Politecnico di Milano

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science in Mathematical Engineering

A Numerical Realization of the Wiener-Hopf Method for the Kolmogorov Backward Equation

Supervisor
Chiar.mo Prof. Daniele MARAZZINA

Candidate
Sonia JUGGOO – 892154

Academic Year 2019 – 2020

Acknowledgements

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale, il cui supporto è stato un elemento fondamentale e necessario per il raggiungimento di questo grande traguardo.

Un sentito grazie al mio relatore Daniele Marazzina per il supporto e i consigli. Grazie per avermi fornito ogni materiale utile alla stesura dell'elaborato.

Ringrazio col cuore i miei genitori per avermi supportato sia a livello materiale che emotivo durante tutto il percorso di studi che mi ha portato al raggiungimento dei miei obiettivi. In particolar modo, voglio ringraziare mia madre per avermi insegnato come vanno affrontati i problemi, per avermi spinto a dare sempre di più e per essere stata sempre il mio modello. Questa laurea non è un traguardo solo mio, ma anche suo.

Ringrazio mia sorella per essere sempre stata lì quando ne avevo bisogno. Perché mi ha aiutato a non impazzire, perché cucinava anche per me quando ero stressata per gli esami, e per avermi sempre incoraggiato durante il mio percorso accademico.

Un enorme grazie è per il mio fidanzato Simone. Non ci sono parole per descrivere il supporto morale e pratico che mi hai dato durante la preparazione degli ultimi esami, durante la ricerca del lavoro e la stesura della tesi. Grazie per avermi sopportato anche quando ero chiaramente insopportabile e per essere rimasto comunque al mio fianco.

Ringrazio con tutto il mio cuore Yuri per essermi stato sempre accanto. Perché anche quando tutto andava male lui riusciva sempre a trovare una soluzione. Per avermi aiutato nei momenti più bui. Per aver creduto in me anche quando io non ci credevo e per avermi spinto a provare gli esami anche quando non volevo. Per essere stato il mio airbag in qualsiasi occasione.

Grazie a Virginie. Abbiamo cominciato questo percorso accademico insieme e sin dal primo giorno ci siamo sempre supportate a vicenda. Grazie per essere stata mia compagna non solo di università ma anche di vita, per tutti i tuoi consigli, per tutte le passeggiate che abbiamo fatto e per i momenti belli che hanno reso questo difficile percorso molto più sopportabile.

Grazie a Luca per avermi supportato e soprattutto ascoltato sempre. Per essermi stato sempre vicino anche da lontano. Condividere con te i miei problemi li rendeva immediatamente più leggeri.

Grazie ad Alessandro, Gaia e Giovanni per tutto l'affetto che mi avete sempre dato, per essermi stati sempre vicini e per tutti i momenti bellissimi che mi avete fatto passare.

Grazie a JJ e Veronica per il grande supporto che mi hanno dato, per i consigli e per aver reso le mie giornate più felici.

Grazie a Momo, per essere stato sempre come un fratello. Grazie per tutto il supporto, i consigli e l'incoraggiamento che mi ha dato.

Grazie a Marta e Daniele per essere stati dei colleghi fantastici. Per tutti i nostri pranzi insieme al lavoro, le risate, le condivisioni e i consigli. Vorrei inoltre ringraziare Marta per avermi coperto quando durante il lavoro avevo bisogno di studiare, per essere stata un po' la mia valvola di sfogo e per aver fatto sì che la mia esperienza di stage fosse speciale.

Grazie a Valeria, Nicola, Federica, Francesco, Andrea, Giulio per essere stati degli ottimi amici. Per avermi aiutato a studiare, a capire meglio le lezioni e a preparare molti esami. Non avrei mai passato Analisi 1 senza l'aiuto di Nicola, né Numerica senza Valeria!

Grazie a tutti voi!

Contents

1 Introduction

The financial world used the Black-Scholes model (hereinafter 'BS', [4]) for decades to price derivative instruments until it has been proved that the model presents several drawbacks both from a statistical and a financial point of view. It is well known that:

- the density function of the logarithm of the stock process has fatter tails than a normal distribution and it is not really symmetric;
- BS ignores the correlation between the stock price and the volatity;
- BS does not explain the "volatility smile"observed in the market.

For these reasons, several other models were introduced in the literature. The Stochastic Volatility Models are models where another source of randomness is added, representing the market volatility as a stochastic process. The Heston model [25] belongs to this class and it assumes that the variance is driven by Cox-Ingersoll-Ross (CIR) process [21]. Pricing exotic option within the Heston model is quite complicated since it requires to solve the Kolmogorov backward equation with initial and boundary conditions depending on the derivative contract. Generally, to price this kind of contracts, for which there is not a closed formula, one may use one of the following numerical techniques:

- Monte Carlo, which is easy to implement, but sometimes computationally expensive in order to obtain accurate results.
- Finite Difference, which leads to good results with an efficient grid, but again computationally expensive.
- Hybrid method, whose advantages in terms of accuracy and convergence is discussed in [37].

The hybrid method consists in building an approximation for the variance process and then proceeding from a model with stochastic variance to a regime switching model (see [18], [12]).
The aim of this final work is to study a hybrid method presented by Oleg Kudryavtsev and Vasily Rodochenko in [37]. The new approach is used to solve some types of boundary value problems for 3-dimensional partial differential equations. Our analysis will be focused on the problem of pricing a first touch digital option, assuming that the stock and the variance process dynamics are the ones described by the Heston model. Considering the Kolmogorov backward equation, first we use first the Carr's time randomization; then, thanks to the Markov chain approximation of the variance, we

move from the original problem to a sequence of 1-dimensional differential equations where the variance is just a constant. Finally, the Wiener-Hopf factorization is used to determine an analytical solution for each problem.

2 Problem description

As mentioned above, the problem of pricing a financial instrument can be identified with the problem of solving a 3-dimensional partial differential equation with suitable boundary conditions. A 3-dimensional partial differential equation of a price function $u(t,x)$, with $x=(x_1,x_2)\in R^2$ and $t\in R$, looks as follows:

$$\left(\frac{\partial}{\partial t}+L\right)u=0\,, \tag{1}$$

where

$$L=\sum_i \mu_i(x)\frac{\partial}{\partial x_i}+\frac{1}{2}\sum_{i,j}(\sigma\sigma^T)_{i,j}(x)\frac{\partial^2}{\partial x_i\partial x_j}\qquad i=1,2\;\; j=1,2\,. \tag{2}$$

Given an absorbing barrier $H>0$, since we want to price a first touch digital option, our problem will be the following:

$$\begin{cases} \left(\frac{\partial}{\partial t}+L\right)u=0\,, & x_1<H, x_2\in R, t<T \\ \\ u(x_1,x_2,T)=0\,, & x_1<H, x_2\in R \\ \\ u(x_1,x_2,T)=1\,, & x_1\geq H, x_2\in R, t\leq T\,. \end{cases}$$

If we assume that x_1 represents the stock price and x_2 the variance process, the solution of the problem is the price of a first touch digital option. A first touch digital option is a financial instrument which pays 1 to its holder at the first time t in which the stock crosses the barrier. If the stock does not cross the barrier H before or at the maturity T, nothing will be paid. This kind of option are really important for risk management.

Theory proved that the problem of solving a PDE is equivalent to the one of computing the conditional expected value of a function of a stochastic process. In our case, we need to compute the conditional expectation of the indicator function for the event of the first entrance of two-dimensional stochastic process $X_t=(X_1(t);X_2(t))$ into the region $x_1\geq H$. The dynamics of the processes considered are the following:

$$\begin{aligned} dX_1(t)&=\mu_1 dt+\sigma_{1,1}dB_1(t)+\sigma_{1,2}dB_2(t)\,,\\ dX_2(t)&=\mu_2 dt+\sigma_{2,1}dB_1(t)+\sigma_{2,2}dB_2(t)\,, \end{aligned} \tag{3}$$

where $B_1(t)$ and $B_2(t)$ are two independent Brownian Motions.

Defining

$$T_H := inf\{t > 0 | X_1(t) \geq H\}\,,$$

we can write the solution function as follows:

$$u(x_1, x_2, 0) = P(T_H \leq T) = P(\overline{X}_1(T) \geq H)\,,$$

where

$$\overline{X}_1(t) = sup_{0 \leq s \leq t} X_1(s)\,.$$

Indeed, if for some time $t \leq T$, x_1 goes above the barrier H, for sure also the supremum value assumed by the process on the time interval $[0, T]$ is bigger or equal than the barrier.

2.1 Assumptions

We will assume that the drift function $\mu(x) = (\mu_1(x), \mu_2(x))$ and the diffusion function

$$\sigma(x) = \begin{pmatrix} \sigma_{1,1}(x) & \sigma_{1,2}(x) \\ \sigma_{2,1}(x) & \sigma_{2,2}(x) \end{pmatrix}$$

satisfy the below conditions from the existence and uniqueness theorem for stochastic differential equations [42].
Given $T > 0$, if the drift and the diffusion functions satisfy the following assumptions

- **Measurability** of $\mu : R^2 \to R^2$ and $\sigma : R^2 \to R^{2x2}$.
- **Sublinear grotwh**:
  $|\mu(x)| + |\sigma(x)| \leq C(1 + |x|)$ with $x \in R^2$ for some positive constant C.
- **Lipschitzianity**:
  $|\mu(x) - \mu(y)| + |\sigma(x) - \sigma(y)| \leq D|x - y|$, $x, y \in R^2$ for some positive constant D, where $|\sigma(x)|^2 = \sum \sigma_{i,j}^2$.

Then, given some initial conditions $X(0) = (X_1(0), X_2(0))$, there exists a unique solution $X(t) = (X_1(t), X_2(t))$ of (3).

2.2 Heston model

As mentioned before, $X_1(t)$ and $X_2(t)$ represent respectively the stock price and the variance process, whose dynamics, in this study, will be assumed to be the one described in the Heston model. Belonging to the class of stochastic volatility model, the Heston model assumes that the variance process is not constant but it is a stochastic process with its own source of randomness. The classical Heston assumes that the stock price is similar to a geometrical Brownian motion but with the hypothesis that the variance is a CIR process described by:

$$\begin{aligned} dS_t &= \mu_s S_t dt + \sqrt{V_t} S_t (\rho dB_1(t) + \hat{\rho} dB_2(t)\,, \\ dV_t &= k_v(\theta_v - V_t)dt + \sigma_v \sqrt{V_t} dB_1(t)\,. \end{aligned} \tag{4}$$

Notice that $S_t > 0$ and $V_t > 0$. If $V_0 > 0$ and the Feller condition $2k_v\theta_v \geq \sigma_v^2$ is satisfied, it could be shown that the variance process V_t is strictly positive.
The parameters in the above equations represent the following:

- μ is the rate of return of the asset.
- $\theta_v > 0$ is the long variance, or long run average price variance; as t tends to infinity, the expected value of V_t tends to 0.
- $k_v > 0$ is the rate at which V_t reverts to 0.
- $\sigma_v > 0$ is the volatility of the volatility, or "vol of vol", and determines the variance of v_t.
- $\rho \in (-1, 1)$ is the correlation between processes $(\rho dB_1(t) + \hat{\rho})dB_2(t)$ and $B_1(t)$, while $\hat{\rho} = \sqrt{1-\rho^2}$.

