23}$$

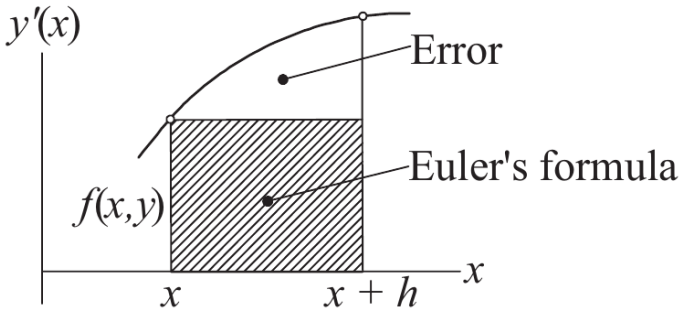

Figure 1.5: Graphical representation of Euler's formula

Source: Numerical methods in Engineering with Python 3.

[15] See Kiusalaas [27]

Euler method - Heston model

We now proceed to apply the method to our model of interest. We retake the equations (1.16) and (1.17). We begin showing how to discretize the latter [16,24]:

$$V_{t+dt} = V_t + \int_t^{t+dt} \kappa(\theta - V_u)du + \int_t^{t+dt} \sigma\sqrt{V_u}dB_u \tag{1.24}$$

Which discretized turns out as:

$$\begin{aligned}\int_t^{t+dt} \kappa(\theta - V_u)du &\approx \kappa(\theta - V_t)dt \\ \int_t^{t+dt} \sigma\sqrt{V_u}dB_t &\approx \sigma\sqrt{V_t}(W_{t+dt} - W_t) \\ &= \sigma\sqrt{V_t dt}Z_v\end{aligned}$$

And leaves us with:

$$V_{t+dt} = V_t + \kappa(\theta - V_t)dt + \sigma\sqrt{V_t dt}Z_v \tag{1.25}$$

Z_v is a standard normal variable. To avoid problems with negative values in $\sqrt{V_t}$ we apply the *full truncation* scheme, which substitutes V_t with $V_t^+ = max(0, V_t)$.[16]

For the S_t SDE we proceed similarly:

$$S_{t+dt} = S_t + \mu\int_t^{t+dt} S_u du + \int_t^{t+dt} \sqrt{V_u}S_u dW_u \tag{1.26}$$

Discretizing we have:

$$\begin{aligned}\int_t^{t+dt} S_u du &\approx S_t dt \\ \int_t^{t+dt} \sqrt{V_u}S_u dW_u &\approx \sqrt{V_t}S_t(W_{t+dt} - W_t) \\ &= \sqrt{V_t dt}S_t Z_s\end{aligned}$$

Z_s is a standard normal variable with correlation ρ with Z_v. We have:

$$S_{t+dt} = S_t + \mu S_t dt + \sqrt{V_t dt}S_t Z_s \tag{1.27}$$

1.4.3 Stability

Most differential equations, deterministic or stochastic, cannot be solved explicitly [22]. Hence, stability studies begin with computers and is associated with numerical methods and approximations. Convergent methods were resulting in bigger errors than what was expected that could not be only due to discretization error. Eventually, scientists discovered that this unexpected problem was caused by accumulation of successive

[16] Another possible scheme (not used in this work) is the *reflection* scheme where we replace V_t with $\mid V_t \mid$

truncation errors. Figure 1.6 retakes the cooling example previously approached to show instability due to an increase in size of the timestep and extending to 600 seconds. The top plot, with a step of 25 still converges to the real solution, but we already observe an odd behaviour since the numerical method doesn't follow the analytical solution, but instead revolver around it until convergence. The bottom plot, presents a small increase in the step $h = 29$, and this small increment is enough results in numbers completely off target.

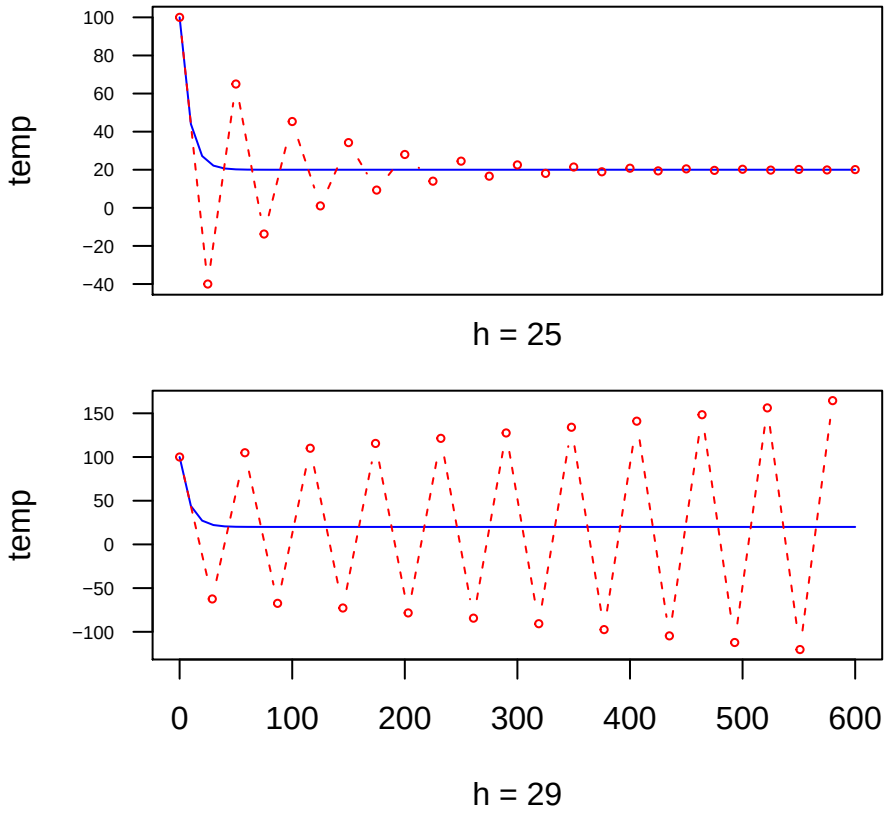

Figure 1.6: Euler's stability whith different timesteps

We know that binary machines like computers are not able to represent all the real numbers, but only a subset of them. Thus, solving these errors is not straightforward since it's not possible to eliminate *all* truncation error when using a computer and dealing with numerical solutions. When faced to an incorrect (not acceptable) solution, we have to evaluate and distinguish between two distinct situations:

i Rounding errors are considerably amplified by the algorithm. This situation is called numerical instability.

ii Small perturbations of data generate large changes in the solution. This is

termed an ill-conditioned (or sensitive) problem.

