POLITECNICO DI MILANO
Ingegneria Industriale e dell'Informazione
Laurea Magistrale in Ingegneria Matematica

Applicazioni del metodo della quantizzazione alla valutazione di derivati

Relatore:
Prof. Daniele Marazzina

Tesi di Laurea di:
Francesco Perinti
Matr. 852577

Anno Accademico 2017-2018

Abstract

L'obiettivo di questo lavoro è quello di illustrare il metodo della quantizzazione e le sue applicazioni al pricing di prodotti derivati. Seguendo la letteratura già esistente, ne daremo le principali nozioni teoriche e ne mostreremo una diretta applicazione al pricing dell'opzione call plain vanilla, grazie all'uso della trasformata di Fourier; vedremo in seguito come è possibile estendere questo metodo per prezzare opzioni path-dependent, in particolare lookback e barriere, grazie alle equazioni di Spitzer. Infine mostreremo i risultati ottenuti utilizzando diversi modelli di Lévy, particolarmente congeniali a questo metodo grazie al teorema di Lévy–Khintchine, che dà una forma esplicita alla loro funzione caratteristica.

Abstract

The aim of this work is to illustrate the quantization method and its applications to derivatives pricing. Following the existing literature, we will first give the main theoretical notions and present a direct application to the plain vanilla call, by using the Fourier transform; then we will show how it is possible to extend this method to price also path-dependent options, such as lookback and knock-out, thanks to the Spitzer's equations. Finally, we will show the results obtained using different Lévy models, which are particularly suitable to this method thanks to the Lévy-Khintchine theorem, which gives an explicit form to the characteristic function.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La quantizzazione nasce negli anni '50 come metodo di discretizzazione e compressione per la trasmissione di segnali, ed è stata poi applicata in diversi ambiti come la ricerca operativa, la teoria dell'informazione e il riconoscimento vocale. A partire dagli anni '70 è stata oggetto di studio anche come metodo numerico utilizzabile in probabilità, per esempio da Pierce in [12], e conseguentemente anche in finanza matematica da Pagès e Sagna in [10] e poi da Fiorin et al. in [2], che hanno sfruttato la trasformata di Fourier insieme alla quantizzazione come metodo di pricing di derivati plain vanilla. Il contributo di questa tesi è quello di estendere il metodo proposto da Fiorint et al. [2] al pricing di derivati path-dependent, in particolare a lookback e barriere knock-out, sfruttando le equazioni di Spitzer, che permettono di trovare la legge congiunta del sottostante a scadenza e degli estremi da esso assunti durante la vita dell'opzione. Per farlo utilizzeremo alcuni processi esponenziali di Lévy, la cui funzione caratteristica è nota esplicitamente grazie al teorema di Lévy-Khintchine; in particolare tratteremo i seguenti modelli: Gaussiano, Merton, Kou, il Normal Inverse Gaussian e il Variance Gamma. Confronteremo quindi i risultati con quelli del metodo Monte Carlo: a nostro avviso la quantizzazione risulta troppo costosa computazionalmente e a volte troppo imprecisa nel caso di opzioni path-dependent, come vedremo nell'ultimo capitolo.
Il lavoro è suddiviso in questo modo: nel Capitolo 1 esporremo i principali risultati teorici; nel Capitolo 2 mostreremo il metodo di quantizzazione che sfrutta la trasformata di Fourier; nel Capitolo 3 vedremo come è possibile estendere il metodo ai derivati path-dependent manipolando la funzione caratteristica; infine nel Capitolo 4 introdurremo brevemente i modelli di Lévy ed esporremo i risultati ottenuti.

Capitolo 1

La quantizzazione

In questo capitolo viene introdotto il metodo della quantizzazione, per cui si fa principalmente riferimento a [2].

1.1 Introduzione

Si immagini di voler calcolare il valore atteso di una variabile aleatoria X: se questo non è conosciuto analiticamente, si dovranno usare dei metodi numerici per approssimarlo. La quantizzazione consiste nel ridurre i possibili esiti di X ad una griglia finita Γ, in modo da poter approssimare il valore atteso con una sommatoria finita. Il fulcro di questa procedura è quindi la scelta della discretizzazione ottima Γ, che non sarà molto fitta, ma composta da pochi punti scelti ad hoc, ottimizzati per minimizzare l'errore commesso.

1.2 Framework teorico

Sia $(\Omega, \mathcal{F}, \mathbb{P})$ uno spazio di probabilità e X una variabile aleatoria in $\mathbb{R}^d$ con legge μ su Ω. Sia $N \in \mathbb{N}$ il numero di elementi della griglia $\Gamma^N = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$, detta anche quantizzatore. Per alleggerire la notazione spesso si ometterà la N di Γ^N.

Definizione 1.1. Considerata una norma $|\cdot|$ su $\mathbb{R}^d$, si definisce *quantizzatore* una funzione Borel-misurabile $q_\Gamma : \mathbb{R}^d \to \Gamma$ tale che

$$q_\Gamma(\xi) = \mathrm{Proj}_\Gamma(\xi) = \arg\min_{x \in \Gamma} |x - \xi| \,. \tag{1.1}$$

Definizione 1.2. La griglia Γ individua una partizione misurabile $\big(C_i(\Gamma)\big)_{1\le i\le N}$ di $\mathbb{R}^d$ chiamata *partizione di Voronoi*, tale che

$$q_\Gamma(\xi) = \sum_{i=1}^{N} x_i \mathbb{1}_{C_i}(\xi)\,, \tag{1.2}$$

dove gli insiemi $C_i(\Gamma)$ vengono chiamati *celle di Voronoi*.

La partizione dipende quindi dalla norma scelta. In particolare, rispetto alla metrica euclidea, le celle C_i sono convesse e vale

$$\mathring{C}_i(\Gamma) = \Big\{\xi \in \mathbb{R}^d : |\xi - x_i| < \min_{j\neq i} |\xi - x_j|\Big\}, \tag{1.3}$$

$$\overline{C}_i(\Gamma) = \Big\{\xi \in \mathbb{R}^d : |\xi - x_i| = \min_{1\le j\le N} |\xi - x_j|\Big\}. \tag{1.4}$$

Facciamo presente che non è necessario definire rigorosamente in quali celle includere i bordi visto che tratteremo solo variabili aleatorie che assumono valori sul bordo con probabilità nulla. Inoltre spesso verrà omesso l'argomento Γ di $C_j(\Gamma)$ per alleggerire la notazione. Il problema principale ora è trovare la griglia ottima

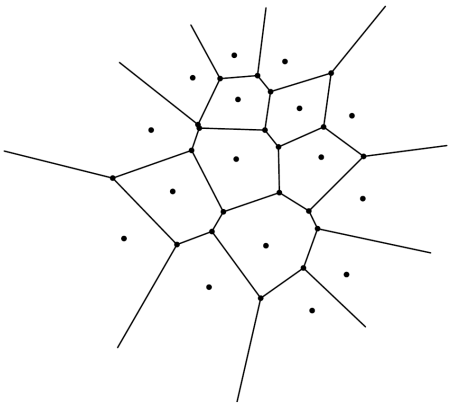


Figura 1.1: Partizione di Voronoi in $\mathbb{R}^2$.

di quantizzazione. Per farlo, si calcola l'errore commesso dalla discretizzazione e si minimizza rispetto alle possibili griglie. Si noti che il numero di punti N è fissato a priori.

Definizione 1.3. Sia $p \in [1, +\infty)$. Data la variabile aleatoria X, si definisce *funzione di distorsione in norma L^p* la funzione $D_p : (\mathbb{R}^d)^N \to \mathbb{R}$ come

$$D_p(\Gamma) = \mathbb{E}\big[|X - q_\Gamma(X)|^p\big]. \tag{1.5}$$

Osservazione 1. È utile osservare che

$$D_p(\Gamma) = \mathbb{E}\big[|X - q_\Gamma(X)|^p\big] = \int_{\mathbb{R}^d} |\xi - \arg\min_{1\le i\le N} |\xi - x_i||^p \mu(d\xi)$$
$$= \int_{\mathbb{R}^d} \min_{1\le i\le N} |\xi - x_i|^p \mu(d\xi)\,.$$

Il ruolo del parametro p viene discusso nell'appendice A. Si vuole quindi ora trovare la griglia ottima che minimizzi la funzione di distorsione, ammesso che esista e sia unica.

Teorema 1.2.1. *Se $X \in L^p(\mathbb{P})$, allora D_p ammette almeno un minimo $\hat{\Gamma}$. Inoltre* $\lim_{N\to\infty} D_p(\Gamma^N) = 0$.[1]

Osservazione 2. In generale, se $d \ge 2$ non si ha l'unicità, per esempio a causa dell'invarianza di μ rispetto a varie trasformazioni. Quando invece $d = 1$, se μ è assolutamente continua rispetto ad una densità log-concava, allora esiste ed è unica una griglia ottima di quantizzazione[2].

Data l'esistenza, ci si chiede ora come ottenere un quantizzatore ottimo. Si cerca quindi di differenziare la funzione di distorsione.

Teorema 1.2.2. *Sia $|\cdot|$ la norma euclidea e $p \in (1, +\infty)$, allora la funzione di distorsione D_p è differenziabile e*

$$\nabla D_p(x_1, \dots, x_N) = p\bigg(\int_{C_i(\Gamma)} \frac{x_i - \xi}{|x_i - \xi|} |x_i - \xi|^{p-1} \mu(d\xi)\bigg)_{1\le i\le N} \tag{1.6}$$
$$= p\Big(\mathbb{E}\Big[\mathbb{1}_{X\in C_i(\Gamma)} \frac{x_i - X}{|x_i - X|} |x_i - X|^{p-1}\Big]\Big)_{1\le i\le N}. \tag{1.7}$$

Si dà ora la definizione di quantizzatore stazionario.

Definizione 1.4. Sia $p \in (1, +\infty)$ e X un vettore aleatorio in L^{p-1} con distribuzione μ. Un quantizzatore $\Gamma = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$ si dice *p-stazionario* se valgono le seguenti proprietà:

i) $\mu\Big(\bigcup_{i=1}^N \partial C_i(\Gamma)\Big) = 0\,,$

ii) $\mathbb{E}\Big[\mathbb{1}_{X\in C_i(\Gamma)} \frac{x_i - X}{|x_i - X|} |x_i - X|^{p-1}\Big] = 0 \quad \forall\, i = 1, \dots, N\,.$

La prima richiesta significa che la frontiera della partizione di Voronoi sia trascurabile rispetto a μ, ovvero che quasi certamente X non assume valori sulla frontiera, mentre la seconda implica la stazionarietà della griglia.

[1] Per la dimostrazione si veda [10], Prop 1.1 .
[2] Per risultati più precisi si veda [10] Sect. 2

1.3 Stime a priori sull'errore

In generale, quantizzatori ottimi sono stazionari, ma il viceversa non è necessariamente vero, visto che gli ultimi sono solo punti che annullano il gradiente della funzione di distorsione. Vengono quindi presentati ora due risultati circa la stima degli errori a priori per variabili aleatorie X reali positive, dopo aver definito l'errore commesso dalla quantizzazione.

Definizione 1.5. Siano $p \in [1, +\infty)$, X un vettore aleatorio e Γ un suo quantizzatore. Si definisce *errore di quantizzazione in norma* L^p la quantità

$$e_p(X, \Gamma) = ||X - \mathrm{Proj}_\Gamma(X)||_{L^p}\,, \tag{1.8}$$

dove abbiamo usato la stessa notazione della definizione (1.1).

Si dà ora il primo risultato sulla stima dell'errore, dovuto a Pierce [12]. Per entrambi i risultati supporremo che X sia una variabile aleatoria reale, positiva e che ammetta densità.

Lemma 1.3.1 (Lemma di Pierce)**.** *Sia $p \in [1, +\infty)$. Se esiste $\delta > 0$ tale che $\mathbb{E}\big[|X|^{p+\delta}\big] < +\infty$, allora*

$$e_p(X, \Gamma) \leq \frac{C_1 \mathbb{E}\big[|X|^{p+\delta}\big] + C_2}{N^p}\,, \tag{1.9}$$

per qualche costante $C_1, C_2 > 0$, dipendenti da δ e p, ma non da X, N e Γ.

Si noti che la qualità dell'errore nel lemma di Pierce dipende dal livello di regolarità di X. Nelle applicazioni finanziarie, X è il processo che modellizza il sottostante: nel caso di modelli a volatilità stocastica, fenomeni come la non limitatezza dei momenti possono rendere inutile il risultato, visto che le costanti C_1, C_2 dipendono da δ. Per queste è utile presentare un altro risultato, come in [6] le quali costanti non dipendono dall'integrabilità di X.

Teorema 1.3.2. *Sia $p \in [1, +\infty)$. Se X ha il p-esimo momento finito e la sua densità f ha andamento polinomiale in intorni di 0 e $+\infty$, allora*

$$\lim_{N \to +\infty} N^p e_p(X, \Gamma) \leq \frac{||f||_{\frac{1}{p+1}}}{2^p (p+1)}\,. \tag{1.10}$$

Dimostrazione. La dimostrazione segue le argomentazioni di [6] ed è divisa in vari step.

Step 0: Osserviamo che avendo per ipotesi

$$\begin{aligned} f(z) &\sim z^{-\alpha} && \text{se} \quad z \to +\infty\,, \\ f(z) &\sim z^{\gamma} && \text{se} \quad z \to 0\,, \end{aligned}$$

e sapendo che X ha momento p-esimo finito, segue che

$$\begin{aligned} &\alpha > p+1\,, \\ &\gamma > -(1+p)\,. \end{aligned}$$

Queste condizioni insieme garantiscono che $\|f\|_{\frac{1}{p+1}} < \infty$.

Step 1: Scriviamo ora la funzione di distorsione come segue:

$$\begin{aligned} D(x_1,\dots,x_N) &= \sum_{i=1}^{N} \int_{C_i} |z-x_i|^p f(z)dz \\ &= \int_0^{x_1} (x_1-z)^p f(z)dz + \int_{x_1}^{x_1^+} (z-x_1)^p f(z)dz + \\ &+ \int_{x_1^+}^{x_2} (x_2-z)^p f(z)dz + \int_{x_2}^{x_2^+} (z-x_2)^p f(z)dz + \ldots + \\ &+ \int_{x_{N-1}^+}^{x_N} (x_N-z)^p f(z)dx + \int_{x_N}^{+\infty} (z-x_N)^p f(z)dz\,, \end{aligned}$$

e definiamo

$$\begin{aligned} \xi_1 &\coloneqq \operatorname*{arg\,max}_{z\in[x_1,x_1^+]} f(z)\,, \\ \xi_i &\coloneqq \operatorname*{arg\,max}_{z\in C_i} f(z)\,, \quad i=2,\dots,N-1\,, \quad \text{e} \\ \xi_N &\coloneqq \operatorname*{arg\,max}_{z\in[x_{N-1}^+,x_N]} f(z)\,, \end{aligned}$$

allora

$$\begin{aligned} D(x_1,\dots,x_N) &\le \int_0^{x_1} (x_1-z)^p f(z)dx + f(\xi_1)\int_{x_1}^{x_1^+} (z-x_1)^p dz + \ldots + \\ &+ f(\xi_N)\int_{x_{N-1}^+}^{x_N} (x_N-z)^p dz + \int_{x_N}^{+\infty} (z-x_N)^p f(z)dz \\ &= \int_0^{x_1} (x_1-z)^p f(z)dz + \int_{x_N}^{+\infty} (z-x_N)^p f(z)dz \\ &+ \sum_{i=1}^{N-1} \frac{f(\xi_i)+f(\xi_{i+1})}{p+1}\Big(\frac{x_{i+1}-x_i}{2}\Big)^{p+1}\,. \end{aligned}$$

Step 2: Dato lo Step 0, è possibile definire una griglia di quantizzazione $\bar\Gamma = (\bar x_1, \dots, \bar x_N)$ tale che

$$\frac{\displaystyle\int_0^{\bar x_i} f^{\frac{1}{p+1}}(z)dz}{\|f\|_{\frac{1}{p+1}}^{\frac{1}{p+1}}} = \frac{\displaystyle\int_0^{\bar x_i} f^{\frac{1}{p+1}}(z)dz}{\displaystyle\int_0^{+\infty} f^{\frac{1}{p+1}}(z)dz} = \frac{2i-1}{2N}\,, \quad \text{per} i = 1, \dots, N\,.$$

Allora, per $i = 1, \dots, N-1$ si ha che

$$\int_{\bar x_i}^{\bar x_{i+1}} f^{\frac{1}{p+1}}(z)dz = \frac{\|f\|_{\frac{1}{p+1}}^{\frac{1}{p+1}}}{N}\,.$$

Essendo f continua, grazie al teorema della media sappiamo che per ogni $i = 1, \dots, N-1$ esiste $\zeta_i \in [\bar x_i, \bar x_{i+1}]$ tale che

$$(\bar x_{i+1} - \bar x_i)^p = \frac{\|f\|_{\frac{1}{p+1}}^{\frac{p}{p+1}}}{f^{\frac{p}{p+1}}(\zeta_i)N^p}\,.$$

Step 3: Diamo ora un limite all'errore, e poi studieremo il comportamento per $N \to \infty$. Definendo

$$E(x_1, \dots, x_N) \coloneqq \sum_{i=1}^{N-1} \frac{f(\xi_i) + f(\xi_{i+1})}{p+1}\Big(\frac{x_{i+1} - x_i}{2}\Big)^{p+1}\,,$$

possiamo scrivere

$$D(x_1, \dots, x_N) \le E(x_1, \dots, x_N) + \int_0^{x_1} (x_1 - z)^p f(z)dz + \int_{x_N}^{+\infty} (z - x_N)^p f(z)dz\,.$$