The features of the Heston model with respect to other stochastic volatility models are the following:

- it allows to reflect the correlation between the volatility and the asset price that we often observe in the market.
- It assumes that the volatility is a mean reverting process.

So, replacing $x_1(t) \in R$ and $x_2(t) \in R$ respectively with $S_t \in R^+$ and $V_t \in R^+$, and assuming the parameter $\mu_s = 0$, the drift and the diffusion functions become the following:

$$\mu = (0, k_v(\theta_v - V_t)) \,,$$

$$\sigma(S_t, V_t) = \begin{pmatrix} \rho\sqrt{V_t}S_t & \hat{\rho}\sqrt{V_t}S_t \\ \sigma_v\sqrt{V_t} & 0 \end{pmatrix} .$$

The operator L can be rewritten in terms of the function $u = u(S, v, t)$ as follows:

$$L = \frac{1}{2}S^2 v\frac{\partial^2}{\partial S^2} + \rho\sigma_v vS\frac{\partial^2}{\partial S\partial v}m + \frac{1}{2}\sigma^2 v\frac{\partial^2}{\partial v^2} + k_v(\theta_v - v)\frac{\partial}{\partial v} \,. \tag{5}$$

Thus the original problem becomes:

$$\begin{cases} \left(\frac{\partial}{\partial t} + L\right)u = 0 \,, & 0 < S < H, v \geq 0, t < T \\ u(x_1, x_2, T) = 0 \,, & 0 < S < H, v \geq 0 \\ u(x_1, x_2, T) = 1 \,, & S \geq H, v \geq 0, t \leq T \,, \end{cases} \tag{6}$$

and its solution at time $t = 0$:

$$u(S, v, 0) = E\Big[\mathbb{1}_{[0,T]}(T_H)|S_0 = S, V_0 = v\Big] \,, \tag{7}$$

where T_H is the first time that S_t crosses the barrier H.

3 General overview of the Hybrid method

In order to price a financial instrument, we often need to solve a 3-dimensional problem. As mentioned above, one can use different kind of techniques. Here we describe a hybrid method that basically, thanks to the discretization of one of the variables, consists in simplify the original more complicated problem into a sequence of easier problems.

The steps of our procedure are the following:

1. **Substitution**: in order to obtain a more convenient operator which does not contain mixed derivative, the first step will be the one to apply a change of variable.

2. **Carr's randomization**: it consists in replacing a deterministic maturity date T with a suitably chosen random maturity date whose mean is equal to T. When this random maturity is a sum of independent exponentially distributed maturity dates, the new pricing problem often reduces to a sequence of smaller problems, which are easier to solve.

3. **Markov chain approximation of the variance process**: we discretize the variance process, while the stock remains continuous. Since the expectation will be much easier to compute if V_t can be approximated with a constant value on a small time interval, the discretization is performed through a recombining tree.

4. **Wiener-Hopf factorization**: in order to solve the sequence of smaller problems with constant variance we use the Wiener-Hopf factorization technique.

Now we will enter into a more detailed description of each step.

3.1 Substitution

It is not convenient to keep studying our problem in terms of the variables S_t and V_t, mainly for three reasons:

- the mixed partial derivative term in the L operator makes the problem difficult to be solved from a numerical point of view;
- it is more convenient to better reflect the correlation between the price and the variance process;
- from a numerical point of view, it is better to move to a logarithmic scale and make a normalization with respect to the barrier H.

Thus, we introduce the process Y_t as

$$Y_t = ln\frac{S_t}{H} - \frac{\rho}{\sigma_v}V_t\,. \tag{8}$$

In these terms, we can re-define S_t as

$$S_t = H \cdot exp\left(Y_t + \frac{\rho}{\sigma_v}V_t\right)\,. \tag{9}$$

Moreover, by applying the Ito's formula to Y_t we can find its dynamics.
Then, from system (4) we move to:

$$\begin{aligned} dY_t &= \mu_Y(V_t)dt + \hat{\rho}\sqrt{V_t}dB_2(t) \\ dV_t &= \mu_V(V_t)dt + \sigma_v\sqrt{V_t}dB_1(t)\,, \end{aligned} \tag{10}$$

where the drift functions are the following:

$$\begin{aligned} \mu_Y(V_t) &= -\tfrac{1}{2}V_t - \tfrac{\rho}{\sigma_V}k_v(\theta_V - V_t) \\ \mu_V(V_t) &= k_v(\theta_V - V_t)\,. \end{aligned} \tag{11}$$

Notice that Y_t, V_t are driven by two independent Wiener processes.
After some calculations, the L operator becomes (see [37]):

$$L = \frac{1}{2}\hat{\rho}^2 v\frac{\partial^2}{\partial y^2} + \frac{1}{2}\sigma_V^2 v\frac{\partial^2}{\partial v^2} + \mu_Y(v)\frac{\partial}{\partial y} + \mu_V(v)\frac{\partial}{\partial v}\,. \tag{12}$$

As we can notice, there are no more mixed partial derivatives. In this way, the PDE will be easier to solve.

By defining a new function in terms of Y_t, V_t:

$$f(y,v,t) = u\left(H \cdot exp(y + \frac{\rho}{\sigma_v}v),\ \ v,\ \ t\right), \tag{13}$$

system (6) becomes

$$\begin{cases} \left(\frac{\partial}{\partial t} + L\right)u = 0\,, & y + \frac{\rho}{\sigma_v}v < 0, v \geq 0, t < T \\ u(x_1, x_2, T) = 0\,, & y + \frac{\rho}{\sigma_v}v < 0, v \geq 0 \\ u(x_1, x_2, T) = 1\,, & y + \frac{\rho}{\sigma_v}v \geq 0, v \geq 0, t \leq T\,. \end{cases} \tag{14}$$

Thus, at time $t = 0$ we can write

$$f(y,v,0) = E^{y,v}\left[\mathbb{1}_{[0,T]}(T_H)\right], \tag{15}$$

where $T_H = inf_{t\geq 0}\{t : Y_t + \frac{\rho}{\sigma_V}v \geq 0\}$ and $E^{y,v}[\cdot]$ means that we calculate an expectation conditioned on $Y_0 = y$ and $V_0 = v$.

Defining the process $Z_t = Y_t + \frac{\rho}{\sigma_V}V_t$ the solution can be written as

$$f(y,v,0) = E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_T})\right], \tag{16}$$

where $\overline{Z_t} = sup_{0\leq s\leq t}\{Z_s\}$.

4 Carr's randomization

In general, the randomization technique proposed by Carr in [16] describes a three-step procedure that can be used to solve a set of problems. The first step requires to substitute a parameter with a suitable random variable. In this case, suitable means that the random variable should have a plausible distribution, and then calculate the expected value in this random parameter setting. There are several possibilities for the parameter that needs to be randomized: the initial stock price, the strike price, the initial time, or the maturity date. In our analysis, we decided to randomize the maturity date of the financial contract.

We want to define a random variable T' that will substitute the maturity T. First, we discretize the time interval $[0, T]$ into $N \in \mathbf{N}$ smaller intervals of length $\Delta\tau = \frac{T}{N}$: set $t_i = i\Delta\tau$ for $i = 0, ..., N$.
The random variable is built as the sum of $N \in \mathbf{N}$ independent identically exponentially distributed variables with average $\Delta\tau$:

$$T' = \sum_{i=1}^{N} \tau_i \,, \quad \tau_i \sim \xi(\Delta\tau^{-1}) \quad \text{for } i = 1, ..., N \text{ i.i.d. and } \perp \text{ from } Y_t \text{ and } V_t \,.$$

Now, we have that:

$$T' \sim \Gamma(N, N/T) \,, \quad E[T'] = T \,, \quad Var[T'] = \tfrac{T^2}{N} \to 0 \text{ as } N \to \infty \,.$$

Instead of computing the exact solution:

$$f(y, v, 0) = E^{y,v}\left[\mathbb{1}_{[0,T]}(T_H)\right] = E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_T})\right] ,$$

we can now compute its approximation:

$$f_0(y, v) = E^{y,v}\left[\mathbb{1}_{[0,T']}(T_H)\right] = E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T'}})\right] .$$

The Post-Widder formula (explained in the next section) ensures that the approximating formula converges to the exact one and that the order of convergence is $O(N^{-1})$.

4.1 Post-Widder formula

The Post-Widder formula tells us the following (see e.g. [1]):

Considering a function of a non negative variable $f(T)$ and denoting:

- its Laplace transform as $\tilde{f}(q) = \int_0^\infty e^{-qT} f(T) dT$;
- the $N-th$ derivative of its Laplace transform as $\tilde{f}^{(N)}(q)$;

we have that:

$$f(T) = \lim_{N\to\infty} f_N(T)\,,$$

where

- $f_N(T) = \frac{(-1)^N}{N!}\left(\frac{N}{T}\right)^{N+1} \tilde{f}^{(N)}\left(\frac{N}{T}\right)$;
- the order of convergence is $O(N^{-1})$.

Considering our case, if we identify $f(T)$ with the exact solution $f(y, v, 0)$ (remembering that of course it can be seen also as a function of the non negative T), we have that $f_N(T)$ will corresponds exactly to the approximated solution $f_0(y, v)$ obtained via the Carr's randomization technique. To show this, we now compute $f_{N-1}(T)$ for the exact solution of our problem:

$$\begin{aligned}
f(T) \quad &\to \quad f(y,v,0) = E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_T})\right].\\
\tilde{f}(q) \quad &\to \quad \tilde{f}(q) = \int_0^\infty e^{-qT} E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_T})\right] dT\\
&\qquad\quad = q^{-1} E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T_q}})\right], \qquad\qquad \text{with } T_q \sim \xi(q)\,.\\
\tilde{f}^{(N-1)}(q) \to \quad &\partial_q^{n-1}\tilde{f}(q) = \partial_q^{n-1}\left(\int_0^\infty e^{-qT} E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_T})\right] dT\right)\\
&\qquad\quad = \int_0^\infty (-1)^{N-1} T^{N-1} e^{-qT} E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_T})\right] dT\,.
\end{aligned}$$

$$\begin{aligned}
f_{N-1}(T) \to \quad &\text{setting } q = N/T\ ,\\
&f_{N-1}(T) = \tfrac{(-1)^{N-1}}{(N-1)!} q^n \partial_q^{n-1}\tilde{f}(q)\\
&\qquad = \tfrac{q^n}{(n-1)!}\int_0^\infty T^{N-1} e^{-qT} E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_T})\right] dT\\
&\qquad = E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{\Gamma(n,q)}})\right] = f_0(y,v)\,.
\end{aligned}$$

So, the approximated solution of our problem converges to the exact one. Moreover the order of convergence is $O(N^{-1})$ (see [1]).

