Lévy Processes and Stochastic Volatility

Richard Lo
Part C Maths, MMath Mathematics

Supervisor: Professor Matthias Winkel

Trinity Term, 2020

Acknowledgements

Left blank for examination anonymity.

Length of the Dissertation

The word count (7265 as of 04/05/2020) for the dissertation was conducted using "TeXcount" and excludes table of contents, all mathematical equations and symbols, diagrams, tables, bibliography and the texts of computer programs.

Editing Note

The document has been edited to include the author's name, and to add pdf hyperlinks: it is not the same copy that has been submitted for examination.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Aims

It is widely known that the Black-Scholes model fails to accurately price options due to counter-factual assumptions. Consequently, alternative mathematical models have been investigated. One hopes to yield tractable models, based on Brownian motion's close relatives, that can be applied in industry. This dissertation is intended for an audience with similar background knowledge in the following courses:

1. A4 Integration;
2. A8 Probability;
3. B8.1 Measure, Probability, and Martingales;
4. B8.2 Continuous Martingales and Stochastic Calculus;
5. B8.3 Mathematical Models of Financial Derivatives;

hence there is no need to define objects such as *martingale*, *arbitrage*, *option*, and so on. The scope is broken into several parts. One starts by introducing suitable stochastic processes that can describe stochastic volatility models and investigates model calibration through *European* calls. After that, one explores multiple methods of exotic option pricing.

1.2 Financial Assumptions

One follows the same assumptions presented in the B8.3 course and [33]. To name a few: there is a risk-less investment that grows at some constant, continuously compounded interest rate $r > 0$; there is a frictionless market; and there are no arbitrage opportunities. Additionally, one assumes no dividends are paid to simplify computation, and traded assets are positively valued in spite of the April 2020 oil crisis [37].

1.3 Black-Scholes

Recall from [5]/[16]/[20] that Black-Scholes models the underlying $S = (S_t)_{t\geq 0}$ via

$$\mathrm{d}S_t = S_t(\mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t) \qquad S_0 > 0 \tag{1.1}$$

where $W = (W_t)_{t\geq 0}$ is standard $\mathbb{P}$-Brownian motion, μ is the constant average growth rate of the underlying, and $\sigma > 0$ is the constant volatility.

Theorem 1.1. Suppose a European option has an integrable payoff function $P(S_T)$ at maturity T. Denote its time-t value by $V_t = V(S_t, t)$. If $V(x, t)$ is twice continuously-differentiable then

$$\mathrm{e}^{-rt}\,V(S_t, t) = \mathrm{e}^{-rT}\,\mathbb{E}^{\mathbb{Q}}[P(S_T)|S_t] \tag{1.2}$$

where $\mathbb{Q}$ is a measure that is mutually absolutely continuous with $\mathbb{P}$,

$$S_T \overset{\mathbb{Q}}{\sim} S_t \exp((r - \tfrac{1}{2}\sigma^2)(T - t) + \sigma(W_T^{\mathbb{Q}} - W_t^{\mathbb{Q}})) \tag{1.3}$$

is the risk-neutral price process, and $W^{\mathbb{Q}}$ is a $\mathbb{Q}$-Brownian motion. Explicitly, (1.2) is

$$V(S_t, t) = \frac{\mathrm{e}^{-r(T-t)}}{\sqrt{2\pi\sigma^2(T-t)}} \int_0^\infty \frac{P(x)}{x} \exp\big(-\tfrac{1}{2}d_*^2\big)\,\mathrm{d}x \tag{1.4}$$

where

$$d_* = \frac{\log(x/S_t) - (r - \frac{1}{2}\sigma^2)(T-t)}{\sqrt{\sigma^2(T-t)}}.$$

Proof. Consider a Delta Hedging argument: i.e. construct a risk-free portfolio given by $\mathcal{P}_t = V_t - \Delta_t^S S_t$ where an option worth V_t is purchased and Δ_t^S stocks are sold at time t. That is,

$$\mathrm{d}\mathcal{P}_t = \Big(\frac{\partial V}{\partial t}(S_t, t) + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial x^2}(S_t, t)\Big)\,\mathrm{d}t + \Big(\frac{\partial V}{\partial x}(S_t, t) - \Delta_t^S\Big)\,\mathrm{d}S_t$$

by Itô's Lemma. Imposing the $\mathrm{d}S_t$ coefficient vanishes, one ensures that the portfolio is instantaneously risk-free and $r\mathcal{P}_t\,\mathrm{d}t = \mathrm{d}\mathcal{P}_t$ by no-arbitrage. Rearranging this yields the Black-Scholes PDE

$$\frac{\partial V}{\partial t}(x,t) + \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 V}{\partial x^2}(x,t) + rx\frac{\partial V}{\partial x}(x,t) - rV(x,t) = 0, \qquad x \in (0,\infty), t \in [0, T)$$

and terminal condition $V(x, T) = P(x)$ for $x > 0$. Substituting $\mathrm{e}^{r(T-t)}\,V(x, t)$ into the PDE gives (1.2) and (1.3), by Feynman-Kăc's formula. To conclude, observe

$$Z = W_T^{\mathbb{Q}} - W_t^{\mathbb{Q}} \sim \mathcal{N}(0, T - t)$$

so that (1.2) can be written as (1.4) by using the push-forward measure $\mathbb{Q} \circ Z^{-1}$. $\square$

It is dangerous to use Black-Scholes without understanding its limitations. Moreover, one can see Black-Scholes fails by performing a reality check with empirical data. The following figures present several flavours of visualisation methods.

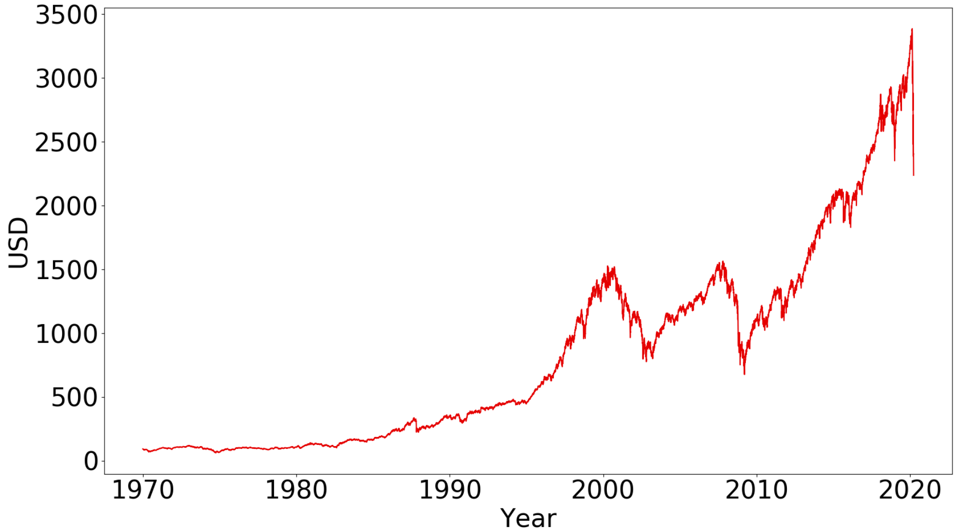


Figure 1.1: 50 years of historical S&P500 daily (closing) stock prices [36]. One can see the early 2000s recession, the 2008 recession, and the 2020 recession due to COVID-19.

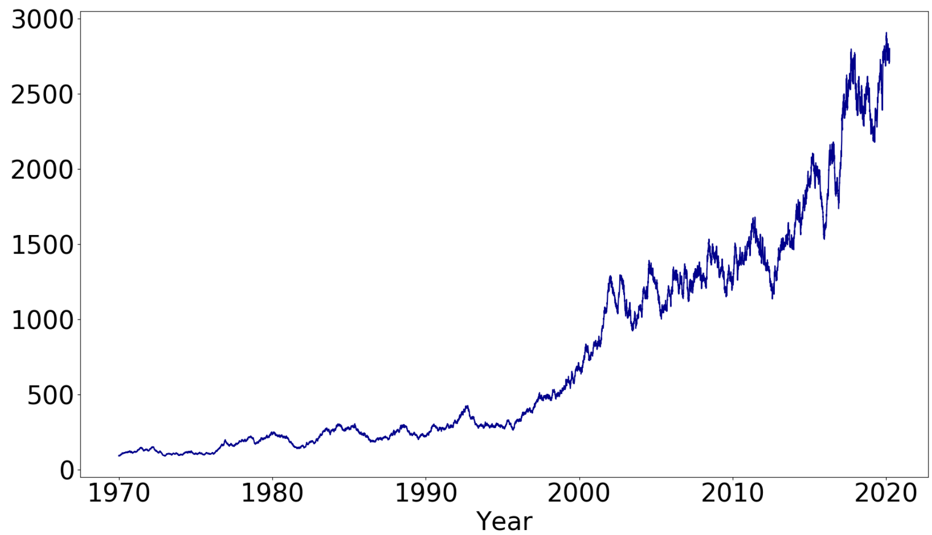


Figure 1.2: A simulation of Geometric Brownian motion subject to the average drift and average volatility in Figure 1.1 in order to mimic the S&P500 data.

Figure 1.1 and Figure 1.2 expose discrepancies. Although the exponential shape looks correct, the latter does not appear to capture the correct stochastic behaviour.

Table 4.1 in [33] suggests log-returns are negatively skewed and have higher kurtosis than any typical Gaussian distribution appearing in Black-Scholes. If log-returns are indeed Gaussian then a QQ-plot would admit a clear straight line.

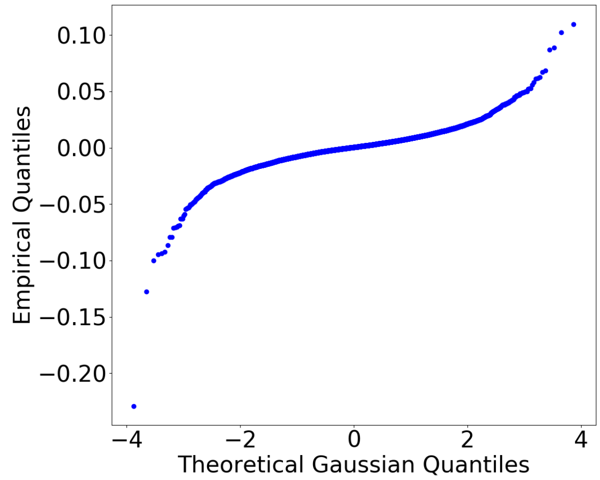


Figure 1.3: The last 50 years of S&P500 daily log-returns [36] against Gaussian quantiles, plotted on a QQ-plot. If one excludes outliers then over 12500 data points remain to provide overwhelming evidence that a straight line is not supported.

The evolution of log-returns suggest volatility should be stochastic and admit *mean-reversion*.

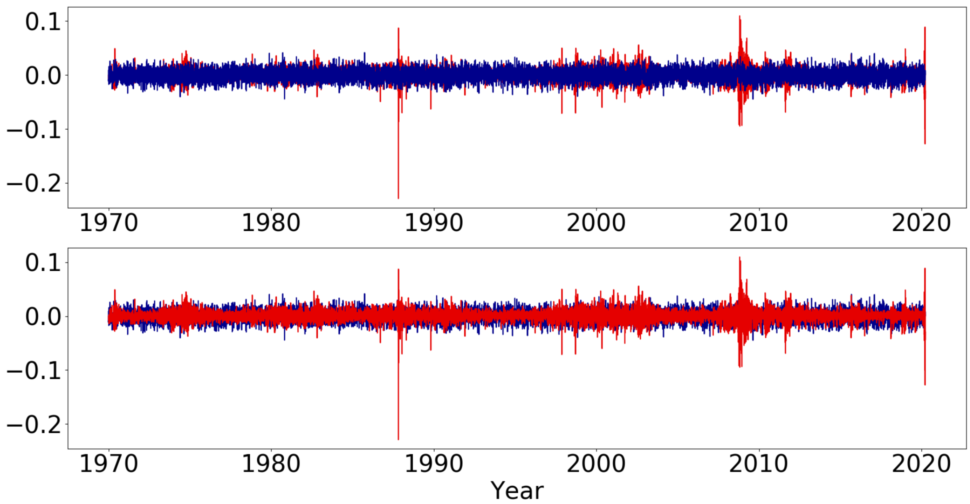


Figure 1.4: An overlay of S&P500 daily log-returns [36] (red) and white noise (blue), where the white noise mimics data by having variance equal to average volatility.

Crucially, empirical evidence does not support the idea that the log-returns admit a Gaussian distribution, and that volatility stays constant! One must relax these assumptions to find a better model.

Chapter 2

Stochastic Processes

The construction of viable models begins by seeking alternatives to Brownian motion, the building block of Black-Scholes. Assume that some probability space $(\Omega, \mathscr{F}, \mathbb{P})$ is given and that all mentioned objects (sets and functions) are Borel measurable. To start off, one cites [35] and observes that Brownian motion is "nice" in the following sense.

Definition 2.1. Given some process $X = (X_t)_{t\geq 0}$ that is adapted to a filtration $\mathbb{F} = (\mathscr{F}_t)_{t\geq 0}$, it has *independent increments* if $(X_{t+s} - X_s)$ is independent of $\mathscr{F}_s$ for every $s, t \geq 0$. It has *stationary increments* if $(X_{t+s} - X_t) \sim X_s$ for every $s, t \geq 0$.

One keeps these properties of Brownian motion, discarding the Gaussian character and a.s. continuity, to obtain the following.

Definition 2.2. Suppose an adapted, $\mathbb{R}^d$-valued, stochastic process $X = (X_t)_{t\geq 0}$ is continuous in probability. If X has a.s. càdlàg paths with independent and stationary increments then it is a *Lévy process*. Moreover, if $t \mapsto X_t$ is non-decreasing then it is a *subordinator*.

Remark. All continuous Lévy processes reduces to Brownian motion with drift, so it is vital to analyse jumps to appreciate Lévy processes in full generality.

2.1 Poisson Point Processes

Definition 2.3. Let $D \subseteq \mathbb{R}^d$ and $\mathscr{B}(D)$ denote its Borel sets. A random measure $N\colon \mathscr{B}(D) \to \mathbb{N}_0$ satisfying

(a) $N(A) \sim \mathrm{Poi}(\int_A \lambda(x)\,\mathrm{d}x)$ and $\lambda\colon D \to [0, \infty)$ whenever $A \subseteq D$;

(b) $N(A_1), \ldots, N(A_k)$ are independent whenever $A_1, \ldots, A_k \subseteq D$ are disjoint;

is a *Poisson counting measure* with *intensity measure* $\Lambda(A) = \int_A \lambda(x)\,\mathrm{d}x$, where λ is the (locally integrable) *intensity function*.

Example 2.4. Let $D = [0, \infty)$, and λ be constant. Then $t \mapsto N((0, t])$ is a Poisson process: a Lévy process with Poisson distributed increments.

Definition 2.5. A random countable subset $\Pi \subseteq D$ is a *spatial Poisson process* if the function $N\colon A \mapsto \#\Pi \cap A$ is a Poisson counting measure.

Definition 2.6. Let ν be locally finite on $D_* \subseteq \mathbb{R}^{d-1}\backslash\{0\}$ and $(\Delta_t)_{t\geq 0}$ be a process in $D_* \cup \{0\}$ such that, for $0 \leq a < b$ and $A_* \subseteq D_*$,

$$N((a,b] \times A_*) = \#\{t \in (a,b]\colon \Delta_t \in A_*\}$$

is a Poisson counting measure with intensity $\Lambda((a,b] \times A_*) = (b-a)\nu(A_*)$. Then $(\Delta_t)_{t\geq 0}$ is a *Poisson point process* with intensity measure ν.

The following work by [35] provides a guide to the construction of point processes in 1-dimension.

Lemma 2.7. Suppose $(Z_m)_{m\in\mathbb{N}}$ is an increasing sequence of $[0,\infty)$-valued random variables. Then there exists a $[0,\infty]$-valued random variable Z such that $Z_m \to Z$ almost surely. In particular,

$$\mathbb{E}[\exp(\gamma Z_m)] \to \mathbb{E}[\exp(\gamma Z)]$$

as $m \to \infty$ for every $\gamma \neq 0$.

Proof. [35, Lemma 21]. $\square$

Lemma 2.8 (Thinning)**.** Let $X, Z_1, Z_2, \ldots$ be independent with $X \sim \mathrm{Poi}(\lambda)$ and $Z_1, Z_2, \ldots$ be i.i.d. categorical $p_1, \ldots, p_\kappa$ random variables taking values in $\{1, \ldots, \kappa\}$. Define for $j = 1, \ldots, \kappa$,

$$X_j = \delta_j(Z_1) + \delta_j(Z_2) + \cdots + \delta_j(Z_X)$$

where $\delta_j(\kappa)$ is Kronecker's delta. Then $X_j \sim \mathrm{Poi}(p_j\lambda)$ for $j = 1, \ldots, \kappa$, and are independent.

Proof. Condition on some fixed X to see that $(X_1, \ldots, X_\kappa)$ is multinomial.

$$\begin{aligned}
\mathbb{E}[s_1^{X_1} \cdots s_\kappa^{X_\kappa}] &= \sum_{m=0}^{\infty} \mathbb{P}[X = m]\mathbb{E}[s_1^{\delta_1(Z_1)+\cdots+\delta_1(Z_m)} \cdots s_\kappa^{\delta_\kappa(Z_1)+\cdots+\delta_\kappa(Z_m)}] \\
&= \sum_{m=0}^{\infty} \frac{\lambda^m}{m!}\,\mathrm{e}^{-\lambda} \sum_{z_1+\cdots z_\kappa=m} \frac{m!}{z_1!\cdots z_\kappa!} p_1^{z_1} \cdots p_\kappa^{z_\kappa} s_1^{z_1} \cdots s_\kappa^{z_\kappa} \\
&= \sum_{m=0}^{\infty} \frac{\lambda^m}{m!}\,\mathrm{e}^{-\lambda}(p_1 s_1 + \cdots s_\kappa p_\kappa)^m \\
&= \exp\Big(-\lambda \sum_{j=1}^{\kappa} p_j(1-s_j)\Big)
\end{aligned}$$

so the p.g.f. factorises giving independence and one deduces the Poisson distributions as claimed. $\square$

Theorem 2.9. Let Λ be an intensity measure on $D \subseteq \mathbb{R}^d$ and $(I_n)_{n\in\mathbb{N}}$ be a partition of D with $0 < \Lambda(I_n) < \infty$. Then there exists a spatial Poisson process Π with intensity Λ.

Proof. Consider independent random variables $\{N_n, Y_j^{(n)}, Y_j^{(n)}, \dots : j, n \in \mathbb{N}\}$ where

$$N_n \sim \mathrm{Poi}(\Lambda(I_n)) \qquad \mathbb{P}[Y_j^{(n)} \in A] = \frac{\Lambda(I_n \cap A)}{\Lambda(I_n)}$$

and define

$$\Pi_n = \{Y_j^{(n)} \colon 1 \le j \le N_n\} \qquad \Pi = \bigcup_{n\in\mathbb{N}} \Pi_n.$$

Fixing $n \in \mathbb{N}$, appeal to Lemma 2.8 in the following with $X = N_n$ to obtain each property in Definition 2.3.

(a) Given $A \subseteq I_n$, let $\kappa = 2$ and $\delta_1(Z_j) = 1$ if and only if $Y_j^{(n)} \in A$;

(b) Given disjoint subsets $A_1, \dots, A_k \subseteq I_n$, let $\kappa = k+1$ and $\delta_l(Z_j) = 1$ if and only if $Y_j^{(n)} \in A_l$ for $l = 1, \dots, k$. For $l = k+1$, let $\delta_{k+1}(Z_j) = 1$ if and only if $Y_j^{(n)} \in I_n \setminus (A_1 \cup \dots \cup A_k)$.

Hence Π_n is a spatial Poisson process with intensity $\Lambda|_{I_n}$. These properties also extend to D.

(a) By Lemma 2.7, there exists $N(A)$ such that $N_1(A\cap I_1)+\dots+N_n(A\cap I_n) \to N(A)$ as $n \to \infty$. Computing the m.g.f.

$$\mathbb{E}[\mathrm{e}^{\gamma N(A)}] = \prod_{n=1}^{\infty} \mathbb{E}[\mathrm{e}^{\gamma N_n(A\cap I_n)}] = \prod_{n=1}^{\infty} \mathrm{e}^{(\mathrm{e}^{\gamma}-1)\Lambda(A\cap I_n)} = \mathrm{e}^{(\mathrm{e}^{\gamma}-1)\Lambda(A)}$$

where the first equality follows from monotone convergence and independence.

(b) $N(A_1), \dots, N(A_k)$ are independent since $\{N_n(A_l \cap I_n) \colon n \in \mathbb{N}, l = 1, \dots, k\}$ are too.

Hence $N \colon A \mapsto \#\Pi \cap A$ is a Poisson counting measure. $\square$

Corollary 2.10. A Poisson point process in $\mathbb{R}$ with intensity ν exists.

Proof. Let $\Lambda((a,b] \times A_*) = (b-a)\nu(A_*)$ for each $0 \le a < b$ and $A_* \subseteq D_* = \mathbb{R} \setminus \{0\}$. Then by Theorem 2.9 there exists a spatial Poisson process Π and

$$\Delta_t = \begin{cases} 0 & \text{if } \Pi \cap (\{t\} \times D_*) = \varnothing \\ x & \text{if } \Pi \cap (\{t\} \times D_*) = \{(t,x)\} \end{cases}$$

is well-defined by Proposition 19 in [35]. It follows that $(\Delta_t)_{t\ge 0}$ is a Poisson point process in $D_* \cup \{0\}$ with intensity measure ν. $\square$

Theorem 2.11 (Exponential Formula)**.** Suppose $(\Delta_t)_{t\geq 0}$ is a Poisson point process in $\mathbb{R}$ with locally finite intensity measure ν on $\mathbb{R}\setminus\{0\}$. Let $(I_n)_{n\in\mathbb{N}}$ partition $\mathbb{R}\setminus\{0\}$ such that $0<\nu(I_n)<\infty$. Then, for every $\gamma\in\mathbb{R}$,

$$\mathbb{E}\left[\exp\left(\gamma\sum_{0\leq s\leq t}\Delta_s\right)\right]=\exp\left(t\int_{\mathbb{R}\setminus\{0\}}(\mathrm{e}^{\gamma x}-1)\nu(\mathrm{d}x)\right).$$

Proof. Let $(\Delta_t^+)_{t\geq 0}$ be the restriction of $(\Delta_t)_{t\geq 0}$ to $[0,\infty)$ and $\nu^+:=\nu|_{(0,\infty)}$. Only countably many Δ_t^+ are non-zero by Corollary 2.10. Thus one can consider setting $T_m :=$ "time of the m^{th} jump" to obtain

$$\sum_{0\leq s\leq t}\Delta_s^+=\sum_{m=1}^{\infty}\Delta_{T_m}^+\mathbb{1}_{\{T_m\leq t\}}$$

and can observe $\Pi=\{(T_m,\Delta_{T_m}^+)\colon m\in\mathbb{N}\}$ is a spatial Poisson process with intensity measure $\Lambda((a,b]\times A_*)=(b-a)\nu^+(A_*)$. Define

$$\Delta_t^{(n)}:=\Delta_t^+\mathbb{1}_{\{\Delta_t\in I_n\}}$$

where $(I_n)_{n\in\mathbb{N}}$ partitions $(0,\infty)$ such that $0<\nu^+(I_n)<\infty$. Fix $n\in\mathbb{N}$ and consider the construction of Π in Theorem 2.9. Then

$$\{(T_j,\Delta_{T_j}^+)\colon m\in\mathbb{N},\Delta_{T_j}^+\in I_n\}\sim\Pi_n=\{Y_j^{(n)}\colon 1\leq j\leq N_n\}$$

where $\{N_n,Y_j^{(n)},Y_j^{(n)},\dots : j,n\in\mathbb{N}\}$ are all independent with distribution

$$N_n\sim\mathrm{Poi}(t\nu^+(I_n))\qquad\qquad Y_j^{(n)}\sim\frac{\nu^+(I_n\cap\cdot\,)}{\nu^+(I_n)}$$

in the sense that $\mathbb{P}[Y_j^{(n)}\in(0,t]\times A_*]=\nu^+(I_n\cap A_*)/\nu^+(I_n)$. Hence

$$\sum_{0\leq s\leq t}\Delta_s^{(n)}\sim\sum_{j=0}^{N_n}Y_j^{(n)}\tag{2.1}$$

and one need only compute the following m.g.f. to verify the claim

$$\begin{aligned}\mathbb{E}\left[\exp\left(\gamma\sum_{j=1}^{N_n}Y_j^{(n)}\right)\right]&=\sum_{m=0}^{\infty}\mathbb{P}[N_n=m]\mathbb{E}\left[\exp\left(\gamma\sum_{j=1}^{m}Y_j^{(n)}\right)\right]\\&=\sum_{m=0}^{\infty}\mathbb{P}[N_n=m]\big(\mathbb{E}[\exp(\gamma Y_1^{(n)})]\big)^m\\&=\exp(-t\nu^+(I_n))\sum_{m=0}^{\infty}\frac{1}{m!}\left(t\int_{I_n}\mathrm{e}^{\gamma x}\,\nu^+(\mathrm{d}x)\right)^m\\&=\exp\left(t\int_{I_n}(\mathrm{e}^{\gamma x}-1)\nu^+(\mathrm{d}x)\right).\end{aligned}$$

By Lemma 2.7, the double sum

$$\sum_{n\in\mathbb{N}}\sum_{0\le s\le t}\Delta_s^{(n)}\uparrow\sum_{0\le s\le t}\Delta_s^+$$

converges since the summands are non-negative random variables. Hence

$$\prod_{n=1}^{m}\exp\left(t\int_{I_n}(\mathrm{e}^{\gamma x}-1)\nu^+(\mathrm{d}x)\right)\to\exp\left(t\int_0^\infty(\mathrm{e}^{\gamma x}-1)\nu^+(\mathrm{d}x)\right)$$

as $m\to\infty$. The conclusion of this proof is obtained by the superposition

$$\Delta_t=\Delta_t^+-\Delta_t^-$$

of two independent processes $(\Delta_t^+)_{t\ge0}$ and $(\Delta_t^-)_{t\ge0}$ with intensity measure ν^+ and ν^- respectively. Here, $(\Delta_t^-)_{t\ge0}$ is a point process with non-negative jumps as before and follows the same construction of $(\Delta_t^+)_{t\ge0}$ but

$$\nu^-((a,b])=\nu([-b,-a))$$

for all intervals $(a,b]\subseteq(0,\infty)$. □

Corollary 2.12. If $(\Delta_t)_{t\ge0}$ is a point process with intensity ν on $\mathbb{R}\setminus\{0\}$ then

(a) $\displaystyle\int_{\mathbb{R}\setminus\{0\}}x\nu(\mathrm{d}x)<\infty \quad\Rightarrow\quad \mathbb{E}\Big[\sum_{s\le t}\Delta_s\Big]=t\int_{\mathbb{R}\setminus\{0\}}x\nu(\mathrm{d}x)$

(b) $\displaystyle\int_{\mathbb{R}\setminus\{0\}}x^2\nu(\mathrm{d}x)<\infty \quad\Rightarrow\quad \mathrm{Var}\Big[\sum_{s\le t}\Delta_s\Big]=t\int_{\mathbb{R}\setminus\{0\}}x^2\nu(\mathrm{d}x).$

Proof. (a) is similar to Proposition 40(i) of [35]. The second moment is

$$\begin{aligned}
\mathbb{E}\Big[\Big(\sum_{s\le t}\Delta_s\Big)^2\Big]&=\frac{\partial^2}{\partial\gamma^2}\exp\left(t\int_{\mathbb{R}\setminus\{0\}}(\mathrm{e}^{\gamma x}-1)\nu(\mathrm{d}x)\right)\Bigg|_{\gamma=0}\\
&=\frac{\partial}{\partial\gamma}\left[t\exp\left(t\int_{\mathbb{R}\setminus\{0\}}(\mathrm{e}^{\gamma x}-1)\nu(\mathrm{d}x)\right)\int_{\mathbb{R}\setminus\{0\}}x\,\mathrm{e}^{\gamma x}\,\nu(\mathrm{d}x)\right]\Bigg|_{\gamma=0}\\
&=\exp\left(t\int_{\mathbb{R}\setminus\{0\}}(\mathrm{e}^{\gamma x}-1)\nu(\mathrm{d}x)\right)\\
&\qquad\times\left[t\int_{\mathbb{R}\setminus\{0\}}x^2\,\mathrm{e}^{\gamma x}\,\nu(\mathrm{d}x)+\left(t\int_{\mathbb{R}\setminus\{0\}}x\,\mathrm{e}^{\gamma x}\,\nu(\mathrm{d}x)\right)^2\right]\Bigg|_{\gamma=0}\\
&=t\int_{\mathbb{R}\setminus\{0\}}x^2\nu(\mathrm{d}x)+\left(t\int_{\mathbb{R}\setminus\{0\}}x\nu(\mathrm{d}x)\right)^2
\end{aligned}$$

and conclude using $\mathrm{Var}[X]=\mathbb{E}[X^2]-(\mathbb{E}[X])^2$. □

2.2 Lévy Processes

One follows [7], [13], [32], and [35] closely, with an understanding of point processes, to construct Lévy processes through their intimate connection with the following.