Usiamo ora la griglia $(\bar x_1, \dots, \bar x_N)$ per definire, come nello Step 1,

$$\begin{aligned}
\bar\xi_1 &\coloneqq \operatorname*{arg\,max}_{z \in [\bar x_1, \bar x_1^+]} f(z)\,,\\
\bar\xi_i &\coloneqq \operatorname*{arg\,max}_{z \in \bar C_i} f(z)\,, \quad i = 2, \dots, N-1 \quad \text{e}\\
\bar\xi_N &\coloneqq \operatorname*{arg\,max}_{z \in [\bar x_{N-1}^+, \bar x_N]} f(z)\,.
\end{aligned}$$

Allora si ha che

$$\begin{aligned}
E(\bar x_1, \dots, \bar x_N) &= \sum_{i=1}^{N-1} \frac{f(\bar\xi_i) + f(\bar\xi_{i+1})}{2^{p+1}(p+1)}(\bar x_{i+1} - \bar x_i)^p(\bar x_{i+1} - \bar x_i)\\
&= \frac{\|f\|_{\frac{1}{p+1}}^{\frac{p}{p+1}}}{2^{p+1}(p+1)N^p} \sum_{i=1}^{N-1} \frac{f(\bar\xi_i) + f(\bar\xi_{i+1})}{f^{\frac{p}{p+1}}(\zeta_i)}(\bar x_{i+1} - \bar x_i)\,.
\end{aligned}$$

Dalla definizione dell'errore di quantizzazione $e_p(X,\Gamma)$ segue che

$$e_p(X,\Gamma) \le D(\bar{\Gamma}) \le E(\bar{\Gamma}) + \int_0^{\bar{x}_1} (\bar{x}_1 - z)^p f(z)dz + \int_{\bar{x}_N}^{+\infty} (z - \bar{x}_N)^p f(z)dz\,. \tag{1.11}$$

Step 4: Per $N \to \infty$ abbiamo $\bar{x}_1 \to 0\,, \bar{x}_N \to +\infty$ e

$$\sum_{i=1}^{N-1} \frac{f(\bar{\xi}_i) + f(\bar{\xi}_{i+1})}{f^{\frac{p}{p+1}}(\zeta_i)} (\bar{x}_{i+1} - \bar{x}_i) \to 2\int_0^{+\infty} f^{\frac{1}{p+1}}(z)dz = 2\,\|f\|_{\frac{1}{p+1}}^{\frac{1}{p+1}}\,.$$

Mostriamo ora che gli integrali in (1.11) tendono a zero più velocemente di N^{-p} per $N \to \infty$. Dallo Step 2 si ha che

$$\int_{\bar{x}_N}^{+\infty} f^{\frac{1}{p+1}}(z)dz = \frac{\|f\|_{\frac{1}{p+1}}^{\frac{1}{p+1}}}{2N}\,,$$

quindi

$$\lim_{N\to+\infty} N^p \int_{\bar{x}_N}^{+\infty} (z - \bar{x}_N)^p f(z)dz = \frac{2^p}{\|f\|_{\frac{1}{p+1}}^{\frac{p}{p+1}}} \lim_{N\to+\infty} \frac{\displaystyle\int_{\bar{x}_N}^{+\infty} (z - \bar{x}_N)^p f(z)dz}{\left(\displaystyle\int_{\bar{x}_N}^{+\infty} f^{\frac{1}{p+1}}(z)dz\right)^p}\,.$$

Ricordando le conclusioni dello Step 0 abbiamo

$$\lim_{y\to+\infty} \frac{\displaystyle\int_y^{+\infty} (z-y)^p f(z)dz}{\left(\displaystyle\int_y^{+\infty} f^{\frac{1}{p+1}}(z)dz\right)^p} = \lim_{y\to+\infty} \frac{\displaystyle\int_y^{+\infty} (z-y)^p z^{-\alpha}dz}{\left(\frac{y^{-\frac{\alpha}{p+1}+1}}{\frac{\alpha}{p+1}-1}\right)^p}\,.$$

Operando all'integrale al numeratore la sostituzione $t = \frac{y}{z}$ otteniamo

$$\int_y^{+\infty} (z-y)^p z^{-\alpha}dz = y^{p+1-\alpha} \int_0^1 (1-t)^p t^{\alpha-2-p}dt\,,$$

da cui segue che

$$\lim_{N\to+\infty} N^p \int_{\bar{x}_N}^{+\infty} (z - \bar{x}_N)^p f(z)dz = 0\,.$$

Lo stesso ragionamento si può applicare al secondo integrale in (1.11), per mostrare che anch'esso tende a zero più velocemente di N^{-p}.

Step 5: Nello Step 4 abbiamo mostrato che, per $N \to +\infty$ vale

$$e_p(X,\Gamma) \leq E(\bar{\Gamma}) + o\left(\frac{1}{N^p}\right) \leq \frac{\|f\|_{\frac{p}{p+1}}^{\frac{p}{p+1}}}{2^{p+1}(p+1)} \frac{2\,\|f\|_{\frac{1}{p+1}}^{\frac{1}{p+1}}}{N^p} + o\left(\frac{1}{N^p}\right)$$
$$\sim \frac{1}{N^p} \frac{\|f\|_{\frac{1}{p+1}}}{2^p(p+1)},$$

da cui segue la tesi.

□

Osservazione 3. Le ipotesi del Teorema 1.3.2 sono più forti di quelle del Lemma di Pierce 1.3.1; infatti l'esistenza del p-esimo momento insieme all'andamento polinomiale all'infinito implicano l'esistenza del momento (p+δ)-esimo, per un qualche δ sufficientemente piccolo. Tuttavia, è importante notare che questo risultato dipende solo dalla distribuzione di X e non da δ.

Capitolo 2

Quantizzazione e trasformata di Fourier

2.1 Introduzione

Nel caso di pricing di prodotti derivati, la variabile aleatoria X rappresenterà il sottostante. Introduciamo uno spazio di probabilità filtrato $\big(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P}\big)$ che soddisfi le condizioni usuali[1] e un processo $(S_t)_{t\in[0,T]}$, dove T sarà la maturity dell'opzione; nel caso di derivati plain vanilla si avrebbe quindi $d = 1$ e $X = S_T$. Sia ora

$$\varphi(u) \coloneqq \varphi_T(u) \coloneqq \mathbb{E}\big[e^{iu\log(S_T)}|\mathcal{F}_0\big], \quad u \in \mathbb{R},$$

la funzione caratteristica della variabile aleatoria $\log(S_T)$, dove per alleggerire la notazione si è omesso il pedice T. Vedremo come è possibile ricavare un quantizzatore ottimo tramite la funzione caratteristica. Ricordando inoltre che grazie al teorema di inversione della trasformata di Fourier è possibile ricavare la densità, qualora essa esista, conoscere la funzione caratteristica è sufficiente per l'intera procedura. Ciò è particolarmente utile per i processi di Lévy, grazie al teorema di Lévy–Khintchine, che dà una forma esplicita per la funzione caratteristica, come vedremo nel Capitolo 4.1. Per i risultati citati in questo capitolo seguiremo [2], tranne dove specificato diversamente.
Riportiamo ora le principali relazioni che ci permetteranno di calcolare la densità di S_T data la funzione caratteristica φ di $\log(S_T)$, che seguono da una diretta

[1] Una filtrazione $(\mathcal{F}_t, \mathbb{P})$ soddisfa le cosiddette *condizioni usuali* se

i) $\mathcal{F}_0$ è completa rispetto a $\mathbb{P}$;

ii) per ogni $t \geq 0$ vale $\mathcal{F}_t = \bigcap_{\epsilon>0} \mathcal{F}_{t+\epsilon}$.

applicazione della trasformata inversa di Fourier, di cui riportiamo l'enunciato da [9].

Teorema 2.1.1 (Inversione della trasformata di Fourier)**.** *Sia X una variabile aleatoria reale con legge μ su $\mathbb{R}$ e ϕ la sua funzione caratteristica, definita come*

$$\phi(u) := \mathbb{E}\big[e^{iuX}\big] = \int_{\mathbb{R}} e^{iux}\mu(dx).$$

Allora, se $\int_{\mathbb{R}} |\phi(u)|du < +\infty$, si ha che μ ammette densità limitata g, tale che $\mu(dx) = g(x)dx$ e

$$g(x) = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}\phi(u)du.$$

Un'applicazione dell'inversione ci permette di enunciare, come in [2], la seguente proposizione.

Proposizione 2.1.2.

$$\mathbb{P}\big(S_T \in (z, z+dz]\big) = \left(\frac{1}{\pi}\frac{1}{z}\int_0^\infty \Re(e^{-iu\log(z)}\varphi(u))du\right)dz =: f(z)dz\,, \tag{2.1}$$

$$\mathbb{P}(S_T \le z) = \frac{1}{2} - \frac{1}{\pi}\int_0^\infty \Re\left(\frac{e^{-iu\log(z)}\varphi(u)}{iu}\right)du, \quad z \in (0, +\infty)\,, \tag{2.2}$$

dove $\Re(\xi)$ rappresenta la parte reale del numero complesso ξ.

Ora non resta altro che esplicitare una procedura per il calcolo di quantizzatori ottimi. Questo viene fatto con un algoritmo numerico à la Newton-Raphson: per ricavarlo si esprime la funzione di distorsione tramite la funzione caratteristica φ, si differenzia rispetto alle possibili griglie e si ricava la *Master Equation*, da risolvere per trovare la griglia ottima cercata.

2.2 La Master Equation

Nel caso monodimensionale la griglia $\Gamma = \{x_1, \dots, x_N\}$ non è altro che un insieme di punti ordinati, quindi le celle di Voronoi si riducono a semplici intervalli. Considerando che S_T è sempre positivo, si ha

$$C_j(\Gamma) = (x_j^-, x_j^+)\,; \tag{2.3}$$

$$x_j^- = \frac{1}{2}(x_{j-1} + x_j)\,, \quad j = 2, \dots, N\,; \qquad x_1^- = 0\,; \tag{2.4}$$

$$x_j^+ = \frac{1}{2}(x_j + x_{j+1})\,, \quad j = 1, \dots, N-1\,; \qquad x_N^+ = +\infty\,. \tag{2.5}$$

Ricordando l'espressione per la funzione di distorsione nel nostro caso specifico si ha che

$$D_p(\Gamma) = \mathbb{E}\big[|S_T - q_\Gamma(S_T)|^p\big] = \int_{\mathbb{R}} \min_{1\le j\le N} |s - x_j|^p \mu_{S_T}(ds)\,, \tag{2.6}$$

dove abbiamo indicato con μ_{S_T} la distribuzione di S_T. Sfruttando la natura monodimensionale del problema, considerando che q_Γ è costante sulle celle $C_j(\Gamma)$ e che ovviamente il minimo viene ottenuto in j quando $s \in C_j(\Gamma)$, si può scrivere

$$D_p(\Gamma) = D_p(x_1,\dots,x_N) = \int_{\mathbb{R}} \min_{1\le j\le N} |s - x_j|^p \mu_{S_T}(ds) \tag{2.7}$$

$$= \int_{\mathbb{R}} \sum_{j=1}^{N} |s - x_j|^p \mathbf{1}_{C_j}(s) \mu_{S_T}(ds) = \sum_{j=1}^{N} \int_{C_j} |s - x_j|^p \mu_{S_T}(ds)\,. \tag{2.8}$$

Assumendo che la densità di S_T esista assolutamente continua e concentrata su $(0,+\infty)$, è possibile differenziare la funzione di distorsione dentro l'integrale. Indicando con $\nabla_D(\Gamma)$ il gradiente di $D(\Gamma)$, si tratta quindi di risolvere il sistema N-dimensionale $\nabla_D(\Gamma) = 0$, le cui equazioni sono date dal seguente teorema. Per comodità spesso ometteremo il pedice D.

Teorema 2.2.1 (La Master Equation)**.** *Sia $p \in (1,+\infty)$. Allora $\Gamma = \{x_1,\dots,x_N\}$ è p-stazionaria per la funzione di distorsione (nel senso della Definizione 1.4) se per ogni $j = 0,\dots,N$ vale*

$$\int_0^{+\infty} \Re\left[\varphi(u) e^{-iu\log(x_j)} \left(\beta\left(\frac{x_j^-}{x_j}, -iu, p\right) - \beta\left(\frac{x_j}{x_j^+}, 1-p+iu, p\right)\right)\right] du = 0\,, \tag{2.9}$$

dove, posto $\mathbb{C}^+ := \{z \in \mathbb{C} : \Re[z] > 0\}$,

$$\beta : (0,1) \times \mathbb{C} \times \mathbb{C}^+ \to \mathbb{R}\,, \tag{2.10}$$

$$(x,a,b) \mapsto \beta(x,a,b) = \int_x^1 t^{a-1}(1-t)^{b-1} dt\,. \tag{2.11}$$

Dimostrazione. Si tratta di differenziare la funzione di distorsione espressa in (2.8) rispetto a x_j, per ogni j. Fissato quindi j, esplicitiamo i termini che dipendono da x_j: ricordando l'espressione (2.3) di C_j, è chiaro che gli addendi che ci interessano sono gli integrali su C_{j-1}, C_j, C_{j+1}.

$$D(\Gamma) = \dots \quad + \int_{C_{j-1}} |s-x_j|^p \mu_{S_T}(ds) + \int_{C_j} |s-x_j|^p \mu_{S_T}(ds) + \int_{C_{j+1}} |s-x_j|^p \mu_{S_T}(ds)\,.$$

Riscriviamo ora i vari termini, spezzando gli integrali ed esplicitando il valore assoluto:

$$\int_{C_{j-1}} |s-x_{j-1}|^p \mu_{S_T}(ds) = \int_{x_{j-1}^-}^{x_{j-1}} (x_{j-1}-s)^p \mu_{S_T}(ds) + \int_{x_{j-1}}^{x_j^-} (s-x_{j-1})^p \mu_{S_T}(ds)$$

$$\int_{C_j} |s-x_j|^p \mu_{S_T}(ds) = \int_{x_j^-}^{x_j} (x_j-s)^p \mu_{S_T}(ds) + \int_{x_j}^{x_j^+} (s-x_j)^p \mu_{S_T}(ds)$$

$$\int_{C_{j+1}} |s-x_{j+1}|^p \mu_{S_T}(ds) = \int_{x_j^+}^{x_{j+1}} (x_{j+1}-s)^p \mu_{S_T}(ds) + \int_{x_{j+1}}^{x_{j+1}^+} (s-x_{j+1})^p \mu_{S_T}(ds)\,.$$

Si può quindi semplificare ancora l'espressione per $D(\Gamma)$, esplicitando i termini dipendenti da x_j:

$$\begin{aligned} D(\Gamma) = \ldots \quad &+ \int_{x_{j-1}}^{x_j^-} (s-x_{j-1})^p \mu_{S_T}(ds) \qquad + \int_{x_j^-}^{x_j} (x_j-s)^p \mu_{S_T}(ds) + \\ &+ \int_{x_j}^{x_j^+} (s-x_j)^p \mu_{S_T}(ds) \qquad + \int_{x_j^+}^{x_{j+1}} (x_{j+1}-s)^p \mu_{S_T}(ds). \end{aligned}$$

Differenziamo ora dentro il segno d'integrale, essendo le funzioni integrande di classe C^1 sull'intervallo d'integrazione. Ricordando inoltre la Definizione 2.1 di f e la definizione (2.4) di $x_j^\pm$, si ottiene:

$$\begin{aligned} \frac{\partial D(\Gamma)}{\partial x_j} = \quad & \frac{1}{2}(x_j^- - x_{j-1})^p f(x_j^-) - \frac{1}{2}(x_j - x_j^-)^p f(x_j^-) + \int_{x_j^-}^{x_j} p(x_j-s)^{p-1} f(s) ds \\ & + \frac{1}{2}(x_j^+ - x_j)^p f(x_j^+) - \int_{x_j}^{x_j^+} p(s-x_j)^{p-1} f(s) ds - \frac{1}{2}(x_{j+1} - x_j^+)^p f(x_j^+) \\ = \quad & \frac{1}{2}\left(\frac{x_j - x_{j-1}}{2}\right)^p f(x_j^-) - \frac{1}{2}\left(\frac{x_j - x_{j-1}}{2}\right)^p f(x_j^-) + \int_{x_j^-}^{x_j} p(x_j-s)^{p-1} f(s) ds \\ & + \frac{1}{2}\left(\frac{x_{j+1} - x_j}{2}\right)^p f(x_j^-) - \int_{x_j}^{x_j^+} p(s-x_j)^{p-1} f(s) ds - \frac{1}{2}\left(\frac{x_{j+1} - x_j}{2}\right)^p f(x_j^-) \\ = \quad & \int_{x_j^-}^{x_j} p(x_j-s)^{p-1} \left(\frac{1}{\pi}\frac{1}{s}\int_0^\infty \Re(e^{-iu\log(s)}\varphi(u)) du\right) ds \\ & - \int_{x_j}^{x_j^+} p(s-x_j)^{p-1} \left(\frac{1}{\pi}\frac{1}{s}\int_0^\infty \Re(e^{-iu\log(s)}\varphi(u)) du\right) ds \\ = \quad & \frac{p}{\pi}\int_0^{+\infty} \Re\left[\varphi(u)\left(\int_{x_j^-}^{x_j} (x_j-s)^{p-1} s^{-1-iu} ds - \int_{x_j}^{x_j^+} (s-x_j)^{p-1} s^{-1-iu} ds\right)\right] du. \end{aligned}$$