4.2 Recursive formula

Since we know that at the maturity date the option price will be given by the following deterministic function:

$$f_N(y,v) = \begin{cases} 0\,, & \text{if } y < -\frac{\rho}{\sigma_v}v \\ 1\,, & \text{if } y \geq -\frac{\rho}{\sigma_v}v \end{cases}\,. \tag{17}$$

Our objective will be the one to find a recursive backward procedure that from f_N moves to f_0. This procedure is reported in [37].
It is clear that $f_n(y,v)$ can be rewritten in terms of $f_{n+1}(y,v)$.

$$\begin{aligned} f_n(y,v) =& E\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_{n+1}}})\mathbb{1}_{[0,\infty)}(\overline{Z_{T'}})|Y_{T'_n} = y, V_{T'_n} = v\right] + \\ & E\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_{n+1}}})\mathbb{1}_{[0,\infty)}(\overline{Z_{T'}})|Y_{T'_n} = y, V_{T'_n} = v\right] = \\ & E\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_{n+1}}})|Y_{T'_n} = y, V_{T'_n} = v\right] + \\ & E\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_{n+1}}})f_{n+1}(Y_{T'_{n+1}}, V_{T'_{n+1}})|Y_{T'_n} = y, V_{T'_n} = v\right]. \end{aligned}$$

The reasons why the computations above make sense are the following:

- $\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_{n+1}}}) + \mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_{n+1}}}) = 1\,.$
- $\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_{n+1}}})\mathbb{1}_{[0,\infty)}(\overline{Z_{T'}}) = \mathbb{1}_{[0,\infty)}(\overline{Z_{T'_{n+1}}})$:

  because if $\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_{n+1}}}) = 1$ then $\mathbb{1}_{[0,\infty)}(\overline{Z_{T'}}) = 1$.

- $E\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_{n+1}}})\mathbb{1}_{[0,\infty)}(\overline{Z_{T'}})|Y_{T'_n} = y, V_{T'_n} = v\right] =$
  $E\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_{n+1}}})E\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T'}})|Y_{T'_{n+1}}, V_{T'_{n+1}}\right]|Y_{T'_n} = y, V_{T'_n} = v\right] =$
  $E\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_{n+1}}})f_{n+1}(Y_{T'_{n+1}}, V_{T'_{n+1}})|Y_{T'_n} = y, V_{T'_n} = v\right],$

  because of the tower property of the conditional expectation.

For a certain $l \in \mathbf{N}$ s.t $n + l \leq N$, the formula can be generalized as:

$$\begin{aligned} f_n(y,v) =& E\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_{n+l}}})|Y_{T'_n} = y, V_{T'_n} = v\right] + \\ & E\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_{n+l}}}) f_{n+l}(Y_{T'_{n+l}}, V_{T'_{n+l}})|Y_{T'_n} = y, V_{T'_n} = v\right]. \end{aligned}$$

Applying now the Markov property we obtain:

$$f_n(y,v) = E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{\tau_{n+1}}})\right] + E^{y,v}\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{\tau_{n+1}}}) f_{n+1}(Y_{\tau_{n+1}}, V_{\tau_{n+1}})\right],$$

and more in general

$$f_n(y,v) = E^{y,v}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_l}})\right] + E^{y,v}\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_l}}) f_{n+l}(Y_{T'_l}, V_{T'_l})\right],$$

where $T'_l = \sum_{i=1}^{l} \tau_i \sim \Gamma(l, N/T)$.

5 Variance tree

As already mentioned, in order to simplify the computations, we will discretize the variance process and, on a sufficiently small time interval, we will approximate it with a constant suitable value. We will use a lattice methodology and build a recombining tree. A recombining binomial tree is convenient from a computational point of view because of its fast convergence property and its ability to work within a large set of parameters.

5.1 General binomial recombining tree

Consider a generic Ornstein-Uhlenbeck Gaussian process:

$$dX_t = \beta(\alpha - X_t)dt + \sigma dW_t$$

where W_t is a Wiener process. Starting from the initial value of the process X_0, the tree is built in such a way that, from a given node, the process can jump up or down. The size of the jump is computed taking into consideration the diffusion term and the length of the time interval.

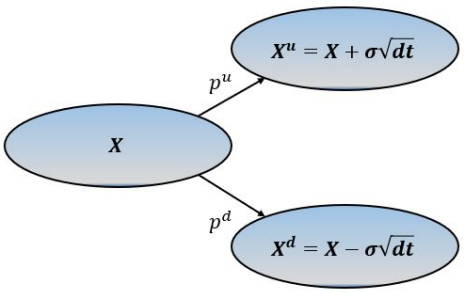

Figure 1. Example of one node of the tree with its possible next values Up or Down.

The probabilities to go up and down are computed in such a way that the average between the two nodes is equal to the expected future value of the process. The system we need to solve is the following:

$$\begin{cases} X + \beta\,(\alpha - X)\,dt = p^u X^u + p^d X^d \\ p^u + p^d = 1 \\ p^u, p^d \in [0,1] \end{cases} .$$

Thus, by iterating the procedure for each node, we obtain a tree of the following form:

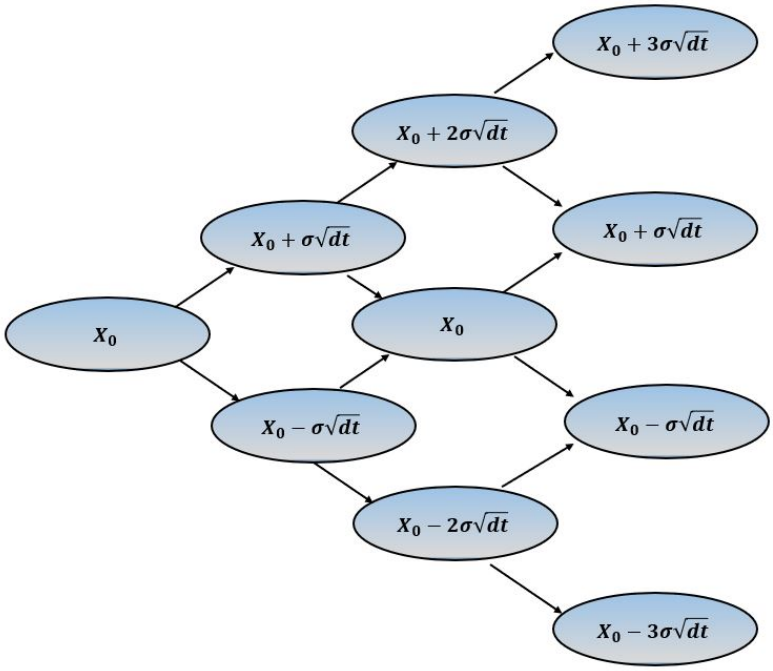

Figure 2. Binomial recombining tree for the O.U. process.

By using a shorter notation, we can write:

$$X(n', k) = X_0 + (2k - n')\sigma\sqrt{dt}\,,$$

where

- n' is the time index.
- $k = 0, ..., n'$ is the height of the node.

In our case, we will use a recombining tree in order to simulate possible paths for the square root of the stochastic variance process V_t. This is possible because the process defined by $\sqrt{V_t}$ has a constant diffusion, as shown in the next section. If we wanted to build a tree for the variance process V_t directly, that tree would not recombine because of its non-constant diffusion.

5.2 Binomial tree for the CIR process

We recall that we suppose that our variance process is a Cox–Ingersoll–Ross (CIR) process:

$$dV_t = k_v(\theta_v - V_t)dt + \sigma_v\sqrt{V_t}dB_1(t)\,.$$

Looking at the diffusion term, it is clear that we cannot build a recombining tree for the variance process. To deal with this problem, we will use a trick: we actually build the tree for the square root of the variance process. Indeed, if we apply the Ito's formula to $\sqrt{V_t}$ we obtain the following dynamics:

$$d\left(\sqrt{V_t}\right) = \frac{1}{2\sqrt{V_t}}\left(k_v(\theta_v - V_t) - \frac{1}{4}\sigma_V^2\right)dt + \frac{1}{2}\sigma_V dW_t\,. \tag{18}$$

Since the diffusion process is a constant value, we can build the tree as follows:

$$\sqrt{V}\,(n,k) = \sqrt{V_0} + (2k-n)\,\frac{\sigma_v}{2}\sqrt{dt}\,.$$

Finally, remembering that the variance allows just positive values, we set:

$$V(n,k) = \left(\sqrt{V_0} + (2k-n)\,\frac{\sigma_v}{2}\sqrt{dt}\right)^2 \cdot \mathbb{1}_{[0,\infty]}\left(\sqrt{V_0} + (2k-n)\,\frac{\sigma_v}{2}\sqrt{dt}\right)\,.$$

In our analysis, we will took into account the innovative concept from [47] and [28]. While for the general lattice tecnique transitions from $V_{n,k}$ are only allowed to $V_{n+1,k}$ and $V_{n+1,k+1}$, now we will choose the subsequent nodes from two different sets of plausible nodes. We define:

- the set of indexes of possible values for the subsequent "UP"node of $V_{n,k}$ as: $K_u(n,k) := \{k^* : k+1 \le k^* \le n+1, V_{n,k} + \mu_V\,(V_{n,k})\,dt \le V_{n+1,k^*}\}$;
- the set of indexes of possible values for the subsequent "DOWN"node of $V_{n,k}$ as: $K_d(n,k) := \{k^* : 0 \le k^* \le k, V_{n,k} + \mu_V\,(V_{n,k})\,dt \ge V_{n+1,k^*}\}$.

The basic idea is the one to consider as possible "UP"node each node with a value higher than the expected value of $V_{n,k}$ (i.e higher than $\mu_V\,(V_{n,k})\,dt$).
Similarly, the possible "DOWN"nodes are the one with value lower than the expected value of $V_{n,k}$.

Once the sets are defined, the subsequent nodes are chosen in the following way:

$$k_u(n,k) = \begin{cases} \min\{K_u(n,k)\} & \text{if } K_u(n,k) \neq \emptyset \\ n+1 & \text{if } K_u(n,k) = \emptyset \end{cases},$$

$$k_d(n,k) = \begin{cases} \max\{K_d(n,k)\} & \text{if } K_d(n,k) \neq \emptyset \\ 0 & \text{if } K_d(n,k) = \emptyset \end{cases}.$$

For instance, assuming to be in the node $V_{2,2}$ of the tree in Figure 3, we have $k_u(2,2) = 2$ and $k_d(2,2) = 1$.

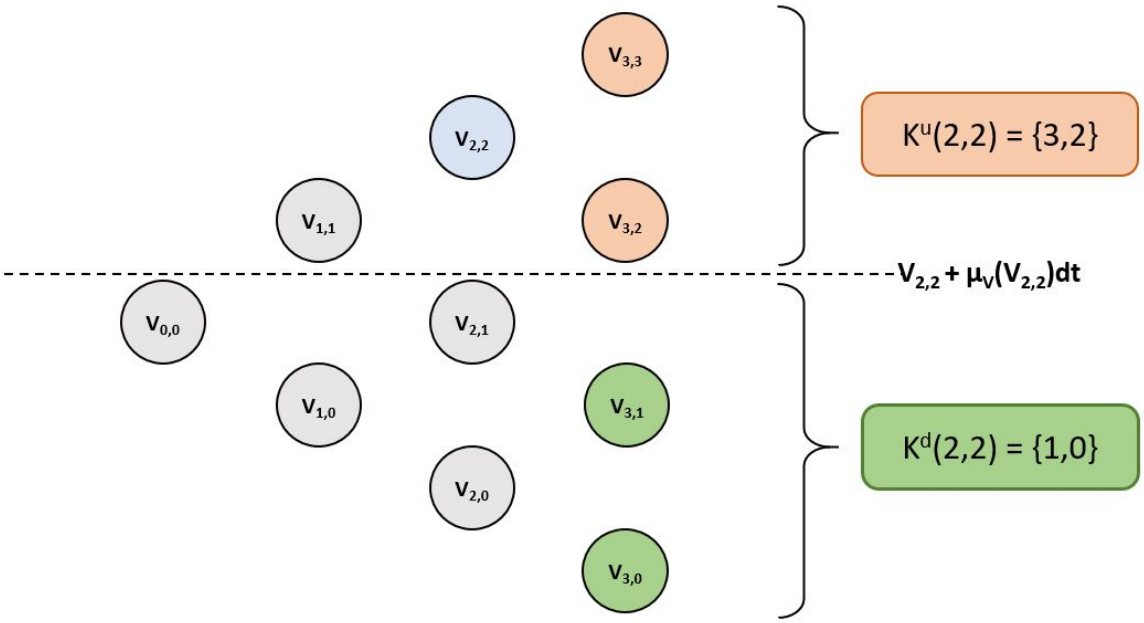

Figure 3. Example of $K_u(n,k)$ and $K_d(n,k)$ assuming we are at $V_{2,2}$.