Definition 2.13. A probability measure υ is *infinitely divisible* if, for all $m \in \mathbb{N}$ there exists a probability measure $\bar{\upsilon}$ such that υ can be expressed as the m^{th} convolution power of $\bar{\upsilon}$. By uniqueness of the Fourier transform, this means

$$\phi_\upsilon(\xi) = (\phi_{\bar{\upsilon}}(\xi))^m$$

where $\phi_\upsilon, \phi_{\bar{\upsilon}}$ are the characteristic functions of υ, $\bar{\upsilon}$ respectively. A random variable is *infinitely divisible* if its law is infinitely divisible.

Remark. Convolution powers correspond to sums of independent random variables so X is infinitely divisible if, for all $m \in \mathbb{N}$ there are m i.i.d. random variables whose sum has the same distribution as X.

Proposition 2.14. Lévy processes are infinitely divisible.

Proof. Let $X = (X_t)_{t\geq 0}$ be a Lévy process and $m \in \mathbb{N}$. Fixing $t \geq 0$, one may produce the telescoping sum

$$X_t \equiv X_{t_m} - X_{t_{m-1}} + X_{t_{m-1}} - X_{t_{m-2}} + \cdots + X_{t_1} - X_{t_0}$$

where $t_j = tj/m$, and (clearly) $X_0 = 0$. The random variables $(X_{t_j} - X_{t_{j-1}})$ for $j = 1, \ldots, m$ are i.i.d. due to independent and stationary increments. □

It will soon be clear that all Lévy process are constructed from infinitely divisible distributions. Covered by this dissertation are 1-dimensional Lévy processes, leaving the d-dimensional generalisation to [7] and [32].

Theorem 2.15 (Lévy-Khintchine Formula)**.** Let $\Psi : \mathbb{R} \to \mathbb{C}$ be the characteristic exponent of some infinitely divisible probability measure on $\mathbb{R}$. Then there exists a triplet (μ, σ^2, ν) such that for every $\xi \in \mathbb{R}$,

$$\Psi(\xi) = i\mu\xi - \frac{1}{2}\sigma^2\xi^2 + \int_{\mathbb{R}\setminus\{0\}} (\mathrm{e}^{ix\xi} - 1 - ix\xi \mathbb{1}_{\{|x|\leq 1\}})\nu(\mathrm{d}x) \tag{2.2}$$

where $\mu \in \mathbb{R}$, $\sigma^2 \geq 0$, and ν is a measure on $\mathbb{R} \setminus \{0\}$ with

$$\int_{\mathbb{R}\setminus\{0\}} (1 \wedge x^2)\nu(\mathrm{d}x) < \infty. \tag{2.3}$$

Proof. [32, Theorem 8.1(i)]. □

Definition 2.16. The triplet (μ, σ^2, ν) in Theorem 2.15 are called *Lévy-Khintchine characteristics* (LKC). The components are the *drift*, *diffusion component*, and *Lévy measure* respectively. If $\nu(\mathrm{d}x) = g(x)\,\mathrm{d}x$ then g is the *Lévy density*.

Theorem 2.17 (Lévy-Itô Decomposition)**.** Let (μ, σ^2, ν) be LKC. Then there exists a Lévy process $X = (X_t)_{t \geq 0}$ given by

$$X_t = \mu t + \sigma W_t + C_t + M_t \tag{2.4}$$

where

$$C_t = \sum_{0 \leq s \leq t} \Delta_s \mathbb{1}_{\{|\Delta_s| > 1\}}$$

is a (compound Poisson) process of big jumps and

$$M_t = \lim_{\varepsilon \downarrow 0} \Big(\sum_{0 \leq s \leq t} \Delta_s \mathbb{1}_{\{\varepsilon < |\Delta_s| \leq 1\}} - t \int_{\{x \in \mathbb{R} \colon \varepsilon < |x| \leq 1\}} x \nu(\mathrm{d}x) \Big)$$

is a martingale of small jumps compensated by linear drift. Here, $W = (W_t)_{t \geq 0}$ is a standard Brownian motion, independent of a Poisson point process $\Delta = (\Delta_t)_{t \geq 0}$ with intensity measure ν. The characteristic exponent of X_t is $\xi \mapsto t\Psi(\xi)$ where Ψ appears in (2.2).

Proof. One follows the exposition from proofs provided by Theorem 1 in [7] and Theorem 42 in [35]. Existence of W is well-known and Δ can independently be constructed using Corollary 2.10 so it is immediate that $X_t - M_t$ exists. Clearly, $\mu t + \sigma W_t$ has characteristic exponent

$$i\mu t \xi - \frac{1}{2} t \sigma^2 \xi^2$$

and is a Lévy process.

For large jumps, note $\Delta_s \mathbb{1}_{\{|\Delta_s| > 1\}}$ is a point process with integrable intensity measure $\nu_{\text{big}}(\mathrm{d}x) = \mathbb{1}_{\{|x| > 1\}} \nu(\mathrm{d}x)$ by (2.3). So C_t is well-defined with characteristic exponent

$$t \int_{\mathbb{R} \setminus \{0\}} (\mathrm{e}^{ix\xi} - 1) \mathbb{1}_{\{|x| > 1\}} \nu(\mathrm{d}x)$$

due to analytic continuation of Theorem 2.11. It follows that C_t is a Lévy process by recognising $t \mapsto C_t$ is càdlàg, and stationary-independent increments follow from the characteristic function.

For small jumps, introduce a point process $\Delta_s \mathbb{1}_{\{|\Delta_s| \leq 1\}}$ with intensity measure $\nu_{\text{small}}(\mathrm{d}x) = \mathbb{1}_{\{|x| \leq 1\}} \nu(\mathrm{d}x)$. It never jumps simultaneously with $\Delta_s \mathbb{1}_{\{|\Delta_s| > 1\}}$ due to Proposition 19 in [35] and independence follows from Definition 2.3. Define an $(\mathscr{F}_t^\varepsilon)_{t \geq 0}$-adapted process $M^\varepsilon = (M_t^\varepsilon)_{t \geq 0}$ by

$$M_t^\varepsilon := \sum_{0 \leq s \leq t} \Delta_s \mathbb{1}_{\{\varepsilon < |\Delta_s| \leq 1\}} - t \int_{\{x \in \mathbb{R} \colon \varepsilon < |x| \leq 1\}} x \nu(\mathrm{d}x)$$

for every $\varepsilon \in (0, 1)$. By similar arguments, M_t^ε has characteristic exponent

$$t \int_{\mathbb{R} \setminus \{0\}} (\mathrm{e}^{ix\xi} - 1 - ix\xi) \mathbb{1}_{\{\varepsilon < |x| \leq 1\}} \nu(\mathrm{d}x)$$

and is a Lévy process. Since $\mathbb{1}_{\{\varepsilon<|x|\leq 1\}}\nu(\mathrm{d}x)$ is integrable, M_t^ε is a.s. bounded and integrable for all $t \geq 0$. By appealing to independent increments and Corollary 2.12,

$$\mathbb{E}[M_t^\varepsilon|\mathscr{F}_s^\varepsilon] = \mathbb{E}[M_t^\varepsilon - M_s^\varepsilon|\mathscr{F}_s^\varepsilon] + M_s^\varepsilon = \mathbb{E}[M_t^\varepsilon - M_s^\varepsilon] + M_s^\varepsilon = M_s^\varepsilon$$

shows M^ε is a martingale. Similar arguments show $M^\varepsilon - M^\delta$ is also a martingale for any $0 < \delta < \varepsilon < 1$, so that

$$\mathbb{E}\Big[\sup_{0\leq s\leq t}|M_s^\varepsilon - M_s^\delta|^2\Big] \leq 4\mathbb{E}\big[|M_t^\varepsilon - M_t^\delta|^2\big] = 4t\int_{\{x\in\mathbb{R}\colon\, \delta<|x|\leq\varepsilon\}} x^2\nu(\mathrm{d}x)$$

by Doob's L^2-inequality. Hence $\{M_s^\varepsilon\colon s\in[0,t], \varepsilon\in(0,1)\}$ is a Cauchy family as $\varepsilon\downarrow 0$ for the norm

$$\|\{Y_s\colon 0\leq s\leq t\}\| = \big(\mathbb{E}\big[\sup\{|Y_s|^2\colon 0\leq s\leq t\}\big]\big)^{1/2}.$$

Completeness of L^2-space implies $M = (M_t)_{t\geq 0}$ exists as a càdlàg martingale with characteristic exponent

$$t\int_{\mathbb{R}\setminus\{0\}} (\mathrm{e}^{ix\xi} - 1 - ix\xi)\mathbb{1}_{\{|x|\leq 1\}}\nu(\mathrm{d}x)$$

and is a Lévy process. It is also adapted to the sigma-algebra generated by Δ, and is independent of $X - M$.

It is not hard to see that finite sums of independent Lévy processes also yield a Lévy process and finally, X is a Lévy process with the required characteristic exponent $t\Psi(\xi)$ since $\mu + \sigma W$, C, and M are independent. □

Corollary 2.18. There is a bijection between the distinct types of infinitely divisible distributions and Lévy processes. Hence Lévy processes can be uniquely identified with LKC.

Proof. Lévy processes are infinitely divisible by Proposition 2.14. Conversely, suppose X_1 is infinitely divisible. Then it has characteristic exponent Ψ by Theorem 2.15 and the corresponding Lévy process $(X_t)_{t\geq 0}$ exists by Theorem 2.17. □

Corollary 2.19. All continuous Lévy processes are Brownian motions with drift.

Proof. By Theorem 2.17, continuous Lévy processes satisfy $C_t = 0 = M_t$. Imposing continuity is now equivalent to imposing $\nu \equiv 0$. □

Proposition 2.20. A Lévy process is of finite variation if and only if

$$\sigma = 0 \text{ and } \int_{0<|x|\leq 1} |x|\nu(\mathrm{d}x) < \infty$$

where (μ, σ^2, ν) are its LKC. In particular, Lévy processes are semimartingales.

Proof. [13, Proposition 3.9]. Hence, $t \mapsto \mu t + C_t$ is of finite variation and (2.4) defines a semimartingale. □

Example 2.21. The *compound Poisson process* is the only piecewise constant Lévy process (other Lévy processes have infinitely many jumps on any bounded interval: see Theorem 21.2 in [32]) and is given by

$$C_t := Z_1 + \cdots + Z_{N_t} \tag{2.5}$$

for some Poisson process $(N_t)_{t\geq 0}$ with rate $0 < \nu(\mathbb{R}\setminus\{0\}) < \infty$, independent of $\mathbb{R}\setminus\{0\}$-valued random variables $(Z_j)_{j\in\mathbb{N}}$ that are i.i.d. and satisfy

$$Z_j \sim \frac{\nu(\,\cdot\,)}{\nu(\mathbb{R}\setminus\{0\})} \tag{2.6}$$

and (2.3). The characteristic function of C_t is

$$\mathbb{E}[\exp(i\xi C_t)] = \exp\Big(t\int_{\mathbb{R}\setminus\{0\}} (\mathrm{e}^{ix\xi} - 1)\nu(\mathrm{d}x)\Big)$$

by analytic continuing Theorem 2.11. LKC $(\mu, 0, \nu)$ are deduced from Theorem 2.15 where

$$\mu = \int_{[-1,1]\setminus\{0\}} x\nu(\mathrm{d}x).$$

Remark. The following examples are centred infinitely divisible distributions. This is much like discussing a $\mathcal{N}(0,\sigma^2)$ distribution instead of $\mathcal{N}(\mu,\sigma^2)$. One omits this detail since this mean correction is just a drift term.

Example 2.22. The Generalised Hyperbolic distribution $\mathrm{GH}(\alpha,\beta,\delta,\upsilon)$ is discussed in [33] and is defined by its characteristic function

$$\phi_{\mathrm{GH}}(\xi;\alpha,\beta,\delta,\upsilon) = \Big(\frac{\alpha^2-\beta^2}{\alpha^2-(\beta+i\xi)^2}\Big)^{\upsilon/2}\frac{\mathrm{K}_\upsilon(\delta\sqrt{\alpha^2-(\beta+i\xi)^2})}{\mathrm{K}_\upsilon(\delta\sqrt{\alpha^2-\beta^2})}$$

with parameters

$$\begin{aligned} &\delta \geq 0, \quad |\beta| < \alpha \quad \text{if } \upsilon > 0,\\ &\delta > 0, \quad |\beta| < \alpha \quad \text{if } \upsilon = 0,\\ &\delta > 0, \quad |\beta| \leq \alpha \quad \text{if } \upsilon < 0,\end{aligned}$$

and

$$\mathrm{K}_\upsilon(x) = \frac{1}{2}\int_0^\infty \xi^{\upsilon-1}\exp(-\tfrac{1}{2}x(\xi+\xi^{-1}))\,\mathrm{d}\xi \qquad x > 0$$

is the modified Bessel function of the second kind. Some sub-classes include the: Hyperbolic ($\upsilon = 1$), and Normal Inverse Gaussian (NIG, $\upsilon = -1/2$). It is proven in [23] that

$$\alpha \downarrow 0 \qquad \beta = 0 \qquad \delta = r^{1/2} \qquad \upsilon = -r/2$$

is the Student's t-distribution with r degrees of freedom.

Example 2.23. The α-stable distribution (for $\alpha \in (0,2]$) is infinitely divisible and closely related to Gaussian distributions: §3.7 [13]. It is rejected in §7.4.2 [13] on the basis that it has infinite variance. The Tempered Stable distribution is a close relative, with finite variance, and introduced in §4.5 [13]. Equation (4.26) [13] links the first five examples in the following tables. In particular, the CGMY distribution has links to Gaussian distributions!

Distribution	Notation	Support	Parameter Space
Gamma	$\Gamma(at, b)$	$(0, \infty)$	$a, b > 0$
Inverse Gaussian	$\mathrm{IG}(at, b)$	$(0, \infty)$	$a, b > 0$
Tempered Stable	$\mathrm{TS}(at, b, c)$	$(0, \infty)$	$a, b > 0, 0 < c < 1$
VG	$\mathrm{VG}(Ct, G, M)$	$\mathbb{R}$	$C, G, M > 0$
CGMY	$\mathrm{CGMY}(Ct, G, M, Y)$	$\mathbb{R}$	$C, G, M > 0, Y < 2$
NIG	$\mathrm{NIG}(\alpha, \beta, \delta t)$	$\mathbb{R}$	$\alpha, \delta > 0, -\alpha < \beta < \alpha$
Meixner	$\mathrm{Mxn}(\alpha, \beta, \delta t)$	$\mathbb{R}$	$\alpha, \delta > 0, -\pi < \beta < \pi$

Table 2.1: Examples of infinitely divisible distributions. The second column denotes the distribution of the corresponding Lévy processes X_t at time $t > 0$.

Distribution	Characteristic Exponent
Gamma	$-a \log(1 - i\xi/b)$
Inverse Gaussian	$ab - a\sqrt{b^2 - 2i\xi}$
Tempered Stable	$ab - a(b^{1/c} - 2i\xi)^c$
VG	$C \log\left(\dfrac{GM}{GM + (M - G)i\xi + \xi^2}\right)$
CGMY	$C\Gamma(-Y)((M - i\xi)^Y - M^Y + (G + i\xi)^Y - G^Y)$
NIG	$-\delta(\sqrt{\alpha^2 - (\beta + i\xi)^2} - \sqrt{\alpha^2 - \beta^2})$
Meixner	$2\delta \log\left(\dfrac{\cos(\beta/2)}{\cosh((\alpha\xi - i\beta)/2)}\right)$

Table 2.2: Infinitely divisible distributions and the logarithm of their characteristic functions.

Distribution	Drift μ
Gamma	$\frac{a}{b}(1-\mathrm{e}^{-b})$
Inverse Gaussian	$\frac{a}{b}\,\mathrm{erf}\left(\frac{b}{\sqrt{2}}\right)$
Tempered Stable	$\frac{2^c ac}{\Gamma(1-c)}\int_0^1 x^{-c}\exp(-\frac{1}{2}b^{1/c}x)\,\mathrm{d}x$
VG	$\frac{C}{G}(\mathrm{e}^{-G}-1)-\frac{C}{M}(\mathrm{e}^{-M}-1)$
CGMY	$C\Big(\int_0^1 \mathrm{e}^{-Mx}\,x^{-Y}\,\mathrm{d}x-\int_{-1}^0 \mathrm{e}^{Gx}\,\lvert x\rvert^{-Y}\,\mathrm{d}x\Big)$
NIG	$\frac{2\delta\alpha}{\pi}\int_0^1 \sinh(\beta x)K_1(\alpha x)\,\mathrm{d}x$
Meixner	$\alpha\delta\tan(\beta/2)-2\delta\int_1^\infty \frac{\sinh(\beta x/\alpha)}{\sinh(\pi x/\alpha)}\,\mathrm{d}x$

Table 2.3: Infinitely divisible distributions and its corresponding drift which appears in Lévy-Khintchine's formula. All distributions have no diffusion part. That is, $\sigma = 0$.

Distribution	Lévy Measure $\nu(\mathrm{d}x)$
Gamma	$\frac{1}{x}a\,\mathrm{e}^{-bx}\,\mathbb{1}_{\{x>0\}}\,\mathrm{d}x$
Inverse Gaussian	$\frac{a}{\sqrt{2\pi}x^{3/2}\exp(\frac{1}{2}b^2x)}\mathbb{1}_{\{x>0\}}\,\mathrm{d}x$
Tempered Stable	$\frac{2^c ac}{\Gamma(1-c)}x^{-1-c}\exp(-\frac{1}{2}b^{1/c}x)\mathbb{1}_{\{x>0\}}\,\mathrm{d}x$
VG	$\frac{C}{\lvert x\rvert}(\mathrm{e}^{Gx}\,\mathbb{1}_{\{x<0\}}+\mathrm{e}^{-Mx}\,\mathbb{1}_{\{x>0\}})\,\mathrm{d}x$
CGMY	$\frac{C}{\lvert x\rvert^{1+Y}}(\mathrm{e}^{Gx}\,\mathbb{1}_{\{x<0\}}+\mathrm{e}^{-Mx}\,\mathbb{1}_{\{x>0\}})\,\mathrm{d}x$
NIG	$\frac{\delta\alpha\,\mathrm{e}^{\beta x}\,K_1(\alpha\lvert x\rvert)}{\pi\lvert x\rvert}\,\mathrm{d}x$
Meixner	$\frac{\delta\exp(\beta x/\alpha)}{x\sinh(\pi x/\alpha)}\,\mathrm{d}x$

Table 2.4: Infinitely divisible distributions and its corresponding Lévy measure which appears in Lévy-Khintchine's formula.

2.3 Classical Mean Reversion

Classical mean-reverting processes, discussed in [33], use standard Brownian motion $W = (W_t)_{t \geq 0}$ as its driving noise.

Example 2.24. The classical Ornstein-Uhlenbeck process $y = (y_t)_{t \geq 0}$ solves the SDE

$$\mathrm{d}y_t = -\lambda y_t \,\mathrm{d}t + \sigma \,\mathrm{d}W_t \qquad -\infty < y_0 < \infty$$

where $\lambda, \sigma > 0$.

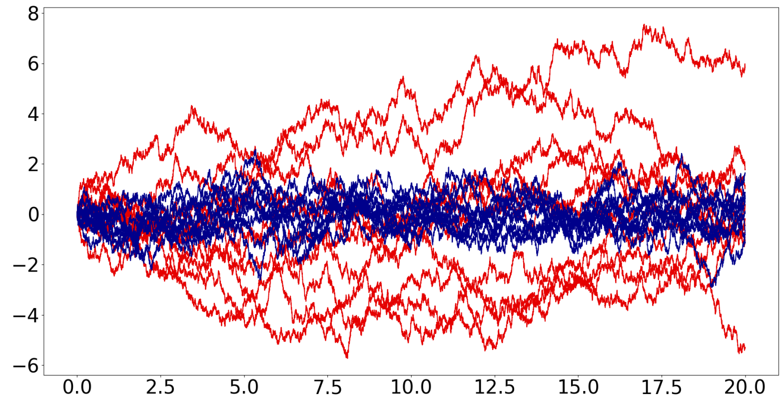


Figure 2.1: Several simulations comparing trajectories of y in blue and W in red.

Example 2.25. The CIR process $y = (y_t)_{t \geq 0}$ solves the SDE

$$\mathrm{d}y_t = \kappa(\eta - y_t) \,\mathrm{d}t + \lambda y_t^{1/2} \,\mathrm{d}W_t \qquad y_0 > 0$$

where $\kappa, \eta, \lambda > 0$ can be interpreted respectively as the rate of mean-reversion, the long-term mean, and the volatility of y. Continuity follows from standard SDE theory and, given $2\kappa\eta \geq \lambda^2$, Corollary 5 in [12] proves positivity. It is a Markov process by Proposition 1 in [12].

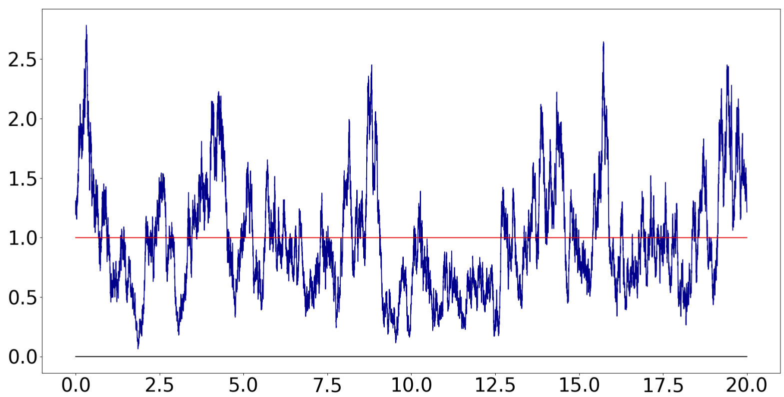


Figure 2.2: A possible sample path of the CIR process (blue) fluctuating around a mean (red). The path never hits zero (black).

2.4 Ornstein-Uhlenbeck Processes

In the spirit of §2.2, one investigates consequences of generalising Ornstein-Uhlenbeck Brownian noise to Lévy processes using [25] and [32].

Definition 2.26. Given $\lambda > 0$, an *Ornstein-Uhlenbeck process* $y = (y_t)_{t\geq 0}$ solves

$$\mathrm{d}y_t = -\lambda y_t \,\mathrm{d}t + \mathrm{d}z_{\lambda t} \qquad -\infty < y_0 < \infty \tag{2.7}$$

where $z = (z_t)_{t\geq 0}$ is the *Background Driving Lévy Process*. The abbreviations are OUP and BDLP respectively.

Remark. The mean may be altered in Definition 2.26 by adding drift to the BDLP:

$$-\lambda y_t \,\mathrm{d}t + \mathrm{d}z_{\lambda t} = (\alpha - \lambda y_t)\,\mathrm{d}t + \mathrm{d}\tilde{z}_{\lambda t}$$

where $(z_t)_{t\geq 0}$ is given by $z_t = \tilde{z}_t + \alpha t/\lambda$.

Theorem 2.27. There exists a unique càdlàg Markov process solving (2.7).