Examples of these two classes of problem can be found in [21].

Stiff equations appear very often in mathematical problems and refer to differential equations for which a numerical methods might be unstable for not small enough stepsizes [28]. A differential equation of the form $y' = f(t, y)$, if its exact solution $y(t)$ includes a term that decays exponentially to zero as t increases, but whose derivatives are greater in magnitude than the term itself. In other words, if it requires a significant depression of the stepsize to avoid stability lost. This is a loose definition but, since we are dealing with numerical methods a proper mathematical definition isn't required. Tipically, these equations are of the form e^{-ct}, where c is a large positive constant [29]. A practical example of the stiff behavior is the following differential equation [30]:

$$y' = -100y, \quad t > 0, \quad y_0 = 1$$

Whose exact solution is $y_t = e^{-100t}$ and goes to zero as t increases. Applying Euler's method to this equation with $h = 0.1$ we stumble in the following equation

$$y_{n+1} = y_n - 100hy_n = -9y_n$$

which is wrong, since it yields an exponencially growing solution $y_n = (-9)^n$. On the other hand, if our timestep is smaller $h = 10^{-3}$, our solution using Euler's method becomes $y_n = (0.9)^n$. This solution leads to an accurate behavior regarding the exact solution, it rapidily decays to zero.

Sometimes, it is interesting to rank differential equations that are more or less stiff. Thus, people compute the quotient of the largest and the smallest eigenvalues of a linear system. They call it the equations' *stiffness ratio* and, usually the bigger the stiffness ratio, the more likely is to be stiff [28].

Stability Domain

Let's take the equation:

$$y' = \lambda y, \quad t \geq 0, \qquad y_0 = 1 \tag{1.28}$$

where $\lambda \in \mathbb{C}$ or in other terms $\lambda = \lambda_r + i\lambda_i$ and whose solution is $y + t = e^{\lambda t}$. We can rewrite this equation as a system:

$$\frac{d}{dt}\begin{bmatrix} y^1 \\ y^2 \end{bmatrix} = \begin{pmatrix} \lambda_r & -\lambda_i \\ \lambda_i & -\lambda_r \end{pmatrix}\begin{bmatrix} y^1 \\ y^2 \end{bmatrix} \tag{1.29}$$

The $\lim_{t\to\infty} y_t = 0$ if and only if $\mathbb{R}\lambda < 0$. The *linear stability domain* $\mathcal{D}$ is defined as the set of all numbers $\Delta\lambda \in mathbbC$ such that $\lim_{n\to\infty y+n=0}$, with $\Delta > 0$ being the stepsize. Or, as stated in Kloeden [22], the suitable values of the stepsize are expressed in terms of *region of absolute stability*, consisting of the complex numbers $\lambda\Delta$ for which an error in y_0 at t_0 will not grow in subsequent iterations of the method.

Without entering all the details, for these we recommend [22,28,30], the euler's stability domain is:

$$\mathcal{D}_{Euler} = z \in \mathbb{C} :\mid 1 + z \mid < 1$$

which represents the interior of a complex disc of unit radius and centre $z = -1$ as can be seen in Figure 1.7, on the left. The right side of the Figure 1.7 shows the stability region called A-stability.[17] If a method is A-stable, the stepsize Δ is only constraint by accuracy.

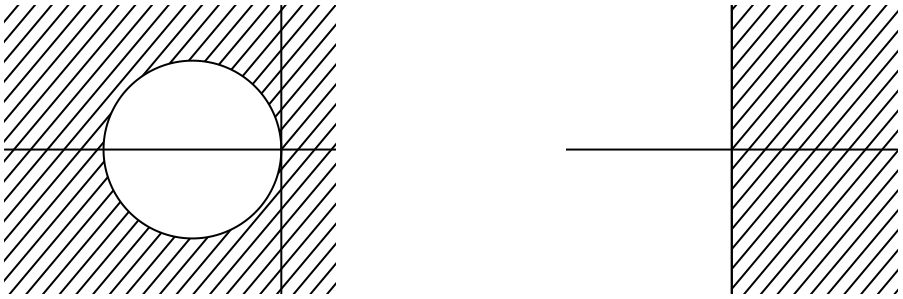

Figure 1.7: Stability domains

Thereby, we claim that stability method study is an important topic, since it enables achieving solutions that are good to stiff equations without having to overly reduce our timesteps which can be very computationally costly.

[17]Mathematically: $\mathcal{D} \subseteq \{z \in \mathbb{C} : \mathbb{R}z < 0\}$.

Chapter 2

The Heston Model Implementation

In section 1.3.2 we presented Heston's SDE system in one of its structures. Another common way [31–33] to write down the system is using the property presented in subsection 1.1.2 as in equation (2.1).

$$\begin{aligned} dS_t &= \mu S_t dt + \rho\sqrt{V_t} dB_t + \sqrt{1-\rho^2}\sqrt{V_t} S_t dW_t \\ dV_t &= k(\theta - V_t)dt + \sigma\sqrt{V_t} dB_t \end{aligned} \tag{2.1}$$

2.1 Characteristic Function

The Heston model characteristic function is firstly presented in the 1993 Steven Heston's paper [20] and is described below [34]:

$$f(S_t, V_t, t) = e^{A(T-t)+B(T-t)S_t+C(T-t)V_t+i\phi S_t} \tag{2.2}$$

If we let $\tau = T - t$, then the explicit form of the Heston characteristic function is:

$$\begin{aligned} f(i\phi) &= e^{A(\tau)+B(\tau)S_t+C(\tau)V_t+i\phi S_t} \\ A(\tau) &= ri\phi\tau + \frac{\kappa\theta}{\sigma^2}\left[-(\rho\sigma i\phi - \kappa - M)\tau - 2\ln\left(\frac{1-Ne^{M\tau}}{1-N}\right)\right] \\ B(\tau) &= 0 \\ C(\tau) &= \frac{(e^{M\tau}-1)(\rho\sigma i\phi - \kappa - M)}{\sigma^2(1-Ne^{M\tau})} \end{aligned}$$

Where:

$$\begin{aligned} M &= \sqrt{(\rho\sigma i\phi - \kappa)^2 + \sigma^2(i\phi + \phi^2)} \\ N &= \frac{\rho\sigma i\phi - \kappa - M}{\rho\sigma i\phi - \kappa + M} \end{aligned}$$

