POLITECNICO DI MILANO
FACOLTÀ DI INGEGNERIA DEI SISTEMI
CORSO DI LAUREA IN INGEGNERIA MATEMATICA

CALIBRAZIONE DI MODELLI A VOLATILITÀ STOCASTICA SU ARCHITETTURA PARALLELA

Relatore: Dr. Marazzina DANIELE

Tesi di Laurea di:
Ngjela ARBER
Matricola n. 734863

ANNO ACCADEMICO 2010-2011

Ja dedikoj familjes time edhe Rotary Club di San Donato Milanese D.2050

Ringraziamenti

Ringrazio Dr. Daniele Marazzina per la sua disponibilità e per avermi guidato e criticato nell'approccio ai problemi incontrati nel corso della tesi. Desidero ringraziare il gruppo di ricerca MOX per l'utenza di accesso alla macchina parallela e in particolare Luca Paglieri per il supporto informatico e i consigli nell'affrontare le difficoltà di utilizzo di IDRA.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Nei mercati finanziari vengono introdotti giornalmente derivati con nuove caratteristiche. Il loro prezzamento costituisce un ramo molto interessante della finanza. Il primo passo per un'analisi quantitativa del derivato è analizzare le caratteristiche del sottostante (cioè il tiolo finanziario su cui il derivato è costruito), adottando un modello che ne "descriva" l'evoluzione temporale. Successivamente si può calibrare tale modello con i dati di mercato e utilizzare i parametri stimati per prezzare il nuovo derivato.
Obiettivo della tesi è studiare e implementare su architettura parallela il problema di calibrazione. Saranno considerati dei data set da poter apprezzare i metodi sviluppati e trarre delle conclusioni su essi e sui modelli implementati.
L'opera che segue introduce inizialmente il problema di calibrazione, proseguendo poi nel secondo capitolo con i concetti fondamentali matematici e finanziari. Si parte dal processo di Wiener e si arriva ai modelli a volatilità stocastica. Il capitolo 3 è dedicato alla calibrazione, descrive in profondità il problema affrontato e la soluzione adottata. Il capitolo 4 presenta l'architettura parallela utilizzata insieme al paradigma della programmazione parallela Master-Slave. Nel capitolo 5, con l'ausilio dei data set, si pongono a confronto i metodi sviluppati e i modelli considerati. Vengono convalidati i risultati con una tecnica di validazione. Nell'ultimo capitolo si traggono le conclusioni sui metodi e i modelli valutando tempi di esecuzione e stabilità della soluzione.

1.1 Il problema inverso

Quando si parla di *problema inverso* viene spontaneo chiedersi "inverso rispetto a cosa?". Dati due problemi, uno di essi viene chiamato problema inverso se la sua formulazione coinvolge anche l'altro. Quindi una volta stabilito quale è il *problema diretto*, l'altro sarà il *problema inverso*. Per esempio, in fisica, se è noto lo stato attuale di un corpo e la sua legge allora è possibile predire la posizione futura del corpo. Questo approccio viene generalmente definito come *problema diretto*. Mentre, se si vuol stabilire la posizione attuale del corpo dalle osservazioni future allora si parla di *problema inverso*. Ci possono essere varie motivazioni per trovare una soluzione a questo: voler conoscere i parametri del sistema rappresentante il fenomeno oppure capire la causa che ha portato all'effetto osservato.
Tipicamente questo genere di argomenti fanno parte di una classe più grande di problemi, noti come problemi non ben posti. Ossia quelli che non soddisfano le caratteristiche di un problema ben posto secondo Hadamard:

1. per tutti i dati ammissibili esiste una soluzione
2. per tutti i dati ammissibili la soluzione è unica
3. la soluzione dipende con continuità dai dati

La violazione della seconda condizione è più preoccupante della prima condizione poichè quest'ultima può essere rafforzata rilassando il concetto di soluzione. La violazione della seconda condizione invece, comporta l'esistenza di tante soluzioni e perciò bisogna definire quali soluzioni hanno interesse oppure effettuare un controllo di completezza del modello ed eventualmente aggiungere informazioni. L'ultima condizione afferma che un piccolo errore sui dati provoca un piccolo errore nella soluzione, in altre parole è una condizione sulla stabilità locale della soluzione rispetto ai dati. La sua violazione genera grande preoccupazione quando si passa all'implementazione del modello, causando instabilità numerica. Una sensibilità eccessiva ai dati iniziali che porta ad una soluzione approssimata che potrebbe essere molto lontana dalla soluzione originale. Generalmente non esiste nessun rimedio in matematica che possa permettere il passaggio di un problema

instabile ad uno stabile.
La definizione precedente non è matematicamente precisa, è necessario fornire il concetto di soluzione, l'insieme di dati ammissibili e la topologia utilizzata per misurare la continuità. Per poter passare alla definizione formale si deve definire un operatore lineare. Siano X e Y due spazi normati allora una funzione $T : X \to Y$ è detta un operatore lineare da X a Y se per ogni $x, y \in X$ e per ogni $\lambda \in \mathbb{R}$ si ha che

$$T(\lambda x + y) = \lambda T(x) + T(y). \tag{1.1}$$

Un operatore lineare $T : X \to Y$ è limitato se esiste $M \geq 0$ tale che

$$\|T(x)\|_Y \leq M \|x\|_X \forall x \in X.. \tag{1.2}$$

L'espressione può essere interpretata in questo modo: un operatore lineare limitato porta insiemi limitati in insiemi limitati. Questo concetto è legato strettamente con la continuità per quanto riguarda gli operatori lineari. Infatti, dato un operatore lineare se è limitato allora è anche continuo, e vale anche il viceversa [2]. Quindi si può parlare indifferentemente di operatore lineare limitato o continuo.
Attraverso l'ausilio di queste definizioni è possibile definire un problema ben posto secondo Hadamard. Se si definisce il problema come un'equazione dell'operatore lineare limitato

$$T(x) = y, \tag{1.3}$$

allora la prima condizione di Hadamard equivale a richiedere che ogni $y \in Y$ ammetterà una soluzione. L'unicità della soluzione è ottenibile se e solo se $ker(T) = 0$, equivalente ad asserire che l'operatore è iniettivo; ossia elementi distinti del dominio X finiscono in immagini distinte di Y. Se sono verificate le prime due condizioni allora T^{-1} esiste e l'ultima condizione implica che T^{-1} è limitato (continuo) [1].

1.2 Option pricing e calibrazione

Una questione importante in finanza quantitativa è la calibrazione dei modelli, un problema *inverso* rispetto al problema di *pricing*. Mentre il problema di *pricing* riguarda il calcolo del valore delle opzioni dati i parametri del modello, la calibrazione si interessa della stima di essi. Per entrambi i problemi è indispensabile assumere un modello che possa rappresentare bene l'evoluzione del sottostante. Allora è naturale chiedersi: chi sceglie il modello? La risposta necessita di alcuni concetti finanziari, essenzialmente è il mercato che lo sceglie [3]. Questo rende chiaro anche il motivo per cui sia utile cogliere bene il fenomeno che si vuol modellizzare in modo da poter fornire un prezzo affidabile.
In figura 1.1 è mostrato l'andamento dei prezzi simulati e quelli osservati sul mercato [6]. Innanzitutto si nota subito che si tratta di un fenomeno casuale e di cui

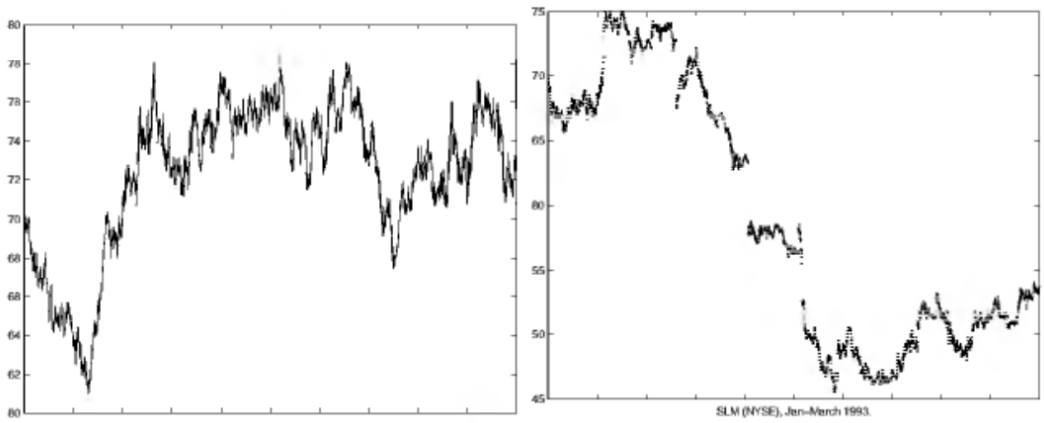

Figura 1.1: Evoluzione dei prezzi: confronto tra uno scenario simulato di MBG a sinistra e i prezzi di SLM(NYSE) a destra, nel periodo Gennaio-Marzo 1993.

è difficile fare una previsione sul valore puntuale successivo. Inoltre il grafico di sinistra ha un'evoluzione continua, invece quello di destra mostra chiaramente dei punti di salto.
Da queste osservazioni nasce l'esigenza di un processo che possa tener conto della casualità e dei salti. Il Moto Browniano Geometrico (MBG) è un processo stoca-

stico che soddisfa solamente la prima richiesta, per poter replicare anche i salti è necessario considerare i processi di *Lévy*. Il modello di Black-Scholes considera che l'evoluzione del sottostante segua un MBG in cui il coefficiente della diffusione sia costante, chiamato volatilità. Invertendo la formula di Black-Scholes, fissando il prezzo al valore di mercato del derivato e ricavando la volatilità si nota che essa non ha un andamento costante, in contraddizione con l'assunzione del modello. Allora si ipotizza che la volatilità possa esser non osservabile nel mercato e che possa evolvere secondo una legge stocastica, tipicamente un'equazione differenziale stocastica (EDS). Si passa così ai modelli di Heston e Bates, in cui il sottostante evolve seguendo due leggi stocastiche; un'EDS per l'andamento del prezzo in cui la volatilità dipende a sua volta da un'EDS.

Il modello considerato deve tener conto di un'ipotesi ragionevole sul mercato come il principio di non arbitraggio, ossia che non si possono creare i soldi dal nulla senza assumersi un rischio. Il principio stabilisce il termine di drift del processo stocastico che rappresenta i prezzi. Si dice che il modello considerato evolve sotto la misura $\mathbb{Q}$ e che i prezzi del mercato seguono la misura $\mathbb{P}$. Allora l'obiettivo dell'*option pricing* è fornire i prezzi delle opzioni sotto la misura $\mathbb{Q}$ che siano in completa coerenza con il principio, ossia che non diano opportunità di arbitraggio nel mercato. Mentre lo scopo della calibrazione è trovare i parametri del modello considerato sotto la misura $\mathbb{Q}$ che meglio approssima i prezzi osservati nella misura $\mathbb{P}$, cioè si osserva l'evoluzione del mercato e si cerca di estrarre le informazioni che hanno causato tale evoluzione. Per entrambi i problemi è fondamentale che i prezzi siano in accordo con il mercato.
Si nota che la calibrazione rientra nell'ambito dell'ottimizzazione in quanto si vogliono stimare i parametri che minimizzano la distanza tra i prezzi. La sua complessità è determinata dalla natura del problema.

