Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Matematica

Anno accademico 2023/2024

Sessione di Laurea di Ottobre 2024

Tesi di Laurea Magistrale

Sviluppo di un motore di pricing per derivati esotici tramite metodi Monte Carlo

Relatore

prof. Paolo Brandimarte

Tutor Aziendale

Matteo Rolle

Candidato

Giulio Merlo

Ringraziamenti

Prima di procedere con la trattazione, vorrei dedicare alcune righe a tutte le persone che mi hanno supportato in questo percorso.

Innanzitutto, un ringraziamento va al mio relatore, prof. Paolo Brandimarte, per avermi dato l'opportunità di svolgere questo lavoro di tesi, che ho trovato estremamente interessante e mi ha arricchito profondamente. Desidero inoltre ringraziare il mio tutor aziendale, Matteo Rolle, per la sua indispensabile guida durante lo svolgimento del lavoro e la stesura di questo elaborato. Ringrazio anche Edoardo Siccardi e Alessandro Ruo per la loro immensa disponibilità durante la realizzazione del progetto e il prezioso aiuto fornitomi in numerose occasioni. Un ringraziamento doveroso va a tutti i dipendenti di Sella Financial Markets, per avermi fatto sentire accolto fin dal primo giorno di questa esperienza.

Desidero ringraziare la mia famiglia, mamma, papà, Stefano e Arianna, per aver sempre creduto in me incoraggiandomi nei momenti più difficili e per aver condiviso con gioia ogni passo di questo percorso. Un ringraziamento speciale va a Iris il cui affetto è stato fonte di forza per affrontare tutte le sfide; grazie per esserci sempre stata, per tutto il tempo che mi hai dedicato e per aver fatto la differenza nella mia vita giorno dopo giorno. Infine, un ringraziamento speciale va a tutti i miei amici e ai miei colleghi: grazie per tutte le risate, per tutte le serate assieme, per tutti i caffè alle macchinette del poli, per tutti i momenti vissuti assieme in questi anni; senza di voi non ce l'avrei mai fatta.

Abstract

Il pricing dei derivati è uno dei problemi centrali nell'ambito della finanza quantitativa e riveste un'importanza cruciale nel processo di sintesi di tali strumenti, oltre che nello sviluppo di strategie di trading e nella gestione dei rischi. Nel contesto dei mercati contemporanei, caratterizzati da elevata incertezza, la capacità di stimare accuratamente il prezzo di questi strumenti influisce direttamente sulla competitività dei prodotti finanziari, poiché consente di ottimizzare la progettazione e l'offerta di nuovi titoli. Inoltre, il calcolo del fair value di un derivato fornisce importanti informazioni su variabili non direttamente osservabili, come la volatilità del titolo sottostante.

Questa tesi, di natura sperimentale e condotta in collaborazione con Sella Financial Markets, la divisione mercati del gruppo Sella, ha portato allo sviluppo di un motore di pricing basato su metodi Monte Carlo, con lo specifico obiettivo di valutare strumenti finanziari esotici, come certificati d'investimento e opzioni con multiple barriere. Approcci di tipo Monte Carlo sono particolarmente adatti ad affrontare le sfide del pricing di titoli non standard, in quanto consentono di stimarne il prezzo senza richiedere la conoscenza esplicita della distribuzione di probabilità del sottostante. Tuttavia, la precisione di tali metodi dipende fortemente dalla disponibilità di elevate risorse computazionali, necessarie per garantire una convergenza accurata delle stime.

Per migliorare l'efficacia delle simulazioni Monte Carlo, particolare attenzione è stata dedicata alle tecniche di generazione dei numeri casuali. Sono stati implementati sia metodi di generazione pseudo-casuale che generatori di numeri quasi-casuali basati su sequenze a bassa discrepanza. Questi ultimi si sono rivelati particolarmente vantaggiosi nel migliorare la velocità di convergenza delle simulazioni, riducendo l'errore rispetto ai metodi pseudo-casuali tradizionali e ottimizzando l'efficienza computazionale.

Uno degli aspetti cruciali del lavoro è stato lo sviluppo di un solido modulo software, costruito secondo il paradigma della programmazione a oggetti. Questa scelta ha permesso di creare un'architettura modulare e flessibile, in cui i componenti principali, come i processi stocastici e i generatori di scenari Monte Carlo, possono essere facilmente estesi o modificati

per adattarsi a nuove esigenze. Tale architettura ha garantito la scalabilità del sistema e la possibilità di integrare ulteriori miglioramenti in futuro.

Un altro elemento cardine del motore di pricing sviluppato è la possibilità di impiegare modelli a volatilità stocastica, in particolare il modello di Heston, che consentono di catturare in maniera più accurata le dinamiche di mercato rispetto ai modelli a volatilità costante. La volatilità stocastica riflette meglio le fluttuazioni reali dei mercati, consentendo una modellazione più realistica del comportamento del sottostante e una calibrazione più precisa che garantisce una rappresentazione fedele delle reali condizioni del mercato.

Le funzionalità implementate sono state tutte accuratamente testate, evidenziando la capacità del motore di pricing di produrre stime precise del fair price di derivati esotici anche in contesti di applicazione al mondo reale.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

L'ingegneria finanziaria è la disciplina che utilizza strumenti matematici e statistici per affrontare problemi complessi nel mondo della finanza. La sua rilevanza è cresciuta esponenzialmente negli ultimi anni, in particolare nell'ambito della gestione dei rischi finanziari e nella progettazione di nuovi strumenti. Nel contesto dei mercati contemporanei, questa disciplina è diventata cruciale per gestire i rischi, sviluppare nuovi strumenti finanziari e migliorare l'efficienza operativa delle istituzioni finanziarie.

In questo campo i derivati sono elementi di particolare interesse: essi sono contratti il cui valore dipende dall'andamento di un'attività sottostante, come azioni o tassi d'interesse. Una delle principali sfide legate a tali strumenti è il problema del loro pricing, ovvero la stima del valore che tali strumenti dovrebbero assumere per far sì che questo rispecchi le attuali condizioni del mercato. Per le tipologie più semplici di derivati, esistono formule analitiche ben consolidate che permettono di stimare il loro valore. Tuttavia, queste tecniche si basano su ipotesi generalmente molto semplicistiche per quanto riguarda le dinamiche dell'attività sottostante e queste non sempre rispecchiano fedelmente le complessità e le irregolarità dei mercati finanziari reali. Ciò può portare a significativi errori di stima e limita fortemente l'applicabilità di queste tecniche in determinati contesti. D'altro canto, per i derivati più complessi, detti derivati esotici, non si è in grado di ricavare formule di pricing nemmeno ricorrendo ad assunzioni che vadano a descrivere in maniera semplificata la reale dinamica dei mercati.

In questo lavoro si andrà ad approfondire il problema del pricing dei derivati, focalizzandosi su contratti esotici aventi come sottostanti azioni o indici azionari, e a presentare un approccio di soluzione basato sull'impiego dei celebri metodi di stima Monte Carlo. La tesi è di natura sperimentale e ha previsto lo sviluppo, l'implementazione e la validazione di un motore di pricing in linguaggio Python che andasse a integrarsi all'interno della libreria quantitativa di Sella Financial Markets, la divisione del gruppo Sella responsabile di attività di trading in conto proprio e market-making per numerosi strumenti finanziari. L'aggiunta di tali funzionalità all'insieme degli strumenti quantitativi a disposizione dell'area è di vitale importanza ai fini

di sintetizzare in maniera efficiente prodotti commercialmente competitivi per i clienti della banca e, al tempo stesso, costituisce la base per lo sviluppo di analisi di sensitività più precise che permettano di gestire in maniera assai più efficace i rischi legati alle attività di trading.

Questo lavoro mira a presentare in maniera chiara e puntuale il formalismo necessario a descrivere il problema del pricing di un derivato in maniera astratta per poi approfondire le scelte implementative effettuate ai fini di garantire una perfetta integrazione con l'architettura preesistente all'interno della libreria e conferire al codice un alto livello di riusabilità e adattabilità. Infine, le performance del motore di pricing verranno testate ai fini di validarne il corretto funzionamento anche in contesti di applicazione al mondo reale. Gli aspetti teorici saranno presentati usando il corretto rigore formale, tuttavia, vista la natura sperimentale e applicativa del lavoro, non verranno approfonditi in maniera completamente esaustiva i dettagli più tecnici, per i quali si rimanda ai testi di riferimento per i vari capitoli.

Per poter formalizzare in maniera adeguata il problema del pricing di un derivato è necessario descrivere l'andamento del valore del titolo sottostante e ciò viene effettuato mediante equazioni differenziali stocastiche. Per questo motivo nel Capitolo 1 verranno introdotte le principali famiglie di processi stocastici e caratterizzato il processo di Wiener che gioca un ruolo fondamentale nella definizione dell'integrale stocastico di Itô. Come anticipato in precedenza, il motore sviluppato è in grado di stimare il prezzo di un derivato sfruttando un approccio basato sui metodi Monte Carlo, che vengono presentati nel Capitolo 2 insieme alle tecniche utilizzate per la generazione di numeri casuali e la simulazione di vettori aleatori. Dopodiché, nel Capitolo 3 verranno introdotti i concetti base della finanza quantitativa e descritte le principali tipologie di titoli e strumenti finanziari, con particolare attenzione ai derivati.

Il formalismo e i concetti introdotti in questi primi capitoli verranno applicati al problema del pricing di un derivato nel Capitolo 4, nel quale verranno presentati due modelli di pricing: il celebre modello di Black-Scholes e il modello a volatilità stocastica di Heston. Ciascun modello verrà analizzato nel dettaglio evidenziandone i punti di forza e le principali criticità. Nel Capitolo 5 verrà effettuata una descrizione del funzionamento della libreria quantitativa SFMQuantLib e delle funzionalità implementate che hanno permesso di effettuare il pricing di derivati esotici. Infine, nel Capitolo 6 il corretto funzionamento del motore di pricing viene verificato tramite esperimenti mirati.

Capitolo 1

Equazioni differenziali stocastiche

La trattazione della teoria necessaria per comprendere il funzionamento del motore di pricing sviluppato inizia con la presentazione dei principali concetti relativi ai processi stocastici e al calcolo differenziale stocastico. Questi, insieme alla teoria dei metodi Monte Carlo, costituiscono le fondamenta matematiche per descrivere e affrontare il problema del pricing di un derivato.

Nella Sezione 1.1, dopo aver introdotto il concetto di processo stocastico, ci concentreremo sulle proprietà fondamentali di alcune delle principali famiglie di processi. Nella Sezione 1.2, introdurremo il processo di Wiener, che serve da base per lo sviluppo della teoria dell'integrale di Itô e dei processi di Itô discussa nella Sezione 1.3.

1.1 Processi stocastici

Sia $(\Omega, \mathscr{F}, \mathbb{P})$ uno spazio di probabilità fissato a priori. Definiamo un *processo stocastico* $\{ X(t) : t \in T \}$ come una famiglia di variabili aleatorie indicizzata dal parametro t. L'indice t viene spesso interpretato come tempo, pertanto diciamo che $X(t)$ rappresenta il *valore* del processo al *tempo* t. L'insieme T, noto come *insieme degli indici*, deve essere dotato di una relazione d'ordine totale. Se T è un insieme numerabile, il processo stocastico si dice *a tempo discreto*, mentre, quando T è un intervallo della retta reale, si parla di processo *a tempo continuo*.

Possiamo concepire un processo stocastico come un'entità matematica che descrive l'evoluzione temporale di un determinato fenomeno, come, ad esempio, il numero di individui appartenenti a una popolazione, il numero di guasti di un macchinario, o, come esploreremo in

dettaglio nei capitoli successivi, l'andamento di un titolo in borsa.

La definizione appena fornita, sebbene intuitiva e orientata alle applicazioni pratiche, risulta carente dal punto di vista formale. Per costruire una base teorica solida, è necessario approfondire alcuni aspetti più delicati, partendo da una definizione più rigorosa del concetto di processo stocastico. D'ora in avanti, faremo riferimento ai testi di Baldi (2017) e Pascucci (2008).

Definizione 1.1 (Processo stocastico)**.** Siano dati due spazi misurabili

$$(\Omega \times T, \mathscr{F} \otimes \mathscr{T}) \qquad \text{e} \qquad (E, \mathscr{E}),$$

dove $\mathscr{F} \otimes \mathscr{T} = \sigma\{A \times B : A \in \mathscr{F}, B \in \mathscr{T}\}$ denota la σ-algebra prodotto. Una funzione $\mathscr{F} \otimes \mathscr{T}$-misurabile

$$X : (\Omega \times T, \mathscr{F} \otimes \mathscr{T}) \to (E, \mathscr{E})$$

è detta *processo stocastico*. Inoltre, se $X(t) \in L^1(\Omega, \mathscr{F}, \mathbb{P})$ per ogni $t \in T$, si dice che X è un processo stocastico *sommabile*.

Osserviamo che la Definizione 1.1 è in linea con la caratterizzazione di processo stocastico data in precedenza: fissato un qualsiasi tempo $t \in T$, $X(t, \omega)$ è $\mathscr{F}$-misurabile, quindi è una variabile aleatoria. Tuttavia, è possibile costruire famiglie indicizzate di variabili aleatorie che non costituiscono processi stocastici; basta considerare una funzione $X(t, \omega) = f(t)$ con f $\mathscr{F}$-misurabile, ma non $\mathscr{F} \otimes \mathscr{T}$-misurabile (ossia f non è $\mathscr{T}$-misurabile). D'ora in avanti, chiameremo gli elementi $\omega \in \Omega$ *scenari*, mentre denomineremo *traiettoria* $(X_\omega(t))$ la funzione (deterministica) ottenuta fissando un singolo scenario

$$X_\omega : T \to E \quad \text{tale che} \quad X_\omega(t) = X(t, \omega), \qquad \omega \in \Omega.$$

Infine, indicheremo con $X(t)$ la variabile aleatoria che descrive il valore del processo stocastico al tempo t.

Come già detto, il nostro interesse per i processi stocastici nasce dalla necessità di rappresentare l'evoluzione temporale di fenomeni intrinsecamente aleatori. Tuttavia, per descrivere tali fenomeni in maniera completa, è essenziale considerare il flusso di informazioni a cui siamo esposti durante la realizzazione del processo. Dal punto di vista formale, ciò viene rappresentato come segue.

Definizione 1.2 (Filtrazione)**.** Dato uno spazio di probabilità $(\Omega, \mathscr{F}, \mathbb{P})$, la collezione $\{\mathscr{F}_t\}_{t \in T}$ è detta *filtrazione* se:

- $\forall t \in T$, $\mathscr{F}_t \subseteq \mathscr{F}$ ($\mathscr{F}_t$ è una sotto-σ-algebra di $\mathscr{F}$),
- $\forall s,t \in T$, $s < t \implies \mathscr{F}_s \subseteq \mathscr{F}_t$.

Se $\{\mathscr{F}_t\}_{t\in T}$ è una filtrazione, lo spazio $(\Omega, \mathscr{F}, \mathbb{P}, \{\mathscr{F}_t\}_{t\in T})$ è detto *spazio di probabilità filtrato*.

In altre parole, una filtrazione è una collezione crescente di sotto-σ-algebre. Dato un processo stocastico $X(t,\omega)$, possiamo definirne la *filtrazione naturale* come la famiglia $\{\mathscr{F}_t\}_{t\in T}$ costituita dalle σ-algebre generate dal processo al variare del tempo t, ossia

$$\mathscr{F}_t = \sigma\{X(s) : s \leq t\}.$$

Definizione 1.3. Dato uno spazio di probabilità filtrato $(\Omega, \mathscr{F}, \mathbb{P}, \{\mathscr{F}_t\}_{t\in T})$, un processo stocastico $X(t,\omega)$ si dice *adattato* se $X(t)$ è $\mathscr{F}_t$-misurabile $\forall t \in T$.

Da questo momento, lavoreremo esclusivamente con processi adattati. Considerare un processo stocastico adattato in uno spazio di probabilità filtrato significa descrivere l'evoluzione di un fenomeno aleatorio dal punto di vista di un osservatore che ne osserva l'evoluzione "progressivamente" e dispone di tutte le informazioni necessarie a stabilire quali eventi siano avvenuti e quali no. In effetti, a un qualsiasi istante $t \in T$, gli elementi della filtrazione $\mathscr{F}_t$ descrivono il passato del processo e gli eventi osservabili (anche con probabilità nulla) nel presente.

Consideriamo ora un processo stocastico a tempo continuo a valori reali

$$X : (\Omega \times (0,+\infty), \mathscr{F} \otimes \mathscr{B}_{(0,+\infty)}) \to (\mathbb{R}, \mathscr{B}),$$

dove $\mathscr{B}$ denota la σ-algebra di Borel-Lebesgue. Definiamo l'*incremento del processo stocastico* sull'intervallo $(a,b) \subset (0,+\infty)$ come la variabile aleatoria

$$\Delta X_{a,b} = \Delta X_{a,b}(\omega) = X(b,\omega) - X(a,\omega) = X(b) - X(a),$$

e diamo le seguenti definizioni:

- il processo X ha *incrementi indipendenti* se, dati $(q,r),(s,t) \subset (0,+\infty)$ tali che $(q,r) \cap (s,t) = \emptyset$, si ha che $\Delta X_{q,r} \perp\!\!\!\perp \Delta X_{s,t}$ (gli incrementi sono variabili aleatorie indipendenti),
- il processo X ha *incrementi stazionari* se, dati $s,t \in T$, $s \neq t$ e un valore $h > 0$, si ha che $\Delta X_{t,t+h}$ e $\Delta X_{s,s+h}$ hanno la stessa distribuzione.

Lo studio degli incrementi di un processo stocastico fornisce informazioni preziose sulle sue dinamiche. Inoltre, gli incrementi sono fondamentali per la definizione del concetto di variazione, che, come vedremo, sarà cruciale per il calcolo differenziale stocastico secondo Itô.

Prima di passare all'analisi di alcune famiglie di processi stocastici con rilevanti applicazioni finanziarie, proponiamo un esempio.

Esempio 1.1. Consideriamo il processo stocastico

$$Y : (\Omega \times (0,+\infty), \mathscr{F} \otimes \mathscr{B}_{(0,+\infty)}) \to (\mathbb{R}, \mathscr{B}),$$

definito in modo che, per ogni istante di tempo t, $Y(t)$ sia una variabile aleatoria gaussiana standard, ovvero

$$Y(t) \sim \mathscr{N}(0,1) \qquad \forall t \in (0,+\infty),$$

e, dati $s,t \in T$, $s \neq t \implies Y(s) \perp\!\!\!\perp Y(t)$. Un tale processo viene comunemente chiamato *rumore bianco* o *white noise* e si indica con la notazione $\mathrm{WN}(\sigma^2)$, dove σ^2 è la varianza delle variabili aleatorie che costituiscono il processo (in questo caso consideriamo un $\mathrm{WN}(1)$). In generale, un processo stocastico composto da variabili aleatorie gaussiane si dice *processo gaussiano*. Il processo $\{Y(t)\}_{t\in(0,+\infty)}$ è ampiamente utilizzato in contesti come l'analisi delle serie temporali. Ricordando la definizione del momento di ordine p di una variabile aleatoria

$$\mathbb{E}[X^p] = \int_\Omega X^p \mathbb{P},$$

e le proprietà della distribuzione gaussiana, è facile verificare che $\forall t \in (0,+\infty)$

$$\begin{aligned} \mathbb{E}[Y(t)] &= 0, \\ \mathrm{Var}(Y(t)) = \mathbb{E}\left[Y(t)^2\right] - \mathbb{E}[Y(t)]^2 &= 1. \end{aligned}$$

Studiamo ora gli incrementi del processo $\mathrm{WN}(1)$. Consideriamo la proprietà della distribuzione gaussiana, per cui, dati $s,t \in (0,+\infty)$

$$Y(s) \perp\!\!\!\perp Y(t) \iff \mathrm{Cov}(Y(s),Y(t)) = 0, \tag{1.1}$$

e definiamo il vettore aleatorio

$$\mathbf{Y} = \begin{pmatrix} Y(q) \\ Y(r) \\ Y(s) \\ Y(t) \end{pmatrix} \sim \mathscr{N}_4(\mathbf{0}_4, \mathbf{I}_4),$$

dove $q, r, s, t \in (0, +\infty)$ sono tali che $(q, r) \cap (s, t) = \emptyset$. Definiamo $\mathbf{A} \in \mathbb{R}^{2\times 4}$ come

$$\mathbf{A} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

Consideriamo ora il vettore aleatorio degli incrementi come la trasformazione lineare del vettore $\mathbf{Y}$ definita da

$$\mathbf{Z} = \mathbf{AY} = \begin{pmatrix} Y(r) - Y(q) \\ Y(t) - Y(s) \end{pmatrix} = \begin{pmatrix} \Delta Y_{q,r} \\ \Delta Y_{s,t} \end{pmatrix},$$

e, utilizzando le proprietà della distribuzione gaussiana, è facile concludere che

$$\mathbf{Z} \sim \mathcal{N}_2 \left(\mathbf{A0}_4, \mathbf{AI}_4\mathbf{A}^\intercal \right),$$

dove

$$\begin{aligned} \mathbf{A0}_4 &= \mathbf{0}_2, \\ \mathbf{AI}_4\mathbf{A}^\intercal &= 2\mathbf{I}_2. \end{aligned}$$

Questo, insieme all'equivalenza espressa in (1.1), permette di concludere che

$$\Delta Y_{q,r}, \Delta Y_{s,t} \quad \text{sono i.i.d. e seguono una distribuzione} \quad \mathcal{N}(0, 2),$$

e, di conseguenza, che il processo $\mathrm{WN}(1)$ ha incrementi indipendenti e stazionari.

1.1.1 Processi markoviani

Quando si desidera descrivere l'evoluzione di un fenomeno, è naturale scegliere un modello che includa una relazione che esprima come il futuro di questo dipenda dal suo passato. In altre parole, ci aspettiamo che la distribuzione della variabile aleatoria che rappresenta il valore di un processo stocastico (adattato) X al tempo t dipenda dalle informazioni relative al passato del processo, contenute nella filtrazione $\mathcal{F}_t$, come introdotto in precedenza.

In generale, questa relazione può essere estremamente complessa, rendendo il modello versatile e adatto a descrivere una grande varietà di fenomeni. Tuttavia, ciò complica la caratterizzazione delle distribuzioni coinvolte e limita la capacità predittiva del modello. D'altro canto, processi semplici, come quello descritto nell'Esempio 1.1, sono facilmente descrivibili, il processo di stima dei parametri è immediato e la simulazione delle traiettorie è semplice ed efficiente. Tuttavia, tali processi non sono adeguati per rappresentare l'evoluzione di fenomeni

complessi, a causa della loro semplicità.

Diventa dunque necessario trovare un buon equilibrio tra queste caratteristiche, ossia individuare classi di processi stocastici che permettano di rappresentare una struttura di dipendenza tra passato e futuro senza complicare eccessivamente la stima dei parametri e la simulazione. A tal fine, introduciamo la seguente famiglia di processi stocastici.

Definizione 1.4 (Processo markoviano)**.** Un processo stocastico X è detto *processo markoviano* se soddisfa la seguente proprietà: dati $s,t \in T$, con $s < t$

$$\mathbb{P}(X(t) \in A|\mathscr{F}_s) = \mathbb{P}(X(t) \in A|X(s)) \quad \text{quasi certamente.} \tag{1.2}$$

Questa è nota come *proprietà di Markov*.

In altre parole, la distribuzione di probabilità del futuro di un processo stocastico dipende esclusivamente dal suo valore nel presente ed è indipendente dal passato. Ciò implica che il processo è "privo di memoria" e, una volta raggiunto un determinato stato, dimentica immediatamente il percorso compiuto per arrivarvi. Questa caratteristica, oltre a garantire una struttura di dipendenza semplice tra le variabili aleatorie, è strettamente collegata a determinati aspetti del comportamento dei titoli in un mercato efficiente.

Dato un processo stocastico markoviano, è possibile definire le *probabilità di transizione* dal tempo s al tempo t come le quantità $P_{s,t}(A,x)$ tali che

$$P_{s,t}(A,x) = \mathbb{P}(X(t) \in A|X(s) = x),$$

dove richiediamo che $s,t \in T$ e $s < t$. Inoltre, se il processo X è sommabile, sotto le stesse ipotesi per s e t si ha che

$$\mathbb{E}[X(t)|\mathscr{F}_s] = \mathbb{E}[X(t)|X(s)],$$

dove $\mathbb{E}[X(t)|\mathscr{F}_s]$ rappresenta l'attesa condizionata alla σ-algebra $\mathscr{F}_s$, ossia il valore atteso di $X(t)$ dato il flusso di informazioni osservato fino al tempo s, mentre $\mathbb{E}[X(t)|X(s)]$ denota l'attesa condizionata alla σ-algebra generata da $X(s)$. Poiché supponiamo che il processo stocastico sia adattato, abbiamo che

$$\sigma(X(s)) \subseteq \mathscr{F}_s,$$

ma le attese condizionate risultano uguali grazie alla proprietà di Markov, che implica che le uniche informazioni rilevanti disponibili al tempo s per determinare il futuro $X(t)$ sono quelle relative al valore assunto dal processo al tempo s.

Un processo markoviano che assume valori in un insieme finito o numerabile viene detto *catena di Markov*. Tali processi trovano applicazione in molteplici campi, dalla teoria dei controlli automatici, alla teoria delle code, allo studio delle dinamiche di popolazione, grazie alla loro semplicità unita a risultati teorici consolidati (distribuzioni asintotiche, caratterizzazione degli intertempi tra eventi successivi, ecc.) che consentono di effettuare analisi approfondite sui sistemi modellati.

1.1.2 Martingale

Introduciamo ora una famiglia di processi stocastici di estrema rilevanza sia in ambito matematico-teorico che in ambito finanziario, grazie alle loro proprietà distintive.

Definizione 1.5. Sia M un processo stocastico sommabile e $\{\mathscr{F}_t\}_{t\in T}$ adattato. Diciamo che M è:

- una *martingala* rispetto a $\{\mathscr{F}_t\}_{t\in T}$ e alla misura $\mathbb{P}$ se

$$M(s) = \mathbb{E}[M(t)|\mathscr{F}_s] \qquad \forall s \leq t,$$

- una *super-martingala* se

$$M(s) \geq \mathbb{E}[M(t)|\mathscr{F}_s] \qquad \forall s \leq t,$$

- una *sub-martingala* se

$$M(s) \leq \mathbb{E}[M(t)|\mathscr{F}_s] \qquad \forall s \leq t.$$

Ricordando che

$$\mathbb{E}[M(t)] = \mathbb{E}[\mathbb{E}[M(t)|\mathscr{F}_s]],$$

per le proprietà dell'attesa condizionata a una σ-algebra, si ottiene che se M è una martingala allora ha media costante:

$$\mathbb{E}[M(t)] = \mathbb{E}[\mathbb{E}[M(t)|\mathscr{F}_s]] = \mathbb{E}[M(s)] \qquad \forall s \leq t.$$

Questa proprietà rende le martingale particolarmente adatte a rappresentare situazioni in cui si ha un "gioco equo" e ci si aspetta che il valore futuro del processo sia mediamente uguale a quello osservato attualmente. Proprio per questa caratteristica, le martingale sono uno strumento chiave nell'ambito del pricing dei derivati, in quanto, come approfondiremo

nel capitolo Capitolo 4, assumeremo che il valore attuale netto di un titolo sia una martingala rispetto a una misura di probabilità $\mathbb{Q}$ adeguata.

Lo studio delle martingale riveste un'importanza fondamentale per via dell'esistenza di numerosi risultati di convergenza, che le rendono estremamente utili per l'analisi e la caratterizzazione dei processi stocastici. La trattazione qui esposta rappresenta una breve introduzione all'argomento e non ne approfondisce tutti gli aspetti.

1.2 Il processo di Wiener

Passiamo ora alla caratterizzazione di un particolare processo a tempo continuo che riveste un ruolo fondamentale nella teoria del calcolo differenziale stocastico. Sebbene dal punto di vista formale esso sia il risultato di un teorema molto tecnico e difficile da dimostrare, è possibile fornirne una descrizione euristica a partire da un processo stocastico molto semplice, noto come *random walk* o passeggiata aleatoria.

1.2.1 Costruzione euristica

Consideriamo le variabili aleatorie i.i.d. $R_1, \dots, R_n$ tali che

$$R_i = \begin{cases} +1 & \text{con probabilità } \frac{1}{2}, \\ -1 & \text{con probabilità } \frac{1}{2} \end{cases} \qquad i = 1, \dots, n.$$

Si verifica facilmente che:

$$\begin{aligned} \mathbb{E}[R_i] &= 0, \\ \operatorname{Var}(R_i) &= 1, \\ \operatorname{Cov}(R_i, R_j) &= 0 \qquad \forall i \neq j. \end{aligned}$$

Sia ora $S_n : \Omega \times \left\{ 0, t \cdot \frac{1}{n}, t \cdot \frac{2}{n}, \dots, t \cdot \frac{n-1}{n}, t \right\} \to \mathbb{R}$ il processo stocastico definito in maniera tale che:

$$\begin{cases} S_n(0) = 0, \\ S_n(k) = \sqrt{\frac{t}{n}} \left(\sum_{i=1}^{\frac{nk}{t}} R_i \right) \quad k \in \left\{ t \cdot \frac{1}{n}, t \cdot \frac{2}{n}, \dots, t \cdot \frac{n-1}{n}, t \right\}. \end{cases}$$

Possiamo pensare a questo processo come al moto di una particella in uno spazio monodimensionale. Al tempo $k = 0$, la particella si trova nel punto di coordinata 0, e alla fine di ogni

intervallo di tempo $(k, k+\frac{t}{n})$, essa compie uno spostamento di ampiezza $\sqrt{\frac{t}{n}}$ in una direzione scelta casualmente con uguale probabilità. Il parametro n rappresenta la risoluzione, poiché, all'aumentare di n, si ottiene una descrizione più dettagliata del moto della particella. Infine, t è un istante di tempo fissato.

Questo processo permette di esaminare come cambia la distribuzione di probabilità della posizione della particella al tempo t al variare della risoluzione temporale con cui ne descriviamo il movimento.

Una semplice analisi di questo processo rivela che:

$$\mathbb{E}[S_n(k)] = \mathbb{E}\left[\sqrt{\frac{t}{n}}\left(\sum_{i=1}^{\frac{nk}{t}} R_i\right)\right] = \sqrt{\frac{t}{n}}\sum_{i=1}^{\frac{nk}{t}} \mathbb{E}[R_i] = 0,$$

$$\mathrm{Var}(S_n(k)) = \mathrm{Var}\left(\sqrt{\frac{t}{n}}\left(\sum_{i=1}^{\frac{nk}{t}} R_i\right)\right) = \frac{t}{n}\left(\sum_{i=1}^{\frac{nk}{t}} \mathrm{Var}(R_i)\right) + \frac{t}{n}\left(\sum_{i\neq j} \mathrm{Cov}\left(R_i, R_j\right)\right) = \frac{t}{n}\cdot\frac{nk}{t} = k.$$

È interessante notare che i momenti principali delle variabili aleatorie $S_n(k)$ non dipendono dal parametro di risoluzione n. Inoltre, S_n è un processo markoviano poiché il valore del processo al tempo $k+\frac{t}{n}$ dipende esclusivamente dal suo valore al tempo k, e non dal percorso compiuto precedentemente. Infine, si ha che, dati $k, h \in \left\{\frac{1}{n}, \frac{2}{n}, \ldots, t\cdot\frac{n-1}{n}, t\right\}$ con $h < k$,

$$\mathbb{E}[S_n(k)|\mathscr{F}_h] = \mathbb{E}\left[\left(\sum_{i=\frac{nh}{t}+1}^{\frac{nk}{t}} \sqrt{\frac{t}{n}} R_i\right) + S_n(h)\middle|\mathscr{F}_h\right] =$$

$$= \sqrt{\frac{t}{n}}\left(\sum_{i=\frac{nh}{t}+1}^{\frac{nk}{t}} \mathbb{E}[R_i|\mathscr{F}_h]\right) + \mathbb{E}[S_n(h)|\mathscr{F}_h] = 0 + S_n(h) = S_n(h),$$

ovvero, S_n è una martingala e $S_n(k) - S_n(h) \perp\!\!\!\perp S_n(h)$, da cui si deduce che il processo ha incrementi indipendenti[1].

Osserviamo ora cosa accade alla distribuzione della posizione finale della particella al crescere della risoluzione temporale:

$$S_n(t) = \sqrt{\frac{t}{n}}\left(\sum_{i=1}^{n} R_i\right) = \sqrt{t}\cdot\sqrt{n}\left(\frac{\sum_{i=1}^{n} R_i}{n}\right) \xrightarrow[n\to+\infty]{\mathscr{L}} \mathscr{N}(0,t),$$

[1] È possibile mostrare che il processo ha anche incrementi stazionari, poiché $\mathbb{E}[S_n(k) - S_n(h)] = 0$ e $\mathrm{Var}(S_n(k) - S_n(h)) = (k-h)$, e la distribuzione dell'incremento dipende solo dalla lunghezza dell'intervallo di tempo.

per effetto del *teorema limite centrale*. Ciò suggerisce la costruzione di un nuovo processo stocastico a tempo continuo $W : \Omega \times (0, +\infty) \to \mathbb{R}$ tale che per ogni istante t fissato si abbia:

$$W(t) \overset{\mathscr{L}}{=} \lim_{n \to +\infty} S_n(t).$$

Questo procedimento, sebbene non costituisca una dimostrazione formale dell'esistenza di un tale processo, è utile per comprendere meglio la definizione formale che verrà presentata e fornisce un'interpretazione "fisico-meccanica" di alcune sue caratteristiche. Secondo il ragionamento appena esposto, W descrive il moto unidimensionale di una particella, quindi è lecito aspettarsi che, fissato uno scenario ω, la relativa traiettoria $W_\omega(t)$ sia una funzione continua nel tempo, poiché il movimento descritto è costituito da contributi sempre più piccoli, che vanno a approssimare un fenomeno continuo. Le caratteristiche del processo S_n sono indipendenti dal parametro di risoluzione n, il che suggerisce che anche il processo limite sarà una martingala dotata della proprietà di Markov e che i momenti delle variabili $W(t)$ saranno:

$$\begin{aligned} \mathbb{E}[W(t)] &= 0, \\ \mathrm{Var}(W(t)) &= t. \end{aligned}$$

1.2.2 Definizione formale e proprietà

Una descrizione formalmente accurata del processo stocastico descritto in precedenza fu formulata da Wiener nel 1918. Per questo motivo tale processo è conosciuto come *processo di Wiener*. È anche chiamato *moto browniano* in onore del botanico Robert Brown, il quale, nel 1827, fu il primo a descrivere euristicamente un movimento con caratteristiche analoghe a quelle del processo S_n, osservando al microscopio il comportamento di piccole particelle di polline e pietra lavica.

Teorema 1.1. *Sia* $\left(\Omega, \mathscr{F}, \mathbb{P}, \{\mathscr{F}_t\}_{t \geq 0}\right)$ *uno spazio di probabilità filtrato. Esiste un processo stocastico* $W : \Omega \times (0, +\infty) \to \mathbb{R}$ *tale che:*

i) $W(0) = 0$ *quasi certamente,*

ii) W *è adattato a* $\{\mathscr{F}_t\}_{t \geq 0}$ *e le sue traiettorie* $W_\omega(t)$ *sono continue quasi certamente,*

iii) dati $t > s \geq 0$*, l'incremento del processo* $\Delta W_{s,t}$ *ha distribuzione normale con media* 0 *e varianza* $t - s$*, ed è indipendente da* $\mathscr{F}_s$*.*

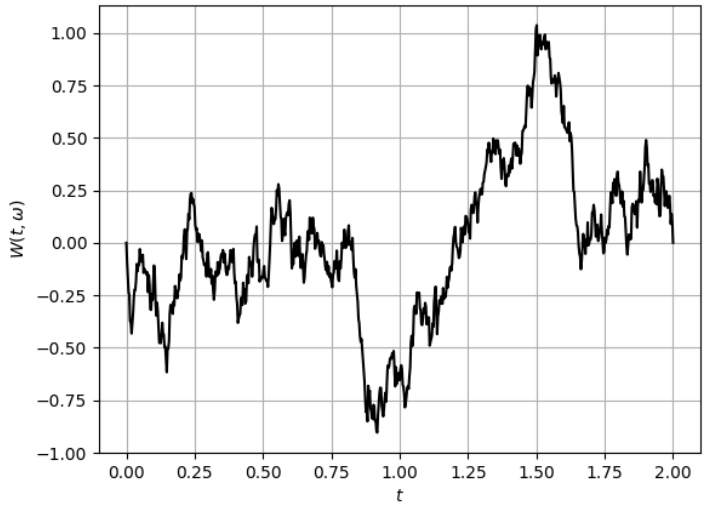

Figura 1.1 Il grafico di una traiettoria del processo di Wiener generata mediante le funzionalità implementate in *SFMQuantLib*.

Osservazione 1.2. *La proprietà i) insieme alla proprietà iii) implicano che:*

$$W(t) \sim \mathcal{N}(0,t),$$

infatti si ha che $W(t) = W(t) - 0 = W(t) - W(0) = \Delta W_{0,t}$.

Utilizzando la caratterizzazione del Teorema 1.1, è possibile costruire un algoritmo per generare approssimazioni delle traiettorie di un processo di Wiener, ottenendo risultati come quello mostrato in Fig. 1.1.

Nella restante parte di questa sezione dimostreremo alcune proprietà del moto Browniano.

Proposizione 1.3. *Dati* $s,t \in (0,+\infty)$*, si ha che:*

$$\mathrm{Cov}\left(W(s),W(t)\right) = \min\left(s,t\right),$$

e, conseguentemente:

$$\rho\left(W(s),W(t)\right) = \sqrt{\frac{\min\left(s,t\right)}{\max\left(s,t\right)}},$$

dove ρ *indica la correlazione di Pearson:* $\rho\left(X,Y\right) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\cdot\mathrm{Var}(Y)}}$.

Dimostrazione. Ricordando che $\mathbb{E}[W(t)] = \mathbb{E}[W(s)] = 0$, si ha che, supponendo $t \geq s$,

$$\begin{aligned}
\mathrm{Cov}(W(s), W(t)) &= \mathbb{E}[W(t) \cdot W(s)] - \mathbb{E}[W(t)] \cdot \mathbb{E}[W(s)] \\
&= \mathbb{E}[[(W(t) - W(s)) + W(s)] \cdot W(s)] - 0 \\
&= \mathbb{E}[(W(t) - W(s)) \cdot W(s)] + \mathbb{E}\left[W(s)^2\right] \\
&= \mathbb{E}[\Delta W_{s,t} \cdot W(s)] + \mathrm{Var}(W(s)) \\
&= 0 + s = s = \min(t, s),
\end{aligned}$$

dove al penultimo passaggio abbiamo sfruttato l'indipendenza dell'incremento da $W(s)$. Applicando la definizione di correlazione di Pearson, si ottiene:

$$\rho(W(s), W(t)) = \frac{\mathrm{Cov}(W(s), W(t))}{\sqrt{t \cdot s}} = \frac{\min(t, s)}{\sqrt{t \cdot s}},$$

e, supponendo $t \geq s$, si ha:

$$\rho(W(s), W(t)) = \frac{s}{\sqrt{t \cdot s}} = \sqrt{\frac{s}{t}} = \sqrt{\frac{\min(s, t)}{\max(s, t)}}.$$

□

Proposizione 1.4. *Il processo di Wiener è un processo markoviano.*

Dimostrazione. Questa proprietà discende direttamente dal Teorema 1.1, infatti, fissati $s, t \in (0, +\infty)$ con $s < t$ e $a, b \in \mathbb{R}$, $a < b$, si ha che:

$$\begin{aligned}
\mathbb{P}(W(t) \in (a, b) | \mathscr{F}_s) &= \mathbb{P}(\Delta W_{s,t} + W(s) \in (a, b) | \mathscr{F}_s) \\
&= \mathbb{P}(\Delta W_{s,t} + W(s) \in (a, b) | W(s)) \\
&= \mathbb{P}(W(t) \in (a, b) | W(s)),
\end{aligned}$$

dove al secondo passaggio abbiamo usato l'indipendenza dell'incremento $\Delta W_{s,t}$ rispetto a $\mathscr{F}_s$, implicando che l'unica informazione necessaria per calcolare tale probabilità è il valore assunto da $W(s)$. □

Questa proprietà permette di dimostrare facilmente il seguente risultato.

Proposizione 1.5. *Il processo di Wiener è una martingala.*

Dimostrazione. La Proposizione 1.4 implica che, dati $t, s \in (0, +\infty)$ con $t > s$,

$$\mathbb{E}[W(t)|\mathscr{F}_s] = \mathbb{E}[W(t)|W(s)].$$

A questo punto, la tesi si dimostra esplicitando l'incremento:

$$\begin{aligned}\mathbb{E}[W(t)|\mathscr{F}_s] &= \mathbb{E}[W(t)|W(s)] \\ &= \mathbb{E}[\Delta W_{s,t} + W(s)|W(s)] \\ &= \mathbb{E}[\Delta W_{s,t}|W(s)] + \mathbb{E}[W(s)|W(s)] \\ &= 0 + W(s) = W(s).\end{aligned}$$

□

Concludiamo la nostra analisi iniziale delle proprietà del processo di Wiener con il seguente risultato.

Proposizione 1.6. *Le traiettorie del processo di Wiener non sono differenziabili in alcun punto quasi certamente.*

La dimostrazione di questo risultato è assai tecnica ed esula dagli obiettivi di questa trattazione. Una giustificazione non rigorosa di tale affermazione può essere ricondotta alla costruzione euristica presentata precedentemente. Infatti, le traiettorie del processo S_n diventano sempre più irregolari man mano che $n \to +\infty$, suggerendo che il processo limite presenti un comportamento irregolare in ogni suo punto. Le traiettorie del processo di Wiener sono quindi frattali, caratterizzate da una quantità infinita di dettaglio (indipendentemente da quanto piccolo sia l'intervallo $(t, t+dt)$, il grafico della traiettoria rimane frastagliato).

Processo di Wiener multidimensionale

Il Teorema 1.1 può essere esteso al caso multidimensionale.

Teorema 1.7. *Sia $\left(\Omega, \mathscr{F}, \mathbb{P}, \{\mathscr{F}_t\}_{t\geq 0}\right)$ uno spazio di probabilità filtrato. Esiste un processo stocastico $W : \Omega \times (0, +\infty) \to \mathbb{R}^d$ tale che:*

i) $W(0) = \mathbf{0}_d$ quasi certamente,

ii) W è adattato a $\{\mathscr{F}_t\}_{t\geq 0}$, e le sue traiettorie $W_\omega(t)$ sono quasi certamente continue componente per componente,

iii) dati $t > s \geq 0$, l'incremento del processo $\Delta W_{s,t}$ ha una distribuzione normale multivariata con media 0 e matrice di covarianza $(t-s)\mathbf{I}_d$, ed è indipendente da $\mathscr{F}_s$.

È possibile inoltre definire un moto browniano d-dimensionale con incrementi correlati, caratterizzato da incrementi del tipo:

$$\Delta W_{s,t} \sim \mathscr{N}_d\left(\mathbf{0}_d, \Sigma\right),$$

dove

$$\Sigma = \mathbf{A}((t-s)\,\mathbf{I}_d)\mathbf{A}^\intercal,$$

con $\mathbf{A} \in \mathbb{R}^{d\times d}$ non singolare.

Nella prossima sezione, descriveremo il calcolo differenziale stocastico di Itô, focalizzandoci sul caso monodimensionale, sebbene i risultati ottenuti siano estendibili al caso d-dimensionale.

1.3 Calcolo differenziale stocastico secondo Itô

Il processo di Wiener è di vitale importanza per la definizione di un *calcolo differenziale stocastico* che risulta essere fondamentale per affrontare problemi differenziali che coinvolgono processi aleatori. Ai fini di affrontare il problema del pricing di un derivato desideriamo descrivere l'andamento nel corso del tempo t del valore V di un portafoglio rischioso. Dato che il suo valore nel corso del tempo risulta essere incerto, esso può essere descritto da un processo stocastico. Potremmo, ad esempio, modellare la dinamica di tale processo usando il seguente sistema:

$$\begin{cases} V(t+\Delta t) - V(t) = \mu\Delta t + \sigma\Delta W_{t,t+\Delta t} \\ V(0) = V_0 \end{cases}, \tag{1.3}$$

dove μ è un parametro di drift e σ indica la volatilità del portafoglio. Ispirandoci all'approccio tipico della teoria dei sistemi di ODE, potremmo dividere la prima equazione per Δt e prenderne il limite per $\Delta t \to 0$ ottenendo

$$V'(t) = \mu + \sigma\frac{\mathrm{d}W(t)}{\mathrm{d}t},$$

tuttavia un simile approccio non è percorribile in pratica, dato che le traiettorie del processo di Wiener non sono derivabili quasi certamente per la Proposizione 1.6. Una maniera intelligente di aggirare questa criticità è quello di trasformare il problema (1.3) in un'equazione integrale

del tipo

$$V(t) = V(0) + \int_0^t \mu \mathrm{d}t + \int_0^t \sigma \mathrm{d}W(s), \tag{1.4}$$

dove il simbolo $\int_0^t f \mathrm{d}z$ denota l'integrale di Lebesgue-Stieljes. Questo approccio ci lascia la libertà di definire in maniera agevole le dinamiche del processo stocastico (come nel caso di un sistema di ODE) senza considerare la derivata del processo W. L'unico punto delicato riguarda l'effettiva integrabilità nel senso di Lebesgue-Stieljes delle traiettorie del processo di Wiener, ovvero l'esistenza finita della quantità $\int_0^t \sigma \mathrm{d}W(s)$.

Per determinare ciò è necessario introdurre il concetto di *variazione*. Consideriamo un intervallo $[a,b] \subset \mathbb{R}$, una funzione $g : [a,b] \to \mathbb{R}^n$ e una partizione $\varsigma = \{ t_0, t_1, \ldots, t_N \}$ di $[a,b]$. La variazione di g relativa a ς è definita da

$$V_{[a,b]}(g,\varsigma) = \sum_{k=1}^{n} |g(t_k) - g(t_{k-1})|,$$

dove $|\cdot|$ denota la norma euclidea.

Definizione 1.6 (Variazione prima)**.** La funzione g ha variazione limitata su $[a,b]$ e scriviamo $g \in BV([a,b])$, se l'estremo superiore di $V_{[a,b]}(g,\varsigma)$ al variare di tutte le partizioni ς di $[a,b]$ risulta essere finito:

$$V_{[a,b]}(g) = \sup_{\varsigma} V_{[a,b]}(g,\varsigma) < +\infty.$$

$V_{[a,b]}(g)$ è detta *variazione prima* o, più semplicemente, *variazione* di g su $[a,b]$.

La maggior parte delle funzioni con cui si è soliti lavorare sono funzioni a variazione limitata, tuttavia non è difficile trovare esempi di funzioni che non lo siano.