This function is the driving force behind the following formula, that calculates the fair value of a European call option at time t, given a strike price K, that expires at

time T [34]:

$$\begin{aligned} C =& \frac{1}{2}S(t) + \frac{e^{-r(T-t)}}{\pi}\int_0^\infty \Re\left[\frac{K^{-i\phi}f(i\phi+1)}{i\phi}\right]d\phi \\ & - Ke^{-r(T-t)}\left(\frac{1}{2} + \frac{1}{\pi}\int_0^\infty \Re\left[\frac{K^{-i\phi}f(i\phi)}{i\phi}\right]d\phi\right) \end{aligned} \tag{2.3}$$

2.2 Euler Scheme

Given the fact that the underlying asset is temporal dependent upon the solution of the SDE's volatility, we simulate the volatility's path before the asset's. If the Black-Scholes model enabled using Ito's Lemma directly for solving S_t, this equation system requires numerical methods. We present here the Euler Scheme - Full Truncation algorithm (and compare to other similar schemes) [31] along with some insights on how it was implemented in R. The Euler discretization brings approximation paths to stock prices and variance processes. If we set $t_0 = 0 < t_1 < \cdots < t_M = T$ as partitions of a time interval of M equal segments of length δt, we have the following discretization for the stock price:

$$S_{t+1} = S_t + rS_t + \sqrt{V_t}S_tZ_s \tag{2.4}$$

And for the variance process:

$$V_{t+1} = f_1(V_t) + \kappa(\theta - f_2(V_t)) + \sigma\sqrt{f_3(V_t)}Z_v \tag{2.5}$$

Z_s being a standard normal random variable, i.e. $N \sim (0, 1)$, we set Z_t and Z_v as two independent standard normal random variables and Z_s and Z_v having correlation ρ. This means we can write $Z_s = \rho Z_v + \sqrt{1-\rho^2}Z_t$.

The immediate observable problem in the proposed discretization scheme is that V can become negative with non-zero probability making the computation of $\sqrt{V_t}$ impossible [32]. There are several proposed fixes that can be used as you can see below:

Table 2.1: Truncation schemes

Scheme	$f_1(V_t)$	$f_2(V_t)$	$f_3(V_t)$
Reflection	$\mid V \mid$	$\mid V \mid$	$\mid V \mid$
Partial Truncation	V	V	V^+
Full Truncation	V	V^+	V^+

Where $V^+ = \max(V, 0)$ and $\mid V \mid$ is the absolute value of V.

We chose to fix our discretization using the Full-Truncation (FT) scheme and thus,

rewrite the equations as follows:

$$S_{t+1} = S_t + rS_t + \sqrt{V_t^+} S_t Z_s \tag{2.6}$$

$$V_{t+1} = V_t + \kappa(\theta - V_t^+) + \sigma\sqrt{V_t^+} Z_v \tag{2.7}$$

Our R implementation follows the euler's scheme with hardly any modifications. It draws two Gaussian random variables (Z_v and Z_t) using the function *rnorm* to create Z_s with correlation ρ with Z_v.

```
Zv <- stats::rnorm(N)
Zt <- stats::rnorm(N)
Zs <- rho * Zv + (sqrt(1 - (rho^2)) * Zt)
```

And use it to compute S and V as in the code snippet bellow. The two modifications we apply are previously computing the square roots of the stepsize *dt* and the *aux* variable as to improve speed. The *aux* variable is an help variable created to impose positivity to V as we are operating the full-truncation euler scheme.

```
S <- S * (1 + r * dt + sqrt_aux * Zs * sqrt_dt)
v <- v + k * dt * (theta - aux) +
                    sigma * sqrt_aux * Zv * sqrt_dt
```

We could have used the *pmax* function in R, but this function is slow, therefore we opted to create the *aux* variable and impose positive values using the following R syntax:

```
aux <- v
aux[v < 0] <- 0
```

Depending on the reader's R fluency, other parts of the scheme might present a challenge, and that is why a version of the function is fully presented in appendix C.1.

2.3 Kahl-Jackel

Kahl-Jackel propose a discretization method they refer to as the "IJK" method [32,33] that coupled with the implicit Milstein scheme for the variance lands the system of equations (2.8) and (2.9). It is possible to verify that this discretization always results in positive paths for V if $4\kappa\theta > \sigma^2$. Unfortunately, this inequality is rarely satisfied when we plug real market data to calibrate the parameters. This means we must have a defined strategy for when the inequality doesn't hold. We use the scheme proposed in Andersen [32], where a truncation similar to the Euler's is applied. Whenever our volatility V_t drops below zero we use (2.7), and implement $\hat{V}(t+\Delta)^+$ and $\hat{V}(t)^+$ instead of $\hat{V}(t+\Delta)$ and $\hat{V}(t)$ of equation (2.8). The code guidance to this method

can be found in appendix C.2.

$$\ln \hat{S}(t+\Delta) = \ln \hat{S}(t) - \frac{\Delta}{4}\left(\hat{V}(t+\Delta) + \hat{V}(t)\right) + \rho\sqrt{\hat{V}(t)}Z_v\sqrt{\Delta} \tag{2.8}$$
$$+ \frac{1}{2}\left(\sqrt{\hat{V}(t+\Delta)} + \sqrt{\hat{V}(t)}\right)\left(Z_S\sqrt{\Delta} - \rho Z_V\sqrt{\Delta}\right) + \frac{1}{4}\sigma\rho\Delta\left(Z_V^2 - 1\right)$$

$$\hat{V}(t+\Delta) = \frac{\hat{V}(t) + \kappa\theta\Delta + \sigma\sqrt{\hat{V}(t)}Z_V\sqrt{\Delta} + \frac{1}{4}\sigma^2\Delta\left(Z_V^2 - 1\right)}{1+\kappa\Delta} \tag{2.9}$$

2.4 Exact Algorithm

In 2006, Broadie-Kaya [31] propose a method that has a faster convergence rate, $\mathcal{O}\left(s^{-1/2}\right)$ than some of the more famous schemes, such as Euler's and Milstein's discretizations. They build their idea to generate an exact sample from the distribution of the terminal stock price based on numerous papers [20]. The stock price and variance are as follows:

$$S_t = S_0\, exp\left[\mu t - \frac{1}{2}\int_0^t V_s ds + \rho\int_0^t \sqrt{V_s dB_s} + \sqrt{1-\rho^2}\int_0^t \sqrt{V_s}dW_s\right] \tag{2.10}$$