Capitolo 2

Modelli stocastici in Finanza

Per poter affrontare il problema della calibrazione è indispensabile introdurre i concetti matematici e finanziari alla base dei modelli stocastici. Si parte dal derivato e si prosegue poi con le opzioni e il sottostante. Quest'ultimo viene tipicamente modellizzato con un Moto Browniano Geometrico, composto da una parte deterministica e una aleatoria. Ovviamente nella parte stocastica si concentra la rappresentazione del rischio. Mettendo insieme un titolo rischioso e uno risk-free si costruisce un modello di mercato completo che con l'impiego di altre ipotesi come; il principio di non arbitraggio e la perfetta liquidità del mercato, permette di derivare l'equazione di Black-Scholes. Nonostante esso venga utilizzato molto tuttora, le osservazioni empiriche mostrano caratteristiche difficilmente raccontabili con un MBG e di conseguenza il modello derivante sarà limitato. Si accantona così Black-Scholes e si presentano i modelli di Heston e Bates basati sui processi di *Lévy*, in cui la volatilità è un'equazione stocastica a parte. Nuove difficoltà nascono considerando i processi di salto ma anche dall'aggiunta della componente stocastica nella volatilità ed è necessario calcolare la funzione caratteristica per Heston e Bates allo scopo di ottenere i prezzi delle opzioni.

2.1 Alcuni concetti fondamentali

Uno strumento finanziario molto interessante dal punto di vista matematico è il derivato. Esso rappresenta un contratto o un titolo che dipende dal sottostante , che può essere un'azione, un'obbligazione, un indice, una commodity o anche un altro derivato. La catena di derivati così creati può essere molto complicata. Questi strumenti vengono contrattati in molti mercati finanziari e il loro utilizzo è principalmente per arbitraggio, speculazione e copertura.
Le opzioni sono strumenti derivati che conferiscono al possessore il diritto, ma non l'obbligo, di comprare (*Call*) o vendere (*Put*) a una determinata data (detta exercise date) il sottostante a un determinato prezzo, detto *strike price*. Tipicamente quando non vengono aggiunti vincoli ulteriori le opzioni *Call* e *Put* così definite vengono dette *opzioni Europee*. Se il possessore ha il diritto di esercitare l'opzione in una qualsiasi data tra la sottoscrizione del contratto e la sua scadenza allora l'opzione è nota come *opzione Americana*. Queste ultime insieme alla famiglia delle opzioni Europee costituiscono le opzioni standard note come *plain vanilla*. Tuttavia, esistono altre opzioni che possono essere complicate richiedendo delle caratteristiche sull'andamento del sottostante oppure offrire ulteriori diritti al possessore. Alcuni esempi sono:le opzioni asiatiche, le opzioni esotiche, le opzioni lookback ecc.
Oltre a questi strumenti nel mercato reale si trovano anche i sottostanti elencati precedentemente che tipicamente in finanza vengono modellizzati con un processo stocastico particolare noto come Moto Browniano Geometrico (MBG) (2.2). Il processo vero di prezzo evolve seguendo la logica del mercato, cioè della domanda e dell'offerta. Risulta naturale pensare ad un flusso di informazioni che segue l'evoluzione del mercato e precisamente del singolo titolo. A tale scopo viene impiegato il simbolo di $\mathcal{F}_t^S$ denominata filtrazione, indica tutto il flusso di informazioni generati dal processo stocastico S nell'intervallo [0,t]. Quando la filtrazione accompagna l'evoluzione del processo stocastico, ossia se $S(t) \in \mathcal{F}_t^S$ per ogni $t \geq 0$, allora il processo S è detto adattato rispetto alla filtrazione $\left\{\mathcal{F}_t^S\right\}_{t\geq 0}$.

Nella struttura classica del modello di mercato viene aggiunto anche il titolo

risk-free che rappresenta il rendimento bancario

$$dB_t = rB_t dt \tag{2.1}$$

$$S_t = \alpha S_t dt + \sigma S_t dW_t \tag{2.2}$$

La (2.2) è un'equazione differenziale stocastica composta da un termine di drift α costante e il termine di diffusione σ, anche esso costante. Questa è una abbreviazione dell'equazione integrale:

$$S_t = S_0 + \int_0^t \alpha S_t dt + \int_0^t \sigma S_t dW_t \tag{2.3}$$

Sono note anche le soluzioni delle equazioni (2.1) e (2.2)

$$B_T = B_t e^{-r(T-t)}, \tag{2.4}$$

$$S_T = S_t e^{(\alpha - \frac{\sigma^2}{2})(T-t)+\sigma(W_T - W_t)}, \forall T \geq t. \tag{2.5}$$

W_t rappresenta il rumore Gaussiano e denota il processo di Wiener, utilizzato molto anche in fisica per modellizzare fenomeni casuali o semplicemente il rumore. Un processo stocastico W è chiamato processo di Wiener se valgono le seguenti:

1. W(0) = 0;
2. il processo ha incrementi indipendenti, presi due intervalli i tempo [s,t] e [u,v] tale che $t > s \geq v > u$ allora $\{W_t - W_s\}$ e $\{W_v - W_u\}$sono statisticamente indipendenti;
3. ha incrementi stazionari, la legge di $\{W_{t+h} - W_t\}$ non dipende da t ma solo da h ed è data dalla distribuzione Gaussian N[0, $\sqrt{h}$];
4. le traiettorie sono continue.

Chiaramente l'assenza della componente casuale W_t nell'equazione (2.1) è coerente con il nome del titolo, ossia la mancanza di rischio. La dinamica del MBG può essere vista sotto la misura $\mathbb{P}$ o $\mathbb{Q}$, nel primo caso si parla tipicamente del processo di prezzo osservato nel mercato. Le due misure sono collegate e il passaggio dalla misura $\mathbb{P}$ alla misura $\mathbb{Q}$ si effettua con il teorema di Girsanov [3].

Tipicamente la stocasticità viene inglobata nel processo di Wiener e di conseguenza l'integrale che si ottiene nella (2.3) viene chiamato integrale stocastico. Sostanzialmente si richiede che il processo da integrare S sia finito nello spazio $\mathbb{L}^2$ e che sia adattato rispetto alla filtrazione del processo di Wiener. Oltre all'integrale stocastico risulta indispensabile un altro strumento fornito dalla formula di Itô. Dato un processo stocastico definito così

$$dX_t = \alpha_t dt + \sigma_t dW_t \tag{2.6}$$

con α e σ processi adattati e f sia appartenete allo spazio delle funzioni continue $\mathbb{C}^{1,2}$. Si definisce un processo Z come $Z = f(t, X(t))$. Allora la formula di Itô assicura che il differenziale stocastico del processo Z sia dato da

$$df(t,X_t) = \left\{ \frac{\partial f}{\partial t} + \alpha \frac{\partial f}{\partial x} + \frac{1}{2}\sigma^2 \frac{\partial^2 f}{\partial^2 x} \right\} dt + \sigma \frac{\partial f}{\partial x} dW_t \tag{2.7}$$

Per la costruzione precisa dell'integrale stocastico e la dimostrazione della formula di Itô si rimanda al [4].

Con il bagaglio tecnico sopra elencato è possibile affrontare in modo formale alcuni concetti della finanza, a partire dai derivati. Un'opzione Europea non è altro che una funzione del sottostante, che in generale prende il nome di contratto. Fissato l'exercise date T e lo strike price K, il payoff dell'opzione Call Europea è data in termini formali da

$$C(T,S_T) = max(S_T - K, 0) \tag{2.8}$$

In figura 2.1 è mostrato il payoff della Call Europea con payoff dato dall'equazione (2.8), dal grafico si deduce che il possessore della Call vorrà esercitare l'opzione alla data di scadenza qualora il valore del sottostante superasse lo strike price $(S_T > K)$. Invece, se il sottostante dovesse essere inferiore rispetto allo strike $(S_T < K)$ il possessore non ha nessun interesse nell'esercitare l'opzione.
Analogamente si definisce il payoff dell'opzione Put Europea

$$P(T,S_T) = max(K - S_T, 0) \tag{2.9}$$

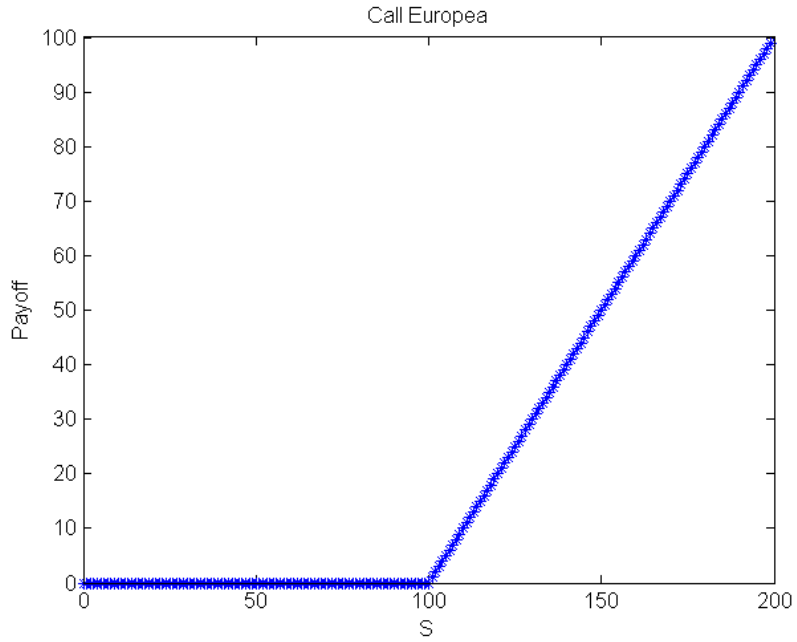

Figura 2.1: Payoff di una Call Europea con strke price K = 100.

Come mostra il grafico 2.2 si può fare un discorso simile alla Call Europea invertendo il segno della disuguaglianza tra sottostante a scadenza e strike. Il sottostante può avere un andamento simile al MBG, con una traiettoria imprevedibile e presentare così rischi di oscillazioni del prezzo. Per esempio il compratore del sottostante vorrebbe aver un prezzo massimo garantito (K) in un istante futuro, uno strumento di copertura utilizzato in questo caso è la Call. Il compratore paga l'opzione per proteggersi da un'eventuale rialzo del sottostante. Nel caso della Put l'attore principale è un venditore del sottostante che vuol proteggersi dal ribasso del prezzo.

Quindi se si considera un portafoglio costituito dagli strumenti finanziari presentati finora allora esso avrebbe al suo interno uno o più derivati, un o più titoli rischiosi e/o un titolo non rischioso. In questo mercato, ipotizzando che i titoli siamo sempre disponibili, un agente sceglierebbe quale titolo acquistare e quale vendere in qualsiasi istante di tempo creando così una sua strategia. In altre pa-

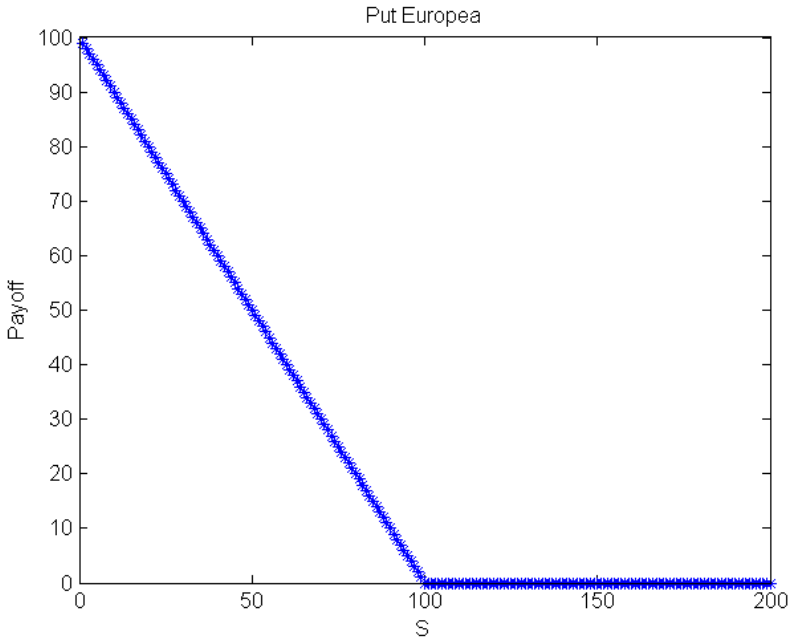

Figura 2.2: Payoff di una Put Europea con strke price K = 100.

role una strategia è una possibile allocazione degli strumenti del portafoglio sul mercato.