Esempio 1.2. La funzione

$$g(t) = \begin{cases} 0 & \text{per } t = 0, \\ t \sin\left(\frac{1}{t}\right) & \text{per } t \in (0,1], \end{cases}$$

è continua su $[0,1]$ ma non ha variazione limitata. Basti considerare una partizione con elementi del tipo $t_n = \left(\frac{\pi}{2} + n\pi\right)^{-1}$ per verificare che $V_{[a,b]}(g) = +\infty$.

Osservazione 1.8. *Geometricamente, la variazione $V_{[a,b]}(g,\varsigma)$ di una funzione*

$$g : [a,b] \to \mathbb{R}^n$$

*rappresenta la lunghezza della spezzata in $\mathbb{R}^n$ di estremi $g(t_k)$ per $k = 0,\dots,N$. Intuitivamente, se g è **continua** allora $V_{[a,b]}(g,\varsigma)$ approssima, al tendere di $|\varsigma|$ a zero[2], la lunghezza della curva g in $\mathbb{R}^n$; in altre parole la curva g è a variazione limitata (o rettificabile) se ha lunghezza finita approssimabile con spezzate.*

In maniera analoga a quanto fatto per la variazione prima, possiamo dare anche la seguente definizione.

Definizione 1.7 (Variazione quadratica)**.** Data una funzione $g : [a,b] \to \mathbb{R}^n$ e una partizione $\varsigma = \{t_0,\dots,t_N\}$ di $[a,b]$, la variazione quadratica di g relativa a ς è definita da

$$V_t^{(2)}(g,\varsigma) = \sum_{k=1}^{N} |g(t_k) - g(t_{k-1})|^2.$$

Se esiste il limite

$$\lim_{|\varsigma|\to 0} V_t^{(2)}(g,\varsigma) = \langle g\rangle_t,$$

allora diciamo che $\langle g\rangle_t$ è la variazione quadratica di g su $[a,b]$.

La relazione tra variazione prima e variazione quadratica delle funzioni continue è espressa dal risultato seguente.

Proposizione 1.9. *Se $g \in BV([a,b]) \cap C^0([a,b])$, allora*

$$\langle g\rangle_t = 0.$$

Dimostrazione. La funzione g è continua su $[a,b]$, di conseguenza per ogni $\varepsilon > 0$ esiste $\delta > 0$ tale per cui si ha che

$$|g(t_k) - g(t_{k-1})| \le \varepsilon$$

per ogni partizione $\varsigma = \{\, t_0, t_1, \dots, t_N \,\}$ di $[a,b]$ per cui si ha che e $|\varsigma| < \delta$. La tesi è conseguenza del fatto che

$$0 \le V_t^{(2)}(g,\varsigma) = \sum_{k=1}^{N} |g(t_k) - g(t_{k-1})|^2 \le \varepsilon \sum_{k=1}^{N} |g(t_k) - g(t_{k-1})| \le \varepsilon V_{[0,t]}(g),$$

dove la variazione prima $V_{[0,t]}(g)$ è finita per ipotesi. □

[2]Definiamo $|\varsigma| = \sup_{k=1,\dots N} t_k - t_{k-1}$

Teorema 1.10 (Regolarità delle traiettorie del processo di Wiener)**.** *Se W è un moto browniano si ha*

$$\lim_{|\varsigma|\to 0} V_t^{(2)}(W,\varsigma) = t \qquad \text{in } L^2(\Omega,\mathscr{F},\mathbb{P}). \tag{1.5}$$

Di conseguenza, per ogni $t > 0$, vale

$$\langle W_\omega \rangle_t = t = var(W(t)) \qquad \text{per quasi ogni } \omega \in \Omega, \tag{1.6}$$

e, per la Proposizione 1.9, W non ha variazione limitata su $[0,t]$ quasi certamente.

Dimostrazione. Fissiamo una partizione

$$\varsigma = \{ t_0, \ldots, t_N \},$$

e consideriamo gli incrementi $\Delta W_{t_{k-1},t_k} = \Delta_k$ per $k = 1,\ldots,N$. Ricordiamo che vale

$$\mathbb{E}\left[\Delta_k^2\right] = t_k - t_{k-1},$$

mentre il quarto momento risulta essere

$$\mathbb{E}\left[\Delta_k^4\right] = 3(t_k - t_{k-1})^2.$$

Partiamo provando l'Eq. (1.5): si ha

$$\begin{aligned}
\mathbb{E}\left[\left(V_t^{(2)}(W,\varsigma) - t\right)^2\right] &= \mathbb{E}\left[\left(\sum_{k=1}^N \Delta_k^2 - t\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_{k=1}^N (\Delta_k^2 - (t_k - t_{k-1}))\right)^2\right] \\
&= \sum_{k=1}^N \mathbb{E}\left[\left(\Delta_k^2 - (t_k - t_{k-1})\right)^2\right] + \\
&\qquad 2\sum_{h<k} \mathbb{E}\left[\left(\Delta_k^2 - (t_k - t_{k-1})\right)\left(\Delta_h^2 - (t_h - t_{h-1})\right)\right].
\end{aligned}$$

Ora si osserva che

$$\begin{aligned}
\mathbb{E}\left[\left(\Delta_k^2 - (t_k - t_{k-1})\right)^2\right] &= \mathbb{E}\left[\Delta_k^4\right] - 2(t_k - t_{k-1})\mathbb{E}\left[\Delta_k^2\right] + (t_k - t_{k-1})^2 \\
&= 2(t_k - t_{k-1})^2,
\end{aligned}$$

mentre (ricordando che Δ_k e Δ_h sono indipendenti se $h < k$)

$$\mathbb{E}\left[\left(\Delta_k^2 - (t_k - t_{k-1})\right)\left(\Delta_h^2 - (t_h - t_{h-1})\right)\right] =$$
$$\mathbb{E}\left[\left(\Delta_k^2 - (t_k - t_{k-1})\right)\right]\mathbb{E}\left[\left(\Delta_h^2 - (t_h - t_{h-1})\right)\right] = 0.$$

Quindi si ha

$$\mathbb{E}\left[\left(V_t^{(2)}(W,\varsigma) - t\right)^2\right] = 2\sum_{k=1}^{N}(t_k - t_{k-1})^2 \leq 2t|\varsigma|,$$

che implica

$$V_t^{(2)}(W,\varsigma) \xrightarrow[|\varsigma|\to 0]{L^2} t,$$

provando la (1.5).

A questo punto possiamo provare la (1.6) per assurdo: negare la tesi equivale all'affermare l'esistenza di $\varepsilon > 0$ e un evento A con $\mathbb{P}(A) > 0$, tali che per ogni $n \in \mathbb{N}$

$$|V_t^{(2)}(W_\omega, \varsigma_n) - t| > \varepsilon, \tag{1.7}$$

per ogni $\omega \in A$ e per una certa successione di partizioni $\{\varsigma_n\}_{n\in\mathbb{N}}$, tale che $|\varsigma_n| < \frac{1}{n}$. D'altra parte abbiamo appena dimostrato che la successione $\left\{V_t^{(2)}(W,\varsigma_n)\right\}$ converge a t in $L^2(\Omega,\mathscr{F},\mathbb{P})$, quindi ammette una sotto-successione convergente a t quasi certamente, il che contraddice (1.7). □

Il Teorema 1.10 ha una grandissima rilevanza dato che caratterizza la regolarità delle traiettorie del processo di Wiener, inserendole nella famiglia delle funzioni a variazione non limitata. Ciò risulta cruciale nell'ambito della nostra analisi per via del seguente risultato che non andiamo a dimostrare in maniera rigorosa.

Proposizione 1.11. *Le funzioni a variazione non limitata non sono integrabili secondo Lebesgue.*

Per farsi un'idea del motivo per cui ciò avvenga, si deve ricordare che l'integrazione secondo Lebesgue è definita a partire dall'integrazione di funzioni semplici e poi estesa alle altre funzioni mediante il *teorema di convergenza monotona* e il *teorema di convergenza dominata*. Tuttavia nel caso di funzioni irregolari come le traiettorie del processo di Wiener tali risultati non possono essere applicati; di conseguenza l'integrale dell'Eq. (1.4) risulta non essere definito. In conclusione, se si desidera applicare l'approccio descritto all'inizio di questa sezione, è necessario definire un nuovo tipo di integrale.

1.3.1 Integrale di Itô

La restante parte di questo capitolo è stata realizzata seguendo il libro di Pascucci (2008). La necessità di dare un significato all'espressione $\int_0^t \sigma \mathrm{d}W(s)$, richiede di formulare un'estensione della teoria dell'integrazione secondo Lebesgue che permetta di integrare funzioni rispetto a traiettorie di processi stocastici a *variazione quadratica limitata*. Gli integrali che permettono di fare ciò sono detti *integrali stocastici*. L'uso del plurale deve far intendere l'esistenza di diverse teorie, ma in questa trattazione ci concentreremo esclusivamente su una di esse: l'*integrale di Itô*. Chiaramente, la definizione di un simile integrale è un processo estremamente tecnico e delicato e la teoria ad esso collegata è di livello estremamente avanzato. Una presentazione puntuale ed esaustiva di essa esula dall'obiettivo di questa trattazione; ciò nonostante si cercherà di fornire un'idea generale dei principali risultati che costituiscono la teoria dell'integrazione secondo Itô.

Il procedimento per definire l'integrale sopracitato ricalca a grandi linee il processo attraverso il quale è possibile formulare l'integrale di Lebesgue: si definisce l'integrale per le funzioni semplici e poi si estende a una famiglia più ampia di funzioni sufficientemente regolari. Per fare ciò, introduciamo alcune definizioni di base. Consideriamo un processo stocastico $u : \Omega \times [0,T] \to \mathbb{R}$, dove $T \in \mathbb{R}$, $T > 0$.

Definizione 1.8 (Processo progressivamente misurabile)**.** Il processo stocastico u si dice *progressivamente misurabile* rispetto alla filtrazione $\{\mathscr{F}_t\}_{t\in[0,T]}$ se, per ogni $t \in [0,T]$, $u|_{\Omega\times[0,t]}$ è $\mathscr{F}_t \otimes \mathscr{B}([0,t])$-misurabile.

Definizione 1.9. Il processo stocastico u appartiene alla classe $\mathbb{L}^2$ se

i) u è progressivamente misurabile rispetto alla filtrazione $\{\mathscr{F}_t\}_{t\in[0,T]}$,

ii) $\int_0^T \mathbb{E}\left[(u(t))^2\right] \mathrm{d}t$ esiste finito[3].

Quest'ultima definizione può essere data più in generale, indicando con $\mathbb{L}^p$ lo spazio dei processi progressivamente misurabili di $L^p(\Omega \times [0,T])$, al variare di $p \geq 1$.

Definizione 1.10. Un processo $u \in \mathbb{L}^2$ si dice *semplice* se è della forma

$$u = \sum_{k=1}^{N} e_k(\omega)\mathbf{1}_{(t_{k-1},t_k)}, \tag{1.8}$$

[3]Ciò equivale a chiedere che $u \in L^2(\Omega \times [0,T])$.

dove $0 \leq t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_N$ e le e_k sono variabili aleatorie[4] su $(\Omega, \mathscr{F}, \mathbb{P})$.

Le funzioni semplici permettono di definire in maniera agevole l'integrale di Itô, nel senso della seguente definizione.

Definizione 1.11. Dato un processo stocastico $u \in \mathbb{L}^2$ semplice sia

$$\int u(t)dW(t) = \sum_{k=1}^{N} e_k \left(W(t_k) - W(t_{k-1})\right),$$

dove W è il processo di Wiener. Si definisce *integrale di Itô* di u sull'intervallo (a,b) la variabile aleatoria definita come

$$\int_a^b u(t)dW(t) = \int u(t)\mathbf{1}_{(a,b)}(t)dW(t).$$

Il seguente teorema enuncia alcune importanti proprietà dell'integrale di Itô di processi semplici.

Teorema 1.12. *Per ogni $u, v \in \mathbb{L}^2$ semplici, $\alpha \in \mathbb{R}$ e $0 \leq a < b < c$, valgono le seguenti proprietà*

1. *linearità:*
$$\int (\alpha u(t) + v(t))\, dW(t) = \alpha \int u(t)dW(t) + \int v(t)dW(t); \tag{1.9}$$

2. *additività:*
$$\int_a^b u(t)dW(t) + \int_b^c u(t)dW(t) = \int_a^c u(t)dW(t); \tag{1.10}$$

3. *attesa nulla:*
$$\mathbb{E}\left[\int_a^b u(t)dW(t)\middle|\mathscr{F}_a\right] = 0, \tag{1.11}$$

   e
$$\mathbb{E}\left[\int_a^b u(t)dW(t)\int_b^c v(t)dW(t)\middle|\mathscr{F}_a\right] = 0; \tag{1.12}$$

4. *isometria di Itô:*
$$\mathbb{E}\left[\int_a^b u(t)dW(t)\int_a^b v(t)dW(t)\middle|\mathscr{F}_a\right] = \mathbb{E}\left[\int_a^b u(t)v(t)dt\middle|\mathscr{F}_a\right]. \tag{1.13}$$

[4] È comodo assumere che $\mathbb{P}(e_k = e_{k-1}) = 0$, $\forall k = 2, \ldots, N$, così da far sì che la rappresentazione (1.8) sia unica.

Osservazione 1.13. *Ricordando che*

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|\mathscr{F}_a]],$$

si ricavano le versioni non condizionate delle proprietà 3. e 4.:

$$\begin{aligned}
\mathbb{E}\left[\int_a^b u(t)dW(t)\right] &= 0,\\
\mathbb{E}\left[\int_a^b u(t)dW(t)\int_b^c v(t)dW(t)\right] &= 0,\\
\mathbb{E}\left[\int_a^b u(t)dW(t)\int_a^b v(t)dW(t)\right] &= \mathbb{E}\left[\int_a^b u(t)v(t)dt\right];
\end{aligned}$$

ponendo $u = v$ nell'ultima identità, si ottiene un'uguaglianza di norme L^2

$$\left\|\int_a^b u(t)dW(t)\right\|_{L^2(\Omega)} = \|u\|_{L^2(\Omega,\times[a,b])}$$

che giustifica in nome "isometria".

Dimostrazione. Le proprietà 1., 2., 3. sono facili da dimostrare usando la Definizione 1.11 e le proprietà fondamentali dell'operatore di attesa. Proviamo l'isometria di Itô: assumendo u e v semplici, si ha

$$\begin{aligned}
\mathbb{E}\left[\int_a^b u(t)dW(t)\int_a^b v(t)dW(t)\middle|\mathscr{F}_a\right] &= \mathbb{E}\left[\sum_{k=1}^N \int_{t_{k-1}}^{t_k} e_k dW(t)\sum_{h=1}^N \int_{t_{h-1}}^{t_h} d_h dW(t)\middle|\mathscr{F}_a\right]\\
&= \sum_{k=1}^N \mathbb{E}\left[\int_{t_{k-1}}^{t_k} e_k dW(t)\int_{t_{k-1}}^{t_k} d_k dW(t)\middle|\mathscr{F}_a\right]\\
&\quad + 2\sum_{h<k}^N \mathbb{E}\left[\int_{t_{k-1}}^{t_k} e_k dW(t)\int_{t_{h-1}}^{t_h} d_h dW(t)\middle|\mathscr{F}_a\right] =
\end{aligned}$$

(per la (1.12) i termini della seconda sommatoria sono tutti nulli)

$$= \sum_{k=1}^N \mathbb{E}\left[e_k d_k (W(t_k) - W(t_{k-1}))^2\middle|\mathscr{F}_a\right] = \sum_{k=1}^N \mathbb{E}[e_k d_k|\mathscr{F}_a]\,\mathbb{E}\left[(W(t_k) - W(t_{k-1}))^2\right] =$$

(usando l'indipendenza di $W(t_k) - W(t_{k-1})$ da $e_k d_k$ e $\mathscr{F}_a$)

$$= \sum_{k=1}^{N} \mathbb{E}\left[e_k d_k | \mathscr{F}_a\right](t_k - t_{k-1}) = \mathbb{E}\left[\sum_{k=1}^{N} e_k d_k (t_k - t_{k-1}) \middle| \mathscr{F}_a\right]$$
$$= \mathbb{E}\left[\int_a^b u(t)v(t)dt \middle| \mathscr{F}_a\right].$$

□

A questo punto, come accennato in precedenza, si procede estendendo la definizione dell'integrale di Itô a tutti i processi di $\mathbb{L}^2$, usando il seguente risultato.

Teorema 1.14. *Per ogni $u \in \mathbb{L}^2$ esiste una successione $\{u_n\}_{n\in\mathbb{N}}$ di processi semplici di $\mathbb{L}^2$ tale che*

$$\lim_{n\to+\infty} \int_0^T \mathbb{E}\left[(u(t) - u_n(t))^2\right] dt = \lim_{n\to+\infty} \|u - u_n\|^2_{L^2(\Omega\times[0,T])} = 0.$$

Definizione 1.12. Dati $u \in \mathbb{L}^2$ e una successione approssimante $\{u_n\}_{n\in\mathbb{N}}$ di processi semplici di $\mathbb{L}^2$, l'integrale di Itô di u è definito come

$$\int_0^t u(s)dW(s) = \lim_{n\to+\infty} \int_0^t u_n(s)dW(s). \tag{1.14}$$

Chiaramente abbiamo sorvolato alcuni dettagli delicati, a partire dalla dimostrazione del Teorema 1.14. Inoltre, pur ammettendo l'esistenza di una successione approssimante, non è in generale banale dimostrare che il limite del membro di destra della (1.14) esista. Un altro punto che richiede particolare attenzione riguarda la dimostrazione del fatto che la funzione integrale

$$I(\omega,t) = \int_0^t u(\omega,s)dW(s),$$

sia un processo stocastico (ovvero misurabile rispetto alle dovute σ-algebre). Per fornire una dimostrazione completa di questi fatti è necessario ricorrere a risultati relativi allo studio delle martingale (disuguaglianza di Doob) e dei tempi d'arresto, oltre che a una discreta dose di nozioni provenienti dall'analisi funzionale.

Il seguente teorema estende le proprietà dell'integrale di Itô per funzioni semplici (Teorema 1.12) a tutte le funzioni di $\mathbb{L}^2$.

Teorema 1.15. *Per ogni $u, v \in \mathbb{L}^2$, $\alpha \in \mathbb{R}$ e $0 \le a < b < c$, valgono le seguenti proprietà*

1. *linearità:*

$$\int_0^a (\alpha u(t) + v(t))\, dW(t) = \alpha \int_0^a u(t) dW(t) + \int_0^a v(t) dW(t);$$

2. *additività:*

$$\int_a^b u(t) dW(t) + \int_b^c u(t) dW(t) = \int_a^c u(t) dW(t);$$

3. *attesa nulla:*

$$\mathbb{E}\left[\int_a^b u(t) dW(t) \middle| \mathscr{F}_a\right] = 0,$$

e

$$\mathbb{E}\left[\int_a^b u(t) dW(t) \int_b^c v(t) dW(t) \middle| \mathscr{F}_a\right] = 0;$$

4. *isometria di Itô:*

$$\mathbb{E}\left[\int_a^b u(t) dW(t) \int_a^b v(t) dW(t) \middle| \mathscr{F}_a\right] = \mathbb{E}\left[\int_a^b u(t) v(t) dt \middle| \mathscr{F}_a\right].$$

Non entriamo nel dettaglio delle dimostrazioni di tali risultati, dato che si basano tutte su procedimenti di passaggio al limite di quelli del Teorema 1.12.

Infine è doveroso sottolineare che l'integrale di Itô può essere esteso a una famiglia di funzioni più ampia di quelle in $\mathbb{L}^2$, individuate dallo spazio $\mathbb{L}^2_{\text{loc}}$ così definito.

Definizione 1.13. Dato $p \geq 1$, indichiamo con $\mathbb{L}^p_{\text{loc}}$ la famiglia dei processi progressivamente misurabili u, tali che

$$\int_0^T |u(t)|^p dt < +\infty \qquad \text{quasi certamente.}$$

Chiaramente $\mathbb{L}^2 \subseteq \mathbb{L}^2_{\text{loc}}$ e tale inclusione risulta essere stretta, in quanto è possibile mostrare che esistono processi che sono in $\mathbb{L}^2_{\text{loc}} \setminus \mathbb{L}^2$. Tale estensione risulta vantaggiosa per via della seguente proposizione.

Proposizione 1.16. *Ogni processo stocastico progressivamente misurabile, con traiettorie continue quasi certamente appartiene allo spazio* $\mathbb{L}^2_{\text{loc}}$.

Quest'ultimo risultato, unito alla estensione dell'integrale di Itô allo spazio $\mathbb{L}^2_{\text{loc}}$, ci garantisce la possibilità di calcolare integrali stocastici del tipo

$$\int_a^b u(t) dW(t)$$

dove $u(t)$ è un qualsiasi processo stocastico con traiettorie quasi certamente continue.

1.3.2 Processi di Itô e formula di Itô

L'integrale stocastico ci permette di dare un senso a espressioni come la (1.4) dove compaiono integrali rispetto alle traiettorie del processo di Wiener. A questo punto, desideriamo fornire una caratterizzazione dei processi stocastici ottenibili mediante l'integrazione secondo Itô e studiarne alcune proprietà.

Iniziamo presentando il seguente risultato.

Proposizione 1.17. *Dato $u \in \mathbb{L}^2_{\text{loc}}$, consideriamo il processo stocastico integrale*

$$X(t) = \int_0^t u(s)dW(s).$$

Il processo

$$\langle X \rangle_t = \int_0^t u^2(s)ds,$$

è detto processo variazione quadratica e vale

$$\langle X \rangle_t = \lim_{|\varsigma| \to 0} \sum_{k=1}^{N} |X(t_k) - X(t_{k-1})|^2 \qquad \textit{quasi certamente},$$

dove ς è una partizione dell'intervallo $(0,t)$.

Dimostrare esaustivamente questo risultato richiede l'utilizzo della caratterizzazione dello spazio $\mathbb{L}^2_{\text{loc}}$, sulla quale non ci siamo soffermati in precedenza, quindi non riportiamo i passaggi. Vedremo che questa proprietà ha implicazioni abbastanza rilevanti per quanto riguarda il calcolo differenziale stocastico.

Siamo finalmente pronti a dare la definizione della classe di processi stocastici con cui lavoreremo in pratica.

Definizione 1.14 (Processo di Itô)**.** Un *processo di Itô* è un processo stocastico X della forma

$$X(t) = X(0) + \int_0^t \mu(s)ds + \int_0^t \sigma(s)dW(s), \qquad t \in [0,T], \tag{1.15}$$

dove $X(0)$ è una variabile aleatoria $\mathscr{F}_0$-misurabile, $\mu \in \mathbb{L}^1_{\text{loc}}$ e $\sigma \in \mathbb{L}^2_{\text{loc}}$.

Per alleggerire la notazione, i processi di Itô vengono spesso indicati con la seguente notazione, detta *forma differenziale*,

$$dX(t) = \mu(t)dt + \sigma(t)dW(t),$$

Il processo μ è detto coefficiente di *drift* (o deriva). Il processo σ è detto coefficiente di *diffusione*. Intuitivamente μ "imprime la direzione" alle traiettorie di X, mentre i termine contenente σ è un "contributo stocastico" all'evoluzione del processo.

Un equazione della forma (1.15) nell'incognita X con $X(0)$ dato è detta *equazione differenziale stocastica*. Lo studio delle soluzioni di simili equazioni è un argomento vastissimo. In questa trattazione ci concentreremo sul come risolvere tali problemi mediante metodi numerici che vadano a generare approssimazioni delle traiettorie del processo X.

Osservazione 1.18. *Chiaramente il processo di Wiener è un processo di Itô, la cui forma differenziale è data dall'uguaglianza*

$$dW(t) = dW(t)$$

Osservazione 1.19. *Il processo V considerato all'inizio della sezione è un processo di Itô. Infatti, la sua forma differenziale è data da*

$$dV(t) = \mu dt + \sigma dW(t),$$

con $\mu, \sigma \in \mathbb{R}$ costanti.

È possibile dimostrare che la rappresentazione in forma differenziale di un processo di Itô è unica quasi ovunque, cioè dato un processo di Itô X i processi $\mu(t)$ e $\sigma(t)$ che ne caratterizzano la forma differenziale sono unici a meno di insiemi trascurabili. Se i processi di drift e diffusione sono deterministici (ovvero non dipendono dallo scenario $\omega \in \Omega$), è possibile caratterizzare facilmente il processo di Itô risultante.

Proposizione 1.20. *Se $\mu \in L^1$ e $\sigma \in L^2$ sono funzioni deterministiche allora il processo definito da*

$$dS(t) = \mu(t)dt + \sigma(t)dW(t), \qquad S(0) = S_0,$$

è un processo gaussiano e vale

$$\mathbb{E}[S(t)] = S_0 + \int_0^t \mu(s)ds \qquad e \qquad Var(S(t)) = \int_0^t \sigma^2(s)ds \qquad \forall t \in [0,T]$$

Dimostrazione. Dalla definizione di processo di Itô, si ha

$$S(t) = S_0 + \int_0^t \mu(s)ds + \int_0^t \sigma(s)dW(s).$$

Mostriamo che

$$\int_0^t \sigma(s)dW(s) \sim \mathcal{N}\left(0, \int_0^t \sigma^2(s)ds\right).$$

Si ha che

$$\mathbb{E}\left[\int_0^t \sigma(s)dW(s)\right] = 0$$

per le proprietà dell'integrale di Itô; si ha inoltre

$$\mathbb{E}\left[\int_0^t \sigma(s)dW(s)\right] = \mathbb{E}\left[\int_0^t \sigma^2(s)ds\right] \qquad \text{(isometria di Itô)},$$

che dimostra immediatamente

$$\mathrm{Var}\left(\int_0^t \sigma(s)dW(s)\right) = \mathbb{E}\left[S(t)^2\right] - \mathbb{E}\left[S(t)\right]^2 = \int_0^t \sigma^2(s)ds.$$

La normalità di $\int_0^t \sigma(s)dW(s)$ è dimostrabile mediante lo studio della funzione caratteristica $\mathbb{E}\left[e^{iz\int_0^t \sigma(s)dW(s)}\right]$. Non riportiamo qui i passaggi che sono tutti estremamente agevoli. A questo punto, si ha che $S(t)$ è una trasformazione lineare di una variabile gaussiana, quindi è anch'essa gaussiana ed è immediato dimostrare che i suoi momenti sono quelli dell'enunciato. □

Il processo descritto dalle ipotesi della precedente proposizione viene chiamato *moto browniano con drift* e si indica con il simbolo $\mathrm{BM}(\mu(t), \sigma(t))$.

Siamo ora pronti per enunciare il teorema fondamentale del calcolo differenziale stocastico secondo Itô.

Teorema 1.21 (Formula di Itô)**.** *Sia X un processo di Itô e $f : \mathbb{R}^2 \to \mathbb{R}$, $f(t,x) \in C^2(\mathbb{R}^2)$. Allora il processo stocastico*

$$Y(t) = f(t, X(t))$$

è un processo di Itô e vale

$$df(t,X(t)) = \partial_t f(t,X(t))dt + \partial_x f(t,X(t))dX(t) + \frac{1}{2}\partial_{xx} f(t,X(t))d\langle X\rangle_t.$$

Osservazione 1.22. *Ricordando la Proposizione 1.17, si ha che*

$$d\langle X\rangle_t = \sigma^2(t)dt,$$

e, usando la rappresentazione differenziale di X, possiamo riscrivere la formula di Itô nel seguente modo

$$df = \left(\partial_t f + \mu(t)\partial_x f + \frac{1}{2}\sigma^2(t)\partial_{xx} f\right)dt + \sigma(t)\partial_x f dW(t) \tag{1.16}$$

La formula di Itô ci dice che il calcolo differenziale relativo ai processi di Itô è fondamentalmente diverso da quello a cui siamo abituati. Il Teorema 1.21 è una diretta conseguenza della Proposizione 1.17 che ci dice che il differenziale di ordine due di un processo di Itô ($dX^2(t) = d\langle X\rangle_t$) è equivalente al differenziale di ordine uno in tempo dt. Quindi se si va a considerare uno sviluppo in serie di Taylor al primo ordine di una funzione sufficientemente liscia f, compare un termine aggiuntivo al differenziale di f pari a $\frac{1}{2}\partial_{xx} f(t,X(t))dX^2(t)$. Seppur qui non sia presentata in maniera formale, questa è la dimostrazione del teorema sopracitato.

Esempio 1.3. La formula di Itô ci permette di verificare

$$\int_0^t W(s)dW(s) \neq \frac{1}{2}(W(t)^2 - W(0)^2) = \frac{1}{2}W(t)^2,$$

Applichiamo la formula di Itô a $f(t,x) = x^2$ e otteniamo

$$d\left(W^2(t)\right) = \left(0 + 0\cdot 2W(t) + \frac{1}{2}\cdot 1\cdot 2\right)dt + 1\cdot 2W(t)dW(t) = dt + 2W(t)dW(t),$$

da cui si ottiene

$$\int_0^t W(s)dW(s) = \frac{W^2(t) - t}{2}.$$

Concludiamo questo primo capitolo presentando le proprietà di alcuni processi di Itô con interessanti applicazioni finanziarie.

Moto browniano geometrico

Il processo di Itô, dato dalla forma differenziale

$$dS(t) = \mu(t)S(t)dt + \sigma(t)S(t)dW(t),$$

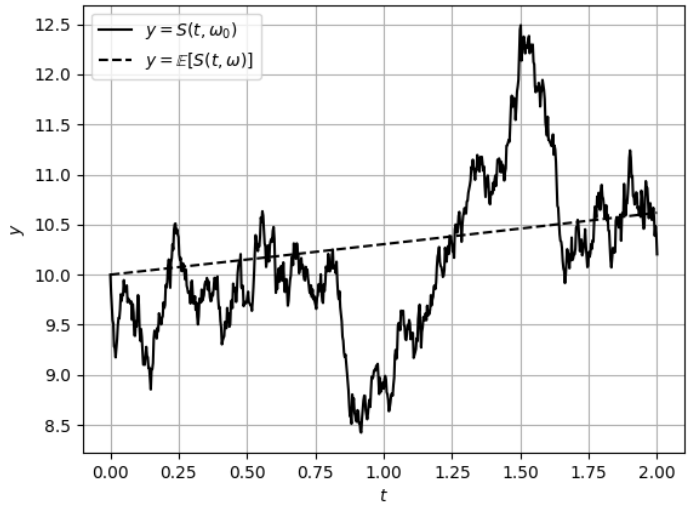

Figura 1.2 Il grafico di una traiettoria del moto browniano geometrico generata mediante le funzionalità implementate in *SFMQuantLib*, ottenuta con $S(0) = 10$, $\mu = 3\%$ e $\sigma = 20\%$. Il grafico include la traiettoria e il valore atteso al tempo t.

o, equivalentemente,

$$\frac{dS(t)}{S(t)} = \mu(t)dt + \sigma(t)dW(t),$$

è comunemente noto come *moto browniano geometrico*. È possibile osservare una traiettoria in Fig. 1.2. Questo processo è stato utilizzato da Black, Scholes e Merton per descrivere l'andamento temporale del prezzo di un'azione (vedi Sezione 4.3), poiché, scelto un valore iniziale $S(0) > 0$, si ha che $S(t) > 0$ per ogni $t \in [0, T]$. Tale processo può essere ottenuto come l'esponenziale di un moto browniano con drift. Infatti, se consideriamo il processo $Y(t) = \log(S(t))$, applicando la formula di Itô, otteniamo la forma differenziale:

$$dY(t) = \left(\mu(t) - \frac{1}{2}\sigma^2(t)\right)dt + \sigma(t)dW(t),$$

che corrisponde a un moto browniano con drift $\mu(t) - \frac{1}{2}\sigma^2(t)$ e volatilità $\sigma(t)$. Di conseguenza, non è sorprendente osservare che:

$$S(t) \sim \text{Log-Normale}\left(\int_0^t \mu(s)ds, \int_0^t \sigma^2(s)ds\right),$$

ricordando che la distribuzione log-normale si ottiene dall'esponenziale di una variabile aleatoria gaussiana.

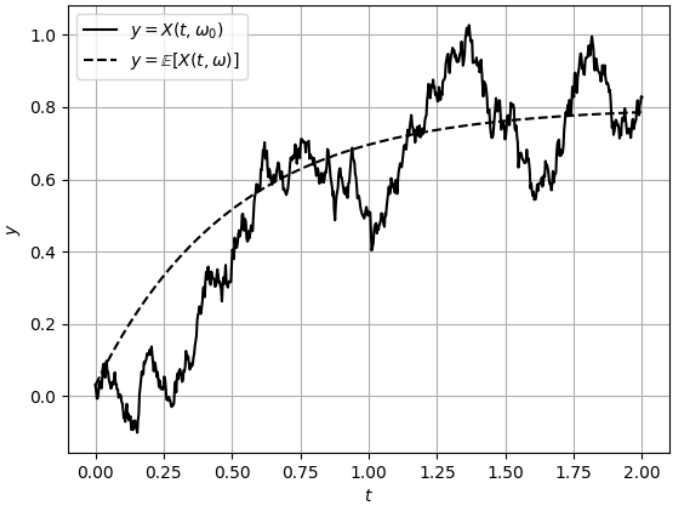

Figura 1.3 Il grafico di una traiettoria del processo di Ornstein-Uhlenbeck generata mediante le funzionalità implementate in *SFMQuantLib*, ottenuta con $X(0) = 0.03$, $\kappa = 2$, $\theta = 0.8$ e $\sigma = 0.4$. Il grafico include la traiettoria e il valore atteso al tempo t.

Processo di Ornstein-Uhlenbeck

Il processo di Itô

$$dX(t) = \kappa(t)\left(\theta - X(t)\right)dt + \sigma(t)dW(t)$$

è comunemente noto come *processo di Ornstein-Uhlenbeck*, in onore dei fisici olandesi Leonard Ornstein e George Eugene Uhlenbeck che ne fornirono una caratterizzazione. Una traiettoria di questo processo è mostrata in Fig. 1.3. La principale caratteristica di questo processo è la tendenza delle sue traiettorie a ritornare, nel lungo termine, verso il valore θ. Il parametro κ controlla la velocità con cui il processo raggiunge il suo regime stazionario, mentre σ è il coefficiente di diffusione. Queste proprietà rendono il processo un modello utile per descrivere quantità come tassi di interesse o volatilità, per le quali ci si aspetta che l'andamento nel tempo sia caratterizzato da oscillazioni intorno a un valore nominale fisso. È possibile dimostrare che questo processo è gaussiano e che:

$$\mathbb{E}\left[X(t)|X(0)\right] = X(0)e^{-\kappa t} + \theta\left(1 - e^{-\kappa t}\right),$$
$$\mathrm{Var}\left(X(t)|X(0)\right) = \frac{\sigma^2}{2\kappa}\left(1 - e^{-2\kappa t}\right).$$

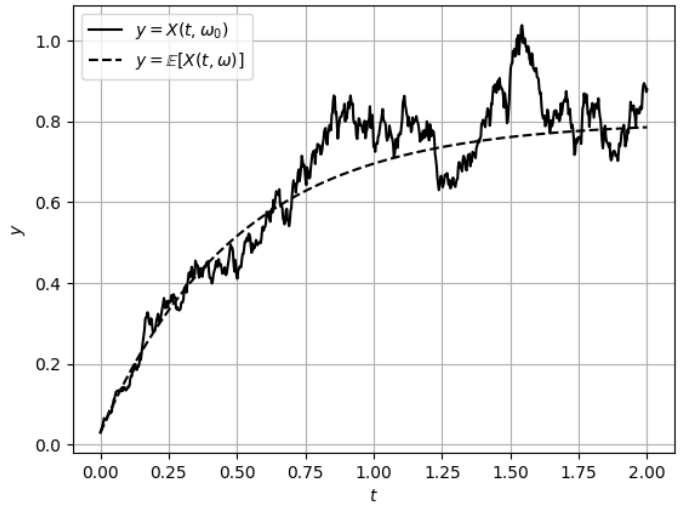

Figura 1.4 Il grafico di una traiettoria del processo CIR generata mediante le funzionalità implementate in *SFMQuantLib*, ottenuta con $X(0) = 0.03$, $\kappa = 2$, $\theta = 0.8$ e $\sigma = 0.4$. Il grafico include la traiettoria e il valore atteso al tempo t.

Processo di Cox-Ingersoll-Ross

Un limite nell'utilizzo del processo di Ornstein-Uhlenbeck per descrivere l'andamento di un tasso di interesse o di una volatilità è che le traiettorie ottenute possono assumere valori negativi con probabilità positiva. Una soluzione consiste nel modificare leggermente la SDE del processo, ottenendo:

$$dX(t) = \kappa(t)\,(\theta - X(t))\,dt + \sigma(t)\sqrt{X(t)}dW(t),$$

che è la forma differenziale del *processo di Cox-Ingersoll-Ross* (CIR), di cui è riportata una traiettoria in Fig. 1.4. Questo processo mantiene le proprietà di *mean-reversion* (ritorno alla media) del precedente, ma, a differenza di esso, dato un valore iniziale positivo, le sue traiettorie assumono esclusivamente valori non negativi. In particolare, se $2\kappa\theta \geq \sigma^2$, allora le traiettorie sono sempre positive. Il processo CIR non è gaussiano, ma può essere caratterizzato in termini di una variabile aleatoria con distribuzione χ^2 non centrata. È possibile mostrare che:

$$\begin{aligned}
\mathbb{E}[X(t)|X(0)] &= X(0)e^{-\kappa t} + \theta\left(1 - e^{-\kappa t}\right), \\
\mathrm{Var}(X(t)|X(0)) &= X(0)\frac{\sigma^2}{\kappa}\left(e^{-\kappa t} - e^{-2\kappa t}\right) + \frac{\sigma^2\theta}{2\kappa}\left(1 - e^{-\kappa t}\right)^2.
\end{aligned}$$

Capitolo 2

Metodi Monte Carlo

In questo secondo capitolo verranno presentati i principali concetti legati al tema dei *metodi Monte Carlo* per la stima di parametri e quantità legati a variabili aleatorie e processi stocastici. Come vedremo, questi nascono dall'unione della teoria dell'integrazione Monte Carlo con diversi approcci per la simulazione di variabili aleatorie e la generazione di numeri casuali.

Prima di approfondire tutti questi aspetti, è opportuno dare un'idea del motivo per cui si è interessati a tali approcci. In svariati contesti, è necessario stimare quantità che possono essere ricondotte alla forma

$$\mathbb{E}[f(X)] = \int_{\Omega} f(X) d\mathbb{P},$$

con $X : \Omega \to E$ variabile aleatoria (o vettore aleatorio), $f : E \to \mathbb{R}$ funzione misurabile e integrabile, e $\mathbb{P}$ misura di probabilità su $(\Omega, \mathscr{F})$. Tale integrazione può essere un problema assai difficile da affrontare se X prende valori in un insieme a dimensione elevata, se non ne si conosce la distribuzione, o se non si conosce f in maniera analitica ma si è semplicemente in grado di valutarla sugli elementi di E (una simile funzione è detta *black-box*). I metodi Monte Carlo sono una famiglia di procedure che permettono di risolvere simili problemi richiedendo ipotesi minime su f e X. Essi risultano essere particolarmente competitivi nel caso di problemi complessi e in dimensione elevata come quelli che descriveremo nei capitoli successivi.

Nella Sezione 2.1 presenteremo le principali nozioni relative ai metodi Monte Carlo come tecnica di integrazione numerica, mentre nella Sezione 2.2 effettueremo una breve analisi dei principali metodi per la simulazione di variabili aleatorie.

2.1 Integrazione Monte Carlo

Questa sezione è stata realizzata seguendo il testo di Glasserman (2004). Innanzitutto, enunciamo i due principali teoremi limite del calcolo della probabilità che costituiscono le fondamenta dell'integrazione Monte Carlo.

Teorema 2.1 (Legge forte dei grandi numeri)**.** *Sia* $\{X_i\}_{i\geq 1}$ *una successione di variabili casuali i.i.d. e in* L^1*. Sia* μ *il loro comune valor medio e*

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i,$$

la loro media campionaria. Allora

$$\bar{X}_n \xrightarrow[n\to+\infty]{q.c} \mu.$$

Teorema 2.2 (Teorema limite centrale)**.** *Sia* $\{X_i\}_{i\geq 1}$ *una successione di variabili casuali i.i.d. e in* L^2*. Sia* μ *il loro valor medio,* σ^2 *la loro varianza e*

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i,$$

la loro media campionaria. Allora

$$\sqrt{n}\,(\bar{X}_n - \mu) \xrightarrow[n\to+\infty]{\mathscr{L}} \mathscr{N}(0,\sigma^2).$$

In maniera informale possiamo riassumere l'enunciato di questi due teoremi nel seguente modo:

- data una collezione di variabili aleatorie i.i.d. $\{X_i\}_{i=1}^n$, con n sufficientemente grande (ovvero un campione statistico casuale sufficientemente numeroso), è lecito aspettarsi che
$$\bar{X}_n \approx \mu = \mathbb{E}[X_i], \qquad i = 1,\ldots,n;$$
- la probabilità di osservare l'errore $(\bar{X}_n - \mu)$ in un determinato range è approssimativamente data da una distribuzione di tipo normale di media nulla e deviazione standard $\frac{\sigma}{\sqrt{n}}$, da cui è lecito concludere che l'ordine di convergenza di $\bar{X}_n$ a μ è $O\left(\frac{1}{\sqrt{n}}\right)$.

Questi due fatti avranno un ruolo fondamentale nella costruzione del metodo d'integrazione che andremo ora a descrivere.

L'idea alla base dei metodi Monte Carlo è quella di sfruttare l'analogia che esiste tra la probabilità e la massa (o il volume). La teoria della misura formalizza l'idea intuitiva di probabilità di un evento, descrivendola appunto come il "peso" di quest'ultimo relativamente a quello dello spazio di tutti i possibili risultati. Gli approcci Monte Carlo sfruttano questa analogia al contrario: calcolano la massa relativa di un insieme e interpretano poi questa come una probabilità. Ad esempio, supponiamo di essere interessati a stimare la probabilità che l'altezza di un individuo appartenente a una popolazione sia maggiore di una certa soglia. Formalmente, data una collezione di variabili aleatorie i.i.d. $\{H_i\}_{i=1}^n$ vogliamo stimare

$$p = \mathbb{P}(H_i > \alpha).$$

Possiamo stimare questa quantità usando lo stimatore Monte Carlo

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{(\alpha,+\infty)}(H_i),$$

che, in termini più semplici, equivale al contare quanti individui tra quelli del campione statistico soddisfano la proprietà desiderata e poi dividere per la numerosità del campione stesso o, nell'ottica dell'analogia massa-probabilità, misurare la massa relativa dell'evento rispetto all'intero campione. La legge dei grandi numeri assicura che la stima $\hat{p}$ converge al valore corretto p a mano a mano che la dimensione del campione considerato cresce, infatti

$$p = \mathbb{P}(H > \alpha) = \mathbb{E}\left[\mathbf{1}_{(\alpha,+\infty)}(H)\right] \approx \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{(\alpha,+\infty)}(H_i) = \hat{p}.$$

Da qui il passaggio agli integrali è immediato. Consideriamo, ad esempio, il problema di integrare una funzione f sull'intervallo $(0,1)$

$$\phi = \int_0^1 f(x)dx.$$

Possiamo vedere tale quantità come l'attesa $\mathbb{E}[f(U)]$, dove U è una variabile aleatoria di distribuzione uniforme sull'intervallo $(0,1)$. A questo punto, supponiamo di avere una procedura per campionare in modo indipendente da questa distribuzione, ottenendo così una collezione di valori $\{u_i\}_{i=1}^n \subset (0,1)$. Valutando f in questi punti e prendendone il valor medio, si ottiene la stima Monte Carlo dell'integrale

$$\hat{\phi}_n = \frac{1}{n}\sum_{i=1}^{n} f(u_i).$$

Per quanto detto in precedenza, abbiamo che se $f \in L^1(0,1)$ allora $\hat{\phi}_n \to \phi$ quasi certamente.

Se $f \in L^2(0,1)$, allora esiste finita la quantità

$$\sigma_f^2 = \int_0^1 (f(x) - \phi)^2 \, dx,$$

e, per il teorema limite centrale, l'errore $\phi - \hat{\phi}$ è distribuito in modo normale con media 0 e deviazione standard $\frac{\sigma_f}{\sqrt{n}}$. Il parametro σ_f sarebbe normalmente sconosciuto in un contesto in cui non si conosce ϕ, ma possiamo stimarlo utilizzando il suo stimatore Monte Carlo

$$s_f = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(f(u_i) - \hat{\phi}\right)^2},^{1}$$

Quindi, da un singolo campione $\{ u_i \}_{i=1}^n$ otteniamo sia una stima della quantità desiderata sia un'idea dell'errore commesso. La forma dell'errore standard $\sigma_f/\sqrt{n}$ è una caratteristica fondamentale dell'integrazione Monte Carlo. Ciò significa che per dimezzare l'errore è necessario quadruplicare la dimensione del campione, mentre aumentare la precisione di stima di un ordine di grandezza (aggiungere alla stima una cifra decimale corretta) richiede di aumentare la dimensione del campione di cento volte. A confronto, l'ordine di convergenza del metodo dei trapezi risulta essere $O\left(n^{-2}\right)$ (purché f sia due volte derivabile), che è molto migliore dell'$O\left(n^{-\frac{1}{2}}\right)$ ottenuto con un metodo Monte Carlo.

Tuttavia, il valore di quest'ultimo come metodo computazionale consiste nel fatto che tale ordine di convergenza viene mantenuto a prescindere dal dominio di integrazione. Infatti, la procedura precedentemente descritta può essere facilmente adattata per stimare un integrale sopra $(0,1)^d$ (o anche $\mathbb{R}^d$ stesso) per qualsiasi valore di d. Chiaramente, cambiando la dimensione del problema, cambia anche f e, di conseguenza, anche σ_f, ma l'errore standard resterà della forma $\sigma_f/\sqrt{n}$ e quindi l'ordine di convergenza rimarrà $O\left(n^{-\frac{1}{2}}\right)$. Invece, per il metodo dei trapezi si può mostrare che l'ordine di convergenza per un integrale in d dimensioni è $O\left(n^{-\frac{2}{d}}\right)$ e, più in generale, il peggioramento dell'ordine di convergenza con l'aumentare della dimensione è tipico di tutti i metodi di integrazione deterministici. Quindi, i metodi Monte Carlo sono la miglior scelta per la valutazione di integrali in spazi ad alta dimensione.