The squared volatility of the variance process is:

$$V_t = V_0 + \kappa\theta t - \kappa\int_0^t V_s ds + \sigma\int_0^t \sqrt{V_s dB_s} \tag{2.11}$$

The algorithm used to generate the model consists in four steps as follows:

Step 1. Generate a sample of V_t given V_0

Step 2. Generate a sample of $\int_0^t V_s ds$ given V_t, V_0

Step 3. Compute $displaystyle \int_0^t \sqrt{V_s}dB_s$ given V_t, V_0 and $\int_0^t V_s ds$

Step 4. Generate a sample from the probability distribution of S_t, given $displaystyle \int_0^t \sqrt{V_s}dB_s$ and $displaystyle \int_0^t V_s ds$

2.4.1 Generate a sample of V_t given V_0

The distribution of V_t given V_0 for $0 < t$ is a non-central chi-squared distribution [19,38]:

$$V_t = \frac{\sigma^2(1-e^{-\kappa t})}{4\kappa}\mathcal{X}_\delta^2\left(\frac{4\kappa e^{-\kappa t}}{\sigma^2(1-e^{-\kappa t})}\times V_0\right)$$

where $\delta = \frac{4\theta\kappa}{\sigma^2}$ and $\mathcal{X}_\delta^2(\lambda)$ denotes a non-central chi-squared random variable with δ degrees of freedom and λ as its non-centrality parameter.

Broadie and Kaya [31] sample generating Poisson and gamma distributions as in Johnson et al. [39]. We used the built-in function in R *rchisq*, which uses this exact method for sampling, see chunk below.

```
d1 <- (4 * k * theta)/(sigma)^2
c0 <- (sigma^2 * (1 - exp(-k*tau)))/(4*k)
dt <- (tau-t)

# sampling V
lambda <- (4*k*exp(-k*dt)*v)/(sigma^2 * (1-exp(-k*dt)))
vt <- c0 * stats::rchisq(n = 1, df = d1, ncp = lambda)
```

2.4.2 Generate a sample of $\int_0^t V_s ds$ given V_t, V_0

After generating V_t, we follow the instructions in [31,39]. We use the characteristic function (2.12) to compute the probability density function $F(x)$.

$$\begin{aligned}\Phi(a) &= \mathbb{E}\left[exp\left(ia\int_0^t V_s ds \mid V_0, V_t\right)\right] \\ &= \frac{\gamma(a)e^{(-1/2)(\gamma(a)-\kappa)t}(1-e^{-\kappa t})}{\kappa(1-e^{-\gamma(a)t})} \\ &\times exp\left\{\frac{V_0+V_t}{\sigma^2}\left[\frac{\kappa(1+e^{-\kappa t})}{1-e^{-\kappa t}} - \frac{\gamma(a)(1+e^{-\gamma(a)t})}{1-e^{-\gamma(a)t}}\right]\right\} \\ &\times \frac{I_{0.5\delta-1}\left[\sqrt{V_0V_t}\frac{4\gamma(a)e^{-0.5\gamma(a)t}}{\sigma^2(1-e^{-\gamma(a)t})}\right]}{I_{0.5\delta-1}\left[\sqrt{V_0V_t}\frac{4\kappa e^{-0.5\kappa t}}{\sigma^2(1-e^{-\kappa t})}\right]}\end{aligned} \tag{2.12}$$

where $\gamma(a) = \sqrt{\kappa^2 - 2\sigma^2 ia}$, δ was previously defined and $I_v(x)$ is the modified Bessel function of the first kind.[1] There are no mysteries implementing the characteristic function as you can observe in the chunk below. Although R has a built-in Bessel function, it only accounts for real numbers. Thus, we were obliged to use the Bessel package [40] that accounts for complex numbers. Once again, we pre compute some of the operations that are repeated through the function as to reduce computational time. This is specially important for this method since it involves operations of high complexity.

```
phi_heston <- function(a, v0, v_t, d){

gamma_a <- sqrt(k^2 - 2 * sigma^2 * 1i*a)
gammadt <- gamma_a * (tau-t)
sqrtv0vt <- sqrt(v0*v_t)
```

[1] *See Appendix refbessel for more information.*

```
  delta <- -k * (tau-t)

  part1 <- (gamma_a * exp(-(gamma_a - k)/2 * (tau-t)) * (1 - exp(delta)))/
           (k * (1- exp(- gammadt)))
  part2 <- exp((v0+v_t)/(sigma^2) * ( (k * (1 + exp(delta)))/(1-exp(delta)) -
           (gamma_a * (1 + exp(- gammadt)))/(1-exp(- gammadt))))
  part3 <- Bessel::BesselI(z = ((4 * gamma_a * sqrtv0vt)/(sigma^2) *
                           exp(- gammadt/2)/
                           (1 - exp(- gammadt))), nu = 0.5*d - 1) /
           Bessel::BesselI(z = ((4 * k * sqrtv0vt)/(sigma^2) *
                           (exp(delta/2))/ (1-exp(delta))), nu = 0.5*d - 1)
  return (part1 * part2 * part3)
}
```

The probability distribution function is obtained in [31,38] by Fourier inversions using Feller [41]. We use the approach in Gil-Pelaez [42], equation (2.13). We define $V(u,t)$ the random variable with the same distribution as the integral $\int_u^t V_s ds$, conditional on V_u and V_t:

$$F(x) \equiv Pr\left\{V(u,t) \leq x\right\} = F_X(x) = \frac{1}{2} - \frac{1}{\pi}\int_0^\infty \frac{\mathrm{Im}[e^{-iux} phi(u)]}{u}\, du \tag{2.13}$$

Im denotes the imaginary part of $e^{-iux} phi(u)$. Equation (2.13) is computed numerically and we then sample it by inversion. The integral function of the volatility is composed by five other functions inside of it. Since it acts only as a wrapper to theses functions we are going to omit it below, but the complete code can be found in appendix C.3.The first represents our integrand. The following function takes our integrand and actually performs the integration, returning the value. For speed purposes, we limited the integral upper bound to 1000 and increased the tolerance to 10^{-3}. After these calculations we have now the function's cumulative function.

```
  integrand <- function(x, phi = cf){

    f2 <- function(u){
      Im(phi(u) * exp(-1i * u * x)) /u
    }
    return(f2)
  }

  ## integrate to "cdf"
  F_x <- function (x) {
    y <- 0.5 - 1/pi * stats::integrate(integrand(x),  lower= 0, upper= 1000,
                                rel.tol = 0.001, stop.on.error = FALSE)$value
    return(y)
  }


  ## endsign
  endsign <- function(f, sign = 1) {
    b <- sign
    while (sign * f(b) < 0) b <- 10 * b
```

```
  return(b)
}
```