After some computations, one can find that the transition probabilities are equal to the following:

$$p_u(n,k) = \frac{\mu_V\left(V_{n,k}\right)dt + V_{n,k} - V_{n+1,k_d(n,k)}}{V_{n+1,k_u(n,k)} - V_{n+1,k_d(n,k)}},$$

$$p_u(n,k) = 1 - p_d(n,k).$$

Then, to avoid the possibility to have infeasible values for the probabilities above, we

redefine them as:

$$p_u(n,k) = \begin{cases} 1 & \text{if } p_u(n,k) > 1 \\ p_u(n,k) & p_u(n,k) \in [0,1] \\ 0 & p_u(n,k) < 0 \end{cases},$$

$$p_d(n,k) = 1 - p_u(n,k)\,.$$

6 Pricing formula by using the Variance tree

Once the variance tree is built, the following step is the one to introduce this approximation in our pricing formula.

First of all, we notice that the convergence speed of the Variance process is higher than the one of the Post-Widder formula: this means that the grid of the Variance tree can be less dense than the one of the pricing function.
When we introduced the Carr's randomization technique, we substituted T with the random variable $T' = \sum_{i=1}^{N} \tau_i$, with τ_i identically exponential distributed random variable. Because of what we just mentioned, it is not necessary to build a variance tree where the variance is supposed to change for every τ_i . We will instead introduce the parameters $l \in \mathbf{N}$, and suppose that the variance process is the same for l consecutive τ_i .

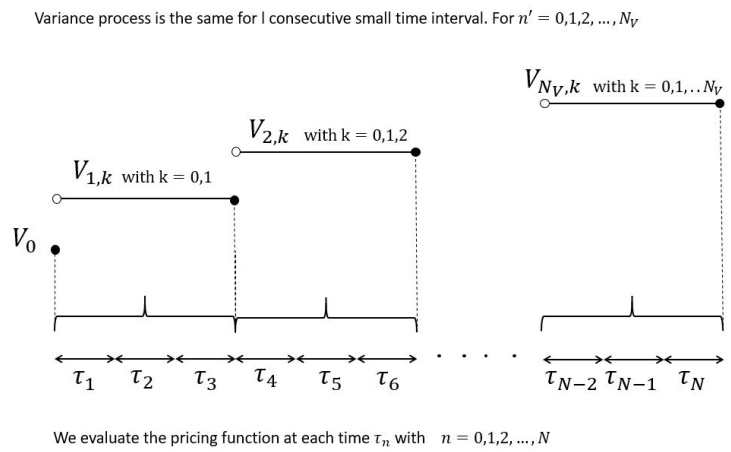

Figure 4. Example of variance discretization for l=3.

The nodes of the variance process for $n' = 0, 1, ...N_V$, $k = 0, 1, ...n'$ are:

$$V(n', k) = \left(\sqrt{V_0} + (2k - n')\,\frac{\sigma_v}{2}\sqrt{dt}\right)^2 \cdot \mathbb{1}_{[0,\infty]}\left(\sqrt{V_0} + (2k - n')\,\frac{\sigma_v}{2}\sqrt{dt}\right) ,$$

where the time discretization is $dt = T/N_v$. Once the process $V_{n',k}$, we define the following processes:

- $Y_t^{n',k}$ as Y_t with coefficients generated by $V_{n',k}$.
- $Z_t^{n',k} = Y_t^{n',k} + \frac{\rho}{\sigma} \cdot V_{n',k}$.

Notice that in terms of indexes, we can say that the variance process jumps at each time $T'_{n'l}$ with $n' = 0, 1, ..., N_V$ (i.e $T'_3, T'_6, ...$).

We can compute the pricing function at each time in which the variance process jumps as:

$$\begin{aligned} f_{n'l}(y, V_{n',k}) =& E^{y,V_{n',k}}\left[\mathbb{1}_{[0,\infty)}(\overline{Z_{T'_l}})\right] + E^{y,V_{n',k}}\left[\mathbb{1}_{(-\infty,0)}(\overline{Z_{T'_l}}) f_{n'l+l}(Y_{T'_l}, V_{T'_l})\right] \\ &= p_u f^u_{n',k}(y) + p_d f^d_{n',k}(y)\,, \end{aligned}$$

where[1]

$$\begin{aligned} f^u_{n',k}(y) =& E^y\left[\mathbb{1}_{[0,\infty)}(\overline{Y}^{n'+1,k_u}_{T'_l} + \frac{\rho}{\sigma_V} V_{n'+1,k_u})\right] + \\ &+ E^y\left[\mathbb{1}_{(-\infty,0)}(\overline{Y}^{n'+1,k_u}_{T'_l} + \frac{\rho}{\sigma_V} V_{n'+1,k_u}) f_{n'l+l}(Y^{n'+1,k_u}_{T'_l}, V_{n'+1,k_u})\right]. \end{aligned}$$

$V_{n'+1,k_u}$ is the upper subsequent node of $V_{n',k}$. $f^d_{n',k}(y)$ is defined by analogy. Thus, when we compute $f^u_{n',k}(y)$ and $f^d_{n',k}(y)$, we assume the variance to be respectively equal to $V_{n'+1,k_u}$ and $V_{n'+1,k_d}$. This assumption will simplify a lot the computation of the expectation, that now is a function of just one random variable. $f^u_{n',k}(y)$ and $f^d_{n',k}(y)$ are calculated in a recursively way.

For $m = l-1, l-2, ..., 0$ we define:

$$\begin{aligned} \omega^u_{m,k}(y) = E^y\left[\mathbb{1}_{[0,\infty)}(\overline{Y}^{n'+1,k_u}_{\tau'_{m+1}} + \frac{\rho}{\sigma_V} V_{n'+1,k_u})\right] + \qquad (19) \\ + E^y\left[\mathbb{1}_{(-\infty,0)}\left(\overline{Y}^{n'+1,k_u}_{\tau_{m+1}} + \frac{\rho}{\sigma_V} V_{n'+1,k_u}\right) \omega^u_{m+1,k}\left(Y^{n'+1,k_u}_{\tau_{m+1}}\right)\right]. \end{aligned}$$

$\omega^u_{0,k}(y)$ approximates $f^u_{n',k}(y)$, while $\omega^u_{l,k}(y) = f_{n'l+l}(y, V_{n'+1,k_u})$. Similarly we can compute $f^d_{n',k}(y)$.

[1] This formula differs from the one in [38]. Indeed, in [38], the two expected values are multiplied by some indicator functions which were taken into consideration by mistake.

7 Wiener-Hopf factorization approach

The main instrument we will use to compute expectations in (19) is the Wiener-Hopf factorization method. This formula is able to compute the expectation of functions of the supremum/infimum process of some Lévy processes. Now we will explain in details this methodology which finds many of applications in the financial world.

7.1 Preliminary concepts about Lévy processes

Given a general Lévy process X_t, we define its characteristic function as

$$\phi_{X_t}(u) = E^y\left[e^{i\cdot u X_t}\right]. \tag{20}$$

It is well known that the characteristic function of a Lévy process satisfies the following properties.

- **Multiplicative property**:

  $$\phi_{X_{t+s}}(u) = \phi_{X_t}(u)\phi_{X_s}(u)\,.$$

  This results comes from the fact that a Lévy process has independent increments

- **Characteristic exponent existence**: it exists a function $\psi : \mathbb{R} \to \mathbb{C}$ such that the characteristic function can be written as:

  $$\phi_{X_t}(u) = e^{-t\psi(u)}\,,$$

  and $\psi(\cdot)$ does not depend on the time t.

Notice that the second property is really important because it affirms that it is enough to compute just once the characteristic exponent, and then obtain the characteristic function for every time t.

The characteristic exponent of our process $Y_t^{n',k}$ is defined as follows:

$$\phi(u) = \frac{\sigma_{n',k}^2}{2}u^2 - i\gamma_{n',k}u\,,$$

where $\sigma_{n',k} = \hat{\rho}\sqrt{V_{n',k}}$ and $\gamma_{n',k} = \mu_Y(V_{n',k})$.
This will be used to apply the Wiener-Hopf factorization and compute expectations in (19).

7.2 General technique

The Wiener-Hopf factorization method can be presented in several different ways. Here we present it from a probability point of view.

Let us consider a Gaussian process X_t (the procedure can be extended to a general Lévy) and an independent exponentially distributed random variable $\tau \sim \xi(q)$. We define the infimum and supremum of the Lévy process as:

$$\overline{X}_t = inf_{0\leq s\leq t} X_s, \qquad \overline{X}_t = sup_{0\leq s\leq t} X_s$$

The Wiener-Hopf factorization identity used in probability is the following (see details in [13], [14] and [33]):

$$E\left[e^{i\xi X_\tau}\right] = E\left[e^{i\xi \overline{X}_\tau}\right] E\left[e^{i\xi \overline{X}_\tau}\right] \qquad \forall \xi \in \mathbb{R} \tag{21}$$

This result comes from the useful formula: $\overline{X}_t \sim X_t - \overline{X}_t$. Now we can introduce the following notation:

$$\phi_q^+(\xi) = E\left[e^{i\xi \overline{X}_\tau}\right] = qE\left[\int_0^\infty e^{-qt} e^{i\xi \overline{X}_t} dt\right] \tag{22}$$

$$\phi_q^-(\xi) = E\left[e^{i\xi \overline{X}_\tau}\right] = qE\left[\int_0^\infty e^{-qt} e^{i\xi \overline{X}_t} dt\right] \tag{23}$$

Since we have that:

$$E\left[e^{i\xi X_\tau}\right] = q\int_0^\infty e^{-qt} E\left[e^{i\xi X_t}\right] dt$$

$$= q\int_0^\infty e^{-t\left(q+\psi(\xi)\right)} dt$$

$$= \frac{q}{q+\psi(\xi)}$$

identity (21) can be rewritten as:

$$\frac{q}{q+\psi(\xi)} = \phi_q^+(\xi)\phi_q^-(\xi) \tag{24}$$

The functions $\frac{q}{q+\psi(\xi)}, \phi_q^+(\xi), \phi_q^-(\xi)$ will be used for computing the Expected Present Value operators ε_q, ε_q^+, ε_q^- (EPVs) introduced in [13].

