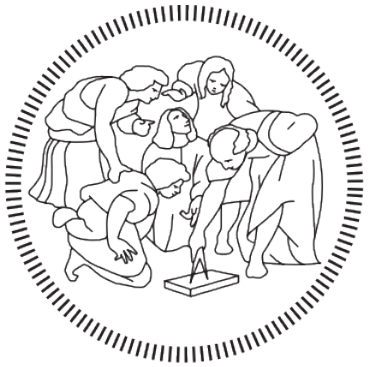

POLITECNICO
MILANO 1863

Simulation of Lévy-driven OU Processes

Financial Engineering - A.A. 2023-2024

Group 4a:

Catelli Andrea - CP: 10720665 - MAT: 247446
Marchetto Erica - CP: 10700150 - MAT: 232637
Urso Giovanni - CP: 10652148 - MAT: 221591

Contents

1 Introduction

The aim of this project is to explore the simulation of two broad categories of mean-reverting stochastic processes: Ornstein-Uhlenbeck (OU-Lévy) and Lévy-Ornstein-Uhlenbeck (Lévy-OU). Specifically, we focus on simulating Ornstein-Uhlenbeck Tempered Stable (OU-TS) and Tempered Stable Ornstein-Uhlenbeck (TS-OU) processes. To achieve this, we employ both the Exact Decomposition algorithm based on the property of self-decomposability (as done in Sabino [4] and Sabino Cufaro [5]) and the Fast General Monte Carlo method which requires only the characteristic function of the process (Baviera & Manzoni [2]). The ultimate goal of our work is to utilize these simulations to price energy European and American call options, since these kind of processes are able to capture the dynamics of energy markets.

We have developed a library in MATLAB and in Python in order to meet the requirements for the processes under study. For a better comprehension, the main script does not include all the tests conducted over the past weeks, but it does fulfill the assignment's requirements. We focused our attention on the optimization in the MATLAB code. The results reported in this document are obtained using MATLAB; in Python the results are available in the Jupyter notebook.

2 General overview

Energy and commodities markets exhibit seasonal patterns, mean reversion (tendency to return to a long-term average), high volatility, and occasional price spikes. These characteristics create the need for financial instruments, such as derivatives, that can mitigate these risks for both buyer and seller of these assets. Accurate models are crucial for effectively pricing and utilizing such instruments.

In equity market it is well known that asset returns could not be described by a Gaussian distribution, therefore analysts often rely on Lévy process in order to take into account heavy tails and the presence of jumps in the returns. Since in the energy and commodities markets it is possible to notice a mean reverting behaviour of the prices, we can not rely on a classical Lévy process.

In order to capture the mean reverting dynamics of energy markets a reasonable choice could be a Gaussian Ornstein-Uhlenbeck (OU) (described in Section 7), but this model is not able to take into account the heavy tails and the presence of jumps in the energy and commodities returns.

Lévy-driven Ornstein-Uhlenbeck processes represent a stochastic model suitable to incorporate all the features of energy and commodities markets. Lévy-driven Ornstein-Uhlenbeck processes are stochastic processes built starting from the classical Ornstein-Uhlenbeck stochastic differential equation and adding a Lévy process, meaning that they utilize a selected Lévy process as the driving background process.

Let $(\Omega, \mathcal{F}, \mathcal{F}_t, P)$ a filtered probability space, a Lévy-driven OU process is the solution of the following equation

$$\begin{cases} dX_t = -bX_t dt + dL_t \\ X_0 \quad known \end{cases} \tag{1}$$

where b is a positive constant (b>0) called *mean-reversion parameter* and L_t an adpated Lévy process to the filtration $\mathcal{F}_t$ called *background driving Lévy process.*

The unique strong solution of the Stochastic Differential Equation (1) is the process

$$X_t = X_0 e^{-bt} + Z_t \quad where \quad Z_t := \int_0^t e^{-b(t-s)} dL_s \tag{2}$$

The process L_t represents the stationary increments and completly define X_t given an initial condition, while Z_t is an additive process, i.e. an inohomogeneous Lévy process.

There are two standard ways to associate a Levy Process as a driver of an Ornstein-Uhlenbeck dynamics.

- **OU-Lévy**: Firstly, if we define the dynamics of background Levy process $L(\cdot)$ in equation (1), then the process $X(\cdot)$ driven by this driver will be called OU-Lévy process.

- **Lévy-OU**: On the other hand, it is possible to select the stationary distribution of $X(t)$. By stationary distribution of the process A (we refer to it by overlining the distribution, for instance with the characteristic function: $\overline{\phi}_A$), we mean the distribution to which the process tends for t going to infinity. The literature refers to it as Lévy-OU.

In our study we explore specifically two classes of Lévy-driven Ornstein-Uhlenbeck (OU) processes having as background Levy driver a Tempered Stable (TS) process, which represents a broad class of stochastic processes commonly used in the financial literature. We will refer to these processes as OU-TS and TS-OU depending on the choice of the way we have built them.

2.1 Self decomposability

A fundamental concept in order to study these Levy-driven OU processes is the *self-decomposability*, as explained in Sabino [4] and [5]. Recalling that a characteristic function (CF) $\eta(u)$ is said to be *self-decomposable* if for every $0 < a < 1$ it is possible to find another law with CF $\chi_a(u)$ such that:

$$\eta(u) = \eta(au)\chi_a(u) \tag{3}$$

Obviously this property about characteristic functions, can be transferred also to *random variables (rv)*: a *rv* is said to be *self decomposable* when its law is *sd*. It can be proven that for every $0 < a < 1$ we can always find two independent random variables Y and Z_a such that in distribution:

$$X = aY + Z_a \tag{4}$$

where the rv Y has the same law of X and Z_a is called the a-remainder.

Leveraging on the concept of self-decomposability it is possible to study the processes of our interest by looking at the law of the a-remainder and its properties, indeed the transition law between t and t+ Δt of a Levy-driven OU coincides with the law of the a-remainder. Sabino in [3] and [4] exploits this property in order to achieve algorithms for the simulation based on an exact decomposition of the process. See section 4.

We now exploit the different relations between the processes $X(\cdot)$, $Z(\cdot)$ and $L(\cdot)$ (in the SDE (1)) depending on the possible way to assign the law. We will concentrate on the characteristic function of the processes since it is a tractable and useful way to characterise them and it is the only ingredient necessary for the Fast General Monte Carlo (FGMC) approach proposed by Baviera and Manzoni in [1]. We will also focus on the analyticity strip of the characteristic function, which is an horizontal strip in the complex plane (containing the origin) where the characteristic function is well behaved, in order to avoid numerical errors in the simulation.

2.2 OU-Lévy process

As already said, this kind of processes are built by selecting the distribution of the background Lévy driver L_t. Is it possible to link the characteristic exponent (also said Log-Characteristic Function

LCF) of the driver L_t to the process Z_t (the a-remainder in the formulation of Sabino [4] and Sabino & Cufaro [5]).

$$\psi_Z(u,t) = \int_0^t \psi_L(ue^{-bs}) \tag{5}$$

In the case of a OU-Lévy process the analytical strip of $\psi_L(u)$ coincides with the one of $\psi_Z(u,t)$. This allows us to find a well-behaved function for our numerical experiments.

In order to check the accuracy of our simulation we provide the relation between the cumulants of the driver L_1 and the process Z_t:

$$c_k(Z_t) = \frac{1-e^{-kbt}}{kb} c_k(L_1) \tag{6}$$

2.3 Lévy-OU process

By selecting the stationary distribution of the process X_t in (1), we should ensure that there exist a consistent Lévy driver. Since the solution of (1) is stationary if and only if the characteristic function CF of the process X, $\phi_X(u,t)$, is *self decomposable*, it can be decomposed using (3) as follows:

$$\overline{\phi}_X(u) = \overline{\phi}_X(ue^{-bt})\phi_Z(u,t) \tag{7}$$

Also in this case the analyticity strip of the characteristic exponent ψ_X coincides with one of ψ_Z.

This allows to find an explicit characteristic exponent for the a-remainder (useful for the simulation procedure)

$$\psi_Z(u,t) = \overline{\psi}_X(u) - \overline{\psi}_X(ue^{-bt}) \tag{8}$$

Moreover, as done for the previous case, we recall the cumulants relation of the Z_t (the a-remainder) and the stationary law $\overline{X}$.

$$c_k(Z_t) = (1-e^{-kbt}) \cdot c_k(\overline{X}_t) \tag{9}$$

2.4 Tempered Stable process

In this section, we explore the class of Tempered Stable processes, which are the background drivers of the mean reverting dynamics of our study.

The process $L(t)$ in (1) can be seen as the difference of two indipendent subordinators :

$$L(t) = L_p(t) - L_n(t) \tag{10}$$

where both $L_p(t)$ and $L_n(t)$ are classical tempered stable processes in Sabino [4] and [5] called CTS, with Levy densities ν_p and ν_n. We will follow the dissertation of Baviera & Manzoni [2], where the Levy Driver is a bilateral classical tempered stable process (BCTS) also called *generalised* TS process.

This process is characterised by 7 parameters: α_p, α_n stability parameters (both < 2); β_p, β_n; c_p, c_n and γ_c which is related to the drift of the process. We will treat cases where α_p, α_n are different from 0 and 1: α_p, $\alpha_n \in \mathcal{A}_{TS}$ where $\mathcal{A}_{TS} := (-\infty, 2)\backslash\{0,1\}$, the treatise for the case 0 and 1 can be found in Annex B of Baviera [2]. This last parameter γ_c presented in the paper [2] allows to control the drift of the processes and it is called center of the process.

A Tempered Stable process is thus a Lévy process with characteristic exponent:

$$\begin{aligned}\psi(u) = iu\gamma_c &+ c_p\Gamma(-\alpha_p)[(1-\frac{iu}{\beta_p})^{\alpha_p} - 1 + iu\frac{\alpha_p}{\alpha_p}] \\ &+ c_n\Gamma(-\alpha_n)[(1-\frac{iu}{\beta_n})^{\alpha_n} - 1 + iu\frac{\alpha_n}{\alpha_n}]\end{aligned} \tag{11}$$

where α_p, $\alpha_n \in \mathcal{A}_{TS}$, β_p, $\beta_n >0$, $c_p, c_n >= 0$, $\gamma_c \in \mathbb{R}$. Then the characteristic triplet is $(0, \eta, \gamma)$ with

$$\begin{cases} \nu(x) = c_p \frac{e^{-\beta_p x}}{x^{1-\alpha_p}} \mathbf{1}_{x>0} + c_n \frac{e^{-\beta_n |x|}}{|x|^{1-\alpha_p}} \mathbf{1}_{x<0} \\ \gamma = \gamma_c - c_p \beta_p^{\alpha_p - 1} \Gamma_U(1-\alpha_p, \beta_p) + c_n \beta_n^{\alpha_n - 1} \Gamma_U(1-\alpha_n, \beta_) \end{cases} \tag{12}$$

where $\Gamma_U(\alpha, x)$ is the upper incomplete gamma function.

As reminded early, we report the analyticity strip of the CF for a TS process

$$(p_-, p_+) = (-\beta_p, \beta_n) \tag{13}$$

Thanks to Lukacs we are able to link the analyticity strip of a TS process to the one of the whole OU-Lévy driven process, object of our study.

Moreover a useful formula, in order to check the precision of the simulations we are going to perform, is the one for the computation of the cumulants of a TS process (see [2]):

$$C_k^{TS}(L_1) = C_k^{BCTS}(L_1) = \begin{cases} \gamma_c & if\ k = 1 \\ c_p \beta_p^{\alpha_p - k} \Gamma(k - \alpha_p) + (-1)^k c_n \beta_n^{\alpha_n - k} \Gamma(k - \alpha_n) & if\ k \geq 2 \end{cases} \tag{14}$$

By combining equation (6) and equation (14) we are able to compute the cumulants for the whole process $X(t)$, solution of the initial SDE (1), in the case we select the distribution of Lévy driver (OU-TS). On the other hand, we can compute the cumulants of the same process $X(t)$ for the TS-OU process by combining equation (9) and the usual one for TS comulants (14).