Proof. See Lemma 17.1 in [32] for existence. Now suppose (2.7) has two solutions and $y = (y_t)_{t\geq 0}$ denotes their difference. One claims

$$y_t = \frac{(-\lambda)^n}{(n-1)!}\int_0^t (t-s)^{n-1} y_s \,\mathrm{d}s \tag{2.8}$$

for any $n \in \mathbb{N}$. Proceeding by induction, the base case follows from substituting y into (2.7) since z cancels out. Suppose (2.8) holds for some $n \in \mathbb{N}$. Then

$$\begin{aligned} y_t &= \frac{(-\lambda)^n}{(n-1)!}\int_0^t (t-s)^{n-1}\Big(-\lambda\int_0^s y_{s'}\,\mathrm{d}s'\Big)\,\mathrm{d}s \\ &= \frac{(-\lambda)^n}{(n-1)!}\Big[\frac{\lambda(t-s)^n}{n}\int_0^s y_{s'}\,\mathrm{d}s'\Big]_0^t - \frac{(-\lambda)^{n+1}}{n!}\int_0^t -(t-s)^n y_s\,\mathrm{d}s \\ &= \frac{(-\lambda)^{n+1}}{n!}\int_0^t (t-s)^n y_s\,\mathrm{d}s \end{aligned}$$

follows by substituting the base case into (2.8) and using IBP. Since solutions to (2.7) are càdlàg, $t \mapsto y_t$ is bounded on any finite interval and for any $t \geq 0$ and $n \in \mathbb{N}$,

$$|y_t| \leq \frac{\lambda^n (t-s)^n}{(n-1)!}\sup_{0\leq s\leq t}|y_s|$$

by (2.8). Passing $n \to \infty$, one finds that $y \equiv 0$. □

Remark. Alternatively, §8.1.4 in [13], shows existence of an OUP

$$y_t = \mathrm{e}^{-\lambda t} y_0 + \int_0^t \mathrm{e}^{-\lambda(t-s)}\,\mathrm{d}z_{\lambda s} \tag{2.9}$$

by constructing the integral. When z is of finite variation, the integral is trivially the Riemann-Stieltjes integral.

Lemma 2.28. If $f\colon [0,t] \to \mathbb{R}$ is left-continuous and $z = (z_t)_{t\geq 0}$ is a Lévy process then

$$\mathbb{E}\left[\exp\left(i\int_0^t f(s)\,\mathrm{d}z_s\right)\right] = \exp\left(\int_0^t \psi(f(s))\,\mathrm{d}s\right) \tag{2.10}$$

where ψ is the characteristic exponent of z_1.

Proof. [13, Lemma 15.1]. □

Corollary 2.29. Let $y = (y_t)_{t\geq 0}$ be an OUP. Then

$$\mathbb{E}[\exp(i\xi y_t)|y_0] = \exp\left(i\xi\,\mathrm{e}^{-\lambda t}\,y_0 + \int_0^t \psi(\xi\,\mathrm{e}^{-\lambda(t-s)})\,\mathrm{d}s\right)$$

where ψ is the characteristic exponent of z_1.

Proof. Apply Lemma 2.28 to (2.9). □

In general, the process constructed by Theorem 2.27 is not mean-reverting. However stationary Markov processes are, and Theorem 2.32 motivates the following.

Definition 2.30. Let v be a probability measure with characteristic function ϕ. If for all $\xi \in \mathbb{R}$ and $c \in (0,1)$ there exists some characteristic function ϕ_c satisfying

$$\phi(\xi) = \phi(c\xi)\phi_c(\xi) \tag{2.11}$$

then v is said to be *self-decomposable*. A random variable is *self-decomposable* if its law is self-decomposable.

Theorem 2.31. If an infinitely divisible distribution has Lévy density h such that $x \mapsto |x|h(x)$ is non-decreasing on $(-\infty,0)$ and non-increasing on $(0,\infty)$ then it is self-decomposable.

Proof. [32, Corollary 15.11]. □

Theorem 2.32. Suppose Z is self-decomposable. Then there exists some BDLP such that Z is the stationary distribution of the corresponding OUP.

Proof. [13, Proposition 15.4]. □

Theorem 2.32 gives a sufficient condition for an OUP to have a stationary distribution, but it is not clear how the BDLP is found. This is addressed by [2], [25], and [33].

Lemma 2.33. For every self-decomposable Z there is a Lévy process $z = (z_t)_{t\geq 0}$ satisfying

$$Z \sim \int_0^\infty \mathrm{e}^{-s}\,\mathrm{d}z_s := \lim_{t\to\infty}\int_0^t \mathrm{e}^{-s}\,\mathrm{d}z_s \tag{2.12}$$

where the convergence is in distribution.

Proof. Observe (2.11) corresponds to $Z \sim cZ + Z_c$ for independent Z_c and Z and apply Theorem 3.2 in [25]. □

Remark. Suppose (2.12) holds. Then one can find a relationship

$$g(x) = -h(x) - xh'(x) \tag{2.13}$$

between respective Lévy densities g of z_1 and h of Z. One only needs to verify h is differentiable on $\mathrm{R} \setminus \{0\}$ before appealing to the discussion below Theorem 2.2 in [2].

Theorem 2.34. Suppose (2.12) holds and ψ, κ denotes the respective characteristic exponents of z_1, Z. If $\kappa(\xi)$ is differentiable for $\xi \neq 0$ and $\xi\kappa'(\xi) \to 0$ as $0 \neq \xi \to 0$ then $\psi(\xi) = \xi\kappa'(\xi)$.

Proof. Let $\xi \neq 0$. Then

$$\begin{aligned}
\kappa(\xi) &= \log \mathbb{E}\Big[\exp\Big(i\xi \int_0^\infty \mathrm{e}^{-s}\,\mathrm{d}z_s\Big)\Big] && \text{by (2.12)} \\
&= \int_0^\infty \psi(\mathrm{e}^{-s}\,\xi)\,\mathrm{d}s && \text{by Lemma 2.28} \\
&= \lim_{t\to\infty} \int_0^t \psi(\mathrm{e}^{-s}\,\xi)\,\mathrm{d}s \\
&= \lim_{t\to\infty} \int_\xi^{\mathrm{e}^{-t}\,\xi} \psi(u)\frac{1}{-u}\,\mathrm{d}u && \text{by substituion of } u = \mathrm{e}^{-s}\,\xi \\
&= \int_0^\xi \frac{\psi(u)}{u}\,\mathrm{d}u && \text{since } \mathrm{e}^{-t} \to 0 \text{ as } t \to \infty \qquad (2.14)
\end{aligned}$$

and conclude by applying the fundamental theorem of calculus to (2.14). □

2.5 Integrated Processes

Definition 2.35. The *integrated Ornstein-Uhlenbeck process* (IOUP) is the process $Y = (Y_t)_{t\geq 0}$ where

$$Y_t = \int_0^t y_s\,\mathrm{d}s$$

for an OUP $y = (y_t)_{t\geq 0}$. Similarly, if y is the CIR process then Y defines the *integrated CIR process*.

Example 2.36. In [33, §7.2.1], there is an expression for

$$\varphi_t(\xi; \kappa, \eta, \lambda, y_0) = \mathbb{E}[\exp(i\xi Y_t)|y_0]$$

when y is the CIR process. It is given by

$$\frac{\exp(\kappa^2\eta t/\lambda^2)\exp(2y_0 i\xi/(\kappa + \gamma\coth(\gamma t/2)))}{(\cosh(\gamma t/2) + \kappa\sinh(\gamma t/2)/\gamma)^{2\kappa\eta/\lambda^2}}$$

where $\gamma = (\kappa^2 - 2\lambda^2 i\xi)^{1/2}$.

Proposition 2.37. Suppose $y = (y_t)_{t\geq 0}$ is an OUP with $y_0 > 0$ and the BDLP is a subordinator. Then $t \mapsto Y_t$ is strictly increasing.

Proof. Suppose y satisfies the hypothesis. Then (2.7) is bounded below by

$$\mathrm{d}\tilde{y}_t = -\lambda \tilde{y}_t \,\mathrm{d}t \qquad\qquad y_0 > 0$$

which can be solved deterministically to yield

$$y_t \geq \tilde{y}_t = y_0 \,\mathrm{e}^{-\lambda t} > 0$$

as the required lower bound. □

Proposition 2.38. Let $y = (y_t)_{t\geq 0}$ be a positive OUP with BDLP $z = (z_t)_{t\geq 0}$. Then the IOUP $Y = (Y_t)_{t\geq 0}$ satisfies

$$Y_t = \lambda^{-1}(z_{\lambda t} - y_t + y_0) = \lambda^{-1}(1 - \mathrm{e}^{-\lambda t})y_0 + \lambda^{-1}\int_0^t (1 - \mathrm{e}^{\lambda(s-t)})\,\mathrm{d}z_{\lambda s}$$

and has continuous sample paths.

Proof. Continuity follows from applying MCT to

$$Y_{t_n} = \int_0^\infty y_s \mathbb{1}_{s\in[0,t_n]}\,\mathrm{d}s$$

where $t_n \uparrow t$ or $t_n \downarrow t$ for an arbitrary sequence $(t_n)_{n\in\mathbb{N}}$. Also,

$$\begin{aligned}
\lambda Y_t &= \int_0^t \lambda y_s \,\mathrm{d}s \\
&= \int_0^t \mathrm{d}z_{\lambda s} - \int_0^t \mathrm{d}y_s && (2.15) \\
&= z_{\lambda t} - y_t + y_0 \\
&= z_{\lambda t} - \left(\mathrm{e}^{-\lambda t} y_0 + \int_0^t \mathrm{e}^{\lambda(s-t)}\,\mathrm{d}z_{\lambda s}\right) + y_0 \\
&= (1 - \mathrm{e}^{-\lambda t})y_0 + \int_0^t (1 - \mathrm{e}^{\lambda(s-t)})\,\mathrm{d}z_{\lambda s} && (2.16)
\end{aligned}$$

where (2.15) follows from (2.7). □

Corollary 2.39. If $Y = (Y_t)_{t\geq 0}$ is an IOUP for some positive OUP $y = (y_t)_{t\geq 0}$ then

$$\mathbb{E}[\exp(i\xi Y_t)|y_0] = \exp\left(i\xi\lambda^{-1}(1 - \mathrm{e}^{-\lambda t})y_0 + \int_0^t \psi(\xi\lambda^{-1}(1 - \mathrm{e}^{\lambda(s-t)}))\,\mathrm{d}s\right)$$

where ψ is the characteristic exponent of the BDLP.

Proof. Apply Lemma 2.28 to (2.16). □

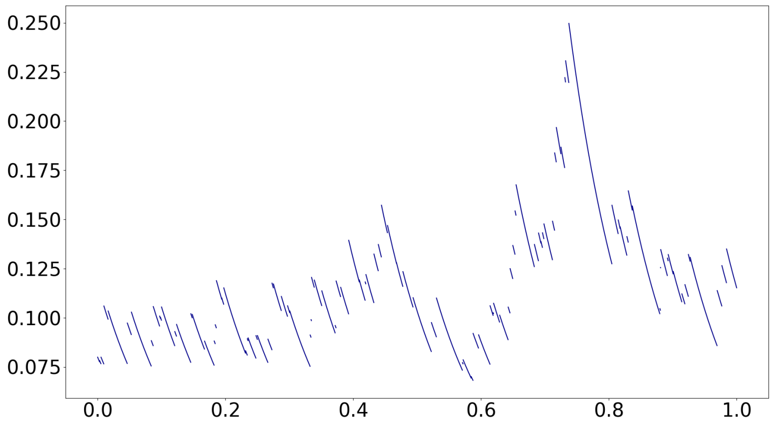


Figure 2.3: A sample path of a Gamma-OUP with $\lambda = 10$, $a = 10$, $b = 100$, and $y_0 = 0.08$. Note the sudden jumps followed by gradual decrease.

The theory in §2.4 and §2.5 is put together by [33] to produce the following example, and it is presented here for further clarity.

Example 2.40 (Gamma-OU)**.** The Gamma distribution has Lévy density

$$h(x) = \frac{1}{x} a \, \mathrm{e}^{-bx} \, \mathbb{1}_{\{x>0\}}$$

so it is self-decomposable by Theorem 2.31. Thus a mean-reverting OUP exists by Theorem 2.27 and Theorem 2.32. The Lévy density of the corresponding BDLP is

$$\begin{aligned} g(x) &= -\frac{1}{x} a \, \mathrm{e}^{-bx} \, \mathbb{1}_{\{x>0\}} - x\Big(-\frac{1}{x^2} a \, \mathrm{e}^{-bx} \, \mathbb{1}_{\{x>0\}} - \frac{1}{x} ab \, \mathrm{e}^{-bx} \, \mathbb{1}_{\{x>0\}} \Big) \\ &= ab \, \mathrm{e}^{-bx} \, \mathbb{1}_{\{x>0\}} \end{aligned}$$

by (2.13) and its characteristic exponent is

$$\begin{aligned} \psi(\xi) &= \xi \frac{\mathrm{d}}{\mathrm{d}\xi} (a \log(b) - a \log(b - i\xi)) \\ &= \frac{i\xi a}{b - i\xi} \end{aligned}$$

by Theorem 2.34. Since one can verify

$$\psi(\xi) = \int_0^\infty (\mathrm{e}^{ix\xi} - 1) g(x) \, \mathrm{d}x$$

it can be deduced from Example 2.21 that $z = (z_t)_{t\geq 0}$ is a compound Poisson process

$$z_t = \sum_{j=1}^{N_t} Z_j$$

where $N = (N_t)_{t\geq 0}$ is a Poisson process with rate a and $Z_j \sim \Gamma(1, b) \sim \mathrm{Exp}(b)$ are i.i.d. random variables. Consequently z is a subordinator, and the characteristic function of the IOUP $Y = (Y_t)_{t\geq 0}$ can be found through Corollary 2.39. If

$$\kappa(\xi) = \varphi_t(\xi; \lambda, a, b, y_0) := \mathbb{E}[\exp(i\xi Y_t)|y_0]$$

then

$$\begin{aligned}
\kappa(\xi) &= \exp\left(i\xi\lambda^{-1}(1-\mathrm{e}^{-\lambda t})y_0 + \int_0^t \psi(\xi\lambda^{-1}(1-\mathrm{e}^{\lambda(s-t)}))\,\mathrm{d}s\right) \\
&= \exp\left(i\xi\lambda^{-1}(1-\mathrm{e}^{-\lambda t})y_0 + \frac{a}{i\xi-\lambda b}\left(b\log\left(\frac{b}{b-i\xi\lambda^{-1}(1-\mathrm{e}^{-\lambda t})}\right) - i\xi t\right)\right)
\end{aligned}$$

since

$$\begin{aligned}
&\int_0^t \psi(\xi\lambda^{-1}(1-\mathrm{e}^{\lambda(s-t)}))\,\mathrm{d}s \\
&\quad= \int_{u_0}^0 \psi(u)\cdot\frac{1}{\lambda u-\xi}\,\mathrm{d}u, \text{ where } u_0 = \xi\lambda^{-1}(1-\mathrm{e}^{-\lambda t}) && (2.17)\\
&\quad= \int_{u_0}^0 \frac{iua}{b-iu}\cdot\frac{1}{\lambda u-\xi}\,\mathrm{d}u \\
&\quad= \frac{a}{\lambda^2 b^2+\xi^2}\int_0^{u_0}\left(\frac{\lambda b^2+ib\xi}{u+ib} + \frac{\xi^2\lambda^{-1}-ib\xi}{u-\xi\lambda^{-1}}\right)\mathrm{d}u && (2.18)\\
&\quad= \left[\frac{\lambda ab^2+iab\xi}{\lambda^2b^2+\xi^2}\log(u+ib) + \frac{a\xi^2\lambda^{-1}-iab\xi}{\lambda^2b^2+\xi^2}\log(u-\xi\lambda^{-1})\right]_0^{u_0} \\
&\quad= \frac{\lambda ab^2+iab\xi}{\lambda^2b^2+\xi^2}\log\left(\frac{b-iu_0}{b}\right) + \frac{a\xi^2\lambda^{-1}-iab\xi}{\lambda^2b^2+\xi^2}\log(1-\xi^{-1}\lambda u_0) \\
&\quad= ab\frac{\lambda b+i\xi}{\lambda^2b^2+\xi^2}\log\left(\frac{b-i\xi\lambda^{-1}(1-\mathrm{e}^{-\lambda t})}{b}\right) - a\xi t\frac{\xi-i\lambda b}{\xi^2+\lambda^2b^2} && (2.19)\\
&\quad= \frac{a}{i\xi-\lambda b}\left(b\log\left(\frac{b}{b-i\xi\lambda^{-1}(1-\mathrm{e}^{-\lambda t})}\right) - i\xi t\right)
\end{aligned}$$

where (2.17) follows from the u-substitution with $u = \xi\lambda^{-1}(1-\mathrm{e}^{\lambda(s-t)})$ so that

$$\frac{\mathrm{d}u}{\mathrm{d}s} = -\xi\,\mathrm{e}^{-\lambda t}\,\mathrm{e}^{\lambda s} \quad\Rightarrow\quad u = \lambda^{-1}\left(\xi + \frac{\mathrm{d}u}{\mathrm{d}s}\right) \quad\Rightarrow\quad \frac{\mathrm{d}s}{\mathrm{d}u} = \left(\frac{\mathrm{d}u}{\mathrm{d}s}\right)^{-1} = \frac{1}{\lambda u-\xi}$$

and (2.18) follows from solving the *partial fraction decomposition*

$$\frac{iua}{(b-iu)(\lambda u-\xi)} = \frac{\alpha+i\beta}{b-iu} + \frac{\gamma+i\delta}{\lambda u-\xi}$$

for some $\alpha, \beta, \gamma, \delta \in \mathbb{R}$. Finally, (2.19) is obtained from $\log(1-\xi^{-1}\lambda u_0) = -\lambda t$.

Chapter 3

Stochastic Volatility Models

A first modification of (1.1) is to model the underlying $S = (S_t)_{t\geq 0}$ with

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

where $X = (X_t)_{t\geq 0}$ is a semimartingale. However, having seen Theorem 1.1, one may anticipate that the next step is to recast the model into a no-arbitrage setting. That is,

$$S_t = S_0 \exp(rt) \exp(X_t) \tag{3.1}$$

where r is the fixed interest rate. One makes brief comments on martingales, based on [14], before considering possible models.

3.1 Equivalent Martingale Measures

Definition 3.1. A probability measure $\mathbb{Q}$ is an *equivalent martingale measure* of $\mathbb{P}$ if $\mathbb{P}$ and $\mathbb{Q}$ are mutually absolutely continuous, and the discounted underlying given by $\mathrm{e}^{-rt} S_t$ is a $\mathbb{Q}$-martingale. The abbreviation is EMM.

Theorem 1.1 uses Delta Hedging to obtain an EMM. This relies upon Itô's formula, and consequently continuity of the semimartingales involved, so the argument cannot be repeated. Luckily, there is the following.

Theorem 3.2 (First Fundamental Theorem of Asset Pricing)**.** *Assume the asset price process S is bounded. Then there exists an EMM if and only if the model satisfies "no free lunch with vanishing risk".*

Proof. [14, Theorem 1.1]. □

No-arbitrage essentially implies that price processes are martingales, so (1.2) prices options. There is a discussion on EMMs in [33] and its references which includes, hedging, market incompleteness, completion of an incomplete market, and Malliavin calculus. For simplicity, one will use a very basic EMM throughout the entirety of the dissertation.

Theorem 3.3 (Mean-Correcting Martingale Measure)**.** Suppose $L = (L_t)_{t\geq 0}$ is some $\mathbb{P}$-Lévy process modelling (3.1) and L_1 has $\mathbb{P}$-characteristic function ψ. If one defines $L' = (L'_t)_{t\geq 0}$ via

$$L'_t = L_t + \bar{\mu}t \tag{3.2}$$

where $\bar{\mu} = -\log\psi(-i)$, then the $\mathbb{P}$-characteristic function ψ' of L'_1 is

$$\psi'(\xi) = \psi(\xi)\exp(i\xi\bar{\mu}) \tag{3.3}$$

and $\exp(L')$ is a $\mathbb{P}$-martingale. Moreover,

$$\mathbb{E}^{\mathbb{Q}}[\exp(L_t)|\mathscr{F}^L_t] := \mathbb{E}^{\mathbb{P}}[\exp(L'_t)|\mathscr{F}^L_t] \tag{3.4}$$

defines an EMM $\mathbb{Q}$ where $\mathscr{F}^L$ is the filtration of L.

Proof. Verification of (3.3) is by definition of L'. The filtrations of $L, L', \exp(L')$ coincide as they are related by measurable mappings. L' is càdlàg so it is bounded on any finite interval and $\exp(L'_t)$ integrable for each $t \geq 0$. The martingale property follows from

$$\begin{aligned}
\mathbb{E}^{\mathbb{P}}[\exp(L'_T)|\mathscr{F}^L_t] &= \exp(L'_t)\mathbb{E}^{\mathbb{P}}[\exp(L'_T - L'_t)|\mathscr{F}^L_t] \\
&= \exp(L'_t)\mathbb{E}^{\mathbb{P}}[\exp(L'_T - L'_t)] \\
&= \exp(L'_t)\mathbb{E}^{\mathbb{P}}[\exp(L'_{T-t})] \\
&= \exp(L'_t)\mathbb{E}^{\mathbb{P}}[\exp(L_{T-t} - (T-t)\log\psi(-i))] \\
&= \exp(L'_t)\mathbb{E}^{\mathbb{P}}[\exp(L_{T-t})](\psi(-i))^{-(T-t)} \\
&= \exp(L'_t)
\end{aligned}$$

appealing to stationary-independent increments of L' and

$$\mathbb{E}^{\mathbb{P}}[\exp(L_t)] = (\psi(-i))^t$$

by Theorem 2.17. To see $\mathbb{Q}$ is an EMM, one remarks that null-sets of $\mathbb{P}, \mathbb{Q}$ coincide by inspection of (3.4). It remains to verify a $\mathbb{Q}$-martingale emerges when discounting (3.1). The underlying filtration is the same for either measure, and integrability follows by inspection of (3.4). The martingale property is inherited from that of L'

$$\begin{aligned}
\mathbb{E}^{\mathbb{Q}}[\mathrm{e}^{-rT}S_T|\mathscr{F}^S_t] &= \mathbb{E}^{\mathbb{Q}}[S_0\exp(L_T)|\mathscr{F}^S_t] \\
&= \mathbb{E}^{\mathbb{Q}}[S_0\exp(L_t)\exp(L_T - L_t)|\mathscr{F}^L_t] \\
&= \mathrm{e}^{-rt}S_t\mathbb{E}^{\mathbb{Q}}[\exp(L_T - L_t)|\mathscr{F}^L_t] \\
&= \mathrm{e}^{-rt}S_t\mathbb{E}^{\mathbb{P}}[\exp(L'_T - L'_t)|\mathscr{F}^L_t] \\
&= \mathrm{e}^{-rt}S_t
\end{aligned}$$

since S, L are related by measurable mappings and their filtrations coincide. $\square$

3.2 Volatility and Time

To construct stochastic volatility (SV) models from (3.1), one must find a way of representing volatility. In (1.1), this is achieved scaling the diffusion W by σ. The scaling

$$\sigma W_t \sim W_{\sigma^2 t} \tag{3.5}$$

of a Brownian motion W defines another Brownian motion! This observation hints towards a description of SV through stochastic time change.

Definition 3.4. Let $L = (L_t)_{t\geq 0}$ be some Lévy process and $Y = (Y_t)_{t\geq 0}$ be some independent, strictly increasing process. The composition $L \circ Y$ defined by

$$L \circ Y_t = L_{Y_t}$$

is the *time change* of L by Y. It is said that L is running in *operational time*, or *stochastic business time*.

Heuristically, during periods of high volatility, many transactions occur and vice versa. That is, when volatility is high, several days may have passed in operational time even though few days have passed in real time.

Lemma 3.5. Let $L = (L_t)_{t\geq 0}$ be some Lévy process and $Y = (Y_t)_{t\geq 0}$ be some independent, strictly increasing process. Suppose L is a martingale. Then $L \circ Y$ is a martingale.

Proof. [13, Lemma 15.2]. □

Theorem 3.6. Suppose L is a Lévy process with

$$\exp(t\psi(\xi)) = \mathbb{E}[\exp(i\xi L_t)]$$

and is independent of some time change Y. If

$$\varphi_t(\xi; y_0) = \mathbb{E}[\exp(i\xi Y_t)|y_0]$$

then

$$\mathbb{E}[\mathrm{e}^{i\xi L_{Y_t}}|y_0] = \varphi_t(-i\psi(\xi); y_0)$$

is the characteristic function of $L \circ Y$ conditioned on y_0.

Proof. One follows the proof of Theorem 79 in [35]. That is,

$$\mathbb{E}[\mathrm{e}^{i\xi L_{Y_t}}|y_0] = \mathbb{E}\big[\mathbb{E}[\mathrm{e}^{i\xi L_{Y_t}}|Y_t]\big|y_0\big] = \mathbb{E}[\mathrm{e}^{Y_t\psi(\xi)}|y_0] = \varphi_t(-i\psi(\xi); y_0)$$

by conditioning on Y_t. □

Both deficiencies in §1 can be addressed. The Lévy process L allows the freedom for an infinitely divisible distribution to be fitted and stochastic volatility ensures market shocks are accounted for.

3.3 Lévy Stochastic Volatility Market Model

Consider processes L and Y in the setting of Theorem 3.6. Then, with Theorem 3.3 and Lemma 3.5 in mind, [33] introduces a model

$$S_t = S_0 \frac{\exp(rt)}{\mathbb{E}[\exp(L_{Y_t})|\theta]} \exp(L_{Y_t}) \tag{3.6}$$

which is essentially (3.1) with

$$X_t = L_{Y_t} - \log \mathbb{E}[\exp(L_{Y_t})|\theta] \tag{3.7}$$

where parameters $\theta = (\theta_1, \dots, \theta_d) \in \Theta \subseteq \mathbb{R}^d$ determine the distribution of $L \circ Y$. The Lévy process L ensures a non-Gaussian character (with jumps) and the IOUP Y introduces SV.

Theorem 3.7. Suppose (3.6) models $S = (S_t)_{t \geq 0}$ where L and Y are determined by θ_ψ and θ_φ respectively. Let

$$\phi_t(\xi; \theta) = \mathbb{E}[\exp(i\xi \log S_t)|S_0, \theta]$$

be the characteristic function of $\log S_t$. Then

$$\phi_t(\xi; \theta) = \exp(i\xi(rt + \log S_0)) \frac{\varphi_t(-i\psi(\xi; \theta_\psi); \theta_\varphi)}{\varphi_t(-i\psi(-i; \theta_\psi); \theta_\varphi)^{i\xi}}$$

and the present value of (3.6) is a martingale.