Let us define the EPVs operator as follows:

$$\varepsilon_q g(x) = E\left[\int_0^\infty q e^{-qt} g(X_t) dt | X_0 = x\right] = E\left[g(x + X_\tau)\right], \tag{25}$$

$$\varepsilon_q^+ g(x) = E\left[\int_0^\infty q e^{-qt} g(\overline{X}_t) dt | X_0 = x\right] = E\left[g(x + \overline{X}_\tau)\right], \tag{26}$$

$$\varepsilon_q^- g(x) = E\left[\int_0^\infty q e^{-qt} g(\overline{X}_t) dt | X_0 = x\right] = E\left[g(x + \overline{X}_\tau)\right]. \tag{27}$$

Remember that τ is an exponential distributed random variable with mean q^{-1}.

Pseudo-differential operators' theory tells us that, in an appropriate functional spaces, the following factorization holds:

$$\varepsilon_q = \varepsilon_q^+ \varepsilon_q^- = \varepsilon_q^- \varepsilon_q^+ .$$

The operators for the infimum and supremum can be computes as:

$$\left(\varepsilon_q^\pm f\right)(x) = \frac{1}{2\pi} \int_{-\infty}^\infty e^{ix\xi} \phi^\pm(\xi) \hat{f}(\xi) d\xi , \tag{28}$$

where

$$\hat{f}(\xi) = \int_{-\infty}^\infty e^{-ix\xi} f(x) dx .$$

For the process $Y_t^{n',k}$, we have the following formulae:

$$\phi_q^+(\xi) = \frac{\beta_{n',k}^+}{\beta_{n',k}^+ - i\xi} , \qquad \phi_q^-(\xi) = \frac{\beta_{n',k}^-}{\beta_{n',k}^- - i\xi} ,$$

where

$$\beta_{n',k}^- = -\frac{\gamma_{n',k} + \sqrt{\gamma_{n',k}^2 + 2\sigma_{n',k}^2 q}}{\sigma_{n',k}^2} ,$$
$$\beta_{n',k}^+ = -\frac{\gamma_{n',k} - \sqrt{\gamma_{n',k}^2 + 2\sigma_{n',k}^2 q}}{\sigma_{n',k}^2} .$$

If we consider a Lévy process which admits jumps, there are some approximated formulae in order to compute the factors introduced in [34].

7.3 Wiener-Hopf factorization applied to the pricing function

The last step in our procedure is to compute the functions $\omega^u_{m,k}$ and $\omega^d_{m,k}$. To do this we will use the Wiener-Hopf factorization formula. We recap that $\omega^u_{m,k}$ and $\omega^d_{m,k}$ are defined as follows:

$$\omega^u_{m,k}(y) = E^y\left[\mathbb{1}_{[0,\infty)}(\overline{Y}^{n'+1,k_u}_{\tau'_{m+1}} + \frac{\rho}{\sigma_V}V_{n'+1,k_u})\right] + \tag{29}$$

$$+E^y\left[\mathbb{1}_{(-\infty,0)}\left(\overline{Y}^{n'+1,k_u}_{\tau_{m+1}} + \frac{\rho}{\sigma_V}V_{n'+1,k_u}\right)\omega^u_{m+1,k}\left(Y^{n'+1,k_u}_{\tau_{m+1}}\right)\right].$$

We study separately the two addends.

1. The first part is an EPV operator for the infimum process of Y_t:

$$E^y\left[\mathbb{1}_{[0.\infty)}(\overline{Y}^{n'+1,k_u}_{\tau'_{m+1}} + \tfrac{\rho}{\sigma_V}V_{n'+1,k_u})\right] = \varepsilon^+_q\mathbb{1}_{[0,\infty)}(y + \tfrac{\rho}{\sigma_V}V_{n'+1,k_u}).$$

2. The second part is more complicated. First of all we will use the fact that $Y^{n'+1,k_u}_{\tau_{m+1}}$ and $\overline{Y}^{n'+1,k_u}_{\tau_{m+1}} + \underline{Y}^{n'+1,k_u}_{\tau_{m+1}}$ are identical in law. This will allow us to perform the following steps:

$$\begin{aligned}\omega^u_{m+1,k}\left(Y^{n'+1,k_u}_{\tau_{m+1}}\right) &= \omega^u_{m+1,k}\left(y + \underline{Y}^{n'+1,k_u}_{\tau_{m+1}} + \overline{Y}^{n'+1,k_u}_{\tau_{m+1}}\right)\\ &= \varepsilon^-_q\omega^u_{m+1,k}\left(\overline{Y}^{n'+1,k_u}_{\tau_{m+1}}\right),\end{aligned}$$

so that the second addend becomes:

$$\begin{aligned}&E^y\left[\mathbb{1}_{(-\infty,0)}\left(\overline{Y}^{n'+1,k_u}_{\tau_{m+1}} + \frac{\rho}{\sigma_V}V_{n'+1,k_u}\right)\omega^u_{m+1,k}\left(Y^{n'+1,k_u}_{\tau_{m+1}}\right)\right] =\\ &E^y\left[\mathbb{1}_{(-\infty,0)}\left(\overline{Y}^{n'+1,k_u}_{\tau_{m+1}} + \frac{\rho}{\sigma_V}V_{n'+1,k_u}\right)\varepsilon^-_q\omega^u_{m+1,k}\left(\overline{Y}^{n'+1,k_u}_{\tau_{m+1}}\right)\right] =\\ &\qquad\varepsilon^+_q\left(\mathbb{1}_{(-\infty,-\frac{\rho}{\sigma_V}V_{n'+1,k_u})}(\cdot)\,\varepsilon^+_q\omega^u_{m+1,k}(\cdot)\right)(y).\end{aligned}$$

Finally, we consider the expectation as a function of the infimum operator and putting everything together formula we have:

$$\omega^u_{m,k}(y) = \varepsilon^+_q\mathbb{1}_{[0,\infty)}(y + \frac{\rho}{\sigma_V}V_{n'+1,k_u}) + \varepsilon^+_q\left(\mathbb{1}_{(-\infty,-\frac{\rho}{\sigma_V}V_{n'+1,k_u})}(\cdot)\,\varepsilon^-_q\omega^u_{m+1,k}(\cdot)\right)(y).$$

Similar computations can be done for $\omega^d_{m,k}(y)$.

Notice that analogous passages can be done to find the price of a first touch digital option whose payoff is paid when $S_t \leq H$. If we supposed to me at maturity time the payoff will be:

$$f_N(y,v) = \begin{cases} 0 & \text{if } y > -\frac{\rho}{\sigma_v}v \\ 1 & \text{if } y \leq -\frac{\rho}{\sigma_v}v \end{cases} . \tag{30}$$

Then, in this case we have that

$$T_H = inf_{t\geq 0}\{t : S_t \leq H\} = inf_{t\geq 0}\{t : Y_t + \tfrac{\rho}{\sigma_V}v \leq 0\},$$

and the approximated function changes as follows:

$$\begin{aligned} f_n(y,v) =& E^{y,v}\left[\mathbb{1}_{(-\infty,0]}(\underline{Z_{T'_{n+1}}})\mathbb{1}_{(-\infty,0]}(\underline{Z_{T'}})\right] + E^{y,v}\left[\mathbb{1}_{(0,+\infty)}(\underline{Z_{T'_{n+1}}})\mathbb{1}_{(-\infty,0]}(\underline{Z_{T'}})\right] \\ =& \quad E^{y,v}\left[\mathbb{1}_{(-\infty,0]}(\underline{Z_{T'_{n+1}}})\right] + E^{y,v}\left[\mathbb{1}_{(0,\infty)}(\underline{Z_{T'_{n+1}}})f_{n+1}(Y_{T'_{n+1}}, V_{T'_{n+1}})\right]. \end{aligned}$$

Finally this will be the ω function in case of an upper barrier:

$$\omega^u_{m,k}(y) = \varepsilon^-_q \mathbb{1}_{(-\infty,0]}(y + \frac{\rho}{\sigma_V}V_{n'+1,k_u}) + \varepsilon^-_q\left(\mathbb{1}_{(-\frac{\rho}{\sigma_V}V_{n'+1,k_u},\infty)}(\cdot)\,\varepsilon^+_q\omega^u_{m+1,k}(\cdot)\right)(y).$$

8 Numerical results

In this section we report the numerical results obtained using the Matlab algorithm that is attached in the appendix.
The results obtained with the hybrid method described above have been compared with the ones obtained via the Monte Carlo method with antithetic variables. Notice that, to compute the EPVs, we used the functions *fft* and *ifft* provided by Matlab: this allowed us to reduce the time consumed and to increase the efficiency. For the Y process, an equally spaced grid of M points on the interval $[-L \cdot ln(2), L \cdot ln(2)]$ as suggested in [37]. Thanks to the same mentioned paper, we know that longer is the time period considered, higher is the number of iterations required from the method to converge, while for short time period it provides good and faster results. Moreover, due to the structure of the Y process, for larger value of $\frac{\rho}{\sigma_v}$, it may be necessary to increase L.

8.1 Study case

We consider the problem of pricing a first touch digital option with upper barrier with the following input parameters:

T	V_0	κ	θ	σ	ρ	H
0.5	0.1	2	0.1	0.2	-0.3	100

Figure 5. Study case: parameters of the Heston model and the first touch digital option contract.

When pricing the contract with Monte Carlo, we used $N_{sim} = 10^4$ number of simulations and $M = 180$ time steps, while with the hybrid method we set $L = 2$, $M = 2^{14}$, $l = 5$, $N_v = 30$. The results are the following:

S_0	Hybrid method	CI, 95% level
60	0.008638	[0.008481204, 0.011218796]
65	0.025247	[0.025713441, 0.030286559]
70	0.061478	[0.060802736, 0.067597264]
75	0.127973	[0.125191048, 0.134508952]
80	0.232726	[0.226793765, 0.238506235]
85	0.376683	[0.368887806, 0.382312194]
90	0.551336	[0.541051829, 0.554848171]
95	0.739739	[0.742335201, 0.754364799]

Figure 6. Study case: Hybrid method prices and MC confidence interval.