Quasi giunti alla conclusione, studiamo separatamente i due integrali. Per il primo operiamo il cambio di variabile $t = s/x_j$ ottenendo

$$\begin{aligned}\int_{x_j^-}^{x_j} (x_j - s)^{p-1} s^{-1-iu} ds =& x_j^{p-1-iu} \int_{\frac{x_j^-}{x_j}}^{1} (1-t)^{p-1} t^{-1-iu} dt \\ =& x_j^{p-1-iu} \beta\left(\frac{x_j^-}{x_j}, -iu, p\right);\end{aligned}$$

mentre per il secondo usiamo $1/t = s/x_j$ per trovare

$$\begin{aligned}\int_{x_j}^{x_j^+} (s - x_j)^{p-1} s^{-1-iu} ds =& x_j^{p-1-iu} \int_1^{\frac{x_j}{x_j^+}} \left(\frac{1}{t} - 1\right)^{p-1} \left(\frac{1}{t}\right)^{-1-iu} \left(-\frac{1}{t^2}\right) dt \\ =& x_j^{p-1-iu} \beta\left(\frac{x_j}{x_j^+}, 1 - p + iu, p\right).\end{aligned}$$

Sommando le due espressioni otteniamo quindi

$$\frac{\partial D(\Gamma)}{\partial x_j} = \frac{p}{\pi} x_j^{p-1} \int_0^{+\infty} \Re\left[\varphi(u) e^{-iu\log(x_j)} \left(\beta\left(\frac{x_j^-}{x_j}, -iu, p\right) - \beta\left(\frac{x_j}{x_j^+}, 1-p+iu, p\right)\right)\right] du,$$

da cui segue banalmente la tesi. □

2.3 L'algoritmo di Newton-Raphson

Il sistema non lineare generato dalla Master Equation non è direttamente risolvibile; tuttavia, data l'analiticità delle derivate nel nostro caso, è possibile implementare un algoritmo di Newton-Raphson. Si noti che

$$\nabla_j(\Gamma) = \frac{p}{\pi} x_j^{p-1} \int_0^{+\infty} \Re\left[\varphi(u) e^{-iu\log(x_j)} \left(\beta\left(\frac{x_j^-}{x_j}, -iu, p\right) - \beta\left(\frac{x_j}{x_j^+}, 1-p+iu, p\right)\right)\right] du \tag{2.12}$$

dipende solo da x_{j-1}, x_j, x_{j+1}, ovvero la matrice hessiana $H_D(\Gamma)$ sarà tridiagonale. In particolare, $\nabla_1(\Gamma)$ dipende solo da x_1 e x_2, mentre $\nabla_N(\Gamma)$ dipende solo da x_{N-1} e x_N. Per semplicità spesso ometteremo il pedice D. Trovata la matrice hessiana, l'algoritmo avrà la classica forma

$$\Gamma^{(n+1)} = \Gamma^{(n)} - H\left(\Gamma^{(n)}\right)^{-1} \cdot \nabla\left(\Gamma^{(n)}\right). \tag{2.13}$$

Teorema 2.3.1 (Hessiana della distorsione)**.** *Sia $p \in (1, +\infty)$. La matrice hessiana della funzione di distorsione $D_p(\Gamma)$ definita in Definizione (1.3) è tridiagonale,*

con la seguente espressione per i termini non nulli:

$$H_{j,j+1}(\Gamma) = -\frac{p}{2\pi x_j^+}\left(\frac{x_{j+1}-x_j}{2}\right)^{p-1}\int_0^{+\infty}\Re\Big[\varphi(u)e^{-iu\log(x_j^+)}\Big]du\,, \tag{2.14}$$

$$H_{j,j-1}(\Gamma) = -\frac{p}{2\pi x_j^-}\left(\frac{x_j-x_{j-1}}{2}\right)^{p-1}\int_0^{+\infty}\Re\Big[\varphi(u)e^{-iu\log(x_j^-)}\Big]du\,, \tag{2.15}$$

$$H_{j,j}(\Gamma) = -\frac{p(p-1)}{\pi}x_j^{p-2}\int_0^{+\infty}\Re\Bigg[\varphi(u)e^{-iu\log(x_j)}\left(\beta\left(\frac{x_j^-}{x_j},-iu,p-1\right)\right. \tag{2.16}$$

$$\left.+\,\beta\left(\frac{x_j}{x_j^+},2-p+iu,p-1\right)\right)\Bigg]du + H(\Gamma)_{j,j-1} + H(\Gamma)_{j,j+1}\,. \tag{2.17}$$

Dimostrazione. Nella dimostrazione del Teorema 2.2 abbiamo visto che è possibile scrivere $\frac{\partial D(\Gamma)}{\partial x_i}$ come

$$\frac{\partial D(\Gamma)}{\partial x_i} = \int_{x_j^-}^{x_j} p(x_j-z)^{p-1}f(z)dz - \int_{x_j}^{x_j^+} p(z-x_j)^{p-1}f(z)dz\,.$$

Da questa espressone è facile ricavare le derivate miste, ovvero la sopradiagonale e la sottodiagonale della matrice hessiana. Ricordando che

$$x_j^+ = \frac{1}{2}(x_j+x_{j+1}) \quad \text{e} \quad x_j^- = \frac{1}{2}(x_{j-1}+x_j)\,,$$

si trova

$$\begin{aligned}\frac{\partial^2 D(\Gamma)}{\partial x_j \partial x_{j-1}} &= -\frac{1}{2}p(x_j-x_j^-)^{p-1}f(x_j^-)\\ &= -\frac{p}{2\pi x_j^-}\left(\frac{x_j-x_{j-1}}{2}\right)^{p-1}\int_0^{+\infty}\Re\Big[\varphi(u)e^{-iu\log(x_j^-)}\Big]du\,,\end{aligned}$$

e

$$\begin{aligned}\frac{\partial^2 D(\Gamma)}{\partial x_j \partial x_{j+1}} &= -\frac{1}{2}p(x_j^+-x_j)^{p-1}f(x_j^+)\\ &= -\frac{p}{2\pi x_j^+}\left(\frac{x_{j+1}-x_j}{2}\right)^{p-1}\int_0^{+\infty}\Re\Big[\varphi(u)e^{-iu\log(x_j^+)}\Big]du\,.\end{aligned}$$

Calcoliamo ora la diagonale principale:

$$\begin{aligned}\frac{\partial^2 D(\Gamma)}{\partial x_j \partial x_j} =& -\frac{1}{2}p(x_j - x_j^-)^{p-1} f(x_j^-) + \int_{x_j^-}^{x_j} p(p-1)(x_j - z)^{p-2} f(z)dz \\ & -\frac{1}{2}p(x_j^+ - x_j)^{p-1} f(x_j^+) + \int_{x_j}^{x_j^+} p(p-1)(z - x_j)^{p-2} f(z)dz \\ =& p(p-1)\left(\int_{x_j^-}^{x_j} (x_j - z)^{p-2} f(z)dz + \int_{x_j}^{x_j^+} (z - x_j)^{p-2} f(z)dz\right) \\ & + \frac{\partial^2 D(\Gamma)}{\partial x_j \partial x_{j-1}} + \frac{\partial^2 D(\Gamma)}{\partial x_j \partial x_{j+1}}\,.\end{aligned}$$

Si noti ora che gli integrali rimasti sono uguali a quelli della dimostrazione 2.2, ma con esponente $p-2$ anziché $p-1$. Lo stesso procedimento conclude la dimostrazione. □

Definito un criterio d'arresto, tramite l'algoritmo 2.13 siamo ora in grado di calcolare una discretizzazione ottima di un sottostante S_T, e quindi approssimare il valore atteso di un generico payoff ψ come

$$\mathbb{E}\big[\psi(S_T)\big] \approx \sum_{j=1}^{N} \psi(\hat{x}_j)\mathbb{P}(S_T \in \hat{C}_j)\,, \tag{2.18}$$

dove $\hat{x}_j$ rappresenta il j-esimo punto del quantizzatore ottimizzato $\hat{\Gamma}$ e $\hat{C}_j$ la relativa cella di Voronoi.

Capitolo 3

Applicazioni a derivati path-dependent

Per applicare il metodo della quantizzazione che abbiamo trattato è necessario conoscere la funzione caratteristica del sottostante a maturity. Ciò è molto comodo per prezzare opzioni plain vanilla usando modelli esponenziali di Lévy, grazie al teorema di Lévy-Khintchine, che offre una forma esplicita per la funzione caratteristica. Ci si può domandare se sia possibile usare la stessa metodologia per prezzare opzioni path-dependent, per esempio lookback e barriere, il cui payoff dipende dal massimo o dal minimo assunto dal sottostante durante la vita dell'opzione. Per farlo bisognerà adattare la funzione caratteristica del modello del sottostante al payoff e calcolarla tramite le identità di Spitzer.

3.1 Adattamento della funzione caratteristica

Riportiamo di seguito le principali notazioni e definizioni usate per la trattazione di questa sezione.
Il sottostante S_t sarà modellizzato con un processo esponenziale di Lévy, ovvero $S_t = S_0 e^{X_t}$, con X_t processo di Lévy. Il monitoraggio discreto presuppone l'esistenza di un insieme di istanti temporali equispaziati $(t_i = i\Delta)_{0 \le i \le n}$, con

$$t_0 = 0 \quad \text{l'inizio dell'opzione,} \tag{3.1}$$

$$t_N = T \quad \text{la maturity dell'opzione,} \tag{3.2}$$

$$\Delta = \frac{T}{n} \quad \text{il passo temporale.} \tag{3.3}$$

Da ora in avanti indicheremo con X_i il processo di Lévy valutato all'i-esimo istante temporale $t_i = i\Delta$ e con $M := \max_{i=0,\dots,n} X_i$ il massimo di X sulle date di monitoraggio. Definiamo infine la trasformata di Fourier:

Definizione 3.1 (Trasformata di Fourier)**.** Sia $g : \mathbb{R} \to \mathbb{C}$ una funzione integrabile. Indichiamo con $\mathscr{F}g : \mathbb{R} \to \mathbb{C}$ la sua trasformata di Fourier, definita da

$$\big(\mathscr{F}g\big)(\xi) \coloneqq \int_{-\infty}^{+\infty} e^{i\xi x} g(x)dx\,. \tag{3.4}$$

3.1.1 Call barriera up and out

Prendiamo in esempio il caso di una call barriera up and out con monitoraggio discreto. Denotando con $f_{X,M}(x,m)$ la densità congiunta di (X,M), possiamo esprimere il valore atteso del payoff come

$$\mathbb{E}\Big[\big(S_0e^X - K\big)^+ \mathbf{1}_{\{M<U\}}\Big] = \int_{\mathbb{R}^2} \big(S_0e^x - K\big)^+ \mathbf{1}_{\{m<U\}} f_{X,M}(x,m)d(x,m)\,, \tag{3.5}$$

con l'ovvio significato dei simboli usati. A prima vista sembra che sia necessario usare una quantizzazione bidimensionale nelle variabili X ed M, cosa non molto agevole. In realtà, grazie alla semplicità della dipendenza da M, è possibile ridurre il problema al caso monodimensionale già discusso, applicando una correzione alla funzione caratteristica. Infatti, essendo l'integranda in (3.5) positiva, grazie al teorema di Tonelli possiamo scrivere

$$\int_{\mathbb{R}^2} \big(S_0e^x - K\big)^+ \mathbf{1}_{\{m<U\}} f_{X,M}(x,m)d(x,m) = \int_{\mathbb{R}} \big(S_0e^x - K\big)^+ \int_{-\infty}^{U} f_{X,M}(x,m)dm \quad dx\,. \tag{3.6}$$

Notiamo che la funzione

$$\hat{f}(x) \coloneqq \int_{-\infty}^{U} f_{X,M}(x,m)dm \tag{3.7}$$

rappresenta la densità della legge condizionata di $X|M < U$ moltiplicata per la costante $\mathbb{P}(M < U)$, ovvero

$$\int_{-\infty}^{U} f_{X,M}(x,m)dm = \mathbb{P}(M < U) f_{X|M<U}(x)\,. \tag{3.8}$$

Infatti, ricordando che vale

$$\mathbb{P}(X \le x, M < U) = \mathbb{P}(X \le x | M < U)\mathbb{P}(M < U)\,, \tag{3.9}$$

ma anche

$$\mathbb{P}(X \leq x, M < U) = \int_{-\infty}^{x} \int_{-\infty}^{U} f_{X,M}(z,m)dm \quad dz\,, \tag{3.10}$$

uguagliando le due equazioni otteniamo

$$\mathbb{P}(X \leq x | M < U)\mathbb{P}(M < U) = \int_{-\infty}^{x} \int_{-\infty}^{U} f_{X,M}(z,m)dm \quad dz\,. \tag{3.11}$$

Ora è sufficiente derivare entrambi i membri per ottenere la relazione (3.8). Abbiamo quindi riscritto il valore atteso del payoff come

$$\mathbb{E}\Big[\big(S_0 e^X - K\big)^+ \mathbf{1}_{\{M<U\}}\Big] = \mathbb{P}\big(M < U\big)\mathbb{E}\Big[\big(S_0 e^X - K\big)^+ \Big| M < U\Big]\,. \tag{3.12}$$

Intuitivamente questo risultato ci dice che è possibile prezzare un'opzione barriera correggendo opportunamente la densità del sottostante, considerando la legge condizionata al massimo assunto fino alla maturity, che ricaveremo grazie alle identità di Spitzer. In pratica, vedremo come ottenere direttamente la trasformata di Fourier della funzione $\hat{f}$, che rappresenta una pseudodensità, non integrando a uno.

3.1.2 Call lookback fixed strike

Il caso del pricing di una call lookback è simile a quello della barriera, ma presenta una nuova difficoltà. Utilizzando la stessa notazione del caso precedente, dobbiamo calcolare

$$\mathbb{E}\Big[\big(S_0 e^M - K\big)^+\Big]\,. \tag{3.13}$$

In questo caso sarebbe possibile ottenere, grazie alle identità di Spitzer, la funzione caratteristica di M, ma non potremmo direttamente usare la quantizzazione, perché questa variabile aleatoria non è assolutamente continua. Infatti, dato che il monitoraggio è discreto, esiste una probabilità positiva che il massimo assunto nelle date di monitoraggio sia proprio il valore iniziale, cioè zero. Questo ovviamente significa che il massimo ha una massa di probabilità in zero, cosa che rende inutilizzabile la sua funzione caratteristica, ai fini della quantizzazione. Infatti, la quantizzazione richiede che la funzione caratteristica $\varphi(u)$ si annulli per $u \to \infty$, cosa che non accade per variabili aleatorie con componenti singolari. Per vederlo si può scrivere

$$\varphi_M(u) = \mathbb{E}\Big[e^{iuM}\Big] = \mathbb{E}\Big[e^{iuM}\big(\mathbf{1}_{\{M=0\}} + \mathbf{1}_{\{M>0\}}\big)\Big] \tag{3.14}$$

$$= \mathbb{P}\big(M = 0\big) + \mathbb{E}\Big[e^{iuM}\mathbf{1}_{\{M>0\}}\Big]\,, \tag{3.15}$$

dove abbiamo sfruttato il fratto che M sia non negativo. Dobbiamo ora mostrare che

$$\lim_{u\to\infty} \mathbb{E}\Big[e^{iuM}\mathbf{1}_{\{M>0\}}\Big] = 0\,. \tag{3.16}$$

Per fare ciò, è sufficiente mostrare che anche questo valore atteso può essere visto come una trasformata di Fourier, stando attenti al fatto che M non ammette densità, e poi applicare il lemma di Riemann-Lebesgue[1]. Definita una nuova misura μ su $\big(\Omega, \mathcal{F}\big)$ come

$$\mu(A) \coloneqq \mathbb{P}\big(A, M > 0\big) \quad \forall A \in \mathcal{F}\,, \tag{3.17}$$

possiamo scrivere

$$\mathbb{E}\Big[e^{iuM}\mathbf{1}_{\{M>0\}}\Big] = \int_\Omega e^{iuM(\omega)}\mathbf{1}_{\{M>0\}}(\omega)d\mathbb{P}(\omega) \tag{3.18}$$

$$= \int\limits_{\Omega\cap\{M>0\}} e^{iuM}d\mathbb{P}(\omega) = \int_\Omega e^{iuM(\omega)}d\mu(\omega)\,, \tag{3.19}$$

visto che $\mathbb{P}(A) = \mu(A)$ per ogni $A \subset \big\{\omega \in \Omega \mid M(\omega) > 0\big\}$ e che $\mu(M \leq 0) = 0$. Possiamo quindi ora esplicitare la derivata di Radon-Nikodym rispetto alla misura di Lebesgue e ottenere

$$\int_\Omega e^{iuM(\omega)}d\mu(\omega) = \int_{\mathbb{R}} e^{ium}\frac{d}{dm}\mu(M \leq m)dm \tag{3.20}$$

$$= \mathbb{P}\big(M > 0\big)\int_{\mathbb{R}} e^{ium}\frac{d}{dm}\mathbb{P}\big(M \leq m|M > 0\big)dm\,. \tag{3.21}$$

Essendo

$$\frac{d}{dm}\mathbb{P}\big(M \leq m|M > 0\big) = f_{M|M>0}(m) \tag{3.22}$$

la densità della variabile aleatoria $\big(M|M > 0\big)$, quindi integrabile, abbiamo mostrato che il valore atteso in (3.14) è effettivamente una trasformata di Fourier moltiplicata per una costante; applicando ora il lemma di Riemann-Lebesgue otteniamo

$$\lim_{u\to\pm\infty} \varphi_M(u) = \mathbb{P}\big(M = 0\big)\,, \tag{3.23}$$

ciò che volevamo dimostrare. Abbiamo quindi mostrato che, pur conoscendo la funzione caratteristica di M e nonostante la semplicità del payoff, non possiamo applicare direttamente la quantizzazione. Per poterla applicare, dovremo escludere

[1] Il lemma di Riemann-Lebesgue afferma che la trasformata di Fourier di una funzione $h : \mathbb{R}^d \to \mathbb{R}$ integrabile si annulla all'infinito.

il caso $M = 0$, e poi ragionare in modo analogo al caso barriera, visto che è possibile scrivere il payoff della lookback come

$$\mathbb{E}\Big[\big(S_0 e^M - K\big)^+\Big] = \big(S_0 - K\big)^+ \mathbb{P}\big(M = 0\big) + \mathbb{E}\Big[\big(S_0 e^M - K\big)^+ \mathbf{1}_{\{M>0\}}\Big]\,. \quad (3.24)$$

Applicando gli stessi passaggi del caso precedente, si trova che per applicare la quantizzazione per il calcolo del valore atteso al secondo membro della (3.24) bisogna usare la trasformata di Fourier della funzione

$$m \mapsto \frac{d}{dm}\mathbb{P}\big(M \le m, M > 0\big) = \mathbb{P}\big(M > 0\big) f_{M|M>0}(m)\,. \quad (3.25)$$

Riassumendo brevemente, per prezzare derivati plain vanilla con il metodo della quantizzazione, serve la funzione caratteristica del processo di Lévy che modellizza il logaritmo del sottostante. Nel caso della barriera, abbiamo visto come sia possibile cambiare probabilità per incorporare nella trasformata di Fourier gli effetti del payoff sul prezzo. Questo caso può essere risolto utilizzando la trasformata di Fourier della funzione

$$x \mapsto \frac{d}{dx}\mathbb{P}\big(X \le x, M < U\big)\,. \quad (3.26)$$

Nel caso della lookback, invece, si può calcolare direttamente la funzione caratteristica di M. Tuttavia, come abbiamo visto, questa non è applicabile al nostro metodo perché non tende a zero. Per ovviare a questo problema, scomponiamo il calcolo del valore atteso nella parte singolare e in quella assolutamente continua: per entrambi gli aspetti useremo la funzione caratteristica $\varphi_M(u)$ di M, dalla quale otteniamo

$$\mathbb{P}\big(M = 0\big) = \lim_{u\to+\infty} \varphi_M(u) \quad (3.27)$$

e poi

$$\mathbb{E}\Big[e^{iuM}\mathbf{1}_{\{M>0\}}\Big] = \varphi_M(u) - \mathbb{P}\big(M = 0\big)\,. \quad (3.28)$$

3.2 Le identità di Spitzer

Per questa sezione faremo riferimento principalmente a [5].