Proof. This is immediate by Lemma 3.5 and Theorem 3.6. □

Remark. It is noted in [33] that one may impose $y_0 = 1$ for convolution-closed Lévy processes L by scaling other parameters. Examples include

$$\begin{aligned}
\phi_t(\xi; C, G, M, \lambda, a, b, y_0) &= \phi_t(\xi; Cy_0, G, M, \lambda, a, by_0, 1) \\
\phi_t(\xi; C, G, M, Y, \lambda, a, b, y_0) &= \phi_t(\xi; Cy_0, G, M, Y, \lambda, a, by_0, 1) \\
\phi_t(\xi; \alpha, \beta, \delta, \lambda, a, b, y_0) &= \phi_t(\xi; \alpha, \beta, \delta y_0, \lambda, a, by_0, 1) \\
\phi_t(\xi; \alpha, \beta, \delta, \lambda, a, b, y_0) &= \phi_t(\xi; \alpha, \beta, \delta y_0, \lambda, a, by_0, 1)
\end{aligned}$$

where L is a Variance Gamma, CGMY, NIG, Meixner process respectively. A similar formula does not hold in the case of the Generalised Hyperbolic process (see [28]) since it is not generally convolution-closed. That is, i.i.d. GH random variables do not sum to yield another GH random variable.

Chapter 4

Model Calibration

Parameter estimation is necessary before models have applications. In [33], it suggests optimising

$$\theta_\star = \operatorname*{arg\,min}_{\theta\in\Theta} \sqrt{\frac{1}{\#\{\text{data}\}} \sum_{j\in\{\text{data}\}} (C_j(\theta) - C_j^{\mathrm{m}})^2} \tag{4.1}$$

where $C_j(\theta)$ are model call-prices corresponding to a market call-price C_j^{m}. Since each parameter in Table 2.1 and Example 2.40 is valid in some open interval, there is a coordinate-wise surjection

$$\tau\colon \mathbb{R}^d \to \Theta \qquad\qquad \tau(\vartheta) = (\tau_1(\vartheta_1), \ldots, \tau_d(\vartheta_d))$$

where $\tau_1, \ldots, \tau_d$ are given by

$$\begin{aligned}
&\mathbb{R} \to (a, b) &\qquad& x \mapsto a + (b-a)\Big(\frac{1}{2} + \frac{1}{\pi}\arctan x\Big) \\
&\mathbb{R} \to (a, \infty) && x \mapsto a + \mathrm{e}^x \\
&\mathbb{R} \to (-\infty, a) && x \mapsto a - \mathrm{e}^x
\end{aligned}$$

depending on $\Theta \subseteq \mathbb{R}^d$. Recall every surjection maps onto the codomain so all possible $\theta \in \Theta$ are covered. Hence

$$\vartheta_\star = \operatorname*{arg\,min}_{\vartheta\in\mathbb{R}^d} E(\vartheta) \tag{4.2}$$

where

$$E(\vartheta) := \sqrt{\frac{1}{\#\{\text{data}\}} \sum_{j\in\{\text{data}\}} (C_j(\tau(\vartheta)) - C_j^{\mathrm{m}})^2}$$

recasts (4.1) as an unconstrained minimisation problem with $\theta_\star = \tau(\vartheta_\star)$. Clearly, this optimisation assumes one can price calls. It is addressed in [11] using numerical techniques from [24].

4.1 European Call Options

One seeks from [11] an analytic expression for the time-0 call price with strike $K = \mathrm{e}^\zeta$ and maturity T. Assume (3.6) models the underlying, subject to $\theta \in \Theta$. Let $\rho_T(\,\cdot\,; \theta)$

and $\phi_T(\,\cdot\,;\theta)$ denote the respective density and characteristic function of $\log S_T$. Let

$$\phi_T(\xi;\theta) = \mathcal{F}\{\rho_T\}(\xi;\theta) := \int_{-\infty}^{\infty} \mathrm{e}^{ix\xi}\,\rho_T(x;\theta)\,\mathrm{d}x$$

define the Fourier transform, and $C_T(\zeta;\theta)$ be the desired time-0 price. Recall

$$C_T(\zeta;\theta) = \mathbb{E}[\mathrm{e}^{-rT}(S_T - \mathrm{e}^{\zeta})_+|S_0,\theta] = \int_{\zeta}^{\infty} \mathrm{e}^{-rT}(\mathrm{e}^{x} - \mathrm{e}^{\zeta})\rho_T(x;\theta)\,\mathrm{d}x \tag{4.3}$$

by Theorem 3.2. To avoid $\rho_T(\,\cdot\,;\theta)$, [11] applies the identity map to (4.3) and describes $\rho_T(\,\cdot\,;\theta)$ through $\phi_T(\,\cdot\,;\theta)$. It is necessary to define the *modified call option*

$$c(\zeta) := \exp(\alpha\zeta)C_T(\zeta;\theta)$$

for some $\alpha > 0$ so $c(\zeta) = \mathcal{F}^{-1}\{\mathcal{F}\{c\}\}(\zeta)$ due to square-integrability of c.

Theorem 4.1. Suppose α is chosen so c is square integrable. Then

$$C_T(\zeta;\theta) = \frac{\mathrm{e}^{-\alpha\zeta}}{\pi}\int_0^{\infty} \mathrm{e}^{-i\zeta\xi}\,f_T(\xi;\theta)\,\mathrm{d}\xi \tag{4.4}$$

where

$$f_T(\xi;\theta) := \frac{\mathrm{e}^{-rT}\,\phi_T(\xi - (\alpha+1)i;\theta)}{\alpha^2 + \alpha - \xi^2 + (2\alpha+1)\xi i}.$$

Proof. By square-integrability of c,

$$c(\zeta) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{-i\zeta\xi}\,\mathcal{F}\{c\}(\xi)\,\mathrm{d}\xi = \frac{1}{\pi}\int_0^{\infty} \mathrm{e}^{-i\zeta\xi}\,\mathcal{F}\{c\}(\xi)\,\mathrm{d}\xi \tag{4.5}$$

where the second equality holds since c is real - i.e. $\mathcal{F}\{c\}$ is odd in its imaginary part and even in its real part. Fubini's theorem is justified by square-integrability so

$$\begin{aligned}
\mathcal{F}\{c\}(\xi) &= \int_{-\infty}^{\infty} \mathrm{e}^{i\zeta\xi}\,\mathrm{e}^{\alpha\zeta}\,C_T(\zeta;\theta)\,\mathrm{d}\zeta \\
&= \int_{-\infty}^{\infty} \mathrm{e}^{i\zeta\xi}\,\mathrm{e}^{\alpha\zeta}\int_{\zeta}^{\infty} \mathrm{e}^{-rT}(\mathrm{e}^{x} - \mathrm{e}^{\zeta})\rho_T(x;\theta)\,\mathrm{d}x\,\mathrm{d}\zeta \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{i\zeta\xi}\,\mathrm{e}^{\alpha\zeta}\,\mathrm{e}^{-rT}(\mathrm{e}^{x} - \mathrm{e}^{\zeta})\rho_T(x;\theta)\mathbb{1}_{\{x>\zeta\}}\,\mathrm{d}x\,\mathrm{d}\zeta \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-rT}\,\rho_T(x;\theta)(\mathrm{e}^{x+\alpha\zeta} - \mathrm{e}^{(1+\alpha)\zeta})\,\mathrm{e}^{i\zeta\xi}\,\mathbb{1}_{\{x>\zeta\}}\,\mathrm{d}\zeta\,\mathrm{d}x \\
&= \int_{-\infty}^{\infty} \mathrm{e}^{-rT}\,\rho_T(x;\theta)\int_{-\infty}^{x}(\mathrm{e}^{x+\alpha\zeta} - \mathrm{e}^{(1+\alpha)\zeta})\,\mathrm{e}^{i\zeta\xi}\,\mathrm{d}\zeta\,\mathrm{d}x \\
&= \int_{-\infty}^{\infty} \mathrm{e}^{-rT}\,\rho_T(x;\theta)\Big(\frac{\mathrm{e}^{(\alpha+1+i\xi)x}}{\alpha+i\xi} - \frac{\mathrm{e}^{(\alpha+1+i\xi)x}}{\alpha+1+i\xi}\Big)\,\mathrm{d}x \\
&= f_T(\xi;\theta)
\end{aligned} \tag{4.6}$$

and one yields (4.4) by pre-multiplying (4.5) by $\exp(-\alpha\zeta)$ and substituting (4.6). □

Remark. It is argued in [11] that $\phi_T(-(\alpha+1)i;\theta)$ must be finite to ensure square-integrability of c, and [33] claims that this is achieved when $\alpha = 0.75$ for (3.6).

4.2 Numerical Methods

Theorem 4.1, by itself, does not provide call prices. Computation of (4.4) is infeasible so numerical integration is used instead. One might try integration by substitution

$$\int_0^\infty f(\xi)\,\mathrm{d}\xi = \int_0^1 f\Big(\frac{u}{1-u}\Big)\frac{1}{(1-u)^2}\,\mathrm{d}u$$

where approximations fail for (4.4) due to highly oscillatory behaviour of $\mathrm{e}^{i\zeta\xi}$ as $u \uparrow 1$. This means the truncation

$$\int_0^\infty f(\xi)\,\mathrm{d}\xi \approx \int_0^R f(\xi)\,\mathrm{d}\xi$$

shown in [11] is required.

Lemma 4.2. $|\phi_T(\xi - (\alpha+1)i;\theta)| \leq \phi_T(-(\alpha+1)i;\theta)$ for all $\xi \in \mathbb{R}$.

Proof. Let $\xi \in \mathbb{R}$. Then

$$\begin{aligned}
|\phi_T(\xi - (\alpha+1)i;\theta)| &= |\mathbb{E}\big[S_T^{\alpha+1} S_T^{i\xi}\big|\theta\big]| \\
&\leq \mathbb{E}\big[|S_T^{\alpha+1} S_T^{i\xi}|\big|\theta\big] \\
&= \mathbb{E}\big[S_T^{\alpha+1}\big|\theta\big] \\
&= \phi_T(-(\alpha+1)i;\theta)
\end{aligned}$$

by Jensen's inequality. □

Proposition 4.3. Suppose $\phi_T(-(\alpha+1)i;\theta) < \infty$. Then for every $\varepsilon > 0$ there exists $R > 0$ such that

$$\Big|\frac{\mathrm{e}^{-\alpha\zeta}}{\pi}\int_R^\infty \mathrm{e}^{-i\zeta\xi} f_T(\xi;\theta)\,\mathrm{d}\xi\Big| < \varepsilon.$$

Proof. Let $\varepsilon > 0$ and pick

$$R > \frac{\mathrm{e}^{-\alpha\zeta - rT}}{\pi}\frac{\phi_T(-(\alpha+1)i;\theta)}{\varepsilon}.$$

Then

$$\begin{aligned}
\Big|\frac{\mathrm{e}^{-\alpha\zeta}}{\pi}\int_R^\infty \mathrm{e}^{-i\zeta\xi} f_T(\xi;\theta)\,\mathrm{d}\xi\Big| &\leq \frac{\mathrm{e}^{-\alpha\zeta}}{\pi}\int_R^\infty \Big|\frac{\mathrm{e}^{-i\zeta\xi}\,\mathrm{e}^{-rT}\,\phi_T(\xi-(\alpha+1)i;\theta)}{\alpha^2+\alpha-\xi^2+(2\alpha+1)\xi i}\Big|\,\mathrm{d}\xi \\
&= \frac{\mathrm{e}^{-\alpha\zeta}}{\pi}\int_R^\infty \frac{|\,\mathrm{e}^{-rT}\,\phi_T(\xi-(\alpha+1)i;\theta)|}{\sqrt{(\alpha^2+\alpha-\xi^2)^2+(2\alpha+1)^2\xi^2}}\,\mathrm{d}\xi \\
&\leq \frac{\mathrm{e}^{-\alpha\zeta-rT}}{\pi}\int_R^\infty \frac{\phi_T(-(\alpha+1)i;\theta)}{\sqrt{(\alpha^2+\alpha-\xi^2)^2+(2\alpha+1)^2\xi^2}}\,\mathrm{d}\xi \\
&< \frac{\mathrm{e}^{-\alpha\zeta-rT}}{\pi}\int_R^\infty \frac{\phi_T(-(\alpha+1)i;\theta)}{\xi^2}\,\mathrm{d}\xi \\
&= \frac{\mathrm{e}^{-\alpha\zeta-rT}}{\pi}\frac{\phi_T(-(\alpha+1)i;\theta)}{R}
\end{aligned}$$

by triangle and Jensen's inequalities. The latter inequality follows from

$$(\alpha^2+\alpha-\xi^2)^2+(2\alpha+1)^2\xi^2=\xi^4+(2\alpha^2+2\alpha+1)\xi^2+(\alpha^2+\alpha)^2>\xi^4$$

for each $\xi>0$. $\square$

Theorem 4.4 (Composite Simpson's Rule)**.** Suppose

$$0=\xi_1<\cdots<\xi_{N+1}=R$$

is a partition given by

$$\xi_j=(j-1)\eta$$

for each $j=1,\dots,N+1$, where $\eta N=R$ and $N\in\mathbb{N}$ is even. Then

$$\begin{aligned}\int_0^R f(\xi)\,\mathrm{d}\xi &\approx \frac{\eta}{3}\sum_{j=1}^{N/2}(f(\xi_{2j-1})+4f(\xi_{2j})+f(\xi_{2j+1}))\\ &=\frac{\eta}{3}\Big(\sum_{j=1}^{N}(f(\xi_j)(3+(-1)^j-\mathbb{1}_{\{j=1\}}))+f(\eta N)\Big) \end{aligned}\tag{4.7}$$

with an approximation error

$$|e(\eta)|\le\frac{R\eta^4}{180}|f^{(4)}(\xi)|$$

for some $\xi\in(0,R)$.

Proof. [24, Equation 6.14b]. $\square$

Example 4.5. The desired call price (4.4) is

$$\begin{aligned}C_T(\zeta;\theta)\approx\sum_{j=1}^{N}\mathrm{e}^{-i(j-1)\eta\zeta}\,\frac{\eta}{3\pi}(\mathrm{e}^{-\alpha\zeta}\,f_T((j-1)\eta;\theta)(3+(-1)^j-\mathbb{1}_{\{j=1\}}))\\ +\frac{\eta}{3\pi}\,\mathrm{e}^{-\alpha\zeta}\,\mathrm{e}^{iN\eta\zeta}\,f_T(\eta N;\theta)\end{aligned}\tag{4.8}$$

by Theorem 4.4.

Remark. The approximation (4.8) is best when $\zeta\approx0$, but not so reliable for large $|\zeta|$ due to oscillatory behaviour. Calls that are close to being *at-the-money* can be estimated accurately by applying a change of numéraire where $S_0=1$.

Now that calls can be priced, one returns to optimising (4.2). Initially, one tried applying stochastic gradient descent but it did not deliver promising results. One concludes this is because

$$\nabla_\vartheta E(\vartheta)\approx\Big(\frac{E(\vartheta+he_1)-E(\vartheta)}{h},\frac{E(\vartheta+he_2)-E(\vartheta)}{h},\dots,\frac{E(\vartheta+he_d)-E(\vartheta)}{h}\Big)$$

delivered a poor approximation of the gradient. Unfortunately, the analytical gradient is not available either.

4.3 Derivative Free Optimisation

One cites Chapter 9 of [27]. The *Nelder–Mead method* is an optimisation algorithm that does not use the gradient $\nabla_\vartheta E(\vartheta)$ at all. It iteratively builds a picture of the objective function

$$E\colon \mathbb{R}^d \to \mathbb{R}$$

by sampling a minimal number of points at each iteration. Minimality of sampling helps avoid unnecessary expensive computation of $E(\vartheta)$. The algorithm then uses this picture to have the iterated inputs tend towards a local minimum and returns the input that produces this minima. This picture is understood with the following in mind.

Definition 4.6. Suppose $x_1, \ldots, x_{d+1} \in \mathbb{R}^d$ are affinely independent. That is,

$$x_2 - x_1, \ldots, x_{d+1} - x_1$$

are linearly independent. Then

$$\{\lambda_1 x_1 + \cdots + \lambda_{d+1} x_{d+1} \in \mathbb{R}^d \colon \lambda_j \geq 0, \lambda_1 + \cdots + \lambda_{d+1} = 1\}$$

is defined to be a *d-simplex* by [8].

Example 4.7. A 0-simplex is a point. A 1-simplex is a line segment. A 2-simplex is a triangle. A 3-simplex is a tetrahedron, and so on.

The Nelder–Mead algorithm samples E at the vertices of a d-simplex and seeks to iteratively replace the vertex with the worst function value by another point with a better value. It is presented under the following notation. Let $x_1, \ldots, x_{d+1}$ denote the vertices of the current simplex such that

$$E(x_1) \leq \cdots \leq E(x_{d+1})$$

and let

$$\bar{x} = \frac{1}{d} \sum_{j=1}^{d} x_j$$

be the centroid of the best d points. Then

$$\bar{x}(t) := t x_{d+1} + (1-t)\bar{x} \qquad\qquad t \in \mathbb{R}$$

are points lying along the line joining the centroid $\bar{x}$ and the worst point x_{d+1}. Let

$$E_t := E(\bar{x}(t))$$

denote the value the objective function takes at these points. During each iteration, the new point is obtained by reflecting, expanding, or contracting the simplex along the line joining the worst vertex with the centroid of the remaining vertices. If a better point cannot be found in this manner, only the vertex with the best function value is retained, and the simplex shrinks by moving all other vertices toward this value.

Algorithm 1: The Nelder–Mead Method

```
Randomise points x_1, ..., x_{d+1} ∈ R^d;
while termination criterion is False do
    while the points do not form a valid simplex do
        Re-randomise one or more points;
    Evaluate E at each of the d + 1 points;
    Label the points so that E(x_1) ≤ ··· ≤ E(x_{d+1});
    Compute the centroid x̄, reflected point x̄(−1) and evaluate E_{−1};
    if E(x_1) ≤ E_{−1} < E(x_d) then
        "reflected point is neither best nor worst in the new simplex"
        x_{d+1} ← x̄(−1) and go to the next iteration;
    else if E_{−1} < E(x_1) then
        "reflected point is better than the current best; try to go farther along
         this direction"
        Compute the expansion point x̄(−2) and evaluate E_{−2};
        if E_{−2} < E_{−1} then
            x_{d+1} ← x̄(−2) and go to the next iteration;
        else
            x_{d+1} ← x̄(−1) and go to the next iteration;
    else if E_{−1} ≥ E(x_d) then
        "reflected point is still worse than x_d; contract"
        if E(x_d) ≤ E_{−1} < E(x_{d+1}) then
            "try to perform 'outside' contraction"
            Compute x̄(−1/2) and evaluate E_{−1/2};
            if E_{−1/2} ≤ E_{−1} then
                x_{d+1} ← x̄(−1/2) and go to the next iteration;
        else
            "try to perform 'inside' contraction"
            Compute x̄(1/2) and evaluate E_{1/2};
            if E_{1/2} < E_{−1} then
                x_{d+1} ← x̄(1/2) and go to the next iteration;
        if neither outside nor inside contraction was acceptable then
            "shrink the simplex toward x_1"
            for j ← 2 to d + 1 do
                x_j ← (x_1 + x_j)/2;
    Update the termination criterion;
Result: Propose a local minima of the objective function.
```

Remark. Stochastic minibatches were not used to speed up computation in the same manner that stochastic gradient descent compliments classical gradient descent by introducing a stochastic 'error landscape' form a stochastic objective function. This is because one found from testing that noise in the error landscape interfered with the descent process.

4.4 Calibration Results

Calibration of (3.6) follows from applying Nelder-Mead to (4.2) and using (4.8) to price calls. The stochastic processes involved were Lévy processes in Table 2.1 and Gamma-OU from Example 2.40. Particular values include $N = 512$ and $\eta = 0.25$ for Simpson's Rule since major fluctuations in call prices did not appear when $N \gg 512$ and $\eta \ll 0.25$. Python source code and dataset can be found in the Appendix.

Model	Parameters			
VG-Gamma-OU	C	G	M	
	7.1715	14.8992	293.0556	
	λ	a	b	y_0
	0.0015	0.1905	1.6910	1
CGMY-Gamma-OU	C	G	M	Y
	0.0280	3.3274	70.9006	1.5208
	λ	a	b	y_0
	1.3280	0.8696	2.3268	1
NIG-Gamma-OU	α	β	δ	
	38.1310	-21.6592	0.6984	
	λ	a	b	y_0
	1.1900	0.7030	1.4282	1
Mxn-Gamma-OU (Meixner)	α	β	δ	
	0.0729	-1.3805	7.2706	
	λ	a	b	y_0
	1.2207	0.7210	1.3015	1
GH-Gamma-OU	α	β	δ	υ
	0.0198	0.0152	2.2697	-0.6904
	λ	a	b	y_0
	7.7808	0.8904	3.3508	0.0814

Table 4.1: The output of the Nelder Mead calibration, rounded to 4 decimal places.

Data is sourced from [33] so that the interest rate need not be calculated, and is represented by a 75×3 matrix where colomuns represent the strike price, maturity, and market price. One also imposes a change of numéraire so $S_0 = 1$ and measures the maturity such that $T = 1$ corresponds to one year. In Theorem 4.1, α (not to be confused with parameters in Table 4.1) is set to $\alpha = 0.75$ following [33]. Nelder-Mead has been set to terminate if the RMSE fell below 0.1 or if the number of iterations became large.

Model	RMSE	Benchmark
Black-Scholes	-	6.7335
VG-Gamma-OU	1.8575	0.4393
CGMY-Gamma-OU	0.6053	0.3646
NIG-Gamma-OU	0.6369	0.4510
Mxn-Gamma-OU	0.6505	0.4180
GH-Gamma-OU	7.1663	0.3837

Table 4.2: The RMSE of several models after calibration. The parameters can be seen in Table 4.1. The third column gives the RMSE of models from [33] for further comparison.

The RMSE generated by implementation and the benchmark can be compared to conclude that there are local minima which poorly fit the data and the termination criterion could be improved.

Remark. For the GH-Gamma-OU model, one did not find a surjection τ for simplicity: instead, it maps onto an almost everywhere subset of Θ.

4.5 Regularisation

§13.3 in [13] suggests using K-L divergence

$$D_{\mathrm{KL}}(\mathbb{P} \,\|\, \mathbb{Q}) := \mathbb{E}^{\mathbb{P}}\left[\log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\right)\right]$$

to address overfitting. It intends to penalise the measure $\mathbb{P}$ for not being close to a prior $\mathbb{Q}$, and alleviates how sensitive optimisation algorithms are w.r.t. initial starting points. Hence, (4.1) becomes "calibration problem 4" from §13.3 of [13]:

$$\theta_\star = \arg\min_{\theta\in\Theta} \sum_j \omega_j (C_j(\theta) - C_j^{\mathrm{m}})^2 + \beta D_{\mathrm{KL}}(\mathbb{P}_\theta \,\|\, \mathbb{Q}) \tag{4.9}$$

where ω_j are weights and β controls regularisation. Alas, one does not know how to compute the K-L divergence for (3.7) and could not consider (4.9).

Chapter 5

Exotic Options

Exotic options are significantly harder to compute over their vanilla counterparts, so they are typically avoided before model calibration. One cites [29] and [35] to present multiple pricing methods.

5.1 Integral Pricing

Theorem 5.1. Suppose (3.1) models the underlying under the EMM $\mathbb{Q}$. Under $\mathbb{Q}$, let X_T have density $\rho_T(\,\cdot\,;\theta)$ and characteristic function $\phi_T(\,\cdot\,;\theta)$. Define the *modified payoff function*

$$v(x) := V(\mathrm{e}^{-x})$$

for some European option with payoff function $V(S_T)$ at maturity T. Assume $\rho_T(\,\cdot\,;\theta)$ is absolutely continuous w.r.t Lebesgue measure, $\phi_T(-i;\theta) = 1$, $\phi_T(\xi;\theta)$ is defined for all $\xi \in \mathbb{C}\colon \mathrm{Im}\,(\xi) \in [-1, 0]$, and $x \mapsto \mathrm{e}^{-Rx}\,|v(x)|$ is bounded and integrable for some $R \in \mathbb{R}$ with $\phi_T(iR;\theta) < \infty$. Define the negative log forward price

$$\zeta := -\log(\mathrm{e}^{rT}\, S_0)$$

and let $V_0(\zeta;\theta)$ denote the time-0 price of this option. Then

$$V_0(\zeta;\theta) = \frac{\mathrm{e}^{\zeta R - rT}}{2\pi} \int_{\mathbb{R}} \mathrm{e}^{i\xi\zeta}\, \mathcal{L}\{v\}(R + i\xi)\phi_T(iR - \xi;\theta)\,\mathrm{d}\xi \tag{5.1}$$

whenever it exists as Cauchy's principal value. Here $\mathcal{L}\{v\}(z)$ is the *bilateral Laplace transform* (or *two-sided Laplace transform*) of v given by

$$\mathcal{L}\{v\}(z) := \int_{\mathbb{R}} \mathrm{e}^{-zx}\, v(x)\,\mathrm{d}x$$

for all $z \in \mathbb{C}\colon \mathrm{Re}\,(z) = R$.

Proof. In terms of ζ, one has $S_T = \exp(X_T - \zeta)$ so that

$$V_0(\zeta;\theta) = \mathrm{e}^{-rT}\,\mathbb{E}^{\mathbb{Q}}[V(\mathrm{e}^{-\zeta + X_T})|\theta] = \mathrm{e}^{-rT}\,\mathbb{E}^{\mathbb{Q}}[v(\zeta - X_T)|\theta]$$

by Theorem 3.2. By definition, the expectation is

$$\mathbb{E}^{\mathbb{Q}}[v(\zeta - X_T)|\theta] = \int_{\mathbb{R}} v(\zeta - x)\rho_T(x;\theta)\,\mathrm{d}x$$

which can be viewed as the convolution! By assumption, $x \mapsto \mathrm{e}^{-Rx}\,|v(x)|$ is bounded, integrable, and $x \mapsto \mathrm{e}^{-Rx}\,|\rho_T(x;\theta)|$ is integrable (since $\phi_T(iR;\theta) < \infty$), so convolution theorem yields

$$\mathcal{L}\{V_0\}(R+i\xi;\theta) = \mathrm{e}^{-rT}\,\mathcal{L}\{v\}(R+i\xi)\mathcal{L}\{\rho_T\}(R+i\xi;\theta) \tag{5.2}$$

for every $\xi \in \mathbb{R}$. This shows the bilateral Laplace transform $\mathcal{L}\{V_0\}(\,\cdot\,;\theta)$ converges absolutely and $V_0(\,\cdot\,;\theta)$ is continuous. Hence, by Laplace inversion,

$$\begin{aligned} V_0(\zeta;\theta) &= \frac{1}{2\pi i}\int_{R-i\infty}^{R+i\infty} \mathrm{e}^{\zeta z}\,\mathcal{L}\{V_0\}(z;\theta)\,\mathrm{d}z \\ &= \frac{\mathrm{e}^{\zeta R}}{2\pi}\int_{\mathbb{R}} \mathrm{e}^{i\zeta\xi}\,\mathcal{L}\{V_0\}(R+i\xi;\theta)\,\mathrm{d}\xi \end{aligned} \tag{5.3}$$

where (5.3) is Cauchy's principal value integral. Substituting (5.2) into (5.3) yields

$$V_0(\zeta;\theta) = \frac{\mathrm{e}^{\zeta R - rT}}{2\pi}\int_{\mathbb{R}} \mathrm{e}^{i\xi\zeta}\,\mathcal{L}\{v\}(R+i\xi)\mathcal{L}\{\rho_T\}(R+i\xi;\theta)\,\mathrm{d}\xi$$

and the conclusion follows from $\phi_T(\xi;\theta) = \mathcal{L}\{\rho_T\}(-i\xi;\theta)$. □

This is how [29] generalises §4.1 to price power, and quanto options. The option price (5.1) is computed by numerical integration. Theorem 5.1, while elegant, has its caveats! Namely, the calculation of $\mathcal{L}\{v\}$ might not be feasible and (5.1) does not tackle path dependent options. That is why Monte Carlo is preferred in practice, which gives great motivation to study its prerequisite: simulation.