Per completare il quadro, non ci resta che descrivere come campionare i valori $\{ u_i \}_{i=1}^n$, che sarà ciò su cui ci concentreremo nella prossima sezione. Prima di ciò, tuttavia, si desidera mostrare come l'integrazione Monte Carlo non sia limitata all'utilizzo di nodi generati da una

[1] Qui al denominatore usiamo $n-1$ per ottenere uno stimatore non distorto, come dimostrato da Pearson nel 1894.

distribuzione di tipo uniforme. Osserviamo che, dato l'integrale

$$\int_{\mathscr{X}} f(\mathbf{x})d\mathbf{x},$$

dove $\mathscr{X} \subseteq \mathbb{R}^d$, possiamo considerare una qualsiasi distribuzione con supporto $\mathscr{X}$, ovvero la cui pdf (probability density function) $h : \mathbb{R}^d \to [0, +\infty)$ sia strettamente positiva quasi ovunque su $\mathscr{X}$, e approssimare l'integrale come

$$\int_{\mathscr{X}} f(\mathbf{x})d\mathbf{x} = \int_{\mathscr{X}} f(\mathbf{x}) \cdot \frac{h(\mathbf{x})}{h(\mathbf{x})}d\mathbf{x} = \int_{\mathscr{X}} \frac{f(\mathbf{x})}{h(\mathbf{x})} \cdot h(\mathbf{x})d\mathbf{x} = \mathbb{E}\left[\frac{f(\mathbf{X})}{h(\mathbf{X})}\right] \approx \frac{1}{n}\sum_{i=1}^{n} \frac{f(\mathbf{x}_i)}{h(\mathbf{x}_i)},$$

dove i punti $\{\, \mathbf{x}_i \,\}_{i=1}^{n} \subset \mathbb{R}^d$ provengono da un campionamento indipendente della distribuzione la cui cdf (cumulative density function) è

$$H(\mathbf{x}) = \int_{\mathscr{S}} h(\mathbf{x})d\mathbf{x} \qquad \text{con} \qquad \mathscr{S} = (-\infty, x_1) \times (-\infty, x_2) \times \cdots \times (-\infty, x_d).$$

La scelta della distribuzione è libera, purché abbia il supporto adeguato, tuttavia è necessario rimarcare che alcune scelte siano migliori di altre. Infatti, fissando una specifica pdf h si va a modificare il parametro di errore standard $\sigma_{f/h}$ e quindi a impattare la precisione della stima. Esiste una vasta letteratura relativa alla scelta della distribuzione ottimale per la generazione dei nodi di integrazione (importance sampling, Rao-Blackwellization), tuttavia nessuna di queste tecniche va a modificare il complessivo ordine di convergenza del metodo.

2.2 Simulazione di variabili aleatorie

Nella sezione precedente abbiamo descritto un metodo per stimare il valore di integrali usando nodi generati da una procedura non deterministica. Questo ci richiede di costruire un algoritmo che sia in grado di *simulare* una variabile aleatoria. Per simulare s'intende generarne realizzazioni andando a campionare dalla sua distribuzione di probabilità in maniera efficiente e affidabile.

Un importante distinguo deve essere fatto tra la simulazione di variabili aleatorie "semplici" e la generazione di vettori aleatori. Infatti, nel caso di quest'ultimi, è necessario simulare dei valori che tengano conto non solo delle distribuzioni delle singole componenti, che da ora in avanti chiameremo *marginali*, ma anche del come esse siano correlate tra di loro. Il concetto stesso di correlazione rappresenta un punto delicato che necessiterebbe una discussione più approfondita. In questa trattazione ci limiteremo a presentare un framework che permetta di

generare valori provenienti da una distribuzione multivariata assolutamente continua senza focalizzarci nel dettaglio sulla definizione formale degli oggetti matematici che andremo a sfruttare. Per maggiori dettagli relativi a questi aspetti teorici si rimanda ai testi di Nelsen (2006) e Brandimarte (2014).

L'algoritmo di simulazione di un vettore aleatorio d-dimensionale è una procedura che si compone di tre parti:

1. la generazione di un campione casuale $\mathbf{u} = (u_1, \ldots, u_d)^\intercal$ di valori appartenenti all'intervallo $(0,1)$ mediante un apposito algoritmo;
2. la trasformazione di tale campione usando una funzione $R_C^{-1} : [0,1]^d \to [0,1]^d$ che ha lo scopo di "applicare la correlazione" corretta tra le varie componenti del vettore aleatorio, ottenendo $\tilde{\mathbf{u}} = R_C^{-1}(\mathbf{u})$;
3. la trasformazione di ciascun valore $\tilde{u}_i$, $i = 1, \ldots, d$ mediante una particolare funzione in maniera da ottenere un valore proveniente dalla corrispondente marginale F_i.

Il primo step è ciò che viene effettuato da un *algoritmo di generazione*, di cui approfondiremo alcuni aspetti tra poco. Per ora basti sapere che simili algoritmi sono ad oggi implementati in maniera efficiente in tutti i principali software e librerie di calcolo scientifico. Il secondo passaggio è legato al concetto di *copula di una distribuzione multivariata*. Un breve approfondimento relativo alla teoria delle copule e alla funzione R_C^{-1} sopracitata, insieme all'elenco delle copule implementate numericamente in SFMQuantLib, può essere trovato nell'Appendice A. Infine, la terza fase consiste semplicemente nel trasformare un campione proveniente da una distribuzione di tipo uniforme sull'intervallo (0, 1) in un campione della marginale desiderata. Ciò può essere fatto per mezzo di una semplice trasformazione come enunciato nel seguente risultato.

Teorema 2.3. *Sia* $U \sim$Uniforme$(0,1)$ e F la cdf di una distribuzione di probabilità, allora la variabile aleatoria

$$Y = F^{-1}(U) \sim F,$$

dove F^{-1} denota l'inversa generalizzata di una funzione non decrescente

$$F^{-1}(u) = \inf\{\, x : F(x) \geq u \,\}.$$

Osservazione 2.4. *Dato che la cdf di una distribuzione di probabilità F è per definizione non decrescente e continua a destra, allora si ha che*

$$F^{-1}(u) = \inf\{\, x : F(x) \geq u \,\} = \min\{\, x : F(x) \geq u \,\}.$$

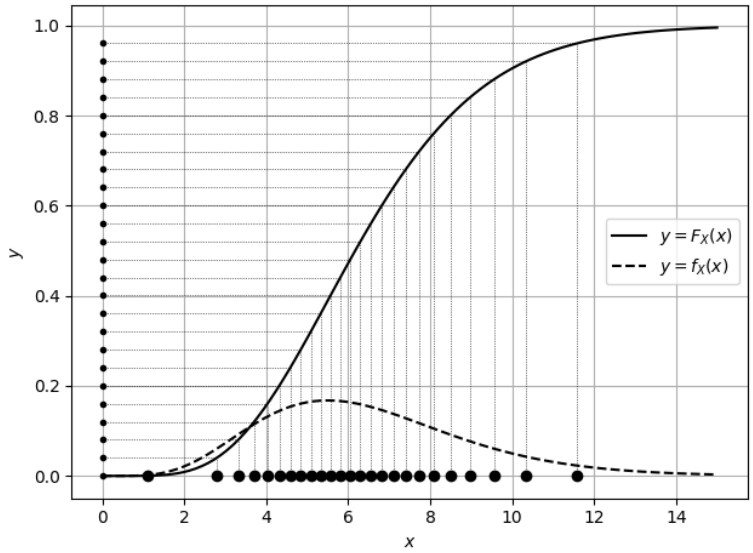

Figura 2.1 L'idea alla base del metodo di simulazione precedentemente descritto è legata alla capacità dell'inversa della cdf di "ridistribuire" i punti equi-spaziati sulla base della pdf.

Dimostrazione. L'enunciato è una diretta conseguenza della definizione di inversa generalizzata, infatti

$$\mathbb{P}(Y \leq z) = \mathbb{P}\left(F^{-1}(U) \leq z\right) = \mathbb{P}(U \leq F(z)) = F(z) \qquad \forall z \in \mathbb{R}.$$

□

Trasformare valori uniformemente distribuiti in $(0,1)$ usando l'inversa della cdf (vedi Fig. 2.1) non è l'unico modo per generare campioni provenienti da una determinata distribuzione di probabilità. Altre tecniche, come ad esempio i metodi Accept-Reject, hanno il vantaggio di funzionare senza richiedere di calcolare $F^{-1}(u)$ numericamente. Tuttavia, in pratica, esistono algoritmi efficienti capaci di valutare in maniera precisa l'inversa della cdf delle principali distribuzioni, quindi il metodo descritto resta valido nella maggior parte delle applicazioni.

Non ci resta che approfondire il funzionamento dell'algoritmo di generazione. Con esso, seguendo il testo di Glasserman, intendiamo un programma capace di generare una sequenza di vettori casuali $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \cdots \in (0,1)^d$ tali che

i) ogni $\mathbf{u}_i$ è la realizzazione di un vettore aleatorio $\mathbf{U}_i \sim \text{Uniforme}(0,1)^d$,

ii) i vettori $\mathbf{U}_i$ sono indipendenti.

Delle due proprietà, la più importante è la *ii*) dato che implica che qualsiasi coppia di valori scelta tra quelli generati dovrebbe essere scorrelata e, crucialmente, che il valore di $\mathbf{u}_i$ dovrebbe essere imprevedibile data la sequenza $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{i-1}$. Risulta dunque sorprendente scoprire che tutti gli algoritmi utilizzati per la generazione di numeri casuali sono invece degli algoritmi completamente deterministici e sequenziali. In SFMQuantLib sono stati implementati due diverse categorie di generatori di numeri casuali:

- i cosiddetti *generatori di numeri pseudo-casuali* che sfruttano operazioni di algebra modulare e permutazioni binarie per costruire una sequenza di valori deterministica e periodica;
- generatori basati su sequenze a bassa discrepanza che permettono di accelerare la convergenza della stima Monte Carlo e, per distinguerli dai precedenti, vengono spesso chiamati *generatori di numeri quasi-casuali*.

2.2.1 Generazione di numeri pseudo-casuali

Diamo una definizione di generatore di numeri pseudo-casuali.

Definizione 2.1 (Generatore di numeri pseudo-casuali)**.** Un generatore di numeri pseudo-casuali è un algoritmo che, partendo da un punto iniziale, comunemente detto *seed* (u_0), e una trasformazione D, produce una sequenza di valori $(u_1, \ldots, u_n)$ con $u_i = D(u_{i-1}) \in [0,1]$. Il vettore $(u_1, \ldots, u_n)$ riproduce le caratteristiche di un vero vettore di realizzazioni di variabili aleatorie $(U_1, \ldots, U_n)$ i.i.d. Uniforme$(0,1)$.

L'algoritmo descritto nella precedente definizione è deterministico, e il suo comportamento è prevedibile noto il punto di partenza u_0 e la trasformazione D. Questa è una caratteristica desiderabile in quanto ci consente di riprodurre con facilità gli esperimenti. Infatti, una volta fissato il valore del seed, la sequenza di numeri prodotta dall'algoritmo sarà sempre la stessa. Con "riproduce le caratteristiche" s'intende che nessun test statistico del tipo

$$H_0 : U_1, \ldots, U_n \quad \text{i.i.d.} \quad \text{Uniforme}(0,1),$$

venga rifiutato.

Diamo ora un esempio di un generatore di numeri pseudo-casuali: il *linear congruential generator* (LCG) è un sistema ricorsivo della forma

$$\begin{cases} x_{i+1} = ax_i + c \mod m \\ u_{i+1} = \frac{x_{i+1}}{m} \end{cases},$$

dove a, c e m sono interi con $0 < a \leq m$, $0 < c \leq m$, mentre in questo caso il seed è un valore x_0 positivo specificato dall'utente. In questo caso, la sequenza $u_1, u_2, \ldots$ oltre a essere deterministica sarà anche periodica.

Un simile algoritmo è troppo semplice e produce sequenze troppo prevedibili per essere utilizzato in pratica. Tuttavia, pone la base per una famiglia di generatori molto utilizzata, ovvero quella dei *permuted congruential generators* (PCG). Un PCG si differenzia da un LCG per tre caratteristiche:

- il parametro m è generalmente scelto come una potenza di 2 al fine di garantire la produzione di una sequenza di bit non distorta e migliorare la qualità dei numeri così prodotti;
- il risultato u_i non è restituito direttamente come output, ma subisce una fase di permutazione dei suoi bit.

I risultati ottenuti da alcuni algoritmi di generazione pseudo-casuali sono riportati in Fig. 2.2. Per le proprietà della distribuzione uniforme sull'ipercubo unitario $(0,1)^d$, è possibile ottenere N punti (scenari) di dimensione d, generando Nd valori e poi andandoli a raggruppare adeguatamente.

2.2.2 Sequenze a bassa discrepanza

Abbiamo già descritto come l'integrazione Monte Carlo sia fortemente legata al concetto di confrontare la massa o il volume di un evento e confrontarlo con quello dello spazio in cui si trova. Scegliere i nodi d'integrazione in maniera aleatoria ci garantisce che, con l'aumentare del loro numero, esploreremo in maniera sufficiente tutto lo spazio campionario e la precisione della stima migliorerà. Tuttavia, abbiamo visto come questo miglioramento sia molto lento e renda necessario l'impiego di enormi risorse dal punto di vista computazionale per ottenere risultati sufficientemente precisi. Inoltre, la stessa aleatorietà nella scelta dei nodi introduce una distorsione nella stima, in quanto andremo inevitabilmente a esplorare lo spazio campionario in maniera non perfettamente equa.

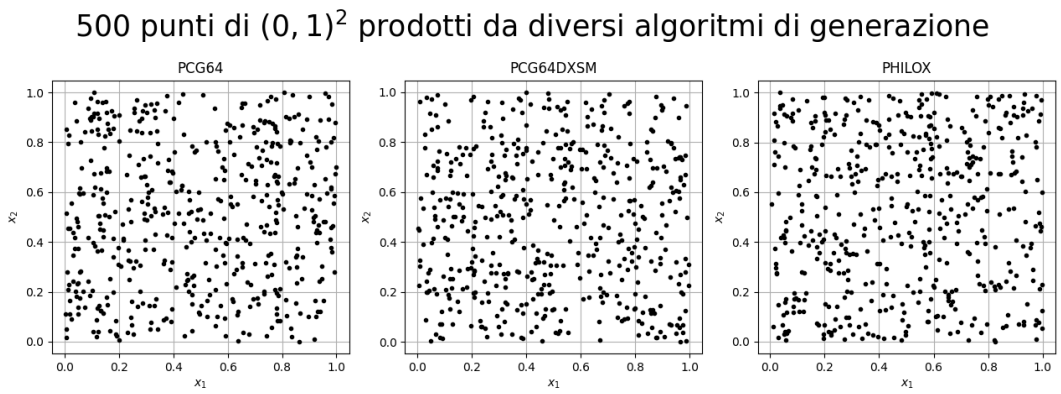

Figura 2.2 I punti dello spazio bidimensionale $(0,1)\times(0,1)$ generati dai diversi generatori di numeri pseudo-casuali implementati in SFMQuantLib. Da sinistra verso destra abbiamo i risultati dell'algoritmo PCG64, PCG64DXSM e PHILOX. Qualitativamente non si notano differenze significative tra i risultati prodotti dai tre algoritmi.

Questo ci suggerisce l'idea di andare a sondare lo spazio utilizzando una procedura che conservi le stesse caratteristiche esplorative garantite dal campionamento casuale, ma garantisca al tempo stesso che i punti estratti siano sufficientemente distanziati tra di loro. Le *sequenze a bassa discrepanza* sono sequenze deterministiche di punti appartenenti all'ipercubo $(0,1)^d$ contraddistinte dalla loro proprietà di andare a riempire lo spazio in maniera molto equa ed uniforme. Matematicamente, la discrepanza di una sequenza $\left\{\mathbf{x}^i\right\}_{i=1}^N \subset (0,1)^d$ può essere definita come

$$D\left(\mathbf{x}^1,\ldots,\mathbf{x}^N\right) = \sup_{\mathbf{x}\in(0,1)^d} \left|S_N\left(G_\mathbf{x}\right) - N x_1 x_2 \ldots x_d\right|,$$

ovvero l'estremo superiore della differenza tra il numero di elementi di una sequenza (S_N) che sono in un iper-rettangolo

$$G_\mathbf{x} = [0,x_1)\times[0,x_2)\times\cdots\times[0,x_d)$$

e il numero di punti che ci si aspetta di trovare in un insieme di tale area se la sequenza è ben distribuita nello spazio.

Molte sequenze a bassa discrepanza possono essere costruite a partire dalle *sequenze di Van der Corput*, che sono costruite nel modo seguente:

- si rappresenta un intero n in base b, dove b è un numero primo

$$n = (\cdots\delta_4\delta_3\delta_2\delta_1\delta_0)_b\,,$$

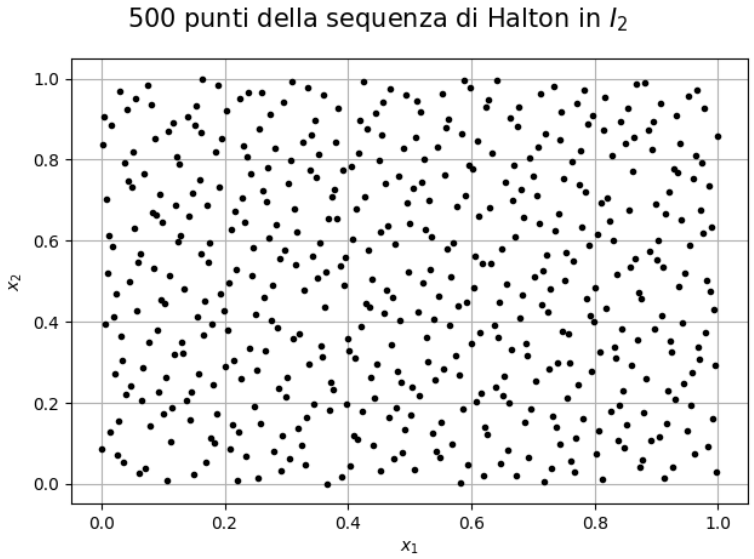

Figura 2.3 I primi 500 punti della sequenza di Halton bidimensionale. Se si compara il risultato con quelli della Fig. 2.2, è immediato notare che i punti della sequenza di Halton sono distribuiti in maniera più equilibrata nello spazio campionario.

o, più formalmente, scriviamo

$$n = \sum_{k=0}^{d} \delta_k b^k;$$

- l'n-esimo numero della sequenza in base b è

$$h(n,b) = \sum_{k=0}^{d} \delta_k b^{-(k+1)}.$$

Al fine di garantire che le sequenze ottenute siano effettivamente a bassa discrepanza (riempiano in maniera uniforme l'ipercubo unitario) è fondamentale utilizzare come base b un numero primo non eccessivamente grande.

Le sequenze di Halton in d dimensioni sono costruite associando un numero primo a ciascuna dimensione e generando la corrispondente sequenza di Van der Corput (vedi Fig. 2.3). Per via delle proprietà di b appena descritte, l'utilizzo di queste sequenze in spazi ad alta dimensione può risultare critico. Per questo motivo, spesso si preferisce utilizzare le sequenze di Sobol (Fig. 2.4). Queste sono costruite utilizzando esclusivamente sequenze di Van der Corput di base 2 e, per generare valori multidimensionali, esse vengono permutate utilizzando un algoritmo basato sulle radici di polinomi in aritmetica binaria. Sia le sequenze di Halton, sia le sequenze di Sobol sono state implementate e sono disponibili come metodi di generazione in

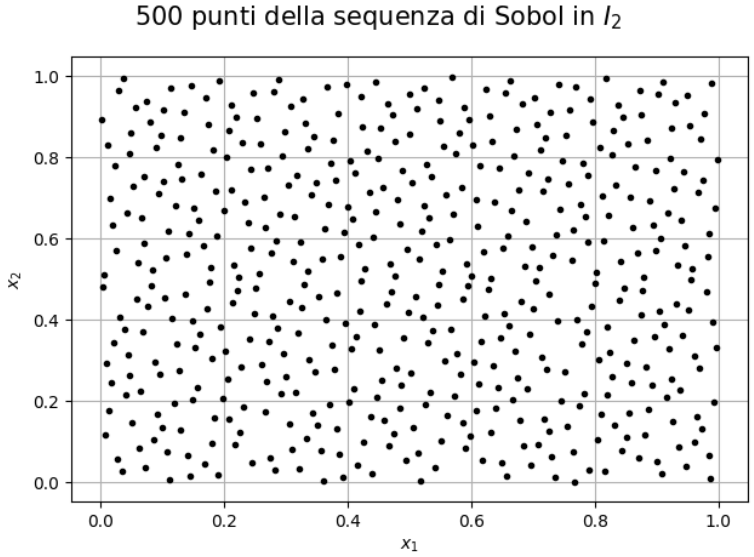

Figura 2.4 I primi 500 punti della sequenza di Sobol bidimensionale. Similmente al caso della Fig. 2.3, i punti della sequenza di Halton sono distribuiti in maniera equilibrata e sufficientemente spaziati tra di loro, a differenza dei risultati ottenuti usando generatori di numeri pseudo-casuali di Fig. 2.2.

SFMQuantLib.

Una sostanziale differenza tra l'impiego di un metodo Monte Carlo basato su sequenze di numeri pseudo-casuali e uno basato su sequenze a bassa discrepanza (quasi casuali) giace nel concetto di dimensionalità del problema considerato. Nel primo caso, per affrontare un problema a dimensione d, è sufficiente generare Nd valori e raggrupparli in N scenari; utilizzando il secondo approccio è invece necessario ricordare che la costruzione dei punti $\mathbf{u}_i \in \mathbb{R}^d$, $i = 1, \ldots, N$, dipende esplicitamente dalla dimensione del problema e sarebbe sbagliato costruirli andando a estrarre d elementi consecutivi dalla sequenza monodimensionale (gli effetti indesiderati prodotti da tale pratica sono riportati in Fig. 2.5).

Un ultimo punto delicato riguarda la quantità di punti da generare perché la sequenza (di Halton o di Sobol) incominci ad avere buone proprietà di bilanciamento e uniformità in alte dimensioni. Infatti, se non vengono generati abbastanza campioni si possono verificare fenomeni come quelli mostrati in Fig. 2.6 che comportano un sostanziale peggioramento della qualità della stima ottenuta.

Nonostante queste criticità, i numeri quasi-casuali hanno un effetto estremamente positivo sull'ordine di convergenza della stima Monte Carlo aumentandone la precisione.

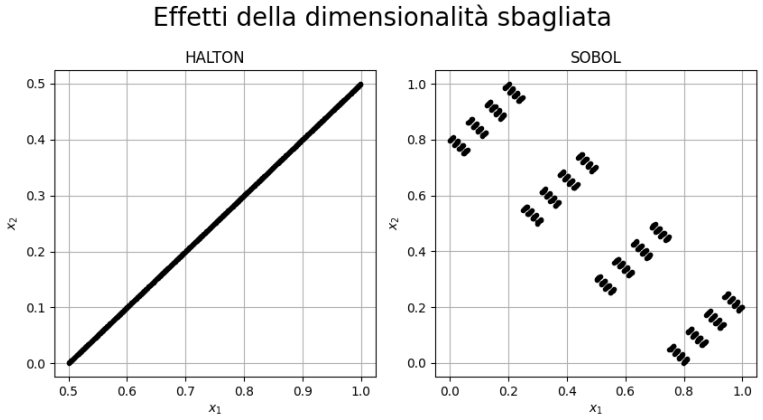

Figura 2.5 In questa figura vengono mostrati gli effetti indesiderati prodotti dalla mancata considerazione della corretta dimensionalità del problema. Queste figure sono state generate andando a campionare i primi 1000 valori della sequenza di Halton (sinistra) e Sobol (destra) e poi raggruppandoli a due a due per ottenere 500 punti bidimensionali. Chiaramente i risultati ottenuti non sono adatti ad effettuare una simulazione Monte Carlo.

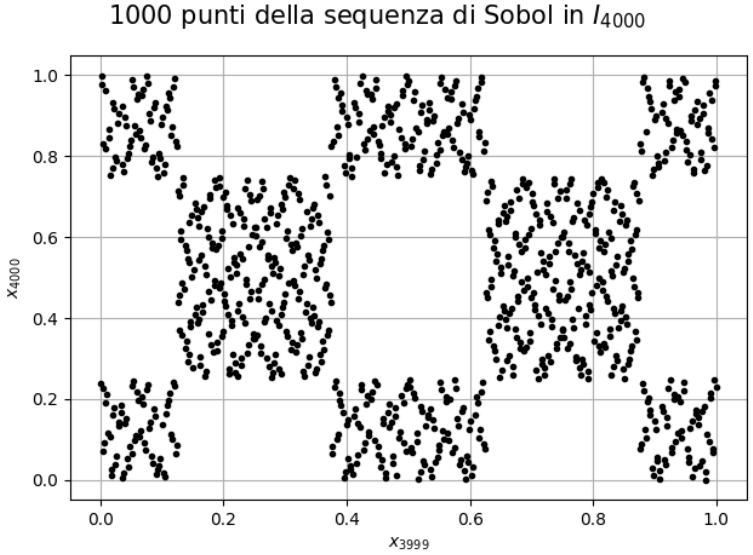

Figura 2.6 In questa figura sono rappresentati i valori assunti dalle ultime due componenti (x_{3999} e x_{4000}) dei primi 1000 punti della sequenza di Sobol in dimensione 4000. È immediato notare che la sequenza non riempie uniformemente tutto lo spazio e si vengono a formare dei "pattern" tra zone densamente campionate e zone vuote. Ciò introduce una forte distorsione (bias) nella stima Monte Carlo. Questo problema è risolvibile aumentando il numero dei punti campionati, tuttavia ciò comporta un maggior impiego di risorse computazionali.

Capitolo 3

Introduzione ai concetti finanziari

Dopo aver introdotto le fondamenta matematiche, passiamo ora alla descrizione del contesto finanziario nel quale è inserito il problema del pricing di un derivato. Nella Sezione 3.1 introdurremo i concetti di *valore temporale del denaro* e *interesse*, mentre in Sezione 3.2 verrà effettuata una presentazione degli strumenti finanziari elementari. In Sezione 3.3 introdurremo i derivati, per poi focalizzarci in particolare su alcune tipologie di opzioni nella Sezione 3.4. Infine, verrà effettuata una breve presentazione relativa ai certificati d'investimento nella Sezione 3.5.

3.1 Valore temporale del denaro

Questa sezione è stata realizzata seguendo il testo di Zastawniak and Capiński (2011). La finanza è lo studio dell'insieme delle tecniche relative alla gestione del denaro e, in particolare, all'allocazione di attività e passività nel corso del tempo. Alla base di essa troviamo l'importantissima tematica del prendere decisioni informate per gestire le risorse a nostra disposizione in maniera efficiente ed efficace anche in condizioni di forte incertezza. Tale definizione da sola è sufficiente a far intendere la rilevanza di temi come la caratterizzazione matematica di fenomeni aleatori e la capacità di stimare in maniera precisa quantità difficili da calcolare che abbiamo approfondito nei due capitoli precedenti. I due elementi alla base di questa disciplina sono il *denaro*, inteso come risorsa utilizzabile al fine di soddisfare i propri fabbisogni, e il *tempo*, cioè la dimensione fisica nella quale gestiamo suddetta risorsa. Di conseguenza, la trattazione della materia inizia proprio dall'analisi della relazione di questi, ovvero il *valore temporale del denaro*.

L'idea alla base è la seguente: il valore di una certa quantità di denaro oggi è diverso da quello relativo alla disponibilità della stessa quantità in futuro. Infatti, il valore di una risorsa monetaria dipende sia dalla sua quantità, sia dalla possibilità di ottenerne benefici trasformandola in un bene o utilizzandola per aumentare la propria ricchezza. Chiaramente, la garanzia di disporre di una certa risorsa un domani non permette di utilizzarla fino ad allora per soddisfare le proprie necessità, quindi il suo *valore attuale* sarà tanto più inferiore rispetto al suo *valore futuro* quanto più è lontana nel tempo la sua ricezione. La differenza tra queste due quantità viene generalmente chiamata *interesse* o anche *costo opportunità del capitale*.

Per fissare le idee, è utile considerare un piccolo esempio: supponiamo di depositare oggi, al tempo t_0, una certa somma di denaro, detta *principale* e denotata con P, su un conto bancario al fine di ottenere degli interessi; la quantità che troveremo sul conto al tempo t, $A(t)$, ovvero il valore futuro dell'investimento, sarà pari alla somma tra il denaro depositato inizialmente e l'interesse maturato nel periodo di tempo di durata $t - t_0$, ovvero

$$A(t) = P + I(t - t_0).$$

Chiaramente, lo stesso ragionamento si applica, a parti invertite, nel caso in cui si vada a richiedere un prestito.

Il valore temporale del denaro è strettamente collegato al concetto di *tasso d'interesse* che, per definizione, è la compensazione finanziaria relativa che viene ricevuta (o pagata) per l'utilizzo o il prestito di denaro nel tempo. Il tasso d'interesse viene solitamente espresso in termini di una percentuale annua (in tal caso è detto nominale), anche se può essere calcolato su periodi di tempo diversi (ad esempio trimestrali, mensili, ecc.) e può essere sia fisso, ovvero costante per l'intera durata del prestito o dell'investimento, che variabile. In questo caso, cambierà in base a indicatori come il tasso di riferimento di una banca centrale o altri fattori di mercato.

L'interesse può inoltre essere:

- *semplice*, se si basa solo sul principale. In questo caso, viene calcolato applicando il tasso di interesse a quest'ultimo moltiplicato per la durata dell'investimento. L'interesse accumulato non viene poi reinvestito e pertanto rimane costante. Indicando con P il principale, r_s il tasso d'interesse e $t - t_0$ la durata espressa in anni, il valore futuro del capitale iniziale alla fine del periodo considerato sarà pari a

$$A(t) = P[1 + r_s(t - t_0)];$$

- *composto*, se viene calcolato non solo sul capitale iniziale, ma anche sugli interessi accumulati precedentemente. Questo significa che l'interesse viene reinvestito periodicamente, contribuendo a generarne altri nel tempo. Inoltre, può essere composto in modo discreto oppure in modo continuo. Nel primo caso, l'ammontare disponibile viene reinvestito a periodi di tempo discreti, ad esempio annualmente, semestralmente, trimestralmente e così via. La formula per calcolare il valore futuro di un principale P con tasso d'interesse annuo r_m dopo un periodo di tempo di durata $t - t_0$ e composto per m volte è
$$A(t) = P\left(1 + \frac{r_m}{m}\right)^{m(t-t_0)}.$$
Supponendo che l'interesse venga calcolato e reinvestito infinite volte a intervalli di tempo di ampiezza tendente a zero (ovvero $m \to +\infty$), si ottiene l'interesse composto in modo continuo, per il quale vale che, dato il tasso r,
$$A(t) = Pe^{r(t-t_0)}.$$

La stessa procedura permette di calcolare il valore al tempo t_0 di una certo flusso di cassa che si verificherà in $t > t_0$. In generale, dato l'ammontare del flusso A, il suo valore attuale è dato dalla formula
$$\text{NPV} = A \cdot Z(t_0, t),$$
dove la quantità Z è detta *fattore di sconto* ed è l'inverso del fattore di crescita del valore del denaro nel tempo
$$Z(t_0,t) = \frac{1}{1 + r_s(t-t_0)} = \frac{1}{\left(1 + \frac{r_m}{m}\right)^{m(t-t_0)}} = e^{-r(t-t_0)}.$$

Quando si parla di *risk-free* in riferimento al tasso di interesse, ci si riferisce a quello di un investimento privo di rischio, ovvero il cui valore nel futuro è certo e noto in anticipo. Ogni altro investimento possibile avrà un rendimento pari a questo tasso maggiorato di *un premio per il rischio*, ovvero una componente commisurata al rischio (ad esempio di insolvenza e volatilità) intrinseco. Il tasso risk-free di riferimento per la valuta euro è il tasso ESTR (Euro Short-Term Rate), che viene calcolato sulla base dei tassi applicati ai prestiti interbancari di durata un giorno, condotti e conclusi il giorno lavorativo precedente tra i principali istituti finanziari europei.

3.2 Strumenti finanziari di base

Dopo aver introdotto le basi della disciplina finanziaria e approfondito il concetto di valore temporale del denaro, rivolgiamo la nostra attenzione verso le principali categorie di strumenti finanziari, anche detti securities, e le principali caratteristiche dei mercati su cui vengono scambiati. In particolar modo, ci focalizzeremo sul mondo azionario e sul mercato fixed-income delle obbligazioni.

3.2.1 Azioni e indici azionari

La prima tipologia di strumento finanziario che andiamo ad analizzare è quella delle *azioni*. Queste sono titoli rappresentativi di una quota di proprietà del patrimonio netto di un'impresa[1]. Il patrimonio netto è detto equity in inglese, da cui proviene il termine *equity market* per riferirsi al mercato azionario. Le azioni sono un esempio di titolo rischioso, in quanto il loro valore cambia costantemente nel corso del tempo ed è influenzato da numerosi fattori come dinamiche di mercato, performance dell'impresa, condizioni politiche, ecc., che lo rendono imprevedibile. Il valore di mercato delle azioni è sempre positivo e raggiunge lo zero nel caso in cui l'azienda emittente dichiari bancarotta o fallimento. Le imprese possono raccogliere capitale emettendo nuove azioni e vendendole sul mercato. I possessori del titolo sono detti *azionisti* e diventano a tutti gli effetti proprietari di una porzione dell'impresa. Così come le ha create, un'azienda può far scomparire delle azioni ricomprandole dagli azionisti.

Gli azionisti traggono un profitto dall'aumento del valore di un'azione. Possiamo definire il ritorno dell'investimento nel periodo $(0,T)$ come

$$R = \frac{S(T) - S(0)}{S(0)},$$

dove $S(t)$ è il valore di mercato dell'azione al tempo t. Un'altra fonte di guadagno legata al possedimento di un'azione è costituita dalla distribuzione di *dividendi*, che sono somme di denaro distribuite dall'azienda a tutti i suoi azionisti, generalmente in seguito al raggiungimento di risultati positivi. Se l'azione stacca una certa quantità di dividendi D durante il periodo $(0,T)$, allora il ritorno dell'investimento dell'azionista è pari a

$$R = \frac{S(T) + D - S(0)}{S(0)}.$$

[1] Con patrimonio netto si intende la quantità di denaro rimanente dopo aver liquidato tutti gli asset dell'impresa e aver saldato i debiti.

È importante tenere conto dei dividendi distribuiti dato che, oltre ad avere un impatto sul rendimento dell'investimento, vanno ad avere effetti rilevanti anche sulle dinamiche del valore di mercato di un'azione. Infatti, successivamente alla distribuzione di un dividendo, il prezzo di un'azione cala di una quantità pari all'importo del dividendo distribuito.

Gli *indici* sono statistiche che misurano le variazioni di un insieme selezionato di valori, rappresentando così un indicatore della performance generale di un mercato. La maggior parte di essi sono relativi a una specifica area geografica, ma esistono anche indici sull'economia globale. Alcuni sono associati a un particolare settore produttivo, mentre altri sono relativi a mercati non finanziari come quello immobiliare. Nel contesto del mercato azionario, un *indice azionario* è composto da un portafoglio di azioni selezionate in base a criteri specifici. Esistono anche indici relativi a metriche di mercato come tassi di interesse o volatilità. Tra gli indici azionari più noti troviamo l'*EUROSTOX*, che traccia le prestazioni delle principali società quotate nelle borse europee, e il *Dow Jones Industrial Average* (DJIA), che traccia 30 delle maggiori aziende statunitensi. In Italia, il *Financial Times Stock Exchange, Milano Indice di Borsa* (FTSE-MIB) è l'indice di riferimento per il mercato azionario, includendo le 40 aziende più liquide e capitalizzate quotate in Borsa Italiana.

In generale, il valore di un indice azionario è dato dalla formula

$$I(t) = \frac{1}{D} \sum_{k=1}^{m} w_k S_k(t),$$

dove $S_1, \ldots, S_m$ sono i prezzi di mercato delle azioni di diverse aziende, $w_1, \ldots, w_m$ sono i pesi di ciascun'azione nell'indice e D è detto divisore. La scelta dei pesi attribuisce a un indice un determinato significato. In particolare, distinguiamo due categorie di indici sulla base della scelta dei pesi:

- indici ponderati per prezzo, in cui i pesi sono tutti pari a 1 e danno peso ai titoli in base al loro prezzo per azione, indipendentemente dalla dimensione complessiva dell'azienda;
- indici ponderati per capitalizzazione di mercato, in cui i pesi assegnati si basano sulla quantità totale di azioni in circolazione sul mercato.

L'EUROSTOX e il FTSE-MIB sono indici ponderati per capitalizzazione di mercato, quindi il loro andamento è più rappresentativo di quello complessivo dei mercati di riferimento, mentre il DJIA è un esempio di indice ponderato per prezzo e rappresenta le performance di uno specifico portafoglio composto da un'azione per ognuna delle 30 aziende considerate. Infine, il divisore gioca il ruolo di parametro di normalizzazione. Viene inizialmente scelto per fare sì che l'indice

parta da un valore specifico e, successivamente, modificato adeguatamente per neutralizzare gli effetti di operazioni di modifica dell'indice (sostituzione di una o più delle aziende considerate con altre) o di fenomeni come acquisizioni e fusioni di aziende, che andrebbero ad alterarne bruscamente e improvvisamente il valore.

3.2.2 Obbligazioni

Un'*obbligazione*, o *bond*, è uno strumento finanziario di debito attraverso il quale un'entità può raccogliere fondi. L'emittente della stessa si impegna a rimborsare al detentore il capitale prestato più gli interessi entro una data futura prestabilita. Un'obbligazione è dunque un contratto che può essere descritto da tre caratteristiche:

- la maturità T, ovvero la data in cui viene saldato il debito;
- il *valore nominale* F, ovvero la somma che l'emittente si impegna a restituire al detentore del contratto alla maturità;
- il *tasso di cedola* c, ovvero il tasso annuale che, applicato al valore nominale, permette di definire l'ammontare dei pagamenti periodici relativi agli interessi, detti *cedole*, che vengono percepiti dal detentore dell'obbligazione.

Se $c = 0$, ovvero il bond non prevede il pagamento di alcuna cedola, è detto *zero-coupon bond* o più semplicemente *zero*.

Le obbligazioni sono considerate tra gli strumenti finanziari di base, perché possono essere scambiate sui mercati. Infatti, il possessore di un bond può vendere a un terzo il proprio titolo di credito nei confronti dell'entità emittente in cambio di una cifra di denaro. Dato che tali securities garantiscono al detentore una serie di entrate deterministiche, il mercato delle obbligazioni è detto *mercato fixed-income*. Data un'obbligazione che garantisce il pagamento di cedole negli istanti di tempo $t_1, t_2, \ldots, t_m = T$, possiamo calcolarne il valore al tempo t_0 usando la formula

$$B(t_0) = \sum_{k=1}^{m} C_k Z(t_0, t_k) + F Z(t_0, T),$$

dove $C_k = F \cdot c \cdot (t_k - t_{k-1})$ e Z è il fattore di sconto introdotto nella Sezione 3.1. È interessante osservare che, dato uno zero con valore nominale F, si ha

$$B(t_0) = Z(t_0, T)F \iff Z(t_0, T) = \frac{B(t_0)}{F},$$

che è una maniera alternativa per definire i fattori di sconto. Tale intuizione è alla base delle tecniche di *bootstrapping*, ovvero il processo di stima dei tassi d'interesse usando le quotazioni di bond o altre securities il cui valore dipende da essi.

La trattazione fino a qui presentata potrebbe suggerire che le obbligazioni non siano un titolo rischioso, tuttavia ciò risulta essere incorretto. Seppur le entrate garantite al possessore di un'obbligazione siano deterministiche, il valore dell'obbligazione è rischioso dato che dipende dal valore dei fattori di sconto e, conseguentemente, dal tasso d'interesse. Ciò rende le attività di trading su bond assai complesse. È infatti possibile caratterizzare metriche di sensibilità (duration, convexity, dollar-duration) che, insieme all'impiego di derivati come contratti swap, permettono di costruire strategie di trading.

3.3 Derivati

Nelle sezioni precedenti sono state fornite alcune nozioni fondamentali nell'ambito finanziario e descritte le principali categorie di strumenti scambiati sui mercati. Tutto ciò ci permette ora di dare un senso al concetto di *titolo derivato*, che è il principale interesse di questa trattazione nonché il focus dell'*ingegneria finanziaria*. Per derivato s'intende uno strumento finanziario il cui valore è determinato da quello di una o più variabili, comunemente dette *sottostanti*. Il sottostante può dunque essere un'azione, un indice, un tasso d'interesse o una materia prima, ma esistono anche derivati *scritti* su attività non finanziarie, come il meteo o l'avvenimento di eventi catastrofici. I titoli derivati sono utilizzati principalmente per tre scopi:

- la copertura del rischio (*hedging*), che permette di proteggere il valore di un investimento da fluttuazioni indesiderate dei prezzi;
- la speculazione, che mira a trarre profitto dalle variazioni di prezzo dell'attività sottostante (leva finanziaria);
- l'arbitraggio, ovvero lo sfruttamento di discrepanze di prezzo per ottenere un guadagno senza rischio, che approfondiremo nel capitolo successivo.

I derivati possono essere negoziati sia in mercati regolamentati, con contratti standardizzati, sia *over-the-counter* (OTC), dove le parti possono personalizzare i termini dell'accordo.

Da qui in avanti ci focalizzeremo principalmente su derivati scritti su equity, ovvero quelli aventi come sottostante azioni o indici azionari. Dal punto di vista formale, definiamo un derivato nel modo seguente.

Definizione 3.1 (Derivato)**.** Consideriamo un mercato su cui venga scambiato un asset, il cui valore di mercato è descritto da un processo stocastico S. Un *derivato* con maturità T è un contratto, valido fino al tempo T, che garantisce al detentore un flusso di cassa il cui ammontare dipende da una funzione Φ, detta *payoff*, del processo S. In particolare, se si ha che $\Phi(S) = \Phi(S(T))$, il derivato è detto *semplice*.

Osservazione 3.1. *Dato che S risulta essere un processo stocastico, è naturale pensare a* $\Phi(S)$ *come a una variabile aleatoria.*

Esempio 3.1 (Contratto forward)**.** Un contratto forward è un accordo tra due parti in cui una delle due (posizione lunga) si impegna a comprare dall'altra (posizione corta) una determinata quantità N di una security alla maturità T a un prezzo prestabilito al momento della stipulazione del contratto $t_0 < T$. Tale prezzo è detto *prezzo forward* e si indica con $F(t_0, T)$. Questo contratto è un derivato, in quanto entrambe le parti osserveranno un flusso di cassa alla maturità il cui ammontare dipende dal prezzo della security al tempo T, $S(T)$. Una semplice analisi del contratto permette di individuare la funzione di payoff per il detentore del contratto che, per convenzione, si identifica sempre con la parte in posizione lunga. Tale funzione risulta essere

$$\Phi(S) = \Phi(S(T)) = N \cdot (S(T) - F(t_0, T)),$$

ovvero la differenza tra il valore di mercato della quantità di commodity prestabilita e il prezzo stabilito dal contratto. Chiaramente, tale derivato risulta essere semplice e, dato che la sua funzione di payoff è lineare in $S(T)$, è anche detto *lineare*.

Nelle sezioni successive ci focalizzeremo su alcune tipologie di derivati facenti parte della categoria dei derivati *non lineari*.

3.4 Opzioni

Questa sezione è stata realizzata seguendo il testo di Hull (1993). Un'*opzione* è un derivato che conferisce al possessore il diritto, ma non l'obbligo, di acquistare (in tal caso si tratta di una *opzione call*) o vendere (*opzione put*) un determinato bene o strumento finanziario, detto sottostante, a un prezzo prestabilito, noto come *prezzo strike*, entro una data specifica che è la maturità del contratto. Possiamo distinguere diverse tipologie di esercizio di un'opzione:

- stile *europeo* se è esclusivamente possibile esercitare l'opzione alla maturità;

- stile *americano* se è possibile esercitare il diritto in qualsiasi momento $t \leq T$ antecedente alla maturità;
- stile *bermuda*, se è possibile decidere di esercitare solo in un insieme di date specifiche o in determinati periodi.

In questa trattazione ci focalizzeremo principalmente su opzioni in stile europeo.

Il valore dell'opzione può essere inteso in diversi modi. Si può voler considerare il suo *valore intrinseco*, ovvero il payoff che si riceverebbe se il sottostante fosse al livello corrente alla maturità, oppure il *valore del tempo*, che è il valore dato dall'incertezza del sottostante ed attribuibile al tempo che manca alla scadenza. Un'opzione è detta essere:

- *in the money*, se il suo valore intrinseco è positivo, ovvero per una call se il prezzo spot è maggiore dello strike, per una put se è minore;
- *out of the money*, se il suo valore intrinseco è nullo e tutto il valore è dato dal tempo. Per una call questo avviene quando il prezzo spot del sottostante è minore dello strike, per una put viceversa;
- *at the money*, se il livello del sottostante è uguale allo strike.

La differenza fondamentale tra le opzioni e altri derivati consiste nel diritto di recesso del possessore, che non è obbligato ad acquistare o vendere il sottostante, ma può scegliere se farlo nel caso in cui risulti economicamente conveniente.

3.4.1 Opzioni vanilla

Le opzioni europee scritte su un unico sottostante sono, in generale, la forma più semplice di tale tipologia di contratto. Per questo motivo ci si riferisce ad esse con il termine *opzioni vanilla*.