As it is shown in the graph, the prices obtained with hybrid method are within the 95%-confidence interval of the Monte Carlo method:

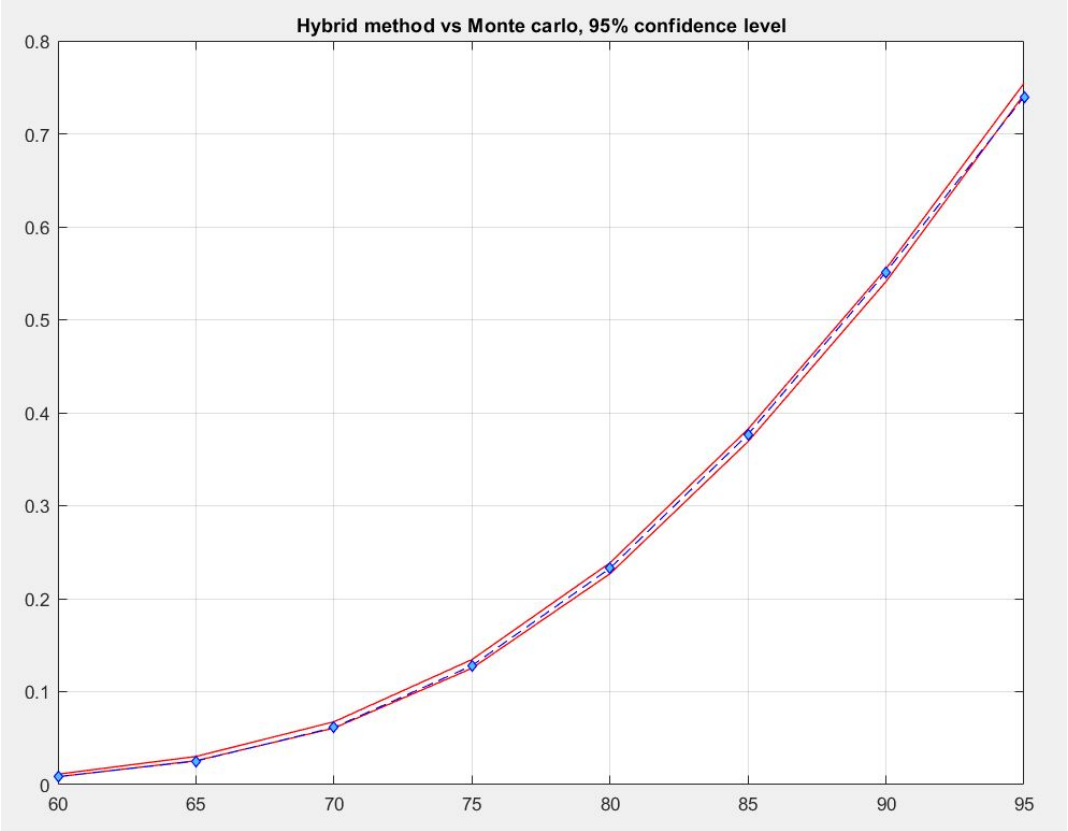

Figure 7. Pricing results for different values of S_0.

9 Appendix

9.1 Run file

```
clear all
close all
clc

%% set the default seed
rng('default')

%% Input
T = 0.5; V0 = 0.1; k = 2; theta = 0.1; sigma = 0.2; rho =
    -0.3;
S0 = 60:5:95; H=100;
%% Monte Carlo

% Number of simulations
Nsim =10^4;
% Number of time steps
MMC =180;
% Confidence Interval at (1-alpha)
alpha =0.05;

% Initialize variable
PriceMC = zeros(length(S0),1);
CI = zeros(length(S0),2);
elapsedTimeMC = zeros(length(S0),1);

% Compute the price using MC
for i=1:length(S0)
        tic
        [PriceMC(i),CI(i,:)] = computePriceMC_AV(S0(i), V0, H
            ,T, rho,k, theta,sigma,Nsim,MMC,alpha);
        elapsedTimeMC(i)= toc;
end

%% Hybrid method
```

```
33
34 % payoff is paid when St >= H
35 flag = 1;
36
37 % Variance Tree time steps
38 Nv = 30;
39
40 % Time steps for the Y process
41 l = 5;
42 Ny = Nv*l;
43
44 % L: parameter used to compute the extreme points of the grid
       for the Y process
45 L=2;
46 % M: number of points in gridY
47 M=2^10;
48
49 % Compute the price using the hybrid method
50 tic
51 PriceHybrid= HybridMethod(k,theta,sigma,rho,l,Nv,Ny,T,V0,S0,H
       ,L,M,flag);
52 elapsedTimeHybrid = toc ;
53
54 %% Plot
55 plotResults
```

9.2 Monte Carlo price function

This function compute the price via Monte Carlo.

```
1 function [PriceMC,CI] = computePriceMC_AV(S0, V0, H,T,rho,k,
      theta,sigma,Nsim,M,alpha)
2 %INPUT
3 % S0 : Intial value of the stock process
4 % V0 : Intial value of the variance process
5 % H : barrier value
6 % T : maturity
7 % rho : correlation coefficient
8 % k : speed of regression of variance process
```

```
% theta : long-run average value
% sigma : volatility of variance
% Nsim : number of simulation
% M : Number of time step
% alpha : desired confidence interval for the price
%
%OUTPUT
% PriceMC: price for an upper/lower barrier using MC
%           upper barrier --> payoff paid when St>=H
%           lower barrier --> payoff paid when St<=H
% CI: Confidence interval at (1-alpha) level

%compute rho hat
rhoHat = sqrt(1-rho^2);

% Initialize stock and variance process
St = zeros(Nsim,M+1);
St(:,1)=S0;
St_AV = St;
Vt = zeros(Nsim,M+1);
Vt(:,1)=V0;
Vt_AV=Vt;

%compute time step
dt = T/M;

% Simulate stock and variance process
rng('default')
for  i=2:M+1
    %simulate first brownian motion
    B1 = randn(Nsim,1);
    %simulate second brownian motion
    B2 = randn(Nsim,1);

    %Simulate variance process
    Vt(:,i)= Vt(:,i-1)+k*(theta-Vt(:,i-1))*dt+...
             sigma*(Vt(:,i-1).^(1/2)).*B1*sqrt(dt);

```

```
47      Vt_AV(:,i)= Vt_AV(:,i-1)+k*(theta-Vt_AV(:,i-1))*dt+...
48                  sigma*(Vt_AV(:,i-1).^(1/2)).*(-B1)*sqrt(dt);
49
50      %Simulate stock process
51      St(:,i)=St(:,i-1)+(Vt(:,i).^(1/2)).*St(:,i-1).*(rho*B1+
            rhoHat*B2)*sqrt(dt);
52      St_AV(:,i)=St_AV(:,i-1)+(Vt_AV(:,i).^(1/2)).*St_AV(:,i-1)
            .*(rho*(-B1)+rhoHat*(-B2))*sqrt(dt);
53
54  end
55
56
57
58
59  if S0 >= H
60     %compute price for a lower barrier
61      StMin = min(St');
62      StMin_AV = min(St_AV');
63      payOff = (StMin<=H); % payoff paid when St <=H
64      payOff_AV = (StMin_AV<=H);
65      [PriceMC,~,CI] = normfit([payOff,payOff_AV],alpha);
66
67     else
68     %compute price for an upper barrier
69      StMax = max(St');
70      StMax_AV = max(St_AV');
71      payOff = (StMax >=H);
72      payOff_AV = (StMax_AV >=H);% payoff paid when St >=H
73      [PriceMC,~,CI] = normfit([payOff,payOff_AV],alpha);
74  end
75
76
77
78  end
```

9.3 Hybrid method function

This function compute the price via the hybrid method. First we build the variance tree, and then we start the backward procedure.

```
function PriceHybrid = HybridMethod(k,theta,sigma,rho,l,Nv,Ny
    ,T,V0,S0,H,L,M,flag)
% INPUT
% k : speed of regression of variance process
% theta : long-run average value
% sigma : volatility of variance
% rho : correlation coefficient
% l : parameter of time steps of Y process
% Nv : number of time steps of the variance tree
% Ny : number of time steps of the Y process
% T : maturity of the digital option
% V0: initial value of the variance process
% S0: initial value(s) of the stock process
% H: barrier value
% L: parameter used for compute the extreme points of the
    grid for the Y process
% M: number of points in gridY
% flag: 0 if the payoff is paid when St <= H,
%       1 if the payoff is paid when St >= H
%
%OUTPUT
% PriceHybrid: price of the digital option for every value on
     S0


%% Compute Variance tree
TreeV = computeVarianceTree(V0,Nv,sigma,T);

%% Define the grid for the X and Y process
% gridY : grid for the Y process, used for the Fourier/
    Inverse Fourier transforms
% gridX : grid used for the integral of the EPV
[gridX,gridY]=computeGrid(M,L);

```

```
%% Compute price value for every point in gridY
if flag == 1
    % Case with Up Barrier: payoff paid when St >=H
    PriceVec = computePriceHybrid(k,theta,sigma,rho, TreeV, l
        ,Nv,Ny,T,gridY,gridX, flag);
else
    % Case with Low Barrier: payoff paid when St <=H
    PriceVec = computePriceHybrid(k,theta,sigma,rho, TreeV, l
        ,Nv,Ny,T,gridY,gridX, flag);
end

%% Interpolate for each value in S0
Y0 = log(S0/H)-rho*V0/sigma;
PriceHybrid = interp1(gridY,real(PriceVec),Y0,'spline');


end
```

9.4 Build Variance Tree

To build the standard variance tree we use the following function:

```
function [TreeV] = computeVarianceTree(V0,Nv,sigma,T)
% INPUT
% V0: variance value at t=0
% Nv: number of time steps of the variance tree
% sigma : volatility of variance
% T : maturity
%
% OUTPUT
% TreeV: Upper triangular matrix representing the variance
    tree


%Initialize Variance tree
TreeV = zeros(Nv+1,Nv+1);
TreeV(1,1)= V0;

for i=2:Nv+1
```

```
17
18      %node of variance tree
19      aux = 2*[i-1:-1:0]' - (i-1);
20
21      %compute subsequent nodes
22      TreeV(1:i,i) = (max(0,sqrt(V0)+ 0.5*sigma*sqrt(T/Nv)*aux)
            ).^2;
23
24  end
25
26
27
28
29
30
31
32
33  end
```

Then, to find the subsequent nodes (as it was described above), we use the following code:

```
1  function [vU, vD, pU, pD] = find_successors_on_tree(presentV,
       dt, futureVectorV,k ,theta)
2  % INPUT
3  % presentV: scalar value representing the value of the
       current variance
4  % dt: time step
5  % futureVecorV: vector representing possible values of
       variance after time dt
6  % k : speed of regression of variance process
7  % theta : long-run average value
8  % OUTPUT
9  % vU : upper future node of the variance tree
10 % vD : lower future node of the variance tree
11 % pU : probability to go up
12 % pD : probability to go down
13
14 %drift function of the variance process
```

```
15 driftV = @(v)k*(theta-v);
16
17
18 %probabilities function
19 auxpU = @(v, vU, vD, dt) (v + driftV(v)*dt - vD)/(vU- vD);
20 prob = @(p)p*(p>=0 & p<=1) + (p>1);
21
22 %compute expected future value
23 expfutureV = presentV + driftV(presentV)*dt;
24
25 %initialize the sets of possible future values with not
      feasible values
26 set_kU = -100;
27 set_kD = -100;
28
29 for i=1: length(futureVectorV)
30     if expfutureV <= futureVectorV(i)
31         set_kU = [set_kU; futureVectorV(i)];
32     end
33     if expfutureV > futureVectorV(i)
34         set_kD = [set_kD; futureVectorV(i)];
35     end
36 end
37
38 % compute upper node
39 if length(set_kU)== 1
40     vU = futureVectorV(1);
41 else
42     vU = min(set_kU(2:end));
43 end
44
45 % compute lower node
46 if length(set_kD)== 1
47     vD = futureVectorV(end);
48 else
49     vD = max(set_kD(2:end));
50 end
51
```

```
52
53 %compute probability to go up
54 pU = auxpU(presentV, vU, vD, dt);
55 %correct the probability if needed
56 pU =prob(pU);
57
58 pD = 1 - pU;
59
60 end
```