After computing the integral, we need to set the inversion function to place. To do that we generate the *endsign* function above. It is a simple function that is used to guarantee that when we call *uniroot*, the function to find roots built-in in R, our bounds have different signs. We set our lower and upper bounds as $-\infty$ and ∞, respectively and define and auxiliary function *subcdf* that will perform the inversion subtracting an uniform from it.

```
## inversion
low_bound = -Inf
upp_bound = Inf
invcdf <- function(u) {
  subcdf <- function(t) F_x(t) - u
  if (low_bound == -Inf)
    low_bound <- endsign(subcdf, -1)
  if (upp_bound == Inf)
    upp_bound <- endsign(subcdf)
  return(uniroot(subcdf, lower=low_bound, upper=upp_bound,
                 tol = 0.001220703)$root)
}
U <- stats::runif(n)
sapply(U, invcdf)
```

Furthermore, we also introduce (not in the same simulations, obviously) a simpler version for this step, that computes this integral approximation, using the solution $\int_u^t V_s ds = \frac{1}{2}(V_u + V_t)$. This solution is called *drift interpolation* [43] and its implementation is out of the box but we provide it anyway:

```
  int_v <- dt * ((1/2) * v + (1/2) * vt)
```

2.4.3 Compute $\int_0^t \sqrt{V_s} dB_s$ given V_t, V_0 and $\int_0^t V_s ds$

From equation (2.11) we are now able to compute this integral:

$$\int_0^t \sqrt{V_s} dB_s = \frac{V_t - V_0 - \kappa\theta t + \kappa \int_0^t V_s ds}{\sigma} \tag{2.14}$$

The last step of the algorithm consists of computing the conditional distribution of $logS_t$ based on the fact that the process for V_t is independent from dB_t, and the distribution of $\int_0^t \sqrt{V_s} dB_s$ is normal with mean 0 and variance $\int_0^t V_s ds$, given V_t.

$$m(u,t) = \log S_0 + \left[\mu t - \frac{1}{2}\int_0^t V_s ds + \rho \int_0^t \sqrt{V_s} dB_s + \sqrt{1-\rho^2} \int_0^t \sqrt{V_s} dW_s\right]$$

and variance

$$\sigma^2(0,t) = \left(1 - \rho^2\right) \int_0^t V_s ds$$

We generate the S_t sample using a standard normal random variable Z and set:

$$S_t = e^{m(0,t)+\sigma(0,t)Z}$$

In R code, we have the following:

```
int_vdw <- (1/sigma) * (vt - v - k * theta * dt + k  * int_v)

m <- log(S) + (r * (tau - t) - (1/2) * int_v + rho * int_vdw)
std <- sqrt((1 - rho^2)) * sqrt(int_v)
S <- exp(m + std * stats::rnorm(1))
```

2.4.4 Limitations

The biggest limitation this scheme presents is that the second step is computationally costly. It demands the inversion of the distribution function of $\left(\int_0^t V_s ds \mid V_t, V_0\right)$ numerically. We must perform a root search of $F(x_i) - U = 0$ testing for different x_i. Notwithstanding, we do not know the cummulative form of $F(x_i)$ distribution and have to perform our root finding strategy starting from the characteristic function, which contains two modified Bessel functions inside, in a structure that is rerun until a given tolerance ϵ is reached. Mathematically: $F(x_i) - U = \epsilon$.

–>

Chapter 3

Results

We present here the results of all the implementations that were disclosed in the previous section. We perform numerical comparisons between all the methods, setting out differences accross number of simulations and timesteps.

Heston [20] gives a closed form used for comparison as the 'true' option value and enabling the results to be exposed in terms of bias[1] and RMSE (root mean square error).[2]

The simulaton experiments were performed on a notebook with an Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz processor and 8GB of RAM running on a linux x86_64 based OS, Fedora 25. Codes were all written in R 3.4.1 "Single Candle" [2].

First of all, we chose a parametrization based on what we saw in other works, made some adjustments, like reducing the options' time to maturity due to the slower nature of R language and compared initial the results with the true option price given by the function *callHestoncf* belonging to the package NMOF [44]. Parameters can be seen in Table below.

Variables	Values
dt	0.05
k	2.00
r	0.05
rho	-0.30
S	100.00
sigma	0.20
t	0.00
tau	1.00
theta	0.09
v	0.09
X	100.00

Table 3.1: Model Parameters

[1] Defined as $\mathbb{E}\left[\hat{\alpha} - \alpha\right]$

[2] Defined as $\sqrt{\mathbb{E}((\hat{\theta} - \theta)^2)}$

To perform our simulations, we fixed a seed and saved the results in Table 3.4. Since the value given by the *callHestoncf* function with the parameters in Table 3.1 is 14.176, the method that best approached the "true" value was the modified (drift interpolated) exact algorithm with $100,000$ simulations. Altough the Euler scheme gives the same result (14.16) as the modified EA with $10,000$ simulations, it moves away from the closed form value when we run $100,000$ simulations. Results

Simulations	Euler	KJ	EA-BK	EA-DI
1,000	13.28	13.05	14.71	14.74
10,000	14.16	13.86	14.45	14.38
100,000	14.11	13.83	14.21	14.16

Note: Simulations performed with 20 steps, except the EA BK

Table 3.2: Results

are observable in Figure 3.1 also. The plot gives a good sense of possible biases associated with each method. To verify if in fact these methods present bias, we performed ten thousand simulations of ten thousand paths to the Euler, Kahl-Jackel (KJ) and drift interpolated exact algorithm (EA-DI). Clearly, from Figure 3.2, all three implementations produce bias (true option value is the black vertical line).

Simulations	Euler	KJ	EA-BK	EA-DI
1,000	0.01	0.01	58.56	0.02
10,000	0.05	0.05	572.83	0.04
100,000	0.46	0.45	5704.57	0.34

Note: Simulations performed with 20 steps, except the EA BK

Table 3.3: Computing time (sec.)