3.2.1 Introduzione

Introduciamo prima le notazioni e definizioni usate, mantenendo quelle esposte precedentemente.

Definizione 3.2 (Trasformata Zeta unilaterale)**.** Sia $v = (v_n)_{n\in\mathbb{N}} \subset \mathbb{C}$ una successione di numeri complessi. La sua trasformata Zeta unilaterale è definita da

$$\big(\mathcal{Z}v\big)(q) \coloneqq \sum_{n=0}^{+\infty} v_n q^n, \quad q \in \mathbb{C}\,. \tag{3.29}$$

È utile riportare anche la sua inversa:

Proposizione 3.2.1 (Inversa della trasformata Zeta)**.** *Sia $v = (v_n)_{n\in\mathbb{N}} \subset \mathbb{C}$ una successione di numeri complessi e $\mathcal{Z}v$ la sua trasformata Zeta. Allora*

$$v_n = \frac{1}{2\pi i}\int_{C(r)} \frac{\big(\mathcal{Z}v\big)(z)}{z^{n+1}} dz\,, \tag{3.30}$$

dove $C(r)$ è una qualsiasi circonferenza di raggio $r \in (0,1)$ centrata nell'origine.

Infine definiamo la decomposizione di Wiener-Hopf:

Definizione 3.3 (Decomposizione additiva di Wiener-Hopf)**.** Sia $\Phi : \mathbb{C} \to \mathbb{C}$. Due funzioni $\Phi_+, \Phi_- : \mathbb{C} \to \mathbb{C}$ costituiscono la decomposizione additiva di Wiener-Hopf di Φ se $\Phi(z) = \Phi_+(z) + \Phi_-(z)$ per ogni $z \in \mathbb{C}$ e Φ_+ e Φ_- sono analitiche rispettivamente nel semipiano complesso superiore e inferiore.

Osservazione 4. Le due funzioni Φ_- e Φ_+ possono essere calcolate in generale grazie alle seguenti equazioni:

$$\Phi_+(\alpha) = \frac{1}{2\pi i}\int_{\gamma_+} \frac{\Phi(z)}{z-\alpha} dz\,, \tag{3.31}$$

$$\Phi_-(\alpha) = -\frac{1}{2\pi i}\int_{\gamma_-} \frac{\Phi(z)}{z-\alpha} dz\,, \tag{3.32}$$

dove γ_+ e γ_- sono due rette parallele all'asse reale rispettivamente sopra e sotto la retta $z = \alpha$.

Osservazione 5. Utilizzando la tecnica di Wiener-Hopf è anche possibile decomporre una funzione in due fattori anziché due addendi: in questo caso si parlerà di fattorizzazione di Wiener-Hopf. Per fattorizzare una funzione f basterà decomporre additivamente il suo logaritmo:

$$\log f(z) \overset{\text{WH}}{=} \bar{f}_-(z) + \bar{f}_+(z) \Rightarrow f(z) = e^{\bar{f}_-(z)} e^{\bar{f}_+(z)}\,. \tag{3.33}$$

3.2.2 Le identità di Spitzer

Manipolando opportunamente le identità di Spitzer in [14], ricavate tramite argomenti di calcolo combinatorio, è possibile trovare una forma implicita per le leggi di vari processi, utili per il pricing di opzioni path-dependent. In particolare, esse forniscono preziose equazioni per le distribuzioni congiunte del sottostante e degli estremi assunti durante il periodo di monitoraggio. Definiamo quindi il massimo e il minimo di X come

$$M_n \coloneqq \max_{i=0,\dots,n} X_i \,, \tag{3.34}$$

$$m_n \coloneqq \min_{i=0,\dots,n} X_i \,. \tag{3.35}$$

Date due soglie $l, u \in \mathbb{R}$, rispettivamente la barriera inferiore e quella superiore, possiamo definire le densità[2] che ci servono:

$$f_X(x,n)dx \coloneqq \mathbb{P}\big(X_n \in [x, x+dx)\big) \,, \tag{3.36}$$

$$f_m(x,n)dx \coloneqq \mathbb{P}\big(m_n \in [x, x+dx)\big) \,, \tag{3.37}$$

$$f_M(x,n)dx \coloneqq \mathbb{P}\big(M_n \in [x, x+dx)\big) \,, \tag{3.38}$$

$$f_{X,m}(x,n)dx \coloneqq \mathbb{P}\big(X_n \in [x, x+dx), m_n > l\big) \,, \tag{3.39}$$

$$f_{X,M}(x,n)dx \coloneqq \mathbb{P}\big(X_n \in [x, x+dx), M_n < u\big) \,. \tag{3.40}$$

Definita ora

$$\Phi(\xi,q) \coloneqq 1 - q\mathbb{E}\big[e^{i\xi X_\Delta}\big] = 1 - q\varphi_\Delta(\xi) \overset{WH}{=} \Phi_+(\xi,q)\Phi_-(\xi,q) \,, \tag{3.41}$$

dove $\varphi_t(\xi)$ rappresenta la funzione caratteristica del processo di Lévy X ad un dato istante t, siamo finalmente in grado di scrivere le equazioni di Spitzer come in [5]:

$$\big(\mathcal{Z}\mathscr{F} f_X\big)(\xi,q) = \frac{1}{\Phi(\xi,q)} \,, \tag{3.42}$$

$$\big(\mathcal{Z}\mathscr{F} f_m\big)(\xi,q) = \frac{1}{\Phi_+(0,q)\Phi_-(\xi,q)} \,, \tag{3.43}$$

$$\big(\mathcal{Z}\mathscr{F} f_M\big)(\xi,q) = \frac{1}{\Phi_+(\xi,q)\Phi_-(0,q)} \,, \tag{3.44}$$

$$\big(\mathcal{Z}\mathscr{F} f_{X,m}\big)(\xi,q) = e^{il\xi}\frac{P_+(\xi,q)}{\Phi_+(\xi,q)} \,, \tag{3.45}$$

$$\big(\mathcal{Z}\mathscr{F} f_{X,M}\big)(\xi,q) = e^{iu\xi}\frac{Q_-(\xi,q)}{\Phi_-(\xi,q)} \,, \tag{3.46}$$

[2]Specifichiamo che le equazioni (3.39) e (3.40) definiscono due pseudodensità, come visto nella sezione precedente, dato che non integrano a 1.

dove

$$P(\xi,q) \coloneqq \frac{e^{-il\xi}}{\Phi_-(\xi,q)} = P_+(\xi,q) + P_-(\xi,q)\,, \quad \text{e} \tag{3.47}$$

$$Q(\xi,q) \coloneqq \frac{e^{-iu\xi}}{\Phi_+(\xi,q)} = Q_+(\xi,q) + Q_-(\xi,q)\,, \tag{3.48}$$

dove in questo caso la decomposizione è di tipo additivo. La dimostrazione rigorosa di queste equazioni non rientra tra gli obiettivi di questa tesi, ma riteniamo opportuno ed interessante riportare in appendice B uno sketch su cui si basa.
Il calcolo della fattorizzazione di Wiener-Hopf è riportato invece in appendice C, dove è illustrato un metodo basato sulla trasformata di Hilbert, per il quale seguiamo [5].
Per quanto riguarda l'inversione della trasformata Zeta, è utile sfruttare l'approssimazione ricavata in [1]:

$$v_n = \frac{1}{2\pi i}\int_{C(r)} \frac{\big(\mathcal{Z}v\big)(z)}{z^{n+1}}dz \approx \frac{1}{2nr^n}\sum_{j=1}^{2n}(-1)^j \Re\Big[\big(\mathcal{Z}v\big)(re^{i\pi\frac{j}{n}})\Big]\,. \tag{3.49}$$

La formula è stata ricavata con una regola di integrazione trapezoidale dopo il cambio di variabili $z = re^{iu}$. Il raggio r della circonferenza di integrazione è arbitrario in $(0,1)$, ma, usando la formula approssimata, per avere una precisione di 10^{-k} serve porre $r = 10^{-\frac{k}{2n}}$, come spiegato in [1].

3.3 L'algoritmo

Diamo ora l'algoritmo da implementare per valutare derivati path-dependent applicando la quantizzazione.

1. Data $\varphi_t(\xi)$ la funzione caratteristica del processo di Lévy che modellizza l'esponente del sottostante, si calcola $\Phi(q,\xi) = 1 - q\varphi_\Delta(\xi)$;

2. si calcola la fattorizzazione di Wiener-Hopf di $\Phi(q,\cdot)$ con il metodo riportato in Appendice C;

3. in base al payoff che si vuole valutare si calcola uno dei membri di destra delle equazioni da (3.42) a (3.46), che rappresentano la trasformata $\mathcal{Z}$ della funzione caratteristica comprensiva dell'informazione sugli estremi del sottostante;

4. si inverte la trasformata $\mathcal{Z}$ con la formula (3.49);

5. se necessario, per esempio nel caso di una call lookback a strike fisso, si elimina la componente singolare dalla funzione caratteristica, come visto nel Paragrafo 3.1.2;

6. si implementa l'algoritmo di Newton-Raphson trattato nella Sezione 2.3 per trovare la griglia ottima di quantizzazione su cui calcolare il valore atteso;

7. si calcolano le probabilità che il sottostante appartenga alle celle di Voronoi con le formule della Proposizione 2.1.2;

8. infine si approssima il valore atteso del payoff con la sommatoria espressa dalla formula (2.18), a cui si aggiunge eventualmente la componente singolare.

Capitolo 4

Modelli e risultati numerici

Presentiamo in questo capitolo i risultati ottenuti, dopo aver descritto brevemente i modelli utilizzati. Abbiamo visto che il metodo di quantizzazione trattato necessita della conoscenza della funzione caratteristica, cosa che rende particolarmente congeniale l'uso dei modelli esponenziali di Lévy: se X_t è un processo di Lévy, il modello esponenziale per il sottostante S_t è $S_t = S_0 e^{X_t}$.
I processi di Lévy permettono di cogliere aspetti importanti di modellizzazione, per esempio i salti, pur mantenendo molte desiderabili proprietà. In particolare useremo i modelli Gaussiano, di Merton, di Kou, il modello Variance Gamma e il Normal Inverse Gaussian.

4.1 I processi di Lévy

In questa sezione diamo brevemente una panoramica sulla teoria riguardante questi processi, per la quale seguiremo principalmente [3] e [11].

Definizione 4.1 (Processo di Lévy)**.** Dato uno spazio di probabilità filtrato che soddisfa le condizioni usuali $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,+\infty)}, \mathbb{P}\right)$, un processo stocastico $(X_t)_{t\geq 0}$ su questo spazio è di Lévy se

i) $X_0 = 0$ quasi certamente;

ii) i suoi incrementi sono indipendenti, ovvero per ogni $0 \leq s \leq t$, si ha che $X_t - X_s$ è indipendente da $\mathcal{F}_s$;

iii) i suoi incrementi sono stazionari, ovvero per ogni $0 \leq s \leq t$, si ha che $X_t - X_s \sim X_{t-s}$;

iv) quasi ogni traiettoria è una funzione càdlàg[1];

v) è stocasticamente continuo, ovvero $\forall \varepsilon > 0, \lim_{h\to 0} \mathbb{P}\big(|X_{t+h} - X_t| > \varepsilon\big) = 0\,.$

Definiamo ora i due ingredienti fondamentali per la modellizzazione dei salti: la misura aleatoria J, che conta il numero di salti di una data ampiezza in un certo intervallo di tempo, e il numero atteso di salti per unità di tempo ν.

Definizione 4.2. Per ogni insieme boreliano $B \subset [0, +\infty) \times \mathbb{R}$ definiamo

$$J_X(B) \coloneqq \#\Big\{\big(t, X_t - X_{t^-}\big) \in B\Big\} \tag{4.1}$$

il numero dei salti di X negli istanti e con le ampiezze date da B, dove abbiamo usato la notazione $X_{t^-} = \lim_{s\to t^-} X_s$. $J_X\big([t_1, t_2] \times A\big)$ rappresenta quindi la variabile aleatoria che conta il numero di salti fatti dal processo X nell'intervallo temporale $[t_1, t_2]$ con ampiezza $l \in A$.

Definizione 4.3 (Misura di Lévy)**.** Per ogni insieme boreliano $A \subset \mathbb{R}$ definiamo

$$\nu(A) \coloneqq \mathbb{E}\Big[\#\big\{t \in [0,1] : \Delta X_t \neq 0, \Delta X_t \in A\big\}\Big] \tag{4.2}$$

il valore atteso del numero di salti del processo X per unità temporale.

Osservazione 6. Se $\nu(\mathbb{R}) < \infty$, allora quasi ogni traiettoria ha un numero finito di salti per ogni intervallo temporale compatto, ed il processo si dice ad *attività finita*. Se invece $\nu(\mathbb{R}) = \infty$, quasi ogni traiettoria ha un numero infinito di salti in ogni intervallo temporale, ed il processo si dice ad *attività infinita*. In quest'ultimo caso la somma delle ampiezze dei salti diventa una serie infinita, la cui convergenza impone dei vincoli alla misura ν, come illustrato nel prossimo teorema.

Teorema 4.1.1 (Decomposizione di Lévy-Itô)**.** *Sia X un processo di Lévy e ν la sua misura di Lévy. Allora ν è una misura di Radon[2] su $\mathbb{R} \setminus \{0\}$ e*

$$\int_{|x|\leq 1} |x|^2 \nu(dx) < \infty, \quad \int_{|x|\geq 1} \nu(dx) < \infty\,. \tag{4.3}$$

Inoltre esistono $b, \sigma \in \mathbb{R}$ e W_t moto browniano tali che

$$X_t = bt + \sigma W_t + X_t^l + \lim_{\varepsilon\to 0^+} \tilde{X}_t^\varepsilon\,, \tag{4.4}$$

[1]Una funzione reale di variabile reale si dice càdlàg se è continua da destra e ha limite sinistro finito in ogni punto.

[2]Una misura è di Radon su un insieme misurabile B se è finita su ogni sottoinsieme compatto e misurabile di B.

dove

$$X_t^l = \int_{1\leq|x|,s\in[0,t]} xJ_X(ds\times dx)\,, \tag{4.5}$$

$$X_t^\varepsilon = \int_{\varepsilon\leq|x|<1,s\in[0,t]} x\big(J_x(ds\times dx)-\nu(dx)ds\big)\,. \tag{4.6}$$

I termini nella rappresentazione (4.4) sono indipendenti e la convergenza dell'ultimo termine è quasi certa e uniforme in t.

Questa rappresentazione mette in evidenza la natura di questi processi, che sono quindi composti da un drift deterministico b, una componente diffusiva σ e una componente che racchiude i salti. Se ne deduce che i processi di Lévy sono completamente caratterizzati dalla tripletta (b,σ,ν).
Enunciamo ora uno dei risultati principali della teoria sui processi di Lévy, che dà una forma esplicita alla funzione caratteristica di questi processi.

Teorema 4.1.2 (Rappresentazione di Lévy-Khintchine)**.** *Sia $(X_t)_t \geq 0$ un processo di Lévy con tripletta caratteristica (b,σ,ν). Allora la sua funzione caratteristica è data da*

$$\varphi_{X_t}(u) = e^{t\psi(u)}\,, \tag{4.7}$$

dove

$$\psi(u) = ibu - \frac{1}{2}\sigma^2u^2 + \int_{\mathbb{R}\setminus\{0\}} e^{iux}-1-iux\mathbf{1}_{|x|<1}\nu(dx)\,. \tag{4.8}$$

Siamo ora finalmente in grado di illustrare i modelli usati per l'implementazione del metodo della quantizzazione.