Andiamo ora a caratterizzare il payoff di un'opzione vanilla. Sia T la maturità dell'opzione, K il suo prezzo strike e $S(t)$ il valore del sottostante al tempo $t \in [0,T]$. Una call viene esercitata se $S(T) > K$, ovvero se il prezzo spot del sottostante alla scadenza è maggiore dello strike. È dunque facile comprendere che il payoff di tale opzione è dato da:

$$\Phi_{\text{Vanilla Call}}(S) = \max\{S(T) - K, 0\} = (S(T) - K)\,\mathbf{1}_{(K,+\infty)}(S(T)),$$

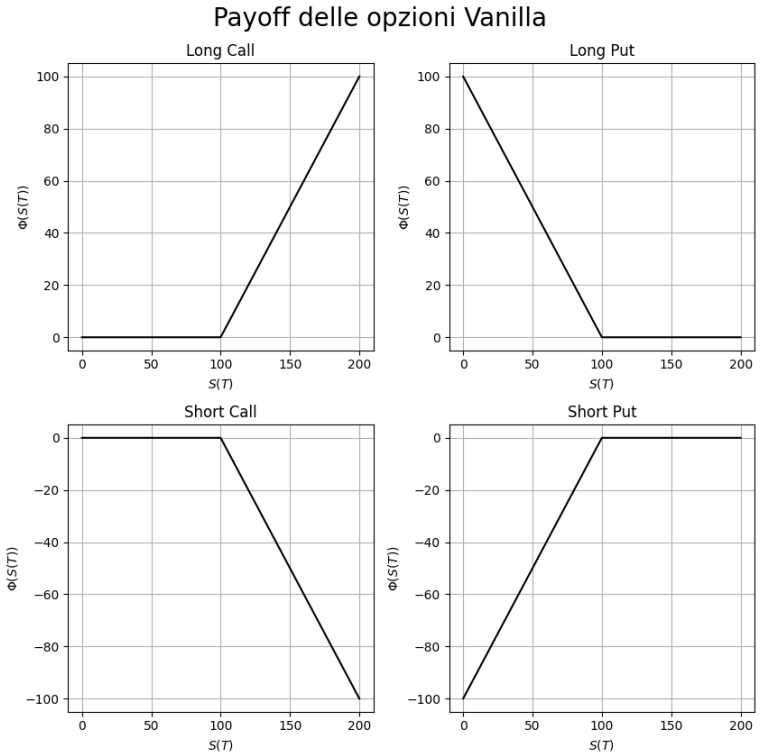

Figura 3.1 Resoconto dei payoff relativi a posizioni lunghe (prima riga) e corte (seconda riga) in opzioni vanilla di tipo call (prima colonna) e put (seconda colonna) con prezzo strike pari a $K = 100$.

dove $\mathbf{1}_A$ è l'indicatrice dell'insieme A. Similmente, un'opzione vanilla put viene esercitata solo se $S(T) < K$, dunque il suo payoff è:

$$\Phi_{\text{Vanilla Put}}(S) = \max\{K - S(T), 0\} = (K - S(T))\,\mathbf{1}_{(0,K)}(S(T)).$$

Chiaramente l'emittente dell'opzione percepisce un payoff che è l'opposto di quello percepito dalla posizione lunga. Di conseguenza, una delle due parti percepisce una ricompensa sempre non negativa, mentre per l'altra il payoff è al più pari a zero. Questo ci suggerisce che, affinché l'accordo sia equo, la posizione lunga dovrà pagare alla posizione corta una somma di denaro strettamente positiva per entrare in un simile contratto. Stabilire quale sia il giusto prezzo da pagare per comprare un derivato è il problema del pricing che affronteremo nel Capitolo 4.

Concludiamo questa breve analisi delle opzioni vanilla descrivendo la relazione che sussiste tra il prezzo di una call e quello di una put sullo stesso sottostante, aventi lo stesso prezzo strike K e la stessa maturità T. Immaginiamo di avere un portafoglio costituito da una posizione lunga nella prima opzione e una posizione corta nella seconda. Il payoff complessivo di tale

strategia è dato da

$$\begin{aligned}\Phi_{\text{Portafoglio}}(S) &= \Phi_{\text{Vanilla Call}}(S) - \Phi_{\text{Vanilla Put}}(S) \\ &= (S(T)-K)\,\mathbf{1}_{(K,+\infty)}(S(T)) - (K-S(T))\,\mathbf{1}_{(0,K)}(S(T)) \\ &= (S(T)-K)\left[\mathbf{1}_{(K,+\infty)}(S(T)) + \mathbf{1}_{(0,K)}(S(T))\right] \\ &= (S(T)-K)\,,\end{aligned}$$

Osserviamo che è possibile ottenere lo stesso payoff comprando al tempo t l'asset sottostante al prezzo $S(t)$ e chiedendo un prestito al tasso d'interesse privo di rischio un ammontare pari a $Ke^{-r(T-t)}$. Dato che il payoff al tempo T dei due portafogli è uguale in qualsiasi scenario, è lecito concludere che il loro valore sia lo stesso per tutti i $t \leq T$, ovvero

$$C(t) - P(t) = S(t) - Ke^{-r(T-t)} \qquad \forall t \leq T, \tag{3.1}$$

dove chiaramente $C(t)$ e $P(t)$ rappresentano il prezzo della call e della put al tempo t. L'Eq. (3.1) prende il nome di *put call parity*, ed è un utile strumento per verificare la qualità dei risultati ottenuti da un modello di pricing.

3.4.2 Opzioni binarie

Le *opzioni binarie* sono opzioni caratterizzate da un payoff discontinuo. Il loro funzionamento è il seguente: nel caso di una call (rispettivamente put) il detentore del contratto riceve il pagamento di un premio se il valore del sottostante, alla maturità del contratto T, si trova al di sopra (al di sotto) di una certa soglia, detta prezzo strike. Senza perdere di generalità, supponiamo che il premio sancito dal contratto sia unitario, quindi possiamo caratterizzare il payoff per una posizione lunga in un contratto call con strike pari a K come

$$\Phi_{\text{Binary Call}}(S(T)) = \mathbf{1}_{(K,+\infty)}(S(T)),$$

mentre per una put

$$\Phi_{\text{Binary Put}}(S(T)) = \mathbf{1}_{(0,K)}(S(T)).$$

A differenza dei contratti vanilla descritti precedentemente, queste opzioni sono meno diffuse su mercati regolamentati e generalmente i piccoli investitori non vi hanno accesso per via della loro elevata rischiosità. Infatti, anche piccole variazioni nel valore del sottostante possono portare a brusche modifiche nel rendimento dell'investimento.

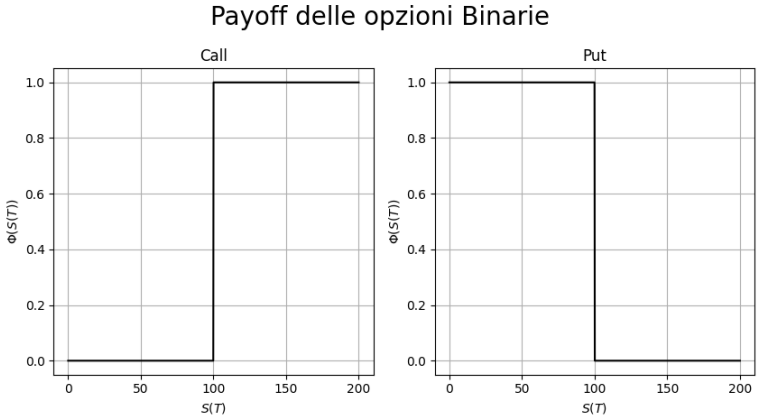

Figura 3.2 La figura riporta i grafici relativi al payoff di posizioni lunghe in opzioni binarie di tipo call (sinistra) e put (destra). I payoff relativi alle posizioni corte possono essere ottenuti invertendo quelli rappresentati (vedi Fig. 3.1).

3.4.3 Opzioni barriera

Le *opzioni barriera* sono opzioni il cui payoff dipende dal raggiungimento da parte del sottostante di determinate soglie in un certo periodo di tempo. Una grande varietà di simili strumenti viene regolarmente scambiata su mercati over-the-counter. L'interesse in questi contratti è legato alla loro caratteristica di essere meno costosi rispetto alle controparti prive di barriera.

Le opzioni barriera più semplici si dividono in opzioni knock-in e opzioni knock-out. Queste ultime sono contratti che smettono di esistere quando il valore del sottostante raggiunge un certo livello barriera; al contrario, una knock-in inizia a valere solo a partire dal momento in cui il sottostante raggiunge una certa soglia. Un esempio relativo al funzionamento di un'opzione knock-out è rappresentato in Fig. 3.3. Oltre a queste due tipologie esistono opzioni barriera più complesse, caratterizzate dalla coesistenza di multiple soglie di knock-in e knock-out, come l'esempio di Fig. 3.4. Al di là dei vincoli legati al raggiungimento dei livelli barriera, il payoff di un'opzione di questo tipo può assumere svariate forme: vanilla, binario, asiatico, ecc. Anche le modalità di esercizio sono molteplici tuttavia, come anticipato in precedenza, in seguito ci focalizzeremo su contratti in stile europeo.

3.5 Certificati d'investimento

Concludiamo questo capitolo dedicato al mondo finanziario introducendo una tipologia di derivato non molto conosciuta, ma di grande interesse per l'area mercati del Gruppo Sella: i

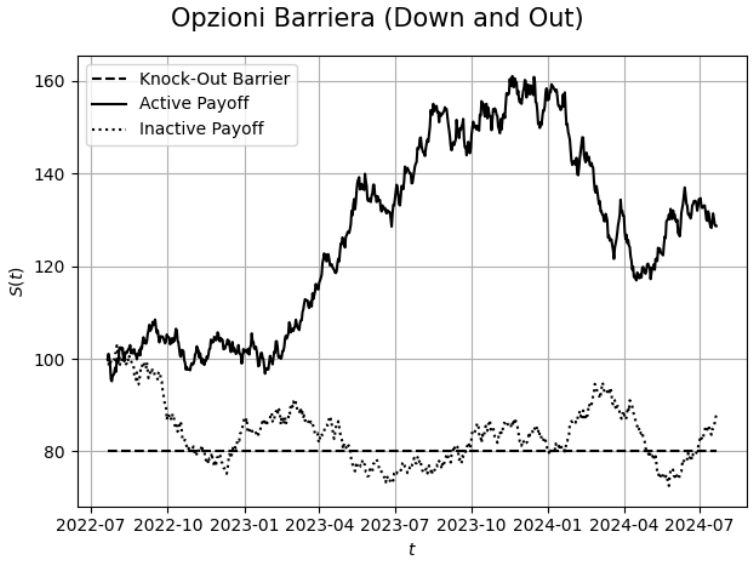

Figura 3.3 Il funzionamento di un'opzione barriera knock-out. La traiettoria più scura non tocca la barriera, quindi il contratto resta attivo. Al contrario, la traiettoria puntinata tocca il livello barriera pari a 80, quindi il contratto perde validità. L'immagine è stata prodotta sfruttando le funzionalità implementate in SFMQuantLib, in particolare le traiettorie appartengono a un moto browniano geometrico con valore iniziale $S(0) = 100$, parametro di drift $10\%, volatilit 20\%$, algoritmo di generazione PCG64 e seed pari a 14.

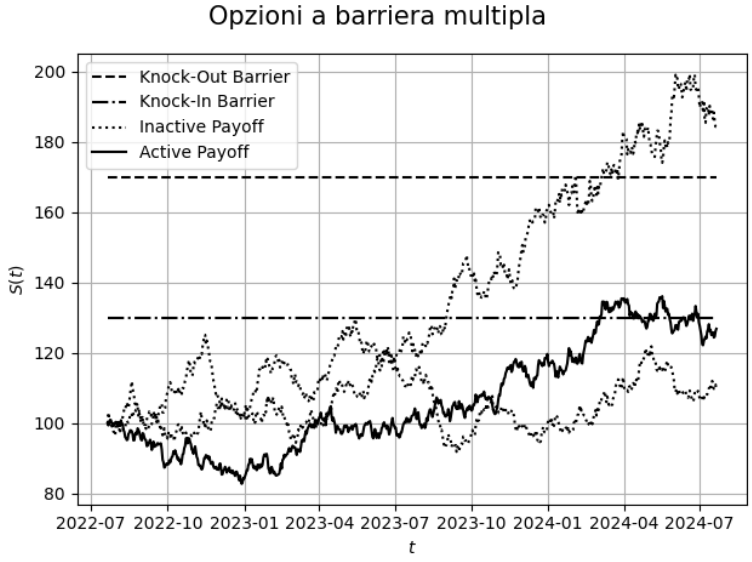

Figura 3.4 Funzionamento di un'opzione a barriera multipla. Essa è caratterizzata da una barriera knock-in di livello 130 e da una seconda barriera knock-out di livello 170. Il grafico è stato realizzato sfruttando le funzionalità implementate in SFMQuantLib, in particolare le traiettorie appartengono a un moto browniano geometrico con valore iniziale $S(0) = 100$, parametro di drift $10\%, volatilit 20\%$, algoritmo di generazione PCG64 e seed pari a 3.

certificati d'investimento. Per presentare questa tipologia di derivato faremo riferimento al libro di Bellelli (2021). Essi sono derivati cartolarizzati, ovvero una combinazione di contratti finanziari incorporati in un titolo, negoziabile come un'azione. Questi prodotti sono emessi da istituzioni finanziarie che si assumono l'obbligo del pagamento dei flussi di cassa dovuti. I certificati esistenti sul mercato possono essere divisi in varie categorie, a seconda della presenza o meno di protezione del capitale investito:

- i *certificati a capitale protetto* offrono la possibilità di investire in attività finanziarie garantendo la tutela del capitale investito;
- i *certificati a capitale condizionatamente protetto* consentono l'esposizione a particolari asset offrendo una garanzia parziale del capitale, condizionata al non raggiungimento di determinati livelli barriera stabiliti all'emissione;
- i *certificati a capitale non protetto* replicano fedelmente le dinamiche dei sottostanti.

Di seguito ci focalizzeremo in particolar modo su alcune tipologie di certificati a capitale condizionatamente protetto. Oltre alla partecipazione all'andamento del sottostante, i certificati prevedono spesso anche il pagamento di cedole che possono essere sia deterministiche che legate al verificarsi di determinati eventi. In quest'ultimo caso esse possono essere dotate del cosiddetto *effetto memoria* che permette all'investitore di ricevere assieme al premio associato al raggiungimento di determinati livelli del sottostante anche eventuali cedole non pagate in precedenza.

Oltre che per il livello di protezione del capitale, è possibile distinguere diverse tipologie di certificati sulla base del numero di sottostanti e delle regole usate per stabilirne il payoff. Tra le svariate possibilità si possono avere certificati:

- *standard*, ovvero scritti su un singolo sottostante il cui valore di mercato ne determina direttamente il payoff;
- *worst of*, il cui rendimento dipende dall'andamento del sottostante con le performance peggiori tra quelli stabiliti alla stipulazione del contratto;
- *best of*, analogo al precedente ma che considera il sottostante con le performance migliori;
- *asian*, in cui il livello del sottostante non viene rilevato puntualmente all'emissione e alla scadenza del prodotto, ma anche durante la vita. Per esempio il valore finale del sottostante è ottenuto dalla media delle rilevazioni mensili.

L'acquisto di un certificato corrisponde all'investimento in uno zero-coupon bond (emesso dall'ente che sintetizza il derivato) e in un portafoglio di opzioni finanziarie costituito da posizioni sia lunghe che corte. Le opzioni presenti all'interno dei certificati possono essere vanilla, oppure "esotiche" come le opzioni binarie. In particolar modo, è frequente trovare all'interno dei certificati a capitale condizionatamente protetto delle opzioni barriera che permettono di coprirsi dalla discesa del sottostante fintanto che questo non raggiunge un limite prestabilito.

Infine, è possibile caratterizzare i certificati sulla base del payoff che garantiscono al detentore. Tra le principali caratteristiche di questa tipologia di contratto troviamo l'alto livello di personalizzazione di questi contratti che permettono di costruire un prodotto su misura per le esigenze dei clienti, quindi distinguere nettamente una tipologia di payoff da un'altra non è sempre immediato o significativo. Di seguito presenteremo tre tipologie di contratto tra le più diffuse sufficientemente distinte l'una dall'altra pur tenendo in conto che ciascuna di esse rappresenta in realtà una vera e propria famiglia di payoff accomunati dallo stesso obiettivo di massima.

3.5.1 Airbag

Il certificato Airbag rientra nella famiglia dei certificati a capitale parzialmente protetto. Consente al detentore di partecipare al rialzo del sottostante, eventualmente fino a un limite massimo prefissato detto *cap*, garantendo una protezione del proprio investimento fintanto che il valore dei sottostanti alla maturità restano al di sopra di una certa soglia. Anche nell'eventualità in cui tale soglia venga superata, le perdite vengono attenuate di un certo fattore detto *rapporto airbag*. Generalmente questo rapporto viene scelto in maniera tale che nel caso in cui il valore di mercato dell'azione scenda a zero, l'investitore perda la totalità del capitale investito (vedi Fig. 3.5 sinistra).

Oltre a questa versione "base" è possibile creare altre versioni del certificato airbag andando ad agire sulla leva finanziaria delle due braccia del payoff, come riportato in Fig. 3.5 sulla destra. In tal caso si rinuncia a parte della potenziale crescita del sottostante, mediante l'introduzione di un cap, per ottenere una leva maggiore per quanto riguarda la crescita e una più attenuata per quanto riguarda il ribasso, garantendo una protezione completa di una frazione del capitale investito.

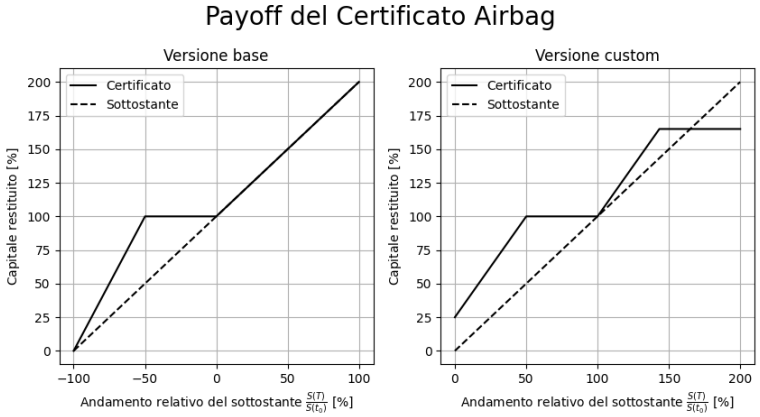

Figura 3.5 Confronto tra l'andamento del sottostante rispetto al rendimento di un certificato airbag. Si suppone che il certificato contenga uno zero-coupon bond con valore nominale pari a 100. L'immagine è stata realizzata sfruttando l'oggetto Certificate implementato in SFMQuantLib.

3.5.2 Bonus

I certificati d'investimento in stile *bonus* prevedono, oltre al rimborso del capitale investito, il pagamento di un premio, il cui ammontare è stabilito dal contratto, qualora il sottostante non sia mai sceso al di sotto di una certa soglia. Nel caso in cui la performance del sottostante sia superiore al bonus stabilito, viene restituita quest'ultima fino a un eventuale cap. Se il sottostante scende al di sotto del limite sancito dal contratto, le opzioni del certificato si disattivano e, alla maturità, viene restituita la percentuale del capitale investito sancita dal rendimento del sottostante. È possibile costruire una versione del certificate bonus a capitale completamente protetto che in caso di raggiungimento del livello barriera alla maturità si limita a restituire il capitale investito inizialmente.

Alcuni esempi di possibili payoff in stile bonus sono riportati in figura Fig. 3.6. Chiaramente l'esempio della figura di sinistra è caratterizzato da un comportamento meno rischioso rispetto alla versione a capitale non protetto della figura di destra.

3.5.3 Twin Win

Per certificato in stile *twin win* s'intende un contratto che permetta di trarre vantaggio sia dal (moderato) ribasso che dal rialzo di un sottostante. Esso può essere a capitale condizionatamente protetto, ovvero presentare delle barriere che se raggiunte comportano l'annullamento delle opzioni del certificato e la restituzione a scadenza della performance del sottostante, o a capitale protetto, ovvero comportare la sola restituzione del valore nominale dello zero-coupon bond

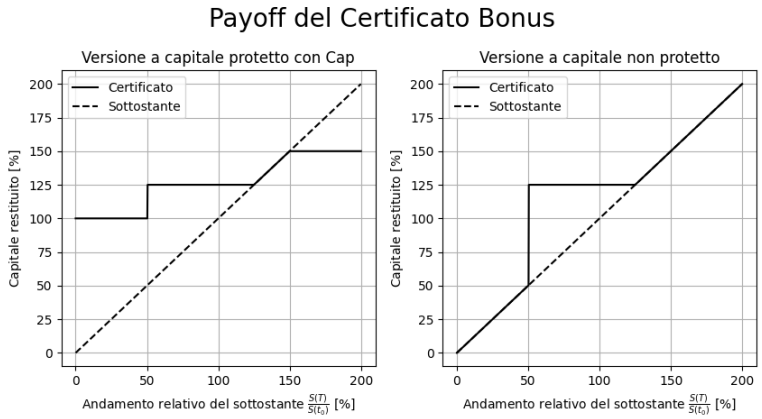

Figura 3.6 Confronto tra l'andamento del sottostante rispetto al rendimento di un certificato bonus. Si suppone che il certificato contenga uno zero-coupon bond con valore nominale pari a 100. L'immagine è stata realizzata sfruttando l'oggetto Certificate implementato in SFMQuantLib.

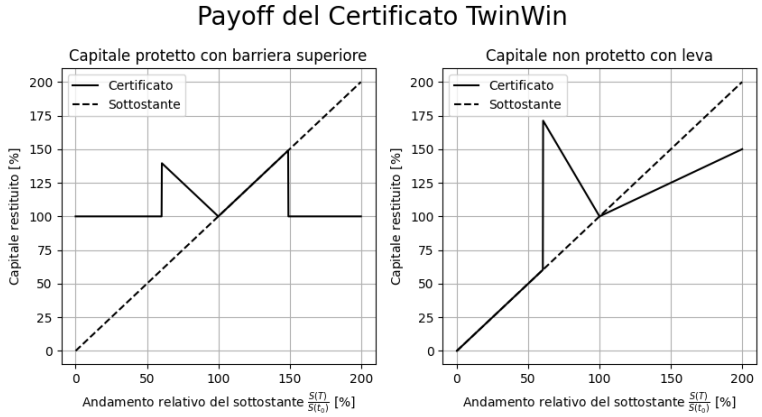

Figura 3.7 Confronto tra l'andamento del sottostante rispetto al rendimento di un certificato twin win. Si suppone che il certificato contenga uno zero-coupon bond con valore nominale pari a 100. L'immagine è stata realizzata sfruttando l'oggetto Certificate implementato in SFMQuantLib.

in caso di raggiungimento del livello barriera. In quest'ultimo caso il contratto viene anche detto certificato *double win* o *butterfly*. Tali barriere sono generalmente fissate a livelli che costituiscono un ribasso del valore del sottostante, tuttavia, in alcuni casi si hanno contratti caratterizzati anche da una barriera al rialzo e/o da un cap sul payoff.

Anche per questa tipologia di contratto è possibile personalizzare ampiamente il payoff come dimostrato in Fig. 3.7. L'immagine di sinistra rappresenta il payoff di un certificato twin win completamente protetto, dotato di barriera sia al rialzo che al ribasso, mentre quella di destra è propria di un certificato a capitale condizionatamente protetto, privo di barriera al rialzo ma con due diverse leve sulle braccia del payoff.

Capitolo 4

Modelli di pricing

Questo capitolo è stato realizzato seguendo i testi di Björk (2005) e Brandimarte (2017). Siamo finalmente pronti a discutere il tema centrale di questa tesi: *il pricing di un derivato*. L'obiettivo di questo capitolo sarà quello di presentare il problema e mostrare un approccio che permetta di affrontarlo andando a combinare la teoria relativa ai processi di Itô, presentati nel Capitolo 1, e i metodi Monte Carlo, introdotti nel Capitolo 2. Con *modello di pricing* s'intende un modello matematico della dinamica dei sottostanti che permetta di calcolare il cosiddetto *fair price* o *fair value* di un derivato, caratterizzato da un certo payoff Φ e un certo stile di esercizio (europeo, americano, bermuda). Prima di ciò bisogna tuttavia soffermarsi sul motivo per cui sia necessario sviluppare dei modelli che permettano di calcolare il valore di titoli finanziari che, in molti casi, sono ottenibili tramite una semplice ricerca su Google. La risposta a questa domanda è molteplice:

- da un lato il processo di *calibrazione* di un modello di pricing, ovvero il processo di stima dei valori dei suoi parametri che meglio rappresentano i prezzi attualmente quotati, permette di ottenere informazioni relative a quantità non osservabili direttamente sul mercato e, conseguentemente, aiuta a comprenderne meglio l'attuale stato;
- una volta validato il funzionamento di un certo modello, questo può essere utilizzato per trovare dei titoli disallineati rispetto al loro fair value e sfruttare questa quotazione anomala per trarne profitto mediante opportune strategie di trading;
- dal punto di vista della gestione del rischio, i modelli di pricing permettono di studiare l'esposizione a determinati fattori di rischio;

- per derivati esotici disponibili esclusivamente sui mercati over-the-counter non si ha a disposizione una quotazione, quindi un modello di pricing può essere fondamentale nello stimare quale sia il prezzo giusto per un determinato contratto;
- nell'ottica di un'azienda che desidera sintetizzare in proprio un derivato per poi venderlo a terzi, è fondamentale avere una stima abbastanza accurata del valore di tale prodotto, così da poter scegliere un prezzo competitivo e soddisfacente per entrambe le parti.

In questa trattazione ci focalizzeremo su modelli di pricing che sfruttano metodi Monte Carlo per stimare il fair price dettato dalla logica di *valutazione neutrale al rischio*. Simili approcci sono caratterizzati dalle stesse debolezze di tutti i metodi di stima Monte Carlo (necessità di impiegare grandi risorse computazionali, ordine di convergenza basso), tuttavia la loro flessibilità li rende uno strumento molto utilizzato in diverse applicazioni pratiche. Nella Sezione 4.1 verrà fornita una breve infarinatura relativa ai concetti di arbitraggio, efficienza e completezza dei mercati, che fungono da ipotesi fondamentali nello sviluppo dell'approccio di valutazione neutrale al rischio presentato in Sezione 4.2, con particolare attenzione alla sua affinità ai metodi Monte Carlo. A seguire presenteremo il celebre modello di Black e Scholes in Sezione 4.3, analizzandone le caratteristiche e i fondamentali limiti che giustificano la formulazione di modelli a volatilità stocastica come quello di Heston analizzato in Sezione 4.4.

4.1 Arbitraggi, mercati efficienti e completi

Per comprendere il funzionamento di un modello di pricing è necessario descrivere alcuni concetti fondamentali relativi al funzionamento dei mercati finanziari. Questi possono essere formalizzati ed è possibile sviluppare una rigorosa teoria che descriva il problema del pricing di un derivato in maniera astratta. Tuttavia, in questa sezione ci limiteremo a presentare tali aspetti e le principali conseguenze di nostro interesse senza entrare nei dettagli più tecnici. Per un approfondimento relativo a tali aspetti si rimanda ai due testi di riferimento per questo capitolo.

4.1.1 Assenza di arbitraggi

Il primo pilastro concettuale per la caratterizzazione di un mercato finanziario dal punto di vista teorico risulta essere la nozione di *arbitraggio*. Con tale termine s'intende una strategia che permetta di ottenere un guadagno privo di rischio senza l'impiego di alcuna risorsa da parte dell'esecutore della stessa. Da notare è la differenza tra questo concetto e un investimento in un asset risk-free: entrambi permettono di ottenere un guadagno sicuro, ma per il secondo è

richiesto l'impiego di un capitale iniziale. Esistono diversi modi per definire un arbitraggio dal punto di vista formale e ancora più modi per realizzarlo in pratica. Tuttavia, da un punto di vista puramente astratto, simili strategie non dovrebbero esistere in un mercato ben funzionante (efficiente) o, quanto meno, esistere solo per un arco di tempo estremamente limitato prima che vengano annullate le condizioni che ne permettono l'esecuzione. Per capire meglio questo fatto è utile ragionare su un piccolo esempio.

Esempio 4.1 (Una strategia cross-market di arbitraggio istantaneo)**.** Supponiamo che il prezzo del petrolio su un certo mercato A sia di 87.20 € al barile. Su un altro mercato B la stessa commodity viene scambiata per 87.70 € al barile. Un trader può sfruttare a suo vantaggio un'opportunità di arbitraggio, legata al disallineamento del prezzo del petrolio tra i due mercati, eseguendo la seguente strategia:

1. Comprare un barile sul mercato A a 87.20 €;
2. Vendere il barile sul mercato B a 87.70 €;

Sommando tutti i flussi di cassa si ha

$$+87.70€ - 87.20€ = 0.50€,$$

ovvero un guadagno netto privo di rischio senza l'impiego di alcun capitale iniziale. La stessa strategia è scalabile a piacere: invece di comprare e vendere un singolo barile, il trader potrebbe scambiarne centinaia, aumentando così il proprio profitto. Tuttavia, così facendo egli/ella avrebbe un impatto sull'equilibrio dei due mercati coinvolti, andando ad aumentare la domanda di petrolio sul mercato A, causandone un innalzamento del prezzo in euro, e, contemporaneamente, aumentando l'offerta sul mercato B diminuendone la quotazione. Il risultato è che lo sfruttamento dell'opportunità di arbitraggio porta all'annullamento delle condizioni che la rendono possibile.

Un ragionamento simile può essere applicato a un qualsiasi arbitraggio, dato che lo sfruttamento di un disallineamento nei prezzi porterà sempre questi a tornare in equilibrio. Quindi è lecito assumere il seguente fatto.

Claim 4.1. *Un mercato perfettamente liquido e privo di frizioni interne (costi di transazione, spread bid-ask, ecc.), i cui partecipanti sono tutti operatori razionali aventi accesso alle stesse informazioni, è naturalmente privo di arbitraggi.*

Chiaramente tali assunzioni sono molto idealizzate. In realtà tutti i mercati sono caratterizzati da qualche tipo di frizione interna, senza considerare che esistono grandi asimmetrie

informative tra i suoi partecipanti e che questi agiscono in maniera non del tutto razionale. Tuttavia, se si spera di trovare un prezzo "giusto" per un derivato, è necessario assumere che non vi siano opportunità di arbitraggio. Infatti, l'impossibilità di costruire strategie che generino ricchezza dal nulla garantisce la possibilità di dare a ciascun titolo scambiato un valore intrinseco che dipenda dalla quantità di ricchezza ottenibile mediante esso.

4.1.2 Efficienza

Un'altra importante ipotesi relativa al funzionamento dei mercati, strettamente legata alla precedente, è quella relativa alla loro *efficienza*. Con ciò s'intende che i mercati da noi considerati siano dotati della seguente proprietà.

Claim 4.2 (Ipotesi di efficienza del mercato)**.** *Il prezzo di un asset scambiato sul mercato riflette tutte le informazioni disponibili relative ad esso.*

La conseguenza di questo fatto è sottile, ma di estrema rilevanza dal punto di vista modellistico. Possiamo riformulare l'ipotesi sopracitata nel seguente modo: l'andamento del valore di mercato di un asset nel corso del tempo, data una condizione iniziale $S(0) = S_0$, è ben rappresentabile da un *processo stocastico markoviano* come quelli descritti in Sezione 1.1.1. Infatti, per un simile processo stocastico, si ha che l'analisi del passato $\{ S(\tau) : \tau < t \}$ non fornisce informazioni utili a determinarne il futuro $S(t+\Delta t)$, $\Delta t > 0$, dato che esso risulta essere dipendente esclusivamente dal valore del processo al tempo t, $S(t)$. Supporre che il mercato che si sta studiando sia efficiente permette di giustificare la scelta di modellare l'andamento dei titoli scambiati usando processi stocastici e ci suggerisce la famiglia più adatta a descrivere le dinamiche di questi. Vista la naturale irregolarità dell'andamento dei titoli in borsa, è naturale scegliere come modello i *processi di Itô* caratterizzati nella Definizione 1.14 della Sezione 1.3.2: infatti, le loro traiettorie sono funzioni a variazioni quadratica limitata, continue ma non differenziabili quasi certamente. Seppur molto conveniente dal punto di vista matematico-modellistico, l'ipotesi di efficienza è un tema assai dibattuto dagli esperti nel settore. Alcuni studiosi sostengono che la digitalizzazione e l'accesso a grandi quantità di informazioni abbiano reso i mercati odierni per lo più efficienti, mentre altri obiettano che se tale ipotesi fosse vera, non sarebbe possibile ottenere consistentemente performance superiori a quelle del mercato come invece hanno dimostrato di saper fare *pochi* fondi e investitori di grande successo.

4.1.3 Completezza

Infine, assumeremo che il mercato considerato sia un mercato *completo*. Come nel caso di arbitraggio, esistono tanti modi di definire questa proprietà. Qui ne riportiamo una definizione molto semplice ma del tutto valida dal punto di vista formale. Per comprenderla appieno dobbiamo introdurre il concetto di strategia di *hedging dinamico*. Una strategia di hedging è un investimento in un portafoglio che ha lo scopo di ridurre l'esposizione a un rischio. L'hedging può essere statico o dinamico: nel primo caso si fa riferimento all'investimento in un portafoglio che poi rimane fisso senza subire aggiustamenti, mentre nel secondo caso s'intende una strategia che preveda di modificare progressivamente i pesi del portafoglio senza mai richiedere l'impiego di ulteriore capitale. Una strategia di hedging perfetto di una posizione in un derivato è un portafoglio la cui variazione di valore è uguale e opposta alla variazione di valore della posizione stessa.

Definizione 4.1. Sia $\mathscr{M} = (B, S_1, \ldots, S_m)$ un mercato su cui vengono scambiati uno zero-coupon bond B privo di rischio e una serie di asset rischiosi $S_1, \ldots, S_m$. $\mathscr{M}$ è *completo* se per ogni derivato X che viene scambiato è possibile costruire una strategia di hedging dinamico perfetto.

In un mercato completo è sempre possibile coprirsi completamente dai rischi legati all'avere una posizione in un derivato, quindi, in altre parole, è possibile sintetizzare tutti i derivati scambiati sul mercato utilizzando esclusivamente gli asset B e S_i, $i = 1, \ldots, m$, prendendo posizioni opposte rispetto a quelle dettate dalla strategia di hedging. Tale portafoglio è detto *portafoglio di replica*. Questo ci dice che, almeno dal punto di vista teorico, è possibile riformulare il problema del pricing di un derivato trasformandolo nel problema di trovare un adeguato portafoglio di replica. A questo punto, se si suppone assenza di arbitraggio, si ha che il prezzo del derivato deve necessariamente essere uguale a quello del portafoglio di replica. Se così non fosse, un trader potrebbe sfruttare il disallineamento, comprando il meno costoso e vendendo il più caro, per trarne un profitto privo di rischio e senza l'impiego di alcun capitale.

4.2 Valutazione neutrale al rischio

I concetti introdotti ci permettono finalmente di trovare una soluzione al problema del pricing di un derivato. Per fare questo, però, è necessario introdurre l'adeguato formalismo.

Consideriamo il mercato $\mathscr{M} = (B, S_1, \ldots, S_m)$, caratterizzato dalle seguenti dinamiche

$$\begin{cases} dB(t) = rB(t)dt \\ dS_i(t) = \mu_i(t, S_i(t))\,dt + \sigma_i(t, S_i(t))\,dW_i(t) \qquad i = 1, \ldots, m \end{cases} \tag{4.1}$$

dove r indica il tasso d'interesse privo di rischio con capitalizzazione continua (supposto noto), S_i è un processo di Itô con coefficiente di drift μ_i e di diffusione σ_i per ogni $i = 1, \ldots, m$, mentre $\mathbf{W} = (W_1, \ldots, W_m)^\intercal$ è un processo di Wiener m-dimensionale. Osserviamo che le dinamiche descritte sono coerenti con l'ipotesi di efficienza dei mercati. Supponiamo inoltre di conoscere il valore spot degli asset al tempo 0

$$B(0) = 1, \qquad S_i(0) = S_i^0,$$

da cui è immediato ricavare l'andamento dell'asset privo di rischio nel corso del tempo

$$B(t) = e^{rt}.$$

Definiamo un derivato in stile europeo X avente maturità T come un contratto che preveda il pagamento di una quantità di denaro sancita da una certa funzione payoff Φ_X alla maturità. Supporremo che Φ_X sia misurabile rispetto a $\mathscr{F}_T$, quindi che sia una variabile aleatoria. Considereremo payoff il cui valore dipenda dal valore assunto da uno o più sottostanti nell'intervallo $[0, T]$.

Ora siamo pronti a introdurre il concetto fondamentale della *valutazione neutrale al rischio*: dal punto di vista di un investitore la cui avversione al rischio è inesistente, il fair value di un derivato è pari al valore atteso del suo payoff adeguatamente scontato per tenere conto del valore temporale. Possiamo vedere questo principio dal punto di vista della costruzione di un gioco equo: se investire in un derivato ha un costo pari al valore attuale medio del payoff ricevuto, si ha che il profitto medio è pari a zero. La scelta di rappresentare un investimento mediante un gioco *equo* è frutto della visione neutrale al rischio: infatti un investitore con tale punto di vista non effettua distinzioni tra l'asset privo di rischio e le securities rischiose aspettandosi che mediamente queste crescano tutte allo stesso modo e che, di conseguenza, il profitto medio di un qualsiasi investimento sia pari a zero. A tal scopo definiamo i processi scontati

$$\begin{cases} B^*(t) = \frac{B(t)}{B(t)} = 1 = B^*(0) \\ S_i^*(t) = \frac{S_i(t)}{B(t)} = e^{-rt} S_i(t) \qquad i = 1, \ldots, m \end{cases}$$

In determinati contesti il fattore di sconto $B(t)$ è detto *numerario*. Usare l'asset privo di rischio

come numerario equivale all'andare a considerare i processi dei valori attuali netti delle varie securities scambiate sul mercato, tuttavia è possibile scegliere come numerario un qualsiasi asset (anche rischioso) tra quelli a disposizione. Come detto in precedenza, in un mondo privo di rischio le dinamiche di tutti gli asset sono equivalenti dal punto di vista del valore attuale netto e, osservando che per lo zero-coupon bond si ha che il processo scontato ha media costante, si conclude che le dinamiche risk-free di tutti gli altri titoli condividano la stessa proprietà.

Dal punto di vista formale, assumere il punto di vista di un investitore neutrale al rischio equivale all'effettuare un cambio di misura di probabilità. Nello specifico vogliamo trovare una misura $\mathbb{Q}$ equivalente a quella storica $\mathbb{P}$, $\mathbb{Q} \sim \mathbb{P}$[1], tale per cui si abbia che

$$\mathbb{E}^{\mathbb{Q}}[S_i^*(t)|\mathscr{F}_0] = S_i^*(0) = S_i^0 \qquad \forall t \in [0,T], \quad i = 1,\dots,m,$$

il che equivale a chiedere che i processi S_i^* siano delle *martingale* sotto la misura di probabilità $\mathbb{Q}$. La misura di probabilità $\mathbb{Q}$, se esiste, è detta *misura neutrale al rischio* o, più accuratamente, *misura equivalente di martingala*.

Esistenza e unicità di una misura equivalente di martingala per un determinato mercato sono conseguenza dell'assenza di arbitraggio e della completezza rispettivamente di quest'ultimo, come garantito dai due seguenti teoremi.

Teorema 4.3 (Primo teorema fondamentale del pricing neutrale al rischio)**.** *Un mercato $\mathscr{M}$ con misura di probabilità storica $\mathbb{P}$ è privo di arbitraggio se e solo se esiste una misura di probabilità $\mathbb{Q}$ tale che $\mathbb{Q} \sim \mathbb{P}$ e i processi*

$$S_i^*(t) = \frac{S_i(t)}{B(t)} = e^{-rt} S_i(t) \qquad i = 1,\dots,m$$

sono $\mathbb{Q}$-martingale.

Teorema 4.4 (Secondo teorema fondamentale del pricing neutrale al rischio)**.** *Sia $\mathscr{M}$ un mercato privo di arbitraggio. La misura equivalente di martingala $\mathbb{Q}$ è unica se e solo se il mercato è completo.*

Questi due risultati ci garantiscono che in un mercato privo di opportunità di arbitraggio e completo, l'esistenza di un'unica misura neutrale al rischio $\mathbb{Q}$ è garantita. Le dimostrazioni sono assai tecniche, dato che richiedono di formalizzare il concetto di arbitraggio e sfruttano numerosi risultati relativi alla teoria delle martingale, quindi non vengono qui riportate.

[1] Ricordiamo che due misure di probabilità sono *equivalenti* se hanno gli stessi elementi trascurabili.

Risulta spontaneo chiedersi se tali ipotesi siano sufficientemente realistiche e adatte a descrivere i mercati del mondo reale. Per quanto riguarda l'assenza di arbitraggio la questione è strettamente legata al dibattito relativo all'efficienza dei mercati: seppur quelli odierni risultino essere relativamente efficienti, in pratica esistono anomalie di mercato e imperfezioni che possono portare a opportunità di arbitraggio. La completezza del mercato, d'altro canto, presuppone che sia possibile coprire perfettamente qualsiasi rischio attraverso strumenti finanziari esistenti ma, in realtà, molti rischi sono difficili da coprire completamente, come, ad esempio, quelli sistemici o legati a eventi estremi e imprevedibili (i cosiddetti *cigni neri*). Quindi, nonostante queste ipotesi siano utili astrazioni per costruire modelli teorici, non sempre riflettono con precisione le condizioni reali dei mercati finanziari.

Siamo finalmente pronti a presentare la formula che permette di calcolare il fair-value di un derivato in un mercato privo d'arbitraggio e completo. Per la completezza di $\mathscr{M}$ si ha che il payoff del derivato Φ_X è descrivibile in termini di un opportuno portafoglio di replica

$$\Phi_X(\omega) = h_0 e^{rT} + \sum_{k=1}^{m} h_i S_i(\omega, T) = V_{\mathbf{h}}(\omega, T) \qquad \forall \omega \in \Omega^2, \tag{4.2}$$

con $\mathbf{h} = (h_0, h_1, \ldots, h_m)^\intercal \in \mathbb{R}^{m+1}$. Per il principio di assenza di arbitraggio, il valore di mercato del derivato al tempo $t \in [0,T]$ deve essere uguale a quello del portafoglio di replica, $V_{\mathbf{h}}(\omega, t)$, allo stesso tempo in qualsiasi scenario. Consideriamo ora il processo scontato

$$V^*_{\mathbf{h}}(\omega, T) = h_0 + \sum_{k=1}^{m} h_i S^*_i(\omega, T) = \frac{V_{\mathbf{h}}(\omega, T)}{B(T)}. \tag{4.3}$$

Dato che le martingale formano uno spazio vettoriale e il processo $V^*_{\mathbf{h}}(\omega, T)$ è combinazione lineare di $\mathbb{Q}$-martingale, è esso stesso una $\mathbb{Q}$-martingala. Infine, osserviamo che il valore iniziale dei processi $V_{\mathbf{h}}$ e $V^*_{\mathbf{h}}$ è dato da

$$V_{\mathbf{h}}(0) = V^*_{\mathbf{h}}(0) = h_0 + \sum_{k=1}^{m} h_i S^0_i. \tag{4.4}$$

Quindi si ha che il prezzo di mercato del contratto X al tempo 0, che denotiamo con $\Pi(X, 0)$,

[2] Per essere precisi, la completezza del mercato garantisce l'esistenza di una strategia di hedging dinamico perfetto, quindi i pesi $h_0, h_1, \ldots, h_m$ sono in realtà funzioni del tempo. Tuttavia, per l'analisi effettuata possiamo fingere che questi siano costanti senza perdere generalità.

soddisfa la seguente catena di uguaglianze

$$\begin{aligned}
\Pi(X,0) &= V_{\mathbf{h}}(0) && \text{(assenza di arbitraggio)}\\
&= V^*_{\mathbf{h}}(0) && \text{(per la Eq. (4.4))}\\
&= \mathbb{E}^{\mathbb{Q}}\left[V^*_{\mathbf{h}}(\omega,T)|\mathscr{F}_0\right] && \text{(proprietà di } \mathbb{Q}\text{-martingala)}\\
&= \mathbb{E}^{\mathbb{Q}}\left[\frac{V_{\mathbf{h}}(\omega,T)}{B(T)}\middle|\mathscr{F}_0\right] && \text{(per la Eq. (4.3))}\\
&= e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[V_{\mathbf{h}}(\omega,T)|\mathscr{F}_0\right] && \text{(definizione di } B(T)\text{)}\\
&= e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[\Phi_X(\omega)|\mathscr{F}_0\right] && \text{(per la Eq. (4.2)),}
\end{aligned}$$

in definitiva si ha

$$\Pi(X,0) = e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[\Phi_X(\omega)|\mathscr{F}_0\right]. \tag{4.5}$$

L'Eq. (4.5) è la *formula di valutazione neutrale al rischio* e afferma che il prezzo del derivato è pari al valore atteso attualizzato del suo payoff sotto la misura equivalente di martingala. Questo approccio ha il vantaggio di fornire il valore del prezzo senza richiedere la conoscenza delle componenti del vettore $\mathbf{h}$, ovvero permette di prezzare un derivato senza richiedere la costruzione di una strategia di hedging dinamico, che risulta essere un problema assai difficile da affrontare.

4.2.1 Pricing tramite metodi Monte Carlo

L'Eq. (4.5) permette di calcolare il prezzo di un derivato mediante il calcolo di un valore atteso. È naturale definire la stima Monte Carlo del fair value come la quantità

$$\hat{\Pi}(X,0) = e^{-rT}\left(\frac{1}{N}\sum_{k=1}^{N}\Phi^{\mathbb{Q}}_{Xk}\right), \tag{4.6}$$

dove $\Phi^{\mathbb{Q}}_{Xk}$, $k = 1,\ldots,N$, sono campionamenti indipendenti della distribuzione del payoff Φ_X sotto la misura equivalente di martingala $\mathbb{Q}$. Per valori di N sufficientemente elevati, la quantità $\hat{\Pi}(X,0)$ è una stima sufficientemente precisa della quantità $\Pi(X,0)$.

L'impiego di un approccio Monte Carlo risulta essere fondamentale dato che, in generale, è estremamente difficile caratterizzare la distribuzione di probabilità di Φ_X, mentre è possibile generare campioni da essa sfruttando la procedura seguente:

1. si sceglie un modello che descriva l'andamento del sottostante, ovvero si specificano i processi μ_i e σ_i dell'Eq. (4.1), e si ricavano le dinamiche nella misura neutrale al rischio;

2. si effettua la *calibrazione* del modello selezionato, cioè usando dei dati di mercato (generalmente le quotazioni di bond, contratti swap e opzioni call vanilla) si vanno a stimare i valori dei parametri che meglio rispecchiano l'attuale stato del mercato;

3. sfruttando i valori dei parametri ottenuti mediante la calibrazione, si vanno a generare N traiettorie approssimate dei processi stocastici che descrivono i sottostanti nelle dinamiche neutrali al rischio mediante un opportuno *schema di discretizzazione*;

4. per ciascuna traiettoria si va a calcolare il relativo payoff garantito dal derivato, ottenendo così un campione $\Phi^{\mathbb{Q}}_{X1}, \Phi^{\mathbb{Q}}_{X2}, \ldots, \Phi^{\mathbb{Q}}_{XN}$ proveniente dalla corretta distribuzione di probabilità.

Una volta ottenuto il campione, si calcola il prezzo del derivato usando la formula dell'Eq. (4.6).

Nei capitoli successivi approfondiremo tutti i passaggi di questa procedura, a partire dalla descrizione di alcuni modelli per le dinamiche del sottostante nel caso in cui questo sia un'azione o un indice azionario. Prima di ciò, tuttavia, proponiamo una breve riflessione sui punti di forza e le principali debolezze del pricing basato sui metodi Monte Carlo.