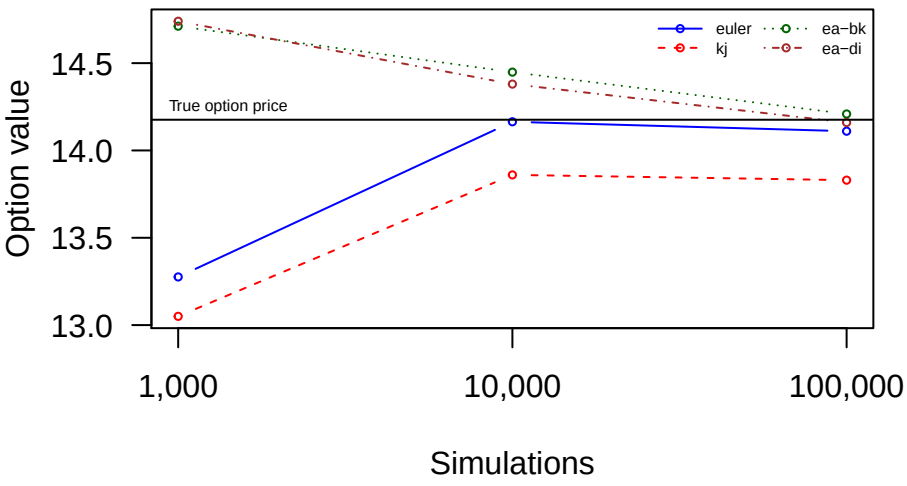

Figure 3.1: Comparison between models

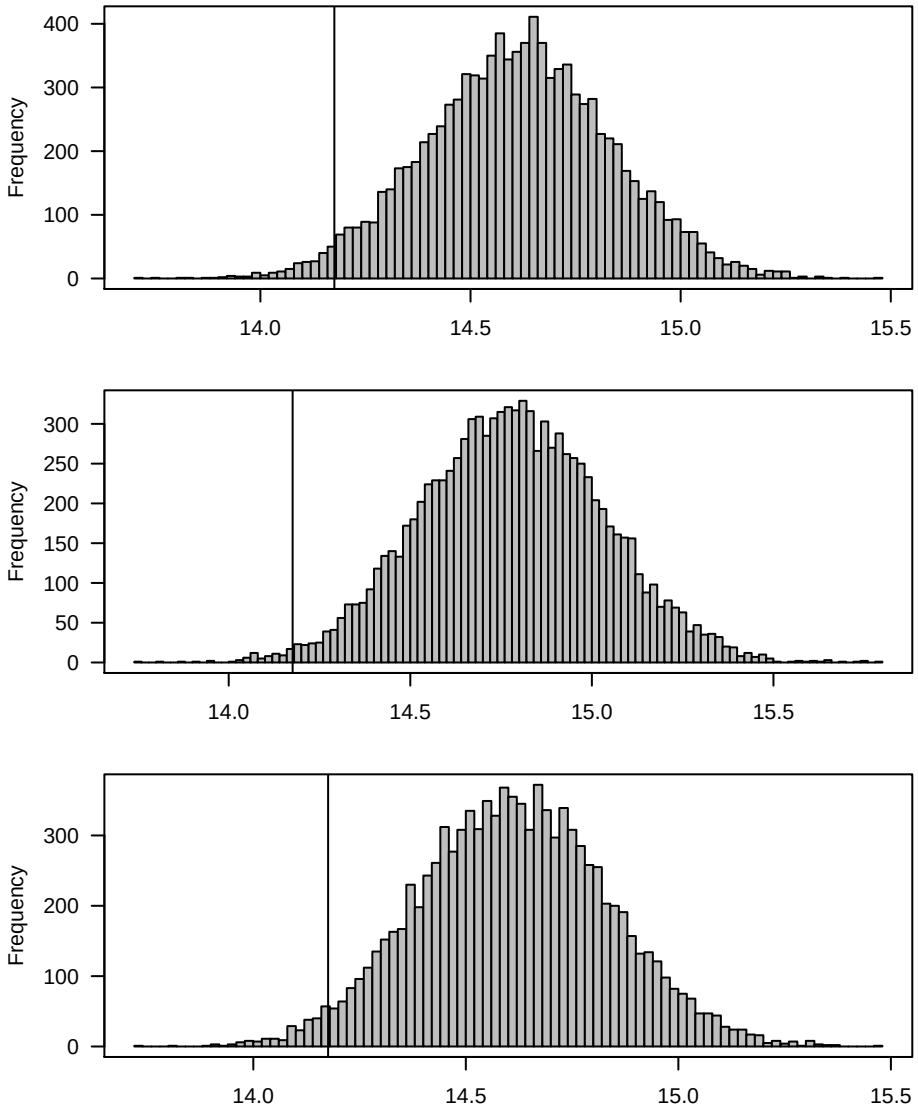

Figure 3.2: Comparison between models

	Euler	KJ	EA-BK	EA-DI
bias	0.44	0.60	0.44	0.00
sd	0.22	0.25	0.22	0.00
RMSE	0.49	0.65	0.49	0.00
time	1.87	15.59	13.27	0.00

Note: Simulations performed with 20 steps, except the EA BK

Table 3.4: Results

Conclusion

As we revisit what was presented through this thesis, we highlight that we were unable find codes of the heston's exact algorithm in any shape or form. That means we searched if the algorithm implementation was available in any programming language, or at least a reference to where to find it and we couldn't.

Firstly, we provided a succint, but rigorous, theoretical framework. Since this topic is very embracing, we provided an abundant amount of literature not only to the core topics approached, but also to the marginal ones.

Thereafter, we thoroughly introduced the algorithms we were going to implement in R, and how we were going to do it. Four different algorithms are presented: the Euler solution, the Kahl-Jackel algorithm, and two versions of what we (and literature) call the "exact algorithm". The first trying to be as similar as possible to the one presented in Broadie-Kaya's paper and the modified version bringing an approximation to the algorithm's second step (the costlier step) as we replace the integral with a drift interpolation.