5.2 Simulation

The aim is to simulate any Lévy process (given its LKC) and OUPs from (2.7). Drift is trivial, so simulation from the diffusion component and Lévy measure are the only concerns. Theorem 2.17 highlights that the components

$$X_t = \mu t + \sigma W_t + C_t + \lim_{\varepsilon\downarrow 0} M_t^{\varepsilon} \tag{5.4}$$

can be simulated independently. Suppose one has access to any number

$$U_1,\ldots,U_n \sim \mathrm{U}(0,1)$$

of i.i.d. continuous uniform random variables. One starts by summarising random sampling from [21], [31] and [35]. Assume χ is $\mathcal{X}$-valued, where $\mathcal{X} = [0,\infty)$ or $\mathcal{X} = \mathbb{R}$, and χ has a $\mathbb{P}$-c.d.f.

$$F(x) = \int_{-\infty}^{x} f(y)\,\mathrm{d}y$$

where f is the $\mathbb{P}$-density on $\mathcal{X}$ and vanishes on $\mathbb{R}\setminus\mathcal{X}$.

Remark. If F is invertible and $U \sim \mathrm{U}(0,1)$ then $F^{-1}(U) \sim \chi$. Otherwise the *quantile function*, $Q(u) := \inf\{x \in \mathbb{R} \colon F(x) \geq u\}$ for every $u \in (0,1)$, must be used in place of F^{-1}.

Algorithm 2: Inverse Transform Sampling

Simulate $U \sim \mathrm{U}(0,1)$;
$\chi \leftarrow F^{-1}(U)$ if possible, otherwise $\chi \leftarrow Q(U)$;
Result: A random sample χ from the desired distribution.

Example 5.2. The exponential distribution $\mathrm{Exp}(\lambda)$ has a c.d.f. $F(x) = 1-\exp(-\lambda x)$ and an inverse $F^{-1}(x) = -\log(1-x)/\lambda$. Recognising $U \sim 1-U$, one has that

$$-\log(U)/\lambda \sim \mathrm{Exp}(\lambda)$$

where $U \sim \mathrm{U}(0,1)$. Consequently one can simulate a sample path of the Poisson process by setting

$$N_t = \#\{n \in \mathbb{N} \colon T_n \leq t\}, \qquad T_n = Z_1 + \cdots + Z_n$$

where $Z_j \sim \mathrm{Exp}(\lambda)$, since the inter-arrival times of the jumps follow an exponential distribution.

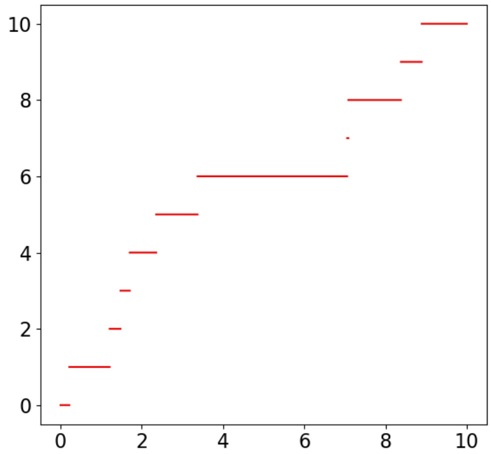


Figure 5.1: A sample path of a Poisson process with rate $\lambda = 1$.

Remark. Vertical lines will be removed whenever the processes has finitely many jumps on any given finite interval.

Theorem 5.3. Let $U \sim \mathrm{U}(0,1)$. Then $Q(U) \sim \chi$.

Proof. The proof in [31] establishes

$$\mathbb{P}[Q(U) \leq x] = \mathbb{P}[U \leq F(x)] = F(x) = \mathbb{P}[\chi \leq x]$$

by claiming: $Q(u) \leq x$ if and only if $u \leq F(x)$. Let $Q(u) \leq x$. Then $F(Q(u)) \leq F(x)$, since F is non-decreasing, and it is sufficient to verify $u \leq F(Q(u))$. This is indeed the case because $\{x \in \mathbb{R} \colon F(x) \geq u\}$ attains its minimum by right-continuity of F which consequently means $Q(u)$ is an element of $\{x \in \mathbb{R} \colon F(x) \geq u\}$. Conversely, let $u \leq F(x)$. Then $x \in \{y \in \mathbb{R} \colon F(y) \geq u\}$ and so $Q(u) \leq x$ by minimality. $\square$

The downside of Inverse Transform Sampling is the limitation imposed by one's ability to compute the quantile function Q or inverse function F^{-1}. Fortunately there is an alternative method if there exists $c > 0$ such that for all $x \in \mathcal{X}$

$$f(x) \leq cg_\star(x)$$

where $g_\star$ is the density of an $\mathcal{X}$-valued random variable which is easy to simulate.

Algorithm 3: Acceptance-Rejection Sampling

Compute c;
while $U > f(\chi)/cg_\star(\chi)$ *(assumed to be vacuously true)* **do**
  Simulate χ from density $g_\star$;
  Simulate $U \sim \mathrm{U}(0,1)$;
Result: A random sample χ given $U \leq f(\chi)/cg_\star(\chi)$.

Theorem 5.4. Samples coming out of Algorithm 3 have the desired distribution with density f.

Proof. The distribution of the sample is given by

$$\mathbb{P}[\chi \in A | U \leq f(\chi)/cg_\star(\chi)] = \frac{\mathbb{P}[\chi \in A, U \leq f(\chi)/cg_\star(\chi)]}{\mathbb{P}[U \leq f(\chi)/cg_\star(\chi)]}$$

for any measurable $A \subseteq \mathcal{X}$. By conditioning on χ, one has that

$$\begin{aligned}
\mathbb{P}[U \leq f(\chi)/cg_\star(\chi)] &= \int_{\mathcal{X}} \mathbb{P}[U \leq f(\chi)/cg_\star(\chi) | \chi = y] g_\star(y)\,\mathrm{d}y \\
&= \int_{\mathcal{X}} \mathbb{P}[U \leq f(y)/cg_\star(y)] g_\star(y)\,\mathrm{d}y \\
&= \int_{\mathcal{X}} \frac{f(y)}{cg_\star(y)} g_\star(y)\,\mathrm{d}y \\
&= \frac{1}{c} \qquad (5.5)
\end{aligned}$$

where $0/0 = 1$ on the event $\{g_\star(y) = 0\}$. Hence,

$$\begin{aligned}
\mathbb{P}[\chi \in A | U \leq f(\chi)/cg_\star(\chi)] &= \frac{\mathbb{P}[\chi \in A, U \leq f(\chi)/cg_\star(\chi)]}{1/c} \\
&= c \int_{\mathcal{X}} \mathbb{1}_{\{\chi \in A\}} \frac{f(y)}{cg_\star(y)} g_\star(y)\,\mathrm{d}y \\
&= \int_A f(y)\,\mathrm{d}y
\end{aligned}$$

where the second equality follows by conditioning on χ again. □

Remark. The probability of a successful sample attempt is given by (5.5). Since each sample attempt is independent, the number of attempts is geometrically distributed with mean c. This is Lemma 57 in [35]. Therefore it is desirable to find a value $c \geq 1$ as small as possible to generate fewer wasted samples.

Example 5.5. Gaussian random variables $\mathcal{N}(0,1)$ can be sampled from the Box-Muller generator or Polar method. The source code in the Appendix assumes there are any number of i.i.d. standard Gaussian random variables similar to [33].

Once individual random variables have been sampled, one can use one of the many methods provided in [33] and [35] to simulate Lévy processes.

Definition 5.6. Suppose $X = (X_t)$ is a Lévy process so that X_t has a density f_t, c.d.f. F_t, and quantile function Q_t. Fix a *time-lag* $\delta > 0$. Then the process

$$X_t^{(1,\delta)} = \sum_{j=1}^{\lfloor t/\delta \rfloor} Z_j \tag{5.6}$$

where $Z_j = Q_\delta(U_j)$, is the *time discretisation* of X with time lag δ.

Proposition 5.7. One has $X_t^{(1,\delta)} \to X_t$ in distribution as $\delta \downarrow 0$.

Proof. Proposition 48 in [35] employs a coupling proof by appealing to stationary independent increments of X and a.s. continuity at fixed t. Stationary independent increments means that (5.6) has the same distribution as $X_{\lfloor t/\delta \rfloor \delta}$. Now suppose t is given. Then, $s \mapsto X_s$ is continuous at t with probability one so that $X_{\lfloor t/\delta \rfloor \delta} \to X_t$ as $\delta \downarrow 0$ a.s.. The claim is deduced by putting together the two previous facts. □

Example 5.8. Standard Brownian motion can be simulated through (5.6) where $Z_j \sim \mathcal{N}(0,\delta)$ variables. In fact, this is proven as a specific case of Donsker's Theorem: Theorem 11 in [35]. Since floating point errors become a serious issue when δ is small, accuracy is improved by observing

$$\sqrt{\delta} Z_j \sim \mathcal{N}(0,\delta)$$

whenever $Z_j \sim \mathcal{N}(0,1)$. Simulation of standard Brownian motion sample paths can be seen in Figure 2.1. Consequently, the diffusion component σW_t in (5.4) can be simulated by pre-multiplying standard Brownian motion by σ.

Example 5.9. Berman's Gamma Generator (see [33]) samples $\chi \sim \Gamma(a,1)$ random variables from $\mathrm{U}(0,1)$ samples. Moreover, $\chi/b \sim \Gamma(a,b)$ so the Gamma process can be simulated from time discretisation.

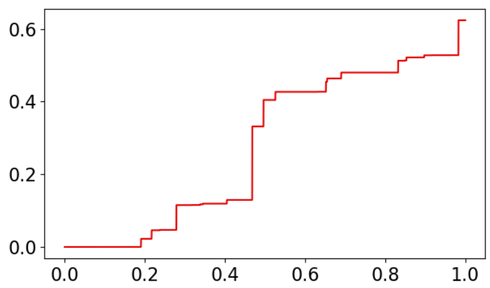


Figure 5.2: A sample path of a $\Gamma(5,10)$ process. It is a subordinator and has finite variation, however, it has infinitely many jumps on any finite interval. The piecewise constant appearance is due to floating point errors: small jumps are ignored.

Example 5.10. Similarly, the Inverse Gaussian (IG) distribution can be sampled from uniform and standard Gaussian samples, and time discretisation applies.

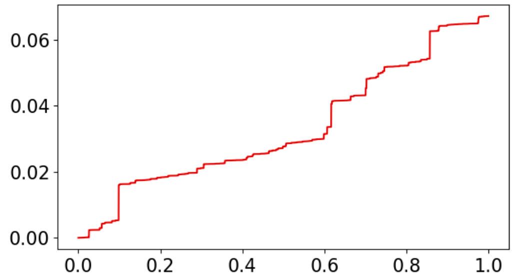


Figure 5.3: A sample path of a IG$(1, 20)$ process.

Example 5.11. Variance Gamma (VG) processes can be represented as the difference of two Gamma processes and has finite variation. The VG(C, G, M) process X can be decomposed into $X_t = G_t^+ - G_t^-$ where G^+ is a $\Gamma(C, M)$ process and G^- is a $\Gamma(C, G)$ process.

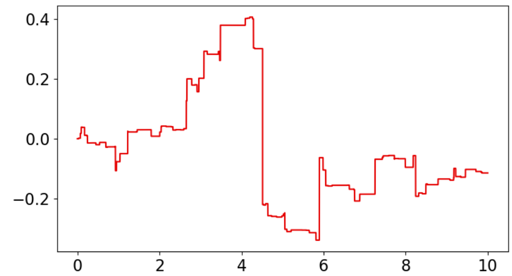


Figure 5.4: A sample path of a VG$(1, 5, 6)$ process.

Example 5.12. A compound Poisson process X with Lévy density g satisfies

$$X_t \sim Q(U_1) + \cdots + Q(U_{N_t}) \qquad U_j \sim \mathrm{U}(0, 1) \text{ i.i.d.} \tag{5.7}$$

where $g(x) = \lambda h(x)$ for a probability density h with quantile function Q. The Poisson process $N = (N_t)_{t \geq 0}$ is simulated from Example 5.2 at rate λ. Compare (5.7) to (2.5).

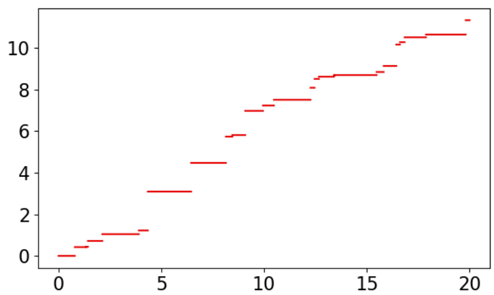


Figure 5.5: A compound Poisson process with rate 0.8696 and $Q(U) \sim \mathrm{Exp}(2.3268)$.

Example 5.12 can be extended to approximate any given Lévy processes with a Lévy density. The idea is to fix $\varepsilon > 0$ in (5.4) and simulate each component independently before summing them up. Credit goes to [35].

Definition 5.13. Let $\varepsilon > 0$ be given. Suppose $\nu(\mathrm{d}x) = g(x)\,\mathrm{d}x$ and

$$(\mu, \sigma, \nu)$$

are the LKC of a Lévy process X. Recall LKC are unique by Corollary 2.18 and Theorem 2.17 ensures g satisfies integrability on $\mathbb{R} \setminus [-\varepsilon, \varepsilon]$. Simulation of the drift is trivial, and the diffusion component is covered by Example 5.8. The remaining terms

$$C_t + M_t^\varepsilon = \sum_{0 \leq s \leq t} \Delta_s \mathbb{1}_{\{|\Delta_s| > \varepsilon\}} - t \int_{\{x \in \mathbb{R}\colon\, \varepsilon < |x| \leq 1\}} x\nu(\mathrm{d}x) \tag{5.8}$$

satisfy one of two possibilities depending on the mass $\nu(\{x \in \mathbb{R}\colon |x| > \varepsilon\})$. If the mass vanishes, then (5.8) is the zero function, otherwise the mass takes some value in $(0, \infty)$ and one can recognise two terms in (5.8). The first term is a Poisson point process which may be identified with a compound Poisson process in the same way as (2.1). That means, one can consider a Poisson process with a rate

$$\lambda_\varepsilon := \nu(\{x \in \mathbb{R}\colon |x| > \varepsilon\}) \equiv \int_{\{x \in \mathbb{R}\colon\, |x| > \varepsilon\}} g(x)\,\mathrm{d}x$$

and a probability density function

$$h_\varepsilon(x) := \frac{g(x)}{\lambda_\varepsilon} \mathbb{1}_{\{|x| > \varepsilon\}}$$

so that Example 5.12 may be applied. The second term in (5.8) is (compensating) drift, which is deterministic and trivial. Let N be a Poisson process with rate λ_ε and Z_j be i.i.d. with a distribution defined by h_ε. Then the process

$$X_t^{(2,\varepsilon)} = \Big(\mu - \int_{\{x \in \mathbb{R}\colon\, \varepsilon < |x| \leq 1\}} xg(x)\,\mathrm{d}x\Big)t + \sigma W_t + \sum_{j=1}^{N_t} Z_j$$

is the *Lévy process with small jumps thrown away*, and is simulated in three parts: drift, Brownian part, and compound Poisson part. The simulation technique is the *compound Poisson approximation with small jumps thrown away*.

Proposition 5.14. One has $X_t^{(2,\varepsilon)} \to X_t$ in distribution as $\varepsilon \downarrow 0$.

Proof. This uses a coupling idea: one has

$$X_t^{(2,\varepsilon)} \sim \mu t + \sigma W_t + C_t + M_t^\varepsilon$$

and the RHS converges to X_t in L^2 sense by Theorem 2.17. The rest of the details are omitted but the idea is that L^2 convergence implies L^1 convergence by Jensen's inequality, L^1 convergence implies convergence in probability by Markov's inequality, and convergence in distribution follows. □

Example 5.15. Tables 2.1 – 2.4 showcase Lévy processes and their LKC. Suppose one wishes to simulate the Meixner process. Before one can compute the Meixner process with small jumps thrown away, several integrals need to be computed. They are the drift μ, compensating drift in (5.8), and rate of the Poisson process λ_ε. Integrals over an infinite domain are truncated in the fashion covered in §4.2 and a translation

$$\int_a^b f(x)\,\mathrm{d}x = \int_0^{b-a} f(x+a)\,\mathrm{d}x$$

is performed to apply Simpson's Rule. Computations are done in a computer-friendly manner in order to avoid "math overflow" errors. For instance

$$\frac{\exp(\beta x)}{\sinh(\pi x)} \equiv \frac{2\exp((\beta-\pi)x)}{1-\exp(-2\pi x)}$$

are equivalent, but the LHS is prone to the handling of numbers that are too large for a computer. Once the integrals have been computed, one can simulate a Poisson process with rate λ_ε from Example 5.2 and use this in Example 5.12. As it turns out, simulation from the probability density

$$h_\varepsilon(x) = \frac{\delta\exp(\beta x/\alpha)}{\lambda_\varepsilon x\sinh(\pi x/\alpha)}\mathbb{1}_{\{|x|>\varepsilon\}}$$

is highly non-trivial. Good luck finding the quantile function! Rejection sampling is used instead, which reduces the task to finding $c > 0$ in Algorithm 3. There is the upper bound

$$\begin{aligned}
h_\varepsilon(x) &= \frac{2\delta\exp((\beta+\pi)x/\alpha)}{\lambda_\varepsilon x(\exp(2\pi x/\alpha)-1)}\mathbb{1}_{\{x<-\varepsilon\}} + \frac{2\delta\exp((\beta-\pi)x/\alpha)}{\lambda_\varepsilon x(1-\exp(-2\pi x/\alpha))}\mathbb{1}_{\{x>\varepsilon\}} \\
&< \frac{2\delta\exp((\beta+\pi)x/\alpha)}{\lambda_\varepsilon(-\varepsilon)(\exp(2\pi(-\varepsilon)/\alpha)-1)}\mathbb{1}_{\{x<-\varepsilon\}} + \frac{2\delta\exp((\beta-\pi)x/\alpha)}{\lambda_\varepsilon\varepsilon(1-\exp(-2\pi\varepsilon/\alpha))}\mathbb{1}_{\{x>\varepsilon\}} \\
&= \frac{2\delta}{\lambda_\varepsilon\varepsilon(1-\exp(-2\pi\varepsilon/\alpha))}\min\left\{\exp\left(\frac{\beta+\pi}{\alpha}x\right), \exp\left(\frac{\beta-\pi}{\alpha}x\right)\right\}
\end{aligned}$$

which prompts one to consider the following. The (centred) Laplace distribution has a probability density

$$g_\star(x) = \frac{1}{2p}\exp\Big(-\frac{|x|}{p}\Big), \qquad p > 0$$

and is supported on $\mathbb{R}$. It has quantile function

$$Q(u) = -p\,\mathrm{sgn}(u-0.5)\log(1-2|u-0.5|)$$

so it can be trivially sampled from Algorithm 2. The constant

$$c = \frac{4\alpha\delta}{\lambda_\varepsilon\varepsilon(1-\exp(-2\pi\varepsilon/\alpha))}\min\left\{\frac{1}{\beta-\pi}, \frac{1}{\beta+\pi}\right\}$$

comes from comparing $g_\star$ to the upper bound of h_ε. It remains to program these calculations into a computer and let Python carry out the computation. Note that the diffusion component of the Meixner process is zero, but Proposition 2.20 ensures its sample paths are of unbounded variation. So it is expected to look similar to a jump process added to a Brownian motion.

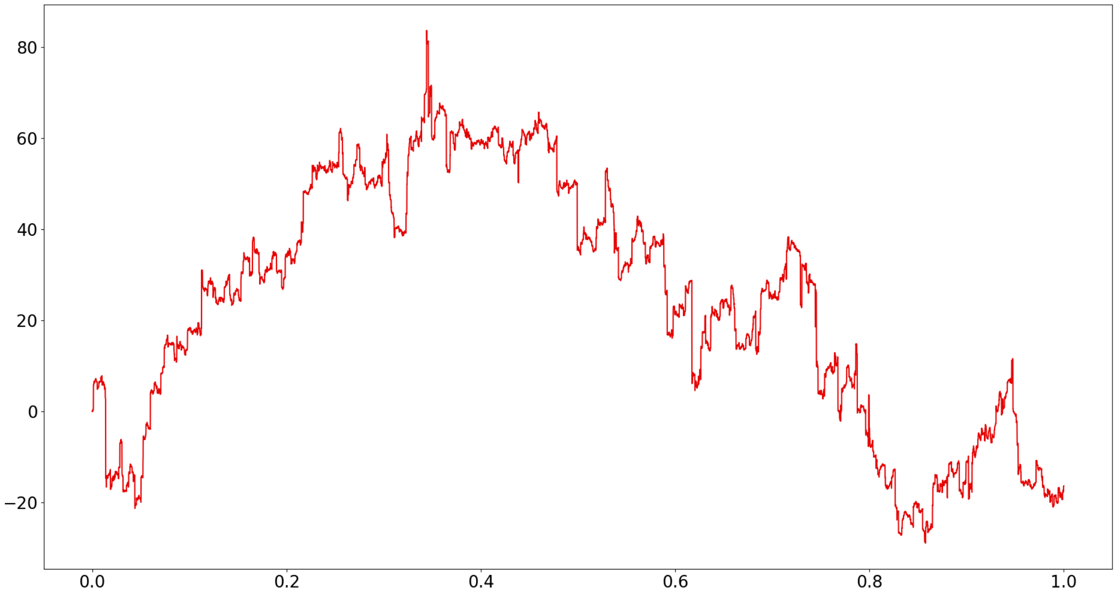
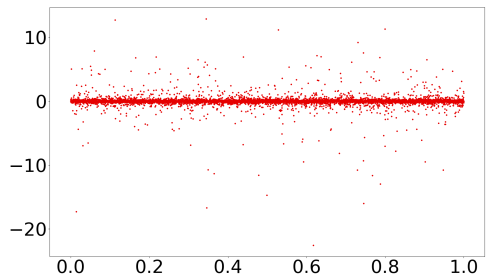
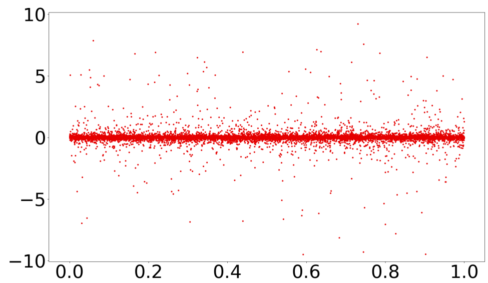


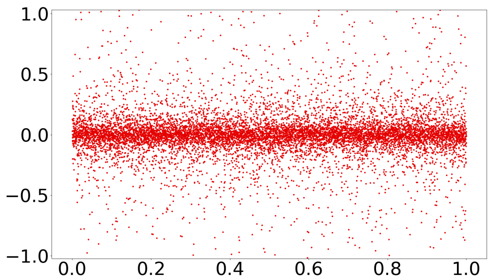
Figure 5.6: A sample path of the Meixner process resulting from the discussion in Example 5.15 where $\alpha = 25$, $\beta = 0$, $\delta = 1$, and $\varepsilon = 0.001$. An ε this small means $c \gg 1$ and the total running time of the simulation was over 20 hours.



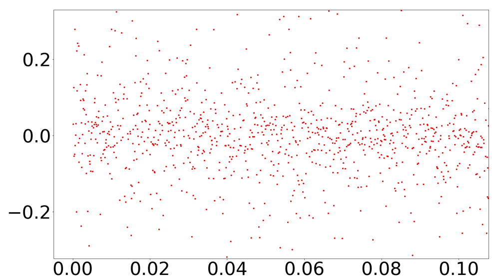
(a) All the jumps made by Figure 5.6.



(b) Symmetric Plot.



(c) Enlarging the y-axis.



(d) Enlarging both axes.

Figure 5.7: The Poisson point process corresponding to Figure 5.6 at different scales. The Poisson point process exhibits behaviour that is more similar to the S&P500 daily log-returns, rather than white noise in Figure 1.4! That is, there is the occurrence of rare events such as large market movement.

Example 5.16. It is claimed in [33] that the Normal Inverse Gaussian (NIG) process can be simulated from a time-changed Brownian motion. Let α, β, δ be valid NIG parameters, W be a standard Brownian motion, and I be an Inverse Gaussian (IG) process with parameters $a = 1$, $b = \delta\sqrt{\alpha^2 - \beta^2}$. Then

$$X_t = \beta\delta^2 I_t + \delta W_{I_t}$$

defines an NIG process, X. Indeed,

$$\begin{aligned}
\mathbb{E}[\exp(i\xi X_t)] &= \mathbb{E}[\mathbb{E}[\exp(i\xi\beta\delta^2 I_t)\exp(i\xi\delta W_{I_t})|I_t]] && \text{tower law}\\
&= \mathbb{E}[\exp(i\xi\beta\delta^2 I_t)\mathbb{E}[\exp(i\xi\delta W_{I_t})|I_t]] && \text{smoothing}\\
&= \mathbb{E}[\exp(i\xi\beta\delta^2 I_t)\exp(-I_t\delta^2\xi^2/2)] && \mathbb{E}[\exp(\sigma W_t)] = -t\sigma^2\xi^2/2\\
&= \mathbb{E}[\exp(i\eta I_t)] && \eta := -i(i\xi\beta\delta^2 - \delta^2\xi^2/2)\\
&= \exp(atb - at\sqrt{b^2 - 2i\eta})
\end{aligned}$$

and the result follows from substituting a and b. Hence, one may recursively define

$$X_0 = 0 \qquad X_{j\Delta t} = X_{(j-1)\Delta t} + \beta\delta^2(I_{j\Delta t} - I_{(j-1)\Delta t}) + \delta\sqrt{I_{j\Delta t} - I_{(j-1)\Delta t}}Z_j$$

where $Z_j \sim \mathcal{N}(0, 1)$ are i.i.d. and Δt is a time-step.