As already said, in Sabino [4] and [5] all the studies are provided just for the unilateral case, but, thanks to the relation (10), the bilateral case is straightforward. We provide here the cumulants for the CTS (unilateral case) that we use in the following sections in order to join the studies of Baviera [2] and Sabino [4] and [5]:

$$C_k^{CTS}(L_1) = c\beta^{\alpha - k} \Gamma(k - \alpha) \quad for\ k \geq 1 \tag{15}$$

where α, β and c are the usual parameters of a TS unilateral process. This formula can be used for both the positive and negative Levy process in (10), paying attention to the sign for the cumulants of odd order. Is it possible to compute the cumulants of the OU-CTS process as already explained for the bilateral case by combining (15) and (6). For the CTS-OU process, since we select the stationary distribution, we can perform the same computation combining (15) and (9).

In the following sections, we will refer to a bilateral tempered stable process for simplicity just as TS following the paper of Baviera & Manzoni [2] except for specific needs.

2.5 Classification of TS processes

Depending on the values of the stability parameter α_p and α_n, we can subdivide the whole class of Tempered Stable process. This classification will lead to different processes' behaviours and to different numerical procedures.

In the literature, the first subclass is the **Finite** and **Infinite Activity**. A process is Finite Activity if, for any finite time interval $(0, T)$, the number of jumps that occur is finite with probability 1. Mathematically, if $N(t)$ denotes the number of jumps of the process up to time t, then $P(N(T) < \infty) = 1$ for all $T > 0$.

It is also possible to divide the Infinite Activity case depending on the **Variation**. A stochastic process $X(t)$ is said to be Finite Variation on a time interval $[0, T]$ if the total variation of the process over this interval is finite almost surely. The total variation of $X(t)$ on $[0, T]$ is defined as:

$$V_{[0,T]}(X) = \sup_{\Pi} \sum_{i=1}^{n} |X(t_i) - X(t_{i-1})| \tag{16}$$

where the supremum is taken over all possible partitions $\Pi = \{0 = t_0 < t_1 < \cdots < t_n = T\}$ of the interval $[0, T]$.

It can be proven that all the finite activity processes have also finite variation and, concerning the TS-OU process, in the case of finite activity, it is well defined just for the stability parameter α equal to zero. We report in the table below a summary of the processes that we took in consideration during this study.

	Finite Activity	**Infinite Activity and Finite Variation**
TS	$max(\alpha_p, \alpha_n) < 0$	$max(\alpha_p, \alpha_n) \in [0, 1)$
OU-TS	$max(\alpha_p, \alpha_n) < 0$	$max(\alpha_p, \alpha_n) \in [0, 1)$
TS-OU	$\alpha_p, \alpha_n = 0$	$max(\alpha_p, \alpha_n) \in [0, 1)$

Table 1: Process Classification

3 Model Parameters

The parameters for all the computations were as follows:

b	β_p	β_n	c_p	c_n	γ_c
0.1	2.5	3.5	0.5,	1	0

Table 2: Parameter values

The stability parameters α_p and α_n were set equal to 0.5 for Finite Variation processes and -1 for Finite Activity processes. In our MATLAB code you will find a unique α without considering the possibility of two different values for α_p and α_n, since it was not required for the assignment. Clearly the generalization is straightforward.

We explored different parameters and diverse time horizons for both methods and we report the found results.

4 Exact Decomposition Simulation

In this section, we employ the notion of *self decomposability* (see section 2.1) in order to simulate the processes using the algorithm of Exact Decomposition. In order to check the correctness of our procedure, firstly we checked the result obtained with the one in Sabino [4] and Sabino & Cufaro [5].

4.1 OU-TS Finite Activity

In the case of Finite Activity, where $\alpha_p < 0$, as stated in Sabino [4], for every $t > 0$ and for $X(0) = X_0$ it holds that the solution of an OU-CTS equation is $P - a.s.$ the sum of two independent *random variables*:

$$X(t) = aX_0 + Z_p(t) = aX_0 + X_1 \tag{17}$$

where $a = e^{-bt}$ and X_1 (the a-remainder in the formula (4)) can be written as $L_p(t) = \sum_{k=0}^{N_p(t)} \tilde{J}_k$ since the Background Driver Lévy Process $L(\cdot)$ coincides with a compound Poisson process.
In particular, $N_p(t)$ represents a Poisson process with intensity $\lambda_p = c_p\Gamma(\alpha_p)\beta_p^{-\alpha_p}$ and the jump sizes J_k are *i.i.d.* with distribution equal to a Gamma with parameters $-\alpha_p$ and β_p.

In order to simulate a bilateral TS (BCTS), we remind that the exact decomposition becomes

$$X(t) = aX_0 + Z_p(t) = aX_0 + X_1^p - X_1^n \tag{18}$$

where the quantities specified above for the unilateral case are the same using the negative parameters.

In order to simulate this process on the time grid of our interest, we considered the recursive algorithm explained in Sabino [4]. The procedure, considering the initial condition $X(t_0) = x_0$ and taking $a_i = e^{-b(t_i - t_{i-1})}$, consist in the implementation of the following expression:

$$X(t_i) = a_i X(t_{i-1}) + Z_{a_i}, \quad i = 1, ..., I \tag{19}$$

For the derivation of the Z term, which will be referred to as x_1, the algorithm starts considering the computation of the two terms $\Delta_{t_i} = t_i - t_{i-1}$ and a_i with the formula above, in our case both constant. It is then required to generate the independent Poisson distribution $N \sim P(\lambda_p, \Delta_{t_i})$ and n *i.i.d.* Uniform *random variables* with distribution $U_m \sim U(0,1)$, $m = 1, ..., n$. The term x_1 of i^{th} iteration is then given by $\sum_{m=1}^{n} \tilde{J_m}$, where $\tilde{j_m}$ is the realization of the simulated distribution $\tilde{J_m} \sim G(-\alpha_p, \tilde{\beta_m})$ with $\tilde{\beta}_m$ given by $\beta_p e^{bu_m dt}$, $m = 1, ..., n$.
The entire computation is made both for the negative and positive jumps that compose the term x_1 using in practice the equation (10) to obtain a bilateral TS.

In our MATLAB implementation, in order to join Sabino [4] and Baviera & Manzoni [2] studies, it was necessary to subtract from the two final terms the cumulants related to the corresponding jump process *OU-CTS cumulants* i.e. the one calculated by using (15) and (6). The final step of every iteration was then given by:

$$X_i = \gamma_c dt + aX(t_{i-1}) + {x^p}_1 - C_{1,p}^{OU-CTS} - ({x^n}_1 - C_{1,n}^{OU-CTS}) \tag{20}$$

Doing this procedure, it is possible to control the drift of the process from the parameter γ_c, since we are compensating the first moments of the a-remainder X_1^p and X_1^n. Otherwise, by following the paper of Sabino [4] given a set of parameters: (α, β, c) the mean of a-remainder is not equal to 0.

In order to understand the correctness of our simulated process, we compared the first four cumulants of the empirical distribution obtained with $\Delta = \frac{1}{12}$ to the corresponding four theorical cumulants of the process X_t computed combining the formula (14) for the cumulants of the TS driver and the relation (6). Furthermore, we decided to repeat the same computation also for an higher time to maturity $T = 1$. The following table shows the numerical results.

	$c_1(X_T) \cdot 1000$		$c_2(X_T) \cdot 1000$		$c_3(X_T) \cdot 1000$		$c_4(X_T) \cdot 1000$		ED
	True	ED	True	ED	True	ED	True	ED	Comp. Time
$T = \frac{1}{12}$	0.000	-0.012	9.144	9.150	3.030	3.030	13.816	13.934	7.088134 s
$T = 1$	0.000	0.034	100.285	100.326	31.807	32.037	138.940	139.315	14.530674 s

Table 3: Comparison of True and Simulated values of cumulants at different times T (scaled by 1000) for OU-TS Finite Activity. ED algorithm settings: 10^7 simulations.

As observed in the table, the results exhibit a high degree of accuracy between the simulated and theoretical values, meaning that our simulated process is correctly capturing the dynamics of the model; but the computational time is not negligible.

For a better comprehension of the problem, we decided to analyze also the path of the simulations. We then ran again the code considering time to maturity $T = 1$ and $M = 1000$ time steps, as shown in the following plot.

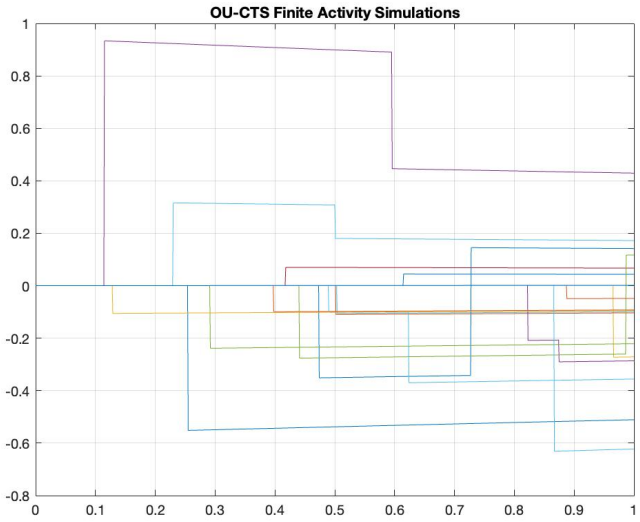

Figure 1: OU-TS Finite Activity

4.2 OU-TS Finite Variation

Concerning the Finite Variation OU-CTS processes, thanks to the *self decomposability* (section 2.1) it holds that for $0 \leq \alpha < 1$ and at every $t > 0$ its pathwise solution is $P - a.s.$ in distribution the sum of three independent *random variables*:

$$X(t) = aX_0 + X_1 + X_2 \tag{21}$$

with $L(1) \sim TS(\alpha, \beta, c)$, $X(0) = X_0$ and $a = e^{-bt}$.

The distribution of X_1 is the law $TS(\alpha, \frac{\beta}{\alpha}, c\frac{1-a^\alpha}{\alpha b})$ (tempered stable, but with different parameters), while $X_2 = \sum_{k=1}^{N_a} J_k$ is a compound Poisson *random variable* with intensity:

$$\Lambda_a = \frac{c\beta^\alpha \Gamma(1-\alpha)}{b\alpha^2 a^\alpha}(1 - a^\alpha + a^\alpha log(a^\alpha)) \tag{22}$$

As already said for the case of finite activity, it is possible to extend to a OU-TS process using (10) obtaining the following result

$$X(t) = aX_0 + X_1^p + X_2^p - X_1^n - X_2^n \tag{23}$$

by using the corresponding negative parameters.

The recursive algorithm considered for Finite Variation OU-CTS processes is the one explained in Sabino [5].

As in the previous section, the first step of the procedure is taking into account the initial condition X_0 and the constant quantity $a = e^{-b\Delta t}$. Then, for every time step, we initially simulate the first process $X_1 \sim TS(\alpha, \frac{\beta}{\alpha}, c\frac{1-a^\alpha}{\alpha b})$ that composes X. Regarding X_2, we need to simulate $N_a \sim P(\Lambda_a)$ and V_i in order to get $J_i \sim G(1-\alpha, \tilde{\beta}_i), i = 1, ..., n$, where $\tilde{\beta}_i$ is given by $\tilde{\beta}_i = \beta v_i$. The random rate parameter V is distributed according to the *pdf*:

$$f_V(v) = \frac{\alpha a^\alpha}{1 - a^\alpha + a^\alpha log(a^\alpha)} \frac{v^\alpha - 1}{v} \tag{24}$$

Finally, after having repeated the procedure for positive and negative cases, the OU-TS process is obtained first by computing $x_2 = \sum_{i=1}^{n} j_i$ and then combining the two simulated processes with the previous step.