9.5 Hybrid price function

```
1  function price = computePriceHybrid(k,theta,sigma,rho, TreeV,
       l,Nv,Ny,T,gridY,gridX,flag)
2  % INPUT
3  % k : speed of regression of variance process
4  % theta : long-run average value
5  % sigma : volatility of variance
6  % rho : correlation coefficient
7  % TreeV : Upper triangular matrix representing the variance
       tree
8  % l : parameter of time steps of Y process
9  % Nv : number of time steps of the variance tree
10 % Ny : number of time steps of the Y process
11 % T : maturity of the digital option
12 % gridY : grid for the Y process
13 % gridX : grid used for the integral of the EPV
14 % flag: 0 if the payoff is paid when St <= H,
15 %       1 if the payoff is paid when St >= H
16 %
17 %OUTPUT
18 % price: price computed with the hybrid method for every
       point in gridY
19
20
21 %Initialize Matrix of vector
22 % vecMatrix is an array matrix with same dimension of the
       variance tree
```

```
23 % It is our price tree
24 % Notice that each cell contains a vector with values for
       each y in gridY
25 vecMatrix = cell(Nv+1,Nv+1);
26
27 % Define payoff function at maturity
28 if flag == 0
29     final = @(y,v) y<= -rho*v/sigma; % if the payoff is paid
           when St <= H
30 else
31     final = @(y,v) y>= -rho*v/sigma; % if the payoff is paid
           when St >= H
32 end
33
34 % compute price for the final nodes of the tree (i.e at
       maturity)
35 for i=1:Nv+1
36     %approximate the variance with the value on the tree
37     finalV = @(y) final(y,TreeV(i,Nv+1));
38
39     %compute price for every value on gridY
40     auxVec = finalV(gridY');
41
42     %save the vector on the corresponding cell
43     vecMatrix{i,Nv+1}= auxVec;
44 end
45
46
47 %BACKWARD PROCEDURE ON THE TREE
48 for j=Nv:-1:1
49
50     % compute all subsequent nodes of the tree
51     futureVectorV = TreeV(1:j+1,j+1);
52
53     for i=1:j
54             % save the current variance value
55             v = TreeV(i,j);
56
```

```
57                % compute the upper and lower nodes of v
58                [vU, vD, pU, pD] = find_successors_on_tree(v, T/
                      Nv, futureVectorV,k,theta);
59
60                % find the corresponding index of the upper and
                      lower nodes of v
61                idxU = find(futureVectorV==vU);
62                idxD = find(futureVectorV==vD);
63
64                % find the corresponding upper and lower nodes of
                       prices
65                wUp = vecMatrix{idxU,j+1};
66                wDown = vecMatrix{idxD,j+1};
67
68                %Backward procedure for l time step while we
                      assume variance equal to vU
69                fU = auxFunction(v,vU,wUp,gridY,gridX, T, Nv, Ny,
                      k,theta,sigma,rho, l,flag);
70                %Backward procedure for l time step while we
                      assume variance equal to vD
71                fD = auxFunction(v,vD,wDown,gridY,gridX, T, Nv,
                      Ny,k,theta,sigma,rho, l,flag);
72
73                %compute vector of prices for the current node
74                vecMatrix{i,j}= fU*pU+fD*pD;
75            end
76        end
77
78        % Compute price vector at t=0
79        price = vecMatrix{1,1};
80        end
```

computePriceHybrid.m calls the *auxFunction.m* to compute the ω function.

```
1   function output = auxFunction(v,v1, w_fut,gridY,gridX, T, Nv,
        Ny,k,theta,sigma,rho, l,flag)
2   % INPUT
3   % v : current variance
4   % v1 : variance value on the subsequent node (up or down
```

```
    subsequent node)
% w_fut: vector of price for the subsequent node (up or down
    subsequent node)
% gridY : grid for the Y process
% gridX : grid used for the integral of the EPV
% T : maturity of the digital option
% Nv : number of time steps of the variance tree
% Ny : number of time steps of the Y process
% k : speed of regression of variance process
% theta : long-run average value
% sigma : volatility of variance
% rho : correlation coefficient
% TreeV : Upper triangular matrix representing the variance
    tree
% l : parameter of time steps of Y process
% flag: 0 if the payoff is paid when St <= H,
%       1 if the payoff is paid when St >= H
%
%OUTPUT
% output: vector of price for the current variance assuming
    the future
%         variance equal to v1

%% Initialize the current value of w
    w_cur=w_fut;
%% Define the time step of the Y process
    q=Ny/T;

  %% Backward procedure for l time step --> if payoff is paid
      when St <=H
   if flag == 0
            %compute first part of w function
            auxf1 = @(y)((y+rho*v1/sigma)<=0);
            EPV_inf_auxf1 =  computeInfEPV(auxf1(gridY'),
                gridY,gridX,q,v1,rho,k,theta,sigma);
            for i=l-1:-1:2
                %compute second part of w function
                EPV_Sup_w_fut = computeSupEPV(w_cur,gridY,
```

```
                    gridX,q,v1,rho,k,theta,sigma);
37                  auxf2=  (gridY'>=-rho*v1/sigma).*
                    EPV_Sup_w_fut;
38                  EPV_inf_auxf2 = computeInfEPV(auxf2,gridY,
                    gridX,q,v1,rho,k,theta,sigma);
39                  %compute w function
40                  w_cur  =  EPV_inf_auxf1 +EPV_inf_auxf2;
41              end
42              % Last case for l=1
43              %compute first part of w function
44              auxf1 = @(y)((y+rho*v/sigma)<=0);
45              EPV_inf_auxf1 =  computeInfEPV(auxf1(gridY'),
                gridY,gridX,q,v,rho,k,theta,sigma);
46              %compute second part of w function
47              EPV_Sup_w_fut = computeSupEPV(w_cur,gridY,gridX,q
                ,v,rho,k,theta,sigma);
48              auxf2=  (gridY'>=-rho*v/sigma).* EPV_Sup_w_fut;
49              EPV_inf_auxf2 = computeInfEPV(auxf2,gridY,gridX,q
                ,v,rho,k,theta,sigma);
50              %compute w function
51              w_cur  =  EPV_inf_auxf1 +EPV_inf_auxf2;
52          else
53          %% Backward procedure for l time step --> if payoff is paid
                when St >=H
54              %compute first part of w function
55              auxf1 = @(y)((y+rho*v1/sigma)>=0);
56              EPV_Sup_auxf1 =  computeSupEPV(auxf1(gridY'),
                gridY,gridX,q,v1,rho,k,theta,sigma);
57              for i=l-1:-1:2
58                  %compute second part of w function
59                  EPV_Inf_w_fut = computeInfEPV(w_cur,gridY,
                    gridX,q,v1,rho,k,theta,sigma);
60                  auxf2=  (gridY'<=-rho*v1/sigma).*
                    EPV_Inf_w_fut;
61                  EPV_Sup_auxf2 = computeSupEPV(auxf2,gridY,
                    gridX,q,v1,rho,k,theta,sigma);
62                  %compute w function
63                  w_cur  =  EPV_Sup_auxf1 + EPV_Sup_auxf2;
```

```
64                 end
65
66                 %Last case for l=1
67                 auxf1 = @(y)((y+rho*v/sigma)>=0);
68                 EPV_Sup_auxf1 =  computeSupEPV(auxf1(gridY'),
                       gridY,gridX,q,v,rho,k,theta,sigma);
69                 %compute second part of w function
70                       EPV_Inf_w_fut = computeInfEPV(w_cur,gridY,
                           gridX,q,v,rho,k,theta,sigma);
71                       auxf2=  (gridY'<=-rho*v/sigma).*
                           EPV_Inf_w_fut;
72                       EPV_Sup_auxf2 = computeSupEPV(auxf2,gridY,
                           gridX,q,v,rho,k,theta,sigma);
73                       w_cur  =  EPV_Sup_auxf1 + EPV_Sup_auxf2;
74
75                 output = w_cur;
76     end
77
78
79
80
81 end
```

9.6 Build Grids

```
1  function [gridX,gridY]=computeGrid(M,L)
2  % INPUT
3  % M: number of nodes for each grid
4  % L: extremes for the gridY interval are -L*log(2) and +L*log
       (2)
5  %
6  % OUTPUT
7  % gridY : grid for the Y process
8  % gridX : grid used for the integral of the EPV
9
10 gridY = linspace(-L*log(2),L*log(2),M);
11 dY=gridY(2)-gridY(1);
12 dX= 2*pi/(M*dY);
```

```
gridX = [-M*dX/2:dX:M*dX/2];
gridX = gridX(1:end-1);

end
```

9.7 Compute EPV under the Supremum

```
function EPVsup = computeSupEPV(fVector,gridY,gridX, q,v,rho,
    k,theta,sigma)
% INPUT
% fVector:vector of function f evaluated at grid Y
% gridY : grid for the Y process
% q : parameter of the EPV --> 1/q = E[tau],
%                                 where tau is an exponential r
    .v representing the time discretization
% v : variance value ( assumed to be constant)
% rho : correlation coefficient
% k : speed of regression of variance process
% theta : long-run average value
% sigma : volatility of variance
%
% OUTPUT
% EPVsup :  Expected Present Value for the f fucntion of the
    Sup process, evaluated at gridY


%drift function of the Y process
driftY = @(v)-0.5*v-rho*k*(theta-v)/sigma;

%compute phiPlus function
gamma = driftY(v);
sigmat = sqrt(1-rho^2)*sqrt(v);
betaPlus = -( gamma - sqrt(gamma^2+2*sigmat^2*q) )/sigmat^2;
phiPlus =@(x) betaPlus./(betaPlus-1i*x);

%compute Fourier transform of f for every point in gridX
fHat = FourierTransform(fVector,gridY,gridX);

```

```
29 %Compute Expected Present Value of the Sup process at gridY
30 auxVector = phiPlus(gridX).'.*fHat;
31 EPVsup =InverseFourierTransform(auxVector,gridX,gridY);
32
33 end
```

9.8 Compute EPV under the Infimum

```
1  function EPVinf = computeInfEPV(fVector,gridY,gridX, q,v,rho,
       k,theta,sigma)
2  % INPUT
3  % fVector:vector of function f evaluated at grid Y
4  % gridY : grid for the Y process
5  % gridX : grid used for the integral of the EPV
6  % q : parameter of the EPV --> 1/q = E[tau],
7  %                                where tau is an exponential r
       .v representing the time discretization
8  % v : variance value (assumed to be constant)
9  % rho : correlation coefficient
10 % k : speed of regression of variance process
11 % theta : long-run average value
12 % sigma : volatility of variance
13 %
14 % OUTPUT
15 % EPVinf :  Expected Present Value for the f fucntion of the
       Inf process, evaluated at gridY
16
17
18 %drift function of the Y process
19 driftY = @(x)-0.5*x-rho*k*(theta-x)/sigma;
20
21 %compute phiMinus function
22 gamma = driftY(v);
23 sigmat = sqrt(1-rho^2)*sqrt(v);
24 betaMinus = -( gamma + sqrt(gamma^2+2*sigmat^2*q) )/sigmat^2;
25 phiMinus = @(x) -betaMinus./(-betaMinus+1i*x);
26
27 %compute Fourier transform of f for every point in gridX
```

```
28 fHat =FourierTransform(fVector,gridY,gridX);
29
30 %Compute Expected Present Value of the inf process at gridY
31 auxVector = phiMinus(gridX).'.*fHat;
32 EPVinf =InverseFourierTransform(auxVector,gridX,gridY);
33
34
35 end
```