Una possibilità di arbitraggio in un mercato finanziario, secondo [3], è un portafoglio auto-finanziante con strategia h tale che

$$V^h(0) = 0, \tag{2.10}$$

$$P(V^h(T) \geq 0) = 1, \tag{2.11}$$

$$P(V^h(T) > 0) > 0 \tag{2.12}$$

Si dice che un portafoglio è *arbitrage free* se non esistono possibilità di arbitraggio. Queste definizioni seguono dall'assunzione accettata tipicamente da molti che i mercati siano efficienti. Sostanzialmente si richiede che non sia possibile fare soldi nel mercato partendo da un portafoglio vuoto e senza accollarsi alcun rischio. Tuttavia, può esistere una possibilità di arbitraggio nel mercato qualora ci sia un errore nel prezzamento di uno strumento.

Un altro concetto fondamentale è la completezza del mercato. Si dice che un con-

tratto Φ può essere replicato se esiste un portafoglio auto-finanziante con strategia h tale che

$$V^h(T) = \Phi,\ P - q.c. \tag{2.13}$$

Se ogni contratto è replicabile allora il mercato si dice completo. Spesso, dire che un contratto è replicabile equivale ad asserire che è raggiungibile o hedgeable. L'utilizzo di questi sinonimi per esprimere la completezza varia in base alla scelta dell'autore.
La teoria moderna dei derivati finanziari è strettamente collegata con la teoria delle martingale. Non è obiettivo di questa tesi fornire una panoramica di questo aspetto ma solo accennare l'idea. Un processo stocastico X è chiamato $\mathcal{F}_t$ *-martingala* se sono verificate le seguenti condizioni:

1. X è adattato rispetto alla filtrazione $\{\mathcal{F}_t\}_{t\geq 0}$;
2. $E\left[\mid X(t) \mid\right] < \infty \forall t$;
3. $\forall s,t$ tale che $s \leq t$ vale la relazione

$$E\left[X(t) \mid \mathcal{F}_s\right] = X(s). \tag{2.14}$$

La prima condizione afferma che possiamo osservare il valore di $X(t)$ al tempo t mentre la seconda è una condizione tecnica, stabilisce che il valore atteso del modulo sia finito. L'ultima condizione è fondamentale, asserisce che il valore atteso delle osservazioni future è dato dall'informazione corrente. Un altra modo per dire che la martingala in forma differenziale stocastica non può avere un componente di drift. Un esempio di martingala è il moto Browniano. Se consideriamo un modello di mercato con un solo titolo risk-free (2.1) e alcuni titoli $S_0, S_1 ... S_n$ come (2.2) allora la misura di martingala $\mathbb{Q}$ su $\mathcal{F}_T$ è chiamata una misura di martingala equivalente rispetto al numerario S_1 e l'intervallo [0,T] se possiede le seguenti proprietà:

1. $\mathbb{Q}$ è equivalente a $\mathbb{P}$ su $\mathcal{F}_T$;

2. tutti i processi di prezzo $S_0 ... S_n$ sono martingale sotto la misura $\mathbb{Q}$.

La prima proprietà stabilisce che per ogni evento ω tale che $\mathbb{P}(\omega) = 0$, anche $\mathbb{Q}(\omega) = 0$.
Con gli strumenti teorici presentati, è possibile fornire un paio di risultati cruciali noti come teoremi fondamentali della matematica finanziaria [3].
Il primo teorema fondamentale della matematica finanziaria asserisce: Un modello è arbitrage free se e solo se esiste una misura di martingala $\mathbb{Q}$. Quindi se si assume che il modello di mercato sia arbitrage free allora sicuramente esiste una misura di martingala. Il passo successivo è chiedersi se tale misura è unica.
Il secondo teorema fondamentale della matematica finanziaria: Se si assume l'assenza di arbitraggio, un modello di mercato è completo se e solo se la misura di martingala $\mathbb{Q}$ è unica. In un mercato completo il prezzo dei derivati è unicamente determinato dalla richiesta di assenza di arbitraggio e ogni derivato è replicabile con un portafoglio costituito dagli oggetti presenti nel mercato. Un esempio di modello in cui vale sia la completezza di mercato che l'assenza di arbitraggio è fornito dal modello di Black-Scholes. Esistono alcune regole del "pollice" proposte su [3] che permettono di riconoscere immediatamente se un mercato soddisfa il principio di non arbitraggio e/o è completo. Se si denota R come numero delle sorgenti di casualità e M il numero dei sottostanti rischiosi scambiabili nel mercato, allora:

1. il modello di mercato è arbitrage free se e solo se $M \leq R$
2. il modello di mercato è completo se e solo se $M \geq R$
3. il modello di mercato è completo e arbitrage free se e solo se M = R.

Un esempio della validità di tale regola è fornita dal modello di Black-Scholes, infatti si ha che M = 1 e R = 1.

2.2 Dal MBG ai processi a volatilità stocastica

Oltre al Moto Browniano Geometrico con coefficiente di drift e diffusione deterministico sono stati introdotti nuovi modelli con volatilità locale non lineare o stocastica. Ma tanti di questi modelli, anche se ben soddisfano alcune proprietà statistiche, sono continui e non rappresentano bene il fenomeno fisico. Sostanzialmente le osservazioni empiriche mostrano che i prezzi si muovono con dei salti continui, in completo disaccordo con la continuità del MBG [6]. Esso evolve seguendo una componente deterministica e i piccoli shock forniti dal processo di Wiener che possono essere visti come salti continui. L'interesse per questo fenomeno sale se si pensa che nei salti è contenuto il rischio di un processo stocastico. Per alcune attività finanziarie si cerca di modellizzare il rischio osservato negli eventi che accadono nei mercati finanziari. Esso può essere suddiviso in due macro aree: rischi finanziari e rischi non finanziari. La prima area è categorizzato in tre tipi [21]; rischio di mercato, rischio di credito e rischio di liquidità.
Una particolare classe di processi di salto sono i processi di *Lévy*. Un processo di *Lévy* $\{X_t\}_{t\geq 0}$ è un processo stocastico se soddisfa le seguenti proprietà:

1. è un processo cadlag, continuità delle traiettorie da destra con limite sinistro;

2. ha incrementi indipendenti, presi due intervalli i tempo [s,t] e [u,v] tale che $t > s \geq v > u$ allora $\{X_t - X_s\}\, e\, \{X_v - X_u\}$ sono statisticamente indipendenti;

3. presenta incrementi stazionari, la legge di $\{X_{t+h} - X_t\}$ non dipende da t ma solo da h;

4. ha continuità stocastica,

$$\forall \varepsilon > 0 \lim_{h\to 0} \mathcal{P}\{\mid X_{t+h} - X_t \mid > \varepsilon\} = 0. \tag{2.15}$$

La prima proprietà afferma che i salti siano imprevedibili, un requisito fondamentale nella modellizzazione, così il valore del processo è noto dopo il salto. L'ultima caratteristica non richiede che il processo sia continuo, ma che si escludano tutti i

gli effetti di calendario, cioè non considerare i salti in periodi di tempo prefissati. La seconda e la terza proprietà sono requisiti analoghi al processo di Wiener.
Secondo la decomposizione di *Lévy*-Itô un processo di *Lévy* si ottiene come somma di 4 elementi:

$$X_t = \gamma t + W_t + X_t^l + \tilde{X}_t \tag{2.16}$$

dove γt è il termine di deriva e W_t il quello di diffusione. Mentre

$$X_t^l = \sum_{0 \le s \le t}^{|\Delta X_s| \ge 1} \Delta X_s \tag{2.17}$$

rappresenta un processo di Poisson composto, la somma di un numero finito di salti in modulo maggiore di 1. L'ultimo elemento raccoglie il contributo di salti piccoli e uniformemente ravvicinati

$$\tilde{X}_t = \lim_{\varepsilon \to 0} \tilde{X}_t^{\varepsilon} \tag{2.18}$$

dove

$$\tilde{X}_t^{\varepsilon} = \sum_{0 \le s \le t}^{1 > |\Delta X_s| \ge \varepsilon} \Delta X_s. \tag{2.19}$$

Si osserva immediatamente che nella (2.16) se si pongono gli ultimi due termini a 0 si ottiene una classica equazione differenziale stocastica. Perciò si può pensare ai processi di *Lévy* come un'estensione del Moto Browniano Geometrico. Se in un modello si considera un processo di *Lévy* e un altro processo stocastica che descrive la volatilità presente nel primo allora si parla di modelli a volatilità stocastica, eventualmente con salti.
E' possibile definire la misura di salto per i processi con salti e trarre una serie di conclusioni indispensabili per la trattazione di questi processi, come: la condizione di martingala, la formula di Itô, il teorema di Girsanov ecc. Si rimanda a [6].

2.2.1 Il modello di Black-Scholes

Come detto alla fine del capitolo precedente il modello di Black-Scholes è definito sotto l'assunzione dell'assenza di arbitraggio e la completezza dei mercati. Inoltre

si ipotizza che ci sia perfetta liquidità del mercato: non ci sono costi di transazione, è possibile vendere allo scoperto e si può vendere o comprare un'arbitraria quantità di titoli. Si prende il modello di mercato presentato precedentemente (2.2) e (2.1) e si considera un'opzione Call Europea sul sottostante $S_t, C(t,S_t)$. Applicando la formula di Itô si ottiene l'equazione di Black-Scholes

$$\begin{aligned} \frac{\partial C}{\partial t}(t,S_t) + rS_t\frac{\partial C}{\partial S}(t,S_t) + \frac{1}{2}S_t^2\sigma^2\frac{\partial^2 C}{\partial^2 S}(t,S_t) - r\frac{\partial C}{\partial S} &= 0, \qquad (2.20)\\ C(T,S_T) = max(S_T - K, 0). \qquad (2.21)\end{aligned}$$

La seconda equazione denota la condizione finale. Comunque il sistema così presentato non è completo mancano le condizioni al bordo spaziali. Un altro modo di esprimere il prezzo di $C(t,S_t)$ è attraverso la valutazione neutrale al rischio

$$C(t,S_t) = e^{-r(T-t)}E^{Q}[max(S_t - K, 0) \mid \mathcal{F}_t] \qquad (2.22)$$

$\mathbb{Q}$ identifica la misura di martingala neutrale al rischio che descrive la dinamica di S, la sua esistenza è garantita dall'assunzione di assenza di arbitraggio nel mercato. La formula di valutazione neutrale al rischio esprime il prezzo del derivato come valore atteso del derivato a scadenza scontato dell'interesse bancario.Quest'ultimo è stato ipotizzato costante nel periodo di valutazione dell'opzione. E' possibile passare dalla formula di Black-Scholes alla formula di valutazione neutrale al rischio attraverso la formula di rappresentazione stocastica di Feynman-Kaĉ [3]. In pratica, per prezzare un'opzione con la formula di valutazione neutrale al rischio bisogna stimare il valore atteso che tipicamente si effettua con il metodo Monte Carlo. Il costo computazionale è un fattore proibitivo per MC quando si ha bisogno di un risultato in tempi rapidi.
Per alcuni derivati è nota la soluzione dell'equazione di Black-Scholes in forma esplicita. Infatti per una Call Europea con strike K e data di scadenza T si ha la formula di Black-Scholes

$$\begin{aligned} C(t,S_t) &= S_t N[d1(t,S_t)] - e^{-r(T-t)}KN[d2(t,S_t)], \qquad (2.23)\\ d1(t,S_t) &= \frac{1}{\sigma\sqrt{T-t}}\left\{ln\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)\right\}, \qquad (2.24)\\ d2(t,S_t) &= d1(t,S_t) - \sigma\sqrt{T-t} \qquad (2.25)\end{aligned}$$

dove N denota la funzione di ripartizione della distribuzione normale standard.