4.2 I modelli utilizzati

Illustriamo i modelli utilizzati, specificandone l'esponente caratteristico. Ricordiamo che, essendo modelli esponenziali, con X_t indichiamo il processo di Lévy, mentre il sottostante sarà $S_t = S_0e^{X_t}$.
Siccome per il pricing di derivati è necessario usare sempre il drift risk-neutral, ometteremo il drift dall'esponente caratteristico.

4.2.1 Gaussiano

Il modello Gaussiano è il più semplice modello esponenziale di Lévy possibile, che non presenta salti. È caratterizzato soltanto dal parametro diffusivo σ e si ottiene prendendo come esponente un moto browniano, ovvero

$$X_t = bt + \sigma W_t\,. \tag{4.9}$$

come nel modello di Black e Scholes.
L'esponente caratteristico in questo caso è

$$\psi(u) = -\frac{1}{2}\sigma^2 u^2 \,. \tag{4.10}$$

Per i risultati numerici abbiamo utilizzato $\sigma = 0.3$.

4.2.2 Merton

Il modello di Merton è un processo ad attività finita, ovvero con numero finito di salti per ogni intervallo temporale. Le ampiezze dei salti sono distribuite come una variabile aleatoria normale con media μ e varianza δ^2. Questo modello comprende anche una componente diffusiva, caratterizzata dal parametro σ, e per questo fa parte dei modelli detti *jump-diffusion*. Possiamo quindi scrivere

$$X_t = bt + \sigma W_t + \sum_{i=1}^{N_t} Y_i \,, \tag{4.11}$$

dove N_t è un processo di Poisson con parametro λ, che modellizza il numero (aleatorio) di salti fino all'istante t, mentre Y_i è l'ampiezza (con segno) dell'i-esimo salto. L'esponente caratteristico di questo processo è

$$\psi(u) = -\frac{1}{2}\sigma^2 u^2 + \lambda\Big(e^{-\frac{1}{2}\delta^2 u^2 + iu\mu} - 1\Big) \,. \tag{4.12}$$

Per i risultati numerici abbiamo utilizzato $\sigma = 0.1$, $\lambda = 3$, $\delta = 0.01$ e $\mu = 0.01$.

4.2.3 Kou

Il modello di Kou è molto simile a quello di Merton, da cui differisce soltanto per la legge dei salti. Le ampiezze dei salti positivi e di quelli negativi sono distribuite entrambe con legge esponenziale, ma di parametri diversi: λ_+ per i salti positivi e λ_- per i salti negativi. Anche il modello di Kou si definisce *jump-diffusion*, potendo essere scritto nella forma

$$X_t = bt + \sigma W_t + \sum_{i=1}^{N_t} Y_i \,, \tag{4.13}$$

con la stessa notazione del caso del modello di Merton. Stavolta i salti Y_i avranno densità

$$f_{Y_i}(x) = p\lambda_+ e^{-\lambda_+ x}\mathbf{1}_{x>0} + (1-p)\lambda_- e^{-\lambda_- |x|}\mathbf{1}_{x<0} \,, \tag{4.14}$$

dove p è la probabilità che il salso sia positivo. L'esponente caratteristico è dato da

$$\psi(u) = -\frac{1}{2}\sigma^2 u^2 + iu\lambda\left(\frac{p}{\lambda_+ - iu} - \frac{1-p}{\lambda_- + iu}\right). \tag{4.15}$$

Per i risultati numerici abbiamo utilizzato $\sigma = 0.1$, $\lambda = 3$, $p = 0.3$, $\lambda_+ = 40$ e $\lambda_- = 12$.

4.2.4 Variance Gamma

Il modello Variance Gamma è un processo ad attività infinita e, non avendo componente diffusiva, è detto anche *pure-jumps*. Si ottiene tramite subordinazione[3] di un moto browniano con drift ϑ e diffusione σ con un processo Gamma[4] con media del tasso di crescita μ unitaria e varianza κ. Detto W_t il moto browniano aritmetico (con drift ϑ e diffusione σ), si ha dunque

$$X_t = W_{S_t}, \tag{4.16}$$

dove S_t è il processo Gamma suddetto.
L'esponente caratteristico è

$$\psi(u) = -\frac{1}{\kappa}\log\left(1 + \frac{u^2\sigma^2\kappa}{2} - i\vartheta\kappa u\right). \tag{4.17}$$

Per i risultati numerici abbiamo utilizzato $\sigma = 0.1$, $\kappa = 0.1$ e $\vartheta = 0.04$.

4.2.5 Normal Inverse Gaussian

Anche il modello NIG, come il Variance Gamma, è un processo *pure-jumps* ad attività infinita ottenuto tramite subordinazione. In questo caso il subordinatore è dato da un processo Inverse Gaussian[5] e l'esponente caratteristico è

$$\psi(u) = \frac{1}{\kappa} - \frac{1}{\kappa}\sqrt{1 + u^2\sigma^2\kappa - 2i\vartheta u\kappa}, \tag{4.18}$$

dove ϑ e σ sono il drift e la diffusione del moto browniano subordinato, mentre κ è la varianza per unità di tempo del processo subordinatore.
Per i risultati numerici abbiamo utilizzato $\sigma = 0.2$, $\kappa = 0.1$ e $\vartheta = 0.1$.

[3] Con subordinazione si intende il cambio della variabile temporale con un processo detto subordinatore, che deve quindi essere un processo di Lévy crescente.

[4] Il processo Gamma è un processo di Lévy crescente e pure-jumps, con misura di Lévy $\nu(x) = \frac{\mu^2}{\kappa}x^{-1}e^{-\frac{\mu}{\kappa}x}$.

[5] Un processo Inverse Gaussian A_t è definito come il primo tempo di passaggio di un moto browniano ad un livello α, ovvero $A_t := \inf\{s > 0 \mid W_s = \alpha t\}$.

4.3 Risultati numerici

In generale il metodo della quantizzazione richiede la scelta di numerosi parametri arbitrari e l'applicazione del metodo di Newton-Raphson multidimensionale, che può non convergere a una soluzione. Per quanto riguarda l'applicazione al pricing di derivati plain vanilla, esso risulta sufficientemente stabile ed efficiente, essendo la funzione caratteristica nota analiticamente: in questo caso l'algoritmo converge senza problemi numerici rilevanti. Il pricing di derivati path-dependent è possibile teoricamente, come abbiamo visto, ma risulta molto più instabile e meno preciso, dato che solo il calcolo della funzione caratteristica richiede prima la fattorizzazione di Wiener-Hopf e poi l'inversione della trasformata Zeta. In questo caso quindi l'errore introdotto sulla funzione caratteristica può influenzare negativamente l'esito dell'algoritmo di Newton-Raphson: questo si manifesta numericamente per esempio con quantizzatori con punti non ammissibili, o con la non convergenza degli integrali per il calcolo delle derivate. Inoltre la fattorizzazione di Wiener-Hopf e conseguentemente l'inversione della trasformata Zeta diventano molto onerose all'aumentare delle date di monitoraggio, tanto da rendere il metodo troppo pesante computazionalmente ad esempio per un monitoraggio giornaliero.
Vediamo ora i risultati numerici ottenuti usando i modelli che abbiamo elencato, confrontandoli con un benchmark dato dal metodo FFT per le call plain vanilla e con il metodo Monte Carlo (con 10 milioni di simulazioni) per lookback e barriere.

4.3.1 I parametri delle opzioni e dei modelli

I seguenti risultati sono stati ottenuti ipotizzando un tasso di interesse privo di rischio $r = 5\%$ e prezzo spot $S_0 = 100$ per il sottostante; mentre come barriere superiori ed inferiori abbiamo usato rispettivamente $U = 130$ e $L = 70$. Per la quantizzazione delle call plain vanilla abbiamo usato $N = 200$ punti per la griglia, ottenendo buoni risultati. In questo caso le performance dell'algoritmo implementato in MATLAB sono buone, con tempi riportati nelle tabelle per il calcolo di tutti gli strike, utilizzando un laptop con una CPU da 2.7 GHz. Per quanto riguarda i derivati path-dependent, abbiamo empiricamente posto $N = 60$, ma la convergenza è meno stabile ed il metodo risulta molto meno efficiente: servono decine di secondi per il calcolo di tutti gli strike con solo 12 date di monitoraggio, i cui risultati sono riportati dalla Tabella 4.6 alla Tabella 4.20, mentre aumentandone la frequenza il calcolo diventa ancora più oneroso e impreciso, a causa della fattorizzazione di Wiener-Hopf. In particolare il metodo può ancora essere utilizzato con 52 date di monitoraggio, cioè con frequenza settimanale, come mostrano le Tabelle 4.21 e 4.22, ma risulta praticamente inutilizzabile con un monitoraggio giornaliero. In tutte le tabelle è riportato l'errore relativo espresso in basis point (centesimi di

punto percentuale) e l'intervallo di confidenza al 99% per i prezzi calcolati con le simulazioni Monte Carlo.

4.3.2 Risultati numerici

Mostriamo ora le tabelle riassuntive dei risultati numerici, dove il tempo complessivo riportato si riferisce al calcolo dei prezzi con la quantizzazione su tutti gli strike, ma non comprende il calcolo dei valori di riferimento.

4.3.2.1 Risultati call plain vanilla

In questa sezione riportiamo i risultati relativi al pricing della call plain vanilla con la quantizzazione, usando tutti i modelli citati. Il prezzo di benchmark è dato dal metodo FFT, tranne per il modello di Black e Scholes, per cui abbiamo usato la formula chiusa. Riteniamo i risultati sufficientemente precisi, essendo quasi tutti sotto la soglia di 1 basis point.

Tabella 4.1: Call plain vanilla con modello Gaussiano

Strike	Formula chiusa	Quantizzazione	Err. rel (bp)
80	26.46209	26.46147	0.23198
90	19.69744	19.69707	0.18950
100	14.23125	14.23082	0.30414
110	10.02008	10.02004	0.03769
120	6.90400	6.90396	0.05486

Tempo complessivo: 1.76s.

Tabella 4.2: Call plain vanilla con modello di Merton

Strike	FFT	Quantizzazione	Err. rel (bp)
80	23.91229	23.91230	0.00626
90	14.65968	14.65970	0.01109
100	6.90416	6.90423	0.10493
110	2.28692	2.28683	0.39911
120	0.52153	0.52145	1.48978

Tempo complessivo: 5.40s.

Tabella 4.3: Call plain vanilla con modello di Kou

Strike	FFT	Quantizzazione	Err. rel (bp)
80	24.96132	24.96120	0.04708
90	16.94140	16.94133	0.04274
100	10.21362	10.21338	0.22823
110	5.24735	5.24726	0.17473
120	2.21369	2.21360	0.42095

Tempo complessivo: 5.71s.

Tabella 4.4: Call plain vanilla con modello NIG

Strike	FFT	Quantizzazione	Err. rel (bp)
80	24.56588	24.56535	0.21570
90	16.66288	16.66237	0.30811
100	10.46566	10.46525	0.39426
110	6.16248	6.16232	0.25454
120	3.46697	3.46691	0.18667

Tempo complessivo: 4.12s.

Tabella 4.5: Call plain vanilla con modello VG

Strike	FFT	Quantizzazione	Err. rel (bp)
80	23.91222	23.91211	0.04542
90	14.62285	14.62272	0.08983
100	6.77273	6.77276	0.04292
110	2.21691	2.21664	1.22713
120	0.54782	0.54777	0.90629

Tempo complessivo: 7.29s.

4.3.2.2 Risultati call barriera down and out

In questa sezione sono riportati i prezzi delle call barriera down and out con 12 date di monitoraggio, confrontati con quelli dati dal metodo Monte Carlo con 10 milioni di simulazioni, dei quali per completezza riportiamo anche l'intervallo di confidenza al 99% . I risultati sono buoni e con errori dell'ordine di 1 basis point, mentre notiamo che il calcolo della funzione caratteristica aggiustata fa aumentare sensibilmente il tempo richiesto e che l'errore introdotto può far rallentare la convergenza dell'algoritmo di Newton-Raphson.

Tabella 4.6: Call barriera D&O con modello Gaussiano

Tempo complessivo: 33.24s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	26.22446	26.22199	0.94281	[26.20179, 26.24713]
90	19.60520	19.60549	0.14945	[19.58453, 19.62587]
100	14.19260	14.19172	0.62059	[14.17425, 14.21095]
110	10.00159	10.00078	0.80316	[9.98570, 10.01747]
120	6.89480	6.89534	0.78155	[6.88132, 6.90828]

Tempo calcolo funzione caratteristica: 8.63s.
Tempo algoritmo Newton-Raphson: 24.60.

Tabella 4.7: Call barriera D&O con modello di Merton

Tempo complessivo: 11.80s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	23.91590	23.91085	2.11242	[23.90751, 23.92429]
90	14.66298	14.65806	3.35513	[14.65495, 14.67100]
100	6.90696	6.90352	4.98170	[6.90048, 6.91343]
110	2.28769	2.28642	5.57803	[2.28373, 2.29165]
120	0.52180	0.52121	11.44751	[0.51995, 0.52366]

Tempo calcolo funzione caratteristica: 9.03s.
Tempo algoritmo Newton-Raphson: 2.76.

Tabella 4.8: Call barriera D&O con modello di Kou

Tempo complessivo: 25.30s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	24.94198	24.94372	0.69521	[24.92835, 24.95561]
90	16.93840	16.93886	0.27637	[16.92636, 16.95043]
100	10.21383	10.21267	1.13939	[10.20402, 10.22364]
110	5.24833	5.24498	6.37043	[5.24114, 5.25552]
120	2.21500	2.21293	9.36580	[2.21036, 2.21964]

Tempo calcolo funzione caratteristica: 8.63s.
Tempo algoritmo Newton-Raphson: 16.67s.

Tabella 4.9: Call barriera D&O con modello NIG

Tempo complessivo: 28.20s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	24.55454	24.54550	3.67929	[24.53833, 24.57075]
90	16.66316	16.65522	4.76492	[16.64833, 16.67799]
100	10.46787	10.46026	7.27198	[10.45514, 10.48060]
110	6.16444	6.16176	4.34388	[6.15412, 6.17476]
120	3.46906	3.46642	7.61895	[3.46103, 3.47709]

Tempo calcolo funzione caratteristica: 10.13s.
Tempo algoritmo Newton-Raphson: 18.07s.

Tabella 4.10: Call barriera D&O con modello VG

Tempo complessivo: 12.13s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	23.91326	23.91098	0.95553	[23.90500, 23.92153]
90	14.62339	14.62226	0.77375	[14.61544, 14.63134]
100	6.77169	6.77069	1.48247	[6.76519, 6.77819]
110	2.21561	2.21580	0.89419	[2.21155, 2.21967]
120	0.54751	0.54702	9.06505	[0.54546, 0.54957]

Tempo calcolo funzione caratteristica: 9.25s.
Tempo algoritmo Newton-Raphson: 2.87s.

4.3.2.3 Risultati call barriera up and out

Riportiamo nelle tabelle sottostanti i risultati delle call barriera up and out con 12 date di monitoraggio, confrontati anche in questo caso con quelli dati dal metodo Monte Carlo con 10 milioni di simulazioni. Notiamo che stavolta l'errore è più marcato rispetto ai precedenti, essendo dell'ordine delle decine di basis point.

Tabella 4.11: Call barriera U&O con modello Gaussiano

Tempo complessivo: 25.32s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	8.96793	8.95903	9.92787	[8.95762, 8.97823]
90	5.08089	5.07519	11.23381	[5.07347, 5.08831]
100	2.41860	2.41594	10.98593	[2.41389, 2.42332]
110	0.86594	0.86513	9.33298	[0.86350, 0.86838]
120	0.16587	0.16563	13.93668	[0.16507, 0.16666]

Tempo calcolo funzione caratteristica: 7.94s.
Tempo algoritmo Newton-Raphson: 17.38s.

Tabella 4.12: Call barriera U&O con modello di Merton

Tempo complessivo: 20.64s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	22.80193	22.79813	1.66312	[22.79374, 22.81011]
90	13.75804	13.75411	2.86086	[13.75054, 13.76555]
100	6.21107	6.20658	7.22894	[6.20533, 6.21680]
110	1.80083	1.79946	7.63321	[1.79777, 1.80390]
120	0.24339	0.24271	27.98606	[0.24248, 0.24429]

Tempo calcolo funzione caratteristica: 8.12s.
Tempo algoritmo Newton-Raphson: 12.52s.

Tabella 4.13: Call barriera U&O con modello di Kou

Tempo complessivo: 32.39s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	18.79205	18.79833	3.34039	[18.78009, 18.80402]
90	11.90392	11.90809	3.50825	[11.89439, 11.91344]
100	6.30080	6.30274	3.08916	[6.29407, 6.30752]
110	2.44087	2.43989	4.01975	[2.43708, 2.44467]
120	0.47595	0.47426	35.63500	[0.47466, 0.47725]

Tempo calcolo funzione caratteristica: 8.86s.
Tempo algoritmo Newton-Raphson: 23.52s.

Tabella 4.14: Call barriera U&O con modello NIG

Tempo complessivo: 22.03s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	15.01122	15.00623	3.32567	[15.00019, 15.02226]
90	8.73065	8.72526	6.16941	[8.72214, 8.73915]
100	4.14843	4.14357	11.69904	[4.14275, 4.15411]
110	1.43498	1.43332	11.53457	[1.43199, 1.43796]
120	0.25206	0.25164	16.60392	[0.25110, 0.25301]

Tempo calcolo funzione caratteristica: 8.55s.
Tempo algoritmo Newton-Raphson: 13.47s.