As we did in the Finite Activity case, it was necessary to subtract the theoretical cumulants *(OU-CTS cumulants)* from both the negative and positive parts in order to obtain the final process:

$$X(t_i) = \gamma_c dt + aX(t_{i-1}) + x_1^p + x_2^p - C_{1,p}^{OU-CTS} - (x_1^n + x_2^n - C_{1,n}^{OU-CTS}) \tag{25}$$

In the following table we explore the numerical results obtained for cumulants, comparing different time to maturities.

	$c_1(X_T) \cdot 1000$		$c_2(X_T) \cdot 1000$		$c_3(X_T) \cdot 1000$		$c_4(X_T) \cdot 1000$		ED
	True	ED	True	ED	True	ED	True	ED	Comp. Time
$T = \frac{1}{12}$	0.000	0.060	20.449	20.435	0.762	0.802	8.908	8.919	27.818312 s
$T = 1$	0.000	-0.253	224.271	224.221	7.996	8.165	89.584	90.089	644.089713 s

Table 4: Comparison of True and Simulated values of cumulants at different times T (scaled by 1000) for OU-TS Finite Variation. ED algorithm settings: 10^7 simulations.

As in the previous case, we obtain quite a good matching between simulated and theoretical values, but the computational time of OU-TS Finite Variation processes is much higher than the corresponding Finite Activity process, mainly due to the necessity to simulate a TS variable.

Again we decided to plot the path of some simulation considering $M = 1000$ and time to maturity $T = 1$, as can be seen in the following graph.

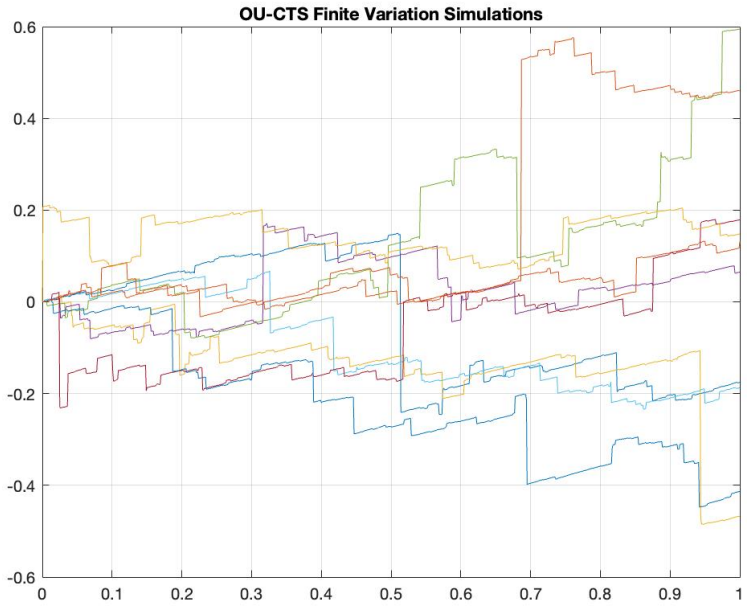

Figure 2: OU-TS Finite Variation

4.3 TS-OU Finite Variation

The Finite Variation CTS-OU processes $X(t)$, as stated in Sabino [5], with initial condition $X(0) = X_0$ $P - a.s.$, $0 \leq \alpha < 1$ and with stationary distribution $TS \sim (\alpha, \beta, c)$ whose Lévy density has tempering function $q(x) = e^{-\beta x}$, $\beta > 0$, can be represented as:

$$X(t) = X_0 e^{-bt} + X_1 + X_2 \tag{26}$$

Like in the process analyzed before, X_1 is a $TS(\alpha, \beta, c(1 - a^\alpha))$ (still a TS, but with different parameters) and $X_2 = \sum_{i=1}^{N_a} \tilde{J}_i$ is a compound Poisson *random variable* such that $N_a \sim P(\Lambda_a)$, with $\Lambda_a = c\Gamma(1 - \alpha)\frac{\beta^\alpha}{\alpha}(1 - a^\alpha)$.

As done for the previous process, we can easily extend the exact decomposition also for the bilateral TS-OU process:

$$X(t) = aX_0 + X_1^p + X_2^p - X_1^n - X_2^n \tag{27}$$

The simulation algorithm is again the recursive procedure described in Sabino [5].

First, we consider the initial condition X_0 and the quantity $a = e^{-b\Delta t_m}$ which remains constant for every iteration. It is then required to simulate the first process $X_1 \sim TS(\alpha, \beta, c(1 - a^\alpha))$ which already represents an element of the final process X. The next step is about the computation of X_2, which is obtained by $x_2 = \sum_{i=1}^{n} j_i$. In order to obtain J_i, we start simulating the Poisson process $N_a \sim P(\Lambda_a)$ and the Uniform distribution $U_i \sim U(0, 1)$, $i = 1, ..., n$. Then we obtain the distributions $v_i = (1 + (a^{-\alpha} - 1)u_i)^{\frac{1}{\alpha}}$ and $\tilde{\beta}_i = \beta v_i$, $i = 1, ..., n$, from which we can compute $J_i = G(1 - \alpha, \tilde{\beta}_i$, $i = 1, ..., n$.
The procedure is finally repeated for the positive and negative cases, like in the previous computations.

The process X is therefore obtained by the composition of these different processes, considering positive and negative parts, and considering the compensation represented by the cumulants:

$$X(t_i) = \gamma_c dt + aX(t_{i-1}) + x_1^p + x_2^p - C_{1,p}^{CTS-OU} - (x_1^n + x_2^n - C_{1,n}^{CTS-OU}) \tag{28}$$

The numerical results obtained with different time to maturities are represented below.

	$c_1(X_T) \cdot 1000$		$c_2(X_T) \cdot 1000$		$c_3(X_T) \cdot 1000$		$c_4(X_T) \cdot 1000$		ED
	True	ED	True	ED	True	ED	True	ED	Comp. Time
$T = \frac{1}{12}$	0.000	0.006	4.090	4.057	0.229	0.199	3.563	3.438	9.590744 s
$T = 1$	0.000	-0.00063	44.854	44.853	2.399	2.360	35.834	36.176	18.825496 s

Table 5: Comparison of True and Simulated values of cumulants at different times T (scaled by 1000) for TS-OU Finite Variation. ED algorithm settings: 10^7 simulations.

We first notice that the computational times for TS-OU Finite Variation processes are much lower than in the previous analyzed case, but still higher with respect to Finite Activity case.
However, we obtain again a good matching between simulated and true cumulants.

The path of this TS-OU process is represented below, considering $M = 1000$ and time to maturity $T = 1$ as before.

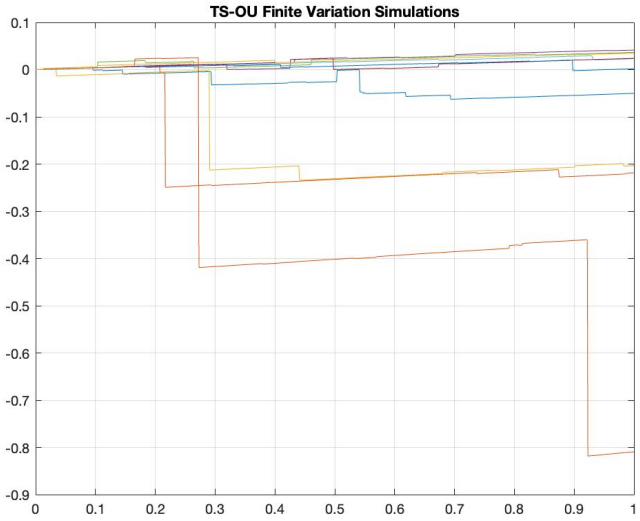

Figure 3: TS-OU Finite Variation

5 Fast and General Monte Carlo Simulation

The exact decomposition technique, described in previous sections, becomes computationally expensive for specific model parameters, due to the acceptance-rejection procedures in random variables sampling. A different approach, proposed in Baviera & Manzoni [2], consists in the extimation of the CDF of the integrated process $Z_{\Delta t}$, exploiting the knowledge of its characteristic function, and then in a simple sampling, through the probability integral transform. As described in the paper, it is possible to retrive the CDF of the integrated Lévy driver using the following formula:

$$CDF_{Z_{\Delta t}}(x) = R_a - \frac{e^{ax}}{\pi}\int_0^\infty Re\left(\frac{e^{-iux}\phi(u+ia,\Delta t)}{i(u+ia)}\right)du \tag{29}$$

where ϕ is the characteristic function of $Z_{\Delta t}$, whose analytical strip is $(-\beta_p, \beta_n)$. The parameter a is the shift in the imaginary axis and in our simulation it is selected to be:

$$a = \begin{cases} \frac{\beta_n}{2} & \text{if } \beta_n > \beta_p \\ -\frac{\beta_p}{2} & \text{if } \beta_p > \beta_n \end{cases} \qquad \text{for Finite Variation process} \tag{30}$$

$$a = \begin{cases} \frac{\beta_n}{4} & \text{if } \beta_n > \beta_p \\ -\frac{\beta_p}{4} & \text{if } \beta_p > \beta_n \end{cases} \qquad \text{for Finite Activity process} \tag{31}$$

This choice allowed us to achieve the maximum distance from integrand function singolarities. The R_a takes value 1 for positive values of a, otherwise it is set to 0.

In the first implementation of the algorithm we tried to solve the integral via quadrature method, using the MATLAB function `integral` only for few points, due to the high computational time. Once we realized the correctness of the procedure, we adopted the FFT technique to solve the integral, as suggested in the article.

Regarding the FFT hyperparameters, we tried to proceed by comparing the values of the integral via FFT with the one via quadrature method, using the already implemented method in MATLAB as a benchmark for hyperparameters search. This procedure was not possible since the computational time was too high when we needed an appropriate integral grid.

Once we recognized that, we needed a different way in order to tune the hyperparameters, we selected by default $M = 16$ and the parameter du which is the step interval in the integral grid of the FFT. However, we relied on the theoretical explanation provided in the article. After several trials, we noticed that in some cases, it was necessary to set an higher value for the FFT hyperparameter M in order to improve the accuracy.

The parameter du plays a crucial role in controlling the error in integral computation and requires careful selection to effectively manage the trade-off of truncation and discretization errors.

As explained by Azzone & Baviera in [1], in order to find a formula for the optimal du the idea is that the truncation and the discretization errors have about the same order of magnitude.

Concerning the Finite Variation processes, since both the truncation and the discretization errors have an exponential order, it is possible to find a closed formula for the optimal du.

$$\mathcal{E}(x) := CDF_{Z_{\Delta t}}(x) - \hat{CDF}_{Z_{\Delta t}}(x) \leq \frac{Be^{ax}}{\pi\omega}\frac{e^{-\ell(Ndu)^\omega}}{\ell(Ndu)^\omega} + \frac{1}{e^{2\pi|a|/du} - e^{-2\pi|a|/du}}\left[1 + e^{2ax}\phi_Z(2ia)\right] \tag{32}$$

$$du(N) = \left(\frac{2\pi|a|}{\ell N^\omega}\right)^{\frac{1}{\omega+1}} \tag{33}$$

where l and ω control the exponential decay of the the characteristic function:

$$|\phi_Z(u+ia,T)| \leq Be^{-l|u|^\omega} \qquad \forall a \in (p_-, p_+) \tag{34}$$

On the other hand for the Finite Activity the characteristic function of the process has a power law decay.

$$|\phi_Z(u+ia,t)| < B|u|^{-\omega} \qquad \forall a \in (p_-, p_+) \tag{35}$$

This fact moves also in the bounds of the errors for the CDF, the truncation error will have a power order with respect to Nh, while the discretization error still has an exponential behaviour.

$$\mathcal{E}(x) := CDF_{Z_{\Delta t}}(x) - \hat{C}DF_{Z_{\Delta t}}(x) \leq \frac{Be^{ax}}{\pi\omega}(Ndu)^{-\omega} + \frac{1}{e^{2\pi a/du} - e^{-2\pi a/du}}\left[1 + e^{2ax}\phi_Z(2ia)\right] \quad (36)$$

Since it is not possible to find an explicit formula in order to find the optimal du, we tried to perform a research of the optimal du looking at the grid in the FFT method.