RESULTS

Appendix A

Black-Scholes formula

In this appendix, we start from the following Geometric Brownian Motion process:

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

And we claim that the solution to this stochastic differential equation applying Itô's calculus is the following formula:

$$S_T = S_0 \times \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma B_T\right)$$

Proof. If S were deterministic, dS_t/S_t would be the derivative of $\ln(S_t)$ with respect to S. This suggests to find an expression for the stochastic differential of $\ln(S_t)$, a function of the single random variable S_t.

$$f(t,S) = \ln(S)$$

$$df(t,S) = \cancelto{0}{\frac{\partial f}{\partial t}}dt + \frac{\partial f}{\partial S}dS + \frac{1}{2}\cancelto{0}{\frac{\partial^2 f}{\partial t^2}}(dt)^2 + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}(dS)^2 + \cancelto{0}{\frac{\partial^2 f}{\partial t \partial S}dtdS}$$

$$d\ln(S) = \frac{d\ln(S)}{dS}dS + \frac{1}{2}\frac{d^2\ln(S)}{dS^2}(dS)^2$$

$$(dS)^2 = \int_0^t (\sigma \times S)^2 ds = \sigma^2 S^2 dt$$

$$d\ln(S) = \frac{1}{S}(\mu S dt + \sigma S dB) + \frac{1}{2}\frac{-1}{S^2}\sigma^2 S^2 dt$$

$$d\ln(S) = \left(\mu - \frac{1}{2}\sigma^2\right)dt + \sigma dB$$

$$\int_{t=0}^{T} d\ln(S_T) = \int_{t=0}^{T}\left(\mu - \frac{\sigma^2}{2}\right)dt + \int_{t=0}^{T}\sigma dB_t$$

$$\ln(S_T) - \ln(S_0) = \left(\mu - \frac{\sigma^2}{2}\right) T + \sigma B_T$$

$$\ln\left(\frac{S_T}{S_0}\right) = \left(\mu - \frac{\sigma^2}{2}\right) T + \sigma B_T$$

$$S_T = S_0 \times \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) T + \sigma B_T\right)$$

□

Appendix B

Bessel Function

The modified Bessel function of the first kind can be described in the shape of a contour integral (below) and is plotted for three different ν in Figure B.1.

$$I_n(z) = \frac{1}{2\pi i} \oint e^{(z/2)(t+1/t)} t^{-n-1} dt$$

However, the Broadie-Kaya paper [31] presents it as a power series:

$$I_\nu(z) = \left(\frac{1}{2}z\right)^\nu \sum_{k=0}^{\infty} \frac{\left(\frac{1}{4}z^2\right)^k}{(k!\Gamma\left(\nu + k + 1\right))}$$

With $\Gamma(x)$ is the gamma function and ν ia a complex number.

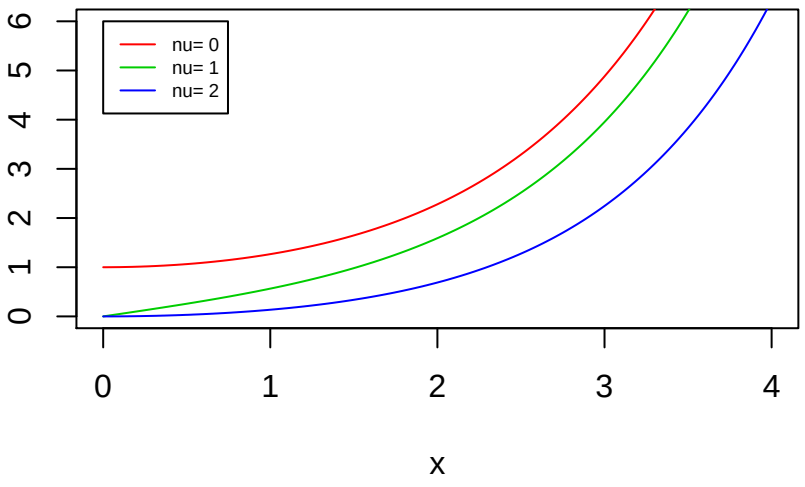

Figure B.1: Modified Bessel Functions of the First Kind

Appendix C

Implementations

C.1 Euler's

```
hestoneuler <- function(S, X, r, v, theta, rho, k,
                        sigma, t = 0, dt = NULL, tau = 1, N){

  if(is.null(dt)){ dt <- (tau-t)/1000}
  sequencia <- seq(t,tau,dt)
  ST <- matrix(NA, length(sequencia), N) #transformar em matrix
  aux <- NULL
  sqrt_dt <- sqrt(dt)

  for(i in sequencia){
    Zv <- stats::rnorm(N)
    Zt <- stats::rnorm(N)
    Zs <- rho * Zv + (sqrt(1 - (rho^2)) * Zt)
    aux <- v
    aux[v < 0] <- 0
    sqrt_aux <- sqrt(aux)
    S <- S * (1 + r * dt + sqrt_aux * Zs * sqrt_dt)
    S[S <= 0] = 0
    v <- v + k * dt * (theta - aux) + sigma * sqrt_aux * Zv * sqrt_dt
    ST[j,] <- S
  }
  rm(aux, v, Zv, Zt, Zs, S, j)
  ST <- as.matrix(ST, ncol=N)
  Result <- ST[nrow(ST),] - X
  Result[Result <= 0] = 0
  call = mean(exp(-r*(tau-t))*Result)
  lista = list('call' = call, 'Result' = Result, 'Spot' = ST)
  return(lista)
}
```

C.2 Kahl-Jackel Implementation

```
Hestoncallkj <- function(S, X, r, q, v, theta, rho, k,
                         sigma, t = 0, dt = NULL, tau = 1, N){

    if(is.null(dt)){ dt <- (T-t)/1000}
    v <- rep(v,N)
    theta<- rep(theta,N)
    sequencia <- seq(t,tau,dt)
    ST <- matrix(NA, length(sequencia), N) #transformar em matrix
    S <- log(S)

    for(i in seq(t,tau,dt)){
        Zv <- stats::rnorm(N)
        Zt <- stats::rnorm(N)
        Zs <- rho * Zv + sqrt(1 - rho^2) * Zt
        vt <- (v + k * theta * dt + sigma * sqrt(v) * Zv * sqrt(dt) +
                  (1/4) * sigma^2 * dt * ((Zv)^2 - 1))/(1 + k * dt)
        vt[vt <= 0] <- v[vt <= 0] + k * dt * (theta[vt <= 0] -
                       v[vt <= 0],0) + sigma * sqrt(v[vt <= 0],0) *
                       Zv[vt <= 0] * sqrt(dt)
        v <- vt
        v[v<=0] <- 0
        vt[vt<=0] <- 0
        S <- S + (r - (v+vt)/4) * dt + rho * sqrt(v) * Zv * sqrt(dt) +
             (1/2) * (sqrt(v) + sqrt(vt)) * (Zs + rho * Zv) * sqrt(dt) +
             ((rho * sigma * dt)/2) * ((Zv)^2 - 1)
        S[S <= 0] = 0
        ST[j,] <- S
    }
    ST <- as.matrix(ST, ncol=N)
    Result <- exp(ST[nrow(ST),]) - X
    Result[Result <= 0] = 0
    call = mean(exp(-r*tau)*Result)

    lista = list('call' = call, 'Result' = Result, 'Spot' = ST)
    return(lista)
}
```