Oltre alle assunzione fatte in precedenza il modello di Black-Scholes ipotizza che la volatilità sia costante, fatto questo smentito dai dati di mercato. Infatti considerando come dato il prezzo dell'opzione Call e lasciando come incognita la volatilità si ottiene la superficie della volatilità implicita invertendo la formula di Black-Scholes. Ossia si vuole

$$min_{I\geq 0}(\mid C_{BS}(t,S_t;I) - C_{Market} \mid) \tag{2.26}$$

dove $C_{BS}(t,S_t;I)$ è la formula di Black-Scholes di una Call Europea fissando tutti i parametri tranne la volatilità I e C_{Market} sono i prezzi della corrispondente Call osservati nel mercato. Il problema così imposto potrebbe aver molti minimi locali, la sua unicità è garantita se si impone che $C_{BS}(t,S_t;0) < C_{Market}$ e $I \geq 0$. Se si fissano tutti i parametri tranne la volatilità, la funzione di Black-Scholes è convessa in essa perché si riesce dimostrare che

$$\frac{\partial C_B S}{\partial I} = \frac{S_t e^{\frac{-d1}{2}} \sqrt{T-t}}{\sqrt{2\pi}} \tag{2.27}$$

cioè la derivata è crescente al crescere della volatilità implicita [5]. La figura 2.3 evidenzia la dipendenza della volatilità implicita dallo strike e dal time-to-maturity $\tau = T - t$. E' possibile mostrare formalmente la sua dipendenza unicamente dal moneyness (S/K) e dal time-to-maturity [6]. Questa variabile mostra chiaramente le differenze tra i valori di mercato e i prezzi teorici, è un indice delle capacità previsionali del modello di Black-Scholes. Dalla figura 2.3 si nota che fissando il time-to-maturity la volatilità implicita presenta una convessità verso l'alto al variare dello strike, conosciuto come lo smile di volatilità. In particolare osservazioni empiriche mostrano che per derivati su tassi di cambio lo smile è simmetrico, mentre per Equities e Commodities è asimmetrico con sbilanciamento rispettivamente verso sinistra e verso destra.

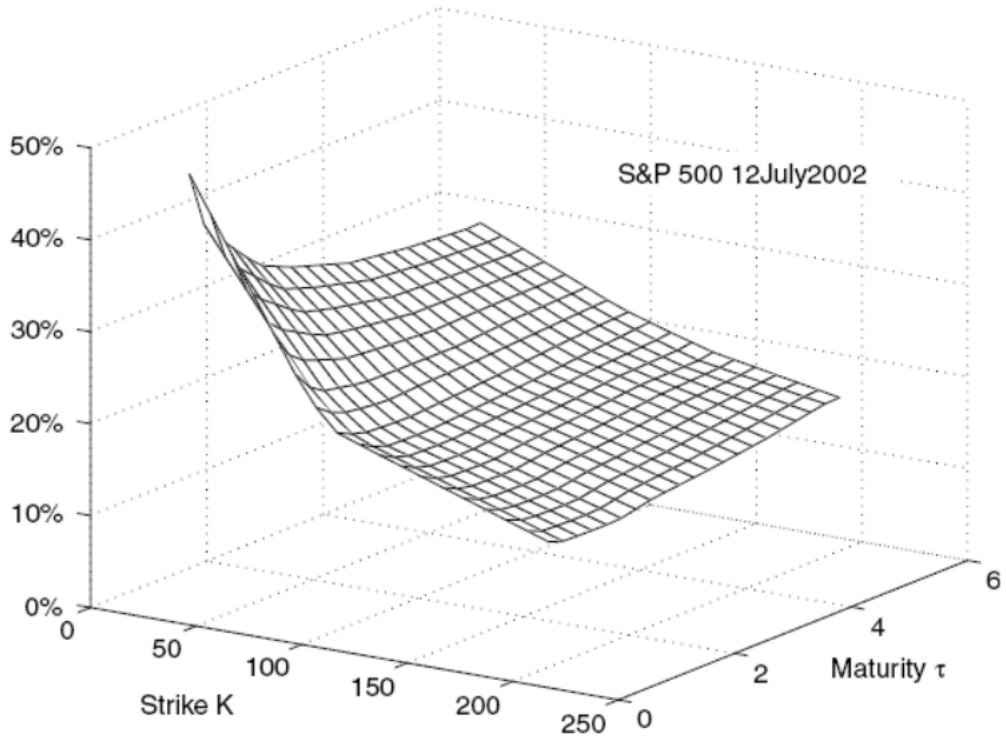

Figura 2.3: Superficie di volatilità per l'indice S&P 500

2.2.2 Il modello di Heston

Come descritto nella sezione precedente il modello di Black-Scholes è inconsistente dal punto di vista empirico. Sempre le osservazioni empiriche mostrano che generalmente il logaritmo dei rendimenti, ossia $log\left(\frac{dS_t}{S_t}\right)$, non è distribuito come una normale in quanto gli eventi estremi si verificano con maggiore intensità rispetto a quanto atteso dal modello. Si dice che le code della distribuzione empirica sono grasse, cioè si distanziano dalle code della distribuzione normale.
Per superare i limiti di Black-Scholes sono stati introdotti due tipologie di modelli sulla volatilità; modelli a volatilità locale e modelli a volatilità stocastica. I primi prevedono che la volatilità sia dipendente unicamente dal processo di prezzo, mentre nei secondi la volatilità è descritta con un altro processo e una nuova

casualità. Un esempio dei modelli a volatilità stocastica è il modello di Heston

$$\frac{dS_t}{S_t} = \mu dt + \sqrt{V_t} dW_t^S, \tag{2.28}$$

$$dV_t = \xi(\eta - V_t)\, dt + \theta\sqrt{V_t} dW_t^V. \tag{2.29}$$

dove dW_t^S e dW_t^V sono due processi di Wiener correlati

$$dW_t^S dW_t^V = \rho dt. \tag{2.30}$$

La seconda equazione di (2.29) evidenzia le caratteristiche innovative del modello di Heston presentato nel suo lavoro [7]. Innanzitutto è immediato osservare che l'equazione della volatilità è non lineare per la presenza della radica quadrata. Tuttavia la volatilità potrebbe assumere valori negativi, scadendo così di ogni significato. Per impedirlo è necessario imporre che essa non raggiunga neanche lo zero. In letteratura questo risultato è noto come la condizione di Feller $2\xi\eta > \theta^2$ [19]. Si nota che le oscillazioni tendono a ritornare verso il valore di η, che identifica la varianza di lungo periodo. Mentre ξ è velocità di ritorno al valore medio poiché $\frac{1}{\xi}$ rappresenta la frequenza delle oscillazioni. Le evidenze empiriche giustificano l'impiego della mean-reversion, mentre per quanto riguarda i rendimenti il modello di Heston permette di adattare la distribuzione a quella osservata nel mercato non solo nel momento primo e secondo ma anche terzo(skewness) e quarto(kurtosis). Infatti il termine di correlazione ρ ha un effetto sul terzo momento della distribuzione dei rendimenti, in modo che essa sia simmetrica ($\rho = 0$) e asimmetrica ($\rho \neq 0$) in base alle osservazioni.
θ è la volatilità della volatilità, indica l'intensità del rumore generato da dW_t^V.Per $\theta = 0$ si ha che l'equazione della volatilità è puramente deterministica, invece per $\theta \gg 0$ la parte deterministica è trascurabile. Oltre a queste osservazioni immediate θ agisce sul momento quarto della distribuzione dei rendimenti, influendo sullo spessore delle code.

2.2.3 Il modello di Bates

I processi di *Lévy* definiscono una classe di processi più generale di quelli di MBG aggiungendo salti ad essi, però mantenendo l'indipendenza dei ritorni logaritmici.

Il valore aggiunto fornito dai salti permette di descrivere meglio alcuni fenomeni osservati nei mercati. Per esempio si osserva che la volatilità dipende fortemente dallo *strike* e dal *time to maturity*. In questo modo, attraverso la dipendenza da due variabili, viene definito la superficie di volatilità con un alcune caratteristiche, fra le quali: lo smile di volatilità, lo skewness e alcune particolarità per scadenze ravvicinate. Nel modello di Black&Scholes la volatilità è considerata costante per tutto il tempo di vita dell'opzione, perciò questa ipotesi non consente di replicare molti fenomeni legati ad essa. Come detto nella precedente sezione, il modello di Heston permette di superare queste limitazioni. Tuttavia anche se il modello di Heston risulta adeguato per il clustering della volatilità, per lo smile e skew a lungo termine, esso non rappresenta bene alcune caratteristiche che si osservano nel mercato per scadenze ravvicinate.

Una soluzione a questo problema è servita dal modello di Bates [9] che estende Heston. Esso è descritto da un'equazione per la volatilità stocastica e una per il sottostante. Quest'ultima aggiunge salti al processo di prezzo di Heston, facendolo diventare un jump-diffusion che è un processo di *Lévy*:

$$\frac{dS_t}{S_t} = \mu dt + \sqrt{V_t} dW_t^S + dZ_t, \tag{2.31}$$

$$dV_t = \xi(\eta - V_t)\, dt + \theta\sqrt{V_t} dW_t^V. \tag{2.32}$$

Si nota subito la componente di salto dZ_t nel processo di prezzo. Infatti Z_t è un processo di Poisson composto ad intensità λ e con distribuzione dei salti di tipo log-normale $k_i \sim logN(\bar{k}, \delta^2)$.

$$Z_t = \sum_{i=1}^{N_t} k_i \tag{2.33}$$

Nella formula si nota N_t, che segue una distribuzione di Poisson con intensità λ mentre se k è l'ampiezza dei salti allora la sua distribuzione segue una normale con $ln(1+k_i) \sim N(ln(1+\bar{k}) - \frac{1}{2}\delta^2, \delta^2)$. Quest'ultima considerazione deriva direttamente dalla proprietà della distribuzione lognormale.

In figura 2.4 viene mostrato il processo di jump diffusion (2.31) con i seguenti parametri: $\mu = 0.03, \sqrt{V_t} = 0.2, \lambda = 5, \bar{k} = 10, S_0 = 100$ e passo di discretizzazio-

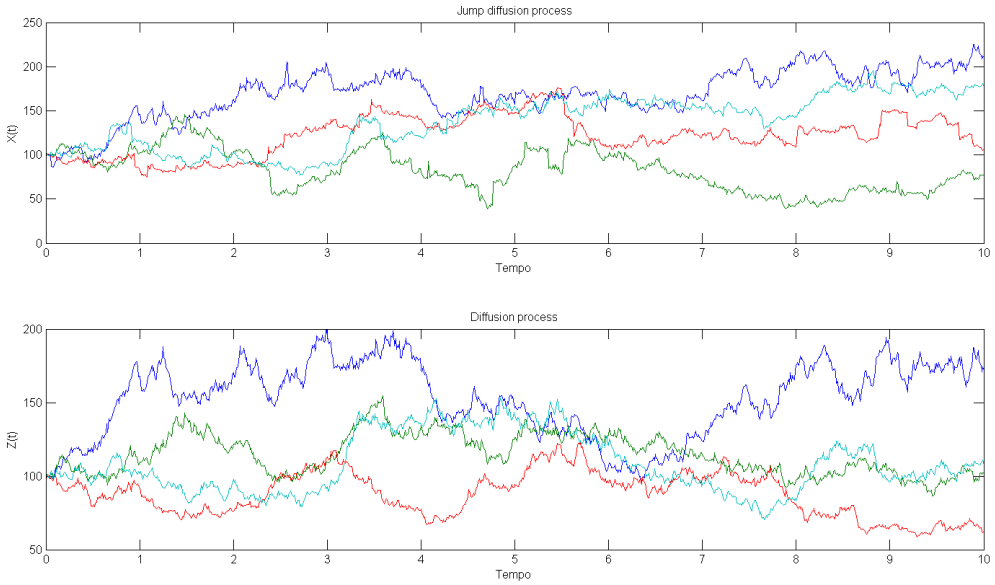

Figura 2.4: Simulazione di (2.31): in alto il jump diffusion e in basso il corrispondente processo di diffusione.

ne 1000. Si vuol metter l'accento sulle differenze tra i due processi in termini di variabilità e shock subiti nel tempo. I salti avvengono in corrispondenza di tratti verticali della traiettoria.