After having computed the integral using the FFT procedure with the determined hyperparameters, we compared the resulting values with those obtained using quadrature. This comparison provided evidence supporting the optimality of our parameter selection.

Once we found the CDF values for a discrete grid, the following steps in the article rely on its continuous extension and on the sampling process. We started by selecting the largest not decreasing (except from a given tolerance, when it was necessary) sequence of points between 0 and 1, among the one coming from the FFT computation. We then consider the inverse of the CDF, and we sampled from it by finding $CDF^{-1}(U)$ where U is a uniform random variable. Interpolation is employed to evaluate the inverse function for U values falling between the selected grid points; otherwise, an exponential decay approach is applied, using the extreme points for extrapolation when U approaches 0 or 1. Despite the suggestion in the article to employ spline interpolation, we found that this resulted in a non-monotonic sample with respect to the U vector for some trials, hence we opted for linear interpolation instead.

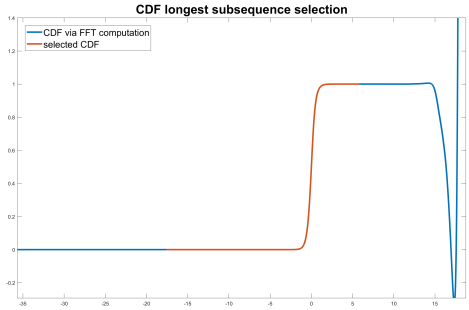

Figure 4: Selection of the longest subsequence that satisfy increasing monotonicity and values between 0 and 1: the line in blue is the output of Eq.(29), the red one is the longest sequence.

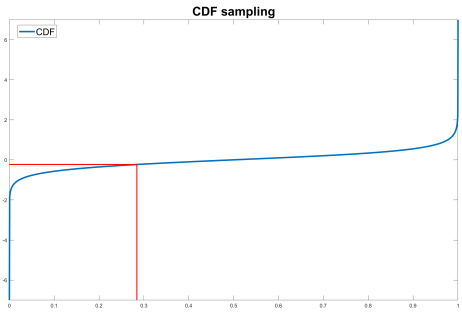

Figure 5: In order to perform sampling from our discrete CDF, we inverted the points and perform interpolation or extrapolation based on the value of the quantile. The red vertical line represents a random quantile and the horizontal line is the respective sampled value.

In the following sections, we will delve into the results, findings and additional insights gained by employing FGMC for each of the processes presented before. Specifically the simulation, we will show, are performed using as time horizons $T = \frac{1}{12}$, as requested, and $T = 1$. Moreover we will consider a single time step, as we are interested in the value of the process at the end of the simulation. However, we will address the issues and challenges of time discretization in subsequent paragraphs, particularly when dealing with path-dependent derivatives.

5.1 OU-TS Finite Variation

In the OU-TS Finite Variation the characteristic exponent of Z_T reads:

$$\Psi_Z(u,T) = iu\frac{1-e^{-bT}}{b}\gamma_c + \frac{c_p\beta_p^\alpha\Gamma(-\alpha)}{b}\left(\int_{\beta_p}^{\beta_p e^{bT}}\frac{(z-iu)^\alpha}{z^{\alpha+1}}\,dz - bT + \frac{\alpha}{b}iu(1-e^{-bT})\right) + \frac{c_n\beta_n^\alpha\Gamma(-\alpha)}{b}\left(\int_{\beta_p}^{\beta_p e^{bT}}\frac{(z+iu)^\alpha}{z^{\alpha+1}}\,dz - bT - \frac{\alpha}{b}iu(1-e^{-bT})\right) \tag{37}$$

Moreover, we found the values l and ω of Eq.(34) and consequently the optimal du exploiting Lemma 11 in [2], from which we retrived:

$$\omega = \alpha$$
$$l = -(c_p + c_n)\left(\Gamma(-\alpha)\,cos\left(\alpha\frac{\pi}{2}\right)\frac{1-e^{-\alpha bT}}{\alpha b}\right)$$

We decided to check the formula (33), finding the parameters using `lsqnonlin` to minimize the difference of the characteristic function and our extimated exponential decay for large values of u using the inequality (34). The obtained results for $T = \frac{1}{12}$ in Table (6) gave us evidence of the correctness of the theoretical du result.

Theoretical du	**Extimated du**
0.16771	0.171878

Table 6: du optimal values for FFT algorithm (M=18). The first column refers to the result found in Lemma 11 [2], the second column contains the du found, numerically retriving the decay parameters.

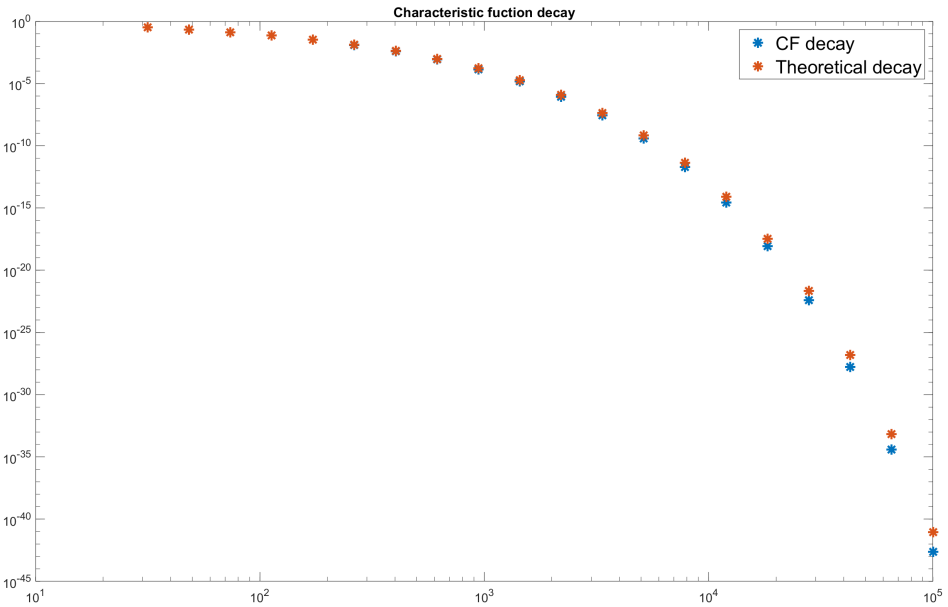

Figure 6: Characteristic exponent decay for large values of u in loglog scale. Blue points refer to the actual characteristic function, in red the exponential decay using the fitted parameters.

The simulation results are shown in Table (7). When comparing these computational times with those in Table (4), we observe a significant reduction, particularly for $T = 1$. This is because the Exact

Decomposition algorithm experiences a slowdown due to the acceptance-rejection technique used in the sampling process. As for $T = \frac{1}{12}$ the CDF becomes steep we would need more precision in the FFT computation, so we decided to increase the M parameter to 18, until we have similar results for cumulants' values.

	$c_1(X_T) \cdot 1000$		$c_2(X_T) \cdot 1000$		$c_3(X_T) \cdot 1000$		$c_4(X_T) \cdot 1000$		FGMC
	True	FGMC	True	FGMC	True	FGMC	True	FGMC	Comp. Time
$T = \frac{1}{12}$	0.000	0.007	20.449	20.398	0.762	0.733	8.908	8.675	1.183568 s
$T = 1$	0.000	0.001	224.271	224.195	7.995	7.998	89.584	89.567	1.124111 s

Table 7: OU-TS Finite Variation theorical, simulated cumulants and computational time for different time horizons. Model parameters: $[b, \alpha, \beta_p, \beta_n, c_p, c_n, \gamma] = [0.1, 0.5, 2.5, 3.5, 0.5, 1, 0]$. FGMC settings: 10^7 simulations, M=16 for $T = \frac{1}{12}$, M=18 for $T = 1$, optimal du step for FFT computation.

5.2 TS-OU Finite Variation

In the TS-OU Finite Variation model, the characteristic function for the process Z_T can be derived using (7), given that X is a TS process with its characteristic exponent provided in (11).

To determine the optimal choice for du for this process, we followed Lemma 13 in [2], which states:

$$\omega = \alpha$$

$$l = -(c_p + c_n)\left(\Gamma(-\alpha)\, cos\left(\alpha\frac{\pi}{2}\right)(1 - e^{-\alpha bT})\right)$$

We repeated the same optimal du check made before. Using $M = 24$ for $T = \frac{1}{12}$, we obtained similar du values through both theoretical calculations and by estimating decay parameters.

Theoretical du	**Extimated du**
0.308931	0.312251

Table 8: du optimal values for FFT algorithm (M=24). The first column refers to the result found in Lemma 13 [2], the second column contains the du found, numerically retriving the decay parameters.

We exhibit the simulation summary in Table(9). The selection of the M parameter as 24 for $T = \frac{1}{12}$ was made to ensure the matching of cumulants. However, this led to a spike in computational time. Given that the bottleneck in the FGMC algorithm lies in the FFT computation, the advantage becomes more pronounced when we increase the number of time steps compared to the exact algorithm.

	$c_1(X_T) \cdot 1000$		$c_2(X_T) \cdot 1000$		$c_3(X_T) \cdot 1000$		$c_4(X_T) \cdot 1000$		FGMC
	True	FGMC	True	FGMC	True	FGMC	True	FGMC	Comp. Time
$T = \frac{1}{12}$	0.000	0.002	4.090	4.124	0.229	0.230	3.563	3.576	5.450157 s
$T = 1$	0.000	0.001	44.854	44.880	2.399	2.356	35.834	35.645	0.900466 s

Table 9: TS-OU Finite Variation theorical, simulated cumulants and computational time for different time horizons. Model parameters: $[b, \alpha, \beta_p, \beta_n, c_p, c_n, \gamma] = [0.1, 0.5, 2.5, 3.5, 0.5, 1, 0]$. FGMC settings: 10^7 simulations, M=24 for $T = \frac{1}{12}$, M=16 for $T = 1$, optimal du step for FFT computation.

5.3 OU-TS Finite Activity

For OU-TS Finite Activity process the algorithm employed for the simulation of the integrand process Z_T differs slightly. For each simulation, a Bernoulli random variable is sampled to determine whether a jump has occurred. If a jump occurs, we proceed with the same approach used in other cases but with a different ϕ_V, which results to be:

$$\phi_V(u,T) = \frac{e^{\lambda T \phi_J(u,T)} - 1}{e^{\lambda T} - 1} \tag{38}$$

where:

$$\lambda = (c_p \beta_p^\alpha + c_n \beta_n^\alpha)\Gamma(-\alpha) \tag{39}$$

$$\phi_J(u,T) = \frac{1}{bt}\left(\int_{\beta_p}^{\beta_p e^{bT}} \frac{(z-iu)^\alpha}{z^{\alpha+1}}\,dz + \int_{\beta_p}^{\beta_p e^{bT}} \frac{(z+iu)^\alpha}{z^{\alpha+1}}\,dz\right) \tag{40}$$

An optimal du could be equal to 0.4 since by looking at the grid of the FFT this could be a good choice for the following reasons. The integration grid is sufficiently large in order to minimize the truncation error, taking into account the fact that in this case the decay of this error is power law. By using this choice of du the grid is still dense enough to make a reasonable discretization error. We found that this could be a good trade-off between the two different error in formula (36).

	$c_1(X_T)\cdot 1000$		$c_2(X_T)\cdot 1000$		$c_3(X_T)\cdot 1000$		$c_4(X_T)\cdot 1000$		FGMC
	True	FGMC	True	FGMC	True	FGMC	True	FGMC	Comp. Time
$T=\frac{1}{12}$	0.000	-0.111	9.144	9.124	3.030	3.290	13.816	14.964	1.066717 s
$T=1$	0.000	-1.671	100.285	100.511	31.807	34.578	138.940	159.022	1.105846 s

Table 10: OU-TS Finite Activity theorical, simulated cumulants and computational time for different time horizons. Model parameters: $[b,\alpha,\beta_p,\beta_n,c_p,c_n,\gamma] = [0.1,-1,2.5,3.5,0.5,1,0]$. FGMC settings: 10^7 simulations, M=16, optimal du step for FFT computation.