Il principale vantaggio di quest'approccio si trova nella sua estrema flessibilità. Infatti, la procedura appena descritta permette di stimare il prezzo di un qualsiasi derivato con stile di esercizio europeo purché si sia in grado di calcolarne il payoff. Non vengono infatti effettuate ipotesi sulla natura o sul numero dei sottostanti se non quella di poter descrivere il loro andamento nel corso del tempo mediante un opportuno processo stocastico e di essere capaci di generarne traiettorie in maniera sufficientemente precisa. Il metodo è inoltre adattabile in maniera tale da renderlo capace di prezzare derivati con stile di esercizio di tipo americano o bermuda. Scegliendo opportuni processi stocastici è possibile caratterizzare le dinamiche di una grande varietà di oggetti finanziari (azioni, tassi d'interesse, volatilità, ecc.) e modellare in modo attendibile fenomeni di mercato anche molto complessi. Inoltre, seppur per i derivati più semplici esistano formule analitiche o semi-analitiche che permettono di calcolarne il prezzo sotto determinate ipotesi, per derivati esotici e complessi (come certificati path dependent, opzioni a barriera multipla o derivati scritti su più sottostanti) il pricing tramite metodi Monte Carlo rappresenta l'unica strategia percorribile in pratica.

Questa strategia non è però priva di debolezze. La prima di queste è senz'altro la necessità di effettuare una discretizzazione di un processo stocastico a tempo continuo, che risulta essere una procedura spesso caratterizzata da problemi di instabilità numerica che possono avere effetti estremamente nocivi sulla precisione della stima. Un'altra questione delicata è quella relativa al processo di calibrazione che corrisponde al risolvere un problema di ottimizzazione rispetto a un numero più o meno elevato di variabili decisionali (i parametri del modello). Come

approfondiremo in seguito, la funzione obiettivo è, in genere, un qualche tipo di funzione di loss che misuri la "distanza" tra le quotazioni osservate sul mercato e quelle predette dal modello data una scelta dei parametri. Per far sì che questo problema possa essere risolto in maniera numerica con un livello di precisione soddisfacente in un tempo ragionevole, non è possibile calcolare il prezzo del derivato tramite simulazione Monte Carlo ed è necessario ricorrere a modelli che permettano di calcolare il prezzo di almeno alcuni derivati semplici in maniera analitica o semi-analitica, e ciò limita fortemente le dinamiche effettivamente utilizzabili.

4.3 Il modello di Black-Scholes

Il primo modello per descrivere l'andamento del valore di un'azione che andiamo ad analizzare è quello di Black-Scholes, che prende il nome da Fischer Black e Myron Scholes, i quali nel loro articolo del 1973 (vedi Black and Scholes (1973)) proposero un metodo per determinare il fair price delle opzioni vanilla basandosi su un insieme di semplici ipotesi. La stessa tecnica fu sviluppata in maniera indipendente da Robert Merton, che la pubblicò in un articolo nello stesso anno (vedi Merton (1973)). Questo modello è di estrema rilevanza dal punto di vista storico e rappresenta una vera e propria pietra miliare nell'ambito della finanza quantitativa per via della possibilità di effettuare il pricing di opzioni vanilla (o altre opzioni semplici) mediante l'impiego di una semplice formula analitica.

Nei loro lavori, Black, Scholes e Merton descrissero l'andamento del prezzo di mercato di un'azione mediante un *moto browniano geometrico*. Nello specifico, assunsero che il processo stocastico S del prezzo di mercato di un'azione, sotto la misura di probabilità storica $\mathbb{P}$, soddisfacesse la seguente SDE

$$dS(t) = \mu S(t)dt + \sigma S(t)dW^{\mathbb{P}}(t), \tag{4.7}$$

con condizione iniziale $S(0) = S_0 > 0$, $\mu, \sigma \in \mathbb{R}^+$. Questo modello, nella sua semplicità, riesce a descrivere in maniera abbastanza fedele l'andamento di un sottostante di tipo equity a partire dalla sua positività e il generale andamento irregolare come mostrato in precedenza in Fig. 1.2.

Per poter applicare la formula dell'Eq. (4.5) e la sua approssimazione numerica data dall'Eq. (4.6), dobbiamo ricavare le dinamiche del processo S sotto la misura neutrale al rischio $\mathbb{Q}$. Per fare questo, richiamiamo due risultati teorici che permettono di passare agevolmente dalla misura storica alla misura equivalente di martingala.

Teorema 4.5 (Derivata di Radon-Nikodym)**.** *Siano $\mathbb{P}$ e $\mathbb{Q}$ due misure σ-finite ed equivalenti sullo stesso spazio di misura $(\Omega, \mathscr{F})$, allora esiste una funzione $f : \Omega \to (0, +\infty)$ unica $\mathbb{P}$-quasi*

certamente (equivalentemente $\mathbb{Q}$-quasi certamente) detta derivata di Radon-Nikodym *di $\mathbb{Q}$ rispetto a $\mathbb{P}$ tale che per ogni $A \in \mathscr{F}$*

$$\mathbb{Q}(A) = \int_A f d\mathbb{P}$$

e si denota $f = \frac{d\mathbb{Q}}{d\mathbb{P}}$.[3]

Teorema 4.6 (di Girsanov)**.** *Sia $(\Omega, \mathscr{F}, \mathbb{P}, \{\mathscr{F}_t\}_{t\geq 0})$ uno spazio di probabilità filtrato. Sia $W^{\mathbb{P}}(t)$ un moto browniano standard sotto $\mathbb{P}$, adattato alla filtrazione $\{\mathscr{F}_t\}_{t\geq 0}$. Supponiamo che $\theta(t)$ sia un processo adattato, integrabile e tale che soddisfi la cosiddetta* condizione di Novikov

$$\int_0^T \theta(s)^2 ds < \infty \quad \mathbb{P}\text{-quasi certamente per ogni } T > 0.$$

Definiamo una nuova misura di probabilità $\mathbb{Q}$ su $(\Omega, \mathscr{F})$ tramite la derivata di Radon-Nikodym

$$\left.\frac{d\mathbb{Q}}{d\mathbb{P}}\right|_{\mathscr{F}_t} = \exp\left(-\int_0^t \theta(s)\, dW^{\mathbb{P}}(s) - \frac{1}{2}\int_0^t \theta^2(s)\, ds\right).$$

Allora, sotto la misura $\mathbb{Q}$, il processo

$$W^{\mathbb{Q}}(t) = W^{\mathbb{P}}(t) + \int_0^t \theta(s)\, ds$$

è un moto browniano standard.

Il Teorema 4.6 è un risultato estremamente potente, dato che ci permette di ottenere le dinamiche di un processo di Itô nella misura neutrale al rischio a partire da quelle nella misura storica, come dimostreremo ora a partire dalle dinamiche Black-Scholes espresse nell'Eq. (4.7). Usando la formula di Itô (Teorema 1.21) si ricavano le dinamiche del processo scontato $S^*(t) = e^{-rt}S(t)$ che sono descritte dalla SDE

$$dS^*(t) = (\mu - r)S^*(t)dt + \sigma S^*(t)dW^{\mathbb{P}}(t).$$

A questo punto, per il teorema di Girsanov, le dinamiche del processo scontato in una nuova misura equivalente $\mathbb{Q}$ sono date dall'equazione

$$dS^*(t) = (\mu - r)S^*(t)dt + \sigma S^*(t)\left(dW^{\mathbb{Q}}(t) - \theta(t)dt\right),$$

[3] Una misura su uno spazio $(\Omega, \mathscr{F})$ è detta σ-finita se possiamo coprire l'intero spazio Ω con una successione numerabile di elementi di $\mathscr{F}$ ciascuno dei quali ha misura finita.

che con qualche semplice passaggio diventa

$$dS^*(t) = (\mu - r - \theta(t)\sigma)\, S^*(t)dt + \sigma S^*(t)dW^{\mathbb{Q}}(t).$$

Per definizione, la misura neutrale al rischio è la misura di probabilità in cui il processo stocastico S^* è una martingala. Una condizione sufficiente perché ciò avvenga è richiedere che il coefficiente di drift della forma differenziale di S^* sia costantemente nullo quasi certamente e ciò può essere ottenuto fissando

$$\theta(t) = \frac{\mu - r}{\sigma} = \lambda \qquad \forall t \in [0, +\infty) \quad \text{quasi certamente,}$$

che soddisfa la condizione di Novikov. La quantità λ è anche detta *prezzo di mercato del rischio* e rappresenta la compensazione in termini di rendimento atteso che i partecipanti al mercato si aspettano di ricevere per unità di rischio. Tale rapporto è anche noto come *Sharpe ratio* e ha importanti implicazioni nell'ambito di costruzione di portafogli efficienti in media-varianza. Fissato il processo θ, è immediato ricavare le dinamiche di S sotto la misura equivalente di martingala $\mathbb{Q}$ che risultano essere

$$\begin{aligned} dS(t) &= \mu S(t)dt + \sigma S(t)dW^{\mathbb{P}}(t) \\ &= \mu S(t)dt + \sigma S(t)\left(dW^{\mathbb{Q}}(t) - \lambda dt\right) \\ &= (\mu - \lambda\sigma)\, S(t)dt + \sigma S(t)dW^{\mathbb{Q}}(t) \\ &= rS(t)dt + \sigma S(t)dW^{\mathbb{Q}}(t), \end{aligned} \tag{4.8}$$

ovvero quelle di un moto browniano geometrico con drift pari al tasso d'interesse privo di rischio e volatilità pari a quella del processo nella misura storica $\mathbb{P}$.

4.3.1 Pricing per le opzioni call vanilla

Black e Scholes ricavarono una formula per prezzare le opzioni vanilla costruendo la strategia di hedging dinamico per un'opzione call e riconducendo tale problema a quello di trovare la soluzione di un'equazione del calore. Tuttavia, in questa trattazione seguiremo un altro ragionamento per ricavare tale formula che parte dalla valutazione neutrale al rischio di Eq. (4.5) e il Teorema 4.6. Ricordando che il payoff per un'opzione call vanilla con maturità T e strike K è pari a

$$\Phi_{\text{Vanilla Call}}(S) = \max\{S(T) - K, 0\} = (S(T) - K)\, \mathbf{1}_{(K,+\infty)}(S(T)),$$

si ha che il suo fair price al tempo $t < T$ è dato da

$$\begin{aligned}
C(t) &= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[\Phi_{\text{Vanilla Call}}(S)|\mathscr{F}_t\right] \\
&= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[(S(T)-K)\,\mathbf{1}_{(K,+\infty)}(S(T))\big|\mathscr{F}_t\right] \\
&= e^{-r(T-t)}\left\{\mathbb{E}^{\mathbb{Q}}\left[S(T)\mathbf{1}_{(K,+\infty)}(S(T))\big|\mathscr{F}_t\right] - \mathbb{E}^{\mathbb{Q}}\left[K\mathbf{1}_{(K,+\infty)}(S(T))\big|\mathscr{F}_t\right]\right\} \\
&= e^{-r(T-t)}\left\{\mathbb{E}^{\mathbb{Q}}\left[S(T)\mathbf{1}_{(K,+\infty)}(S(T))\big|\mathscr{F}_t\right] - K\mathbb{E}^{\mathbb{Q}}\left[\mathbf{1}_{(K,+\infty)}(S(T))\big|\mathscr{F}_t\right]\right\} \\
&= e^{-r(T-t)}\left\{\mathbb{E}^{\mathbb{Q}}\left[S(T)\mathbf{1}_{(K,+\infty)}(S(T))\big|\mathscr{F}_t\right] - K\mathbb{Q}\left(S(T)>K|\mathscr{F}_t\right)\right\} \\
&= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[S(T)\mathbf{1}_{(K,+\infty)}(S(T))\big|S(t)\right] - Ke^{-r(T-t)}\mathbb{Q}\left(S(T)>K|S(t)\right).
\end{aligned}$$

La probabilità nel secondo termine della sottrazione può essere facilmente calcolata usando le proprietà del moto browniano geometrico; infatti, data una variabile aleatoria $Z \sim \mathscr{N}(0,1)$, si ha

$$\begin{aligned}
\mathbb{Q}\left(S(T)>K|S(t)\right) &= \mathbb{Q}\left(S(t)e^{\left(r-\frac{1}{2}\sigma^2\right)(T-t)+\left(\sigma\sqrt{T-t}\right)Z} > K\right) \\
&= \mathbb{Q}\left(\left(\sigma\sqrt{T-t}\right)Z > \log\left(\frac{K}{S(t)}\right) - \left(r-\frac{1}{2}\sigma^2\right)(T-t)\right) \\
&= \mathbb{Q}\left(Z > \frac{\log\left(\frac{K}{S(t)}\right) - \left(r-\frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}\right) \\
&= \mathbb{Q}\left(Z \leq \frac{\log\left(\frac{S(t)}{K}\right) + \left(r-\frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}\right) \\
&= N(d_2),
\end{aligned}$$

dove N denota la cdf di una distribuzione normale standard e

$$d_2 = \frac{\log\left(\frac{S(t)}{K}\right) + \left(r-\frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}. \tag{4.9}$$

Il calcolo del valore atteso

$$\mathbb{E}^{\mathbb{Q}}\left[S(T)\mathbf{1}_{(K,+\infty)}(S(T))\big|S(t)\right],$$

risulta essere, invece, più complicato. Per esplicitarlo sfruttiamo un "trucco" legato a un cambio di misura probabilità. Consideriamo una nuova misura $\tilde{\mathbb{Q}}$ tale che

$$\left.\frac{d\tilde{\mathbb{Q}}}{d\mathbb{Q}}\right|_{\mathscr{F}_t} = \frac{S(T)}{S(t)e^{r(T-t)}} = e^{-\frac{1}{2}\sigma^2(T-t)+\int_t^T \sigma dW(s)}, \tag{4.10}$$

che soddisfa le ipotesi del teorema di Girsanov con $\theta(t) = -\sigma$. Quindi le dinamiche di S nella nuova misura $\tilde{\mathbb{Q}}$ sono date da

$$dS(t) = \left(r + \sigma^2\right) dt + \sigma dW^{\tilde{\mathbb{Q}}}(t).$$

Il cambio di misura appena descritto risulta essere vantaggioso dato che si ha

$$\begin{aligned}
\mathbb{E}^{\mathbb{Q}}\left[S(T)\mathbf{1}_{(K,+\infty)}(S(T))\middle|S(t)\right] &= \mathbb{E}^{\tilde{\mathbb{Q}}}\left[S(T)\mathbf{1}_{(K,+\infty)} \cdot \left.\frac{d\mathbb{Q}}{d\tilde{\mathbb{Q}}}\right|_{\mathscr{F}_t}\middle|S(t)\right] \\
&= \mathbb{E}^{\tilde{\mathbb{Q}}}\left[S(T)\mathbf{1}_{(K,+\infty)} \cdot \frac{S(t)e^{r(T-t)}}{S(T)}\middle|S(t)\right] \\
&= S(t)e^{r(T-t)}\mathbb{E}^{\tilde{\mathbb{Q}}}\left[\mathbf{1}_{(K,+\infty)}\middle|S(t)\right] \\
&= S(t)e^{r(T-t)}\tilde{\mathbb{Q}}\left(S(T) > K\right).
\end{aligned}$$

Eseguendo passaggi del tutto analoghi a quelli fatti in precedenza si ricava che

$$\tilde{\mathbb{Q}}\left(S(T) > K\right) = N(d_1),$$

con

$$d_1 = \frac{\log\left(\frac{S(t)}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}},$$

quindi in definitiva si ha

$$\begin{aligned}
C(t) &= e^{-r(T-t)}S(t)e^{r(T-t)}N(d_1) - Ke^{-r(T-t)}N(d_2) \\
&= S(t)N(d_1) - Ke^{-r(T-t)}N(d_2),
\end{aligned} \tag{4.11}$$

che è la celebre formula per il pricing di un'opzione call vanilla ricavata da Black, Scholes e Merton.

4.3.2 Proprietà e limiti del modello

Il grande vantaggio di questo modello giace nella sua semplicità: i parametri da stimare sono pochi e densi di significato dal punto di vista economico-finanziario, il processo stocastico può essere simulato in maniera relativamente semplice (come vedremo nel Capitolo 5) e la distribuzione di probabilità delle variabili $S(t)$ è nota, il che permette di ricavare in maniera abbastanza agevole formule per prezzare contratti semplici. Inoltre, la formula (4.11) è fondamentale nella

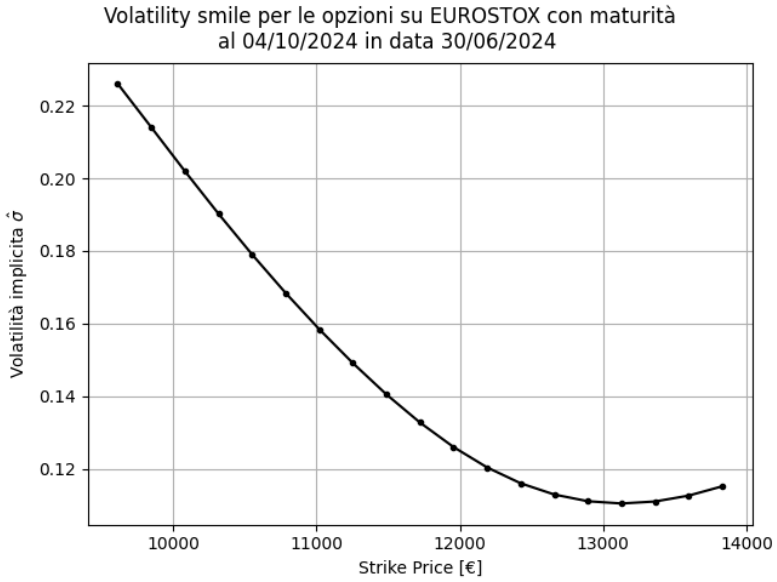

Figura 4.1 La relazione tra strike e volatilità implicita ricavata dai dati di mercato del 30 Giugno 2024 relativi alle opzioni con maturità il 4 Ottobre 2024 sull'indice EUROSTOX.

definizione del concetto di *volatilità implicita*, ovvero il livello di volatilità del sottostante che giustifica un determinato valore di mercato per un'opzione. Per stimare tale volatilità, in genere si ricorre ad approcci basati sulla soluzione di problemi di ottimizzazione del tipo

$$\hat{\sigma} = \arg\min_{\sigma} \sum_{k=1}^{N} \left(C(\sigma, T_k, K_k, r, S(0)) - P_k\right)^2,$$

dove P_k è il prezzo di mercato di un'opzione call vanilla con maturità T_k e strike K_k, r è il tasso d'interesse privo di rischio e $S(0)$ è il valore spot del sottostante.

La principale debolezza di questo modello sta proprio nella maniera in cui esso descrive la volatilità. Questa viene assunta costante nel tempo e uguale per tutti i diversi valori di strike, mentre nella maggior parte dei contesti reali la volatilità varia nel corso del tempo ed esiste una relazione non banale tra lo strike di un'opzione e la sua volatilità implicita. Nello specifico, il grafico strike-volatilità per una data maturità T è caratterizzato da una *skewness* negativa, ovvero è generalmente pendente verso il basso, e da una curvatura di secondo ordine positiva come mostrato in Fig. 4.1. Per questo si parla di *volatility smile* e il modello di Black-Scholes risulta essere poco realistico da questo punto di vista.

4.4 Il modello a volatilità stocastica di Heston

Le principali criticità del modello di Black-Scholes possono essere attenuate mediante l'impiego di un modello a *volatilità stocastica*, in cui la volatilità del prezzo dell'azione sottostante è modellata da un opportuno processo stocastico. Uno dei modelli a volatilità stocastica più noti e utilizzati è il modello di Heston che prende il nome dal matematico Steven Heston che, in un articolo del 1993 (vedi Heston (1993)), assunse che le dinamiche della varianza dell'asset (il quadrato della volatilità) fossero descrivibili mediante un processo di Cox-Ingersoll-Ross (vedi Sezione 1.3.2). Le dinamiche di prezzo e volatilità di un asset nella misura storica $\mathbb{P}$ sono dunque date dal seguente sistema di SDE:

$$\begin{cases} dS(t) = \mu S(t)\,dt + \sqrt{v(t)}S(t)\,dW_S^{\mathbb{P}}(t) \\ dv(t) = \kappa(\theta - v(t))\,dt + \xi\sqrt{v(t)}\,dW_v^{\mathbb{P}}(t) \\ \mathbb{E}^{\mathbb{P}}\left[dW(t)_S^{\mathbb{P}}\,dW(t)_v^{\mathbb{P}}\right] = \rho\,dt \end{cases} . \tag{4.12}$$

Le dinamiche del prezzo di un'azione sono dunque descritte da un moto browniano geometrico con drift pari a μ e coefficiente di diffusione $\sigma(t) = \sqrt{v(t)}$. L'ultima equazione è equivalente al dire che il processo

$$\mathbf{W}^{\mathbb{P}}(t) = \begin{pmatrix} W_S^{\mathbb{P}}(t) \\ W_v^{\mathbb{P}}(t) \end{pmatrix}.$$

è un moto browniano bi-dimensionale a incrementi correlati, con correlazione pari a ρ. Le dinamiche neutrali al rischio possono essere ricavate in maniera agevole usando il teorema di Girsanov (Teorema 4.6) seguendo un ragionamento analogo a quello fatto per il precedente modello ottenendo

$$\begin{cases} dS(t) = rS(t)\,dt + \sqrt{v(t)}S(t)\,dW_S^{\mathbb{Q}}(t) \\ dv(t) = \kappa(\theta - v(t))\,dt + \xi\sqrt{v(t)}\,dW_v^{\mathbb{Q}}(t) \\ \mathbb{E}^{\mathbb{Q}}\left[dW(t)_S^{\mathbb{Q}}\,dW(t)_v^{\mathbb{Q}}\right] = \rho\,dt \end{cases} .$$

È possibile osservare che anche in questo caso l'unico effetto del cambio di misura di probabilità è quello di scambiare il coefficiente di drift del processo S con il tasso d'interesse privo di rischio r. I parametri rilevanti nell'ottica di effettuare il pricing di un derivato sono dunque:

- r, il tasso d'interesse privo di rischio relativo all'intervallo di tempo $(0,T)$, dove T è la maturità del derivato;
- $v(0)$, il valore iniziale della varianza;

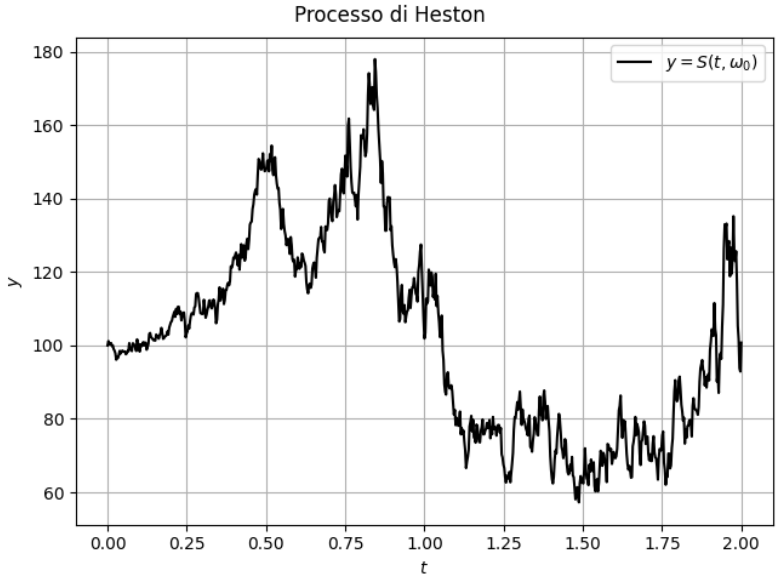

Figura 4.2 Una traiettoria del processo di Heston generata tramite le funzionalità implementate in SFMQuantLib. Nello specifico, la traiettoria è stata ottenuta ponendo $r = 3\%$, $\kappa = 0.5$, $\theta = 60\%$, $\xi = 40\%$, $\rho = 30\%$, $v(0) = 3\%$, $S(0) = 100$ e algoritmo di generazione PCG64 con seed pari a 85.

- κ, il tasso di ritorno alla media della volatilità al quadrato;
- θ, la media asintotica del processo varianza;
- ξ, il parametro di diffusione del processo varianza, comunemente detto *volatility of volatility* o *vol-of-vol*;
- ρ, il coefficiente di correlazione tra le componenti del moto browniano bi-dimensionale.

Come accennato in precedenza, essi si ricavano a seguito di un processo di calibrazione del modello, che approfondiremo nel Capitolo 5. Gli effetti della volatilità stocastica sul generale andamento del prezzo sono visibili in Fig. 4.2.

4.4.1 Pricing per le opzioni call vanilla

Per far sì che il processo di calibrazione sia il più possibile rapido e preciso, è fondamentale esplicitare un qualche tipo di relazione analitica tra il prezzo di una tipologia di derivato e i vari parametri del modello. Tuttavia, nel caso del modello di Heston non è possibile ricalcare quanto fatto per il modello Black-Scholes, dato che la distribuzione di $S(t)$ non è nota analiticamente.

Una soluzione al precedente problema risiede nell'impiego dei metodi di *pricing mediante FFT*, dove FFT è l'acronimo di *Fast Fourier Transform*, il celebre algoritmo numerico che

permette di approssimare in maniera estremamente efficiente gli integrali di Fourier

$$\mathscr{F}[f](u) = \int_{\mathbb{R}} e^{iux} f(x)\, dx,$$

dove i denota l'unità immaginaria. Il collegamento tra il modello di Heston e la teoria degli integrali di Fourier risiede nel concetto di *funzione caratteristica* di una variabile aleatoria X che, per definizione, è la trasformata di Fourier della pdf, vale a dire la funzione φ_X tale che

$$\varphi_X(u) = \mathscr{F}[f_X](u) = \int_{\mathbb{R}} e^{iux} f(x)\, dx = \mathbb{E}\left[e^{iuX}\right].$$

Conoscendo la funzione caratteristica è possibile ricostruire la pdf mediante la *trasformata di Fourier inversa* definita come

$$\mathscr{F}^{-1}[f](x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-iux} f(u)\, du,$$

in modo che

$$\mathscr{F}^{-1}[\mathscr{F}[f]] = \mathscr{F}\left[\mathscr{F}^{-1}[f]\right] = f.$$

Vale inoltre il seguente risultato noto anche come *teorema d'inversione*:

$$\begin{aligned} F_X(x) = \mathbb{P}(X \leq x) &= \int_{-\infty}^{x} f(t)\, dt \\ &= \frac{1}{2} - \frac{1}{2\pi} \int_{\mathbb{R}} \frac{e^{-iux} \varphi_X(u)}{iu}\, du. \end{aligned}$$

Sebbene la distribuzione del processo stocastico al tempo t non sia nota per il modello di Heston, nell'articolo originale del 1993 viene caratterizzata la sua funzione caratteristica, di cui non riportiamo la formulazione analitica che risulta essere molto complessa. Basti sapere che è possibile valutare in modo analitico $\varphi_{S(T)}$ e ciò permette di ricavare informazioni sulla cdf e la pdf della distribuzione di $S(T)$. Ad esempio, ricordando che il fair price di un'opzione call vanilla può essere ottenuto mediante la formula

$$C(t) = S(t)\tilde{\mathbb{Q}}(S(T) > K) - Ke^{-r(T-t)}\mathbb{Q}(S(T) > K),$$

dove $\mathbb{Q}$ è la misura neutrale al rischio e $\tilde{\mathbb{Q}}$ è la misura la cui derivata di Radon-Nikodym rispetto a $\mathbb{Q}$ è descritta nell'Eq. (4.10), un possibile approccio consiste nell'applicare il teorema d'inversione direttamente alle probabilità $\tilde{\mathbb{Q}}(S(T) > K)$ e $\mathbb{Q}(S(T) > K)$ così da ottenere il prezzo dell'opzione in termini di integrali della funzione caratteristica. Tale approccio è noto come *formula di Gil-Pelaez* ed è descritto nell'articolo Gil-Pelaez (1951).

Per ottenere una stima utilizzando il metodo appena descritto è necessario ricorrere a delle formule di quadratura su un intervallo illimitato ed esse ottengono risultati sufficientemente precisi solo usando un numero elevato di nodi di quadratura, il che rende questa tecnica onerosa dal punto di vista computazionale. Nella soluzione numerica di un problema di ottimizzazione, la funzione obiettivo viene valutata anche migliaia di volte prima che il metodo raggiunga una soluzione definibile come ottima, quindi è fondamentale che il metodo di valutazione del prezzo di una call vanilla sia il più efficiente possibile. Ciò giustifica il nostro interesse nella cosiddetta formula di Lewis (vedi Lewis (2002)). Essa sancisce che il prezzo di una call vanilla sia pari alla quantità

$$C(t) = S(t) - \frac{\sqrt{S(t)K}e^{-r(T-t)}}{\pi} Re\left[\int_0^\infty e^{iuk}\varphi_{S(T)}\left(u-\frac{i}{2}\right)\frac{1}{u^2+\frac{1}{4}}du\right], \tag{4.13}$$

dove $k = \log\left(\frac{S(t)}{K}\right)$ viene anche detta *log-moneyness* dell'opzione. L'Eq. (4.13) può essere riscritta in maniera tale da far sì che $C(t)$ sia ottenibile numericamente mediante l'impiego dell'algoritmo FFT, come mostrato nell'Appendice B, permettendo di calcolare in maniera precisa ed efficiente il prezzo di opzioni call vanilla nel modello di Heston.

4.4.2 Proprietà del modello

Il modello di Heston risolve la maggior parte dei limiti del modello precedentemente descritto, a partire dall'indebolire le ipotesi relative alla distribuzione del ritorno logaritmico dell'investimento sull'intervallo $(0,T)$ definito come

$$G = \log\left(\frac{S(T)}{S(0)}\right).$$

L'assunzione di log-normalità di $S(T)$ equivale al considerare

$$G \sim \mathcal{N}(\mu t, \sigma^2 t),$$

tuttavia l'osservazione empirica dei dati di mercato mostra che la distribuzione dei ritorni logaritmici risulta spesso essere fortemente asimmetrica e quindi non ben descrivibile da una distribuzione di tipo normale. Il modello di Heston non è caratterizzato da questo limite, in quanto facendo variare il parametro di correlazione ρ è possibile ottenere distribuzioni terminali caratterizzate da *skewness* (momento di ordine tre che misura l'asimmetricità della distribuzione) fortemente diversa da zero, come mostrato in Fig. 4.3.

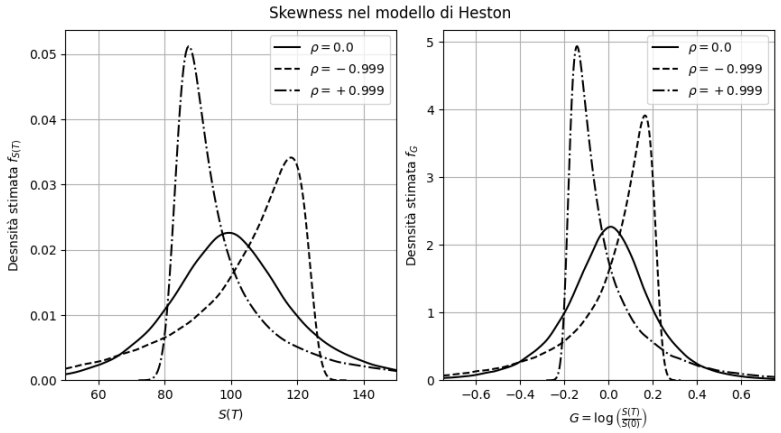

Figura 4.3 Il modello di Heston permette di descrivere distribuzioni terminali asimmetriche. In figura è possibile osservare le stime del kernel di densità dello stato terminale (sinistra) e per il ritorno logaritmico (destra) ottenute generando $2^{18} \simeq 260,000$ traiettorie del processo di Heston implementato in SFMQuantLib e l'oggetto *kdeplot* della libreria *seaborn*.

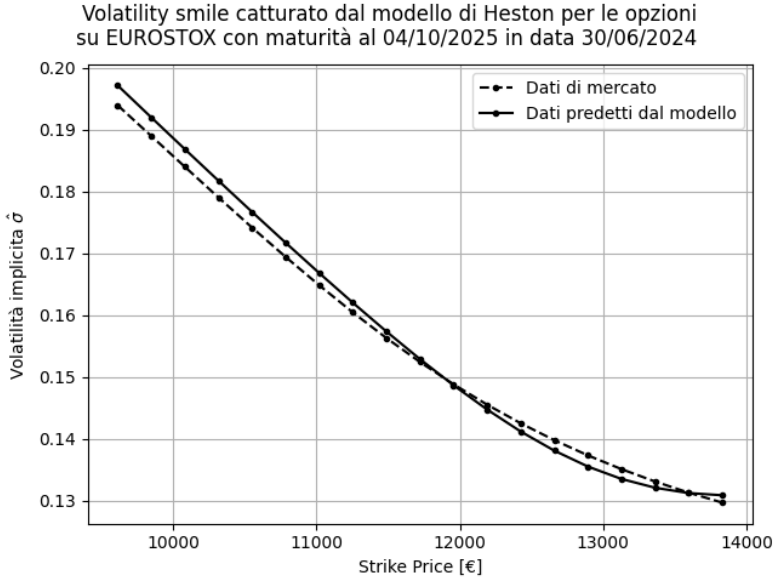

Figura 4.4 Tra i vantaggi del modello di Heston rispetto al modello di Black-Scholes si ha la sua capacità di catturare il volatility smile quando correttamente calibrato. Questa immagine mostra i risultati della calibrazione del modello di Heston implementato in SFMQuantLib sui dati di mercato relativi alle opzioni sull'indice EUROSTOX in data 30 Giugno 2024.

Un altro vantaggio del modello di Heston sta nella sua capacità di catturare il cosiddetto *volatility smile* delle opzioni quando calibrato correttamente, come mostrato in Fig. 4.4.

Il maggior numero di parametri permette al modello di rappresentare in maniera molto più realistica la dinamica del prezzo di un titolo azionario e, in generale, di descrivere in maniera più fedele svariate dinamiche tipiche dei mercati del mondo reale. Chiaramente ciò ha lo svantaggio di rendere più onerosa la calibrazione del modello.

Tra le principali debolezze troviamo sicuramente l'assenza di una formula analitica che permetta di prezzare derivati semplici. Inoltre, come approfondiremo nel capitolo Capitolo 5, la simulazione del processo stocastico descritto nell'Eq. (4.12) è afflitta da caratteristiche di instabilità numerica che rischiano di rovinare significativamente la precisione della stima Monte Carlo.

Capitolo 5

SFMQuantLib

In questo capitolo andremo a presentare in maniera più dettagliata la libreria quantitativa *SFMQuantLib*. Essa nasce con l'obiettivo di raccogliere in un unico progetto tutte le funzionalità quantitative necessarie allo svolgimento delle attività di trading e market-making di Sella Financial Markets. Tra le principali funzionalità della libreria troviamo la descrizione dei principali strumenti finanziari (bond, azioni, opzioni, etc.), la possibilità di calcolare il fair value di questi, strumenti per effettuare analisi di sensitività e quelli necessari al fine di elaborare adeguatamente i dati di mercato (calibrazione, bootstrap, stima di parametri etc.).

La libreria è caratterizzata da un'architettura abbastanza complessa che ha lo scopo di garantire la ri-utilizzabilità del codice implementato e agevolare le procedure di aggiornamento e mantenimento delle funzionalità presenti. Tale architettura fa uso estensivo dei principi della *programmazione orientata agli oggetti* (Object Oriented Programming) che approfondiremo qui di seguito. Come linguaggio di programmazione per lo sviluppo della libreria si è scelto il Python: un linguaggio orientato agli oggetti, estremamente flessibile, dalla sintassi accessibile e dotato di potenti librerie che lo rendono tra gli standard di riferimento nell'ambito del calcolo scientifico, del machine learning e dell'analisi dati.

In questo capitolo, dopo un breve sunto delle principali nozioni legate alla programmazione orientata agli oggetti in Sezione 5.1, verrà fornita una descrizione di massima dell'architettura e delle principali interfacce della libreria (Sezione 5.2) per poi approfondire le interfacce sviluppate ai fini di aggiungere alla libreria le funzionalità per calcolare il fair price di derivati esotici mediante metodi Monte Carlo in Sezione 5.3.

5.1 Programmazione Orientata agli Oggetti

La programmazione a oggetti (OOP) è un paradigma di programmazione che organizza il software in unità discrete chiamate *oggetti*. Questo approccio mira a modellare il software sul mondo reale, dove oggetti con proprietà e comportamenti distinti interagiscono tra loro.

L'elemento strutturale fondamentale è la *classe* che va a definire gli *attributi*, ovvero le proprietà, e i *metodi*, vale a dire le funzionalità, che tutti gli oggetti appartenenti ad essa devono condividere e funziona quindi come una specie di "stampo" per l'oggetto stesso. Tra i metodi definiti dalla classe deve sempre esserci un costruttore, utilizzato per inizializzare gli attributi quando viene costruito un oggetto e, a seconda dei contesti, può essere presente anche un distruttore, per liberare risorse o eseguire azioni prima della distruzione dell'oggetto.

Uno dei concetti più importanti legati ai paradigmi orientati agli oggetti è quello di *astrazione*. L'astrazione è il processo attraverso il quale si vanno a nascondere i dettagli complessi di un sistema e si mostrano solo le funzionalità essenziali. Un'*interfaccia* è un modello che non può essere istanziato direttamente, ma fornisce uno schema che va ad accomunare una famiglia di classi derivate. Spesso risulta utile progettare l'architettura del codice in maniera tale da separare l'interfaccia di una classe dalla sua implementazione, così da semplificare l'interazione con oggetti complessi e favorire un design modulare. Così facendo si ha il vantaggio di scindere completamente gli aspetti legati alla definizione della struttura di una classe dalla sua logica di funzionamento interna e si rende possibile trattare oggetti anche molto diversi in maniera analoga. Nell'ambito del paradigma OOP, tale pratica viene indicata con il termine *polimorfismo*.

L'astrazione è agevole nel contesto di un programma a oggetti grazie al concetto di *ereditarietà*, ovvero il processo di creazione di nuove classi derivate (figlie) a partire da una classe esistente (genitore). La classe figlia, oltre ad ereditare automaticamente tutti gli attributi e i metodi della classe genitore, potrà estenderne o sovrascriverne il comportamento favorendo il riuso del codice e facilitando la gestione gerarchica delle classi.

Un'altra idea fondamentale nella definizione del paradigma OOP è quella di *incapsulamento*, vale a dire la gestione mirata e ottimizzata dell'accesso e della manipolazione dei dati contenuti in un determinato oggetto. È infatti possibile definire diversi livelli di protezione dei metodi e degli attributi di una classe che aiutano a garantire la protezione dei dati interni dell'oggetto e la prevenzione di modifiche non autorizzate. Un metodo o attributo è detto:

- *pubblico* se è accessibile sia dall'interno che dall'esterno della classe;
- *protetto* se è accessibile all'interno della classe stessa e all'interno delle classi derivate

ma non dall'esterno della classe;

- *privato* se è a esclusiva disposizione della classe stessa.

La scelta di basare SFMQuantLib sul paradigma OOP porta numerosi vantaggi. Da un lato, il concetto di avere diversi oggetti che interagiscono tra di loro per produrre un risultato ricalca alla perfezione il contesto reale, nel quale il funzionamento di un singolo strumento è dato dalla sua relazione con altri oggetti o quantità ed è naturale scomporre i problemi di pricing, calibrazione e analisi di sensitività in sub-unità più semplici specializzate su ciascuno di questi aspetti. D'altro canto, suddividere il codice in famiglie di oggetti caratterizzate da una gerarchia ben definita ne agevola immensamente il mantenimento e promuove un alto livello di ri-utilizzabilità del codice.

5.1.1 Design pattern

I *design pattern* sono soluzioni comprovate a problemi comuni di progettazione del software. Essi rappresentano le migliori pratiche sviluppate nel tempo da esperti del settore e forniscono un modo standardizzato per affrontare problemi ricorrenti nella programmazione orientata agli oggetti. È importante rimarcare la differenza tra i pattern e gli algoritmi: mentre quest'ultimi definiscono una precisa serie di azioni che permettono di raggiungere un obiettivo, un pattern è una descrizione più ad alto livello di una soluzione e il codice dello stesso pattern applicato a due programmi diversi è quasi sempre differente.

Seguendo il testo di Gamma et al. (1994), effettuiamo la distinzione tra

- *pattern creazionali* che forniscono meccanismi per la creazione di oggetti che aumentano flessibilità e riutilizzo del codice esistente;
- *pattern strutturali* che riguardano la composizione delle classi o degli oggetti utilizzando l'ereditarietà per comporre interfacce e ottenere nuove funzionalità;
- *pattern comportamentali* che si occupano della comunicazione efficace e dell'assegnazione di responsabilità tra gli oggetti.

Per l'implementazione delle funzionalità di SFMQuantLib si è fatto uso di diversi design pattern, fra cui:

- la *Factory* che fornisce un'interfaccia in grado di costruire oggetti di tipo diverso in un'unica super-classe; questo design pattern risulta assai utile quando non si conoscono

in anticipo i tipi esatti e le dipendenze degli oggetti con cui il codice dovrà interagire dato che consente di separare il codice di costruzione di un oggetto finale da quello che lo va a utilizzare e conferisce al programma una struttura più pulita e soprattutto estendibile;

- il *Singleton* che è la soluzione più adatta nelle situazioni in cui è necessario assicurarsi che una classe abbia una sola istanza e sia dotata di un unico punto di accesso globale; spesso questo pattern viene usato in combinazione con il precedente in modo da poter richiamare sempre la stessa Factory e non doverne costruire una nuova nel caso servisse più di una volta;

- il *Decorator* che è un design pattern strutturale che permette di aggiungere nuovi comportamenti e funzionalità a determinati oggetti già esistenti costruendo una nuova classe che "avvolga" l'oggetto originale, che verrà passato direttamente nel costruttore del decoratore, e vada a richiamare i metodi di quest'ultimo inserendo ove necessario nuove funzionalità; questo design pattern permette di aumentare la flessibilità e la dinamicità del codice, sfruttando a pieno i concetti dell'ereditarietà e del polimorfismo tipici della programmazione ad oggetti.

- il *Composite*, ovvero il design pattern strutturale che permette di trattare un gruppo di oggetti, organizzati secondo una struttura ad albero nella quale le foglie sono gli oggetti semplici, come se fossero l'istanza di un oggetto singolo; il concetto chiave dietro al Composite è che dev'essere possibile manipolare un'istanza singola dell'oggetto nello stesso modo in cui si manipolerebbe un gruppo di essi.

5.2 Descrizione della libreria

In questa sezione verrà effettuata una presentazione dell'architettura e delle principali interfacce che vanno a costituire SFMQuantLib. Come già anticipato in precedenza, la libreria è stata ideata cercando di costruire un'architettura caratterizzata da legami logici e flussi di informazioni il più possibile simili alla realtà, in modo che i procedimenti da seguire siano chiari e che le classi implementate ricalchino in maniera fedele le proprietà degli oggetti del mondo reale.

Cominciamo la presentazione a partire da una descrizione ad alto livello dell'architettura. La libreria nasce con l'obiettivo di descrivere gli strumenti finanziari (che siano azioni, bond, opzioni, swap, ecc.) con tutte le annesse caratteristiche e di poter svolgere operazioni su di essi, come il calcolo del fair price o l'analisi di sensitività rispetto a certi fattori di rischio. A

ogni strumento corrispondono in pratica uno o più flussi di cassa che devono essere prezzati. Chiaramente il procedimento per il calcolo del valore attuale netto di un flusso di cassa varia a seconda dello strumento e delle sue proprietà, quindi tale compito viene delegato a una classe capace di prezzare ogni tipo di flusso di cassa, ma che riesca a farlo nel modo corretto in base allo strumento che si sta considerando. Il calcolo del fair price dipende da una notevole quantità di parametri come tassi d'interesse o volatilità che vengono raggruppati in una collezione ordinata di dati calibrati e pronti per essere usati nelle varie procedure. I dati calibrati si ottengono a seguito di vari processi di raffinamento o calibrazione dei dati osservabili sul mercato che vengono importati al momento della creazione di determinati strumenti.

Quindi, per riassumere, il workflow per il calcolo del fair price di uno strumento all'interno della libreria è dato dal seguente schema:

1. si descrivono gli strumenti che si desidera studiare, istanziando uno o più oggetti della classe `Instrument`, e si importano i dati di mercato relativi ad essi mediante adeguati oggetti di tipo `MarketDataObject`;
2. mediante un oggetto di tipo `Calibrator` si esegue la calibrazione dei dati di mercato usando la procedura più adeguata, ottenendo così uno o più oggetti di tipo `Calibrated-DataObject` che vengono inseriti in un apposito container facente parte della classe `CalibratedDataRepository`;
3. si istanzia un oggetto di tipo `Pricer` dandogli accesso ai dati calibrati;
4. si assegna il `Pricer` appena creato agli strumenti che si desidera studiare specificando per ciascuno di essi il metodo di pricing che si vuole utilizzare (mediante un attributo della enumerazione `Model`);
5. si chiede a ogni strumento di calcolare il proprio valore attuale netto, o *net present value* (NPV); per fare ciò esso passerà tutti i propri flussi di cassa, oggetti di tipo `Cashflow`, al `Pricer` assegnatogli che ne va a calcolare il prezzo, per poi restituire all'utente il totale e le informazioni relative a ciascun flusso di cassa.

Per un'analisi di sensitività il procedimento è simile, ma viene ripetuto più volte facendo variare i valori dei fattori di rischio che si desidera studiare mediante oggetti di tipo `Bumper` e poi calcolando diverse metriche di sensitività (classe `Metric`).

Nella restante parte di questa sezione effettueremo un breve focus su alcune interfacce della libreria ricalcando quanto fatto da Marta Bosio nella sua tesi magistrale intitolata *Sviluppo di una Libreria di Pricing e di Sensitivity Analysis per Opzioni Vanilla Europee* (vedi Bosio (2023)).

5.2.1 Instrument

È l'interfaccia rappresentante tutte le tipologie di strumenti finanziari e pertanto contiene le caratteristiche comuni a questi. In particolare, ogni strumento è identificato da:

- alcune informazioni anagrafiche, tra cui il nome specifico (l'ISIN, International Securities Identification Number, ovvero un codice internazionale che identifica univocamente gli strumenti finanziari), la valuta con cui viene scambiato, la tipologia di calendario con cui vengono calcolate le date dei pagamenti e le convenzioni per il calcolo delle stesse;
- la tipologia, ad esempio bond, forward, future, opzione, ecc.;
- la classe di asset di appartenenza (ad esempio obbligazioni, azioni, valute estere, commodities, ecc.).

Per quanto riguarda i metodi comuni a tutti gli strumenti, essi sono principalmente funzioni che si occupano di ricavare informazioni dagli stessi. Ad esempio, si possono voler estrarre tutte le dipendenze di un oggetto. Nel caso di un'opzione, sicuramente ci sarà una dipendenza dal suo sottostante, una dalla struttura per scadenze dei tassi e una dalla volatilità o da determinati parametri. Tuttavia, la principale informazione che probabilmente si vuole ricavare da uno strumento è il suo fair price. Come detto in precedenza questo si otterrà a partire dai flussi di cassa, rappresentati con oggetti appartenenti all'interfaccia `Cashflow`. Per garantire e preservare un'ampia possibilità di scelta nelle procedure di pricing, prima di poter calcolare il prezzo di uno strumento è necessario specificare un particolare `Pricer` e un particolare `Model`. Questo si può fare agilmente chiamando i due metodi `set_pricer` e `set_model`, sempre implementati per un oggetto della classe `Instrument`.