Per il principio di non arbitraggio il drift del processo sotto la misura neutrale al rischio deve uguagliare il rendimento della banca, perciò $\mu = r - \lambda\bar{k}$. Si vuol ragionare in termini di logaritmo del prezzo per una trattazione analitica più chiara anche in seguito, perciò si pone $X_t = lnS_t$ e si applica il lemma di Ito per i processi con salti.

$$dX_t = \frac{1}{S_{t-}}dS_t - \frac{1}{S_{t-}^2}dS_t^2 + \sum_{0\leq s\leq t}^{\Delta S\neq 0}\left[ln\left(S_{s-}+\Delta S\right) - ln\left(S_{s-}\right) - \frac{\Delta S}{S_{s-}}\right] \quad (2.34)$$

$$= \left(\mu - \frac{V_t}{2}\right)dt + \sqrt{V_t}dW_t^S + \sum_{0\leq s\leq t}^{\Delta S\neq 0}\left[ln\left(\frac{S_{s-}+\Delta S}{S_{s-}}\right) - \frac{\Delta S}{S_{s-}}\right] \quad (2.35)$$

Si ottiene

$$dX_t = \left(r - \lambda \bar{k} - \frac{1}{2} V_t \right) dt + \sqrt{V_t} dW_t^S + d\tilde{Z}_t \tag{2.36}$$

Adesso $\tilde{Z}_t$ è un processo di Poisson con intensità λ ma con distribuzione normale dei salti.

I due processi che costituiscono il modello di Bates hanno moti Browniani correlati $dW_t^S dW_t^V = \rho dt$. Il processo che descrive l'evoluzione della volatilità è stato ampiamente discusso nella presentazione del modello di Heston.

Per il modello di Bates esistono due modi per creare lo skew della volatilità implicita. Un modo è come per il modello di Heston, attraverso il $\rho < 0$ si ha un correlazione negativa tra i rendimenti e la volatilità che porta ad uno spostamento della distribuzione dei rendimenti verso destra. L'altro modo è costituito dai salti asimmetrici del processo di prezzo per opzioni a scadenza ravvicinata. Quindi sia la correlazione che i salti possono dare vita allo smile di volatilità, ma mentre la correlazione crea lo smile per scadenza lontane, i salti lo generano per scadenze ravvicinate e che si appiattiscono al crescere del time-to-maturity.

2.3 Le funzioni caratteristiche

Per alcuni processi di *Lévy* è possibile ottenere la funzione caratteristica del logaritmo del prezzo in forma chiusa [6]. I modelli di Heston e Bates rientrano in questa categoria. Ovviamente la funzione caratteristica di Bates incorpora anche i termini di salto. Siccome i salti sono omogenei e indipendenti dalla parte continua allora la funzione caratteristica può essere scomposta in due funzioni caratteristiche, una per i salti e l'altra per la componente continua del processo.

$$E\left[e^{iuX_t}\right] = E\left[e^{iuX_t^C}\right] E\left[e^{iuX_t^J}\right]$$

Quindi calcolando la funzione caratteristica per la parte continua di Bates si ottiene la corrispondente funzione caratteristica per Heston. Per calcolare la funzione caratteristica della componente continua di Bates non possiamo adottare un approccio diretto

$$f(x,v,t) = E\left[e^{iuX_t^C} | X_t^C = x, V_t = v\right]$$

in quanto non conosciamo la densità del processo. Applichiamo il lemma di Itô ad $f(x,v,t)$:

$$\begin{aligned} df &= \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial x}dx + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}dX^2 + \frac{\partial f}{\partial v}dV + \frac{1}{2}\frac{\partial^2 f}{\partial v^2}dV^2 + + \frac{\partial^2 f}{\partial x \partial v}dXdV \quad (2.37) \\ &= \left(\frac{\partial f}{\partial t} + (r - \lambda\bar{k} - \frac{V_t}{2})\frac{\partial f}{\partial x} + \frac{1}{2}V_t\frac{\partial^2 f}{\partial x^2} + \xi(\eta - V_t)\frac{\partial f}{\partial v} + \frac{1}{2}\theta^2 V_t\frac{\partial^2 f}{\partial v^2}\right)dt \\ &+ \rho\theta V_t\frac{\partial^2 f}{\partial x \partial v}dt + \sqrt{V_t}\frac{\partial f}{\partial x}dW^S + \theta\sqrt{V_t}\frac{\partial f}{\partial v}dW^V \quad (2.38) \end{aligned}$$

E' noto che $f(x,v,t)$ è una martingala sotto la misura $\mathbb{Q}$ e per poter preservare questa proprietà si deve porre il termine di drift a zero:

$$\frac{\partial f}{\partial t} + (r - \lambda\bar{k} - \frac{V_t}{2})\frac{\partial f}{\partial x} + \frac{1}{2}V_t\frac{\partial^2 f}{\partial x^2} + \xi(\eta - V_t)\frac{\partial f}{\partial v} + \frac{1}{2}\theta^2 V_t\frac{\partial^2 f}{\partial v^2} + \rho\theta V_t\frac{\partial^2 f}{\partial x \partial v} = 0 \quad (2.39)$$

Si ottiene un'equazione parabolica all'indietro a cui serve una condizione finale: $f(X_T, V_T, T) = e^{iux}$. Questo modello rientra nella classe dei modelli affini, per essi tipicamente si cerca una soluzione funzionale della seguente forma:

$$f(x,u,t) = exp\,(C(T-t) + vD(T-t) + iux) \quad (2.40)$$

Si può ridurre la dipendenza temporale passando al time-to-maturity.

$$f(x,u,s) = exp\,(C(s) + vD(s) + iux) \tag{2.41}$$

Si sostituisce la soluzione ipotizzata alla funzione caratteristica che si sta cercando.

$$\begin{aligned} - \quad & \left[C'(s) + vD'(s)\right] + \left(r - \lambda\bar{k} - \frac{v}{2}\right) iu + \xi(\eta - v)D(s) + \frac{v}{2}i^2u^2 \\ + \quad & \frac{1}{2}\theta^2 vD^2(s) + \rho\theta vD(s)iu = 0 \end{aligned} \tag{2.42}$$

L'equazione contiene un polinomi di primo grado in v che ha come coefficiente un'equazione differenziale di primo ordine in D

$$-\dot{D} - \frac{iu}{2} - \frac{u^2}{2} + \frac{1}{2}\theta^2 D^2(s) - \xi D(s) + \rho\theta iuD(s) = 0 \tag{2.43}$$

$$D(0) = 0 \tag{2.44}$$

e un'equazione differenziale in C come termine noto

$$-\dot{C} + (r - \lambda\bar{k})iu + \xi\eta D(s) = 0 \tag{2.45}$$

$$C(0) = 0 \tag{2.46}$$

Entrambi i sistemi possono essere risolti esplicitamente, il primo è un'equazione di Riccati. Si ricava

$$D(s) = \frac{-(u^2 + iu)}{\gamma coth(\frac{\gamma s}{2}) + \xi - i\rho\theta u} \tag{2.47}$$

$$\begin{aligned} C(s) &= ius(r - \lambda\bar{k}) + \frac{\xi\eta s(\xi - i\rho\theta u)}{\theta^2} \\ &- \frac{2\xi\eta}{\theta^2} ln\left(cosh(\frac{\gamma s}{2}) + \frac{\xi - i\rho\theta u}{\theta^2} sinh(\frac{\gamma s}{2})\right) \end{aligned} \tag{2.48}$$

dove $\gamma = \sqrt{\theta^2(u^2 + iu) + (\xi - i\rho\theta u)^2}$.

Infine sostituendo alla $f(x,v,t)$ si ricava la seguente funzione caratteristica per

Heston e la componente continua della funzione caratteristica di Bates :

$$\phi_t^C(u) = \frac{exp\left(\frac{\xi\eta t(\xi - i\rho\theta u)}{\theta^2} + iut(r - \lambda\bar{k}) + iux_0\right)}{\left(cosh(\frac{\gamma t}{2}) + \frac{\xi - i\rho\theta u}{\theta^2} sinh(\frac{\gamma t}{2})\right)^{\frac{2\xi\eta}{\theta^2}}} exp\left(\frac{-(u^2 + iu)}{\gamma coth(\frac{\gamma t}{2}) + \xi - i\rho\theta u}\right) \tag{2.49}$$

Come descritto precedentemente, i salti nel modello di Bates rientrano nel processo di prezzo. Essi sono rappresentati con un processo di Poisson composto d'intensità λ e con ampiezza dei salti indipendenti e identicamente distribuiti secondo una legge normale:

$$k_i \, i.i.d. \sim N(\mu, \sigma^2) \tag{2.50}$$

e la corrispondente funzione caratteristica

$$\hat{f(u)} = E\left[e^{iuk_i}\right]. \tag{2.51}$$

Inoltre è noto che la probabilità che si verifichino un certo numero di salti per un processo di Poisson è data da:

$$P[N_t = n] = e^{-\lambda t} \frac{(\lambda t)^n}{n!} \tag{2.52}$$

Mettendo assieme quanto esposto e applicando la regola del valore atteso iterato

alla funzione caratteristica, si ha:

$$
\begin{aligned}
E\left[e^{iuX_t^J}\right] &= E\left[E[e^{iuX_t^J}|N_t]\right] && (2.53)\\
&= E\left[\hat{f}(u)^{N_t}\right] && (2.54)\\
&= \sum_{n=0}^{+\infty} e^{-\lambda t}\frac{(\lambda t)^n}{n!}\hat{f}(u)^n && (2.55)\\
&= \sum_{n=0}^{+\infty} e^{-\lambda t}\frac{\left(\lambda t\hat{f}(u)\right)^n}{n!} && (2.56)\\
&= e^{-\lambda t}\sum_{n=0}^{+\infty}\frac{\left(\lambda t\hat{f}(u)\right)^n}{n!} && (2.57)\\
&= e^{-\lambda t}e^{\lambda t\hat{f}(u)} && (2.58)\\
&= e^{\lambda t\left(\hat{f}(u)-1\right)} && (2.59)\\
&= exp\left\{\lambda t\left(\int_{\mathbb{R}} e^{iuk}\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(k-\mu)^2}{2\sigma^2}}dk-1\right)\right\} && (2.60)\\
&= exp\left\{\lambda t\left(\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(k-\mu)^2+iuk2\sigma^2}{2\sigma^2}}dk-1\right)\right\} && (2.61)\\
&= exp\left\{\lambda t\left(\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-k^2+2k(\mu+iu\sigma^2)-\mu^2}{2\sigma^2}}dk-1\right)\right\} && (2.62)\\
&= exp\left\{\lambda t\left(\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-k^2+2k(\mu+iu\sigma^2)-(\mu+iu\sigma^2)^2+2\sigma^2iu\mu-u^2\sigma^4}{2\sigma^2}}dk-1\right)\right\} && (2.63)\\
&= exp\left\{\lambda t\left(\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(k^2-(\mu+iu\sigma^2))^2+2\sigma^2iu\mu-u^2\sigma^4}{2\sigma^2}}dk-1\right)\right\} && (2.64)\\
&= exp\left\{\lambda t\left(e^{iu\mu-\frac{u^2\sigma^2}{2}}\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(k^2-(\mu+iu\sigma^2))^2}{2\sigma^2}}dk-1\right)\right\} && (2.65)\\
&= exp\left\{\lambda t\left(e^{iu\mu-\frac{u^2\sigma^2}{2}}-1\right)\right\} && (2.66)
\end{aligned}
$$