6 Energy Derivative Pricing

In this section, we apply the results obtained above concerning the simulation of mean-reverting stochastic processes to price European and American call options. In both cases we assume the underlying to be modelled as $S(t) = F_{0,t}e^{X(t)+f_t}$, where $X(t)$ is an OU Lévy driven process.

We found necessary, before implementing the pricing functions, to modify the three processes' simulations in order to get also the *log forward* returns.
Assuming that the market model is specified in the risk-neutral measure, the *forward* has to satisfy the martingality condition.

$$\mathbf{E}[F_{t,t}|\mathcal{F}_t] = F_{0,t} \tag{41}$$

This means that it has to be added a correction term to the derivation of the process $X(t)$ to achieve a correct computation of the European price.

In particular, the correction term f_t, that we call *drift*, it is a deterministic function and it coincides with the value:

$$f_t = -\Psi_Z(-i,t) \tag{42}$$

Since it depends on t, when we are dealing with a simulation with more than one time step, we should compute the drift and sum it for every step.

$$logFwd = X(t) + f_t \tag{43}$$

In order to check the correctness of our procedure, in the code we ensured to compute the statistical mean of the exponential of the *log forward* in the equation (43), checking the martingale condition (41).

After having considered this adjustment, we were able to price a European call and an American Call under the *risk neutral* measure. The results provided in the following sections were found by fixing a constant interest rate (r) equal to 0, in the library provided it is possible to set a different value of the interest rate.

6.1 European Call

We were required to price an European call option using both Exact Decomposition and Fast General Monte Carlo approach. For this option we considered a time to maturity $\Delta = \frac{1}{12}$ and we evaluated it with respect to different levels of *log-moneyness*, in particular we took 10 values in the interval $(-0.2, 0.2)\sqrt{\Delta t}$.

After having considered the adjustment about the drift correction, we were able to price a European call discounting the payoffs obtained.

First, we derived the vector of strikes corresponding to the values of moneyness considered using the relation:

$$K = F_0 e^{-x} \tag{44}$$

where x represents the *forward log-moneyness* and $F_0 = S_0 e^{rT}$.
The simulated underlying was then given by the product of the *forward* F_0 and the exponential of the *log forward* returns in (43). Obtaining the underlying, we could compute the discounted payoff:

$$P_{call} = e^{-rT}[F(t,t) - K, 0]^+ \tag{45}$$

The final price was then computed taking the average value of the prices on all the simulations. Since we are performing a Montecarlo procedure, we also provided confidence intervals. In the tables below we report just the mid prices for seek of readability, in the plots we show these values together with the confidence intervals.

For every process under consideration, we compared the price obtained via the above computation with the one obtained using Lewis formula via Fast Fourier Transform. At the beginning we implemented the integral in the Lewis formula also using the quadrature method, in order to have a good benchmark for the right computation of the integral. We noticed that for the cases of our study, the FFT method was stable to the choice of hyperparameters such as M (power of two of the number of intervals) and x_1 (the initial point in the integration domain). Then we decided to use the FFT method, in order to save computational time without losing accuracy.

The only necessary ingredient for the Lewis calculation was the characteristic function of the *a-remainder* drifted process, which can be found by:

$$\phi_Z(u) \cdot e^{iuf_t} \tag{46}$$

where ϕ_Z is the characteristic function of process in (2) which depends on the model we are dealing with.

We performed the pricing for the OU-TS Finite Variation as requested by the assignment, but we also decided to analyze the prices in the cases of OU-TS Finite Activity and TS-OU Finite Variation.

6.1.1 OU-TS Finite Variation

Concerning the pricing with an OU-TS Finite Variation process, we considered a stability parameter equal to $\alpha = 0.5$ as in the previous computations and then evaluated the European call price with the *log forward* return given by the simulation.

The prices obtained with the chosen values of *log-moneyness* are represented in the following table.

Moneyness	-0.0577	-0.0449	-0.0321	-0.0192	-0.0064	0.0064	0.0192	0.0321	0.0449	0.0577
ED%	2.2844	2.4918	2.7464	3.0820	3.5743	4.2308	5.0010	5.8484	6.7493	7.6880
FGMC %	2.2853	2.4923	2.7467	3.0821	3.5739	4.2301	5.0002	5.8477	6.7487	7.6874
Lewis %	2.2878	2.4951	2.7497	3.0853	3.5773	4.2335	5.0035	5.8508	6.7516	7.6901

Table 11: European Call Prices in percentage for different values of *log-moneyness* with the three different methods. Settings: 10^7 simulations, $M_{fft} = 16$.

As we expected, the Fast and General Monte Carlo approach showed a remarkable efficiency in terms of *computational time* with respect to Exact Decomposition method, reflecting the power of FGMC simulations. It's not surprising that our benchmark, the Lewis formula, was faster than the other methods since it is not a Monte Carlo procedure and it exploits the Fast Fourier Transform in the integral computation.

ED	**FGMC**	**Lewis**
38.394835 s	3.368685 s	0.389538 s

Table 12: Computational Times for ED, FGMC and Lewis Methods.

We then evaluated four different metrics to understand the computation accuracy of the two methods with respect to Lewis formula. In particular we chose as a measure of the error: Infinitive Norm, Root Mean Squared Error, Mean Absolute Percentage Error and Standard Deviation.

Metric	**ED**	**FGMC**
Norm Infinitive	3.388061e-05	3.357027e-05
Root Mean Squared Error	2.845303e-05	3.031222e-05
Mean Absolute Percentage Error	8.020513e-02	8.056749e-02
Mean Standard Deviation	1.481341e-01	1.473811e-01

Table 13: Metrics for Exact Decomposition and Fast General Monte Carlo with respect to Lewis Formula.

Noticing that the metrics are in the order of maximum $1bp$, we can state that the three prices are very close to each other, proving the accuracy of our procedure in simulating the processes. This result allows us to proceed with pricing the American call option using the paths generated for this process.

The following plots show the price evolutions as *log-moneyness* varies, along with a zoomed-in view to better understand the differences between them.

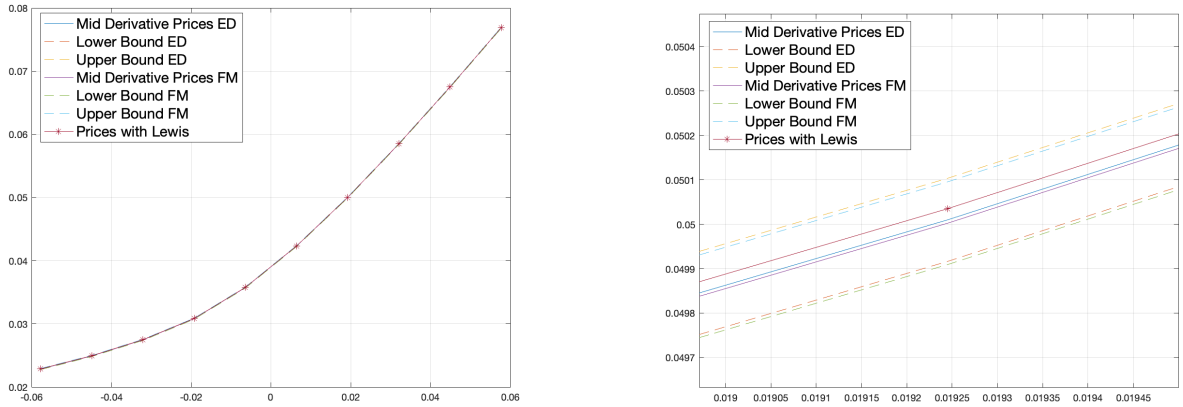

Figure 7: Prices computed with Lewis formula, Exact Decomposition and Fast General Monte Carlo

The plot below illustrates the volatility smiles derived from three pricing methods. The volatility shape is characterized by symmetry due to the input parameters employed in simulating the processes. Varying these parameters we expect to obtain different outcomes, particularly in commodities markets where an asymmetrical shape and a shift towards out-of-the-money values are anticipated.

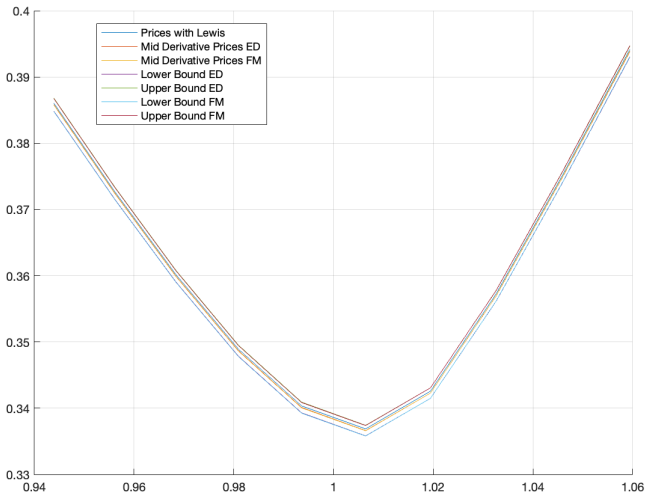

Figure 8: Volatility computed with Lewis formula, Exact Decomposition and Fast General Monte Carlo

6.1.2 OU-TS Finite Activity

For the Finite Activity case, we considered a stability parameter $\alpha = -1$, and in order to achieve a right price computation we should consider the *log forward* returns in (43) to satisfy the martingality condition.

The prices corresponding to the selected moneyness values are shown in the table below.

Moneyness	-0.0577	-0.0449	-0.0321	-0.0192	-0.0064	0.0064	0.0192	0.0321	0.0449	0.0577
ED %	0.9943	1.0135	1.0331	1.0531	1.0735	1.1567	2.3945	3.6179	4.8267	6.0213
FGMC %	1.0282	1.0474	1.0669	1.0869	1.1072	1.1775	2.4154	3.6388	4.8477	6.0424
LEWIS %	0.9938	1.0128	1.0326	1.0525	1.0721	1.1521	2.3939	3.6172	4.8258	6.0206

Table 14: European Call Prices for different values of moneyness with the three different methods. Settings: 10^7 simulations, $M_{fft} = 16$.

Also in this case, the Fast and General Monte Carlo approach showed a better performance than the Exact Decomposition method, obtaining a *computational times* three times lower.

ED	**FGMC**	**Lewis**
9.366223 s	3.030881 s	0.333366 s

Table 15: Computational Times for ED, FGMC and Lewis Methods.

Again we decided to analyze the performance of these pricing methods comparing the same evaluation metrics used in the previous case. The accuracy is almost the same obtained with Finite Variation OU-TS process still below 1 bp.

Metric	**ED**	**FGMC**
Norm Infinitive	4.597324e-05	3.508412e-04
Root Mean Squared Error	1.637146e-05	2.915806e-04
Mean Absolute Percentage Error	8.370443e-02	2.125390e+00
Mean Standard Deviation	1.920336e-01	4.570539e-01

Table 16: Metrics for Exact Decomposition and Fast General Monte Carlo with respect to Lewis Formula.

The graph of the obtained results are below, we observe that the price via Lewis formula is inside the confidence interval of both the two method. Moreover, we notice a strong change in the slope of all three prices near the at-the-money, which is reflected also in the behaviour of the volatility.