5.2.2 Cashflow

Ogni oggetto appartenente alla classe `Instrument`, dovrà essere in grado di restituire tutti i suoi flussi di cassa e i corrispettivi NPV. Questi non vengono calcolati direttamente all'interno dello strumento, ma sfruttano un'architettura più articolata che consente di incapsulare ogni singolo cashflow. Un flusso di cassa sarà sempre caratterizzato dal modello utilizzato per prezzarlo, dalla data di pagamento, dal nozionale del relativo strumento e dalla sua valuta. Le funzioni che ogni `Cashflow` deve implementare sono:

- il metodo che imposta il `Pricer` (`set_pricer`) e quello che imposta il Model al flusso di cassa (`set_model`);

- il metodo che restituisce tutte le dipendenze dell'oggetto in analisi, utile nella fase di pricing per riuscire ad estrarre dall'insieme dei dati di mercato calibrati quelli che servono per lo strumento considerato;
- il metodo `get_expected_amount` che calcola il valore atteso del flusso di cassa futuro;
- il metodo `get_npv` che calcola il valore attuale netto del flusso di cassa, data una curva di tassi d'interesse;
- il metodo che verifica che il flusso non sia già scaduto, ovvero che la data di pagamento sia successiva a quella di valutazione dello stesso;
- il metodo che esprime se il valore atteso futuro del flusso di cassa è certo oppure no.

5.2.3 MarketDataObject

Ogni dato ricavato direttamente dal mercato viene incapsulato in un adeguato oggetto appartenente a una sottoclasse di `MarketDataObject`. Ogni oggetto di questa classe è dotato di un identificativo composto da una terna che specifica:

- la tipologia di dato di mercato (tasso d'interesse, volatilità, prezzo di mercato, ecc.);
- la valuta in cui è espresso il dato;
- la dipendenza del dato di mercato (ad esempio, se il dato è il prezzo di mercato di un'azione, questo campo sarà il nome dell'azione stessa).

Oltre all'identificativo, un oggetto di questa classe è caratterizzato dalla data di riferimento nella quale il dato è stato effettivamente "scaricato" dal mercato e dalla tipologia di calibrazione che si vuole effettuare su di esso. Esempi di sottoclassi che ereditano dall'interfaccia `MarketDataObject` sono:

- `MarketValueMarketData`, utilizzato per salvare le quotazioni di mercato principalmente di azioni o indici ed eventuali dividendi;
- `RateTermStructureMarketData`, utilizzato per raccogliere i dati relativi ai tassi d'interesse;
- `VolatilitySurfaceMarketData`, utilizzato per collezionare al suo interno i prezzi di mercato delle opzioni considerate ai fini di calibrare una superficie di volatilità.

5.2.4 CalibratedDataObject e CalibratedDataRepository

Così come tutti i dati di mercato vengono incapsulati in opportuni oggetti di tipo `MarketDataObject`, tutti i dati frutto della calibrazione di questi ultimi vengono incapsulati in oggetti appartenenti a opportune sottoclassi dell'interfaccia `CalibratedDataObject`. Un oggetto calibrato ha attributi simili al corrispondente oggetto che descrive i dati di mercato di partenza, ma ha metodi che permettono di sfruttarlo per ricavare i dati necessari alle operazioni da svolgere successivamente. Esempi di sottoclassi che ereditano da `CalibratedDataObject` sono:

- `ZeroRateTermStructure` che è una struttura contenente per ogni data un tasso d'interesse ed è utilizzabile per ricavare il tasso di sconto o lo zero rate per una certa data e il tasso forward data una coppia di date;
- `CalibratedVolatilitySurface` ovvero l'oggetto che descrive una superficie di volatilità ottenuta a seguito di un processo di calibrazione su un oggetto di tipo `VolatilitySurfaceMarketData`;
- `CalibratedParameters` ovvero l'oggetto usato per incapsulare determinati parametri utili al calcolo dell'NPV secondo un qualche modello.

La classe `CalibratedDataRepository` ha la funzione di definire un container per raccogliere tutti i dati di mercato già calibrati e pronti per essere usati nelle procedure di pricing o di calcolo delle metriche desiderate. Infatti, a partire dalle dipendenze di ogni oggetto, che saranno anche quelle dei suoi cashflow, ogni oggetto di tipo `CashflowPricer` riuscirà a capire quali sono i dati che gli servono per calcolare il fair value e estrarrà da un oggetto di tipo `CalibratedDataRepository` i dati calibrati corrispondenti. Quest'ultima ha infatti metodi per estrarre, dato un particolare identificativo, il dato calibrato ad esso associato.

5.2.5 Calibrator

Gli oggetti appartenenti alle classi figlie dell'interfaccia `Calibrator` sono i responsabili del processo di calibrazione. Infatti, sfruttando il principio del polimorfismo, ogni tipo di calibratore implementerà in modo diverso il metodo `calibrate`, che restituirà una lista di oggetti di tipo `CalibratedDataObject`. Gli oggetti di tipo `Calibrator` non vengono costruiti dall'utente e la loro inizializzazione avviene in maniera automatica per mezzo della `CalibratorFactory` . Questa è, come dice il nome, un esempio di implementazione del design pattern della

Factory. Il funzionamento è il seguente: ogni `MarketDataObject` avrà un `CalibratorModel` di riferimento, che può essere o fisso o indicato dall'utilizzatore. A seconda della tipologia di dato e del modello di calibrazione scelto, all'interno della Factory verrà costruito uno specifico tipo di calibratore. `CalibratorFactory` è anche un Singleton: in questo modo si raggiunge la certezza che un solo calibratore venga creato e non si rischia di fare eventuali errori o sovrascrivere dati già calibrati. Tra i calibratori implementati in SFMQuantLib troviamo ad esempio:

- `YieldCurveBootstrapper` che produce un oggetto di tipo `ZeroRateTerm Structure` a partire dai dati di mercato relativi ai prezzi di svariate obbligazioni emesse dalla stessa entità contenuti in un oggetto della classe `RateTermStructureMarketData`;
- `ImpliedVolatilityCalibrator` che partendo da un `VolatilitySurfaceMarket-Data`, si occupa di restituire una `CalibratedVolatilitySurface`, calcolando la volatilità implicita dell'elenco di opzioni prese dal mercato e di cui sono noti i prezzi.

5.2.6 Pricer

Infine si ha l'interfaccia `Pricer` che definisce la struttura degli oggetti contenenti la logica necessaria a prezzare i flussi di cassa di uno strumento. Per rendere flessibile e pulita l'architettura di questa interfaccia, che contiene di fatto il cuore del processo di pricing, si è ricorsi all'utilizzo dei design pattern. Dall'interfaccia `Pricer` eredita direttamente la classe `InstrumentPricer`, che utilizza la struttura di un Composite. Tale classe contiene infatti tre oggetti appartenenti alle seguenti tre classi:

1. `InterestRatePricer` ovvero il pricer contenente la logica necessaria a prezzare flussi di cassa il cui valore dipende esclusivamente dal valore dei tassi d'interesse;
2. `OptionPricer` ovvero la classe specializzata nel pricing di derivati non lineari come opzioni;
3. `ForwardPricer` contenente il codice per prezzare contratti forward.

È importante sottolineare che questi tre oggetti non ereditano dall'interfaccia `Pricer`, in quanto non implementano tutte le funzioni per il pricing di qualsiasi strumento, ma solo quelle specifiche. Essi hanno il compito, a loro volta, di creare il `CashflowPricer` adeguato, in base al modello scelto per prezzare lo strumento. Idealmente, la classe `InstrumentPricer` costituisce la radice di un albero, e deve mettere a disposizione tutti i potenziali metodi di

calcolo del prezzo per tutti gli strumenti e per tutti i modelli disponibili. Alla radice sono collegati tre nodi che rappresentano i tre oggetti che si trovano dentro all'`InstrumentPricer` e a ognuno di questi sono collegate un certo numero di foglie che rappresentano i diversi modelli di pricing per quel tipo di strumento e sono in corrispondenza uno a uno con oggetti appartenenti a sotto classi dell'interfaccia `CashflowPricer`.

5.3 Funzionalità implementate in SFMQuantLib

L'obiettivo centrale di questo lavoro di tesi è stato quello di implementare in SFMQuantLib le funzionalità necessarie per effettuare il pricing di derivati esotici usando la combinazione di valutazione neutrale al rischio e metodi Monte Carlo descritta nel Capitolo 4. Chiaramente per fare ciò è stato necessario implementare delle nuove funzionalità all'interno della libreria che andassero a integrarsi nel workflow descritto in precedenza senza alterarne funzionamento e performance. Al tempo stesso era richiesto che le nuove funzionalità fossero esse stesse ben strutturate e caratterizzate da un'architettura logica, ben definita e che ne favorisse la ri-utilizzabilità, così da renderle uno strumento versatile a disposizione degli utilizzatori della libreria. Ciò ha richiesto una attenta fase di studio architetturale che ha permesso di garantire che l'intervento su SFMQuantLib fosse caratterizzato da ottime caratteristiche di incapsulamento, modularità e astrazione.

Innanzitutto ci si è focalizzati sull'implementazione all'interno della libreria delle funzionalità necessarie alla generazione di numeri casuali in maniera efficiente, precisa e riproducibile. Ciò ha portato alla definizione dell'interfaccia `RandomGenerator` che descrive le funzionalità di tutte le tecniche di generazione di numeri casuali implementate. Accanto a essa è stata sviluppata l'interfaccia `ProbabilityDistribution` che invece ha lo scopo di descrivere e caratterizzare distribuzioni di probabilità multivariate in termini di densità, funzione di ripartizione, distribuzioni marginali e copula. Insieme queste due famiglie di oggetti costituiscono il nuovo cuore simulativo della libreria e fungono da nucleo centrale per l'effettuazione di simulazioni Monte Carlo.

Successivamente l'attenzione è stata rivolta alla strutturazione dell'interfaccia `StochasticProcess` in maniera tale da garantire la creazione di classi che permettessero di implementare e manipolare processi stocastici in maniera agevole e versatile. L'obiettivo degli oggetti della famiglia `StochasticProcess` è quello di trasformare il flusso di valori generato da un `RandomGenerator` in traiettorie caratterizzate da specifiche distribuzioni di probabilità. Chiaramente non è possibile rappresentare con precisione assoluta le traiettorie di processi a

tempo continuo, quindi si è fatto ricorso a diversi schemi di discretizzazione che garantiscano un errore di approssimazione basso e non impattante sulle operazioni di stima.

I metodi simulativi e i processi stocastici d'interesse sono i principali ingredienti che permettono di effettuare una simulazione Monte Carlo. Al suo cuore tale simulazione non è nient'altro che un ciclo di lunghezza pari al numero di scenari considerati, che ad ogni iterazione genera una traiettoria del processo stocastico per poi salvarla in un'apposita struttura dati che ne faciliti la manipolazione e l'estrazione. Gli oggetti delle sotto classi di `PathManager` si occupano di gestire la simulazione e incapsularne i risultati. La gestione della simulazione comprende sia la fase di effettiva generazione delle traiettorie sia la fase di lettura dei dati necessari. La comunicazione con i vari `PathManager` avviene per mezzo di oggetti appartenenti all'interfaccia `PathManagerRequest` che definiscono una sorta di "protocollo di comunicazione" efficiente e sicuro.

Successivamente si è implementato un metodo di calibrazione per il modello di Heston utilizzando l'algoritmo FFT applicato alla formula di Lewis dell'Eq. (4.13). La logica è stata implementata all'interno di un'opportuna sotto classe dell'interfaccia `Calibrator` chiamata `LewisFFTHestonParametersCalibrator`. Il risultato di questo processo di calibrazione è una stima dei parametri del modello di Heston che meglio descrivono la superficie di volatilità osservabile sul mercato. Tali stime sono incapsulate in un oggetto appartenente alla classe `MonteCarloParameters` che a sua volta eredita dalla sotto classe `CalibratedParameters` di `CalibratedDataObject`.

Infine si è passati all'implementazione di nuovi strumenti finanziari all'interno della libreria, così da poter sfruttare appieno le nuove funzionalità implementate. Come abbiamo ribadito più volte nel corso della trattazione, i metodi Monte Carlo permettono di prezzare qualsiasi derivato purché si sia in grado di calcolarne il payoff data una realizzazione del processo del sottostante. La gamma di strumenti che possono essere prezzati risulta essere dunque vastissima, ma si è scelto di concentrarsi su opzioni barriera per via del loro payoff path-dependent che ne rende difficile il pricing in modo analitico e la loro utilità nella strutturazione dei certificati d'investimento. Sono state quindi definite le classi `EuropeanBarrierOption` e `Certificate` entrambe figlie di `Instrument` e i rispettivi cashflow.

Nella restante parte di questo capitolo approfondiremo questi aspetti in maggior dettaglio focalizzandoci sulle scelte implementative e architetturali effettuate per la definizione di ciascun oggetto.

5.3.1 RandomGenerator e ProbabilityDistribution

L'interfaccia `RandomGenerator` definisce la struttura di tutti i generatori di numeri casuali implementati nella libreria. Ogni generatore è caratterizzato dai seguenti metodi:

- `dimensionality`, una property che restituisce la dimensionalità del generatore, che come abbiamo visto nel Capitolo 2 è un aspetto fondamentale quando si opera con le sequenze a bassa discrepanza;
- `set_seed` che permette di specificare un seed per il generatore;
- `set_dimensionality`, un setter per modificare la dimensionalità del generatore;
- `reset`, ovvero il metodo che riporta lo stato del generatore a quello specificato dal seed di partenza;
- `get_samples` che permette di estrarre uno o più campioni casuali dal generatore;
- `skip`, un metodo che permette di aggiornare lo stato del generatore facendolo avanzare di un certo numero di estrazioni.

La gerarchia delle sottoclassi di `RandomGenerator` risulta essere abbastanza complessa: da essa eredita direttamente la classe `RandomGeneratorMeta` che implementa le funzionalità di base comuni a tutti i generatori, da cui a loro volta ereditano `PseudoRandomGenerator` e `QuasiRandomGenerator` che implementano le funzionalità di base dei generatori di numeri pseudo-casuali e quasi-casuali rispettivamente. Ciascuna di esse è un esempio del design pattern Decorator. Infatti, `PseudoRandomGenerator` avvolge un oggetto di tipo `Generator` del modulo `random` della libreria `numpy`, mentre `QuasiRandomGenerator` contiene al proprio interno un oggetto di tipo `QMCEngine` facente parte del modulo `stats.qmc` del pacchetto `scipy` . Così facendo si ottengono molteplici vantaggi: da un lato si è evitato di dover implementare da zero tutti gli algoritmi di generazione, che sarebbe stato molto dispendioso in termini di tempo, ma ci si è appoggiati su implementazioni robuste, stabili, efficienti e già pronte; dall'altro non si è dovuto ricorrere a nessun tipo di compromesso per quanto riguarda l'architettura interna della libreria definendo un interfaccia che si confacesse alle esigenze specifiche di essa. Da ciascuna delle due classi precedentemente definite ereditano due classi diverse: una che implementa il campionamento di numeri provenienti da una distribuzione di tipo uniforme (eventualmente multidimensionale) e una capace di produrre valori distribuiti in modo normale. Questa scissione permette di sfruttare appieno gli algoritmi di generazione interni, compilati e scritti in linguaggio C, per garantire il massimo livello di performance anche nel caso di campionamenti

dalla distribuzione gaussiana. Infine è stata definita la classe `DistributionRandomGenerator` che permette di campionare valori provenienti da una o più distribuzioni di probabilità definite dall'utente per mezzo di oggetti della classe `ProbabilityDistribution` che approfondiremo a breve.

La gerarchia così definita ha permesso l'implementazione di un gran numero di generatori estremamente ottimizzati e accomunati da un'unica interfaccia. Per evitare che l'utente debba conoscere in maniera approfondita l'architettura di questo modulo per poterlo utilizzare, si è scelto di definire la classe `RandomGeneratorFactory` che è una Factory-Singleton capace di costruire il generatore più adeguato data la distribuzione di probabilità da cui si vuole campionare, il seed, e l'algoritmo di generazione interno. Quest'ultimo viene specificato per mezzo degli attributi della classe `GenerationAlgorithm`, una sottoclasse di `enum` che specifica gli algoritmi di generazione a disposizione.

Per poter descrivere varie distribuzioni di probabilità e poter eseguire operazioni mediante esse si è implementata la classe astratta `ProbabilityDistribution` che definisce i seguenti metodi:

- `dimensionality` che restituisce la dimensione del vettore aleatorio descritto;
- `type` che ritorna un attributo della `enum` `DistributionType` che specifica la tipologia della distribuzione e permette alla `RandomGeneratorFactory` di scegliere la classe di generatore più adeguata;
- `copula` che permette di ottenere la copula della distribuzione;
- `marginals` che restituisce una lista contenente le distribuzioni marginali delle componenti del vettore aleatorio;
- `apply_to_sample` che, dato un campione casuale standard `u` proveniente da una distribuzione di tipo uniforme sull'ipercubo unitario, restituisce un valore proveniente dalla distribuzione di probabilità descritta.

Dall'interfaccia ereditano le classi `UniformProbabilityDistribution` che, data una scelta del parametro d, descrive una distribuzione di tipo uniforme sull'ipercubo $(0,1)^d$, `Normal ProbabilityDistribution` che descrive una distribuzione normale multivariata dati il vettore delle medie e la matrice di varianza e covarianza e `CustomProbabilityDistribution` che permette all'utente di descrivere una qualsiasi distribuzione di probabilità specificando la copula e le marginali. La copula deve essere specificata mediante un oggetto facente parte di una delle sottoclassi dell'interfaccia `Copula` di cui vengono forniti maggiori dettagli in Appendice A. Similmente le marginali vengono descritte da oggetti di tipo

`MarginalProbabilityDistribution` che ne caratterizzano la cdf F, l'inversa della cdf F^{-1} e la pdf f.

5.3.2 StochasticProcess

Si è voluto scindere completamente l'aspetto della generazione dei numeri pseudo-casuali o quasi-casuali dalla caratterizzazione dei processi stocastici, ai fini di aumentare la modularità del codice e incapsulare al meglio le diverse logiche coinvolte. A grandi linee, il workflow che l'utente segue quando desidera lavorare con un processo stocastico all'interno di SFMQuantLib è definito nel seguente modo:

1. l'utente sceglie un tipo di processo stocastico con cui lavorare sulla base del problema specifico che si desidera affrontare e ne crea un'istanza usando il costruttore della classe corretta o la classe `StochasticProcessFactory`;
2. una volta costruito, il processo è capace di fornire all'utente le informazioni relative alle distribuzioni di probabilità da cui è necessario campionare per generare delle traiettorie;
3. l'utente passa tali informazioni al Singleton `RandomGeneratorFactory` ottenendo così un generatore efficiente per la specifica distribuzione di probabilità;
4. passando i campioni prodotti dal generatore come argomenti agli appositi metodi del processo stocastico si va a costruire una traiettoria di quest'ultimo.

In questo modo si ottiene un'architettura caratterizzata da alti livelli di incapsulamento (infatti è possibile effettuare qualsiasi modifica alla logica del processo senza impattare minimamente quella del generatore e viceversa) ma al tempo stesso di facile uso per l'utente finale al quale viene semplicemente richiesto di scegliere un processo stocastico tra quelli a disposizione.

Prima di approfondire in maniera più dettagliata l'interfaccia che descrive le classi che implementano i processi stocastici della libreria, ci focalizziamo su due interfacce che implementano funzionalità ausiliarie di estrema rilevanza. La prima di esse è la classe `InitialCondition` che, come suggerisce il nome, descrive le condizioni iniziali per la generazione di un processo stocastico o, in maniera più precisa, la σ-algebra della filtrazione corrispondente al tempo iniziale t_0, $\mathscr{F}_{t_0}$. Modellare le classi facendo sì che queste rispecchino i vari concetti astratti che permettono di descrivere i processi stocastici in maniera formalmente rigorosa, rende il codice di più facile comprensione e garantisce un funzionamento aderente a quello della controparte astratta. Gli oggetti appartenenti alle sottoclassi di `InitialCondition` fungono

dunque da container strutturato per le informazioni necessarie ad avviare la generazione delle traiettorie di un processo stocastico. L'interfaccia definisce alcuni metodi che danno accesso alle informazioni in maniera complessiva (restituendole sotto forma di dizionario) o singolarmente, specificando un identificativo per l'informazione cercata. L'adozione di quest'oggetto garantisce che la lettura di questi dati avvenga in maniera controllata e che questi non vengano alterati inavvertitamente.

Nel Capitolo 1 abbiamo approfondito il legame esistente tra una particolare famiglia di processi stocastici (i processi di Itô) e la teoria relativa agli integrali. Per via delle assunzioni presentate nel Capitolo 4 tali processi risultano lo strumento ideale per descrivere l'andamento dei sottostanti di un derivato, quindi vi è la necessità di implementare funzionalità relative all'integrazione numerica negli oggetti che rappresentano questi tipi di processo stocastico. Fare ciò direttamente all'interno della logica del processo stocastico violerebbe il principio dell'incapsulamento, dato che una singola classe si occuperebbe sia di maneggiare i valori casuali per trasformarli in traiettorie, sia di effettuare approssimazioni numeriche di integrali e, sebbene tali funzionalità siano completamente distinte, esse non potrebbero essere separate per essere riutilizzate singolarmente. Si è dunque scelto di definire l'interfaccia `StochasticProcessParameter` in maniera tale che rappresenti i parametri che caratterizzano un certo processo di Itô e vada a definire le operazioni che si possono effettuare su di essi. Nello specifico l'interfaccia definisce i metodi:

- `integral` che permette di calcolare un'approssimazione numerica dell'integrale del parametro su un intervallo $(a,b) \subset \mathbb{R}$;
- `squared_integral` che calcola un'approssimazione numerica dell'integrale del quadrato del parametro su un intervallo $(a,b) \subset \mathbb{R}$;
- `mean` che restituisce il valor medio del parametro su un certo intervallo;
- `mean_squared` che restituisce il valor medio del quadrato del parametro su un intervallo.

Dall'interfaccia ereditano tre classi che rappresentano parametri costanti, costanti a tratti e lineari a tratti rispettivamente. Per il calcolo degli integrali viene fatto uso della formula dei trapezi.

Infine si è definita la classe `StochasticProcess` che descrive le funzionalità di tutti i processi stocastici tramite i seguenti metodi:

- `start_time`, getter che restituisce il tempo t_0 per il quale si conosce il primo valore della traiettoria, ovvero quello prescritto dalla condizione iniziale;

- `initial_value` che ritorna il valore iniziale della traiettoria;
- `end_time` che restituisce l'ultimo valore del tempo per cui è stato calcolato il valore assunto dal processo stocastico;
- `final_value` che ritorna il valore del processo stocastico al tempo restituito dal metodo precedente;
- `dimension` che restituisce la dimensione del processo stocastico, ovvero il numero delle componenti dei suoi valori;
- `number_of_stochastic_increments`, il metodo che restituisce il numero di valori casuali necessari per calcolare un nuovo valore della traiettoria; per la maggior parte dei processi di Itô questo valore è uguale a 1, ma per processi più complessi questo può assumere valori maggiori (ad esempio per il processo del modello di Heston sono necessari due valori casuali per aggiornare la traiettoria, uno per modificare il processo della varianza e uno per quello del prezzo);
- `distributions`, ovvero il metodo che specifica le distribuzioni da cui è necessario campionare per generare traiettorie del processo stocastico;
- `get_times` che restituisce un array contenente i valori del tempo t per cui il valore del processo stocastico è stato calcolato;
- `get_current_trajectory` che permette di avere accesso ai valori che compongono la traiettoria calcolata;
- `reset` che elimina la traiettoria calcolata fino a quel punto e reimposta il processo alla sua condizione iniziale;
- `generate_trajectory`, ovvero il metodo che permette di generare in blocco una traiettoria dati i valori del tempo per cui si desidera conoscere il valore del processo stocastico e un adeguato array di valori casuali provenienti dalla corretta distribuzione di probabilità;
- `step_to` che, dato un adeguato array di valori casuali, aggiunge in coda alla traiettoria un nuovo valore corrispondente al tempo specificato;
- `move_forward_by` che aggiunge in coda alla traiettoria un nuovo valore relativo al tempo pari al tempo finale incrementato di una certa quantità.

Da questa interfaccia ereditano tutte le classi che descrivono i processi stocastici disponibili nella libreria. A partire dalla classe `BrownianMotion` che, data una scelta di un parametro di drift $\mu(t)$ e di un parametro di diffusione $\sigma(t)$ permette di simulare le traiettorie di un moto browniano con drift, mediante lo schema di discretizzazione di Eulero-Maryama che consiste nel sostituire il differenziale deterministico dt con Δt e l'integrale stocastico della forma differenziale di un processo di Itô con un valore proveniente da una normale con varianza pari a $\int_t^{t+\Delta t} \sigma^2(s)\,ds$. A livello di codice interno al processo stocastico, dato un valore estratto da una gaussiana standard `z`, il nuovo valore `new_val` del processo al tempo `t_new` viene dunque calcolato a partire dalla traiettoria del processo stesso e dei suoi parametri `self._drift` e `self._volatility` usando i comandi seguenti.

Listing 5.1 Schema di Eulero-Maryama per il moto browniano

```
drift_component = self._drift.integral(self.end_time, t_new)
vol_component = \
    np.sqrt(self._volatility.squared_integral(self.end_time, t_new))
new_val = self.final_value + drift_component + vol_component * z
```

È importante osservare gli effetti positivi dell'incapsulamento e del polimorfismo: quello che queste linee vanno a descrivere è esclusivamente il metodo di discretizzazione del processo stocastico, che è completamente indipendente dal metodo di integrazione utilizzato o dalla tipologia di funzione usata per descrivere i parametri del processo. Se un domani queste dovessero cambiare, il processo stocastico rimarrebbe perfettamente capace di funzionare purché le modifiche rispettino l'interfaccia precedentemente definita.

La classe `GeometricBrownianMotion` implementa le funzionalità necessarie per la simulazione di un moto browniano geometrico data una scelta dei parametri di drift e volatilità. A differenza del caso precedente, per generare la traiettoria non si ricorre alla discretizzazione della SDE di riferimento ma si sfrutta la log-normalità dei valori di tale processo stocastico.

Listing 5.2 Schema di discretizzazione per il moto browniano geometrico

```
drift_component = self._drift.integral(self.end_time, t_new)
var_component = self._volatility.squared_integral(self.end_time, t_new)
new_val = self.final_value *\
    np.exp(drift_component - 0.5*var_component + np.sqrt(var_component)*z)
```

Nel codice precedente `z` è un valore estratto da una distribuzione gaussiana standard e lo schema di discretizzazione consiste nel trasformare tale valore in un campione proveniente da una distribuzione log-normale avente corrette media e varianza.

La libreria permette anche di simulare il processo di Ornstein-Uhlenbeck tramite gli oggetti della classe `OrnsteinUhlenbeckProcess`. Similmente a quanto fatto nel caso del moto browniano si è scelto di adottare uno schema di discretizzazione di Eulero-Maryama che

permette di calcolare nuovi valori di una traiettoria a partire dai valori degli integrali dei vari parametri del processo come riportato nel codice sottostante.

Listing 5.3 Schema di discretizzazione per il processo di Ornstein-Uhlenbeck

```
kappa_int = self._reversion_coefficient.integral(self.end_time, t_new)
mu_val = self._asymptotic_mean.mean(self.end_time, t_new)
st_dev = np.sqrt(self._volatility.squared_integral(self.end_time, t_new))
new_val = (self.final_value + (mu_val-self.final_value)*kappa_int + st_dev*z)
```

Vista la sua grande rilevanza nei modelli relativi ai tassi d'interesse si è scelto d'implementare anche il processo di Cox-Ingersoll-Ross mediante la classe `CoxIngersollRossProcess`. Per la discretizzazione di questo processo si è ricorso allo schema implicito bilanciato descritto nel testo di Platen and Heath (2006) che, seppur richiedendo un maggior numero di operazioni rispetto alla discretizzazione di Eulero-Maryama, ha il vantaggio di garantire la generazione di traiettorie caratterizzate da valori esclusivamente positivi.

Infine, si è scelto di implementare il processo risultato del sistema di SDE dell'Eq. (4.12) all'interno della classe `HestonProcess` così da poter prezzare titoli utilizzando tale modello per le dinamiche del sottostante. Come schema di discretizzazione per questo processo si è ricorso a quello Quadratico Esponenziale descritto nel paper Andersen (2007), basato su tecniche di moment-matching per la simulazione del processo varianza e da correzioni di martingala per il processo del prezzo. Nei successivi capitoli investigheremo la qualità dei risultati ottenuti usando questo schema.

Infine è importante sottolineare che l'interfaccia definita per il processo stocastico permette di caratterizzare processi stocastici sia monodimensionali che multidimensionali in maniera agevole e non ambigua, permettendo nel caso di questi ultimi di specificare la correlazione tra gli incrementi delle varie componenti mediante l'impiego di un'adeguata copula da parte dell'utente. Al tempo stesso viene garantita la possibilità di generare processi mediante il design pattern del Composite che permettono la simulazione simultanea di quantità caratterizzate da dinamiche anche completamente diverse tra di loro.

5.3.3 PathManager

Dopo aver implementato una gamma di algoritmi di generazione di numeri pseudo-casuali e quasi-casuali e aver aggiunto le funzionalità che permettono di trasformare il loro output in realizzazioni delle varie traiettorie di un processo stocastico, manca solamente un ultimo strumento per poter effettuare pricing tramite simulazione Monte Carlo all'interno di SFM-QuantLib, vale a dire uno o più classi che si occupino di far avvenire la simulazione in maniera

efficiente, salvarne in maniera adeguata i risultati e interfacciarsi con l'architettura già esistente per permettere le operazioni di calcolo degli NPV. Come già detto in precedenza, una simulazione Monte Carlo non è nient'altro che un lungo ciclo sugli scenari; per ciascuno di essi viene generata una traiettoria del processo stocastico desiderato e i suoi valori vengono salvati in un'apposita struttura dati che ne permetta un accesso efficiente e esaustivo tramite operazioni di interrogazione puntuale o di slicing.

Inizialmente potrebbe sembrare che, tra le due operazioni, quella che richiede lo sviluppo della logica più complessa sia la seconda. Tuttavia in questo discorso giocano un ruolo chiave le caratteristiche del linguaggio di programmazione scelto. In Python è infatti possibile appoggiarsi sulle strutture dati della libreria `numpy` e, in particolar modo, sull'array n-dimensionale della classe `ndarray` che permette di creare agevolmente una struttura dati caratterizzata da n-dimensioni nel quale salvare valori numerici. Le prestazioni di tali oggetti sono così buone che si è scelto di appoggiarsi direttamente su di essi per salvare i risultati delle simulazioni.

La parte che risulta essere più delicata è invece quella relativa alla gestione del ciclo di simulazione. Per capire come mai, è necessario fare un piccolo approfondimento relativo ad alcune caratteristiche tecniche del linguaggio Python. Esso è un linguaggio caratterizzato dalla presenza di un sistema di *typing dinamico*, che significa che le variabili non sono caratterizzate da un tipo specifico ma che questo viene dedotto dall'interprete (la parte che traduce il codice scritto in linguaggio macchina) al momento dell'esecuzione (tale pratica è comunemente detta *duck typing*). Al tempo stesso tale linguaggio è dotato di un sistema di gestione automatica della memoria, il che vale a dire che l'utente non deve preoccuparsi di allocare e liberare la memoria necessaria all'esecuzione delle funzionalità del suo programma, dato che tale aspetto viene gestito in maniera autonoma dal linguaggio di programmazione stesso. Queste caratteristiche aiutano a rendere Python un linguaggio estremamente accessibile e versatile ma hanno una grave ricaduta in termini di performance, dato che prima di eseguire una qualsiasi operazione l'interprete deve effettuare una serie di controlli relativi alla posizione in memoria e alla compatibilità dei tipi delle variabili coinvolte. Il risultato è il seguente: le operazioni contenute all'interno di cicli avvengono in maniera molto più lenta rispetto alle loro controparti in altri linguaggi di programmazione, quindi compiti come la simulazione Monte Carlo richiedono un tempo assai superiore per essere completati. Ciò nonostante si è cercato di mitigare il più possibile questa criticità ricorrendo a tecniche come il *multiprocessing* e la *vettorizzazione* delle operazioni riuscendo a migliorare notevolmente le performance di base del linguaggio.

Questi accorgimenti costituiscono la maggior parte della logica contenuta negli oggetti delle sottoclassi dell'interfaccia `PathManger` che sono responsabili sia dell'esecuzione che dell'incapsulamento dei risultati di una simulazione Monte Carlo. La simulazione Monte Carlo

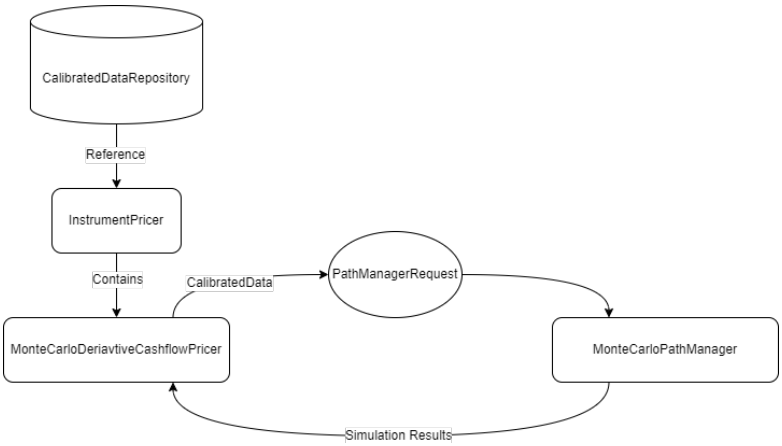

Figura 5.1 Il flusso di comunicazione tra `Pricer` e `MonteCarloPathManager`

viene richiesta nel momento in cui si rende necessario prezzare un titolo con tale metodo. Tale operazione avviene, come spiegato in precedenza, all'interno di un'apposita sottoclasse di `CashflowPricer`: `MonteCarloDerivativeCashflowPricer`. Questo comunicherà con un oggetto della classe `MonteCarloPathManger` (figlia dell'interfaccia `PathManger`) inviandogli una nuova richiesta di simulazione tramite un oggetto appartenente alla classe `SimulationRequest` contenente tutti i dati necessari alla sua esecuzione. Dopodiché la simulazione verrà lanciata e al suo termine verranno restituiti al pricer i dati necessari per il calcolo del payoff atteso del derivato e del suo fair price secondo l'Eq. (4.6). Il flusso appena descritto è rappresentato in Fig. 5.1.

5.3.4 LewisFFTHestonParametersCalibrator

Per poter utilizzare le funzionalità implementate nella classe `HestonProcess` per calcolare il prezzo di derivati, è necessario sviluppare la logica necessaria a definire il processo di stima dei suoi parametri. Come abbiamo già definito in precedenza questo processo è detto *calibrazione* e all'interno della libreria viene eseguito da oggetti appartenenti alle sottoclassi di `Calibrator`. Per fare ciò si è scelto di sfruttare la formula di Lewis dell'Eq. (4.13) e risolverla in maniera approssimata mediante l'algoritmo FFT (vedi Appendice B).

Questa procedura è giustificata sul piano teorico dalla caratterizzazione della funzione caratteristica del modello di Heston descritta nell'articolo Heston (1993). Tuttavia, tale funzione caratteristica risulta essere caratterizzata da forti problemi di instabilità numerica. Per evitare che questi vadano a impattare negativamente il risultato della calibrazione si è scelto di utilizzare una riformulazione di tale funzione caratteristica ottenuta nel lavoro di Schoutens et al. (2004) che risulta essere numericamente stabile.

Calibrare il modello di Heston equivale al risolvere il seguente problema di ottimizzazione

$$\min_{\kappa,\theta,\xi,v_0} \sum_{k=1}^{N} \left(\hat{\sigma}\left(\hat{P}_{\text{FFT}}(T_k,K_k,r,S_0,\kappa,\theta,\xi,v_0),T_k,K_k,r,S_0\right) - \hat{\sigma}\left(P_k,T_k,K_k,r,S_0\right)\right)^2,$$

dove le terne (T_k,K_k,P_k), $k=1,\ldots,N$, sono i prezzi osservabili sul mercato (P) di N opzioni call vanilla caratterizzate da strike K e maturità T. $S(0)$ è invece il valore spot del sottostante, mentre la funzione $\hat{\sigma}$ è la funzione che restituisce il valore del parametro di volatilità del modello di Black Scholes sottinteso dal prezzo dell'opzione. Infine $\hat{P}_{\text{FFT}}$ denota il prezzo dell'opzione ottenuto tramite la soluzione numerica della formula di Lewis tramite FFT. L'operazione di calibrazione avviene, dunque, per mezzo della minimizzazione della distanza euclidea media tra la superficie di volatilità osservata sul mercato e quella descritta da una determinata scelta dei parametri.

L'oggetto `LewisFFTHestonParametersCalibrator` contiene la logica necessaria per risolvere tale problema di ottimizzazione facendo uso delle funzionalità della libreria `lmfit` di Python. La scelta di tale libreria è stata fatta per garantire omogeneità nelle dipendenze dei vari `Calibrator`. Nello specifico, dopo vari esperimenti, si è scelto di affrontare tale problema di ottimizzazione mediante l'algoritmo di Powell: una procedura *derivative-free* caratterizzata da buone caratteristiche in termini di efficacia e rapidità. Il risultato della procedura di calibrazione sono i parametri da utilizzare per simulare traiettorie secondo il modello di Heston e la superficie di volatilità descritta da essi. I primi vengono incapsulati in un'istanza della classe `MonteCarloCalibratedParameters`, mentre la seconda in un oggetto della classe `VolatilitySurfaceMarketData` per poi essere restituiti all'utente che potrà salvarli in un'adeguata istanza della classe `CalibratedDataRepository` per metterli a disposizione degli oggetti incaricati delle procedure di pricing.

5.3.5 Nuovi strumenti finanziari

Per concludere si è scelto di implementare delle nuove tipologie di strumenti all'interno della libreria. In particolare, sono stati aggiunti titoli finanziari per cui il pricing mediante metodi Monte Carlo rappresenta la migliore alternativa a disposizione. Tra le numerose tipologie di derivati esotici e path-dependent a disposizione si è scelto di focalizzarsi sulle opzioni barriera e i certificati d'investimento descritti rispettivamente in Sezione 3.4.3 e Sezione 3.5.

EuropeanBarrierOption

Per le prime si è ricorsi a una strategia implementativa basata sul design pattern Decorator. È stata infatti definita la classe `EuropeanBarrierOption`, figlia di `EuropeanOption`, che viene inizializzata mediante un'opzione europea e una lista di oggetti di tipo `Barrier`. In questo modo si è definita una vasta gamma di strumenti finanziari caratterizzati da svariate forme di payoff possibili (selezionabili fornendo al costruttore l'adeguata opzione europea) e da un numero arbitrario di barriere. Insieme allo strumento sono state definite le classi `EuropeanBarrierCashflow` che ne caratterizza i flussi di cassa e `BarrierPayoff` che permette di calcolarne il payoff date le necessarie informazioni sul sottostante e quelle relative alle barriere presenti nel contratto. Quest'ultimo è un altro esempio di Decorator che va ad avvolgere e incapsulare un payoff interno aggiungendone le proprietà relative al raggiungimento dei livelli barriera. Per caratterizzare le barriere di tali strumenti è stata creata l'interfaccia `Barrier` che va a definire i seguenti metodi:

- `level` che è un getter per il livello a cui è fissata la barriera;
- `type` che restituisce un valore appartenente alle `enum BarrierType` e specifica se si tratta di una barriera knock-in o knock-out;
- `is_barrier_touched` che, dato un adeguato array contenente i valori del sottostante, stabilisce se la barriera è stata raggiunta o meno;
- `is_option_active` che determina se l'opzione è attiva o meno sulla base del raggiungimento dei livelli barriera da parte del sottostante;
- `get_fixing_dates` che ritorna una lista contenente le date per cui il valore del sottostante deve essere noto per poter determinare lo stato della barriera (e, conseguentemente, dell'opzione) alla maturità.

Dall'interfaccia `Barrier` eredita la classe `BarrierMeta` che implementa le funzionalità comuni a tutti i tipi di barriera e da essa discendono `ContinuousTimeBarrier` e `DiscreteTime Barrier` che implementano barriere con periodi di osservazione continui e discreti rispettivamente.

Certificate

Infine, sono state implementate le funzionalità necessarie per descrivere un certificato d'investimento mediante la definizione della classe `Certificate`. La struttura di questa classe è

ispirata alla natura di titolo cartolarizzato del certificato, infatti essa è definita in termini di un oggetto della classe `ZeroCouponBond` e di un portafoglio di opzioni raccolte in un oggetto della famiglia `ExoticEquityDerivative`. Come già anticipato in precedenza ci si è focalizzati sui cosiddetti certificati standard, ovvero quelli caratterizzati da un unico payoff, scritti su un singolo sottostante ed eventualmente dotati di cedole. L'utente può definire un simile contratto mediante la classe `StandardCertificate` che permette di costruire in maniera automatica, tramite un'apposita factory, i flussi di cassa relativi a un determinato stile di payoff del certificato. In questo modo l'utente effettua una descrizione qualitativa del derivato in termini di tipologia di payoff (specificabile tramite gli attributi della classe `CertificatePayoffType`) e dei suoi parametri come livello di strike, cap sul payoff, livelli barriera etc. Nello specifico il payoff associato ai flussi di cassa del portafoglio di opzioni che compone il certificato sono descritti da opportuni oggetti della classe `ExoticPayoff` che è un Composite che va a costruire una funzione di payoff complicata mediante opportune combinazioni lineari di payoff vanilla, binari e barriera.

Capitolo 6

Risultati sperimentali

Dopo aver presentato tutta la teoria necessaria a caratterizzare il problema del pricing di un derivato in stile europeo, approfondito la sua soluzione mediante l'impiego di metodi Monte Carlo e descritto in maniera dettagliata l'architettura della libreria SFMQuantLib assieme alle nuove logiche implementate, siamo interessati a valutare la qualità delle funzionalità aggiunte alla libreria. In questo capitolo presenteremo i risultati di svariati test che hanno lo scopo di validare le scelte implementative descritte nel Capitolo 5 e di dimostrare l'efficacia delle aggiunte alla libreria nel prezzare derivati esotici in maniera precisa e ottimizzata.

Procederemo presentando esperimenti di complessità crescente. In Sezione 6.1 eseguiremo delle prove per verificare che gli algoritmi di generazione implementati nella libreria rispettino le proprietà prescritte dalla teoria in termini di ordine di convergenza e presentino buone caratteristiche in termini di efficienza computazionale. Successivamente, in Sezione 6.2, testeremo il corretto funzionamento dell'architettura di pricing descritta nel Capitolo 5 andando a comparare le stime dei fair price ottenute mediante simulazione Monte Carlo con il loro valore teorico ottenibile mediante le formule del modello di Black-Scholes di Sezione 4.3. In Sezione 6.3 effettueremo la validazione della nostra implementazione del modello di Heston sia dal punto di vista della calibrazione, sia per quanto riguarda la stima del fair price del derivato. Infine in Sezione 6.4 effettueremo un prova di applicazione al mondo reale, prezzando un derivato esotico e comparando la stima ottenuta con la sua quotazione di mercato.

Tutti gli esperimenti presentati in questo capitolo sono stati effettuati sul portatile gentilmente messo a disposizione del sottoscritto da Sella Financial Markets. Tale macchina è dotata di un processore Intel i5 di tredicesima generazione (i5-1345U) con frequenza di 1.60 GHz e 32GB di memoria RAM.

6.1 Convergenza dell'integrazione Monte Carlo

In questa sezione, ci concentriamo sulla verifica degli algoritmi di generazione implementati nella libreria SFMQuantLib, con l'obiettivo di valutare sia la loro accuratezza in termini di ordine di convergenza sia la loro efficienza computazionale. Cominceremo dal caso più semplice possibile, ovvero quello della stima del valore atteso di una variabile aleatoria di distribuzione uniforme sull'intervallo $(0,1)$. Effettueremo lo stesso esperimento sia per i generatori pseudo-casuali che per i generatori quasi-casuali; tuttavia, per via dei diversi ordini di convergenza che li caratterizzano, analizzeremo i risultati separatamente.

L'esperimento è così strutturato: a ogni generatore verrà chiesto di produrre un numero N di campioni che verranno usati per calcolare una stima della media della distribuzione $\hat{\mu}(N)$. Per i risultati ottenuti tramite generatori pseudo-casuali, conoscendo i momenti della distribuzione che risultano essere

$$\mu_U = \frac{1}{2} \qquad \text{e} \qquad \sigma_U^2 = \frac{1}{12},$$

possiamo confrontare l'andamento dell'errore assoluto di stima al variare di N

$$\eta_{MC}(N) = |\hat{\mu}(N) - \mu_U|,$$

con quello teorico

$$\eta_{\text{teorico}}(N) = \frac{\sigma_U}{\sqrt{N}}.$$

Nel caso dei generatori quasi-casuali è difficile caratterizzare in maniera precisa un ordine di convergenza per la stima, quindi ci limiteremo a verificare che la sua precisione migliori con l'aumentare del numero di campioni N e a paragonare l'errore di stima rispetto a quello ottenuto con i numeri pseudo-casuali.

6.1.1 Generatori pseudo-casuali

Eseguendo l'esperimento sopra descritto con i vari algoritmi di generazione pseudo-casuali, si ottengono i risultati riportati in figura Fig. 6.1. Si è fatto variare N considerando 75 punti circa equi-spaziati compresi tra 90.000 e 10.000.000. I grafici evidenziano come l'errore di stima empirico segua in maniera abbastanza precisa quello teorico. Nello specifico, dato che la maggior parte dei punti si trovano al di sotto delle rispettive linee tratteggiate, è lecito concludere che la stima $\eta_{\text{teorico}}(N)$ è, in pratica, un'approssimazione abbastanza conservativa dell'errore e che in molti casi la precisione puntuale dei metodi Monte Carlo risulta essere più elevata.