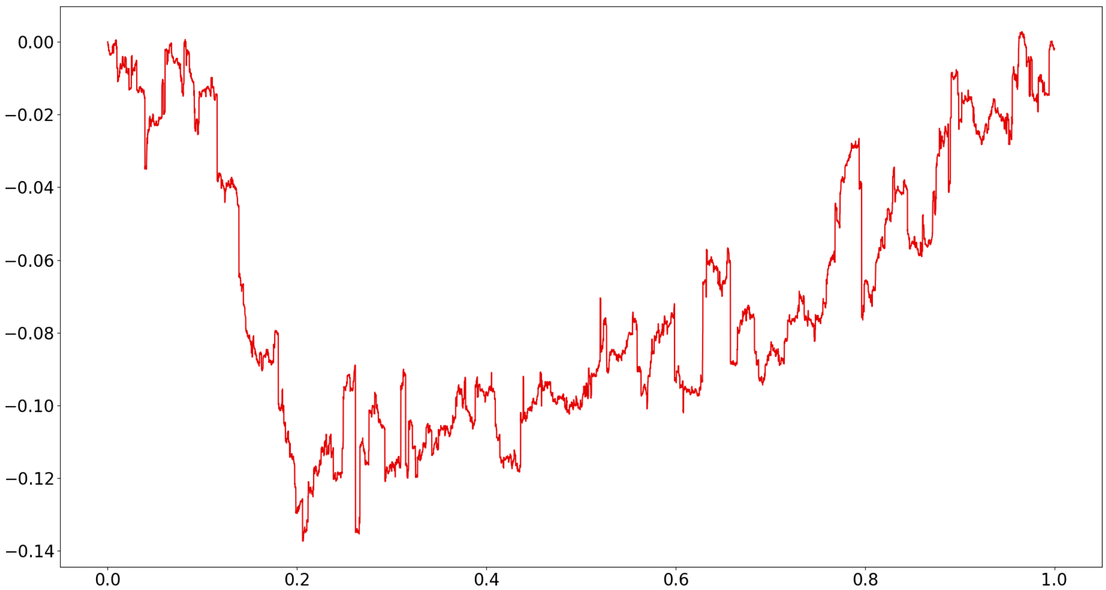


Figure 5.8: A sample path of a NIG$(50, -10, 1)$ process. This particular path seems similar to some sort of stock market crash!

Example 5.17. When $\alpha = 1$ or $\alpha = 2$, the α-stable distributions are the Cauchy and Gaussian distributions receptively. Moreover, the same subordination trick can be applied: let W_t be standard Brownian motion and Y_t be a $1/2$-stable subordinator. Then Proposition 3.11 in Chapter III of [30] states W_{Y_t} is a Cauchy process.

Time discretisation and compound Poisson approximation rely on sums of many, mostly small, i.i.d. random variables. If each is affected by a small error then these errors accumulate, and approximations fail. §9.2 in [35] addresses this. Brownian motion can be simulated in stages by following Lévy construction, and compound Poisson approximations can be simulated in stages too. Let Δ be a Poisson point process with intensity measure $\nu(\mathrm{d}x) = g(x)\,\mathrm{d}x$ and $\infty = \varepsilon_0 > \varepsilon_1 > \varepsilon_2 > \cdots > 0$ be a strictly decreasing sequence with $\varepsilon_n \downarrow 0$ as $n \to \infty$. Define for each $k \geq 1$, $t \geq 0$,

$$M_t^{(k)} := \sum_{0 \leq s \leq t} \Delta_s^{(k)} - t \int_{\varepsilon_k}^{\varepsilon_{k-1}} x g(x) \mathbb{1}_{\{0<x\leq 1\}}\,\mathrm{d}x \tag{$\star$}$$

and

$$M_t^{(-k)} := \sum_{0 \leq s \leq t} \Delta_s^{(-k)} - t \int_{-\varepsilon_{k-1}}^{-\varepsilon_k} x g(x) \mathbb{1}_{\{0>x\geq -1\}}\,\mathrm{d}x \tag{$\star$}$$

where $\Delta_t^{(k)} := \Delta_t \mathbb{1}_{\{\varepsilon_k < \Delta_t \leq \varepsilon_{k-1}\}}$ and $\Delta_t^{(-k)} := \Delta_t \mathbb{1}_{\{-\varepsilon_k > \Delta_t \geq -\varepsilon_{k-1}\}}$. These processes are independent of each other, and can be simulated from Example 61 in [35]. Consider a bounded interval I_k with length $|I_k|$ such that 0 does not lie on its boundary. Then one can use Algorithm 3 with

$$g_\star(x) = \frac{1}{|I_k|} \qquad\qquad c = \frac{|I_k|}{\lambda^{(k)}} \sup_{x \in I_k} g(x) \tag{5.9}$$

to sample i.i.d. random variables for the construction of a compound Poisson process, where

$$\lambda^{(k)} := \int_{I_k} g(x)\,\mathrm{d}x$$

is the rate of the underlying Poisson process. The compound Poisson approximation is obtained through the *superposition* of ($\star$) for as many k as one desires. Under suitable conditions, Figure 5.6 and Figure 5.8 suggests approximating small jumps by a Brownian motion. Indeed, this is addressed in [1], [33], and [35]. Define

$$\sigma^2(\varepsilon) := \int_{\{0<|x|<\varepsilon\}} x^2 g(x)\,\mathrm{d}x$$

for some Lévy process X with Lévy density g.

Theorem 5.18. If

$$\lim_{\varepsilon \downarrow 0} \frac{\sigma(\varepsilon)}{\varepsilon} = \infty \tag{5.10}$$

then

$$\frac{X_t - X_t^{(2,\varepsilon)}}{\sigma(\varepsilon)} \to B_t \text{ in distribution as } \varepsilon \downarrow 0$$

for an independent Brownian motion B.

In particular,

$$X_t^{(2+,\varepsilon)} = X_t^{(2,\varepsilon)} + \sigma(\varepsilon) B_t$$

is an approximation that may be used in the context of simulation. Theorem 5.18 is proven in [1] which rigorously justifies a diffusion appearing, even if one is not present in the original Lévy process.

Example 5.19. Note that (5.10) does not correspond with the Lévy process having infinitely many jumps on finite intervals: see Example 5.9. Nor does it correspond with a Lévy process being of unbounded variation. For a counterexample of the latter erroneous assumption, consider the CGMY process. Theorem 2 in [9] asserts it is of finite variation when $Y < 1$ and Example 66 in [35] shows that approximating small jumps by a Brownian motion is valid when $Y > 0$.

Example 5.20. One tries to replicate Example 43 in [35]. It has Lévy measure

$$g(x) = |x|^{-5/2}\mathbb{1}_{\{x\in[-3,0)\}}$$

with compensating drift that makes it a martingale. It has no diffusion component but is a special case of when Theorem 5.18 and unbounded variation coincide. Note

$$\int_{0<|x|\leq 1} |x|g(x)\,\mathrm{d}x = \int_{-1}^{0} |x|^{-3/2}\,\mathrm{d}x = \infty$$

so it is indeed of unbounded variation by Proposition 2.20 and

$$\lim_{\varepsilon\downarrow 0}\frac{1}{\varepsilon}\Big(\int_{\{0<|x|<\varepsilon\}} x^2 g(x)\,\mathrm{d}x\Big)^{1/2} = \lim_{\varepsilon\downarrow 0}\frac{\varepsilon^{1/4}}{\varepsilon} = \infty$$

so one may choose to use Theorem 5.18. It is not used for the purposes of replication. Note

$$\text{“compensating drift on } (-b,-a)\text{”} = \int_{-b}^{-a} xg(x)\,\mathrm{d}x = 2(a^{-1/2} - b^{-1/2})$$

$$\text{“rate of Poisson process on } (-b,-a)\text{”} = \int_{-b}^{-a} g(x)\,\mathrm{d}x = \frac{2}{3}(a^{-3/2} - b^{-3/2})$$

so compound Poisson approximation can be applied through superposition using (5.9) since $3 > 1 > 0.3 > 0.1 > 0.01$ form bounded intervals without 0 on the boundary.

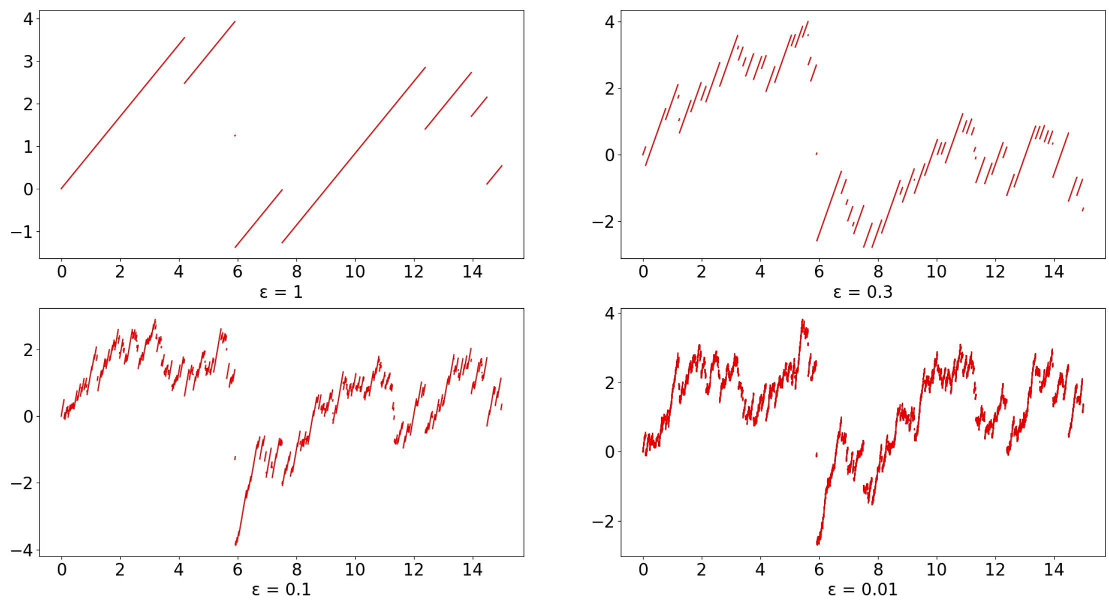


Figure 5.9: A reproduction of the discontinuous martingale (Figure 6.1) in [35].

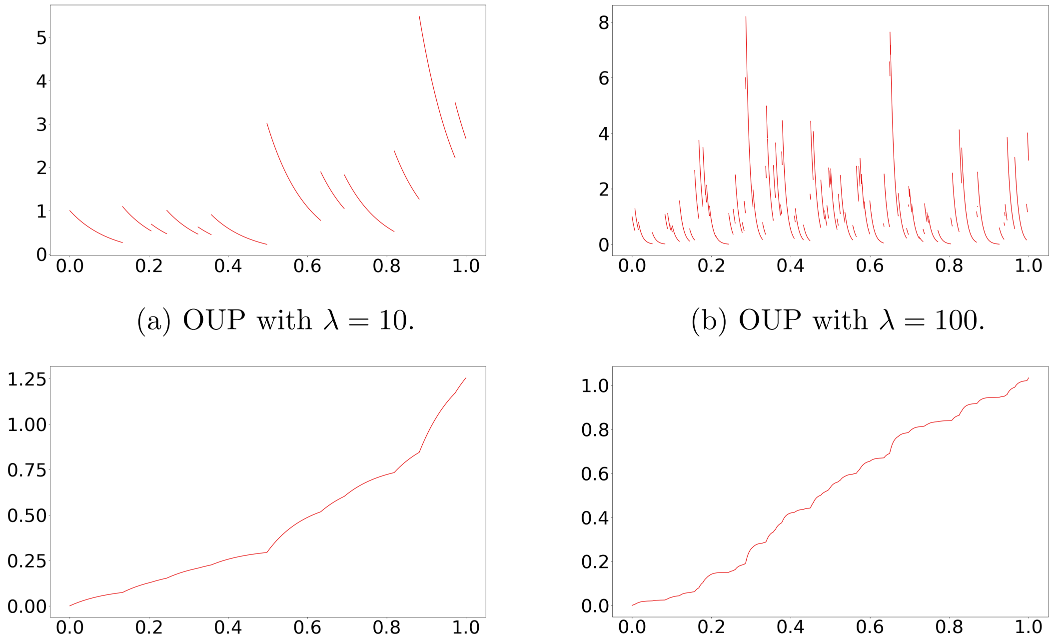
Remark. Further examples include §6 of [26] where Meixner processes are simulated via subordination. Also [6] and [22] respectively sample GH and Meixner random variables, so one may apply time discretisation. Further methods include §6.5 in [13].
It is time to consider OUPs. Recall from Example 2.40 that the BDLP $z = (z_t)_{t\geq 0}$ is a compound Poisson process with rate a and $x_n \sim \mathrm{Exp}(b)$ i.i.d. random variables. In §8.4.6 of [33], the OUP $y = (y_t)_{t\geq 0}$ is simulated through the recursive relation

$$y_{n\Delta t} = (1 - \lambda\Delta t)y_{(n-1)\Delta t} + \sum_{n=N_{(n-1)\Delta t}+1}^{N_{n\Delta t}} x_n \tag{5.11}$$

where N is a Poisson process with rate λa and there is the convention that an empty sum is 0. The simulation is based off of (2.7) and is easy, given that one knows how to simulate Lévy processes. Moreover, one can impose that

$$(y_0, y_{\Delta t}, y_{2\Delta t}, \ldots) \text{ and } (z_0, z_{\Delta t}, z_{2\Delta t}, \ldots)$$

are arrays of the same length by simulating the BDLP from new LKC $(\lambda\mu, \lambda\sigma^2, \lambda\nu)$ instead of the original LKC (μ, σ^2, ν). The IOUP is computed via Simpson's Rule.



(a) OUP with $\lambda = 10$.

(b) OUP with $\lambda = 100$.

(c) The IOUP given by Figure 5.10a.

(d) The IOUP given by Figure 5.10b.

Figure 5.10: Sample paths of a Gamma-OUP and Gamma-IOUP with $a = b = y_0 = 1$.

One may deduce from Figure 5.10 and Figure 15.1 in [13] that (5.11) is a sufficient approximation of (2.7). Alternatively, one may simulate

$$\mathrm{e}^{-\lambda t}\int_0^t \mathrm{e}^{\lambda s}\, \mathrm{d}z_{\lambda s} = \mathrm{e}^{-\lambda t}\int_0^{\lambda t} \mathrm{e}^{s}\, \mathrm{d}z_s$$

in (2.9) directly. A series representation is considered in §8.3 of [33], and simulation from the characteristic function is considered in both §3 of [34] and [15].

5.3 Monte Carlo Pricing

One cites [21], [33], and [35]. Suppose a European option is written on the underlying $S = (S_t)_{0\le t\le T}$ with maturity T and has a payoff function $V((S_t)_{0\le t\le T})$, that may be path dependent. Recall from Theorem 3.2 that the time-0 price of the option is

$$V_\star = \mathrm{e}^{-rT}\,\mathbb{E}[V((S_t)_{0\le t\le T})] \tag{5.12}$$

so the problem of pricing exotic options effectively boils down to the computation of an expectation. From the strong law of large numbers, Monte Carlo asserts that

$$\bar{V}_n = \frac{1}{n}\sum_{k=1}^{n} V((S_t^{(k)})_{0\le t\le T}) \xrightarrow{\text{a.s.}} \mathbb{E}[V((S_t)_{0\le t\le T})] \text{ as } n\to\infty$$

where $(S_t^{(k)})_{0\le t\le T}$ are independent simulations of S.

Algorithm 4: Simulation of the Underlying

1. Simulate a strictly positive mean-reverting process $y = \{y_t : 0\le t\le T\}$;
2. Compute the operational time $Y = \{Y_t = \int_0^t y_s\,\mathrm{d}s : 0\le t\le T\}$;
3. Simulate a Lévy process $L = \{X_t : 0\le t\le Y_T\}$;
4. Compute the time change of L by Y: $X = L\circ Y = \{X_t = L_{Y_t} : 0\le t\le T\}$;
5. Compute the underlying via (3.6) and $\mathbb{E}[\exp(L_{Y_t})|\theta] = \varphi_t(-i\psi(-i;\theta_\psi);\theta_\varphi)$ where φ_t and ψ are in the context of Theorem 3.7.

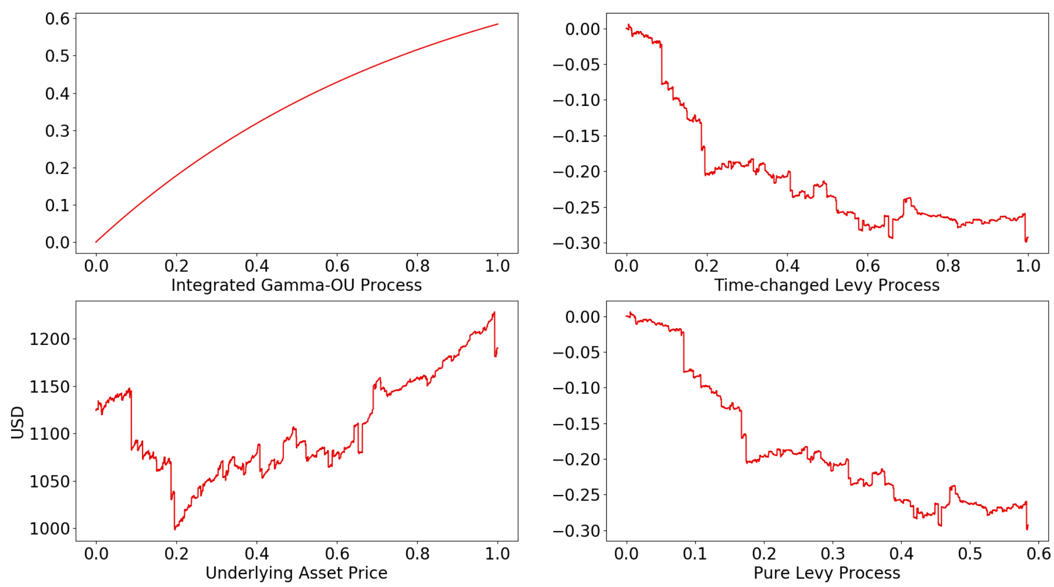


Figure 5.11: A sample path of the underlying over a 1 year period, modelled by the NIG-Gamma-OU with parameters from Table 4.1.

Remark. The standard error of $\bar{V}_n$ is $O(n^{-1/2})$ as $n\to\infty$, meaning convergence is incredibly slow. Increased efficiency, such as variance reduction by control variates or Quasi-Monte Carlo methods, can be found in [33] and [21] respectively.

Chapter 6

Final Remarks

6.1 Alternative Models

The model given in §3.3 is not the only way of describe an underlying asset. Other models from [10], [13], [17], [18], [19], [20], and [33] include the

1. GARCH models. These are discrete time models found in [17].
2. geometric fractional Brownian motion. It is a generalisation of the Black-Scholes model found in §1.6 of [19].
3. diffusion-based stochastic volatility models. These modify Black-Scholes to let volatility be driven by solutions to stochastic differential equations. It can be found in §15.1 of [13] and §2.3 of [20] but they do not incorporate Lévy processes or macroscopic jumps in the underlying. Notable examples include the Stein & Stein, and Heston models.
4. Bates model. It is a modification of the Heston model, incorporating log-normal jumps, found in §15.2 of [13] and §8.2.5 of [18].
5. exponential Lévy models. These generalise geometric Brownian motion and are found in §6 of [33] but do not exhibit stochastic volatility.
6. Barndorff-Nielsen and Shephard (BNS) models. These are found in §7 of [33] and modify Black-Scholes in several ways - see below.
7. Leveraged Lévy stochastic volatility models. This is a modification of (3.6) that further incorporates market shock and is found in [10] - see §6.2.

Let y be an OUP with a BDLP z that is independent of a standard Brownian motion W. Then the BNS model is given by

$$\mathrm{d}\log S_t = (\mu - \tfrac{1}{2}y_t)\,\mathrm{d}t + \sqrt{y_t}\,\mathrm{d}W_t + \rho\,\mathrm{d}z_{\lambda t} \qquad S_0 > 0 \tag{6.1}$$

where μ is some constant drift, and $\rho \leq 0$ is a leverage parameter. A risk neutral characteristic function is given in §7.1 of [33] so that the model can be calibrated and used to price options. One remarks that OUPs are not limited to §2.4 - superposition of OUPs are discussed in [3] and [4]. Since y and z are dependent, the parameter ρ offers interesting behaviour and this deserves a discussion.

6.2 Leverage

Leverage is mentioned in [4] as the observation that a fall in the underlying price is associated with an increase in future volatility. This behaviour occurs in (6.1) since the BDLP directly impacts the underlying price. In §6 of [10] and §8.2.7 of [18], it is incorporated into (3.6): consider

$$X_t = L_{Y_t} + \rho z_{\lambda t} \tag{6.2}$$

where L is a Lévy process, Y is a strictly increasing IOUP with BDLP z, and leverage parameter $\rho \leq 0$. Then

$$S_t = S_0 \frac{\exp(rt)}{\mathbb{E}[\exp(X_t)|\theta]} \exp(X_t) \qquad\qquad \theta \in \Theta \tag{6.3}$$

gives the risk-neutral underlying according to the "first-architecture" in [10]. Option pricing can be done via Monte Carlo provided one can find the characteristic function of $\log S_t$ to perform calibration from §4. One suppresses θ for notational brevity and finds the characteristic function using this basic observation

$$\begin{aligned}
\mathbb{E}[\exp(i\xi X_t)] &= \mathbb{E}[\exp(i\xi L_{Y_t} + i\xi\rho z_{\lambda t})] \\
&= \mathbb{E}[\mathbb{E}[\exp(i\xi L_{Y_t} + i\xi\rho z_{\lambda t})|Y_t]] \\
&= \mathbb{E}[\exp(Y_t\psi_L(\xi) + i\xi\rho z_{\lambda t})]
\end{aligned}$$

where ψ_L is the characteristic exponent of L.

Theorem 6.1. For every $\xi, \eta \in \mathbb{R}$,

$$\Phi_t(\xi, \eta) := \mathbb{E}[\exp(i\xi Y_t + i\eta z_{\lambda t})] = \exp\left(i\xi y_0 \frac{1 - \mathrm{e}^{-\lambda t}}{\lambda} + \int_\eta^{u_0} \frac{\lambda \psi_z(u)}{\xi + \lambda\eta - \lambda u}\,\mathrm{d}u\right)$$

where $u_0 = \eta + \xi(1 - \mathrm{e}^{-\lambda t})/\lambda$ and ψ_z is the characteristic exponent of z.

Proof. Apply (2.16) to see

$$\begin{aligned}
\Phi_t(\xi, \eta) &= \mathbb{E}\Big[\exp\Big(i\xi y_0 \frac{1 - \mathrm{e}^{-\lambda t}}{\lambda} + i\xi \int_0^t \frac{1 - \mathrm{e}^{\lambda(s-t)}}{\lambda}\,\mathrm{d}z_{\lambda s} + i\eta \int_0^t \mathrm{d}z_{\lambda s}\Big)\Big] \\
&= \exp\Big(i\xi y_0 \frac{1 - \mathrm{e}^{-\lambda t}}{\lambda}\Big)\mathbb{E}\Big[\exp\Big(i \int_0^t \Big(\xi \frac{1 - \mathrm{e}^{\lambda(s-t)}}{\lambda} + \eta\Big)\,\mathrm{d}z_{\lambda s}\Big)\Big]
\end{aligned}$$

and consider a new Lévy process $z' = (z'_t)_{t \geq 0}$ where $z'_t := z_{\lambda t}$. Then $\lambda\psi_z(\,\cdot\,)$ is the characteristic exponent of z'_1 and

$$\Phi_t(\xi, \eta) = \exp\Big(i\xi y_0 \frac{1 - \mathrm{e}^{-\lambda t}}{\lambda} + \int_0^t \lambda\psi_z\Big(\xi \frac{1 - \mathrm{e}^{\lambda(s-t)}}{\lambda} + \eta\Big)\,\mathrm{d}s\Big)$$

by Lemma 2.28. The u-substitution $u = \eta + \xi(1 - \mathrm{e}^{\lambda(s-t)})/\lambda$ yields the claim. □

Corollary 6.2. One has

$$\mathbb{E}[\exp(i\xi \log S_t)|S_0] = \exp(i\xi(rt + \log S_0)) \frac{\Phi_t(-i\psi_L(\xi), \xi\rho)}{\Phi_t(-i\psi_L(-i), -i\rho)^{i\xi}}$$

since $\mathbb{E}[\exp(i\xi X_t)] = \Phi_t(-i\psi_L(\xi), \xi\rho)$.

6.3 Conclusion

Lévy and Ornstein-Uhlenbeck processes were introduced to remedy counter-factual assumptions in the Black-Scholes model. Exponentiated Lévy processes, running on stochastic time change, were proposed as alternative models and calibrated. Most models proved to be highly flexible, exhibiting a good fit of the strike-maturity option surface. Some models did not – suggesting the need for more effective optimisation algorithms. After calibration, exotic options were no longer ignored: integration and Monte Carlo were presented as possible pricing methods. Many simulation techniques were demonstrated to verify Monte Carlo is not impossible. It also yielded tangible visualisation of stochastic processes! Possible avenues for further investigation could include alternative models, or modelling of stochastic interest rates.

Appendix

Contents of the Appendix:

1. Appendix A - SP500 Data
2. Appendix B - Calibration code (for Chapter 4)
3. Appendix C - Simulation code

    3a. Code for Chapter 1

    3b. Code for Chapter 2

    3c. Code for Chapter 5

Appendix A: SP500 Data

Strike	May 2002	June 2002	Sep. 2002	Dec. 2002	March 2003	June 2003	Dec. 2003
975			161.60	173.30			
995			144.80	157.00		182.10	
1025			120.10	133.10	146.50		
1050		84.50	100.70	114.80		143.00	171.40
1075		64.30	82.50	97.60			
1090	43.10						
1100	35.60		65.50	81.20	96.20	111.30	140.40
1110		39.50					
1120	22.90	33.50					
1125	20.20	30.70	51.00	66.90	81.70	97.00	
1130		28.00					
1135		25.60	45.50				
1140	13.30	23.20		58.90			
1150		19.10	38.10	53.90	68.30	83.30	112.80
1160		15.30					
1170		12.10					
1175		10.90	27.70	42.50	56.60		99.80
1200			19.60	33.00	46.10	60.90	
1225			13.20	24.90	36.90	49.80	
1250				18.30	29.30	41.20	66.90
1275				13.20	22.50		
1300					17.20	27.10	49.50
1325					12.80		
1350						17.10	35.70
1400						10.10	25.20
1450							17.00
1500							12.20

Figure A.1: Data from [33], where columns determine maturity and rows determine strikes of calls written on the S&P500 at the close of the market on 18/04/2002. On that day the S&P500 closed at 1124.47 where the effective interest rate was 0.7% per year.