Tabella 4.15: Call barriera U&O con modello VG

Tempo complessivo: 25.71s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	22.68376	22.68225	0.66630	[22.67580, 22.69173]
90	13.62008	13.61912	0.70522	[13.61278, 13.62737]
100	5.99456	5.99481	0.40870	[5.98897, 6.00016]
110	1.66463	1.66418	2.71294	[1.66167, 1.66759]
120	0.22174	0.22183	4.00760	[0.22087, 0.22261]

Tempo calcolo funzione caratteristica: 9.18s.
Tempo algoritmo Newton-Raphson: 16.53s.

4.3.2.4 Risultati call lookback a strike fisso

In quest'ultima sezione di risultati riportiamo quelli relativi alle call lookback con strike fisso con 12 date di monitoraggio, confrontate con i prezzi benchmark dati dal metodo Monte Carlo su 10 milioni di simulazioni. Dalle tabelle si nota come la quantizzazione possa portare a risultati imprecisi, tanto da arrivare ad errori relativi dell'ordine di punti percentuale. Ciò è evidente soprattutto nelle Tabelle 4.17 e 4.20, dove i prezzi per strike alti sono ampiamente fuori dall'intervallo di confidenza dei valori di riferimento.

Tabella 4.16: Call lookback con modello Gaussiano

Tempo complessivo: 9.83.s

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	41.70291	41.83005	30.48723	[41.68362, 41.72219]
90	32.19061	32.24242	16.09393	[32.17133, 32.20990]
100	22.67832	22.65479	10.37375	[22.65903, 22.69760]
110	15.63900	15.61706	14.02898	[15.62136, 15.65664]
120	10.52123	10.50140	18.85017	[10.50584, 10.53662]

Tempo calcolo funzione caratteristica: 4.32s.
Tempo algoritmo Newton-Raphson: 5.50s.

Tabella 4.17: Call lookback con modello di Merton

Tempo complessivo: 10.27s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	28.41624	28.45734	14.46527	[28.40992, 28.42256]
90	18.90395	18.90435	0.21167	[18.89763, 18.91026]
100	9.39165	9.35135	42.91531	[9.38533, 9.39797]
110	3.04275	3.00526	123.23590	[3.03842, 3.04708]
120	0.66770	0.63424	501.06466	[0.66564, 0.66976]

Tempo calcolo funzione caratteristica: 4.20s.
Tempo algoritmo Newton-Raphson: 6.07s.

Tabella 4.18: Call lookback con modello di Kou

Tempo complessivo: 7.99s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	33.33576	33.39057	16.44305	[33.32648, 33.34504]
90	23.82346	23.83770	5.97682	[23.81418, 23.83275]
100	14.31117	14.28483	18.40272	[14.30189, 14.32045]
110	7.05648	7.03229	34.28250	[7.04890, 7.06406]
120	2.84801	2.82605	77.10540	[2.84296, 2.85306]

Tempo calcolo funzione caratteristica: 4.16s.
Tempo algoritmo Newton-Raphson: 3.83s.

Tabella 4.19: Call lookback con modello NIG

Tempo complessivo: 15.63s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	34.38035	34.47882	28.64405	[34.36717, 34.39352]
90	24.86805	24.89821	12.12658	[24.85488, 24.88122]
100	15.35576	15.31759	24.85473	[15.34259, 15.36893]
110	8.77504	8.74243	37.16684	[8.76365, 8.78644]
120	4.80904	4.77854	63.43268	[4.79998, 4.81810]

Tempo calcolo funzione caratteristica: 6.43s.
Tempo algoritmo Newton-Raphson: 9.20s.

Tabella 4.20: Call lookback con modello VG

Tempo complessivo: 12.68s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	27.92631	27.93817	4.24599	[27.91992, 27.93270]
90	18.41402	18.39029	12.88524	[18.40762, 18.42041]
100	8.90172	8.84241	66.62898	[8.89533, 8.90812]
110	2.82696	2.77288	191.29385	[2.82255, 2.83136]
120	0.67694	0.62691	738.96333	[0.67469, 0.67919]

Tempo calcolo funzione caratteristica: 4.44s.
Tempo algoritmo Newton-Raphson: 8.23s.

4.3.3 Analisi ulteriori

4.3.3.1 Risultati all'aumentare delle date di monitoraggio

Come mostrano le seguenti tabelle, il metodo può funzionare anche con 52 date di monitoraggio, ovvero con frequenza settimanale su scadenza a 1 anno, ma risulta essere troppo impreciso, oltre che troppo oneroso computazionalmente. Infatti il tempo di calcolo della fattorizzazione di Wiener-Hopf aumenta linearmente rispetto al numero di date, come mostrato in Figura 4.1, e gli errori del calcolo di ogni termine si sommano tra di loro.

Tabella 4.21: Call barriera D&O con modello Gaussiano e 52 date di monitoraggio

Tempo complessivo: 47.61s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	26.12477	26.08929	13.57765	[26.05287, 26.19666]
90	19.57869	19.54946	14.92740	[19.51326, 19.64412]
100	14.19363	14.17234	15.00471	[14.13559, 14.25168]
110	10.01321	9.99527	17.90896	[9.96296, 10.06345]
120	6.90648	6.89360	18.64684	[6.86387, 6.94910]

Tempo calcolo funzione caratteristica: 26.91s.
Tempo algoritmo Newton-Raphson: 20.69s.

Tabella 4.22: Call barriera D&O con modello NIG e 52 date di monitoraggio

Tempo complessivo: 43.88s.

Strike	Monte Carlo	Quantizzazione	Err. rel (bp)	IC 99%
80	24.55648	24.53773	7.63633	[24.50518, 24.60777]
90	16.66961	16.65314	9.87924	[16.62269, 16.71652]
100	10.47656	10.45769	18.01144	[10.43627, 10.51685]
110	6.17474	6.16043	23.17571	[6.14208, 6.20740]
120	3.47531	3.46399	32.58392	[3.44991, 3.50071]

Tempo calcolo funzione caratteristica: 27.30s.
Tempo algoritmo Newton-Raphson: 16.58s.

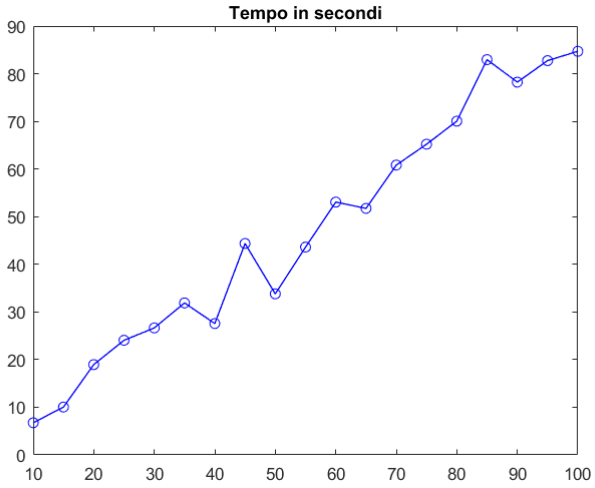
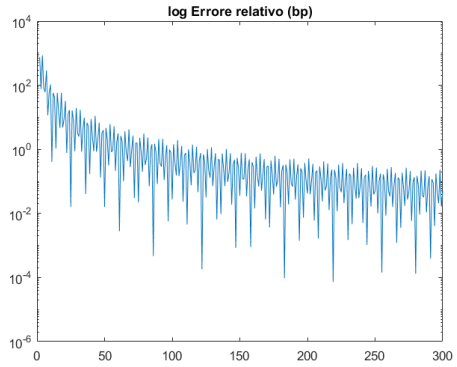


Figura 4.1: Tempo di esecuzione in secondi della fattorizzazione di Wiener-Hopf, al variare del numero di date di monitoraggio, per una call barriera down and out con modello Gaussiano.

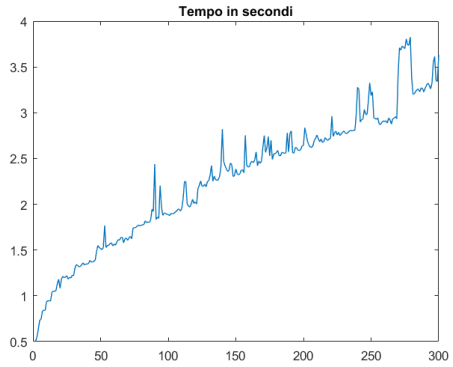
4.3.3.2 Convergenza rispetto alla numerosità del quantizzatore

È interessante studiare la convergenza del prezzo dell'opzione e il tempo richiesto rispetto al numero di punti della griglia di quantizzazione: per fare ciò abbiamo prezzato una call plain vanilla con il modello Gaussiano e una barriera down and out con il modello di Merton. Per il primo derivato abbiamo usato un numero di punti N da 2 a 300, mentre per il secondo da 2 a 100. Riportiamo i risultati nelle Figure 4.2 e 4.3, da cui si evince che la convergenza è veloce in N, mentre il tempo di esecuzione aumenta più o meno linearmente, con alcuni picchi. Il grafico dell'errore sembra suggerire un certo schema di valori per cui l'errore è particolarmente basso, ma in realtà ciò è soltanto dovuto alla definizione del criterio d'arresto dell'algoritmo di Newton Raphson: quando l'errore è poco al di sopra della tolleranza, il passo successivo lo porterà molto al di sotto.

Figura 4.2: Errore e tempo di esecuzione del calcolo di una call plain vanilla con il modello Gaussiano al variare di N.

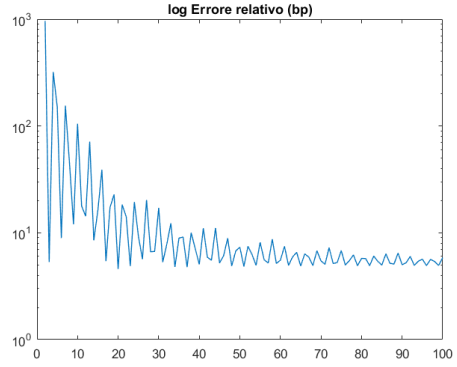


(a) Logaritmo dell'errore relativo espresso in basis point per $N = 2, \ldots, 300$.

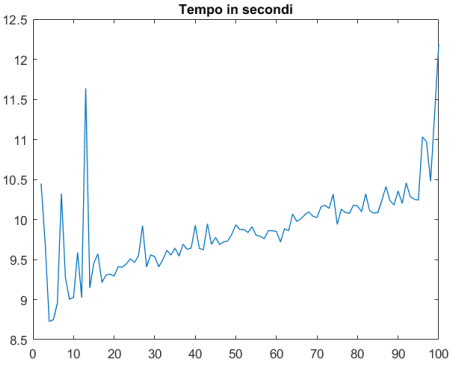


(b) Tempo di esecuzione espresso in secondi per $N = 2, \ldots, 300$.

Figura 4.3: Errore e tempo di esecuzione del calcolo di una barriera down and out con il modello di Merton al variare di N.



(a) Logaritmo dell'errore relativo espresso in basis point per $N = 2, \ldots, 100$.



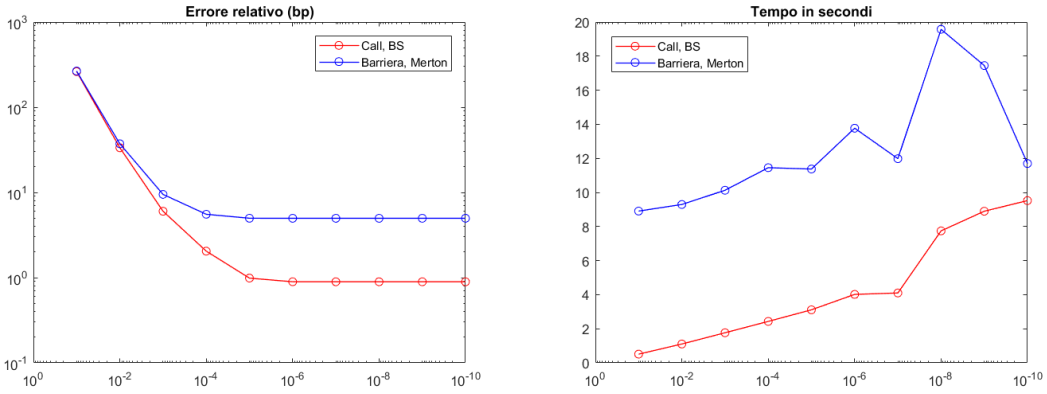
(b) Tempo di esecuzione espresso in secondi per $N = 2, \ldots, 100$.

4.3.3.3 Convergenza rispetto alla tolleranza sul gradiente

Analizziamo ora la convergenza dell'errore e l'andamento del tempo di esecuzione rispetto alla tolleranza ε sul gradiente, che definisce il criterio d'arresto dell'algoritmo di Newton-Raphson. Per farlo abbiamo valutato una call plain vanilla con il modello Gaussiano e una barriera down and out con il modello di Merton, come nell'analisi precedente. Questa volta la numerosità N del quantizzatore è fissata a 100 nel caso della call e 60 nel caso della barriera, mentre per la tolleranza abbiamo

usato $\varepsilon = 10^{-i}$, per $i = 1, \ldots, 10$, in entrambi i casi. Come ci saremmo aspettati, l'errore decresce diminuendo la tolleranza, ma, come si vede in Figura 4.4, esiste un valore oltre la quale il risultato non migliora più.

Figura 4.4: Errore e tempo al variare della tolleranza sul gradiente.



(a) Logaritmo dell'errore relativo espresso in basis point per $\varepsilon = 10^{-1}, \ldots, 10^{-10}$.

(b) Tempo di esecuzione espresso in secondi per $\varepsilon = 10^{-1}, \ldots, 10^{-10}$.

Capitolo 5

Conclusioni

Lo scopo di questa tesi era quello di illustrare il metodo della quantizzazione ed applicarlo al pricing di prodotti derivati, ed in particolare estenderlo a quelli path-dependent. Siamo partiti seguendo [2] per illustrare le principali nozioni riguardanti il metodo e la sua applicazione alle call plain vanilla, come abbiamo visto nei Capitoli 1 e 2. Ci siamo poi chiesti, nel Capitolo 3, se fosse anche possibile applicarlo ai derivati path-dependent, in particolare lookback e barriere: abbiamo visto che grazie alle equazioni di Spitzer è possibile manipolare la funzione caratteristica del sottostante in modo da inglobare l'informazione sui valori estremi assunti durante la vita dell'opzione: per farlo abbiamo sfruttato la semplicità della dipendenza dei payoff rispetto ai valori estremi. Possiamo quindi concludere che è teoricamente possibile prezzare opzioni path-dependent con la quantizzazione, ma ciò risulta molto oneroso dal punto di vista computazionale e numericamente impreciso, come mostrano i risultati.

Bibliografia

[1] Abate, J. & Whitt, W. (1992).Numerical inversion of probability generating functions. *Operations Research Letters, 12,* 245–251.

[2] Callegaro, G., Fiorin, L. & Grasselli, M. (2017). Quantization Meets Fourier: a New Technology for Pricing Options.

[3] Cont, R., & Tankov, P. (2004). Financial modeling with jump processes. *Boca Raton: Chapman & Hall/CRC.*

[4] Feng, L., & Linetsky, V. (2008). Pricing discretely monitored barrier options and defaultable bonds in Lévy process models: a Hilbert transform approach. *Mathematical FInance, 18*(3), 337-384.

[5] Fusai, G., Germano, G. & Marazzina, D. (2015). Spitzer identity, Wiener-Hopf factorization and pricing of discretely monitored exotic options. *European Journal of Operational Research, 251,* 124-134.

[6] Graf, S. & Luschgy, H. (2000). Foundations of quantization for probability distributions. *Springer, New York.*

[7] Green, R., Fusai, G. & Abrahams, I. D. (2010). The wiener-hopf technique and discretely monitored path-dependent option pricing. *Mathematical Finance, 20*(2), 259-288.

[8] King, F. W. (2009). Hilbert transforms. *Cambridge: Cambridge University Press.*

[9] Klenke, A. Probability Theory – A Comprehensive Course, *Springer-Verlag.*

[10] Pagès, G. & Sagna, A. (2015). Recursive marginal quantization of the Euler scheme of a diffusion process. *Applied Mathematical Finance, 22(5)*, 463–498.

[11] Papapantoleon, A. (2008). An Introduction to Lévy Processes With Applications in Finance.

[12] Pierce, J. (1970). Asymptotic quantizing error for unbounded random variables. *IEEE Trans. Inf. Theor.*, 16(1): 81–83.

[13] Shephard, N. G. (1991). From Characteristic Function to Distribution Function: a Simple Framework for the Theory *Econometric Theory, 7*, 519-529.

[14] Spitzer, F. (1956). A combinatorial lemma and its application to probability theory. *Transactions of the American Mathematical Society, 82*(2), 323–339.