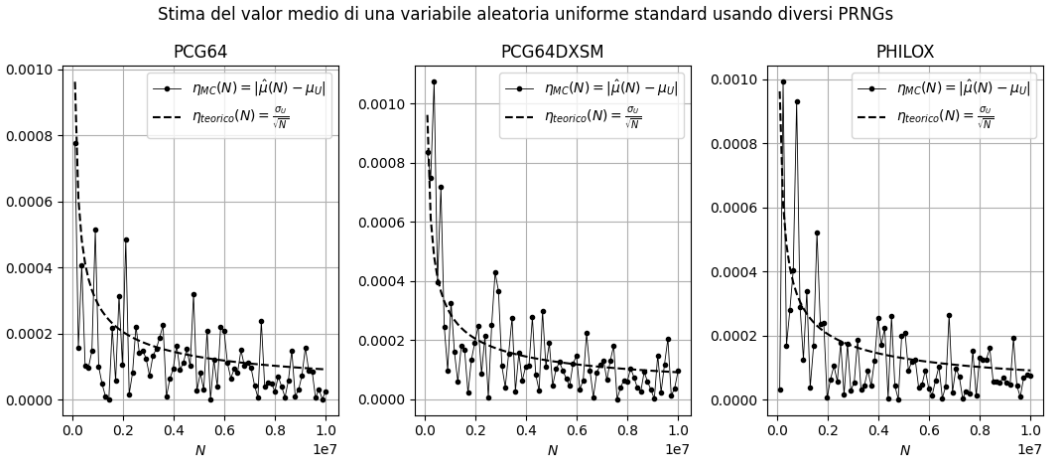

Figura 6.1 L'andamento dell'errore della stima Monte Carlo al variare del numero dei campioni generati. Si può apprezzare come i valori sperimentali ricalchino in maniera fedele il comportamento prescritto dal teorema limite centrale.

Per dare un'idea più precisa del miglioramento della precisione della stima si può far riferimento ai dati di Tabella 6.1 che evidenziano come l'aumento della dimensione del campione produca solamente un leggero miglioramento della stima Monte Carlo. In particolare, a fronte di un incremento di più di cento volte della dimensione del campione, la precisione della stima aumenta di un singolo ordine di grandezza, che è esattamente il comportamento pari a $O\left(N^{-\frac{1}{2}}\right)$ sancito dal teorema limite centrale.

Non si notano differenze sostanziali tra i tre algoritmi che risultano comportarsi in maniera pressoché identica l'uno all'altro. Questo ci suggerisce che la scelta del metodo di generazione non dovrebbe avere grossi impatti sulla qualità della stima ottenuta anche in casi più complicati. Tale ipotesi verrà testata negli esperimenti successivi. Inoltre, l'esecuzione dell'esperimento

Tabella 6.1 In tabella vengono riportati i valori della stima del valor medio $\mu_U = \frac{1}{2}$. Si noti come, pur aumentando di più di cento volte la dimensione del campione, l'errore di stima migliori solo di un ordine di grandezza, esattamente come predetto dalla teoria.

Metodo di Generazione	N	$\hat{\mu}(N)$	$\eta_{MC}(N)$
PCG64	90.000	0,500776	7,7588E-04
	10.000.000	0,500024	2,4222E-05
PCG64DXSM	90.000	0,499165	8,3530E-04
	10.000.000	0,500095	9,5290E-05
PHILOX	90.000	0,500033	3,2731E-05
	10.000.000	0,500074	7,4180E-05

non ha prodotto alcun tipo di errore o warning da parte del programma. L'unica sostanziale differenza relativa al funzionamento dei tre metodi di generazione è quella relativa alle loro performance, ma rimandiamo tali analisi alla Sezione 6.1.3 per una valutazione complessiva comprendente anche gli algoritmi di generazione basati su sequenze a bassa discrepanza analizzati qui di seguito.

6.1.2 Generatori quasi-casuali

Rivolgiamo ora la nostra attenzione ai generatori di numeri casuali basati sulle sequenze a bassa discrepanza. Per via di quanto discusso nel Capitolo 2, ci si aspetta che questi algoritmi migliorino sensibilmente la qualità della stima per via della loro capacità di riempire in maniera uniforme ed equa lo spazio campionario. Per verificare l'impatto relativo all'impiego di questi metodi di generazione sulla qualità della stima ottenuta, si è condotto l'esperimento precedentemente descritto considerando i medesimi valori per la numerosità del campione N, ottenendo i risultati sintetizzati in Fig. 6.2. Un veloce sguardo all'ordine di grandezza degli assi verticali di tali grafici rivela un miglioramento della qualità della stima di circa tre ordini di grandezza a parità del parametro N rispetto agli esiti del precedente esperimento. Inoltre, le curve rappresentate tendono a "schiacciarsi" in maniera molto più aggressiva rispetto a quelle ottenute dagli algoritmi analizzati in precedenza, il che suggerisce un ordine di convergenza della stima Monte Carlo più alto.

Tale intuizione è confermata dai risultati di Tabella 6.2, in cui si evidenzia una performance estremamente sorprendente dell'algoritmo basato sulla sequenza di Sobol che, con un campionamento di appena 90.000 valori, permette di ottenere una stima con una precisione superiore di cinque ordini di grandezza rispetto al miglior risultato ottenuto usando valori generati dagli algoritmi precedenti. Tra i due generatori, quest'ultimo sembra essere quello capace di ottenere le prestazioni migliori quando è necessario stimare valori con un minor numero di dati, mentre l'algoritmo basato sulle sequenze di Halton è caratterizzato da una performance che tende a migliorare in maniera nettamente più marcata all'aumentare del numero dei campioni. Inoltre, i dati riportati in Tabella 6.2 sembrano suggerire che questo generatore permetta di ottenere un ordine di convergenza vicino a $O\left(N^{-1}\right)$, che è superiore a quello dei precedenti algoritmi.

In definitiva, entrambi gli algoritmi permettono di migliorare di svariati ordini di grandezza la precisione della stima Monte Carlo; tuttavia, quello basato sulle sequenze di Sobol sembra essere di gran lunga il più potente dei due. L'esecuzione del programma che permette di condurre l'esperimento produce la generazione di un warning da parte del motore di generazione delle sequenze di Sobol che ci informa che, per ottenere risultati ottimali, il numero di valori

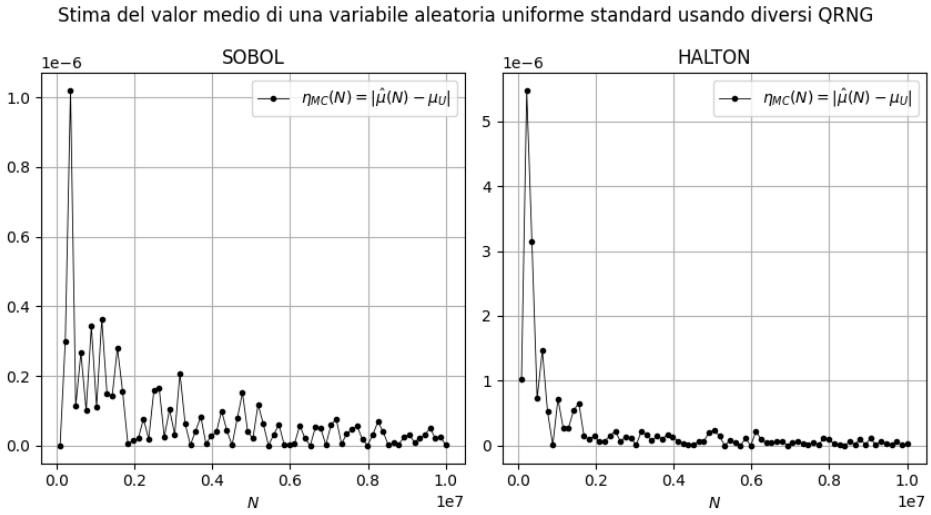

Figura 6.2 L'andamento dell'errore della stima Monte Carlo ottenuta impiegando generatori basati su sequenze a bassa discrepanza al variare del numero dei campioni generati. Se paragonati ai grafici della Fig. 6.1, si nota un netto miglioramento in termini di precisione della stima di circa tre ordini di grandezza.

Tabella 6.2 La tabella riassume i valori relativi alla precisione della stima ottenuti mediante l'impiego di generatori quasi-casuali. In generale, i risultati sono migliori rispetto a quelli ottenuti con i generatori pseudo-casuali di Tabella 6.1. Le righe relative al generatore basato sulla sequenza di Halton evidenziano un ordine di convergenza della stima pari a $O\left(N^{-1}\right)$.

Metodo di Generazione	N	$\hat{\mu}(N)$	$\eta_{MC}(N)$
SOBOL	90.000	0,500000	5,3417E-10
	10.000.000	0,500000	9,8107E-10
HALTON	90.000	0,500001	1,0162E-06
	10.000.000	0,500000	2,3381E-08

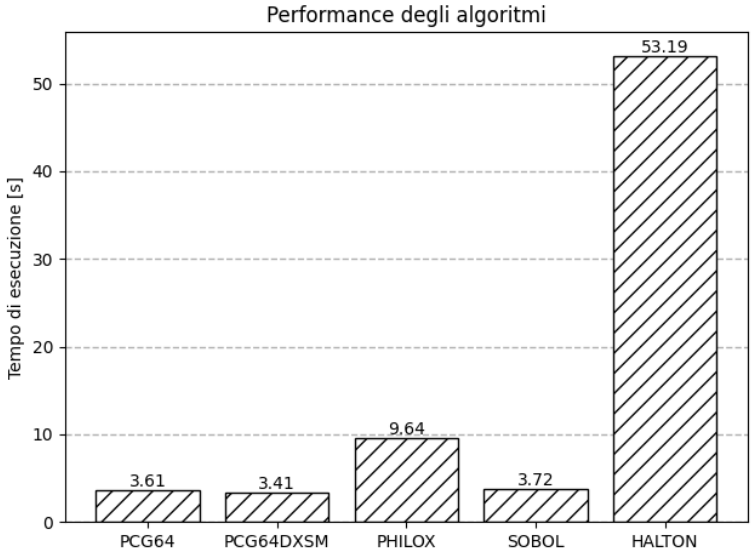

Figura 6.3 Il tempo impiegato da ciascun algoritmo di generazione per completare l'esperimento descritto all'inizio della Sezione 6.1.

campionati dalla sequenza deve essere una potenza di due; tuttavia, ciò non sembra avere impatti indesiderati in pratica. Negli esperimenti successivi verificheremo se la superiorità di questi algoritmi rispetto ai precedenti viene mantenuta anche in contesti più complessi come quello del pricing di derivati.

6.1.3 Performance computazionali

L'esperimento ha permesso di valutare anche le prestazioni dal punto di vista del costo computazionale dei vari algoritmi. Nello specifico, si è scelto di registrare i dati relativi al tempo impiegato da ciascun algoritmo a completare la procedura sancita dall'esperimento. Essi sono poi stati raccolti nel grafico di Fig. 6.3. Osserviamo che i generatori PCG64, PCG64DXSM e SOBOL hanno generato i valori in un tempo paragonabile, mentre l'algoritmo PHILOX è risultato essere leggermente più lento. Le performance peggiori si sono rivelate essere quelle dell'algoritmo basato sulle sequenze di Halton, che ha impiegato poco meno di un minuto a completare l'operazione, risultando assai più lento di tutti i precedenti.

Una potenziale causa di questo fenomeno può essere trovata nella dimensionalità del problema. Infatti, le sequenze a bassa discrepanza sono specialmente utili nella soluzione di problemi a dimensione elevata, quindi può essere che la loro implementazione non sia molto ottimizzata nel caso di campionamenti a bassa dimensione. Per questo motivo, si è deciso di ripetere l'esperimento descritto all'inizio della sezione chiedendo però agli algoritmi

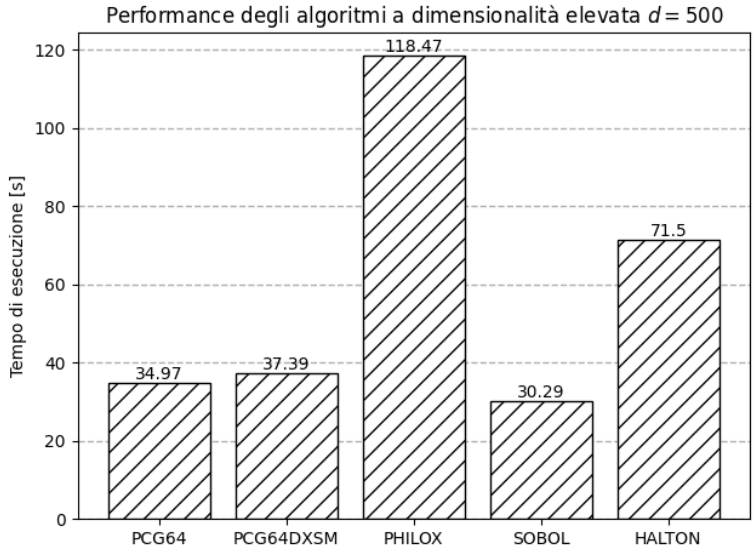

Figura 6.4 I risultati ottenuti ripetendo l'esperimento in uno spazio a dimensione elevata. In questo contesto, l'algoritmo basato sulle serie Halton risulta essere più competitivo ma rimane pur sempre meno efficiente di altri generatori implementati.

di produrre valori appartenenti all'ipercubo unitario in dimensione 500. Per evitare che l'esecuzione dell'esperimento richiedesse troppo tempo, si è deciso di ridurre il massimo valore della numerosità del campione facendo variare N tra 90.000 e 100.000. I risultati di questo secondo esperimento sono riportati in Fig. 6.4. Osserviamo che per i generatori PCG64, PCG64DXSM, SOBOL e PHILOX la situazione non è cambiata: i primi tre risultano essere pressoché equivalenti, mentre l'ultimo esibisce un tempo di calcolo pari a poco più del triplo dei precedenti. Al contrario, la performance relativa del generatore basato sulle sequenze di Halton risulta essere migliore rispetto al risultato precedente, pur rimanendo sensibilmente più lento in termini assoluti.

6.1.4 Commento dei risultati ottenuti

Questo primo esperimento ha messo in luce tutti i vantaggi e le criticità derivanti dalle scelte implementative effettuate. La scelta di sfruttare algoritmi già validati da altri utenti, come quelli presenti nelle librerie `numpy` e `scipy`, garantisce ottimi risultati dal punto di vista della qualità delle stime ottenute e, in generale, un buon livello in termini di performance. Infatti, le funzionalità centrali delle due librerie sopracitate sono scritte in linguaggio C che, una volta compilato, garantisce velocità di esecuzione assai superiori rispetto a quelle ottenibili dall'interprete di Python. Tuttavia, appoggiarsi su soluzioni già pronte fa sì che si perda il

controllo su alcuni dettagli dell'implementazione e ciò può portare a risultati inattesi come quelli relativi alla performance del generatore basato sulla sequenza di Halton della Sezione 6.1.3.

Complessivamente, questi primi risultati possono essere considerati molto soddisfacenti e ci fanno ben sperare per quanto riguarda la buona riuscita degli esperimenti successivi.

6.2 Primi test di pricing

Dopo aver verificato il corretto funzionamento degli algoritmi di generazione, rivolgiamo la nostra attenzione verso un esperimento leggermente più complesso. Desideriamo utilizzare le nuove funzionalità introdotte nella libreria per calcolare il fair price di derivati mediante metodi Monte Carlo e confrontarlo con i valori ottenuti mediante l'impiego di formule analitiche. Per questo motivo, in questo primo esperimento, calcoleremo il prezzo dei derivati nel modello di Black-Scholes descritto in Sezione 4.3, che permette di definire formule semplici per calcolare il fair price di varie famiglie di contratti. L'obiettivo dell'esperimento è verificare che il prezzo stimato mediante simulazione Monte Carlo sia sufficientemente vicino a quello ottenibile in maniera analitica. Come abbiamo già evidenziato in precedenza, il modello di Black-Scholes suppone che il sottostante si muova secondo un moto browniano geometrico e tale processo può essere simulato in maniera sufficientemente precisa usando le funzionalità descritte nel Capitolo 5. Questo test ha quindi lo scopo di fornire un'ulteriore validazione del funzionamento dei vari generatori di valori casuali e, al contempo, di verificare la corretta implementazione del nuovo pricer, del processo stocastico e delle funzionalità dei vari strumenti.

In questo esperimento si fissano i valori dei parametri del modello $r = 5\%$, $\sigma = 20\%$ e $S(t_0) = 100$, andando a creare gli adeguati `CalibratedDataObject`, per poi creare un'istanza dello strumento finanziario che si desidera prezzare e calcolarne l'NPV sia tramite simulazione Monte Carlo sia tramite un'adeguata formula analitica. L'esperimento ha successo se le due quantità sono sufficientemente simili. Gli strumenti che andremo a prezzare sono opzioni vanilla, opzioni binarie e opzioni barriera semplici, per i quali è possibile ricavare una formula analitica per il fair value.

6.2.1 Opzioni vanilla

Come abbiamo approfondito in maniera dettagliata nel Capitolo 4, il modello di Black-Scholes permette di ricavare la formula (4.11) per il calcolo del prezzo di un'opzione call vanilla. Nello specifico, tale formula è contenuta all'interno della logica dei pricer di SFMQuantLib, quindi

Tabella 6.3 Le stime del fair price di un'opzione call vanilla nel modello di Black-Scholes ottenute mediante simulazione Monte Carlo, usando vari metodi di generazione di valori casuali. Gli errori sono calcolati rispetto al valore teorico $C_{\text{Vanilla}}(t_0) = 2.3730$.

	Metodo di Generazione	NPV stimato	Errore Assoluto	Errore Relativo
PRNG	PCG64	2.3757	0.0027	0.115%
	PHILOX	2.3708	0.0022	0.093%
	PCG64DXSM	***2.3724***	***0.0006***	***0.024%***
QRNG	***HALTON***	***2.3731***	***0.0001***	***0.005%***
	SOBOL	2.3727	0.0003	0.014%

l'esperimento risulta assai agevole da condurre. Creeremo due opzioni call vanilla at the money identiche con time-to-maturity pari a un mese e specificheremo che desideriamo prezzare la prima usando una simulazione Monte Carlo (passando al suo metodo `set_model` il valore `Model.MONTECARLO`) mentre la seconda usando la (4.11) (passando il valore `Model.BLACK`[1]). Visto il discreto livello di ottimizzazione del codice, si è potuto settare un valore molto alto per il numero di traiettorie. Nello specifico, i risultati sperimentali di seguito riportati e commentati sono stati ottenuti generando $2^{21} \simeq 2,000,000$ scenari.

Il prezzo dell'opzione vanilla considerata secondo il modello di Black-Scholes risulta essere

$$C_{\text{Vanilla}}(t_0) = 2.3730.$$

I risultati ottenuti tramite il pricing Monte Carlo dell'opzione sono riportati in Tabella 6.3. I valori contenuti in termini di errore assoluto (sempre inferiori a 10^{-2}) ed errore relativo (sempre al di sotto dell'1%) confermano la corretta implementazione e funzionamento della classe `MonteCarloPathManager` e del payoff delle opzioni vanilla. Nonostante il gran numero di scenari considerati, il programma viene eseguito in un tempo relativamente contenuto, complessivamente pari a circa 28s. Le differenze tra i tempi di esecuzione dei vari algoritmi ricalcano il comportamento dimostrato da questi nella seconda prova della Sezione 6.1.3.

6.2.2 Opzioni binarie

Supponendo che le dinamiche neutrali al rischio del sottostante siano quelle del modello di Black-Scholes, ovvero che questo si muova secondo un moto browniano geometrico con

[1]Per essere precisi, all'interno del pricer si fa uso della formula di Black che calcola il prezzo di una call come $C(t_0) = e^{-r(T-t_0)}\,(F_S(t_0,T)N(d_1) - KN(d_2))$, dove $F_S(t_0,T)$ è il prezzo forward del sottostante S relativo alla maturità T al tempo t_0

coefficiente di drift pari al tasso d'interesse privo di rischio e una certa volatilità, è possibile caratterizzare una formula per il fair price delle opzioni binarie presentate in Sezione 3.4.2. Nel caso specifico di un'opzione call, basta ricordare che per la (4.5) si ha che

$$\begin{aligned} C_{\text{Binary}}(t_0) &= e^{-r(T-t_0)}\mathbb{E}\left[\mathbf{1}_{(K,+\infty)}S(T)\middle|\mathscr{F}_{t_0}\right] \\ &= e^{-r(T-t_0)}\mathbb{Q}\left(S(T) > K|S(t_0)\right) \\ &= e^{-r(T-t_0)}N(d_2), \end{aligned} \tag{6.1}$$

dove T è la maturità dell'opzione, K il suo strike price, N è la cdf di una distribuzione gaussiana standard e d_2 è dato dall'Eq. (4.9). Osservando che

$$\begin{aligned} C_{\text{Binary}}(t_0) &= e^{-r(T-t_0)}\left(1-\mathbb{Q}\left(S(T) \leq K|S(t_0)\right)\right) \\ &= e^{-r(T-t_0)} - P_{\text{Binary}}(t_0), \end{aligned}$$

si ricava agevolmente anche una formula per il prezzo di una opzione put.

Queste formule ci permettono di ripetere esattamente lo stesso esperimento del caso precedente, anche per questa tipologia di derivato. Utilizzando la formula (6.1) per una opzione call binaria at the money e con time-to-maturity pari a 3 mesi si ottiene

$$C_{\text{Binary}}(t_0) = 0.5145.$$

Come nel caso precedente, si è stimato il fair price di questo derivato mediante simulazione Monte Carlo, generando 2^{21} diversi scenari, ottenendo i risultati riportati in Tabella 6.4. Al di là delle minime differenze nella performance dei vari metodi di generazione, i risultati ottenuti sono tutti di altissima qualità, con errori assoluti dell'ordine di grandezza di 10^{-4}, il che è estremamente positivo e conferma le capacità delle classi della libreria di produrre stime non distorte dei fair price di derivati. Anche in questo caso, la velocità di esecuzione del programma è soddisfacente con un tempo complessivo pari a 26s.

6.2.3 Opzioni barriera

Il modello di Black-Scholes permette di ricavare formule analitiche per calcolare il prezzo di alcune opzioni barriera. Nello specifico, è possibile determinare il fair price di contratti contraddistinti da un payoff di tipo vanilla e dalla presenza di una singola barriera. In questo esperimento ci focalizzeremo sulle opzioni *down-and-out*, caratterizzate da una barriera knock-out fissata a un valore inferiore allo spot del sottostante, e sulle opzioni *down-and-in*, analoghe

Tabella 6.4 Le stime del fair price di un'opzione call binaria nel modello di Black-Scholes ottenute mediante simulazione Monte Carlo usando vari metodi di generazione di valori casuali. Gli errori sono calcolati rispetto al valore teorico $C_{\text{Binary}}(t_0) = 0.5145$.

	Metodo di Generazione	NPV stimato	Errore Assoluto	Errore Relativo
PRNG	PCG64	0.5148	0.0003	0.064%
	PHILOX	***0.5142***	***0.0002***	***0.044%***
	PCG64DXSM	0.5142	0.0003	0.055%
QRNG	HALTON	0.5146	0.0002	0.030%
	SOBOL	***0.5144***	***0.0001***	***0.014%***

alle precedenti ma caratterizzate da una barriera di tipo knock-in. A differenza del caso precedente, il procedimento per ricavare le formule è abbastanza complesso, quindi si rimanda al Capitolo 24 del testo di Hull (1993) per i passaggi e le espressioni precise. Anche in questo caso, andremo a prezzare ciascuna opzione sia tramite una simulazione Monte Carlo sia tramite la formula analitica e andremo a confrontare i risultati ottenuti in termini di errore assoluto e relativo di stima. Per quanto riguarda il numero di scenari, tutti i risultati riportati di seguito sono stati ottenuti generando 2^{21} traiettorie del processo stocastico del sottostante.

Si è scelto di incominciare a partire dal pricing di un contratto di tipo down-and-out con livello barriera pari a 91, strike pari a 85 e time-to-maturity pari a 3 mesi. Utilizzando le formule del libro di Hull si ottiene un valore teorico del fair price pari a

$$C_{\text{DO}}(t_0) = 14.8365.$$

I risultati di un primo esperimento sono riportati in Tabella 6.5. Osservando tali valori, si nota immediatamente un comportamento nettamente peggiore rispetto ai casi precedenti, non tanto per quanto riguarda l'errore relativo, che si mantiene al di sotto dell'1%, quanto più per

Tabella 6.5 Le stime del fair price nel modello di Black-Scholes di un'opzione call down-and-out con barriera al livello 91, ottenute mediante simulazione Monte Carlo usando vari metodi di generazione di valori casuali. Gli errori sono calcolati rispetto al valore teorico $C_{\text{DO}}(t_0) = 14.8365$. Si può osservare che i risultati ottenuti sono caratterizzati da un buon livello di distorsione che ne peggiora l'accuratezza.

	Metodo di Generazione	NPV stimato	Errore Assoluto	Errore Relativo
PRNG	***PCG64***	***14.7255***	***0.1110***	***0.754%***
	PHILOX	14.7156	0.1209	0.822%
	PCG64DXSM	14.7175	0.1189	0.808%
QRNG	HALTON	14.7168	0.1196	0.813%
	SOBOL	***14.7215***	***0.1149***	***0.781%***

Tabella 6.6 Le stime del fair price nel modello di Black-Scholes di un'opzione call down-and-out con barriera al livello 91, ottenute mediante simulazione Monte Carlo usando vari metodi di generazione di valori casuali. Gli errori sono calcolati rispetto al valore teorico $C_{\mathrm{DO}}(t_0) = 14.8365$. Aggiustando la sensibilità della barriera si è drasticamente ridotta la distorsione della stima.

	Metodo di Generazione	NPV stimato	Errore Assoluto	Errore Relativo
PRNG	***PCG64***	***14.8399***	***0.0035***	***0.023%***
	PHILOX	14.8304	0.0061	0.041%
	PCG64DXSM	14.8319	0.0046	0.031%
QRNG	HALTON	14.8313	0.0052	0.035%
	SOBOL	***14.8352***	***0.0013***	***0.009%***

i valori assunti dall'errore assoluto che non scendono mai al di sotto della soglia di 10^{-2} e risultano essere tutti estremamente vicini tra di loro. Inoltre, gli algoritmi basati su sequenze a bassa discrepanza non hanno prodotto risultati sensibilmente migliori di quelli "tradizionali". Tutto ciò ci fa pensare che qualcosa, all'interno del processo di stima del fair price di questi strumenti, stia introducendo un qualche tipo di *distorsione*, ovvero un sistematico errore di stima (in questo caso di sottostima) del valore teorico. In generale, i fenomeni di distorsione possono essere dovuti a un gran numero di fattori ma, visti i risultati precedenti, è lecito concludere che la causa più probabile dell'errore, in questo caso, sia legata in qualche modo al comportamento delle barriere. Tali oggetti determinano se il payoff di un contratto è attivo o meno a seconda del comportamento del sottostante. Nel mondo reale, la barriera si attiva nell'istante in cui il valore del sottostante raggiunge o oltrepassa il livello prefissato; tuttavia, a livello di simulazione numerica, questi oggetti sono caratterizzati da un livello di tolleranza che permette di determinare il verificarsi degli eventi barriera. Visto che tutte le stime del fair price del contratto sono più basse del valore teorico, è lecito ipotizzare che la sensibilità della barriera possa essere troppo elevata, ovvero il valore della sua tolleranza risulti essere impostato su un valore troppo grande.

Per questo motivo si è ripetuto l'esperimento impostando la tolleranza della barriera su un valore più basso[2]. I risultati ottenuti sono riportati in Tabella 6.6 e confermano l'ipotesi appena effettuata. Semplicemente andando ad agire sul livello di tolleranza della barriera si è ridotto drasticamente il quantitativo di distorsione presente nella stima. L'errore assoluto è ora dell'ordine di grandezza di 10^{-3} che, pur rimanendo assai superiore a quello ottenuto nel caso dei derivati più semplici, è sufficientemente contenuto. Un altro risultato interessante è quello legato alla performance dell'algoritmo di generazione basato sulle sequenze di Halton che

[2]La tolleranza della barriera è espressa in termini di tolleranza relativa e tolleranza assoluta. Nello specifico, per questo esperimento si è andati ad agire sulla tolleranza relativa, abbassandola da $1e-2$ a $6.5e-3$, mentre la tolleranza assoluta è rimasta inalterata al valore $1e-5$.

Tabella 6.7 Le stime del fair price nel modello di Black-Scholes di un'opzione call down-and-in con barriera al livello 91, ottenute mediante simulazione Monte Carlo usando vari metodi di generazione di valori casuali. Gli errori sono calcolati rispetto al valore teorico $C_{\mathrm{DI}}(t_0) = 0.4834$.

	Metodo di Generazione	NPV stimato	Errore Assoluto	Errore Relativo
PRNG	***PCG64***	***0.4850***	***0.0016***	***0.340%***
	PHILOX	0.4866	0.0032	0.655%
	PCG64DXSM	0.4866	0.0033	0.669%
QRNG	HALTON	0.4885	0.0051	1.043%
	SOBOL	***0.4847***	***0.0013***	***0.271%***

risultano nettamente inferiori sia rispetto a quello basato sulla sequenza a bassa discrepanza di Sobol, sia rispetto a quelle ottenute dai più classici generatori di numeri pseudo-casuali. Questo risultato non è del tutto sorprendente: i metodi di generazione basati su tali sequenze sono noti per dare problemi quando la dimensionalità del problema incomincia a crescere e, a differenza dei casi precedenti, la valutazione del payoff di un'opzione barriera viene effettuata usando i valori dell'intera traiettoria, quindi questo metodo di generazione risulta essere il meno efficace.

Infine, si è effettuata la stessa tipologia di esperimento usando un'opzione call di tipo down-and-in con livello barriera pari a 91, strike price pari a 85 e time-to-maturity di 3 mesi il cui prezzo teorico dato dalle formule del testo di Hull è pari a

$$C_{\mathrm{DI}}(t_0) = 0.4834.$$

I risultati ottenuti usando stime Monte Carlo sono riportati in Tabella 6.7. Anche se gli errori relativi risultano essere più alti rispetto all'esperimento precedente, ciò è principalmente dovuto al fatto che la quantità stimata è più piccola. Infatti, gli errori assoluti restano dello stesso ordine di grandezza dell'esperimento precedente pari a 10^{-3}. Anche in questo caso, l'algoritmo di Halton ha prodotto i risultati peggiori fra tutti i metodi di generazione, confermando la non idoneità di questo algoritmo in contesti fortemente multidimensionali.

Prezzare opzioni barriera si è rivelato essere un compito più complesso dal punto di vista computazionale, non solo per la sensibilità della stima rispetto al livello barriera, ma anche per via del tempo di esecuzione del programma che è quasi raddoppiato rispetto al caso precedente. Infatti, il tempo necessario a eseguire l'esperimento è stato pari a circa 48s.

6.2.4 Commento dei risultati ottenuti

I risultati sperimentali presentati, oltre a confermare il corretto funzionamento degli algoritmi di generazione già investigato in Sezione 6.1, validano definitivamente l'implementazione delle funzionalità della classe `MonteCarloPathManager`, che garantisce l'esecuzione di simulazioni Monte Carlo in maniera rapida, efficiente e capace di produrre stime precise e non distorte. Seppur non trattandosi ancora di una definitiva conferma dell'utilizzabilità delle funzioni aggiunte alla libreria in contesti reali, questi risultati confermano l'aderenza dei risultati ottenuti tramite le simulazioni Monte Carlo a quelli prescritti da risultati teorici, il che equivale a dire che le distribuzioni di probabilità descritte dal modello (che nel caso di questi esperimenti è quello di Black-Scholes) vengono ben approssimate dai valori effettivamente generati.

Gli esperimenti hanno anche portato in luce qualche criticità. In particolare, si è visto che un ruolo fondamentale nella precisione delle stime ottenute viene giocato dalle sensibilità delle barriere. Seppur si è mostrato come quest'ultima possa essere regolata per migliorare la qualità dei risultati, ciò rappresenta comunque un punto critico dato che è difficile stabilire come modificare le tolleranze delle barriere in assenza di un valore teorico di riferimento. Un'altra criticità evidenziata in modo netto è quella relativa alla bassa qualità delle stime prodotte dal generatore basato sulle sequenze di Halton nel caso di derivati dotati di barriera. Questo è un risultato assai rilevante, di cui sarà fondamentale tenere conto per un corretto utilizzo della libreria in futuro.

6.3 Il modello di Heston

A questo punto, ci focalizziamo sul testare il corretto funzionamento delle logiche relative al modello di Heston implementate in SFMQuantLib. Le proprietà del modello, approfondite nel dettaglio in Sezione 4.4, lo rendono assai promettente come metodo per prezzare i derivati, tuttavia la simulazione del processo stocastico del sottostante risulta essere un problema numericamente difficile da affrontare. Ai fini di ridurre al minimo gli effetti negativi di tale instabilità si è utilizzato uno schema di discretizzazione complesso e avanzato descritto nel Capitolo 5 di cui ora desideriamo verificare il corretto funzionamento. Al tempo stesso si desidera testare l'efficacia del processo di calibrazione nello stimare parametri coerenti con le assunzioni del modello e che riescano a riprodurre in maniera fedele i dati provenienti dal mercato.

Per verificare la corretta implementazione delle funzionalità relative al modello di Heston, andremo a eseguire una calibrazione del modello su dati provenienti dal mondo reale e ad

analizzare le caratteristiche della superficie di volatilità ottenuta. A seguito di questo eseguiremo un secondo test, nel quale andremo a verificare che, fissati degli specifici parametri del modello, la stima Monte Carlo del prezzo di un'opzione call sia sufficientemente vicina al valore dato dalla formula di Lewis utilizzata nel processo di calibrazione. In altre parole, con il primo test andremo a descrivere la capacità del modello e del processo di calibrazione di "catturare" e descrivere accuratamente la situazione del mercato, mentre nella seconda prova andremo a verificare la consistenza delle stime ottenute tramite simulazione Monte Carlo con la situazione del mercato rilevata dalla calibrazione del modello. Separare l'esperimento in questo modo ci consente di analizzare le performance del calibratore e della simulazione del processo in maniera distinta fornendoci un chiaro prospetto del funzionamento dei vari elementi.

6.3.1 Calibrazione

Per testare il corretto funzionamento delle logiche implementate nella classe che permette di effettuare la calibrazione del modello di Heston, si è deciso di effettuare una prova di calibrazione su dei dati provenienti dal mondo reale. Per calibrare un modello di pricing di derivati su sottostante equity è necessario conoscere il prezzo di mercato P di un numero sufficiente di opzioni call vanilla per un range sufficientemente ampio di maturità T e prezzi strike K. Conoscendo tali dati, il prezzo spot del sottostante $S(t_0)$ e avendo a disposizione una stima della struttura per scadenze del tasso d'interesse privo di rischio $r(t_0,\cdot)$, è possibile costruire la cosiddetta *superficie di volatilità*, ovvero la superficie dello spazio tridimensionale che si viene a costruire andando a rappresentare le terne

$$(T, K, \hat{\sigma}(P,T,K,r(t_0,T),S(t_0)))^{\intercal} \in \mathbb{R}^3,$$

di ciascun'opzione. Come già accennato nel Capitolo 5, calibrare un modello significa andare a scegliere i parametri che meglio descrivono la superficie di volatilità osservabile sul mercato, risolvendo un opportuno problema di ottimizzazione.

La classe `LewisFFTHestonParametersCalibrator` incapsula la logica necessaria alla calibrazione del modello di Heston e ai fini di testarne il corretto funzionamento è stato effettuato il bootstrapping della struttura per scadenza dell'indice ESTR (Euro Short-Term Rate) che, come spiegato nel Capitolo 3, è generalmente considerato il tasso privo di rischio per i mercati europei. Tale procedura è stata fatta utilizzando funzionalità già presenti all'interno della libreria e ampiamente testate che estrapolano le informazioni relative a uno specifico tasso d'interesse partendo dalle quotazioni di mercato di particolari derivati detti *contratti swap*, il cui funzionamento non viene qui approfondito. Una volta estratta la curva dei tassi si

sono scaricati i valori relativi al prezzo di 114 opzioni call vanilla sull'indice Swiss Market Index (SMI) in data 3 Giugno 2024. Tali valori sono stati raccolti in un adeguato oggetto della classe `VolatilitySurfaceMarketData` e passati al calibratore perché effettuasse la stima dei parametri del modello.

Il processo di calibrazione è stato eseguito in 4 minuti e 41 secondi (281 secondi totali) e ha prodotto le seguenti stime dei parametri:

$$\begin{cases} v_0 = 0.0181 \\ \theta = 0.0491 \\ \kappa = 0.6532 \\ \xi = 0.4267 \\ \rho = -0.4994 \end{cases}.$$

Tali valori descrivono una volatilità spot del sottostante pari a $\sqrt{v_0} \approx 13.45\%$ che tenderà a crescere verso il valore $\sqrt{\theta} \approx 22.15\%$ in un arco di tempo pari a $\frac{1}{\kappa} = 1.5309$ anni. La volatilità del processo varianza risulta essere del 42.67%, mentre il valore negativo assunto dal parametro di correlazione $\rho = -49.94\%$ indica una distribuzione dei ritorni logaritmici caratterizzata da una skewness negativa. Tutti i valori stimati sono sufficientemente realistici e aderenti con le assunzioni del modello di Heston, il che è un segno assai positivo per il corretto funzionamento del processo di calibrazione.

Oltre alle stime dei parametri, il processo di calibrazione ha restituito la superficie di volatilità da esse implicata che può essere osservata in Fig. 6.5 assieme alle volatilità implicite osservate sul mercato. L'analisi di tali risultati parte dall'osservare che, in generale, il processo di calibrazione sembra essere capace di catturare il complessivo andamento dei dati di mercato producendo una superficie che "curvi" approssimativamente allo stesso modo. Tuttavia, si nota un comportamento nettamente differente lungo l'asse temporale delle maturità delle opzioni: le volatilità implicite di quelle caratterizzate da un breve time-to-maturity descrivono uno smile caratterizzato da una curvatura molto elevata che non si è riusciti a catturare in maniera fedele durante il processo di calibrazione, mentre per i titoli a maturità più lontana nel tempo si riesce a catturare la volatilità in maniera quasi perfetta. Questo fenomeno è messo in luce in maniera ancora più evidente dai grafici di Fig. 6.6. Le cause di questo fenomeno possono essere sostanzialmente due:

1. la logica utilizzata per il processo di calibrazione non riesce a trovare in maniera efficace i valori corretti dei parametri che descrivono la situazione espressa dai dati di mercato;

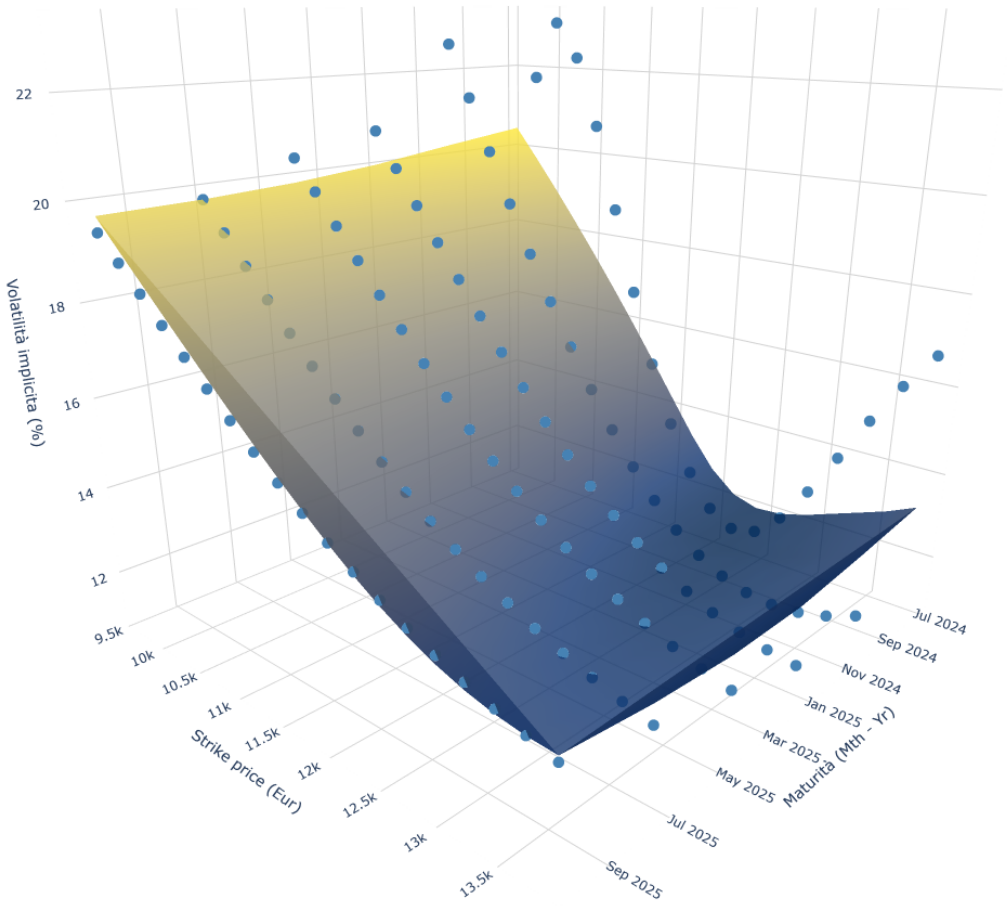

Figura 6.5 La superficie di volatilità del modello di Heston calibrata a partire dai dati di mercato delle opzioni sull'indice SMI in data 3 Giugno 2024 (rappresentati dai marker). L'aderenza della superficie ai dati di mercato è migliore per le opzioni a maturità più lontana nel tempo, mentre il comportamento delle opzioni a maturità più breve nel tempo è rappresentato in maniera meno fedele.

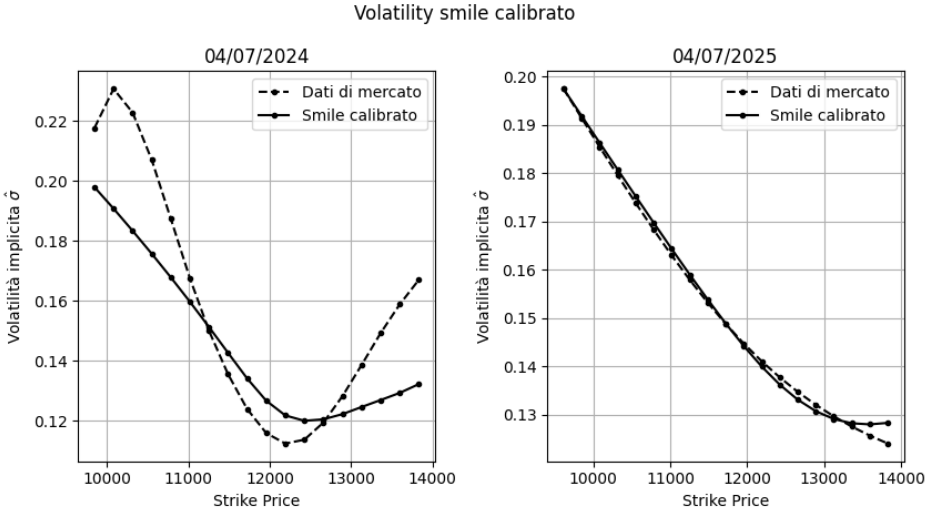

Figura 6.6 Confronto tra alcuni volatility smile ottenuti dal processo di calibrazione e quelli osservati sul mercato. Si osserva come nella figura di sinistra le due curve mostrino un comportamento assai diverso, mentre nella figura di destra le due curve sono quasi identiche.

2. il modello di Heston non è dotato di sufficienti gradi di libertà per esprimere una curvatura così estrema.

Per capire quale di queste due possa essere più probabilmente la causa, si è deciso di ripetere l'esperimento precedente andando a rimuovere dai dati di mercato utilizzati per la calibrazione quelli relativi alle opzioni con scadenza al 4 Luglio 2024 per i quali si osservava un comportamento più problematico. La superficie di volatilità ottenuta al termine del processo di calibrazione, durato circa 5 minuti (305 secondi), è riportata in Fig. 6.7. Nell'immagine si osserva che in questo caso la superficie calibrata è quasi perfettamente aderente ai dati di mercato per tutte le scadenze, il che suggerisce che le logiche del calibratore funzionano adeguatamente e il comportamento precedente era molto probabilmente legato alle caratteristiche del modello di Heston stesso. Come nel caso precedente, si riportano i confronti tra alcuni smile estratti dalla superficie calibrata e quelli osservabili sul mercato in Fig. 6.8.

Un fatto degno di nota riguarda i valori dei parametri stimati da questa seconda calibrazione che risultano essere

$$\begin{cases} v_0 = 0.0154 \\ \theta = 0.0287 \\ \kappa = 4.7890 \\ \xi = 0.8885 \\ \rho = -0.6 \end{cases},$$

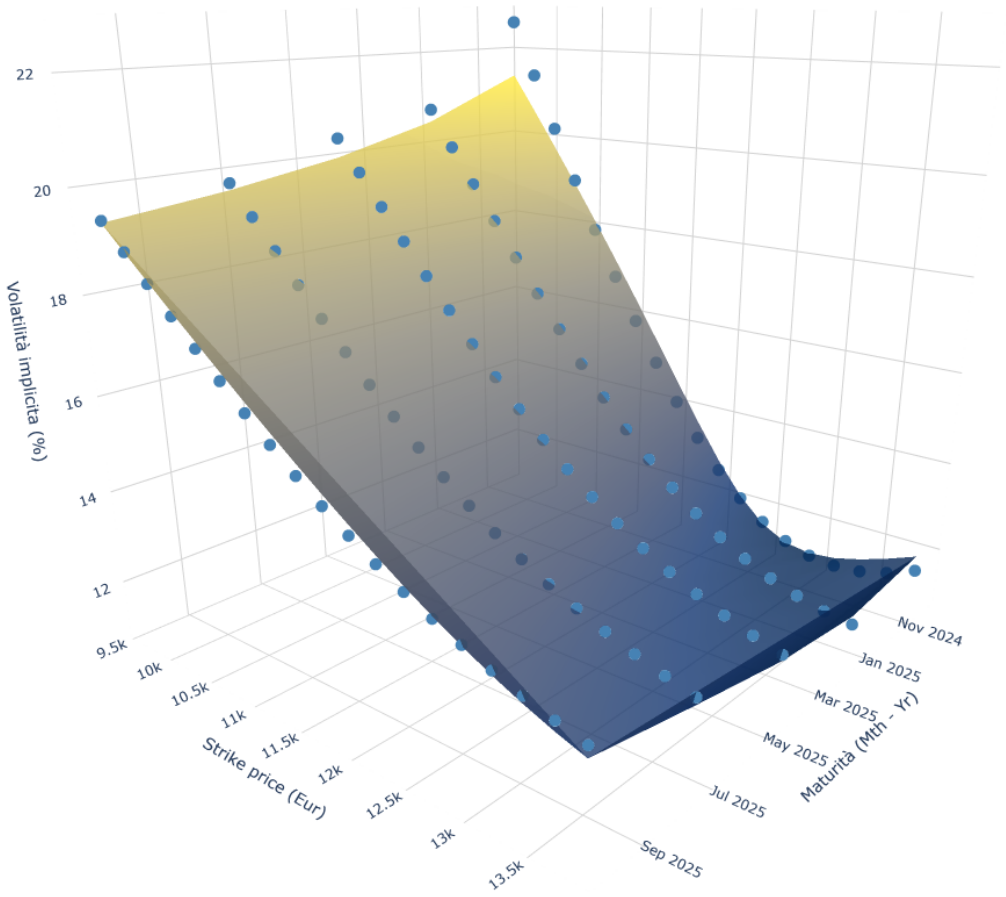

Figura 6.7 La superficie di volatilità del modello di Heston calibrata a partire dai dati di mercato delle opzioni sull'indice SMI in data 3 Giugno 2024 (rappresentati dai marker) rimuovendo le informazioni relative alle opzioni con scadenza il 4 Luglio 2024. Si nota che in questo caso l'aderenza ai dati di mercato risulta essere più uniforme.