C.3 Exact Algorithm

```
phi_heston <- function(a, v0, v_t, d){

    gamma_a <- sqrt(k^2 - 2 * sigma^2 * 1i*a)
    gammadt <- gamma_a * (tau-t)
    sqrtv0vt <- sqrt(v0*v_t)
    delta <- -k * (tau-t)

    part1 <- (gamma_a * exp(-(gamma_a - k)/2 * (tau-t)) * (1 - exp(delta)))/
        (k * (1- exp(- gammadt)))
```

```
    part2 <- exp((v0+v_t)/(sigma^2) * ( (k * (1 + exp(delta)))/(1-exp(delta)) -
                        (gamma_a * (1 + exp(- gammadt)))/(1-exp(- gammadt))))

    part3 <- Bessel::BesselI(z = ((4 * gamma_a * sqrtv0vt)/(sigma^2) *
                                      exp(- gammadt/2)/
                                      (1 - exp(- gammadt))), nu = 0.5*d - 1) /
        Bessel::BesselI(z = ((4 * k * sqrtv0vt)/(sigma^2) * (exp(delta/2))/
                                  (1-exp(delta))), nu = 0.5*d - 1)

    return (part1 * part2 * part3)
}



intv <- function(n, cf, v_t){
    integrand <- function(x, phi = cf){

        f2 <- function(u){
            Im(phi(u) * exp(-1i * u * x)) /u
        }
        return(f2)
    }

    ## integrate to "cdf"
    F_x <- function (x) {
        y <- 0.5 - 1/pi * integrate(integrand(x),  lower= 0, upper= 1000,
                                    rel.tol = 0.001, stop.on.error = FALSE)$value
        return(y)
    }

    ## endsign

    endsign <- function(f, sign = 1) {
        b <- sign
        while (sign * f(b) < 0) b <- 10 * b
        return(b)
    }

    ## inversion

    low_bound = -Inf
    upp_bound = Inf
    invcdf <- function(u) {
        subcdf <- function(t) F_x(t) - u
        if (low_bound == -Inf)
            low_bound <- endsign(subcdf, -1)
        if (upp_bound == Inf)
            upp_bound <- endsign(subcdf)
        return(uniroot(subcdf, lower=low_bound, upper=upp_bound,
                       tol = 0.001220703)$root)
    }
    U <- stats::runif(n)
    sapply(U, invcdf)
```

```
}

hestonea_mod <- function(S, X, r, v, theta, rho, k, sigma, t = 0, tau = 1){

    d1 <- (4 * k * theta)/(sigma)^2
    c0 <- (sigma^2 * (1 - exp(-k*tau)))/(4*k)
    dt <- (tau-t)
    ST <- NULL

    # sampling V

    lambda <- (4*k*exp(-k*dt)*v)/(sigma^2 * (1-exp(-k*dt)))
    vt <- c0 * stats::rchisq(n = 1, df = d1, ncp = lambda)

    # Sampling int{V}

    phi <- function(a, v0=v, v_t=vt, d=d1){phi_heston(a, v0=v, v_t=vt, d=d1)}

    int_v <- intv(1, cf = phi, v_t=vt)
    # OR if you perform the drift interpolation scheme:
    #  int_v <- dt * ((1/2) * v + (1/2) * vt)


    # Sampling int{v}dw
    int_vdw <- (1/sigma) * (vt - v - k * theta * dt + k  * int_v)

    # Sampling S
    if( int_v >= 0){
        m <- log(S) + (r * (tau - t) - (1/2) * int_v + rho * int_vdw)
        std <- sqrt((1 - rho^2)) * sqrt(int_v)
        S <- exp(m + std * rnorm(1))
        v <- vt
        ST <- S
    } else {
        v <- vt
        ST <- rbind(ST,NA)}

    Result <- ST - X
    Result[Result <= 0] = 0
    call = exp(-r*tau)*Result
    lista = list('call' = call, 'Result' = Result, 'Spot' = ST)
    return(lista)
}
```

References

[1] F. Black, M. Scholes, Journal of Political Economy 81 (1973) 637–654.

[2] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2017.

[3] U.F. Wiersema, Brownian Motion Calculus, John Wiley & Sons, 2008.

[4] I. Karatzas, S. Shreve, Brownian Motion and Stochastic Calculus, Springer Science & Business Media, 2012.

[5] A.D. Helgadóttir, L. Ionescu, (2016).

[6] L.C. Evans, An Introduction to Stochastic Differential Equations, American Mathematical Soc., 2012.

[7] J.S. Rosenthal, A First Look at Rigorous Probability Theory, World Scientific Publishing Co Inc, 2006.

[8] Z. Tong, Option Pricing with Long Memory Stochastic Volatility Models, PhD thesis, Université d'Ottawa/University of Ottawa, 2012.

[9] J.M. Steele, Stochastic Calculus and Financial Applications, Springer Science & Business Media, 2012.

[10] K. Itō, On Stochastic Differential Equations, American Mathematical Soc., 1951.

[11] K. Itô, Proc. Int. Congr. Math., Stockholm 2 (1962).

[12] Y. Yang, (2013).

[13] M. de F. Salomão, (2011).

[14] R.C. Merton, The Bell Journal of Economics and Management Science (1973) 141–183.

[15] R.S. Tsay, Analysis of Financial Time Series, John Wiley & Sons, 2005.

[16] S.M. Iacus, Simulation and Inference for Stochastic Differential Equations: With R Examples, Springer Science & Business Media, 2009.

[17] A. Krouglov, arXiv Preprint Physics/0612022 (2006).

[18] M. Gilli, D. Maringer, E. Schumann, Numerical Methods and Optimization in Finance, Academic Press, Waltham, MA, USA, 2011.

[19] J.C. Cox, J.E. Ingersoll Jr, S.A. Ross, Econometrica: Journal of the Econometric Society (1985) 385–407.

[20] S.L. Heston, Review of Financial Studies 6 (1993) 327–343.

[21] M. Gilli, D. Maringer, E. Schumann, Numerical Methods and Optimization in Finance, Academic Press, 2011.