[15] Stenger, F. (1993). Numerical methods based on Sinc and analytic functions. *Berlin: Springer.*

Appendice A

Il ruolo del parametro p

La scelta classica in letteratura è quella di un'ottimizzazione quadratica, ovvero porre $p = 2$. Anche nel nostro caso questa scelta semplifica l'implementazione dell'algoritmo, ma è lecito interrogarsi sull'impatto di questo parametro. Presentiamo quindi un esempio tratto da [2] relativo alla distribuzione normale $\mathcal{N}(0, 1)$, con $N = 10$ il numero di punti della griglia. Facendo variare p tra 1 e 15, questo è il risultato:

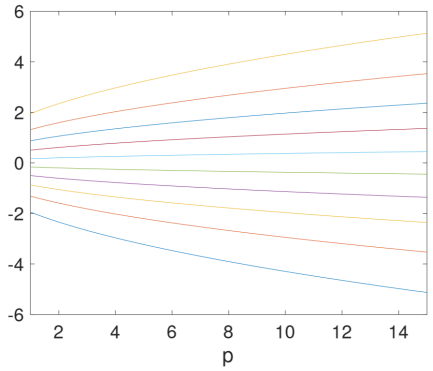
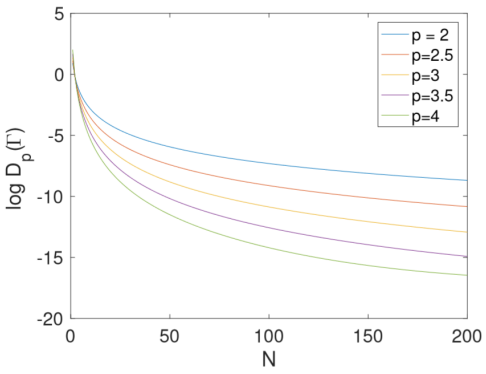
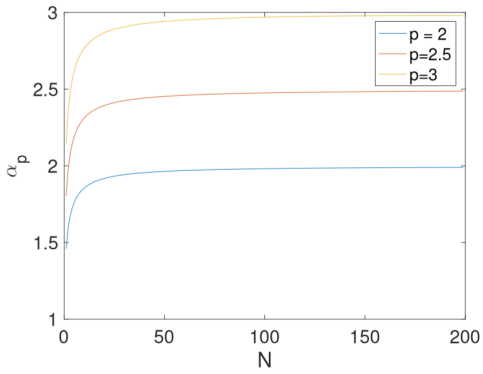


Figura A.1: Griglia ottima relativa a $X \sim \mathcal{N}(0, 1)$ al variare di p, con un numero di punti fissato $N = 10$.

È quindi evidente che al crescere di p la griglia ottima tende ad espandersi, rimanendo centrata sulla media; ovvero la scelta di p dipende dalla rilevanza delle code della distribuzione.
Anche la funzione di distorsione $D_p(\Gamma)$ è influenzata dalla scelta di p. In particolare, come si può notare dalla Figura A.2a, p determina l'andamento di $D_p(\Gamma)$ rispetto a N: maggiore è p e maggiore sarà la velocità di caduta di $D_p(\Gamma)$, soprattutto per N bassi. Nella Figura A.2b è invece illustrato l'andamento della funzione $\alpha_p(N)$, definita dall'espressione $D_p(\Gamma) = \frac{C}{N^{\alpha_p}}$, con C costante. Notiamo che per



(a) Andamento di $\log(D_p(\Gamma))$ al variare di N per diversi valori di p.



(b) Andamento di α_p al variare di N per diversi valori di p.

$N \to +\infty$, il tasso di decrescita α_p tende a p, in linea con i risultati sull'andamento asintotico dell'errore.

Appendice B

Sketch della dimostrazione delle identità di Spitzer

In questa appendice seguiremo il procedimento mostrato in [7] per la dimostrazione delle identità di Spitzer, ma soltanto in modo euristico e non rigoroso. Consideriamo il caso dell'equazione (3.45), riferita alla distribuzione delle traiettorie che stanno sempre sopra una barriera inferiore l. Più genericamente, sia X_n una passeggiata aleatoria monodimensionale, ovvero

$$\begin{cases} X_n = X_{n-1} + Z_n\,, \quad n = 1, 2, \dots \\ X_0 = 0\,, \end{cases} \tag{B.1}$$

dove Z_n sono variabili aleatorie indipendenti e identicamente distribuite con densità k, cioè

$$k(x-y)dx = \mathbb{P}\big(X_{n+1} \in [x, x+dx) \big| X_n = y\big)\,. \tag{B.2}$$

Denotiamo, coerentemente con la trattazione del capitolo 3, con $f_m(x,n)$ la pseudodensità della probabilità che $X_n = x$ e che la traiettoria sia sempre stata sopra la barriera l:

$$f_m(x,n)dx = \mathbb{P}\Big(X_n \in [x, x+dx), \min_{0\le j\le n-1} X_j > l\Big)\,. \tag{B.3}$$

Allora, sfruttando l'indipendenza tra X_n e Z_n, si ha che

$$\begin{cases} f_m(y,n) = \displaystyle\int_l^{+\infty} f_m(x, n-1)k(y-x)dx, \quad n = 1, 2, \dots \\ f_m(x,0) = \delta(x)\,, \end{cases} \tag{B.4}$$

dove δ è la delta di Dirac, con un classico e leggero abuso di notazione. Applichiamo ora la trasformata Zeta alla successione $\big(f_m(y,n)\big)_n$:

$$\tilde{f}(y,q) \coloneqq \mathcal{Z} f_m(y,n) = \sum_{n=0}^{\infty} q^n f_m(y,n) = \delta(y) + q\int_l^{+\infty} \tilde{f}(x,q)k(y-x)dx\,, \tag{B.5}$$

e operando il cambio di variabili $\xi = x - l$, $\nu = y - l$, e definendo $g(\xi) \coloneqq \tilde{f}(\xi + l, q)$, otteniamo:

$$g(\nu) - \delta(\nu + l) = q \int_0^{+\infty} g(\xi) k(\nu - \xi) d\xi \,. \tag{B.6}$$

Definite le semi-trasformate di Fourier $G_-(\xi)$ e $G_+(\xi)$ come

$$G_-(z) \coloneqq \int_{-\infty}^{0} e^{iz\xi} g(\xi) d\xi \,, \tag{B.7}$$

$$G_+(z) \coloneqq \int_0^{+\infty} e^{iz\xi} g(\xi) d\xi \,, \tag{B.8}$$

e indicando con

$$K(z) \coloneqq \int_{-\infty}^{+\infty} e^{iz\xi} k(\xi) d\xi \tag{B.9}$$

la trasformata di Fourier di k, possiamo applicare la trasformata all'equazione (B.6) e applicare il teorema della convoluzione al secondo membro, per ottenere, riordinando i termini:

$$G_+(z)\big(1 - qK(z)\big) = e^{-izl} - G_-(z) \,. \tag{B.10}$$

Assumendo una sufficiente regolarità per g, l'ultima equazione è valida in una striscia orizzontale $\mathcal{S}$ sul piano complesso che comprende l'origine. Applicando la fattorizzazione di Wiener-Hopf alla funzione $L(z) \coloneqq 1 - qK(z) = \Phi_-(z)\Phi_+(z)$ e la decomposizione additiva a $P(z) \coloneqq \frac{e^{-ilz}}{\Phi_-(z)} = P_-(z) + P_+(z)$ si ottiene:

$$G_+(z)\Phi_+(z) - P_+(z) = P_-(z) - \frac{G_-(z)}{\Phi_-(z)} \,. \tag{B.11}$$

Questa equazione uguaglia nella striscia $\mathcal{S}$ una funzione analitica dentro e sopra $\mathcal{S}$ (il membro sinistro) a una funzione analitica dentro e sotto $\mathcal{S}$ (il membro destro), quindi, per prolungamento analitico, si possono estendere le funzioni a tutto il piano complesso. La nuova funzione risulta limitata e quindi costante, per il teorema di Liouville. Siccome per $|z| \to \infty$ essa tende a zero, l'equazione (B.10) dev'essere uguale a zero, per cui possiamo isolare $G_+(z)$:

$$G_+(z) = \frac{P_+(z)}{\Phi_+(z)} \,. \tag{B.12}$$

Infine, ricordando la definizione di g, possiamo calcolare la funzione $f_m(x, n)$ come

$$f_m(x, n) = \mathcal{Z}_n^{-1} g(x - l) = \mathcal{Z}_n^{-1} \mathscr{F}^{-1} e^{ilz} G_+(z) \,, \quad x > l \,, \tag{B.13}$$

che è proprio l'equazione (3.45).
Riassumendo brevemente, per ricavare la distribuzione delle traiettorie di una passeggiata aleatoria che stanno sopra una barriera l

$$f_m(x,n)dx = \mathbb{P}\Big(X_n \in [x, x+dx), \min_{0\leq j\leq n-1} X_j > l\Big), \tag{B.14}$$

si fattorizza dapprima con Wiener-Hopf la funzione $\Phi(z;q) \coloneqq 1 - qK(z) = \Phi_-(z;q)\Phi_+(z;q)$, dove K è la funzione caratteristica degli incrementi, e successivamente si decompone additivamente la funzione $P(z) \coloneqq \frac{e^{-ilz}}{\Phi_-(z)} = P_-(z) + P_+(z)$. Ora è possibile calcolare $f_m(x,n)$ come illustrato nell'equazione (B.13).
Il caso con la barriera superiore è del tutto analogo, mentre quello senza barriera può essere ottenuto facendo tendere l a $-\infty$.

Appendice C

Sketch del calcolo della decomposizione di Wiener-Hopf

In questa appendice seguiamo lo sketch della trattazione di [5] per il calcolo della fattorizzazione di Wiener-Hopf. Il metodo è basato sulla trasformata di Hilbert, che, grazie al teorema della convoluzione, ci permette di ricavare le equazioni di Plemelj-Sokhotsky, come fatto in [8], per esplicitare la decomposizione cercata. Vedremo inoltre come calcolare la trasformata usando le funzioni *seno cardinale*, che definiremo in seguito, come fatto in [15].

Definizione C.1 (Trasformata di Hilbert)**.** Data una funzione $h : \mathbb{R} \to \mathbb{R}$, definiamo la sua *trasformata di Hilbert* la funzione $\mathscr{H}h$, definita come

$$\big(\mathscr{H}h\big)(\xi) = \mathrm{PV}\frac{1}{\pi\xi} * h(\xi) = \mathrm{PV}\frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{h(x)}{\xi - x}dx\,, \tag{C.1}$$

dove $*$ rappresenta la convoluzione e PV il *valore principale di Cauchy*, ovvero

$$\mathrm{PV}\int_{-\infty}^{+\infty}\frac{h(x)}{\xi - x}dx \coloneqq \lim_{\varepsilon\to 0^+}\left(\int_{\xi-1/\varepsilon}^{\xi-\varepsilon}\frac{h(x)}{\xi - x}dx + \int_{\xi+\varepsilon}^{\xi+1/\varepsilon}\frac{h(x)}{\xi - x}dx\right). \tag{C.2}$$

Considerando il teorema della convoluzione, ovvero che date due funzioni f e g si ha

$$\big(\mathscr{F}f\big) * \big(\mathscr{F}g\big) = \mathscr{F}(fg)\,, \tag{C.3}$$

e che, come specificato in [5],

$$\mathrm{PV}\mathscr{F}^{-1}\left(\frac{1}{\pi\xi}\right)(x) = -i\ \mathrm{sign}(x)\,, \tag{C.4}$$

otteniamo

$$i\big(\mathscr{H}\mathscr{F}f\big)(\xi) = \mathscr{F}\Big(\mathrm{sign}(x)f(x)\Big)(\xi)\,. \tag{C.5}$$

Definite delle proiezioni sui due semiassi reali come

$$\mathscr{P}^+ f(x) = \mathbf{1}_{\{x>0\}} f(x) =: f_+(x) \quad \text{e} \tag{C.6}$$
$$\mathscr{P}^- f(x) = \mathbf{1}_{\{x<0\}} f(x) =: f_-(x)\,, \tag{C.7}$$

possiamo sostituire la relazione

$$\operatorname{sign}(x) f(x) = f_+(x) - f_-(x) \tag{C.8}$$

in (C.5) e ricavare l'equazione

$$\big(\mathscr{F} f_+\big)(\xi) - \big(\mathscr{F} f_-\big)(\xi) = i\, \mathscr{H}\Big(\mathscr{F} f\Big)(\xi)\,. \tag{C.9}$$

Sfruttando la linearità della trasformata di Fourier si trova facilmente

$$\big(\mathscr{F} f_+\big)(\xi) + \big(\mathscr{F} f_-\big)(\xi) = \big(\mathscr{F} f\big)(\xi)\,, \tag{C.10}$$

che insieme a C.9 ci permette di scrivere le relazioni di Plemelj-Sokhotsky, per esplicitare le due funzioni che costituiscono la decomposizione di $\mathscr{F} f$:

$$\big(\mathscr{F} f_+\big)(\xi) = \frac{1}{2}\Big(\big(\mathscr{F} f\big)(\xi) + i\big(\mathscr{H}\mathscr{F} f\big)(\xi)\Big)\,, \tag{C.11}$$
$$\big(\mathscr{F} f_-\big)(\xi) = \frac{1}{2}\Big(\big(\mathscr{F} f\big)(\xi) - i\big(\mathscr{H}\mathscr{F} f\big)(\xi)\Big)\,. \tag{C.12}$$

Non ci rimane che calcolare la funzione $\big(\mathscr{H}\mathscr{F} f\big)(\xi)$: una possibilità è fornita dall'equazione C.5, che evidenzia come sia sufficiente il calcolo di una trasformata di Fourier e della sua inversa per ricavare la trasformata di Hilbert; tuttavia questa procedura risulta essere poco accurata, e può essere sostituita da una espansione in serie di funzioni *seno cardinale*, che ne migliora il risultato. Definita la funzione *seno cardinale* come

$$\begin{cases} \operatorname{sinc}(z) = \frac{\sin(z)}{z} & \text{se } z \in \mathbb{C} \setminus \{0\}\,, \\ \operatorname{sinc}(z) = 1 & \text{se } z = 0\,, \end{cases} \tag{C.13}$$

e scelto un passo di discretizzazione $h \in \mathbb{R}^+$, è possibile definire una successione di funzioni *sinc* $\big(S_k\big)_{k\in\mathbb{Z}}$ definite da

$$S_k(z,h) = \operatorname{sinc}\left(\frac{\pi(z-kh)}{h}\right), \tag{C.14}$$

che, come mostrato in [15], possono essere usate per esprimere in serie una funzione f analitica sul piano complesso ed esponenziale[1] di parametro π/h:

$$f(z) = \sum_{k=-\infty}^{+\infty} f(kh) S_k(z,h)\,. \tag{C.15}$$

[1] Una funzione $f : \mathbb{C} \to \mathbb{C}$ si dice *esponenziale di parametro* a se esiste $c \in \mathbb{R}$ tale che $|f(z)| \le c e^{a|z|}$.

Come descritto in [4], si può calcolare la trasformata $\mathscr{H}$ di $S_k(\cdot\,, h)$ come

$$\big(\mathscr{H} S_k\big)(z) = \frac{1 - \cos\big(\pi(z - kh)/h\big)}{\pi(z - kh)/h}\,, \tag{C.16}$$

che, insieme alla C.15, ci dà

$$\big(\mathscr{H} f\big)(z) = \sum_{k=-\infty}^{+\infty} f(kh) \frac{1 - \cos\big(\pi(z - kh)/h\big)}{\pi(z - kh)/h}\,. \tag{C.17}$$

La procedura vista ci permette quindi di calcolare la trasformata di Hilbert di una funzione analitica su tutto il piano complesso, ma può anche essere usata per una funzione analitica soltanto in una striscia che include l'asse reale, che è l'applicazione che ci interessa, come viene descritto in [5]. La convergenza è esponenziale in h e più accurata del metodo suggerito dall'equazione C.5.

Appendice D

Codice MATLAB

LAUNCHER.m

```
% Parametri opzione
paramOpzione.S0 = 100;
paramOpzione.K = 100;
paramOpzione.r = 0.05;
paramOpzione.T = 1;

% Parametri quantizzazione
paramQuant.nVoronoi = 60;
paramQuant.tollGrad = 1e-5;

% Opzioni possibili
% ["call", "lookback", "barriera up and out", "barriera down and out"]
for opzione = ["call"]
    paramOpzione.nome = opzione;
    % Barriere
    if strcmpi(paramOpzione.nome, 'barriera up and out')
        paramOpzione.U = 130;
    elseif strcmpi(paramOpzione.nome, 'barriera down and out')
        paramOpzione.D = 70;
    end
    % Numero date monitoraggio
    if ~strcmpi(paramOpzione.nome, 'call') && ~strcmpi(paramOpzione.nome, 'call Spitzer')
        paramOpzione.nDate = 12;
    end

```

```
    % Modelli possibili = ["BS", "Merton", "Kou", "NIG",
        "VG"]
    for modello = ["BS", "Merton", "Kou", "NIG", "VG"]
        % Parametri modello
        paramModello.nome = modello;
        paramModello = setParametri(paramModello);

        % Quantizzazione
        prezzo = Main(paramOpzione, paramModello,
            paramQuant);
    end
end
```