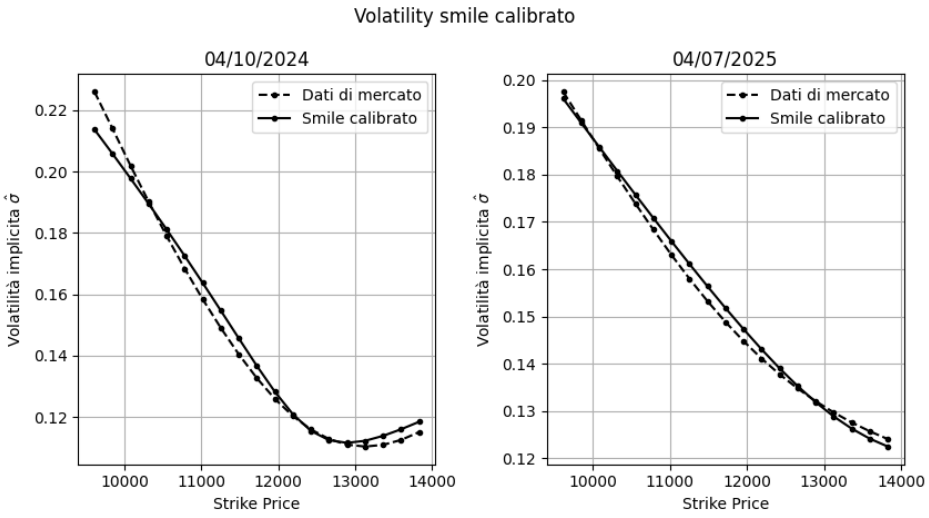

Figura 6.8 Confronto tra alcuni volatility smile ottenuti dal processo di calibrazione sui dati osservati sul mercato privati delle informazioni relative alle opzioni a breve scadenza.

e vanno a descrivere un comportamento molto più estremo e imprevedibile del mercato rispetto al caso precedente: la volatilità del processo varianza e coefficiente di ritorno alla media κ assumono valori molto più elevati, mentre la distanza tra il valore spot v_0 e quello asintotico θ della varianza è inferiore rispetto al caso precedente, andando così a denotare un comportamento del processo varianza complessivamente meno direzionale e più soggetto a bruschi cambiamenti. Ciò sembra suggerire che, nonostante non si riuscisse a riprodurne le caratteristiche in termini di volatilità implicita in maniera fedele, i dati di mercato relativi alle opzioni a breve scadenza permettono di ricavare molte informazioni relative alle dinamiche del sottostante e dovrebbero essere utilizzati in un contesto di applicazione al mondo reale. Un'ulteriore prova di questo fatto si trova andando a confrontare i grafici di destra delle figure 6.6 e 6.8: sebbene la seconda provenga da una superficie che approssima in maniera più fedele i dati di mercato globalmente, lo smile relativo alla data 4 Luglio 2025 viene catturato in maniera più fedele dalla prima.

In conclusione, questo esperimento ha validato in maniera netta il corretto funzionamento delle logiche della classe `LewisFFTHestonParametersCalibrator`, dimostrando la sua capacità di estrapolare adeguatamente i parametri del modello di Heston a partire da dei dati di mercato del mondo reale.

6.3.2 Stima del fair value tramite simulazione Monte Carlo

Come anticipato in precedenza, siamo ora interessati a verificare la consistenza delle stime del prezzo di un derivato ottenute tramite simulazione Monte Carlo con i valori "teorici"

prescritti dalla calibrazione. In parole più semplici, vogliamo confermare che il prezzo ottenuto tramite simulazione Monte Carlo delle traiettorie del processo di Heston sia coerente con quello stabilito dalla formula di Lewis (che è la procedura attraverso la quale andiamo ad effettuare la calibrazione). Se i due valori sono abbastanza vicini, allora le funzionalità implementate relative al modello di Heston costituiscono un insieme coerente e consistente che permette di estrapolare informazioni dal mercato per poi prezzare in maniera adeguata i derivati.

Per verificare l'assenza di notevoli differenze tra i valori dei fair price prescritti dalla superficie di volatilità calibrata e quelli ottenuti tramite simulazione Monte Carlo, si è ideato il seguente esperimento: fisseremo dei valori per i parametri del modello di Heston e considereremo un'opzione call vanilla di cui calcoleremo il fair price sia usando la formula di Lewis implementata nella classe `LewisFFTHestonParametersCalibrator` (che fungerà da valore teorico) sia tramite simulazione Monte Carlo, andando a generare $2^{21} \approx 2,000,000$ traiettorie del processo di Heston. L'esperimento sarà considerato un successo se la distanza tra i due valori sia in termini assoluti che in termini relativi sarà sufficientemente piccola. Visti i risultati dell'esperimento precedente, che hanno evidenziato una difficoltà del modello di Heston nel descrivere in maniera fedele il prezzo di opzioni con time-to-maturity breve, si è scelto di considerare un'opzione tale che

$$T - t_0 = 1.6083 \text{ anni},$$

dove t_0 è la data di riferimento per l'esperimento (fissata al 31 Maggio 2024) e T è la maturità dell'opzione (fissata al 31 Dicembre 2025). Il prezzo spot del sottostante è stato fissato al valore $S(t_0) = 191.09$, mentre il prezzo strike dell'opzione è stato scelto pari a $K = 195.00$. I valori dei parametri del modello di Heston sono stati fissati nel modo seguente:

$$\begin{cases} v_0 = 0.03 \\ \theta = 0.04 \\ \kappa = 2.00 \\ \xi = 0.20 \\ \rho = -0.30 \end{cases},$$

mentre si è supposta una struttura per scadenze piatta con tasso d'interesse privo di rischio pari a $r = 3\%$.

Come per gli esperimenti di Sezione 6.2, si è ripetuta la prova usando i diversi algoritmi di generazione di numeri casuali implementati nella libreria, così da confrontarne le performance sia in termini di precisione delle stime ottenute sia in termini di tempo di esecuzione del

Tabella 6.8 Le stime del fair price di un'opzione call vanilla nel modello di Heston ottenute mediante simulazione Monte Carlo, usando vari metodi di generazione di valori casuali. Gli errori sono calcolati rispetto al valore teorico $C(t_0) = 20.3719$.

	Metodo di Generazione	NPV stimato	Errore Assoluto	Errore Relativo
PRNG	PCG64	20.4301	0.0582	0.285%
	PHILOX	20.3169	0.0550	0.271%
	PCG64DXSM	***20.3467***	***0.0252***	***0.124%***
QRNG	HALTON	20.3514	0.0205	0.100%
	SOBOL	***20.3712***	***0.0007***	***0.003%***

programma. Il prezzo teorico ottenuto dalla formula di Lewis per l'opzione considerata risulta essere

$$C(t_0) = 20.3719.$$

In Tabella 6.8, sono riportate le stime ottenute mediante simulazione Monte Carlo del fair price dell'opzione vanilla considerata. In generale, i risultati ottenuti sono più che soddisfacenti, dato che con tutti i metodi di generazione si riesce a ottenere un errore assoluto pari o inferiore all'ordine di 10^{-2} ed errori relativi al di sotto dell'1%. Tra di essi, il metodo di generazione basato sulle sequenze di Sobol si conferma essere la scelta migliore con un errore assoluto $7 \cdot 10^{-4}$ che garantisce la correttezza delle prime tre cifre decimali. Tra i generatori pseudo-casuali, quello con le performance migliori è stato l'algoritmo PCG64DXSM che ha staccato leggermente gli altri due metodi a disposizione in termini di precisione assoluta. Il risultato complessivamente peggiore è quello ottenuto dal generatore basato sulle sequenze di Halton, i cui risultati sono assai più vicini a quelli dei PRNG rispetto a quelli dell'altro QRNG a disposizione.

Il quadro per questo metodo di generazione peggiora ulteriormente se si considerano i tempi di esecuzione. Infatti, impiegando il generatore HALTON, il codice ha impiegato 170 secondi per essere eseguito, verso i circa 130 secondi richiesti utilizzando gli algoritmi SOBOL e PHILOX (che nello specifico hanno impiegato 135s e 128s rispettivamente) e i 100 secondi circa ottenuti nelle prove con PCG64 (104 s) e PCG64DXSM (108s).

Al di là delle specifiche performance dei vari metodi di generazione, l'esperimento valida l'implementazione del processo di Heston all'interno della libreria e dimostra la capacità delle funzionalità aggiunte di produrre stime del fair price di derivati precise e consistenti con i dati di calibrazione.

6.3.3 Commento dei risultati ottenuti

Quest'ultima coppia di esperimenti ha prodotto risultati estremamente positivi che dimostrano definitivamente il corretto funzionamento di tutte le funzionalità aggiunte alla libreria. Si è potuto osservare come esse vadano a interagire in maniera corretta tra di loro e si integrino alla perfezione all'interno dell'architettura preesistente in SFMQuantLib. Ciò permette agli utilizzatori della libreria di considerare una nuova tipologia di pricing dei derivati, basato su un modello più realistico e meglio capace di descrivere la realtà del tradizionale modello di Black-Scholes, in maniera agevole e senza dover modificare in alcun modo il proprio workflow.

La più grande debolezza evidenziata dagli esperimenti condotti è il costo computazionale legato a tale strategia di pricing. Nonostante le ottimizzazioni consentano di gestire agevolmente simulazioni con un gran numero di scenari e di calibrare il modello a una precisione più che soddisfacente, tali operazioni restano comunque assai onerose in termini di risorse impiegate e tempo di calcolo. Ciò rappresenta senz'altro uno spunto da cui partire per futuri interventi di ottimizzazione delle performance, tuttavia un ruolo fondamentale in questo ambito è giocato dalle caratteristiche del linguaggio Python. Una possibile soluzione consisterebbe nel tradurre parte del codice implementato in una sintassi compatibile con la libreria *Cython* che consente di sfruttare variabili e strutture dati del linguaggio C direttamente all'interno di un programma Python, con un conseguente aumento della performance. Tuttavia un simile intervento richiederebbe una grande operazione di refactoring del codice che deve essere valutata in maniera estremamente attenta.

6.4 Pricing di un certificato d'investimento

Come ultimo esperimento desideriamo mostrare le capacità delle funzionalità implementate in SFMQuantLib di prezzare titoli derivati anche complessi in un contesto di applicazione al mondo reale. Per far ciò proveremo a calcolare il fair price di un certificato d'investimento e lo andremo a confrontare con la sua quotazione di mercato. Come banco di prova si è scelto un certificato emesso dall'azienda finanziaria svizzera Leonteq. Nello specifico, un certificato con payoff di tipo bonus con cap, scritto sul valore dell'indice EUROSTOXX50. Tale strumento è caratterizzato da un valore nominale di 1000 € e prevede il rimborso al 17 marzo 2025 dello stesso più un bonus del 10% qualora il valore del sottostante non scenda mai al di sotto di una barriera, fissata a un valore pari al 70% del valore dell'indice al 9 marzo 2022, e il suo rendimento alla maturità non sia superiore al 10%. Se il rendimento del sottostante alla maturità risultasse essere superiore a tale valore, verrà restituita una somma pari al valore nominale

maggiorata di una quantità pari a essa ma mai superiore a 1400 €. Qualora invece il valore dell'indice scendesse al di sotto del livello barriera, alla maturità verrà restituita una quantità di denaro pari al valore nominale sommato al valore nominale moltiplicato per il rendimento del sottostante ma, anche in questo caso, mai superiore a 1400 €.

Utilizzando la classe `StandardCertificate` è possibile descrivere tale strumento. Per calcolarne il prezzo, è stato effettuato il bootstrapping della curva relativa al tasso d'interesse ESTR in data 8 agosto 2024 e calibrata la superficie di volatilità delle opzioni call vanilla sull'indice EUROSTOXX50 alla stessa data andando a stimare i seguenti valori per i parametri del modello di Heston:

$$\begin{cases} v_0 = 0.0731 \\ \theta = 0.0013 \\ \kappa = 6.6781 \\ \xi = 0.4800 \\ \rho = -0.5123 \end{cases}.$$

Complessivamente la procedura di calibrazione ha richiesto circa 220 secondi per essere ultimata. I grafici delle superfici di volatilità ottenute sono riportati in Appendice C. Dopodiché è stata avviata la procedura per il calcolo del fair price dello strumento. Si è scelto di calcolare tale valore tramite una simulazione Monte Carlo basata su $N = 2^{21}$ scenari e utilizzando come metodo di generazione dei valori casuali l'algoritmo basato sulle sequenze di Sobol, che, negli esperimenti precedenti, si è rivelato essere il più performante tra quelli a disposizione nella libreria. La simulazione ha impiegato 136 secondi per essere portata a termine, producendo come stima del fair price del certificato il valore:

$$\hat{P}_{\text{MC}} = 1234.31€.$$

Consultando gli adeguati portali è possibile verificare che il prezzo di mercato a fine giornata di tale certificato era pari a:

$$P_{\text{Market}}(16:00) = 1230.40€, \qquad P_{\text{Market}}(20:00) = 1236.20€,$$

e, considerando il loro valor medio $\bar{P}_{\text{Market}} = 1233.30€$, si ottiene un errore di stima pari ad appena 1.01€.

Un simile livello di precisione è più che soddisfacente, soprattutto vista la scarsa liquidità di simili strumenti e il fatto che non si è tenuto conto degli effetti di eventuali dividendi sull'andamento del sottostante né in fase di calibrazione né durante la simulazione delle

traiettorie. Tuttavia, tale accuratezza non è del tutto sorprendente, vista la durata del processo di calibrazione, che ha senz'altro permesso di stimare in maniera sufficientemente precisa i valori assunti dai parametri del modello di Heston, e le ottime caratteristiche di liquidità delle opzioni sul sottostante EUROSTOXX50, requisito fondamentale per una calibrazione agevole e precisa. Inoltre, l'impiego dell'algoritmo di generazione basato sulle sequenze di Sobol, che si è mostrato essere capace di produrre risultati di ottima qualità, e il gran numero di scenari considerati hanno permesso di minimizzare il più possibile l'errore di stima. Infine, è importante sottolineare che, alla data in cui è stato effettuato il pricing, il valore del sottostante si trovava assai al di sopra del livello barriera, quindi gli eventuali effetti indesiderati legati alla sensibilità della stessa hanno avuto un impatto minore sulla precisione della stima, rispetto a quanto visto nell'esperimento di Sezione 6.3.

Ancora una volta si è evidenziata la generale lentezza di questo approccio per il calcolo del prezzo di derivati. Infatti, per ottenere la stima $\hat{P}_{\text{MC}}$ è stato necessario attendere quasi 6 minuti dall'avvio del programma. Questo è un fattore critico per l'usabilità degli strumenti di pricing implementati, in quanto in contesti come la definizione di strategie di hedging è fondamentale poter aggiornare le stime del fair price il più velocemente possibile. Parte di questa criticità è caratteristica dei metodi Monte Carlo che, per via del loro ordine di convergenza, richiedono l'impiego di un gran numero di risorse computazionali, tuttavia è possibile investigare quale sia il trade-off tra sforzo computazionale e errore di stima in un contesto di applicazione delle funzionalità implementate a un problema del mondo reale.

Nello specifico, vorremmo investigare gli effetti sulla qualità della stima di due parametri strettamente legati al costo computazionale dell'operazione di pricing:

- il *numero di iterazioni I* dell'algoritmo di ottimizzazione interno al calibratore, ovvero il parametro che sancisce quanti step può eseguire al massimo l'ottimizzatore prima di restituire la soluzione ottima trovata;
- il *numero di traiettorie N* generate durante la simulazione Monte Carlo.

Ridurre il valore assunto da questi parametri implica ridurre drasticamente il numero di operazioni necessarie per ultimare il processo di pricing e, conseguentemente, il tempo necessario alla sua esecuzione.

Per studiare gli effetti di tali riduzioni sulla precisione della stima ottenuta si è deciso di ripetere l'esperimento di pricing precedente (mantenendo come generatore di numeri casuali quello basato sulle sequenze di Sobol) andando a tenere fisso il numero delle traiettorie al valore $N = 2^{17}$ e andando a far variare il valore del parametro I tra 50, 100, e 150, ottenendo

Tabella 6.9 L'effetto sulla stima di diversi valori per il parametro I relativo al numero di iterazioni dell'algoritmo di calibrazione. Si osserva come questo abbia un profondo effetto sulla precisione della stima ottenuta.

N	I	NPV	$T_{\text{calibrazione}}$	$T_{\text{simulazione}}$	T_{totale}
2^{17}	50	1243.23	47.15	86.22	133.37
2^{17}	100	1233.71	106.76	86.91	193.67
2^{17}	150	1234.80	194.32	160.48	354.80

Tabella 6.10 L'effetto sulla stima di diversi valori per il parametro N relativo al numero delle traiettorie. Si osserva come in questo caso l'effetto sulla stima sia assai inferiore.

N	I	NPV	$T_{\text{calibrazione}}$	$T_{\text{simulazione}}$	T_{totale}
2^{13}	100	1233.13	96.74	74.58	171.32
2^{16}	100	1233.64	108.03	144.19	252.22
2^{19}	100	1233.80	87.42	136.41	223.83

i risultati di Tabella 6.9. Una rapida analisi dei valori lì riportati rivela immediatamente un sensibile impatto positivo sulla precisione della stima ottenuta all'aumentare della complessità del programma. Un quadro diverso è invece quello descritto dai risultati della Tabella 6.10, nel quale si è tenuto fisso il numero di iterazioni del calibratore e si è fatto variare il numero delle traiettorie generate N tra valori sufficientemente distanti tra di loro, producendo, tuttavia, un effetto relativamente trascurabile in termini di precisione della stima che rimane sempre sufficientemente vicina al valore ottenuto all'inizio di questa sezione. Ciò valida ulteriormente la qualità dell'algoritmo di generazione impiegato e dello schema di discretizzazione del processo di Heston implementato nella libreria, che consentono di ottenere stime molto precise anche per valori relativamente bassi di N. L'esperimento evidenzia un comportamento inatteso del tempo di simulazione: la variabilità inattesa di tale valore è legata al fatto che per ottimizzare la velocità della simulazione si è fatto ricorso a tecniche di multiprocessing e la generazione (spawn) di nuovi processi a partire da un singolo programma è un processo abbastanza costoso che deve essere gestito dal sistema operativo; quindi il tempo impiegato a eseguire tale operazione può variare in maniera anche molto significativa sulla base delle attività in background svolte dal PC.

Questo esperimento ha rivelato come l'impatto del parametro di calibrazione I sulla qualità della stima sia molto superiore rispetto al numero delle traiettorie N. Ciò ci suggerisce di dare priorità all'aspetto di calibrazione al fine di ottenere stime il più possibile precise in un tempo limitato, anche a costo di effettuare piccoli sacrifici in termini di numero degli scenari della simulazione. Al tempo stesso, tale risultato suggerisce la necessità di verificare le possibilità di

ottimizzare maggiormente il calibratore dei parametri del modello di Heston, in quanto esso costituisce, allo stato attuale, un significativo collo di bottiglia (bottleneck) relativamente alle performance dell'architettura di pricing della libreria. Ciò, insieme all'implementazione delle funzionalità necessarie alla stima, gestione e utilizzo nella procedura di pricing dei dividendi, costituiscono i principali spunti per la prosecuzione dello sviluppo delle funzionalità relative alla stima Monte Carlo del fair price di derivati in SFMQuantLib.

Bibliografia

Andersen, L. (2007). Efficient simulation of the heston stochastic volatility model. *J. Computat. Finance*, 11.

Baldi, P. (2017). *Stochastic Calculus: An Introduction Through Theory and Exercises*. Universitext. Springer International Publishing.

Bellelli, G. (2021). *Investire con i certificati: Selezionare, costruire e gestire un portafoglio con un rischio contenuto*. Hoepli.

Björk, T. (2005). *Arbitrage theory in continuous time*. Oxford Univ. Press, Oxford [u.a.], 2. ed., reprint. edition.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654.

Bosio, M. (2023). Sviluppo di una libreria di pricing e di sensitivity analysis per opzioni vanilla europee. Tesi secretata. Fulltext non presente.

Brandimarte, P. (2014). *Handbook in Monte Carlo Simulation: Applications in Financial Engineering, Risk Management, and Economics*. Wiley Handbooks in Financial Engineering and Econometrics. Wiley.

Brandimarte, P. (2017). *An Introduction to Financial Markets: A Quantitative Approach*. Wiley.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education.

Gil-Pelaez, J. (1951). Note on the inversion theorem. *Biometrika*, 38(3-4):481–482.

Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Springer, New York, NY, USA.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2):327–343.

Hull, J. C. (1993). *Options, futures, and other derivative securities*. Prentice Hall, 2 edition.

Lewis, A. (2002). A simple option formula for general jump-diffusion and other exponential levy processes. *SSRN Electronic Journal*.

Merton, R. (1973). The theory of rational option pricing. *Bell J Econ Manage Sci*, 4:141–183.

Nelsen, R. B. (2006). *An Introduction to Copulas*. Springer, New York, NY, USA, second edition.

Pascucci, A. (2008). *Calcolo stocastico per la finanza*. UNITEXT. Springer Milan.

Platen, E. and Heath, D. (2006). *A Benchmark Approach to Quantitative Finance*. Springer.

Schoutens, W., Simons, E., and Tistaert, J. (2004). A perfect calibration! now what? *Wilmott*, 2004.

Zastawniak, T. and Capiński, M. (2011). *Mathematics for Finance. An Introduction to Financial Engineering*. Springer, Germany, 2nd edition.

Appendice A

Copule

La teoria delle copule rappresenta un importante strumento per modellare la dipendenza tra variabili aleatorie multidimensionali, separando la struttura di dipendenza dalle distribuzioni marginali delle singole variabili. Questo concetto ha numerose applicazioni, specialmente nei campi della finanza, delle scienze attuariali, e delle scienze naturali, dove è fondamentale comprendere le relazioni tra variabili.

A.1 Definizione di copula e proprietà fondamentali

Una *copula* è una funzione che collega la distribuzione congiunta di variabili aleatorie alle loro distribuzioni marginali. Formalmente, è una funzione $C : [0,1]^d \to [0,1]$ che soddisfa le seguenti proprietà.

Marginalità: Per ogni $i \in \{1,\dots,d\}$ e per ogni $u_j \in [0,1]$ con $j \neq i$,

$$C(1,1,\dots,1,u_i,1,\dots,1) = u_i.$$

Questa proprietà garantisce che la copula preservi le distribuzioni marginali uniformi su $[0,1]$.

Groundedness: La funzione C è *grounded*, ovvero assume il valore zero quando uno degli argomenti è pari a zero. In altre parole, per ogni $i \in \{1,\dots,d\}$ e per ogni $u_j \in [0,1]$ con $j \neq i$,

$$C(u_1,\dots,u_{i-1},0,u_{i+1},\dots,u_d) = 0.$$

Questa proprietà garantisce che la copula sia zero quando una delle variabili è al suo valore minimo.

Condizione di d-incremento: Per ogni rettangolo $[a_1,b_1]\times\cdots\times[a_d,b_d]\subseteq[0,1]^d$, dove $0\le a_i\le b_i\le 1$, si ha

$$\sum_{\mathbf{v}\in\{a,b\}^d}(-1)^{N(\mathbf{v})}C(\mathbf{v})\ge 0,$$

dove $N(\mathbf{v})$ è il numero di componenti di $\mathbf{v}$ uguali a b. Questa condizione assicura che C sia una funzione di distribuzione cumulativa valida e non decrescente in ciascuna delle sue variabili.

A.2 Principali risultati teorici

Viste le numerose proprietà che una funzione deve avere per essere considerata una copula, non è sorprendente l'esistenza di dei limiti estremi per una copula, che descrivono i casi di massima dipendenza positiva e negativa tra le variabili. Questi limiti sono detti *bound di Fréchet-Hoeffding* a risultano essere fondamentali per comprendere il comportamento delle copule e la natura delle dipendenze che possono descrivere:

1. **Limite superiore di Fréchet-Hoeffding**: $C^+(u_1,\ldots,u_d)=\min(u_1,\ldots,u_d)$, che rappresenta il caso di dipendenza perfetta positiva.
2. **Limite inferiore di Fréchet-Hoeffding**: $C^-(u_1,\ldots,u_d)=\max\{0,u_1+\cdots+u_d-(d-1)\}$, che rappresenta il caso di massima dipendenza negativa.

Il ruolo delle copule come strumento matematico per modellare la dipendenza e la correlazione tra le marginali di un vettore aleatorio è giustificato dal seguente risultato.

Teorema A.1 (di Sklar)**.** *Sia $F(x_1,\ldots,x_d)$ una funzione di distribuzione congiunta di una variabile casuale $\mathbf{X}=(X_1,\ldots,X_d)$, con distribuzioni marginali $F_1(x_1),\ldots,F_d(x_d)$. Allora esiste una copula $C:[0,1]^d\to[0,1]$ tale che, per ogni $(x_1,\ldots,x_d)\in\mathbb{R}^d$,*

$$F(x_1,\ldots,x_d)=C(F_1(x_1),\ldots,F_d(x_d)).$$

Inoltre, se le distribuzioni marginali $F_1,\ldots,F_d$ sono continue, la copula C è unica. Se invece non lo sono, la copula C è unica su $Im(F_1)\times\cdots\times Im(F_d)$.

Il teorema di Sklar garantisce che, data una qualsiasi distribuzione multivariata, esiste un'unica copula che descriva la dipendenza tra le varie marginali. Ciò permette a tutti gli effetti

di decomporre tale distribuzione in termini di una singola copula (che descrive la dipendenza tra le varie componenti del vettore aleatorio) e le distribuzioni marginali.

A.3 Trasformazione di Rosenblatt

La *trasformazione di Rosenblatt*, è uno strumento utile per convertire variabili aleatorie congiunte, descritte da una copula, in variabili uniformi e indipendenti su $[0,1]$. Dato un insieme di variabili aleatorie uniformi $\mathbf{U} = (U_1, \ldots, U_d)^\intercal$ associate a una copula C, la trasformazione di Rosenblatt $\mathbf{V} = R_C(\mathbf{U})$ è definita in modo ricorsivo:

$$
\begin{aligned}
V_1 &= U_1, \\
V_2 &= C_2^{-1}(U_2 \mid U_1), \\
V_3 &= C_3^{-1}(U_3 \mid U_1, U_2), \\
&\vdots \\
V_d &= C_d^{-1}(U_d \mid U_1, U_2, \ldots, U_{d-1}).
\end{aligned}
$$

La *trasformazione inversa di Rosenblatt* è il processo inverso, che consente di trasformare variabili uniformi indipendenti $\mathbf{V} = (V_1, \ldots, V_d)^\intercal$ in variabili aleatorie uniformi con una determinata struttura di dipendenza. Questo processo è utile quando si vuole generare una distribuzione congiunta con dipendenza predefinita a partire da variabili indipendenti. Data una realizzazione $\mathbf{v}$ del vettore aleatorio $\mathbf{V}$, la funzione inversa segue la costruzione ricorsiva:

$$
\begin{aligned}
u_1 &= v_1, \\
u_2 &= C_2(v_2 \mid u_1), \\
u_3 &= C_3(v_3 \mid u_1, u_2), \\
&\vdots \\
u_d &= C_d(v_d \mid u_1, u_2, \ldots, u_{d-1}).
\end{aligned}
$$

La trasformazione di Rosenblatt inversa è la funzione R_C^{-1} usata all'interno della procedura di simulazione descritta in Sezione 2.2.

A.4 Copule implementate in SFMQuantLib

Di seguito vengono riportati i grafici relativi alle copule implementate nella libreria quantitativa SFMQuantLib.

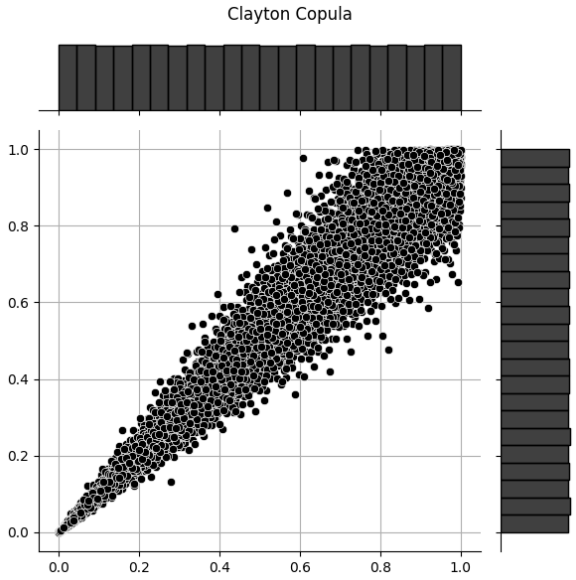

Figura A.1 Il campionamento di 10000 punti provenienti da una copula di Clayton realizzato con le funzionalità implementate in *SFMQuantLib*. Il parametro di correlazione è stato impostato sul valore $\tau = 88\%$.

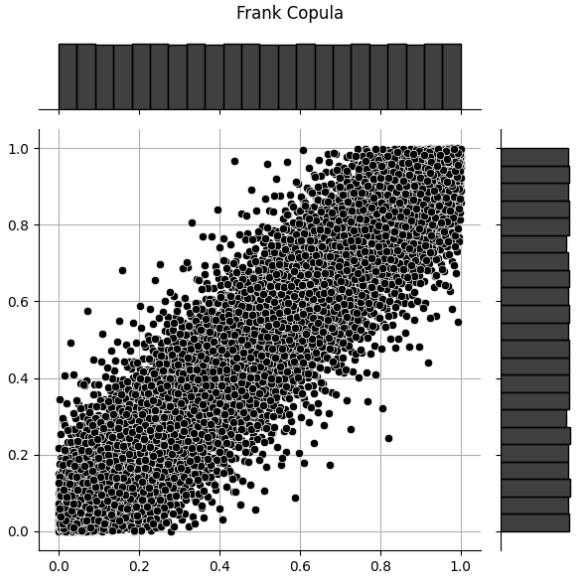

Figura A.2 Il campionamento di 10000 punti provenienti da una copula di Frank realizzato con le funzionalità implementate in *SFMQuantLib*. Il parametro di correlazione è stato impostato sul valore $\tau = 76\%$.

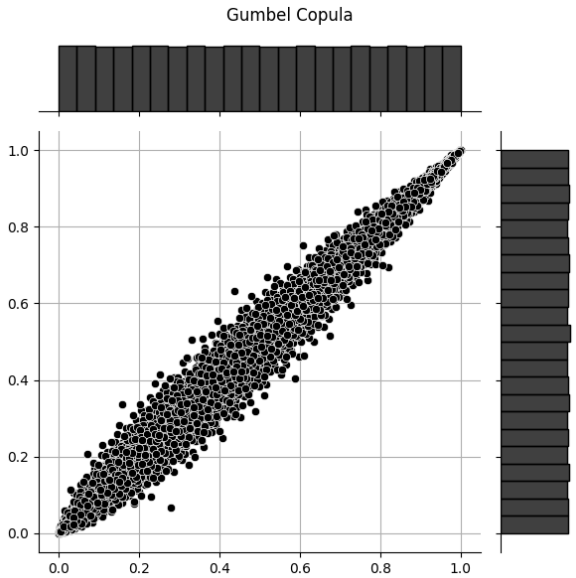

Figura A.3 Il campionamento di 10000 punti provenienti da una copula di Gumbel realizzato con le funzionalità implementate in *SFMQuantLib*. Il parametro di correlazione è stato impostato sul valore $\tau = 93\%$.

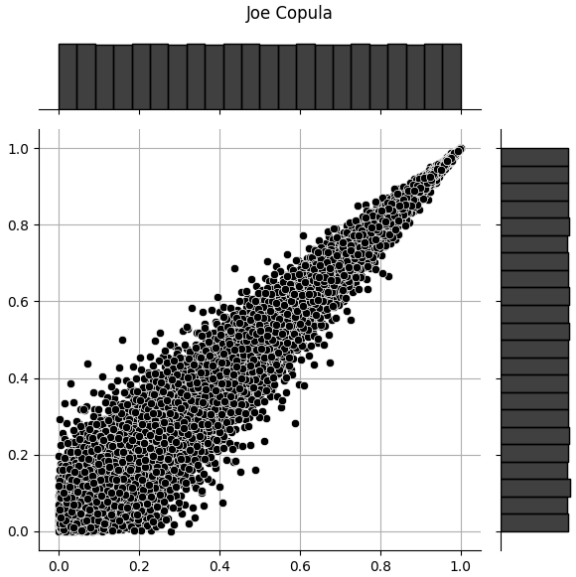

Figura A.4 Il campionamento di 10000 punti provenienti da una copula di Joe realizzato con le funzionalità implementate in *SFMQuantLib*. Il parametro di correlazione è stato impostato sul valore $\tau = 88\%$.

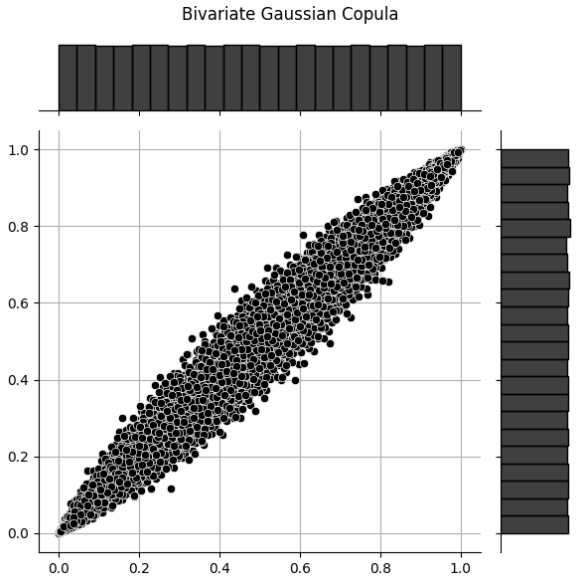

Figura A.5 Il campionamento di 10000 punti provenienti da una copula Gaussiana realizzato con le funzionalità implementate in *SFMQuantLib*. Il parametro di correlazione è stato impostato sul valore $\rho = 99\%$.

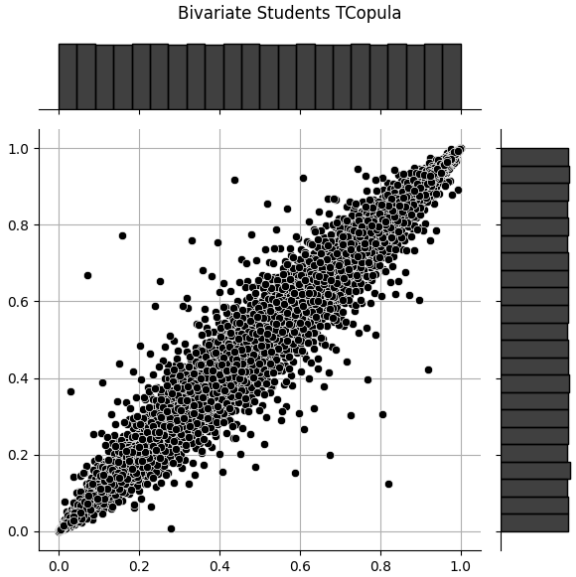

Figura A.6 Il campionamento di 10000 punti provenienti da una copula T di Student realizzato con le funzionalità implementate in *SFMQuantLib*. Il parametro di correlazione è stato impostato sul valore $\tau = 99\%$ e i gradi di libertà pari a 2.

Appendice B

Pricing usando FFT nel modello di Heston

In questa appendice, approfondiremo velocemente come è possibile sfruttare l'algoritmo FFT (Trasformata di Fourier Veloce) per ridurre il costo computazionale legato al calcolo del prezzo delle opzioni call vanilla nel modello di Heston usando la formula di Lewis

$$C_0 = S_0 - \frac{\sqrt{S_0 K} e^{-rT}}{\pi} \int_0^\infty Re\left[e^{iuk} \phi_T\left(u - \frac{i}{2}\right) \frac{1}{u^2 + \frac{1}{4}} \right] du.$$

Ricordiamo che la *parte reale* di un numero complesso è lineare, ovvero per $a, b \in \mathbb{R}$ abbiamo:

$$Re[az_1 + bz_2] = aRe[z_1] + bRe[z_2],$$

e la parte reale di un integrale è l'integrale della parte reale. Grazie a questa proprietà, la formula di prezzo integrale può essere scritta nella forma seguente:

$$C_0 = S_0 - \frac{\sqrt{S_0 K} e^{-rT}}{\pi} Re\left[\int_0^\infty e^{iuk} \phi_T\left(u - \frac{i}{2}\right) \frac{1}{u^2 + \frac{1}{4}} du \right].$$

A questo punto possiamo discretizzare l'integrale. Utilizziamo la regola di Simpson. Il dominio di integrazione è troncato in $[A, B] = [0, B]$, e diviso in N passi di dimensione $\Delta x = \frac{B}{N}$. Abbiamo che $x_0 = 0$ e $x_n = x_0 + n\Delta x$, per $n = 0, 1, 2, \ldots, N$. L'integrale viene valutato in $N + 1$ punti $f(x_n) = f_n$. La regola di Simpson è una regola a 3 punti che approssima l'integrale come:

$$\int_{x_0}^{x_2} f(x)\, dx \approx \frac{\Delta x}{3} \left[f_0 + 4 f_1 + f_2 \right].$$

Se sommiamo su tutto il dominio di integrazione $[x_0, x_{N-1}]$, otteniamo:

$$\int_{x_0}^{x_{N-1}} f(x)\,dx \approx \frac{\Delta x}{3} \sum_{n=0}^{N-1} w_n f_n,$$

con $w_n = 1$ per $n = 0$ e $n = N-1$, $w_n = 4$ per n dispari, e $w_n = 2$ per n pari. Nota che non stiamo considerando l'ultimo punto!

Definiamo un insieme di N valori $k_j \in [-b, b]$, tali che $k_j = -b + j\Delta k$, per $j = 0, 1, 2, \ldots, N-1$. Definiamo il passo Δk come:

$$\Delta k := \frac{2\pi}{B} = \frac{1}{\Delta x}\frac{2\pi}{N},$$

in modo che possiamo ottenere il valore $b = \frac{N\Delta k}{2}$. L'integrale nella funzione di pricing è:

$$\begin{aligned}
I(k_k) &= \int_0^{\infty} e^{ixk_j} \phi_T\left(x - \frac{i}{2}\right) \frac{1}{x^2 + \frac{1}{4}}\,dx \\
&\approx \frac{\Delta x}{3} \sum_{n=0}^{N-1} w_n e^{ik_j x_n} \phi_T\left(x_n - \frac{i}{2}\right) \frac{1}{{x_n}^2 + \frac{1}{4}} \\
&= \frac{\Delta x}{3} \sum_{n=0}^{N-1} w_n e^{i(-b+j\Delta k)n\Delta x} \phi_T\left(x_n - \frac{i}{2}\right) \frac{1}{{x_n}^2 + \frac{1}{4}} \\
&= \frac{\Delta x}{3} \sum_{n=0}^{N-1} e^{i2\pi j \frac{n}{N}}\, w_n e^{-ibn\Delta x} \phi_T\left(x_n - \frac{i}{2}\right) \frac{1}{{x_n}^2 + \frac{1}{4}}.
\end{aligned}$$

La funzione fft di scipy restituisce quanto segue:

```
y(j) = (x * exp(-2*pi*sqrt(-1) * j * np.arange(n) / n)).sum()
```

La funzione ifft di scipy restituisce:

```
y(j) = (x * exp(2*pi*sqrt(-1) * j * np.arange(n) / n)).mean()
```

Pertanto possiamo usare `ifft` e poi moltiplicare il risultato per N.

Appendice C

Superfici di volatilità dell'indice EUROSTOX50

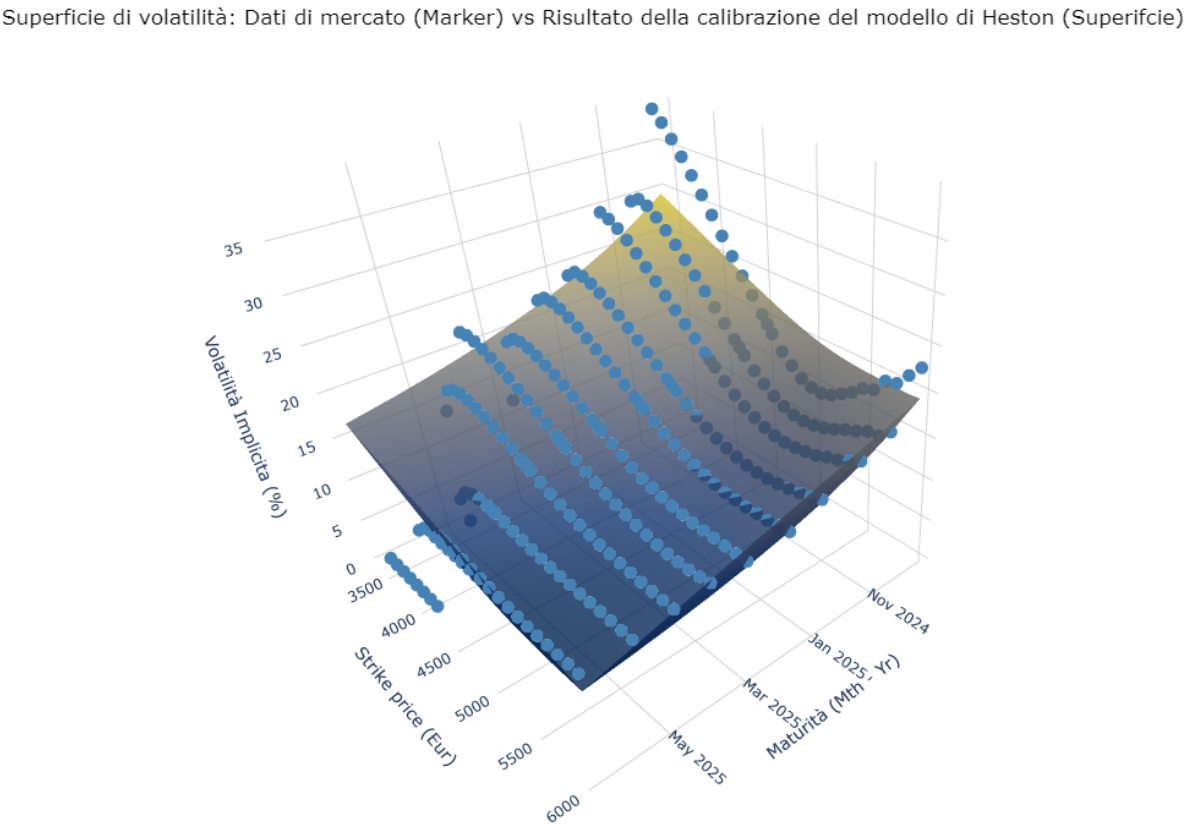

Figura C.1 La superficie di volatilità del modello di Heston calibrata a partire dai dati di mercato delle opzioni sull'indice EUROSTOX50 in data 8 Agosto 2024 (rappresentati dai marker).

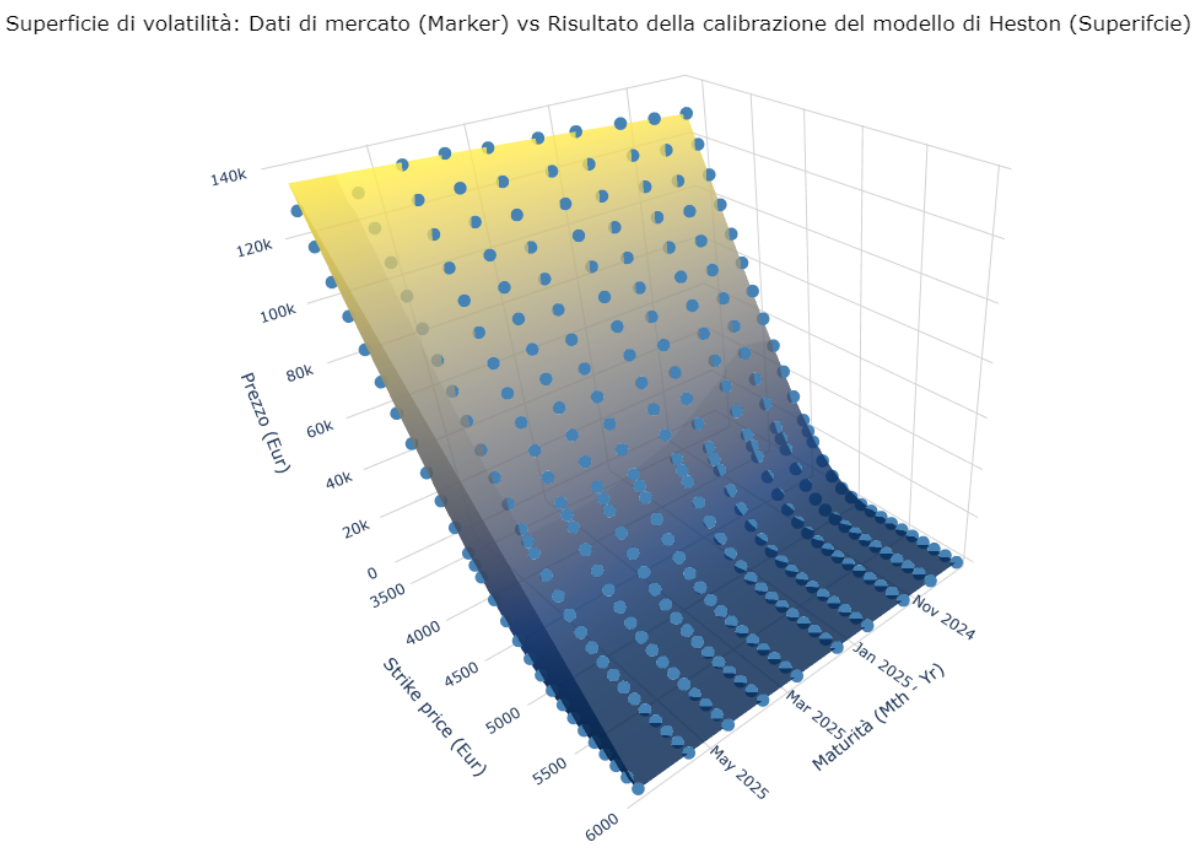

Figura C.2 La superficie dei prezzi secondo il modello di Heston calibrata a partire dai dati di mercato delle opzioni sull'indice EUROSTOX50 in data 8 Agosto 2024 (rappresentati dai marker).