Per il processo che si sta considerando i parametri della normale sono: $\mu = ln(1+\bar{k})-\frac{1}{2}\delta^2, \sigma^2=\delta^2$. Quindi sostituendo nell'ultima espressione i nuovi parametri si ha:

$$\phi_t^J(u) = exp\left\{-\lambda t\left(e^{iu(ln(1+\bar{k})-\frac{1}{2}\delta^2)-\frac{u^2\delta^2}{2}}-1\right)\right\} \quad (2.67)$$

Sostituendo le conclusioni a cui si è arrivati per entrambe le componenti del processo logaritmo di prezzo, la funzione caratteristica di Bates è data dal prodotto

delle funzioni caratteristiche, della parte continua per i salti

$$\phi_t(u) = \phi_t^J(u)\phi_t^C(u). \tag{2.68}$$

Capitolo 3

La calibrazione

La trattazione prosegue coinvolgendo i modelli esposti. Le ipotesi su cui si sorregge Black-Scholes risultano poco realistiche ed è necessario rinunciare alla completezza del mercato per considerare i modelli di Heston e Bates. I loro vantaggi sono stati discussi nel capitolo 2 mentre rimane ancora da discutere come si possono sfruttare per il pricing. Recentemente ha preso piede il metodo ideato da Carr e Madan che permette di prezzare un opzione Call Europea (Put) semplicemente conoscendo la funzione caratteristica del sottostante. I prezzi osservati sul mercato insieme ai prezzi di Carr-Madan sono utilizzati per formare il problema di calibrazione. Si tratta di un problema di minimizzazione in cui non è garantito l'esistenza e neanche l'unicità della soluzione. Si affronta il problema considerando la regione di interesse con tanti punti i partenza distribuiti uniformemente da cui cominciare la ricerca dell'eventuale minimo. L'algoritmo di ricerca considerato, anche se solo un'euristica e sensibile al punto iniziale, presenta dei vantaggi notevoli rispetto ai suo concorrenti in termini di tempo. Nell'ultima sezione viene descritto la sequenza di Halton adoperata nel costruzione dei punti di partenza, indispensabili all'algoritmo di ricerca del minimo.

3.1 Il problema di calibrazione

Nel capitolo precedente sono stati presentati alcuni modelli considerati in problematiche di option pricing, a partire dalla formula di Black-Scholes fino al modello di Bates. E' necessario specificare che alcune opzioni in cui si ipotizza la dinamica del sottostante come nei modelli di Heston e Bates possono avere un'equazione alle derivate parziali simile a quella di Black-Scholes ma di cui non è possibile ricavare una formula esplicita. Requisito fondamentale per usare questi modelli nel prezzamento delle opzioni è conoscere il valore dei parametri che descrivono la dinamica del prezzo ed eventualmente la dinamica della volatilità. Generalmente l'esigenza nasce quando si vuol introdurre nel mercato un nuovo derivato, allora si sfruttano le informazioni presenti nel mercato per poter stimare questi parametri e prezzare successivamente il derivato con il modello. In questo modo si ottiene un prezzo che non dovrebbe introdurre opportunità di arbitraggio nel mercato. Quindi per risolvere un problema diretto, cioè prezzare un derivato, si ha bisogno della soluzione del problema inverso, ossia calibrare i parametri.
Come sottolineato nel capitolo 2, il modello di Black-Scholes opera sotto le ipotesi di assenza di arbitraggio e completezza del mercato. Grazie a queste ipotesi è noto dal secondo teorema fondamentale della matematica finanziari che esiste una misura di martingala equivalente ed essa è unica. Ma se tale misura è unica allora i prezzi forniti dal modello in considerazione sono unici. Le osservazioni fatte nel capitolo 2 permettono di dedurre che tale modello non sia la scelta migliore mentre i modelli di Heston e Bates danno la possibilità di replicare alcuni fenomeni osservabili nel mercato che per Black-Scholes non esistono.
Se si considera un modello di mercato descritto da (2.29) e (2.1) allora il modello di Heston presenta due fonti di casualità per ogni processo di prezzo. La regola del "pollice" assicura che il modello di mercato che si sta considerando non è completo perché il numero di sottostanti scambiabili è sempre uno (M = 1) mentre le sorgenti di casualità sono 2, cioè $R > M$. Se ci si chiede se la misura di martingala è unica, ovviamente la risposta è negativa. Infatti scegliendo il modello di Heston piuttosto che quello di Bates si effettua in automatico una scelta sulla misura di martingala equivalente.

Quindi raccogliendo le informazioni presenti nel mercato attraverso i prezzi e scegliendo un modello con una misura neutrale al rischio in modo tale che possa replicare i prezzi, in sostanza si effettua una calibrazione del modello utilizzato. Tipicamente i prezzi osservati vengono detti *benchmark*. Se supponiamo che i dati di benchmark di una Call *plain vanilla* siano d e il modello che si è scelto possiede n parametri, allora il problema della calibrazione può essere espresso formalmente così:

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^{d} \omega_i \mid C(T_i, K_i, \theta) - C_i^{mkt} \mid^2 \tag{3.1}$$

dove il vettore θ rappresenta i parametri del modello considerato, presi in uno spazio Euclideo di n dimensioni. Con ω_i si indicano i pesi che tipicamente vengono utilizzati per bilanciare l'ampiezza dei termini della sommatoria in base alla confidenza che si ha con i dati [6], ossia come l'inverso della differenza tra i prezzi bid-ask. In questo lavoro vengono imposti a 1. Mentre i $C(T_i, K_i, \theta)$ sono prezzi forniti dal modello con parametri fissati a θ e C_i^{mkt} sono i prezzi osservati nel mercato per l'opzione che si sta valutando.

L'equazione (3.1) fa parte della classe di problemi inversi, associata al problema di prezzamento(diretto). Come accennato precedentemente i prezzi di mercato possono essere replicati con diversi modelli e ciò fa pensare che la soluzione di (3.1) non sarà unica. Non solo, non è garantito nemmeno l'esistenza di una soluzione poiché tipicamente si sceglie un modello, di conseguenza la misura di martingala, ma non esiste nessuna argomentazione teorica per asserire che il mercato segua il modello assunto [6]. Altre complicazioni compaiono nel calcolo della soluzione, essa non dipende con continuità dai dati e può generare instabilità numerica nella ricerca del minimo. Sulla base di queste osservazioni si deduce che il problema è mal posto e può presentare tanti minimi locali, se esistono.

3.2 Il metodo di Carr-Madan

Diversamente dal modello di Black-Scholes, in cui è possibile ricavare una formula chiusa per prezzare un'opzione Call, nei processi di *Lévy* e nei modelli a volatilità stocastica non esistono formule esplicite perché la funzione densità di probabilità di tali processi non è nota in forma chiusa. Tuttavia esiste un metodo, ideato da Carr e Madan [8], che permette di sfruttare la conoscenza della funzione caratteristica della densità per poter prezzare un'opzione Europea. In questo modo è possibile ricavare i prezzi $C(T_i, K_i, \theta)$ del problema (3.1).
Si considera il problema di prezzamento di un'opzione call Europea a scadenza T sottoscritta su un sottostante con prezzo finale S_T. La funzione caratteristica di $X_T = ln(S_T)$ è data da:

$$\phi_T(u) = E[exp(iuX_T)] \tag{3.2}$$

Per un'opzione Europea viene definito anche lo strike K e si sceglie, per coerenza, di operare la stessa trasformazione del prezzo, cioè si considera il logaritmo dello strike. Si definisce quindi $C_T(k)$ il valore dell'opzione a scadenza T e con strike $exp(k)$. Viene richiesto che il sottostante sia a incrementi indipendenti e infinitamente divisibile. Condizioni soddisfatte per entrambi i modelli considerati. Inoltre devono essere soddisfatte alcune condizioni sull'integrabilità in modo da poter utilizzare il teorema di Fubini-Tonelli [2]. La densità neutrale al rischio per il processo del logaritmo del prezzo X_T è data da $q_T(x)$. Per essa è possibile stabilire la funzione caratteristica:

$$\phi_T(u) = \int_{-\infty}^{\infty} e^{iux} q_T(x) dx. \tag{3.3}$$

Secondo il principio di assenza di arbitraggio, il prezzo di un'opzione è dato dal valore atteso rispetto alla misura neutrale al rischio del payoff a scadenza scontato fino all'istante di valutazione. In termini matematici:

$$V_t = e^{-r(T-t)} E^Q [\Psi(S_T) | \mathcal{F}_t] \tag{3.4}$$

il payoff è $\Psi(S_T)$ dipendente dal valore finale del prezzo e $\mathcal{F}_t$ è la filtrazione del processo.

Per un'opzione call Europea si ha:

$$V_t = e^{-r(T-t)} E^Q \left[(S_T - K)^+ \, | \mathcal{F}_t \right] \tag{3.5}$$

passando adesso al problema che stiamo considerando e alle assunzioni fatte:

$$C_{T-t} = e^{-r(T-t)} E^Q \left[\left(e^{X_T} - e^k \right)^+ | \mathcal{F}_t \right]. \tag{3.6}$$

Se si suppone che $t = 0$ e di voler prezzare l'opzione al variare degli log-strike la formula precedente assume questa forma:

$$C_T(k) = e^{-rT} E^Q \left[\left(e^{X_T} - e^k \right)^+ \right] \tag{3.7}$$

più precisamente

$$C_T(k) = \int_{-\infty}^{\infty} e^{-rT} \left(e^x - e^k \right)^+ q_T(x) dx \tag{3.8}$$

$$= \int_k^{\infty} e^{-rT} \left(e^x - e^k \right) q_T(x) dx \tag{3.9}$$

All'interno dell'integrale si nota ancora il payoff, pesato per la densità e il dominio di variazione da k fino all'infinito. Si osserva che per k tendente a $-\infty$ il valore dell'opzione tende a S_0. Infatti:

$$\lim_{k \to -\infty} C_T(k) = \int_{-\infty}^{\infty} e^{-rT} e^x q_T(x) dx \tag{3.10}$$

$$= E^Q \left[e^{-rT} S_T | \mathcal{F}_0 \right] \tag{3.11}$$

$$= S_0 \tag{3.12}$$

quindi la funzione non appartiene ad L^2. Per poter ottenere una funzione quadrato integrabile si modifica il valore dell'opzione per un coefficiente dipendente da k, in questo modo:

$$c_T(k) = exp(\alpha k) C_T(k) \tag{3.13}$$

per $\alpha > 0$ si ha che $c_T(k)$ tende a zero.