Appendix B: Calibration Code

```
############################################################  794 lines of code
#Importing libraries
import numpy as np
import math
import cmath
import scipy.integrate as intgl
from scipy.integrate import simps
import scipy

import copy
import time
################################################################################
#Defining mathematical functions

#Characteristic exponent of the non-SV Levy process (Section 2.2)
def mPsi(Levy,theta,x):
    #Variance Gamma
    if Levy == 'VG':
        C = theta[0]
        G = theta[1]
        M = theta[2]

        if C>0 and G>0 and M>0:
            out = C*cmath.log((G*M)/(G*M + (M-G)*x*1j + x**2))
        else:
            print('Invalid VG parameters!')

    #CGMY
    if Levy == 'CGMY':
        C = theta[0]
        G = theta[1]
        M = theta[2]
        Y = theta[3]

        if C>0 and G>0 and M>0 and Y<2:
            out = C*scipy.special.gamma(-Y)*((M-x*1j)**Y - M**Y + (G+x*1j)**Y - G**Y)
        else:
            print('Invalid CGMY parameters!')

    #Normal Inverse Gaussian
    if Levy == 'NIG':
        A = theta[0]
        B = theta[1]
        D = theta[2]

        if A>0 and D>0 and B>-A and B<A:
            out = -D*( cmath.sqrt(A**2 - (B+x*1j)**2) - cmath.sqrt(A**2-B**2) )
        else:
            print('Invalid NIG parameters!')

    #Meixner
    if Levy == 'MXN':
        A = theta[0]
        B = theta[1]
        D = theta[2]

        if A>0 and D>0 and B>-math.pi and B<math.pi:
            out = 2*D*cmath.log(cmath.cos(B/2)/cmath.cosh((A*x - B*1j)/2))
        else:
            print('Invalid MXN parameters!')

    #Generalised Hyperbolic
    if Levy == 'GH':
        A = theta[0]
        B = theta[1]
        D = theta[2]
        U = theta[3]

        paramValid = False
        if A>0 or A==0:
            if U>0 and (D>0 or D==0) and B>-A and B<A:
                paramValid = True
            if U==0 and D>0 and B>-A and B<A:
                paramValid = True
            if U<0 and D>0 and (B>-A or B==-A) and (B<A or B==A):
                paramValid = True

            if paramValid == True:
                out = (((A**2-B**2)/(A**2-(B+x*1j)**2))**(U/2))*\
                          (scipy.special.kv(U,D*cmath.sqrt(A**2-(B+x*1j)**2))/\
                           scipy.special.kv(U,D*cmath.sqrt(A**2-B**2)))
            else:
                print('Invalid GH parameters')
    return(out)
```

Figure B.2: Python code for model calibration.

```
#Characteristic function of the GammaSV-IOUP (Section 2.5)
def mVarphi(theta,T,x):
    rate = theta[0]
    a = theta[1]
    b = theta[2]
    y0 = theta[3]

    if rate>0 and a>0 and b>0 and y0>0 and (T>0 or T==0):
        out = cmath.exp( (1j*x*y0*(1-math.exp(-rate*T))/rate) +\
                        (a/(1j*x-rate*b))*\
                         ( b*cmath.log(b*rate/(b*rate-1j*x*(1-math.exp(-rate*T)))) -\
                                  1j*x*T ) )
    else:
        print('Invalid Stochastic Volatility parameters')
    return(out)

#Characteristic function of the log-stockprice (Section 3.3)
def mPhi(theta,T,x,r,Levy):
    theta_SV = np.zeros([4])
    theta_SV[0] = theta[0]
    theta_SV[1] = theta[1]
    theta_SV[2] = theta[2]
    theta_SV[3] = theta[3]

    theta_Levy = np.zeros([4])
    theta_Levy[0] = theta[4]
    theta_Levy[1] = theta[5]
    theta_Levy[2] = theta[6]
    theta_Levy[3] = theta[7]

    out = cmath.exp(1j*x*r*T)*\
             ((mVarphi(theta_SV,T,-1j*mPsi(Levy,theta_Levy,x)))/\
              ((mVarphi(theta_SV,T,-1j*mPsi(Levy,theta_Levy,-1j)))**(1j*x)))
    return(out)

#Integrand appearing in the Call Option formula
def mIntegrand(theta,T,x,r,Levy,alpha,zeta):
    temp = (math.exp(-r*T)*mPhi(theta,T,x-(1+alpha)*1j,r,Levy))/\
             (alpha**2+alpha-x**2+(2*alpha+1)*x*1j)
    out = math.exp(-alpha*zeta)*cmath.exp(-1j*zeta*x)*temp/math.pi
    return(out)

################################################################################
#Computing Vanilla European Call Option

#Computation of single strike calls via Simpson's Rule
def compute_call(theta,T,r,S0,Levy,alpha,K):
    N = 512
    eta = 0.25
    axis = np.linspace(0.0, N*eta, N)
    integrand = np.zeros([N])

    T = T/12 #T measured in months, not years
    zeta = math.log(K/S0) #change of numeraire

    for j in range(0,N):
        integrand[j] = mIntegrand(theta,T,j*eta,r,Levy,alpha,zeta).real
    out = simps(integrand, axis)
    out = S0*out #change of numeraire
    return(out)

################################################################################
#Data
def data_Schoutens():
    schoutensVector = np.zeros([75,3])

    #[x,0] = strike
    #[x,1] = time in months
    #[x,2] = market price
    #eg
    schoutensVector[0,0] = 975
    schoutensVector[0,1] = 5
    schoutensVector[0,2] = 161.60
    #and so on for x=0,1,2,...,74

              ⋮

    schoutensVector[74,0] = 1500
    schoutensVector[74,1] = 20
    schoutensVector[74,2] = 12.20
    return(schoutensVector)
```

```
def paramSchoutens(Levy):
    theta = np.zeros([8])

    #Variance Gamma
    if Levy == 'VG':
        theta[0] = 1.2517
        theta[1] = 0.5841
        theta[2] = 0.6282
        theta[3] = 1
        theta[4] = 11.4838
        theta[5] = 23.2880
        theta[6] = 40.1291

    #CGMY
    if Levy == 'CGMY':
        theta[0] = 0.8838
        theta[1] = 0.5946
        theta[2] = 0.8524
        theta[3] = 1
        theta[4] = 0.0415
        theta[5] = 3.9092
        theta[6] = 24.940
        theta[7] = 1.3664

    #Normal Inverse Gaussian
    if Levy == 'NIG':
        theta[0] = 0.6252
        theta[1] = 0.4239
        theta[2] = 0.5962
        theta[3] = 1
        theta[4] = 29.4722
        theta[5] = -15.9048
        theta[6] = 0.5071

    #Meixner
    if Levy == 'MXN':
        theta[0] = 1.22069182
        theta[1] = 0.72096088
        theta[2] = 2.33030738
        theta[3] = 0.55849243
        theta[4] = 0.07287571
        theta[5] = -1.38054549
        theta[6] = 13.01831038

    #Generalised Hyperbolic
    if Levy == 'GH':
        theta[0] = 88006.0692
        theta[1] = 0.0508387971
        theta[2] = 0.0000538540592
        theta[3] = 319.704367
        theta[4] = 0.0313801132
        theta[5] = 0.00754456200
        theta[6] = 5.13786079
        theta[7] = -4.42095282
    return(theta)

def testSchoutensFit(Levy):
    data = data_Schoutens()
    theta = paramSchoutens(Levy)
    E=0

    S0 = 1124.27
    r = 0.007
    alpha = 0.75

    for i in range(0,75):
        if data[i,1] == 1:
            month = 'May 2002       '
        elif data[i,1] == 2:
            month = 'June 2002      '
        elif data[i,1] == 5:
            month = 'September 2002'
        elif data[i,1] == 8:
            month = 'December 2002 '
        elif data[i,1] == 11:
            month = 'March 2003     '
        elif data[i,1] == 14:
            month = 'June 2003      '
        elif data[i,1] == 20:
            month = 'December 2003 '
        C = compute_call(theta,data[i,1],r,S0,Levy,alpha,data[i,0])
        E = E + (C - data[i,2])**2
        C = math.floor(100*C)/100
        if C == data[i,2]:
            print('wow ',i)
        #print('Maturity =',month, 'Strike =',data[i,0],\
        #     'Actual =',data[i,2],'Prediction =',C,'Difference =',\
        #      math.floor(100*abs(C - data[i,2]))/100)
        print('Actual = $',data[i,2],'Prediction = $',C,'Difference = $',\
              math.floor(100*abs(C - data[i,2]))/100)
    E = math.sqrt(E/75)
    print('RMSE =',E)
    return()
```

```
##############################################################################
#Optimisation

def tauInv(Levy,theta):
    out = np.zeros([8])
    for i in range(0,4):
        out[i] = math.log(theta[i])

    #Variance Gamma
    if Levy == 'VG':
        for i in range(4,7):
            out[i] = math.log(theta[i])

    #CGMY
    if Levy == 'CGMY':
        for i in range(4,7):
            out[i] = math.log(theta[i])
        out[7] = math.log(2-theta[7])

    #Normal Inverse Gaussian
    if Levy == 'NIG':
        out[4] = math.log(theta[4])
        out[5] = math.tan(math.pi*(((theta[5]-theta[4])/theta[4])-1)*0.5)
        out[6] = math.log(theta[6])

    #Meixner
    if Levy == 'MXN':
        out[4] = math.log(theta[4])
        out[5] = math.tan(0.5*theta[5])
        out[6] = math.log(theta[6])

    #Generalised Hyperbolic
    if Levy == 'GH':
        out[4] = math.log(theta[4])
        out[5] = math.tan(math.pi*(((theta[5]-theta[4])/theta[4])-1)*0.5)
        out[6] = math.log(theta[6])
        out[7] = theta[7]
    return(out)

def tau(Levy,pos):
    out = np.zeros([8])
    for i in range(0,4):
        out[i] = math.exp(pos[i])

    #Variance Gamma
    if Levy == 'VG':
        for i in range(4,7):
            out[i] = math.exp(pos[i])

    #CGMY
    if Levy == 'CGMY':
        for i in range(4,7):
            out[i] = math.exp(pos[i])
        out[7] = 2-math.exp(pos[7])

    #Normal Inverse Gaussian
    if Levy == 'NIG':
        out[4] = math.exp(pos[4])
        out[5] = 2*out[4]*math.atan(pos[5])/math.pi
        out[6] = math.exp(pos[6])

    #Meixner
    if Levy == 'MXN':
        out[4] = math.exp(pos[4])
        out[5] = 2*math.atan(pos[5])
        out[6] = math.exp(pos[6])

    #Generalised Hyperbolic
    if Levy == 'GH':
        out[4] = math.exp(pos[4])
        out[5] = 2*out[4]*math.atan(pos[5])/math.pi
        out[6] = math.exp(pos[6])
        out[7] = pos[7]
    return(out)

def Err(data,Levy,pos):
    theta = tau(Levy,pos)
    E=0
    dataSize = len(data)

    S0 = 1124.27
    r = 0.007
    alpha = 0.75

    for i in range(0,dataSize):
        C = compute_call(theta,data[i,1],r,S0,Levy,alpha,data[i,0])
        E = E + (C - data[i,2])**2
    E = math.sqrt(E/dataSize)
    return(E)
```

```
def stopYesNo(Error,n):
    if Error < 10**-4 or n > 1000:
        out = True
    else:
        out = False
    return(out)

def initialSimp(Levy):
    if Levy == 'CGMY' or Levy == 'GH':
        dim = 8
    else:
        dim = 7
    out = np.zeros([dim+1,dim+1])
    for i in range(0,dim+1):
        for j in range(0,dim):
            out[i,j] = np.random.normal()
    print('Initial simplex =',out)
    return(out)

def reflect(c,p,t):
    d = len(c)
    out = np.zeros([d])
    for i in range(0,d):
        out[i] = c[i] + t*(p[i] - c[i])
    return(out)

def optimise(Levy):
    simplex = initialSimp(Levy)
    d = len(simplex)-1
    val = np.zeros([d+1])
    sortedSimplex = np.zeros([d+1,d+1])
    data = data_Schoutens()
    out = np.zeros([d])

    stopCriterion = False
    carry = False
    n = 0

    while not stopCriterion:
        print('n =',n)
        checkDegenerate = np.zeros([d,d])
        for i in range(0,d):
            for j in range(0,d):
                checkDegenerate[i,j] = simplex[i+1,j] - simplex[0,j]
        while np.linalg.det(checkDegenerate) == 0:
            print('You should probably restart and obtain a new simplex!')
            time.sleep(30)
            for j in range(0,d):
                simplex[d,j] = np.random.normal()
        if carry == True:
            for i in range(0,d):
                val[i] = sortedSimplex[i,d]
            val[d] = sortedSimplex[d,d]
            carry = False
        else:
            for i in range(0,d+1):
                pos = np.zeros([d])
                for j in range(0,d):
                    pos[j] = simplex[i,j]
                val[i] = Err(data,Levy,pos)
        for i in range(0,d+1):
            simplex[i,d] = val[i]
        tempSorted = np.sort(val)
        for i in range(0,d+1):
            for j in range(0,d+1):
                if simplex[j,d] == tempSorted[i]:
                    sortedSimplex[i] = simplex[j]
        centroid = np.zeros([d])
        badPoint = np.zeros([d])
        for j in range(0,d):
            for i in range(0,d):
                centroid[j] = centroid[j] + sortedSimplex[i,j]
            centroid[j] = centroid[j]/d
            badPoint[j] = sortedSimplex[d,j]
        reflectOne = reflect(centroid, badPoint, -1)
        Err_rOne = Err(data,Levy,reflectOne)

        carry = True
        if (sortedSimplex[0,d] < Err_rOne or sortedSimplex[0,d] == Err_rOne) and\
           Err_rOne < sortedSimplex[d,d]:
            for j in range(0,d):
                sortedSimplex[d,j] = reflectOne[j]
            sortedSimplex[d,d] = Err_rOne
            print('reflect -1')
        elif Err_rOne < sortedSimplex[0,d]:
            reflectTwo = reflect(centroid, badPoint, -2)
            Err_rTwo = Err(data,Levy,reflectTwo)
            if Err_rTwo < Err_rOne:
                for j in range(0,d):
                    sortedSimplex[d,j] = reflectTwo[j]
                sortedSimplex[d,d] = Err_rTwo
                print('reflect -2')
```

```
            else:
                for j in range(0,d):
                    sortedSimplex[d,j] = reflectOne[j]
                sortedSimplex[d,d] = Err_rOne
                print('reflect -1')
        elif Err_rOne > sortedSimplex[d-1,d] or Err_rOne == sortedSimplex[d-1,d]:
            #boolean values
            booA = False
            booB = False
            if Err_rOne < sortedSimplex[d,d]:
                reflectHalf = reflect(centroid, badPoint, -0.5)
                Err_rHalf = Err(data,Levy,reflectHalf)
                if Err_rHalf < Err_rOne or Err_rHalf == Err_rOne:
                    for j in range(0,d):
                        sortedSimplex[d,j] = reflectHalf[j]
                    sortedSimplex[d,d] = Err_rHalf
                    booA = True
                    print('contract -1/2')
            else:
                shrinkHalf = reflect(centroid, badPoint, 0.5)
                Err_sHalf = Err(data,Levy,shrinkHalf)
                if Err_sHalf < sortedSimplex[d,d]:
                    for j in range(0,d):
                        sortedSimplex[d,j] = shrinkHalf[j]
                    sortedSimplex[d,d] = Err_sHalf
                    booB = True
                    print('contract 1/2')
            if not (booA or booB):
                carry = False
                print('shrink!')
                for i in range(1,d+1):
                    for j in range(0,d):
                        sortedSimplex[i,j] = (sortedSimplex[0,j] + sortedSimplex[i,j])/2

        simplex = copy.copy(sortedSimplex)
        print('0thRMSE =',simplex[0,d],'dthRMSE =',simplex[d-1,d],\
              'newest RMSE =', simplex[d,d])
        n = n+1
        for i in range(0,d):
            out[i] = simplex[0,i]
        stopCriterion = stopYesNo(Err(data,Levy,out),n)
        theta = tau(Levy,out)
        print('theta ',theta)

    theta = tau(Levy,out)
    print('Final output is ',theta)
    print('RMSE ',simplex[0,d])
    print('recalc RMSE ',Err(data,Levy,out))
    return()
################################################################################
#Execute Code

#Calibrate models
optimise('NIG')
#VG/ CGMY/ NIG/ MXN/ GH

#Test model fit
#testSchoutensFit('GH')
#VG/ CGMY/ NIG/ MXN/ GH
```

Appendix C: Simulation Code

Chapter 1

```
####################################
##Import libraries
import numpy as np
import pylab
import scipy.stats as stats
import matplotlib.pyplot as plt
import matplotlib

##Set font size
plt.rcParams.update({'font.size': 32})
####################################
##Import raw_data (daily stock price of SP500)
stocks = np.loadtxt('index-sp500-19700101-20200324.txt', delimiter = ',', skiprows = 1)
t_stocks = np.linspace(1970, 2020.23, len(stocks))
#Note: 2020.23 is to take into account the months (march 2020)

##Convert raw_data to daily log returns
logstocks = np.diff(np.log(stocks), n=1, axis=0)
t_logstocks = np.linspace(1970, 2020.23, len(logstocks))

##Summary statistics
mu = np.mean(logstocks)
print('average growth of SP500 =',mu)
sigma = np.std(logstocks)
print('standard deviation of SP500 =',sigma)

##Produce white noise
norm_noise = np.zeros([len(logstocks)])
for j in range(0,len(logstocks)):
    norm_noise[j] = sigma*np.random.normal(0,1)
####################################
##Plot raw_data
plt.figure(1)
plt.plot(t_stocks, stocks,'xkcd:red')
plt.xlabel('Year')
plt.ylabel('USD')

##QQ plot
fig, ax = plt.subplots()
stats.probplot(logstocks, fit=False, plot=pylab)
ax.get_lines()[1].remove()
matplotlib.pyplot.title("")
matplotlib.pyplot.xlabel("Theoretical Gaussian Quantiles")
matplotlib.pyplot.ylabel("Empirical Quantiles")

##Plot comparison between log returns and white noise
fig, axs = plt.subplots(2)
axs[0].plot(t_logstocks, logstocks,'xkcd:red')
axs[0].plot(t_logstocks, norm_noise,'darkblue')
axs[1].plot(t_logstocks, norm_noise,'darkblue')
axs[1].plot(t_logstocks, logstocks,'xkcd:red')
plt.xlabel('Year')

##Plot alternative comparison
plt.figure(4)
plt.plot(t_logstocks, logstocks,'xkcd:red', label='Daily SP500 Log-returns')
plt.plot(t_logstocks, norm_noise,'darkblue', label='White Noise')
plt.xlabel('Year')
plt.ylabel('Percentage Change')
plt.legend(loc='upper left')
####################################
##Geometric Brownian motion
N = len(stocks)
GBM = np.zeros([N])
GBM[0] = 93.5

plt.figure(5)
plt.xlabel('Year')
for i in range(0,1):
    for j in range(0,N-1):
        GBM[j+1] = GBM[j]*np.exp(mu - 0.5*sigma**2 + sigma*np.random.normal(0,1))
    plt.plot(t_stocks,GBM,'darkblue')
####################################
plt.show()
pylab.show()
plt.close()
####################################
```

Figure C.3: Python code for 1.

Chapter 2

```
###################################
##Import libraries
import numpy as np
import matplotlib.pyplot as plt
import math

##Set font size
plt.rcParams.update({'font.size': 32})
###################################
##Sample andom variables
def rv_u():
    return(np.random.uniform(0,1))

def rv_exp(rate):
    #rate = "lambda" so the mean is 1/rate
    if not rate > 0:
        print('Exponential random variable failed to generate!')
        forceTerminatePython = 1/0
    return(-math.log(rv_u())/rate)
###################################
##Simulation

##Brownian Motion
def BM(sigma):
    output = np.zeros([N])
    for j in range(0,N-1):
        output[j+1] = output[j] + sigma*np.sqrt(dt)*np.random.normal(0,1)
    return(output)

##Poisson Process
def fastsim_PoissonProcess(rate):
    if not rate > 0:
        print('Poisson process failed to generate!')
        forceTerminatePython = 1/0
    len_out = N
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + np.random.poisson(rate*dt)
    return(out)

##BDLP
def sim_BDLP(a,b):
    len_out = N
    out = np.zeros([len_out])
    pp = fastsim_PoissonProcess(a)
    for j in range(1,len_out):
        diff = 0
        for k in range(0, math.floor(pp[j]-pp[j-1])):
            diff += rv_exp(b)
        out[j] = out[j-1] + diff
    return(out)

##Classical Ornstein-Uhlenbeck
def Vasicek(a,b,sig,start):
    output = np.zeros([N])
    output[0] = start
    for j in range(0,N-1):
        output[j+1] = output[j] + a*(b - output[j])*dt \
                      + sig*np.sqrt(dt)*np.random.normal(0,1)
    return(output)

##Cox Ingersoll Ross Process
def CIR(kappa,eta,lam,start):
    output = np.zeros([N])
    output[0] = start
    for j in range(0,N-1):
        output[j+1] = output[j] + kappa*(eta - output[j])*dt \
                      + lam*np.sqrt(output[j])*np.random.normal(0,1)*np.sqrt(dt)
    return(output)
```

Figure C.4: Python code for Chapter 2.

```
##Gamma Ornstein-Uhlenbeck
def gamma_OU(rate,a,b,start):
    len_out = N
    out = np.zeros([len_out])
    out[0] = start
    bdlp = sim_BDLP(a*rate,b)
    for j in range(1,len_out):
        out[j] = out[j-1]*(1-rate*dt) + bdlp[j] - bdlp[j-1]
    return(out)
###################################
##Graphing
def addPosJumps(graph):
    out = np.zeros([len(graph)])
    for j in range(0,len(graph)-1):
        if graph[j+1] - graph[j] > 0:
            out[j] = np.nan
        else:
            out[j] = graph[j]
    out[len(graph)-1] = graph[len(graph)-1]
    return(out)

##BM vs Classical_OU (Vasicek_OU)
def plot_MeanReversionDemo():
    for i in range(0,nSim-1):
        plt.plot(t_axis,BM(1),'xkcd:red')
    plt.plot(t_axis,BM(1),'xkcd:red', label='Standard Bronwian Motion')

    for i in range(0,nSim-1):
        plt.plot(t_axis,Vasicek(1,0,1,0),'darkblue')
    plt.plot(t_axis,Vasicek(1,0,1,0),'darkblue', label='Vasicek Ornstein-Uhlenbeck')
    return()

##Cox Ingersoll Ross
def plot_CIR():
    plt.plot(t_axis,CIR(3,1,1.25,1.2),'darkblue')
    mean = np.zeros([N])
    mean[:] = 1
    plt.plot(t_axis,mean,'xkcd:red')
    plt.plot(t_axis,np.zeros([N]),'black')
    return()

##Gamma Driven OU
def plot_OU():
    OU = gamma_OU(10, 10, 100, 0.08)
    plt.plot(t_axis,addPosJumps(OU),'darkblue')
    return()
###################################
##Execute code

nSim = 10
totalTime = 20
resolution = 1000
dt = 1/resolution
N = totalTime*resolution
t_axis = np.linspace(0, totalTime, N)

plt.figure(1)
plot_MeanReversionDemo()

plt.figure(2)
plot_CIR()

totalTime = 1
resolution = 10000
dt = 1/resolution
N = totalTime*resolution
t_axis = np.linspace(0, totalTime, N)

plt.figure(3)
plot_OU()

plt.show()
```

Chapter 5

```
################################################################################
#Importing Libraries
import numpy as np
import matplotlib.pyplot as plt
import math
import cmath
import scipy
import scipy.integrate as intgl
from scipy.integrate import simps

import copy
import time
################################################################################
#Global Variables and Misc

#Number of steps per unit time
resolution = 10000
#Time step
dt = 1/resolution

plt.rcParams.update({'font.size': 16})
################################################################################
#Random Variable Sampling
def rv_u():
    return(np.random.uniform(0,1))

def rv_uab(a,b):
    return(a + ((b-a)*rv_u()))

def rv_sn():
    return(np.random.normal(0,1))

def rv_exp(rate):
    #rate = "lambda" so the mean is 1/rate
    if not rate > 0:
        print('Exponential random variable failed to generate!')
        forceTerminatePython = 1/0
    return(-math.log(rv_u())/rate)

def rv_gam(a,b):
    if a>0 and a<1:
        x = 1
        y = 1
        while x+y > 1:
            u1 = rv_u()
            u2 = rv_u()
            x = u1**(1/a)
            y = u2**(1/(1-a))
        u3 = rv_u()
        u4 = rv_u()
        out = -x*math.log(u3*u4)
        out = out/b
    else:
        print('Gamma random variable failed to generate!')
        forceTerminatePython = 1/0
    return(out)

def rv_ig(a,b):
    if not (a > 0 and b > 0):
        print('Inverse Gaussian random variable failed to generate!')
        forceTerminatePython = 1/0
    v = rv_sn()
    y = v**2
    x = (a/b) + y/(2*(b**2)) - np.sqrt((4*a*b*y)+(y**2))/(2*(b**2))
    u = rv_u()
    if u > a/(a + x*b):
        out = (a**2)/((b**2)*x)
    else:
        out = x
    return(out)

def rv_lap(scale):
    if not scale > 0:
        print('Laplace random variable failed to generate!')
        forceTerminatePython = 1/0
    u = rv_u()
    return(-scale*np.sign(u-0.5)*math.log(1-(2*abs(u-0.5))))
################################################################################
def addPosJumps(graph):
    out = np.zeros([len(graph)])
    for j in range(0,len(graph)-1):
        if graph[j+1] - graph[j] > 0:
            out[j] = np.nan
        else:
            out[j] = graph[j]
    out[len(graph)-1] = graph[len(graph)-1]
    return(out)
```