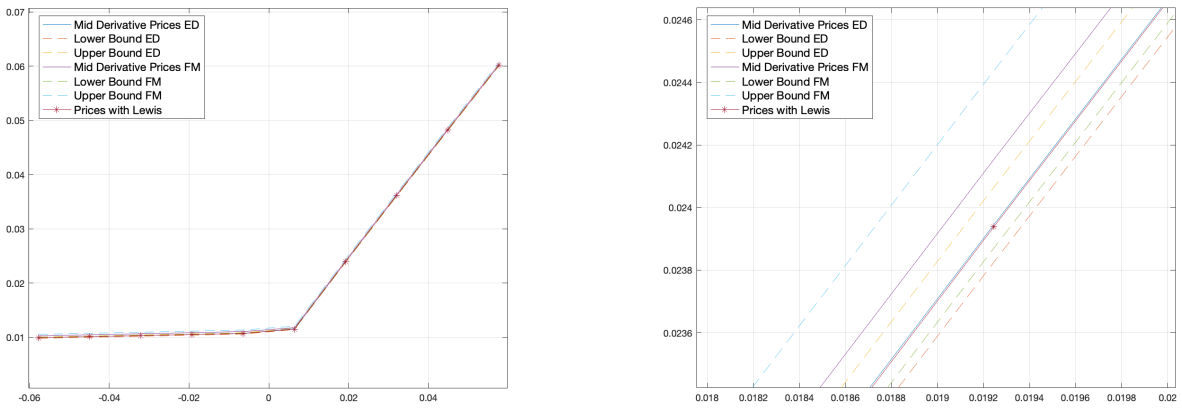

Figure 9: Prices computed with Lewis formula, Exact Decomposition and Fast General Monte Carlo

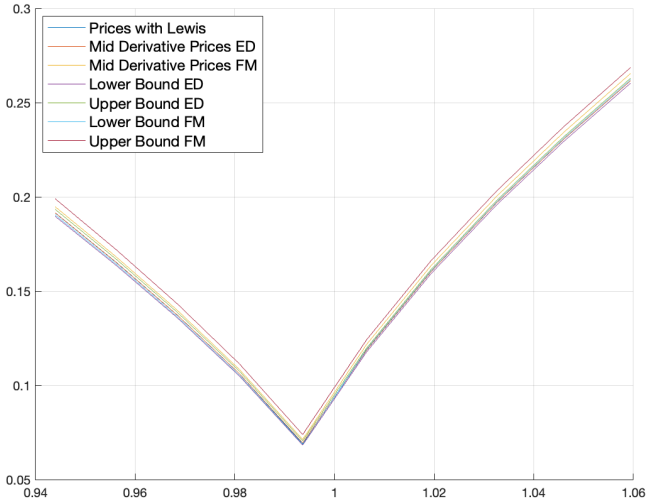

Figure 10: Volatility computed with Lewis formula, Exact Decomposition and Fast General Monte Carlo

We decided not to explore more the pricing methods with this kind of processes since in practice the exponential decay processes are more suitable for this purposes with respect to the power law decay, as stated in Azzone [1]. Furthermore, looking at Figure 1, the jumps in the path seem too large to be used in commodities pricing.

6.1.3 TS-OU Finite Variation

Finally, we explored the same pricing methods using the third process for the simulations: Finite Variation TS-OU with stability parameter $\alpha = 0.5$.

Moneyness	-0.0577	-0.0449	-0.0321	-0.0192	-0.0064	0.0064	0.0192	0.0321	0.0449	0.0577
ED %	0.3965	0.4177	0.4427	0.4740	0.5203	1.1825	2.3639	3.5571	4.7472	5.9295
FGMC %	0.4049	0.4262	0.4512	0.4826	0.5290	1.1911	2.3723	3.5654	4.7555	5.9377
Lewis %	0.4022	0.4236	0.4486	0.4800	0.5263	1.1884	2.3698	3.5630	4.7531	5.9354

Table 17: European Call Prices for different values of *log-moneyness* with the three different methods. Settings: 10^7 simulations, $M_{fft} = 24$.

As done for the previous processes, we compared *computational times* of the three pricing methods. The Fast and General Monte Carlo approach is still more efficient than Exact Decomposition, but in this case we notice that the speed performance is lower than in the previous sections.

ED	**FGMC**	**Lewis**
11.209842 s	6.720375 s	0.016275 s

Table 18: Computational Times for ED, FGMC and Lewis Methods.

Analyzing the same metrics as in the previous cases, we notice a pretty good performance, obtaining for both simulation methods an error of fraction of bps.

Metric	ED	FGMC
Norm Infinitive	5.975272e-05	2.693367e-05
Root Mean Squared Error	5.907610e-05	2.543272e-05
Mean Absolute Percentage Error	7.635743e-01	3.393098e-01
Mean Standard Deviation	8.540314e-02	9.185241e-02

Table 19: Metrics for Exact Decomposition and Fast General Monte Carlo with respect to Lewis Formula.

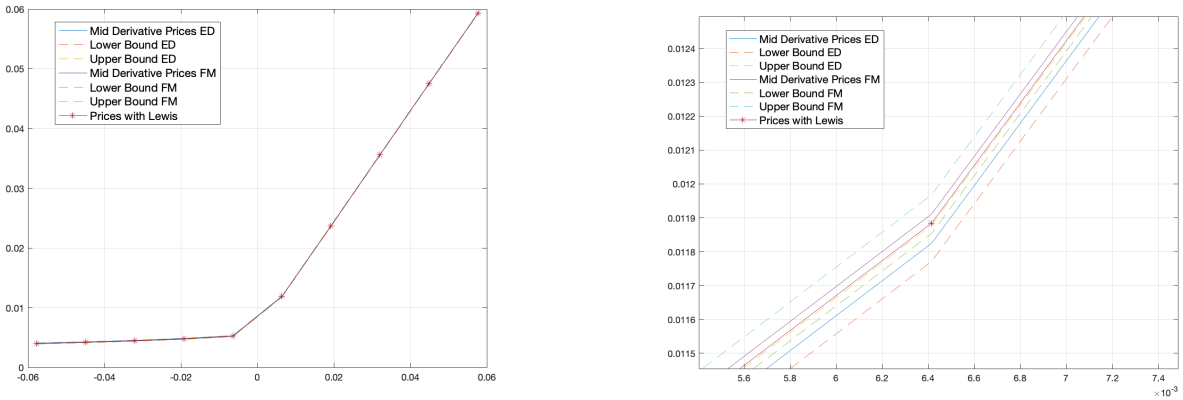

Figure 11: Prices computed with Lewis formula, Exact Decomposition and Fast General Monte Carlo

By looking at the plot of the implied volatilities, we notice a strange shape of the smile. This can be due to the given parameters of stationary distribution, since we are dealing with a TS-OU model. We decided to exploit this behaviour in section 8.

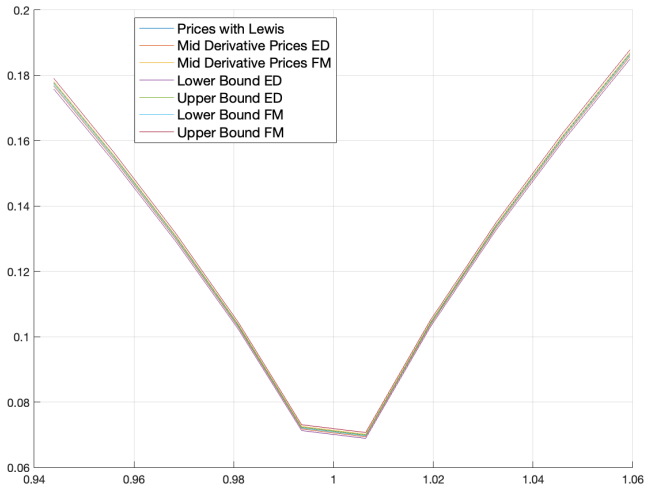

Figure 12: Volatilities computed with Lewis formula, Exact Decomposition and Fast General Monte Carlo

6.2 American Call

In the previous section we focused on European call option, so we were interested in the value of the underlying only at maturity. Switching to path dependent derivatives, such as American options, we would need to discretize the time horizon up to maturity and simulate the evolution of the asset. In general, increasing the density of the time grid enhances pricing accuracy, particularly for American derivatives, as each time node represents a potential early exercise opportunity.

Before pricing the derivative, we have to ensure that the simulations of the underlying remain stable as we increase the time steps in the simulation. The ED algorithm preserves the cumulants, but its computational time and space requirements could make it infeasible. On the other hand, considering FGMC, the challenge with the algorithm is that as the time lag decreases, the CDF of the integrated process $Z_{\Delta T}$ becomes steeper, and the FFT becomes less effective in capturing its shape, even when using the suggested optimal du. As the CDF gets steeper, also the sampling procedure becomes less efficient since there are not enough points in the region of interest.

We noticed that for OU-TS Finite Variation, the FGMC is stable in terms of cumulants up to a 3 days time step, for smaller time lag the simulation moments spike out and the martingale property does not hold anymore. In order to overcome this issue, we recall the characteristic function property:

$$\phi_X(cu) = \phi_{cX}(u)$$

and plugging it into (29), we were able to compute the CDF for the $Z_{\Delta T}$ process scaled by a factor c :

$$CDF_{cZ_{\Delta t}}(x) = R_a - \frac{e^{ax}}{\pi} \int_0^\infty Re\left(\frac{e^{-iux}\phi\left(c(u+ia), \Delta t\right)}{i(u+ia)} \right) du \tag{47}$$

By analyzing Fig.(13), containing the CDF relative to the integrated process Z with time lag of $\frac{1}{12}$, we notice that for scaling factors greater than 1, the CDF got stretched and theoretically the FFT algorithm would be more accurate in integral computation. As expected, all the CDF intersect where the value of the x is the mean, except for the the highest factor 50.

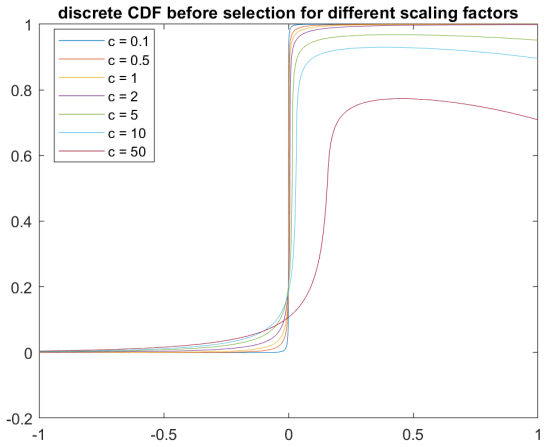

Figure 13: Discrete CDF using FFT method for different values of the scaling factor. Time lag: $\frac{1}{52}$.

Following a qualitative study, we implemented the FGMC algorithm using the optimal parameters and ensured that the values sampled from the CDF were properly scaled by dividing them by the scale factor.

We noticed how the scaling procedure worked well for the case in which we already have a "well shaped" CDF (tipically a time step not smaller than 1 month). On the other hand by exploiting the

scaling factor procedure for time step shorter (cases for which we need it), we experienced some issues. Upon comparing the cumulants for various scaling factor values, we encountered unexpected results: the stretching in the CDF introduced additional sources of error, causing the simulated cumulants to spike out for scale factors greater than 1. As you can see from figure (13), as the scaling factor increases, the CDF tends to stop growing without even getting closer to 1, this obviously leads to errors. Conversely, in some cases the cumulants matched the theoretical values for scale factors less than 1.

Once we found the maximum possible discretization preserving the cumulants, we delved into the Longstaff-Schwartz algorithm implementation for valuing American options.

The algorithm presented in Longstaff & Schwartz [3], involves a backward procedure to determine the price of an American derivative. At each time node, the realized underlying values are used to estimate the continuation value, specifically they are modelled as a polynomial function of the prices:

$$CV_j = \sum_{k=1}^{L} \alpha_{k,j} \phi_k(S_J) \tag{48}$$

where:

- CV is the continuation value
- j is the node in time grid
- L is the grade of the polynomial
- α_k is the k-th coefficient of the polynomial
- ψ is the polynomial bases

Choosing a value of $L << N_{sim}$, we had to solve a least square system in order to determine the weights α for each time step and consequently find an approximation of the continuation value. In our algorithm implementation, we set L to 3, simplifying the polynomial function and the solving of the system. For the polynomial basis, we initially used the canonical polynomial basis but, for the sake of generality, we also tried the orthogonal Legendre basis, which did not result in significant changes to the results. Our implementation procedure can be easily extended to higher grade of polynomial basis and different kind of basis as suggested in [3]; in our case there was not evidence to delve into it.

After comparing the Intrinsic Value with the fitted Continuation Value, the algorithm repeats the same computations back to the first time node. Finally, we averaged the results from different simulations to determine the derivative's price and its confidence interval.