Nel prossimo passaggio viene applicata la trasformata di Fourier alla funzione integrabile in modo da sviluppare un'espressione dipendente dalla funzione caratteristica della densità del processo. Quindi si definisce

$$\psi(v) = \int_{-\infty}^{\infty} e^{ivk} c_T(k) dk \tag{3.14}$$

la trasformata di Fourier del valore dell'opzione modificato. Scritto esplicitamente si ha:

$$\begin{aligned}
\psi(v) &= \int_{-\infty}^{\infty} e^{ivk} \int_{k}^{\infty} e^{\alpha k} e^{-rT} (e^x - e^k) q_T(x) dx dk && (3.15)\\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{\alpha k} e^{ivk} e^{-rT} (e^x - e^k)^+ q_T(x) dx dk && (3.16)\\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{\alpha k} e^{ivk} e^{-rT} (e^x - e^k)^+ q_T(x) dk dx && (3.17)\\
&= \int_{-\infty}^{\infty} e^{-rT} q_T(x) \int_{-\infty}^{x} e^{\alpha k} e^{ivk} (e^x - e^k) dk dx && (3.18)\\
&= \int_{-\infty}^{\infty} e^{-rT} q_T(x) \int_{-\infty}^{x} e^{\alpha k + ivk + x} - e^{\alpha k + ivk + k} dk dx && (3.19)\\
&= \int_{-\infty}^{\infty} e^{-rT} q_T(x) \left[\frac{e^x}{\alpha + iv} e^{\alpha + iv} - \frac{e^{\alpha x + ivx + x}}{\alpha + iv + 1} \right] dx && (3.20)\\
&= \int_{-\infty}^{\infty} e^{-rT} q_T(x) \frac{e^{\alpha x + ivx + x}}{(\alpha + iv)(\alpha + iv + 1)} dx && (3.21)\\
&= \frac{e^{-rT}}{(\alpha + iv)(\alpha + iv + 1)} \int_{-\infty}^{\infty} q_T(x) e^{i(-i\alpha + v - i)x} dx && (3.22)\\
&= \frac{e^{-rT}}{(\alpha + iv)(\alpha + iv + 1)} \phi_T(v - i\alpha - i) && (3.23)
\end{aligned}$$

Nell'ultimo passaggio si nota che per $\alpha = 0$ e $v = 0$,il denominatore si annulla. Siccome la $\psi(v)$ deve essere valutata in zero, viene fatta la scelta del fattore di sconto $exp(\alpha k)$ con $\alpha \neq 0$. Rimane la determinazione di un limite superiore per α. Si osserva che i valori positivi di α aiutano l'integrabilità del valore della Call modificata quando i log strike assumono valori negativi. Mentre la aggravano per log strike positivi. Una condizione sufficiente è fornita per $\psi(0)$ finito, in quanto l'integrale del valore della Call modificata sarebbe finito. Questa condizione è verificata se $\phi_T(-i\alpha - i)$ è finito, ma dalla definizione di funzione caratteristica si ha :

$$\begin{aligned}
\phi_T(v) &= E\left[e^{ivX_T}\right] && (3.24)\\
\phi_T(-i\alpha - i) &= E\left[e^{-i^2(\alpha+1)X_T}\right] && (3.25)\\
&= E\left[e^{(\alpha+1)X_T}\right] && (3.26)\\
&= E\left[S_T^{(\alpha+1)}\right] < \infty && (3.27)
\end{aligned}$$

In pratica il limite superiore si determina dalla condizione precedente e dall'espressione analitica della funzione caratteristica.

Una volta determinata la $\psi(v)$ allora si applica la trasformata inversa di Fourier per poter ottenere il prezzo dell'opzione, scontando per il fattore che ci ha permesso di ottenere una funzione integrabile. Quindi:

$$C_T(k) = \frac{exp(-\alpha k)}{2\pi} \int_{-\infty}^{\infty} e^{-ivk} \psi(v) dv \tag{3.28}$$

$$= \frac{exp(-\alpha k)}{\pi} \int_{0}^{\infty} e^{-ivk} \psi(v) dv. \tag{3.29}$$

L'ultima uguaglianza è possibile poichè il prezzo dell'opzione è reale. Questo implica che la parte immaginaria sarà dispari mentre quella reale pari, cioè l'integrale precedente è uguale a due volte lo stesso integrale da zero all'infinito.

3.3 Fast Fourier Trasform

Come visto precedentemente la trasformata di Fourier è uno strumento molto utile in finanza per la metodologia di pricing che ne deriva e per le prestazioni computazionali. Infatti esistono molti algoritmi che permettono di ottenere la trasformata discreta di Fourier in tempi concorrenziali, la sua complessità è O(N log(N)).

Il valore dell'opzione in considerazione è dato dalla trasformata inversa di Fourier che, passando in forma discreta, diventa:

$$C_T(k) = \frac{exp(-\alpha k)}{\pi} \int_{0}^{\infty} e^{-ivk} \psi(v) dv \tag{3.30}$$

$$\approx \frac{exp(-\alpha k)}{\pi} \sum_{j=1}^{N} e^{-iv_j k_i} \psi(v_j) dv \tag{3.31}$$

Tipicamente N assume una potenza del 2 mentre la discretizzazione del dominio di integrazione segue l'equazione di una retta:

$$v_j = v_1 + (j-1)dv \; con \; j = 1...N$$

con coefficiente dv legato al log strike attraverso $dkdv = \frac{2\pi}{N}$. Analogamente la discretizzazione del log strike è dato dalla retta:

$$k_i = k_1 + (u-1)dk \; con \, u = 1...N$$

Si vuol ricavare un'espressione per il valore dell'opzione $C_T(k)$ che assomigli alla trasformata discreta di Fourier. Quindi

$$\begin{aligned}
C_T(k_u) &= \frac{exp(-\alpha k_u)}{\pi} \sum_{j=1}^{N} e^{-i(v_1+(j-1)dv)(k_1+(u-1)dk)} \psi(v_j) dv && (3.32)\\
&= \frac{exp(-\alpha k_u)}{\pi} \sum_{j=1}^{N} e^{-iv_1 k_i} e^{-i((j-1)dv)(k_1+(u-1)dk)} \psi(v_j) dv && (3.33)\\
&= \frac{exp(-\alpha k_u - iv_1 k_u)}{\pi} \sum_{j=1}^{N} e^{-ik_1(j-1)dv} e^{-i(j-1)(u-1)dvdk} \psi(v_j) dv && (3.34)\\
&= \frac{exp(-\alpha k_u - iv_1 k_u)}{\pi} \sum_{j=1}^{N} e^{-i\frac{2\pi}{N}(j-1)(u-1)} e^{-ik_1(j-1)dv} \psi(v_j) dv && (3.35)\\
&= \frac{exp(-\alpha k_u - iv_1 k_u)}{\pi} \sum_{j=1}^{N} e^{-i\frac{2\pi}{N}(j-1)(u-1)} e^{-ik_1(j-1)dv} \psi(v_j) dv && (3.36)
\end{aligned}$$

Nell'ultima espressione si nota che la funzione a cui applicare la trasformata è data da: $f(j) = e^{-ik_1(j-1)dv}\psi(v_j)$. Per poter avere una griglia fine per l'integrazione si desidera una dv piccola, ma questo implica che dk sia grande e quindi si hanno pochi punti per la stima del prezzo dell'opzione. Tale inconveniente è risolto con la regola di Simpson. Si moltiplica (3.36) per essa

$$\begin{aligned}
C_T(k_u) &= \frac{exp(-\alpha k_u - iv_1 k_u)}{\pi} \sum_{j=1}^{N} e^{-i\frac{2\pi}{N}(j-1)(u-1)} e^{-ik_1(j-1)dv} \psi(v_j)\\
&\cdot \quad \frac{dv}{3}(3 + (-1)^j - \delta_{j-1}). && (3.37)
\end{aligned}$$

L'espressione nella seconda riga rappresenta la regola di Simpson, mentre δ_j è la delta di Kronecker che vale 1 per $j = 0$ e 0 altrimenti.
I prezzi $C_T(k_u)$ sono ottenuti in corrispondenza di $k_u \in k_1, k_N$. Ma i prezzi osservati nel mercato potrebbero avere strike non appartenente a tale insieme, allora si

ricorre all'interpolazione dei prezzi $C_T(k_u)$ e si valuta la curva ottenuto nei strike di mercato. Tipicamente si sceglie un'interpolazione cubica indirizzata dalla forma della curva dei prezzi, sono preferibili le spline.

3.4 L'algoritmo di Levenberg-Marquardt

Il problema di calibrazione descritto in precedenza (3.1) necessita di un algoritmo per la ricerca dei minimi. Nonostante le premesse pessimistiche, esistono dei metodi che permettono di effettuare una ricerca dei minimi locali in ambito multidimensionale, anche se non sarà completa in quanto l'algoritmo usato non assicura il minimo globale. L'algoritmo più utilizzato per i problemi ai minimi quadrati non lineari è quello Levenberg-Marquardt. Esso garantisce alcuni vantaggi rispetto ai metodi classici. Tuttavia, l'applicazione primaria dell'algoritmo è nel fitting della curva per problemi ai minimi quadrati. L'obiettivo della calibrazione è raggiungere la migliore approssimazione dei prezzi forniti perché chiedere che ci sia la calibrazione esatta significherebbe che il mercato debba obbedire al modello considerato. Questa è un'assunzione poco realistica.

Si riformula il problema (3.1) in questo modo

$$r_i(\theta) = \mid C(T_i, K_i, \theta) - C_i^{mkt} \mid^2 \tag{3.38}$$

dove gli ω_i sono posti a 1. Inoltre si definisce la funzione obiettivo $f(\theta) = \|r(\theta)\|^2$, dove $r(\theta) = (r_1(\theta), r_2(\theta), ..., r_d(\theta))$. In seguito a queste trasformazioni il problema di calibrazione diventa; trovare il vettore dei parametri θ tale che la funzione obiettivo sia minima

$$\theta^* = \underset{\theta \in \mathbb{R}^n}{\arg\min}\, f(\theta). \tag{3.39}$$

Lo Jacobiano di r è una matrice data da $J(\theta) = \frac{\partial r_j}{\partial \theta_i}$ con $1 \leq j \leq d$ e $1 \leq i \leq n$. Il

passo successivo è la definizione dello Jacobiano e della Hessiana di f

$$\nabla f(\theta) = \sum_{i=1}^{d} r_i(\theta)\nabla r_i(\theta) = J(\theta)^T r(\theta) \tag{3.40}$$

$$\nabla^2 f(\theta) = J(\theta)^T J(\theta) + \sum_{i=1}^{d} r_i(\theta)\nabla^2 r_i(\theta). \tag{3.41}$$

Si osserva che se fosse possibile approssimare linearmente $r_i(\theta)$ allora $\nabla^2 r_i(\theta)$ diventerebbe trascurabile, mentre la Hessiana sarebbe semplicemente il prodotto degli Jacobiani e non peserebbe molto in termini computazionali. Questa è una approssimazione possibile per velocizzare l'algoritmo che ovviamente ha un costo in termini di accuratezza nel caso in cui la funzione obiettivo sia non lineari. Tipicamente il ventaglio di scelte di algoritmi è costituito dal metodo del gradiente, algoritmo di Gauss-Newton e Levenberg-Marquardt [13]. Essenzialmente per tutti e tre la logica è contenuta nella scelta del passo successivo. Per il metodo del gradiente l'aggiornamento del passo è dato dall'aggiunta del gradiente negativo scontato di un parametro $\lambda \in \mathbb{R}$

$$\theta_{i+1} = \theta_i - \lambda\nabla f(\theta_i) \tag{3.42}$$

variabile ad ogni iterazione. Tralasciando il calcolo di λ, il metodo presenta alcuni problemi di convergenza. Si vorrebbero fare passi grandi quando la discesa è piccola per raggiungere velocemente il minimo, e passi brevi quando la discesa è grande per non allontanarsi dal minimo. Ma con la regola precedente si fa esattamente l'opposto; se il gradiente è elevato si fanno passi grandi rischiando di oscillare nel punto di minimo e se esso è piccolo si eseguono passi brevi aumentando il tempo di convergenza al punto di minimo. Un'altra difficoltà si incontra dall'assenza di informazioni sulla curvatura. Per esempio, in una superficie a forma di valle lunga e stretta in cui il gradiente in una direzione è elevato e nell'altra relativamente piccolo il metodo compierebbe passi grandi lungo la parete e passi piccoli lungo la base. Quindi un modo per migliorare il comportamento del metodo è aggiungere informazioni sulla derivata seconda. Si passa allora al secondo algoritmo, quello di Gauss-Newton. L'aggiornamento del passo si ottiene con

$$\theta_{i+1} = \theta_i - (\nabla^2 f(\theta_i))^{-1}\nabla f(\theta_i) \tag{3.43}$$

approssimando il gradiente con una serie di Taylor arrestata al secondo ordine e imponendo che esso sia nullo. Il vantaggio principale di questa tecnica è la rapidità di convergenza. Tuttavia, la convergenza può dipendere dal punto iniziale. Mentre la complessità computazionale dell'inversione della Hessiana (quando possibile) può essere ridotta notevolmente con l'approssimazione suggerita in precedenza.
L'innovazione introdotta da Levenberg [11] risiede proprio nell'aggiornamento del passo che sembra essere un compromesso tra i due metodi sopra esposti.