Main.m

```
function prezzo = Main(paramOpzione, paramModello,
    paramQuant)
%% Parametri
% Parametri opzione
paramOpzione.Keff = paramOpzione.K ./ paramOpzione.S0;
if strcmpi(paramOpzione.nome, 'barriera up and out')
    paramOpzione.Ueff = paramOpzione.U / paramOpzione.S0;
elseif strcmpi(paramOpzione.nome, 'barriera down and out'
    )
    paramOpzione.Deff = paramOpzione.D / paramOpzione.S0;
end

% Parametri quantizzazione
paramQuant.x0Max = 1.2;
paramQuant.xpMax = 10;
if strcmpi(paramOpzione.nome, 'lookback')
    paramQuant.xmMin = 0.9;
    paramQuant.x0Min = 1 + 1e-6;
else
    paramQuant.xmMin = 1e-6;
    paramQuant.x0Min = 0.9;
end
paramQuant.x0 = linspace(paramQuant.x0Min, paramQuant.
    x0Max, paramQuant.nVoronoi);
paramQuant.nMaxIter = 30;
paramQuant.tollGamma = 1e-0;
paramQuant.absTol = 1e-5;
```

```
25 paramQuant.relTol = 1e-5;
26 % Parametri calcolo funzione caratteristica path
      dependent
27 paramFzCaratt.xMax = 30; % estremo densita' per fft
28 paramFzCaratt.N = 2^17;
29 paramFzCaratt.gamma = 6; % accuratezza inversione
      trasformata Z (Abate)
30
31 %% Funzione caratteristica
32 tic
33 [phi, p0] = funzioneCaratteristicaOpzione(paramOpzione,
      paramModello, paramFzCaratt);
34 paramModello.phi = phi;
35 paramModello.p0 = p0;
36 paramQuant.maxInt = 800;
37
38 %% Quantizzazione
39 % Inizio log
40 disp('
      ------------------------------------------------------
      ')
41 fprintf(' %s %s\n', 'Opzione:', upper(paramOpzione.nome))
42 fprintf(' %s %s\n\n', 'Modello:', upper(paramModello.nome
      ))
43 disp(' LOG QUANTIZZAZIONE')
44 % Prezzo quantizzazione
45 [prezzo, xOttimo, nIter]  =  ...
46     callQuantFourier(paramOpzione, paramModello,
         paramQuant);
47 tempo = toc;
48
49 %% Risultati
50 fprintf(' %s %d\n', 'N Voronoi iniziale:', paramQuant.
      nVoronoi)
51 fprintf(' %s %d\n', 'N Voronoi finale:', numel(xOttimo))
52 fprintf(' %s %d\n', 'Massimo integrali:', paramQuant.
      maxInt)
53 fprintf(' %s %d%s %d\n\n', 'N iterazioni:', nIter, ',
      Massimo iterazioni:', paramQuant.nMaxIter)
54 disp(' PREZZO')
55 disp(' Strike  |  Prezzo')
56 disp([paramOpzione.K, prezzo]);
57 fprintf(' Tempo: %f secondi \n', tempo)
```

```
58
59 end
```

setParametri.m

```
1  function modello  =  setParametri(modello)
2  % Parametri processo di Levy X dove S_T = S0*exp(r*T +
       X_T)
3  switch lower(modello.nome)
4      case 'bs'
5          modello.sigma = 0.3;
6
7      case 'merton'
8          modello.sigma = 0.1;
9          modello.lambda = 3;
10         modello.delta = 0.01;
11         modello.mu = 0.01;
12
13     case 'kou'
14         modello.sigma = 0.1;
15         modello.lambda = 3;
16         modello.p = 0.3;
17         modello.lambdaP = 40;
18         modello.lambdaM = 12;
19
20     case 'vg'
21         modello.sigma = 0.1;
22         modello.k = 0.1;
23         modello.theta = 0.04;
24
25     case 'nig'
26         modello.sigma = 0.2;
27         modello.k = 0.1;
28         modello.theta = 0.1;
29
30     otherwise
31         error('Modello non supportato')
32 end
33
34 % Esponente caratteristico risk-neutral
35 modello.psiRiskNeutral = @(u) espCarattRiskNeutral(u,
       modello);
```

```
36
37 end
```

espCarattRiskNeutral.m

```
1  function psi  =  espCarattRiskNeutral(u, modello)
2  % Esponente caratteristico risk-neutral modelli di Levy
3  switch(lower(modello.nome))
4      case 'bs'
5          psi  =  espCarattBS(u, modello) - 1i*u*
               espCarattBS(-1i, modello);
6      case 'merton'
7          psi  =  espCarattMerton(u, modello) - 1i*u*
               espCarattMerton(-1i, modello);
8      case 'kou'
9          psi  =  espCarattKou(u, modello) - 1i*u*
               espCarattKou(-1i, modello);
10     case 'nig'
11         psi  =  espCarattNIG(u, modello) - 1i*u*
               espCarattNIG(-1i, modello);
12     case 'vg'
13         psi  =  espCarattVG(u, modello) - 1i*u*
               espCarattVG(-1i, modello);
14     otherwise
15         error('Modello non supportato')
16 end
17
18 end
19
20 %% Esponenti caratteristici non risk-neutral
21
22 % Black & Scholes
23 function psi = espCarattBS(u, param)
24 sigma = param.sigma;
25
26 psi = -0.5*sigma^2*(1i*u + u.^2);
27 end
28
29 % Merton
30 function psi  =  espCarattMerton(u, param)
31 sigma = param.sigma;
32 lambda = param.lambda;
```

```
delta = param.delta;
mu = param.mu;

psi = -0.5*(sigma*u).^2 + lambda*(exp(-0.5*(delta*u).^2 +
    1i*mu*u) - 1);
end

% Kou
function psi = espCarattKou(u, param)
sigma = param.sigma;
lambda = param.lambda;
p = param.p;
lambdaP = param.lambdaP;
lambdaM = param.lambdaM;

psi = -0.5*sigma^2*u.^2 + 1i*u*lambda.* ( p./(lambdaP - 1
   i*u) - (1 - p)./(lambdaM + 1i*u) );
end

% Normal Inverse Gaussian
function psi  =  espCarattNIG(u, param)
k = param.k;
sigma = param.sigma;
theta = param.theta;

psi = 1/k - 1/k*sqrt(1 + k*(sigma*u).^2 - 2*1i*theta*u*k)
   ;
end

% Variance Gamma
function psi  =  espCarattVG(u, modello)
sigma = modello.sigma;
k = modello.k;
theta = modello.theta;

psi = -1/k * log(1 + u.^2*0.5*sigma^2*k - 1i*theta*k*u);
end
```

funzioneCaratteristicaOpzione.m

```
function [phi, p0] = funzioneCaratteristicaOpzione(
   paramOpzione, paramModello, paramFzCaratt)
```

```
%Calcolo funzione caratteristica
%phi e' la fz. caratt. di log(S_T) = rT + X_T, X Levy, a cui viene
%aggiunta la correzione per le opzioni path dependent

opzione = paramOpzione.nome;
T = paramOpzione.T;
r = paramOpzione.r;

% Vanilla Call
if strcmpi(opzione,'call')
    phi = @(u)exp(1i*u*r*T).*exp(T*paramModello.psiRiskNeutral(u));
    p0 = 0;
    return
end

% Path dependent (o call con Spitzer)
[csi, fzCarattNumerica] = fzCarattPathDep(paramModello, paramOpzione, paramFzCaratt);
if strcmpi(opzione, 'lookback')
        phi = griddedInterpolant(csi, fzCarattNumerica - fzCarattNumerica(end), 'linear', 'none');
        p0 = real(fzCarattNumerica(end));
else
        phi = griddedInterpolant(csi, fzCarattNumerica, 'linear', 'none');
        p0 = real(1 - phi(0));
end

end
```

fzCarattPathDep.m

```
function [csi, fzCarattNumerica] = fzCarattPathDep(paramModello, paramOpzione, paramFzCaratt)
% Calcolo numerico della funzione caratteristica del massimo di rt + X_t,
% dove X Levy.
nDate = paramOpzione.nDate;
T = paramOpzione.T;
r = paramOpzione.r;
```

```
7  xMax = paramFzCaratt.xMax;
8  N = paramFzCaratt.N;
9  gamma = paramFzCaratt.gamma;
10
11 %% Parametri
12 % Parametri temporali
13 dt = T/nDate;
14
15 % Griglia csi spazio Fourier
16 dx = 2*xMax/N;
17 dcsi = 2*pi/(dx*N); % fft
18 csi = dcsi*(-N:N);
19 nCsi = numel(csi);
20
21 % Parametri per inversione trasformata Z (formula di
       Abate)
22 rho = 10^(-gamma/(2*nDate)); % Raggio circonferenza
       inversione Z
23 jj = (0:2*nDate-1)';
24 qq = rho*exp(1i*pi*jj/nDate); % Cambio variabile
       integrale
25
26 %% Spitzer e fattorizzazione di Wiener Hopf
27 % Funzione caratteristica di r*dt + X_dt
28 fzCarattDt = @(csi) exp(dt*(1i*csi*r + paramModello.
       psiRiskNeutral(csi)));
29
30 % Fattorizzazione Spitzer
31 PHI = 1 - repmat(qq,size(csi)).*fzCarattDt(repmat(csi,
       size(qq)));
32 [PHIp, PHIm] = fattorizzazioneWH(PHI);
33
34 %% Inversione trasformata Z con formula Abate
35 % Trasformata Z (Spitzer)
36 switch lower(paramOpzione.nome)
37     case 'call spitzer'
38         ZfzCaratt = 1./PHI;
39
40     case 'lookback' %call fixed strike
41         % Fz caratt massimo
42         ZfzCaratt = 1./(PHIp.*repmat(PHIm(:,(csi == 0))
               ,1,nCsi));
43
```

```
44      case 'barriera up and out'
45          % Fz caratt sottostante, "condizionato" a non
                toccare barriera
46          u = log(paramOpzione.Ueff);
47          Q = repmat(exp(-1i*u*csi), numel(qq), 1)./PHIp;
48          [~, Qm] = fattorizzazioneWH(exp(Q));
49          Qm = log(Qm);
50          ZfzCaratt = repmat(exp(1i*u*csi), numel(qq), 1).*
                Qm./PHIm;
51
52      case 'barriera down and out'
53          % Fz caratt sottostante, "condizionato" a non
                toccare barriera
54          d = log(paramOpzione.Deff);
55          P = repmat(exp(-1i*d*csi), numel(qq), 1)./PHIm;
56          [Pp, ~] = fattorizzazioneWH(exp(P));
57          Pp = log(Pp);
58          ZfzCaratt = repmat(exp(1i*d*csi), numel(qq), 1).*
                Pp./PHIp;
59
60      otherwise
61          error('Tipo opzione non supportato')
62  end
63
64  % Inversione trasformata Z (formula Abate)
65  addendi = repmat((-1).^jj, 1, nCsi) .* ZfzCaratt / 2;
66  fzCarattNumerica = sum(addendi);
67  fzCarattNumerica = fzCarattNumerica/(nDate*rho^nDate);
68  % plot(csi, real(fzCarattNumerica))
69
70  end
```

fattorizzazioneWH.m

```
1  function [Lp,Lm] = fattorizzazioneWH(L)
2  % Ejor Spitzer identity, Wiener-Hopf factorization and
       pricing of
3  % discretely monitored exotic options. Online material
4  lL = log(L);
5  iHlL = ifht(lL); % imaginary unit times fast Hilbert
       transform of L
6  lLp = (lL+iHlL)/2; % Plemelj-Sokhotsky
```

```
lLm = (lL-iHlL)/2; % Plemelj-Sokhotsky
Lp = exp(lLp);
Lm = exp(lLm);
end

% Fast Hilbert transform: Hilbert transform through
% fast Fourier transforms.
function iHF = ifht(F)
% Setup
[M, N] = size(F); % Dimension: number of equations and
   grid points
P = N; % Number of zero padding elements
Q = N+P; % Number of grid points after zero padding
% Deine the auxiliary vector
t = (1-(-1).^(-Q/2:Q/2-1))./(pi*(-Q/2:Q/2-1));
t(Q/2+1) = 0;
vec = repmat(imag(fft(ifftshift(t))),M,1);
% Compute the Hilbert transform times the imaginary unit
f = ifft(F,Q,2); % Optional padding
iHF = fft(vec.*f,[],2);
iHF = iHF(:,1:N);
end
```

callQuantFourier.m

```
function [prezzo, xOttimo, nIter]  =  callQuantFourier(
   paramOpzione, paramModello, paramQuant)
opzione = paramOpzione.nome;
S0 = paramOpzione.S0;
Keff = paramOpzione.Keff;
T = paramOpzione.T;
r = paramOpzione.r;
p0 = paramModello.p0;

% Calcolo griglia quantizzazione ottima con Newton
   Raphson
[xOttimo, xmOttimo, xpOttimo, nIter] = NewtonRaphson(
   paramModello, paramQuant);

% Calcolo probabilita' sottostante appartenga alle celle
   di Voronoi
prob = probT(paramModello, paramQuant, xmOttimo, xpOttimo
   );
```

```
14 if strcmpi(paramOpzione.nome, 'lookback')
15     prob(xOttimo < 1) = 0;
16 end
17 % Payoff atteso scontato
18 prezzo = zeros(size(Keff));
19 for i = 1:numel(Keff)
20     payoff = subplus(xOttimo - Keff(i));
21     prezzo(i) = exp(-r*T)*S0*dot(payoff, prob);
22
23     % Se opzione e' lookback, aggiungo possibilita'
          massimo = S0
24     if strcmp(opzione, 'lookback')
25         prezzo(i) = prezzo(i) + p0*S0*subplus(1 - Keff(i)
              );
26     end
27 end
28
29 % Plot xOttimo
30 %plot(xOttimo, 0, 'b+', 'MarkerSize', 10);
31 end
```

NewtonRaphson.m

```
1  function [gammaNew, xm, xp, nIter] = NewtonRaphson(
       paramModello, paramQuant)
2  gammaOld = paramQuant.x0;
3  xmMin = paramQuant.xmMin;
4  xpMax = paramQuant.xpMax;
5  tollGrad = paramQuant.tollGrad;
6  tollGamma = paramQuant.tollGamma;
7  nMaxIter = paramQuant.nMaxIter;
8
9  % Calcolo griglia ottima di quantizzazione con algoritmo
       di Newton-Raphson
10 k = 0;
11 errGrad = tollGrad + 1;
12 errGamma = tollGamma +1;
13 while (errGrad > tollGrad || errGamma > tollGamma) && (k
       < nMaxIter)
14     [gammaOld, xm, xp] = partizioneVoronoi(gammaOld,
          xmMin, xpMax);
15     H = hessD(paramModello, paramQuant, gammaOld, xm, xp)
          ;
```

```
16      L = gradD(paramModello, paramQuant, gammaOld, xm, xp)
            ;
17
18      gammaNew = sort(gammaOld' - H\L')';
19      errGrad = norm(L, inf);
20      errGamma = norm(gammaOld - gammaNew, inf);
21      gammaOld = gammaNew;
22      k = k + 1;
23  end
24  [gammaNew, xm, xp] = partizioneVoronoi(gammaNew, xmMin,
        xpMax);
25  nIter = k;
26
27  % Check convergenza
28  if k == nMaxIter
29          warning('Raggiunto numero massimo di iterazioni
                in Newton Raphson')
30  end
31
32  % Log
33  fprintf(' %s %.3e\n', 'Max gradiente:', errGrad)
34  fprintf(' %s %.3e\n', 'Max diff griglia precedente:',
        errGamma)
35  end
```

partizioneVoronoi.m

```
1   function [x, xm, xp] = partizioneVoronoi(x, xmMin, xpMax)
2   % Calcolo partizione di Voronoi
3   x(max(x < xmMin, x > xpMax)) = [];
4
5   % Estremi inferiori
6   xm = zeros(size(x));
7   xm(2:end) = (x(1:end-1) + x(2:end))/2;
8   xm(1) = xmMin;
9
10  % Estremi superiori
11  xp = zeros(size(x));
12  xp(1:end-1) = xm(2:end);
13  xp(end) = xpMax;
14  end
```

hessD.m

```
function H  =  hessD(paramModello, paramQuant, x, xm, xp)
phi = paramModello.phi;
maxInt = paramQuant.maxInt;
absTol = paramQuant.absTol;
relTol = paramQuant.relTol;
n = length(x);

% Sovradiagonale (e sottodiagonale)
integranda = @(u)  real(phi(u).*xp(1:end-1).^(-1i*u));
I = integral(integranda, 0, maxInt, 'ArrayValued', true,
    'AbsTol', absTol, 'RelTol', relTol);
diag = (x(1:end-1)-xp(1:end-1)).*I./xp(1:end-1);
diagUp = diag;

% Sottodiagonale
diagDown = diag;

% Diagonale principale
integrandaPrinc = @(u) real(phi(u).*(xm.^(-1i*u) - xp
    .^(-1i*u))./(1i*u));
IPrinc =  integral(integrandaPrinc, 0, maxInt, '
    ArrayValued', true, 'AbsTol', absTol, 'RelTol', relTol
    );
diagPrinc = 2*IPrinc + [diagUp,  0] + [0,  diagDown];

% Matrice hessiana
H = spdiags([[diagDown, 0]', diagPrinc', [0, diagUp]'],
    -1:1, n, n);

% Check convergenza integranda
if  max(abs(integranda(maxInt))) > 1e-2 ||  ...
    max(abs(integrandaPrinc(maxInt))) > 1e-2

    warning('Estremo integrale hessiana')
end
end
```

gradD.m

```
function L = gradD(paramModello, paramQuant, x, xm, xp)
phi = paramModello.phi;
```

```
maxInt = paramQuant.maxInt;
absTol = paramQuant.absTol;
relTol = paramQuant.relTol;

% Calcolo derivata della funzione di distorsione (
   operatore L)
integranda = @(u)  real(phi(u)./(u.*(u + 1i)).*...
          (xp.^(-1i*u).*(-1i*u./(x./xp)+ 1i*u - 1) - xm
             .^(-1i*u).*(1i*u - 1 - 1i*u*(xm./x))));

L = 2*x.*integral(integranda, 0, maxInt, 'ArrayValued',
   true, 'AbsTol', absTol, 'RelTol', relTol);

% Check convergenza integrale
if max(abs(integranda(maxInt))) > 1e-3
    warning('Massimo integrazione gradiente')
end
end
```

probT.m

```
function prob  =  probT(paramModello, paramQuant, xm, xp)
%Calcolo probabilita' sottostante appartenga alle celle
   di Voronoi a
%maturity
phi = paramModello.phi;
p0 = paramModello.p0;
maxInt = paramQuant.maxInt;
absTol = paramQuant.absTol;
relTol = paramQuant.relTol;

integranda = @(u)  real(phi(u).*(xm.^(-1i*u) - xp.^(-1i*u
   )).*(1i*u));
I = integral(integranda, 0, maxInt, 'ArrayValued', true,
   'AbsTol', absTol, 'RelTol', relTol);
prob = I/pi;

% Check
if abs((sum(prob) + p0) - 1) > 1e-2
    warning('Somma probabilita')
end

end
```