In Table (20) we show the computed prices for American call option with 1 year maturity and moneyness values in $[-0.2, 0.2]\sqrt{T}$. We explored two different values of the discretization, $M = 52$ and $M = 120$, since they both provided a time step small enough to preserve the cumulants of the process, which is a Finite Variation OU-TS in the FGMC.

In the forward dynamics we set $r = 0$, and considering no storage costs, similar to no dividends case in the equity world, we expected the European price to match the American ones, as we were dealing with a underlying which is a martingale and a not decreasing, convex payoff. As confirmed by the table's results and by Fig. (14), depicting the prices for American and European calls, we have very closed values, confirming our initial hypothesis. The small difference in prices, in our opinion, can be due to the Longstaff Schwartz approach to the pricing of the American derivative, in particular in the Least Squares, as the American prices are stable when increasing the density in the time grid.

Moneyness	**M**	-0.2000	-0.1000	0.0000	0.1000	0.2000
ED (A) %	52	12.2531	15.2440	18.8612	23.0639	27.7506
FGMC (A) %	52	12.1199	15.1027	18.7101	22.8974	27.5762
ED (A) %	120	12.1600	15.1535	18.7729	22.9680	27.6441
FGMC (A) %	120	12.1938	15.1911	18.8216	23.0291	27.7115
Lewis (EU) %	–	11.6766	14.6057	18.1473	22.2793	26.8923

Table 20: European Call prices using Lewis and American Call Prices for different values of *log-moneyness* with the two different methods ED and FGMC ($T = 1$). Settings: 10^6 simulations, $M_{fft} = 20$, Time discretization $= \frac{1}{52}, \frac{1}{120}$, LS algorithm: Polynomial grade=2, canonical polynomial basis.

By looking at the computational times, we can clearly notice how relevant becomes the choice of a fast algorithm as FGMC with respect to the Exact Decomposition.

M	**ED**	**FGMC**
52	106.196770 s	10.201602 s
120	253.371947 s	21.256455 s

Table 21: Computational Times for ED, FGMC Methods for different choice of M.

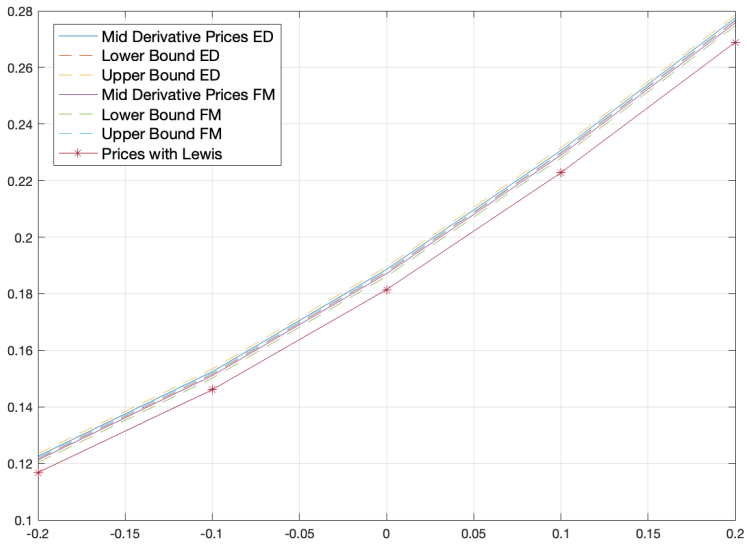

Figure 14: American call prices (maturity T=1) computed via Exact decomposition and FGMC using Longstaff Schwartz algorithm compared to European call computed via Lewis formula. Settings: $M = 120$, $N_{sim} = 10^7$

7 Gaussian OU Process

In the initial overview, we find important to explain the reasons behind the choice of this kind of processes for capturing the dynamics of energy and commodities market. Alternatively, one can select a simple Gaussian Ornstein-Uhlenbeck process to maintain the mean-reverting behavior observed in the market. While choosing a Gaussian distribution does not capture jumps and heavy tails, it offers a

significant advantage in terms of computational efficiency. In this case, the driver in the mean-reverting dynamics is a standard Brownian Motion (BM), coming from the SDE:

$$\begin{cases} dX_t = -bX_t dt + \sigma dW_t \\ X_0 \quad known \end{cases} \tag{49}$$

The unique strong solution to this equation is:

$$X_t = X_0 e^{-bt} + \sigma \int_0^t e^{-b(t-s)} dW_s \tag{50}$$

The main difference with respect to the solution (2) for a generic Lévy process is that the stochastic integral is distributed as a Gaussian. Thanks to this fact by exploiting some computation based on Itô calculus, we can find the following equivalent in law strong solution:

$$X_t = X_0 e^{-bt} + \frac{\sigma e^{-bt}}{\sqrt{2b}} \cdot W_{e^{2bt}-1} \tag{51}$$

In the numerical procedure it is enough to simulate a gaussian randon variable (g) with zero mean and unitary variance, since it holds this simple equivalence in law for the BM

$$W_{e^{2bt}-1} \overset{\mathcal{L}}{=} \sqrt{e^{2bt}-1}\, g \tag{52}$$

Thanks to the equivalence of Eq. (51), the computational time is really low. There is no evidence to exploit a Fast General Monte Carlo method increasing the complexity of the simulation procedure without significantly decreasing the computational cost. Obviously, the FGMC described in [2] can be implemented also in this case since we only require the characteristic function of a gaussian random variable. Here we present the plot of some simulations of the Gaussian OU-Process

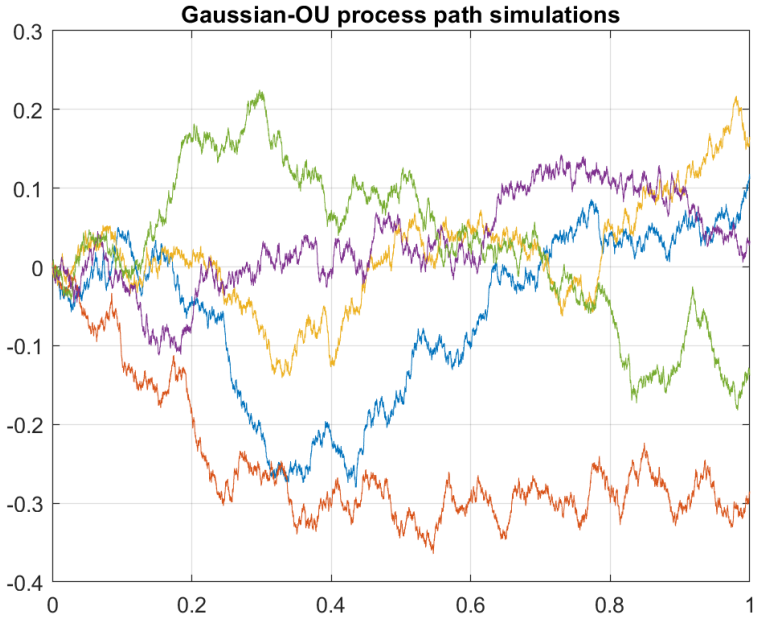

Figure 15: Simulations of a Gaussian-OU process

The Gaussian OU process is an Infinite Variation process as its driver is the Brownian Motion. We can clearly see this feature in the figure (15), by looking at the shape of the different paths.

We report here the comparison between the process' cumulants of for a time horizon of $T = \frac{1}{12}$ and $T = 1$, with parameters $b = 0.1$ and $\sigma = 0.2$. As anticipated, the computational time for this simulation is remarkably low.

	$c_1(X_T) \cdot 1000$		$c_2(X_T) \cdot 1000$		$c_3(X_T) \cdot 1000$		$c_4(X_T) \cdot 1000$		ED
	True	ED	True	ED	True	ED	True	ED	Comp. Time
$T = \frac{1}{12}$	0.000	0.000	3.306	3.306	0.000	-0.000	0.000	0.000	0.759871 s
$T = 1$	0.000	0.000	36.254	36.245	0.000	0.000	0.000	-0.002	0.774490 s

Table 22: Comparison of True and Simulated values of cumulants at different times T (scaled by 1000) for Gaussian-OU. ED algorithm settings: 10^7 simulations.

7.1 European Call

We also performed the pricing procedure for this process both for an European call and an American call. Here the results for a call option with maturity $T = \frac{1}{12}$, the usual set of log-moneyness in $[-0.2, 0.2]\sqrt{T}$ and the parameters of the process $b = 0.1$ and $\sigma = 0.2$.

Moneyness	-0.0577	-0.0449	-0.0321	-0.0192	-0.0064	0.0064	0.0192	0.0321	0.0449	0.0577
Lewis %	0.4890	0.7306	1.0545	1.4726	1.9933	2.6200	3.3506	4.1778	5.0897	6.0716
ED %	0.4883	0.7298	1.0535	1.4715	1.9921	2.6188	3.3495	4.1768	5.0888	6.0708

Table 23: European Call Prices for different values of *log-moneyness* with the two different methods expressed in percentage. Settings: 10^7 simulations, $M_{fft} = 16$.

As expected, the pricing procedure is computationally efficient since in the simulation we are just drawing a normal random variable.

ED	**Lewis**
3.134987 s	0.012915 s

Table 24: Computational Times for ED and Lewis Methods.

By looking at the metrics in the Table (25), we can clearly see how the difference between the prices obtained via simulation and Lewis is completely negligible, just fraction of *bps*.

Metric	**ED**
Norm Infinitive	3.203933e-06
Root Mean Squared Error	1.738349e-06
Mean Absolute Percentage Error	1.283112e-02
Mean Standard Deviation	3.408965e-02

Table 25: Metrics for Exact Decomposition with respect to Lewis Formula.

In the Figure (16) we notice that, as expected, the confidence interval for the European call price contains the prices computed using Lewis formula.

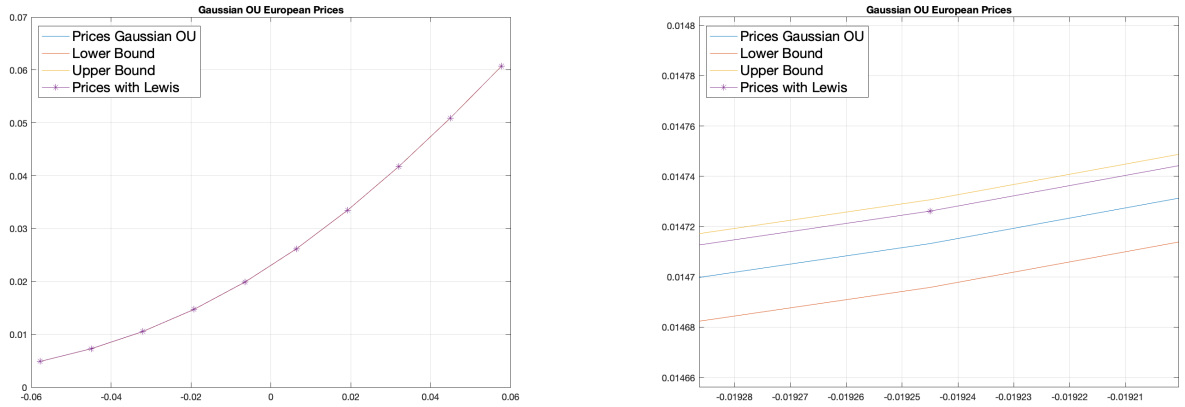

Figure 16: European call prices computed via Exact decomposition and via Lewis formula; T=1/12, b=0.1, $\sigma = 0.2$

7.2 American call

As done for the other model, we deal with a path dependent derivative: the American call. As observed in the section 6.2, we can use as benchmark the European call since we are in presence of no storage cost. In this case, employing an Exact Decomposition procedure to simulate the process proves computationally efficient, allowing us to use a small time step. This approach accurately models the American contract, which permits exercise at any moment during its duration. Since the contract had maturity of 1 year, we exploit a weekly time step (setting M=52), 2-day step (setting M=180) and a daily possible exercise (setting M=365). Here we provide the results obtained with the same parameters as the section for the European call 7.1, moreover we considered 5 levels of moneyness in $[-0.2, 0.2]\sqrt{T}$.