Figure C.5: Python code for simulating stochastic processes.

```
def addNegJumps(graph):
    out = np.zeros([len(graph)])
    for j in range(0,len(graph)-1):
        if graph[j+1] - graph[j] < 0:
            out[j] = np.nan
        else:
            out[j] = graph[j]
    out[len(graph)-1] = graph[len(graph)-1]
    return(out)

def addJumps(graph):
    out = np.zeros([len(graph)])
    for j in range(0,len(graph)-1):
        if abs(graph[j+1] - graph[j]) > 0.1:
            out[j] = np.nan
        else:
            out[j] = graph[j]
    out[len(graph)-1] = graph[len(graph)-1]
    return(out)

def intOU(ou,totalTime):
    N = len(ou)
    axis = np.linspace(0.0, totalTime, N)
    out = np.zeros([N])
    integrand = np.zeros([N])
    for j in range(0,N):
        integrand[j] = ou[j]
        out[j] = simps(integrand, axis)
    return(out)
##############################################################################
#Simulation (Discretisation)
def sim_drift(totalTime,slope):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(0,len_out):
        out[j] = slope*(j*dt)
    return(out)

def sim_PoissonProcess(totalTime,rate):
    if not rate > 0:
        print('Warning: Poisson process!')
        forceTerminatePython = 1/0
    arrivalTime = 0.0
    arrivalTimes = []
    while arrivalTime < totalTime:
        arrivalTimes.append(arrivalTime)
        arrivalTime += rv_exp(rate)
    arrivalTimes.append(totalTime)
    #print(arrivalTimes)

    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(0,len_out):
        k = 0
        while not(j >= float(arrivalTimes[k])*resolution and \
                  j < float(arrivalTimes[k+1])*resolution):
            k += 1
        out[j] = k
    return(out)

def fastsim_PoissonProcess(totalTime,rate):
    if not rate > 0:
        print('Warning: Poisson process!')
        forceTerminatePython = 1/0
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + np.random.poisson(rate*dt)
    return(out)

def sim_stdBM(totalTime):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + math.sqrt(dt)*rv_sn()
    return(out)

def sim_subGamma(totalTime,a,b):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + rv_gam(a*dt,b)
    return(out)

def sim_VG(totalTime,C,G,M):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    g1 = sim_subGamma(totalTime,C,M)
    g2 = sim_subGamma(totalTime,C,G)
    for j in range(1,len_out):
        out[j] = g1[j] - g2[j]
    return(out)
```

```
def sim_subIG(totalTime,a,b):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + rv_ig(a*dt,b)
    return(out)

def sim_NIG(totalTime,a,b,d):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    ig = sim_subIG(totalTime,1,d*math.sqrt((a**2)-(b**2)))
    for j in range(1,len_out):
        out[j] = out[j-1] + b*(d**2)*(ig[j]-ig[j-1]) +\
                 d*math.sqrt(ig[j]-ig[j-1])*rv_sn()
    return(out)

def sim_BDLP(totalTime,a,b):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    pp = fastsim_PoissonProcess(totalTime,a)
    for j in range(1,len_out):
        diff = 0
        for k in range(0, math.floor(pp[j]-pp[j-1])):
            diff += rv_exp(b)
        out[j] = out[j-1] + diff
    return(out)

def sim_OU_G(totalTime,rate,a,b,initial):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    out[0] = initial
    bdlp = sim_BDLP(totalTime,a*rate,b)
    for j in range(1,len_out):
        out[j] = out[j-1]*(1-rate*dt) + bdlp[j] - bdlp[j-1]
    return(out)
################################################################################
#Simulation (Compound Poisson Approx)

#Meixner
def MXN_func1(a,b,x):
    #returns sinh(bx/a)/sinh(pi*x/a)
    y = x/a
    return((math.exp(y*(b-math.pi)) - math.exp(-y*(b+math.pi)))/\
           (1 - math.exp(-2*math.pi*y)))

def drift_MXN(a,b,d):
    N = 2**20
    h = 2**-7
    axis = np.linspace(0,(N-1)*h,N)
    integrand = np.zeros([N])
    for j in range(0,N):
        integrand[j] = MXN_func1(a,b,j*h+1)
    Integral = simps(integrand, axis)
    return((a*d*math.tan(0.5*b)) - (2*d*Integral))

def MXN_func2(a,b,d,x):
    #Levy Density
    #returns d*exp(bx/a)/x*sinh(pi*x/a)
    y = x/a
    return((2*d/x)*(math.exp(y*(b-math.pi))/(1 - math.exp(-2*math.pi*y))))

def MXN_func3(a,b,d,x):
    #x*MXN_func2
    #returns d*exp(bx/a)/sinh(pi*x/a)
    y = x/a
    return((2*d)*(math.exp(y*(b-math.pi))/(1 - math.exp(-2*math.pi*y))))

def MXN_iid_c(eps,a,b,d,mass,scale):
    return(4*d*scale/(mass*eps*(1-math.exp(-2*math.pi*eps/a))))

def MXN_probDensity(eps,a,b,d,mass,x):
    if x > eps:
        out = (2*d*math.exp((b-math.pi)*x/a))/(mass*x*(1-math.exp(-2*math.pi*x/a)))
    elif x < -eps:
        out = (2*d*math.exp((b+math.pi)*x/a))/(mass*x*(math.exp(2*math.pi*x/a)-1))
    else:
        out = 0
    return(out)

def MXN_iid_g(scale,x):
    return((1/(2*scale))*math.exp(-abs(x)/scale))

def MXN_iid(eps,a,b,d,mass,c,scale):
    temp = -1
    while rv_u() > temp:
        out = rv_lap(scale)
        temp = MXN_probDensity(eps,a,b,d,mass,out)/(c*MXN_iid_g(scale,out))
    return(out)
```

```
def sim_MXN(totalTime,eps,a,b,d):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    ppp = np.zeros([len_out]) #PoissonPointProcess

    #Compute Compensating Drift
    N = 2**16
    h = (1-eps)/N
    axisPos = np.linspace(0,((N-1)*h)-eps,N)
    integrand = np.zeros([N])
    for j in range(0,N):
        integrand[j] = MXN_func3(a,b,d,j*h+eps)
    compen_drift = -simps(integrand, axisPos)

    axisNeg = np.linspace(eps-((N-1)*h),0,N)
    for j in range(0,N):
        integrand[j] = MXN_func3(a,b,d,-j*h-eps)
    compen_drift -= simps(integrand, axisNeg)

    #Compute Total Drift
    drift = drift_MXN(a,b,d) + compen_drift

    #Compute rate of Poisson process
    N = 2**12
    h = 2**-4
    axisPos = np.linspace(0,(N-1)*h,N)
    integrand = np.zeros([N])
    for j in range(0,N):
        integrand[j] = MXN_func2(a,b,d,j*h+eps)
    rate = simps(integrand, axisPos)

    axisNeg = np.linspace((1-N)*h,0,N)
    for j in range(0,N):
        integrand[j] = MXN_func2(a,b,d,-j*h-eps)
    rate += simps(integrand, axisNeg)

    #Simulate Poisson point process
    pp = fastsim_PoissonProcess(totalTime,rate)
    print(pp[len(pp)-1])
    time.sleep(2)

    if b+math.pi > b-math.pi:
        temp = (b+math.pi)/a
    else:
        temp = (b-math.pi)/a
    scale = 1/temp
    c = MXN_iid_c(eps,a,b,d,rate,scale)
    indicator = 0
    for j in range(1,len_out):
        progress = 100*j/len_out
        if math.floor(progress) > indicator:
            print('Progress at ',progress,'%')
            indicator += 1
        diff = 0
        for k in range(0, math.floor(pp[j]-pp[j-1])):
            diff += MXN_iid(eps,a,b,d,rate,c,scale)
        out[j] = out[j-1] + diff + drift*dt
        ppp[j] = diff
    return(out,ppp)

#Martingale Example
def mtg_probDensity(a,b,mass,x):
    if x > -b and x < -a:
        out = ((-x)**-2.5)/mass
    else:
        out = 0
    return(out)

def mtg_iid_g(a,b,x):
    return(1/(b-a))

def mtg_iid(a,b,mass,c):
    temp = -1
    while rv_u() > temp:
        out = rv_uab(-b,-a)
        temp = mtg_probDensity(a,b,mass,out)/(c*mtg_iid_g(-b,-a,out))
    return(out)

def sim_mtg(totalTime,a,b):
    len_out = totalTime*resolution
    out = np.zeros([len_out])

    drift = 2*((a**-0.5)-(b**-0.5))
    rate = (2/3)*((a**-1.5)-(b**-1.5))

    pp = fastsim_PoissonProcess(totalTime,rate)

    c = (b-a)*(a**-2.5)/rate
    for j in range(1,len_out):
        diff = 0
        for k in range(0, math.floor(pp[j]-pp[j-1])):
            diff += mtg_iid(a,b,rate,c)
        out[j] = out[j-1] + diff + drift*dt
    return(out)
```

```
################################################################################
#Graphing
def plotDrift():
    T = 10
    m = 1
    line = sim_drift(T,m)
    plt.plot(np.linspace(0.0, T, len(line)), line, 'xkcd:red')
    plt.show(block=False)
    return()

def plotPoisson():
    T = 10
    L = 1
    pp = addPosJumps(sim_PoissonProcess(T,L))
    plt.plot(np.linspace(0.0, T, len(pp)), pp, 'xkcd:red')
    plt.show(block=False)
    return()

def plotstdBM():
    T = 16
    bm = sim_stdBM(T)
    plt.plot(np.linspace(0.0, T, len(bm)), bm, 'xkcd:red')
    plt.show(block=False)
    return()

def plotsubGamma():
    T = 1
    a = 5
    b = 10
    lp = sim_subGamma(T,a,b)
    plt.plot(np.linspace(0.0, T, len(lp)), lp, 'xkcd:red')
    plt.show(block=False)
    return()

def plotVG():
    T = 10
    C = 1
    G = 5
    M = 6
    lp = sim_VG(T,C,G,M)
    plt.plot(np.linspace(0.0, T, len(lp)), lp, 'xkcd:red')
    plt.show(block=False)
    return()

def plotsubIG():
    T = 1
    a = 1
    b = 20
    lp = sim_subIG(T,a,b)
    plt.plot(np.linspace(0.0, T, len(lp)), lp, 'xkcd:red')
    plt.show(block=False)
    return()

def plotNIG():
    T = 1
    a = 50
    b = -10
    d = 1
    lp = sim_NIG(T,a,b,d)
    plt.rcParams.update({'font.size': 20})
    plt.plot(np.linspace(0.0, T, len(lp)), lp, 'xkcd:red')
    plt.show(block=False)
    return()

def plotBDLP():
    T = 20
    a = 0.8696
    b = 2.3268
    cpp = sim_BDLP(T,a,b)
    plt.plot(np.linspace(0.0, T, len(cpp)), addPosJumps(cpp), 'xkcd:red')
    plt.show(block=False)
    return()

def plotouG():
    plt.rcParams.update({'font.size': 48})
    T = 1

    plt.figure(1)
    L = 10
    a = 1
    b = 1
    y0 = 1
    ou = sim_OU_G(T,L,a,b,y0)
    axis = np.linspace(0.0, T, len(ou))
    plt.plot(axis, addPosJumps(ou), 'xkcd:red')

    plt.figure(2)
    iou = intOU(ou,T)
    plt.plot(axis, iou, 'xkcd:red')
```

```
    plt.figure(3)
    L = 100
    a = 1
    b = 1
    y0 = 1
    ou = sim_OU_G(T,L,a,b,y0)
    plt.plot(axis, addPosJumps(ou), 'xkcd:red')

    plt.figure(4)
    iou = intOU(ou,T)
    plt.plot(axis, iou, 'xkcd:red')

    plt.show(block=False)
    return()

def plotMXN():
    T = 1
    a = 25
    b = 0
    d = 1
    eps = 0.001
    [lp,ppp] = sim_MXN(T,eps,a,b,d)
    for j in range(0,len(ppp)):
        if ppp[j] == 0:
            ppp[j] = np.nan

    plt.figure(1)
    plt.rcParams.update({'font.size': 20})
    plt.plot(np.linspace(0.0, T, len(lp)), lp, 'xkcd:red')

    plt.figure(2)
    plt.rcParams.update({'font.size': 48})
    plt.scatter(np.linspace(0.0, T, len(ppp)), ppp, c='xkcd:red', marker='.')
    plt.show(block=False)
    return()

def plotMtg():
    T = 15
    plt.rcParams.update({'font.size': 20})

    plt.subplot(2,2,1)
    a = 1
    b = 3
    mtg = sim_mtg(T,a,b)
    plt.plot(np.linspace(0.0, T, len(mtg)), addNegJumps(mtg), 'xkcd:red')
    plt.xlabel('\u03B5 = 1')

    plt.subplot(2,2,2)
    a = 0.3
    b = 1
    mtg += sim_mtg(T,a,b)
    plt.plot(np.linspace(0.0, T, len(mtg)), addNegJumps(mtg), 'xkcd:red')
    plt.xlabel('\u03B5 = 0.3')

    plt.subplot(2,2,3)
    a = 0.1
    b = 0.3
    mtg += sim_mtg(T,a,b)
    plt.plot(np.linspace(0.0, T, len(mtg)), addNegJumps(mtg), 'xkcd:red')
    plt.xlabel('\u03B5 = 0.1')

    plt.subplot(2,2,4)
    a = 0.01
    b = 0.1
    mtg += sim_mtg(T,a,b)
    plt.plot(np.linspace(0.0, T, len(mtg)), addNegJumps(mtg), 'xkcd:red')
    plt.xlabel('\u03B5 = 0.01')

    plt.show(block=False)
    return()
##############################################################################
#Execute code:
#Uncomment one of the lines below to simulate a process

#plotDrift()
#plotPoisson()
#plotstdBM()
#plotsubGamma()
#plotVG()
#plotsubIG()
#plotNIG()
#plotBDLP()
#plotouG()
#plotMXN()
plotMtg()
```

```python
################################################################################
#Importing Libraries
import numpy as np
import matplotlib.pyplot as plt
import math
import cmath
import scipy
import scipy.integrate as intgl
from scipy.integrate import simps

import copy
import time
################################################################################
#Global Variables and Misc

#Number of steps per unit time
resolution = 10000
#Time step
dt = 1/resolution

plt.rcParams.update({'font.size': 16})
################################################################################
#Random Variable Sampling
def rv_u():
    return(np.random.uniform(0,1))

def rv_sn():
    return(np.random.normal(0,1))

def rv_exp(rate):
    #rate = "lambda" so the mean is 1/rate
    if not rate > 0:
        print('Exponential random variable failed to generate!')
        forceTerminatePython = 1/0
    return(-math.log(rv_u())/rate)

def rv_ig(a,b):
    if not (a > 0 and b > 0):
        print('Inverse Gaussian random variable failed to generate!')
        forceTerminatePython = 1/0
    v = rv_sn()
    y = v**2
    x = (a/b) + y/(2*(b**2)) - np.sqrt((4*a*b*y)+(y**2))/(2*(b**2))
    u = rv_u()
    if u > a/(a + x*b):
        out = (a**2)/((b**2)*x)
    else:
        out = x
    return(out)
################################################################################
def addPosJumps(graph):
    out = np.zeros([len(graph)])
    for j in range(0,len(graph)-1):
        if graph[j+1] - graph[j] > 0:
            out[j] = np.nan
        else:
            out[j] = graph[j]
    out[len(graph)-1] = graph[len(graph)-1]
    return(out)

def addNegJumps(graph):
    out = np.zeros([len(graph)])
    for j in range(0,len(graph)-1):
        if graph[j+1] - graph[j] < 0:
            out[j] = np.nan
        else:
            out[j] = graph[j]
    out[len(graph)-1] = graph[len(graph)-1]
    return(out)

def addJumps(graph):
    out = np.zeros([len(graph)])
    for j in range(0,len(graph)-1):
        if abs(graph[j+1] - graph[j]) > 0.1:
            out[j] = np.nan
        else:
            out[j] = graph[j]
    out[len(graph)-1] = graph[len(graph)-1]
    return(out)

def intOU(ou,totalTime):
    N = len(ou)
    axis = np.linspace(0.0, totalTime, N)
    out = np.zeros([N])
    integrand = np.zeros([N])
    for j in range(0,N):
        integrand[j] = ou[j]
        out[j] = simps(integrand, axis)
    return(out)
```

Figure C.6: Python code for simulating stock prices.

```python
def sim_drift(totalTime,slope):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(0,len_out):
        out[j] = slope*(j*dt)
    return(out)

def sim_PoissonProcess(totalTime,rate):
    if not rate > 0:
        print('Warning: Poisson process!')
        forceTerminatePython = 1/0
    arrivalTime = 0.0
    arrivalTimes = []
    while arrivalTime < totalTime:
        arrivalTimes.append(arrivalTime)
        arrivalTime += rv_exp(rate)
    arrivalTimes.append(totalTime)
    #print(arrivalTimes)

    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(0,len_out):
        k = 0
        while not(j >= float(arrivalTimes[k])*resolution and \
                  j < float(arrivalTimes[k+1])*resolution):
            k += 1
        out[j] = k
    return(out)

def fastsim_PoissonProcess(totalTime,rate):
    if not rate > 0:
        print('Warning: Poisson process!')
        forceTerminatePython = 1/0
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + np.random.poisson(rate*dt)
    return(out)

def sim_stdBM(totalTime):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + math.sqrt(dt)*rv_sn()
    return(out)

def sim_subIG(totalTime,a,b):
    len_out = int(totalTime)
    out = np.zeros([len_out])
    for j in range(1,len_out):
        out[j] = out[j-1] + rv_ig(a*dt,b)
    return(out)

def sim_NIG(totalTime,a,b,d):
    len_out = int(totalTime)
    out = np.zeros([len_out])
    ig = sim_subIG(totalTime,1,d*math.sqrt((a**2)-(b**2)))
    for j in range(1,len_out):
        out[j] = out[j-1] + b*(d**2)*(ig[j]-ig[j-1]) +\
                 d*math.sqrt(ig[j]-ig[j-1])*rv_sn()
    return(out)

def sim_BDLP(totalTime,a,b):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    pp = fastsim_PoissonProcess(totalTime,a)
    for j in range(1,len_out):
        diff = 0
        for k in range(0, math.floor(pp[j]-pp[j-1])):
            diff += rv_exp(b)
        out[j] = out[j-1] + diff
    return(out)

def sim_OU_G(totalTime,rate,a,b,initial):
    len_out = totalTime*resolution
    out = np.zeros([len_out])
    out[0] = initial
    bdlp = sim_BDLP(totalTime,a*rate,b)
    for j in range(1,len_out):
        out[j] = out[j-1]*(1-rate*dt) + bdlp[j] - bdlp[j-1]
    return(out)
```

```
###############################################################################
#Stocks

def psi(a,b,d):
    if a>0 and d>0 and b>-a and b<a:
        out = -d*( math.sqrt(a**2 - (b+1)**2) - math.sqrt(a**2-b**2) )
    else:
        print('Invalid NIG parameters!')
    return(out)

def varphiModified(rate,a,b,y0,T,x):
    if rate>0 and a>0 and b>0 and y0>0 and (T>0 or T==0):
        out = math.exp( (x*y0*(1-math.exp(-rate*T))/rate) +\
                          (a/(x-rate*b))*\
                           ( b*math.log(b*rate/(b*rate-x*(1-math.exp(-rate*T)))) -\
                                             x*T ) )
    else:
        print('Invalid Stochastic Volatility parameters')
    return(out)

def phi(L,A,B,y0,a,b,d,T):
    return(varphiModified(L,A,B,y0,T,psi(a,b,d)))

def getCoefficient(L,A,B,y0,a,b,d,S0,r,length):
    out = np.zeros([length])
    for j in range(0,length):
        out[j] = S0*math.exp(r*j*dt)/phi(L,A,B,y0,a,b,d,j*dt)
    return(out)

def plotStocks():
    plt.rcParams.update({'font.size': 20})
    T = 1

    #OU parameters
    L = 1.19
    A = 0.703
    B = 1.4282
    y0 = 1
    #NIG parameters
    a = 38.131
    b = -21.6592
    d = 0.6984

    ou = sim_OU_G(T,L,A,B,y0)
    axis = np.linspace(0.0, T, len(ou))
    #plt.plot(axis, ou, 'xkcd:red')

    iou = intOU(ou,T)
    plt.subplot(2,2,1)
    plt.plot(axis, iou, 'xkcd:red')
    plt.xlabel('Integrated Gamma-OU Process')

    iouTrunc = np.zeros([len(iou)])
    index = np.zeros([len(iou)])
    for j in range(0,len(iou)):
        index[j] = int(math.floor(iou[j]*resolution))
        iouTrunc = index[j]/resolution

    YT = index[len(iou)-1]
    lp = sim_NIG(YT,a,b,d)
    plt.subplot(2,2,4)
    plt.plot(np.linspace(0.0, YT*dt, len(lp)), lp, 'xkcd:red')
    plt.xlabel('Pure Levy Process')

    X = np.zeros([len(iou)])
    for j in range(1,len(iou)):
        X[j] = lp[int(index[j]-1)]
    plt.subplot(2,2,2)
    plt.plot(axis, X, 'xkcd:red')
    plt.xlabel('Time-changed Levy Process')

    r = 0.007
    S0 = 1124.27
    length = int(len(X))
    stocks = np.zeros([length])
    coeff = getCoefficient(L,A,B,y0,a,b,d,S0,r,length)

    for j in range(0,length):
        stocks[j] = coeff[j]*math.exp(X[j])
    plt.subplot(2,2,3)
    plt.plot(axis, stocks, 'xkcd:red')
    plt.xlabel('Underlying Asset Price')
    plt.ylabel('USD')

    plt.show(block=False)
    return()

plotStocks()
```

Bibliography

[1] Asmussen S. and Rosiński J. (2001) *Approximations of Small Jumps of Lévy Processes with a View Towards Simulation* (Applied Probability Trust).

[2] Barndorff-Nielsen O. (1998) *Processes of normal inverse Gaussian type* (Finance and Stochastics, Springer-Verlag).

[3] Barndorff-Nielsen O. (2001) *Superposition of Ornstein-Uhlenbeck Type Processes* (Theory of Probability and Its Applications).

[4] Barndorff-Nielsen O. and Shephard N. – (2001) *Non-Gaussian Ornstein-Uhlenbeck-based Models and Some of Their Uses in Financial Economics* (Journal of the Royal Statistical Society: Series B (Statistical Methodology)).

[5] Baxter M. and Rennie A. (1996) *Financial Calculus : An Introduction to Derivative Pricing* (Cambridge University Press).

[6] Bee M. and Benedetti R. (2010) *Simulation of the Multivariate Generalized Hyperbolic Distribution using Adaptive Importance Sampling.*

[7] Bertoin J. (2009) *Lévy Processes* (Cambridge University Press).

[8] Boyd S. and Vandenberghe L. (2004) *Convex Optimisation* (Cambridge University Press).

[9] Carr P., Geman H., Madan D., Yor M. (2002) *The Fine Structure of Asset Returns: An Empirical Investigation* (Journal of Business).

[10] Carr P., Geman H., Madan D., Yor M. (2003) *Stochastic Volatility for Lévy Processes* (Wiley Online Library).

[11] Carr P. and Madan D. (1999) *Option Valuation Using the Fast Fourier Transform* (Journal of Computational Finance).

[12] Chou J. and Lin J. (2006) *Some Properties of CIR Processes* (Taylor & Francis Group).

[13] Cont R. and Tankov P. (2004). *Financial Modelling With Jump Processes* (Chapman & Hall/CRC Financial Mathematics Series).

[14] Delbaen F. and Schachermayer W. (1994) *A General Version of The Fundamental Theorem of Asset Pricing* (Mathematlsche Annalen).

[15] Devroye L. (2002) *On the Computer Generation of Random Variables with a given Characteristic Function* (Computers and Mathematics with Applications).

[16] Dewynne J., Howison S., Wilmott P. (1995) *The Mathematics of Financial Derivatives, A Student Introduction* (Cambridge University Press).

[17] Duan J-C. (1995) *The GARCH Option Pricing Model* (Mathematical Finance).

[18] Eberlein E. and Kallsen J. (2019) *Mathematical Finance* (Springer Finance).

[19] Feng Z. (2018) *Stock-Price Modeling by The Geometric Fractional Brownian Motion: A View towards the Chinese Financial Market.*

[20] Fouque J.P., Papanicolaou G., Sircar R. (2000) *Derivatives in Financial Markets with Stochastic Volatility* (Cambridge University Press).

[21] Glasserman P. (2004) *Monte Carlo Methods in Financial Engineering* — (Springer).

[22] Grigoletto M. and Provasi C. (2009) *Simulation and Estimation of the Meixner Distribution* (Communication in Statistics- Simulation and Computation).

[23] Hurst S. (1995) *The Characteristic Function of the Student T Distribution* (Centre for Mathematics and its Applications, School of Mathematical Sciences, ANU).

[24] Johnson L. and Riess R. (1982) *Numerical Analysis* (Second Ed. Addison-Wesley).

[25] Jurek Z. (1983) *An Integral Representation for Self-Decomposable Banach Space Valued Random Variables* (Probability Theory and Related Fields).

[26] Mazzola E. and Muliere P. (2011) *Reviewing Alternative Characterizations of Meixner Process* (Probability Surveys).

[27] Nocedal J. and Wright S. (2006) *Numerical Optimization* (Second Ed. Springer).

[28] Podgórski K. and Wallin J. (2016) *Convolution-invariant Subclasses of Generalized Hyperbolic Distributions* (Communications in Statistics - Theory and Methods).

[29] Raible S. (2000). *Lévy Processes in Finance: Theory, Numerics, and Empirical Facts* (PhD thesis, Freiburg).

[30] Revuz D. and Yor M. (1999). *Continuous Martingales and Brownian Motion* (3rd Ed. Springer).

[31] Ripley B. (1987). *Stochastic Simulation* (Wiley Series in Probability and Mathematical Statistics).

[32] Sato K. (2005) *Lévy Processes and Infinitely Divisible Distributions* (Revised Ed. Cambridge University Press).

[33] Schoutens W. (2003). *Lévy Processes in Finance* (Wiley).

[34] Taufer E. and Leonenko N. —— (2009). *Simulation of Lévy-driven Ornstein–Uhlenbeck Processes with given Marginal Distribution* (Computational Statistics and Data Analysis).

[35] Winkel M. (2010). *MS3b/MScMCF Lévy Processes and Finance* (Lecture Notes).

[36] `https://finance.yahoo.com/quote/%5EGSPC/history?period1=0` &`period 2=1585008000`&`interval=1d`&`filter=history`&`frequency=1d` (Accessed: 24/03/2020).

[37] `https://www.bbc.co.uk/news/business-52350082` (Accessed: 21/04/2020).