9.9 Fourier Transform

```
1  function fHat = FourierTransform(fVector,gridY,gridX)
2  % INPUT
3  % fVector: vector of function f evaluated at gridY
4  % gridY: grid of Y process
5  % gridX : grid used for the integral of the EPV
6  %
7  %OUTPUT
8  % fHat: vector of the fourier transform of f evaluated at
       grid X
9
10 %compute fourier transform of f
11 Nx=length(gridX);
12 dY = gridY(2)-gridY(1);
13 dX = gridX(2)-gridX(1);
14 Y1=gridY(1);
15 X1=gridX(1);
16
17 %evaluate fourier transform of f at every X in gridX
18 % Integral on the real line is approximated via the integral
       on gridY
19 f_fft =fVector.*exp(-1i*X1*gridY');
20 fHat = dY*exp(-1i*Y1*dX*([1:Nx]'-1)).*fft(f_fft);
21
22
23 end
```

9.10 Inverse Fourier Transform

```
function invF_f = InverseFourierTransform(fVector,gridY,gridX
    )
% INPUT
% fVector: vector of function f evaluated at gridY
% gridY: grid of Y process
% gridX : grid used for the integral of the EPV
%
%OUTPUT
% invF_f: vector of the inverse fourier transform of f
    evaluated at grid X

%compute fourier transform of f
Nx=length(gridX);
dY = gridY(2)-gridY(1);
dX = gridX(2)-gridX(1);
Y1=gridY(1);
X1=gridX(1);

%evaluate fourier transform of f at every X in gridX
% Integral on the real line is approximated via the integral
    on gridY
f_ifft =fVector.*exp(+1i*X1*gridY');
invF_f = Nx*dY*exp(+1i*Y1*dX*([1:Nx]'-1)).*ifft(f_ifft)/(2*pi
    );

end
```

Bibliography

[1] Joseph Abate and Ward Whitt. "A unified framework for numerically inverting Laplace transforms". In: *INFORMS Journal on Computing* 18.4 (2006), pp. 408–421.

[2] Aurélien Alfonsi. "High order discretization schemes for the CIR process: application to affine term structure and Heston models". In: *Mathematics of Computation* 79.269 (2010), pp. 209–237.

[3] Elisa Appolloni, Lucia Caramellino, and Antonino Zanette. "A robust tree method for pricing American options with the Cox–Ingersoll–Ross interest rate model". In: *IMA Journal of Management Mathematics* 26.4 (2015), pp. 377–401.

[4] Fischer Black and Myron Scholes. "The pricing of options and corporate liabilities". In: *Journal of political economy* 81.3 (1973), pp. 637–654.

[5] Bruno Bouchard, Nicole El Karoui, Nizar Touzi, et al. "Maturity randomization for stochastic control problems". In: *The Annals of Applied Probability* 15.4 (2005), pp. 2575–2605.

[6] Mitya Boyarchenko. "Carr's randomization for finite-lived barrier options: Proof of convergence". In: *Available at SSRN 1275666* (2008).

[7] Mitya Boyarchenko and Levendorskiĭ. "Ghost calibration and the pricing of barrier options and CDS in spectrally one-sided Lévy models: the parabolic Laplace inversion method". In: ().

[8] Mitya Boyarchenko and Levendorskiĭ. "Prices and sensitivities of barrier and first-touch digital options in Lévy-driven models". In: ().

[9] Svetlana Boyarchenko and Levendorskiĭ. "Efficient pricing of barrier options and credit default swaps in Lévy models with stochastic interest rate". In: ().

[10] Svetlana Boyarchenko and Levendorskiĭ. "Sinh-Acceleration: Efficient Evaluation Of Probability Distributions, Option Pricing, And Monte Carlo Simulations". In: ().

[11] Svetlana Boyarchenko and Sergei Levendorskii. “American options in Lévy models with stochastic interest rates”. In: *The Journal of Computational Finance* 12.4 (2009), p. 51.

[12] Svetlana Boyarchenko and Sergei LevendorskiĬ. “American options in the Heston model with stochastic interest rate and its generalizations”. In: *Applied Mathematical Finance* 20.1 (2013), pp. 26–49.

[13] Svetlana Boyarchenko and Sergei Levendorskii. “American options: the EPV pricing model”. In: *Annals of Finance* 1.3 (2005), pp. 267–292.

[14] Svetlana Boyarchenko and Sergei Z Levendorskii. *Non-Gaussian Merton-Black-Scholes Theory*. Vol. 9. World scientific, 2002.

[15] Maya Briani, Lucia Caramellino, and Antonino Zanette. “A hybrid approach for the implementation of the Heston model”. In: *IMA Journal of Management Mathematics* 28.4 (2017), pp. 467–500.

[16] Peter Carr. “Randomization and the American put”. In: *The Review of Financial Studies* 11.3 (1998), pp. 597–626.

[17] Carl Chiarella, Boda Kang, and Gunter H Meyer. “The evaluation of barrier option prices under stochastic volatility”. In: *Computers & Mathematics with Applications* 64.6 (2012), pp. 2034–2048.

[18] Kyriakos Chourdakis. “Levy processes driven by stochastic volatility”. In: *Asia-Pacific Financial Markets* 12.4 (2005), pp. 333–352.

[19] R Cont and P Tankov. “Financial Modelling with Jump Processes, Chapman & Hall/CRC Financ”. In: *Math. Ser* (2004).

[20] Massimo Costabile, Arturo Leccadito, and Ivar Massabó. “Computationally simple lattice methods for option and bond pricing”. In: *Decisions in economics and finance* 32.2 (2009), pp. 161–181.

[21] John C Cox Jr. “Jonathan E. Ingersoll, and Stephen A. Ross,“A Theory of the Term Structure of Interest Rates,””. In: *Econometrica* 53.2 (1985), pp. 385–407.

[22] Erik Ekström and Johan Tysk. “The Black–Scholes equation in stochastic volatility models”. In: *Journal of Mathematical Analysis and Applications* 368.2 (2010), pp. 498–507.

[23] Gianluca Fusai, Guido Germano, and Daniele Marazzina. “Spitzer identity, Wiener-Hopf factorization and pricing of discretely monitored exotic options”. In: *European Journal of Operational Research* 251.1 (2016), pp. 124–134.

[24] Ross Green, Gianluca Fusai, and I David Abrahams. “THE WIENER–HOPF TECHNIQUE AND DISCRETELY MONITORED PATH-DEPENDENT OPTION PRICING”. In: *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics* 20.2 (2010), pp. 259–288.

[25] Steven L Heston. “A closed-form solution for options with stochastic volatility with applications to bond and currency options”. In: *The review of financial studies* 6.2 (1993), pp. 327–343.

[26] Steven L Heston, Mark Loewenstein, and Gregory A Willard. “Options and bubbles”. In: *The Review of Financial Studies* 20.2 (2007), pp. 359–390.

[27] Peter Hieber. “Pricing exotic options in a regime switching economy: a Fourier transform method”. In: *Review of Derivatives Research* 21.2 (2018), pp. 231–252.

[28] Jimmy E Hilliard, Adam L Schwartz, and Alan L Tucker. “Bivariate binomial options pricing with generalized interest rate processes”. In: *Journal of Financial Research* 19.4 (1996), pp. 585–602.

[29] https://www.rocq.inria.fr/mathi/Premia. “Premia: a platform for pricing financial derivatives.” In: ().

[30] Samuli Ikonen and Jari Toivanen. “Componentwise splitting methods for pricing American options under stochastic volatility”. In: *International Journal of Theoretical and Applied Finance* 10.02 (2007), pp. 331–361.

[31] M de Innocentis and S Levendorskii. “Calibration Heston model for credit risk”. In: *Risk* (2017), pp. 90–95.

[32] Andrey Itkin. “Pricing Derivatives Under Levy Models”. In: *Pseudo-Differential Operators: Theory and Applications*. Vol. 12. Springer, 2017.

[33] Sato Ken-Iti. *Lévy processes and infinitely divisible distributions*. Cambridge university press, 1999.

[34] Oleg Kudryavtsev. “Advantages of the Laplace transform approach in pricing first touch digital options in Lévy-driven models”. In: *Boletin de la Sociedad Matematica Mexicana* 22.2 (2016), pp. 711–731.

[35] Oleg Kudryavtsev. “Finite difference methods for option pricing under Lévy processes: Wiener-Hopf factorization approach”. In: *The scientific world journal* 2013 (2013).

[36] Oleg Kudryavtsev and Sergei Levendorskiĭ. “Fast and accurate pricing of barrier options under Lévy processes”. In: *Finance and Stochastics* 13.4 (2009), pp. 531–562.

[37] Oleg Kudryavtsev and Vasily Rodochenko. “A Numerical Realization of the Wiener–Hopf Method for the Kolmogorov Backward Equation”. In: *International Scientific Conference (on) Modern Methods, Problems and Applications of Operator Theory and Harmonic Analysis*. Springer. 2018, pp. 399–426.

[38] Oleg Kudryavtsev and Vasily Rodochenko. “A Wiener-Hopf factorization approach for pricing barrier options in the Heston model”. In: *Applied Mathematical Sciences* 11.2 (2017), pp. 93–100.

[39] Harold J Kushner. “Numerical methods for stochastic control problems in continuous time”. In: *SIAM Journal on Control and Optimization* 28.5 (1990), pp. 999–1048.

[40] Sergei Levendorski ı. “Convergence of price and sensitivities in Carr’s randomization approximation globally and near barrier”. In: *SIAM Journal on Financial Mathematics* 2.1 (2011), pp. 79–111.

[41] Daniel B Nelson and Krishna Ramaswamy. “Simple binomial processes as diffusion approximations in financial models”. In: *The review of financial studies* 3.3 (1990), pp. 393–430.

[42] “Oksendal, B., Stochastic Differential Equations. Springer-Verlag Heidelberg,New York”. In: (2012).

[43] Carolyn E Phelan et al. “Fluctuation identities with continuous monitoring and their application to the pricing of barrier options”. In: *European Journal of Operational Research* 271.1 (2018), pp. 210–223.

[44] Carolyn E Phelan et al. “Hilbert transform, spectral filters and option pricing”. In: *Annals of Operations Research* 282.1-2 (2019), pp. 273–298.

[45] Erich Rothe. “Zweidimensionale parabolische randwertaufgaben als grenzfall eindimensionaler randwertaufgaben”. In: *Mathematische Annalen* 102.1 (1930), pp. 650–670.

[46] Michel Vellekoop and Hans Nieuwenhuis. “A tree-based method to price American options in the Heston model”. In: *The Journal of Computational Finance* 13.1 (2009), p. 1.

[47] JZ Wei. “Valuing American equity options with a stochastic interest rate: a note”. In: *J. Financ. Eng* 2 (1996), pp. 195–206.

[48] C Yiran. “Rollin, dB, Guido, SG: Full and fast calibration of the Heston stochastic volatility model”. In: *Eur. J. Oper. Res* 263 (2015).

[49] Robert Zvan, Peter A Forsyth, and Kenneth R Vetzal. “Penalty methods for American options with stochastic volatility”. In: *Journal of Computational and Applied Mathematics* 91.2 (1998), pp. 199–218.