Moneyness	**M**	-0.2000	-0.1000	0.0000	0.1000	0.2000
ED (A) %	52	1.6246	3.8906	7.7558	13.2201	19.7465
ED (A) %	180	1.6322	3.8994	7.7512	13.1945	19.7229
ED (A) %	365	1.6245	3.8891	7.7539	13.1823	19.7239
Lewis (EU) %	-	1.5867	3.7969	7.5846	12.9519	19.4260

Table 26: European Call prices using Lewis and American Call Prices for different values of *log-moneyness* with the ED methods ($T = 1$). Settings: 10^6 simulations, $M_{fft} = 16$.

It is possible to notice that the prices of the American call seem to be stable as we increase the number of possible exercise times (i.e. decreasing the time step). This can be due to the presence of numerical errors in the algorithm of Longstaff Schwart in [3]. However, we obtained a price for the American contract which is above the European benchmark and the difference is up to a few *bps*.

M	**ED** (A)
52	13.950090 s
180	60.044049 s
365	130.024383 s

Table 27: Computational Times for ED methods for different choice of M

The computational cost for pricing is not too high considering that we exploit several time step for the simulation and in each one we should perform a least square procedure.

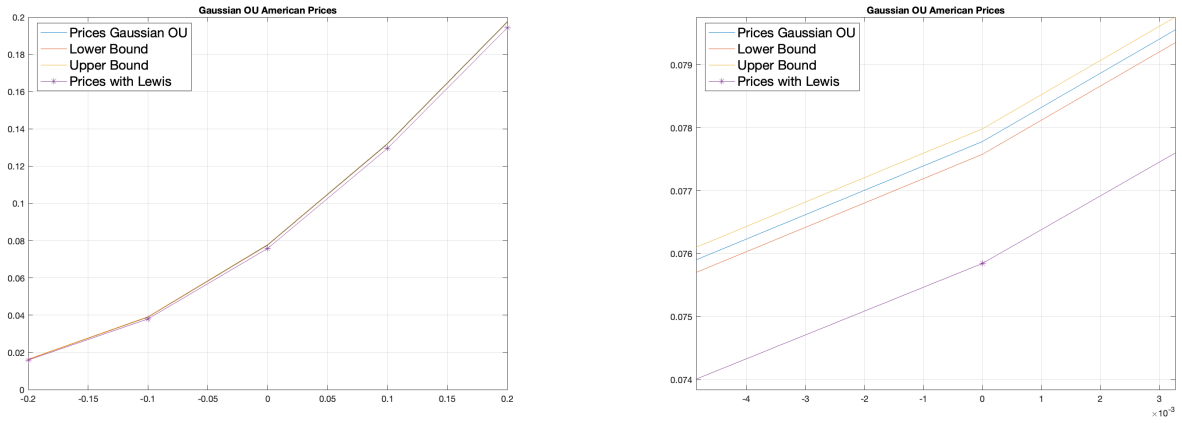

Figure 17: American call prices (maturity T=1) computed via Exact decomposition and Longstaff Schwartz algorithm compared to European call computed via Lewis formula. Settings: $b = 0.1$, $\sigma = 0.2$, $M = 365$

8 Calibration

As we mentioned earlier in the section (6.1), plotting the implied volatility smile for the OU-TS Finite Variation model gives us a reasonable shape. However, when we try to use a TS-OU Finite Variation model, using the given parameters, we end up with an odd shape in the plot for both prices and implied volatilities.

To address this, we decided to calibrate the model using the derivative prices computed using the closed formula of Lewis. For this calibration, we decided to take a more dense grid of moneyness considering 100 levels between $(-0.2, 0.2)\sqrt{T}$. We performed the usual procedure for the calibration using the MATLAB function `lsqnonlin` minimizing the least squares error between the Lewis price with the OU-TS model and the Lewis price with the TS-OU model. We decided to use all the weights equal to 1, because there was not evidence for a different choice. In the table below we report the calibrated parameters:

b_{opt}	α_{opt}	$\beta_{p_{opt}}$	$\beta_{n_{opt}}$	$c_{p_{opt}}$	$c_{n_{opt}}$	$\gamma_{c_{opt}}$
0.7299	0.6450	4.3904	5.1377	0.6226	0.9754	-0.9578

Table 28: Calibrated parameter values

In the plot 18, it is possible to see how the prices overlap one over the other.

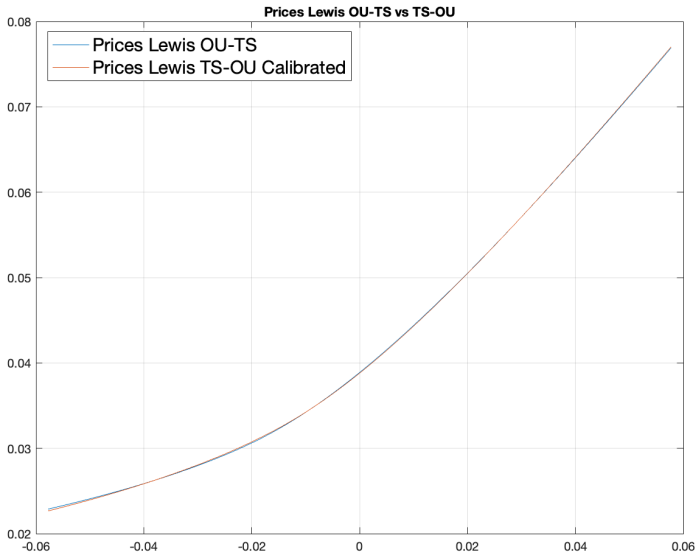

Figure 18: Prices computed with Lewis formula with OU-TS model for the given parameters and with TS-OU model using the calibrated parameters in table 28

In order to check the procedure we also report the implied volatility plot, which allows to see that with this choice of the parameters the smile gets an usual shape also for the TS-OU model.

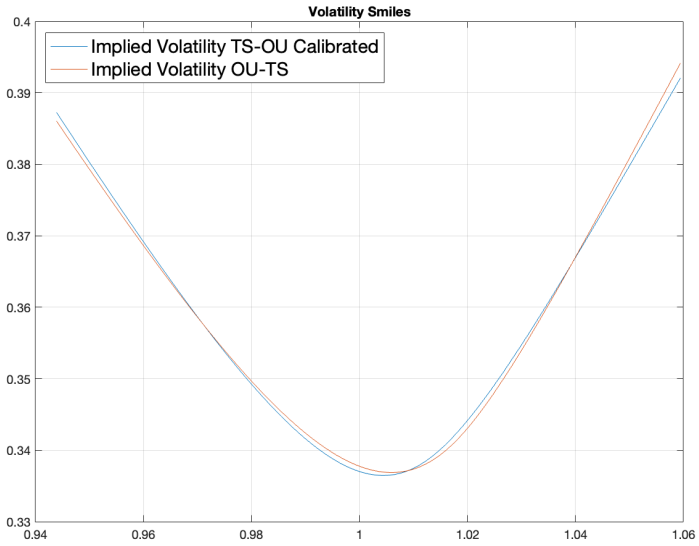

Figure 19: Implied volatilities for the corresponding prices computed with Lewis formula with OU-TS model for the given parameters and with TS-OU model using the calibrated parameters in table 28

Once we found the parameters in Table (28), it is possible to perform the whole simulation of the processes both with Exact Decomposition and Fast and General Monte Carlo Method, finding good matching with the Lewis benchmark. The procedure done for calibrating the TS-OU model can be performed also for the OU-TS Finite Activity, but, since in the literature the processes with a exponential law decay are preferred, we decided not to go deeper.

9 Conclusions

We were tasked with the implementation of a library for simulating OU processes with a non Gaussian driver, and this allowed us to understand their main properties and features, which motivate the choice of them for energy price modelling. After implementing the algorithms for the Exact Decomposition simulations, we tried to follow the idea in [2] in order to increase the performances. We got evidence in stating that Fast and General Monte Carlo is a faster and accurate algorithm than the Exact Decomposition, especially in path simulation with uniform time lag. The reason is the fact that the CDF computation needs to be done only once, after which the method requires merely sampling from a uniform distribution and inverting the CDF. However, the FGMC algorithm is limited by the time step size, as the CDF becomes too steep to be effectively estimated using the proposed procedure.

Our discussion and our implementation rely on Finite Variation processes, as all the proposed articles address to this kind of processes. For Infinite Variation processes, there are no specific algorithms for their simulation, except for Gaussian OU processes. For these ones, we presented the strong solution and successfully simulated their dynamics, as it is sufficient to sample Gaussian random variables.

10 Annex

10.1 Computational Environment

All numerical experiments were performed using MATLAB R2021a on a MacBook Pro with Apple M1 Chip, 8 GB RAM, macOS Monterey 12.3.1.

10.2 Errata Corrige

In this section we report the list of typos found in the papers used for this project, together with the correction we decided to bring.

- In the simulation algorithm of Sabino [4] the correct form of the expression $\tilde{\beta}_m$ is $\tilde{\beta}_m = \beta_m e^{bu_m dt}$, adding a dt to the exponent.
- Again in the same algorithm of Sabino [4] the correct scale parameter for Gamma distribution is $-\alpha_p$.
- In the BCTS cumulants computation we decided to add a $(-1)^k$ factor to the expression of the negative jump side in order to match the notation in Sabino [4].
- In Sabino [5] the Eq.(23) reads: $\lambda_p = c_p \Gamma(-\alpha_p)\beta_p^{\alpha_p}$.
- In the simulation algorithm of Sabino [5] the correct expression of the V distribution is $(1 + (a^{-\alpha} - 1)U)^{\frac{1}{\alpha}}$.
- In our MATLAB function *ctsCumulants* we added a b that was missing in the exponential of OU-TS expression.

10.3 MATLAB profiler

We provide a brief summary for the MATLAB run, showing the most expensive function in computational time terms. The high computational cost arises because the LS algorithm involves iterating through both the moneyness and the time steps. Unfortunately, this process cannot be vectorized due to the fact that it needs to solve the least squares system at each step.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
runProject_Group4A	1	579.240	0.809	
sim_OU_TS_FinVar_ED	4	455.453	3.924	
simulateV	112	422.495	313.976	
priceAmerican	2	266.546	0.499	
priceEuropean	6	131.105	1.839	
simulateV>@(x)-a^alpha*log(a^alpha)/(1-a^alpha+a^alpha*log(a^alpha))*(a^(-alpha*x)-1)	184011319	108.514	108.514	
priceAmericanLS	3	39.523	39.385	
random	6136	36.631	0.086	
priceAmericanGaussian	1	32.574	1.902	
gamrnd	6116	31.021	30.976	
sim_TS_OU_FinVar_ED	3	19.229	5.287	
sim_OU_TS_FinAct_ED	3	15.286	4.004	
simulateTS	4112	12.436	12.436	
sim_TS_OU_FinVar_FGMC	2	11.537	0.414	
normfit	22	10.617	8.702	
integralViaFFT_CDF	7	9.383	4.309	
compCumulants	40	7.207	0.022	
moment	160	7.118	6.747	

Figure 20: MATLAB Functions Computational Times

References

[1] Michele Azzone and Roberto Baviera. A fast Monte Carlo scheme for additive processes and option pricing. *Computational Management Science*, 20(1):31, 2023.

[2] Roberto Baviera and Pietro Manzoni. Fast and general simulation of lévy-driven ou processes for energy derivatives, 2024.

[3] Francis Longstaff and Eduardo Schwartz. Valuing american options by simulation: A simple least-squares approach. *Review of Financial Studies*, 14:113–47, 02 2001.

[4] Piergiacomo Sabino. Pricing energy derivatives in markets driven by Tempered Stable and CGMY processes of Ornstein–Uhlenbeck type. *Risks*, 10(8):148, 2022.

[5] Piergiacomo Sabino and Nicola Cufaro Petroni. Fast simulation of tempered stable ornstein–uhlenbeck processes. *Computational Statistics*, 37:1–35, 02 2022.