$$\theta_{i+1} = \theta_i - (\nabla^2 f(\theta_i) + \lambda I)^{-1} \nabla f(\theta_i) \tag{3.44}$$

Questa regola è utilizzato nel seguente modo:

1. effettua l'aggiornamento con la regola precedente

2. valuta i residui nel nuovo vettore di parametri

3. se i residui sono aumentati rispetto ai vecchi parametri allora ritorna nel passo precedente e aumenta λ di un fattore significativo (10 o più) in modo da seguire il gradiente. Ritorna al primo passo.

4. se i residui sono ridotti allora accetta il nuovo parametro poiché vuol dire che l'approssimazione quadratica sta funzionando e riduci λ di un fattore significativo (10 o più) per risentire meno del peso del gradiente.

Il contributo di Marquardt [12] è stato nel modificare leggermente la regola di aggiornamento in modo che incorporasse una correzione al problema della valle lunga e stretta. Infatti con la seguente

$$\theta_{i+1} = \theta_i - \left\{ \nabla^2 f(\theta_i) + \lambda diag\left[\nabla^2 f(\theta_i)\right] \right\}^{-1} \nabla f(\theta_i) \tag{3.45}$$

si possono effettuare passi lunghi con la curvatura bassa e viceversa per passi grandi. Infine il metodo di Levenberg-Marquardt è un'euristica [13], esso non è l'algoritmo ottimale ma si comporta molto bene in pratica.

3.5 La sequenza di Halton

L'algoritmo di Levenberg-Marquardt è iterativo e ha bisogno di un punto di partenza, che ovviamente inciderà sulla bontà della soluzione in presenza di tanti minimi locali. L'approccio che si vuol impiegare per la calibrazione richiede la copertura dello spazio dei parametri in modo uniforme, da poterlo scandagliare a sufficienza. Per la precisione, la distribuzione dei punti che verrà dato come input all'algoritmo di Levenberg-Marquardt deve essere equamente posizionata, in termini di distanza, nella regione di interesse dei parametri. Non si cerca quindi una distribuzione statistica uniforme poiché non soddisfa le esigenze. Esistono delle sequenze di numeri che rispetto le richieste, dette sequenze a bassa discrepanza.
La discrepanza è una misura della distribuzione equa dei punti nello spazio [10]. Aiutandosi con la figura 3.1 in cui sono mostrati 400 punti distribuiti uniformemente nel senso statistico, la discrepanza significa che la frazione di punti dentro il rettangolo Q deve corrispondere al volume del rettangolo. Si consideri un

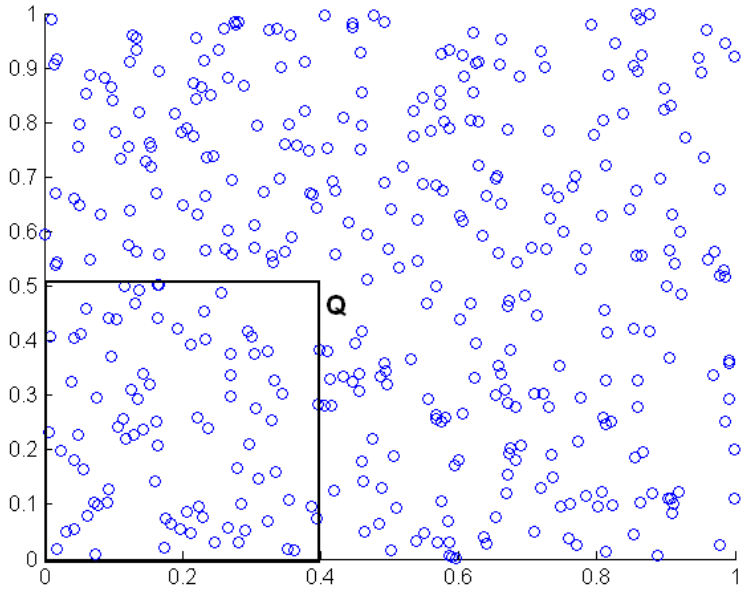

Figura 3.1: 400 punti distribuiti uniformemente in senso statistico.

rettangolo m-dimensionale $Q \subseteq [0,1]^m$ appartenente a un cubo m-dimensionale

immerso in $\mathbb{R}^m$. Siano $x_1,...,x_N \in [0,1]^m$ allora è possibile definire la discrepanza nella norma del sup, detta anche norma uniforme:

$$D_N = \sup_{Q\in\mathbb{R}^m} \left| \frac{\#x_i \in Q}{N} - vol(Q) \right|. \tag{3.46}$$

dove $vol(Q)$ è il volume del rettangolo considerato. Si dice che una sequenza di numeri $x_1,...,x_N \in \mathbb{R}^m$ è a bassa discrepanza se

$$D_N = O\left(\frac{(logN)^m}{N}\right) \tag{3.47}$$

ossia ha l'andamento asintotico del rapporto sopra descritto. Ovviamente per m fissato e non troppo grande si può dire che $D_N \approx O(N^{-1})$, segue un'iperbola.
Un esempio di sequenza molto famoso è quella di van der Corput

$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}, \frac{1}{16}, ... \tag{3.48}$$

Intuitivamente la sequenza si costruisce in questo modo: si prende un intervallo di lunghezza unitaria e si divide esattamente a metà, poi si dividono le metà e così via. Questo è solo un modo per vedere la sequenza, un altro punto di vista è dato dall'algoritmo che permette di replicare tale sequenza. Per esempio, si prende $x_7 = \frac{7}{8}$. L'indice 7 si può rappresentare in binario come

$$7 = (111)_2 = (d_2 d_1 d_0)_2 \, con \, d_i \in \{0,1\} \tag{3.49}$$

si rovescia il numero binario e si mette il punto decimale difronte

$$0.(d_0 d_1 d_2)_2 = \frac{d_0}{2} + \frac{d_1}{2} + \frac{d_2}{2} = \frac{7}{8} \tag{3.50}$$

Si generalizza questo procedimento con la funzione radicale-inversa

$$\phi_b(i) = \sum_{k=0}^{j} d_k b^{-k-1} \tag{3.51}$$

dove j dipende dalla lunghezza della rappresentazione in base b di i, mentre $d_i \in \{0,1,...,b-1\}$.
La sequenza di van der Corput può essere utilizzata in ambito multidimensionale

con alcuni accorgimenti. Si definisce così la sequenza di Halton, dati $p_1,..,p_m$ numeri primi il vettore m-dimensionale è ottenuto da

$$x_i = (\phi_{p_1}(i),...,\phi_{p_m}(i)),\ con\, i = 1,2,...N \tag{3.52}$$

dove $\phi_{p_j}(i)$ è la funzione radicale inversa. In figura 3.2 sono mostrati 400 punti

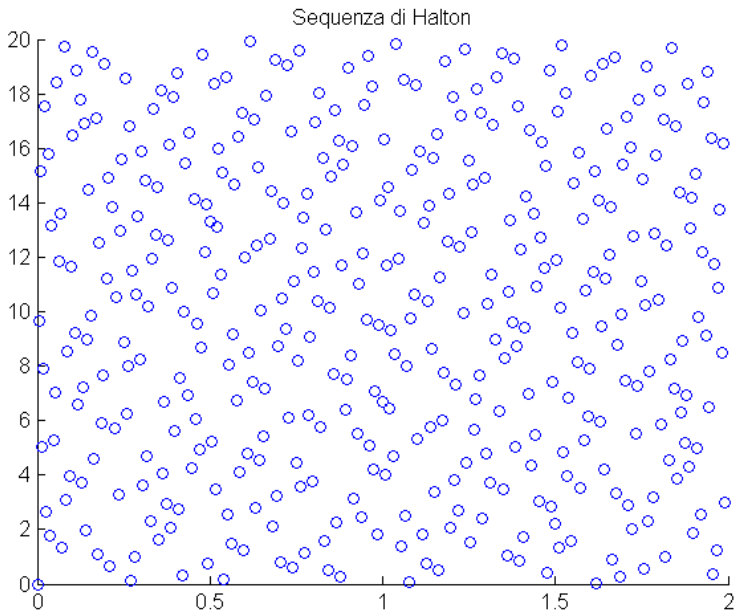

Figura 3.2: 400 punti della sequenza di Halton.

della sequenza di Halton con $p_1 = 2$ e $p_2 = 3$ per il primo e secondo parametro del modello di Heston e Bates. Si denota subito, ad occhio, la differenza rispetto alla figura 3.1 in cui ovviamente la discrepanza è maggiore poiché sono punti statisticamente uniformi. I punti coprono uniformemente lo spazio. Tuttavia, questo modo di generare numeri presenta dei problemi per dimensioni elevate. Infatti oltre il numero primo 17 si nota una correlazione tra i numeri della sequenza, coinvolgendo, per quel che riguarda la tesi, il modello di Bates in quanto possiede 8 parametri. Per evitare questo inconveniente spesso non si considerano i primi punti della sequenza di Halton. Un altro rimedio è quello della permutazione che può essere (pseudo) casuale oppure con un ordine preciso. Si è scelto la seconda soluzione perché ritenuta sufficientemente buona dalle prove effettuate [20].

Capitolo 4

Dettagli implementativi

L'idea con cui si vuol esplorare lo spazio dei parametri del problema di calibrazione richiede risorse computazionali notevoli. I punti di partenza possono essere tanti e se si vuol completare la ricerca dei parametri in tempo accettabile è necessario effettuare la ricerca dei minimi in parallelo. Ovviamente le ricerche sono indipendenti tra di loro. Quindi in questo contesto è ideale l'utilizzo di una macchina parallela come IDRA. Per poter sfruttare l'hardware parallelo è indispensabile un paradigma software adeguato. E' stato implementato il modello Master-Slave in cui ciascun processo del programma occupa un core della macchina parallela. La loro comunicazione avviene attraverso lo standard MPI. Il Master contiene la logica del programma, esso calcola i punti di partenza e li invia agli Slave insieme ad altre informazioni. Esso riceve la soluzione da ciascun Slave e stabilisce se esso deve ricercare un altro punto di minimo oppure terminare. Sono stati sviluppati tre metodi che inglobano il paradigma Master-Slave e si differenziano nella logica del Master. Sostanzialmente il secondo metodo vuol abbattere i tempi di esecuzione del primo, mentre il terzo cerca una soluzione che sia stabile in un'ottica di utilizzo dei parametri calibrati in giornate di mercato finanziario consecutive.

4.1 Architettura HW/SW

La strada scelta nella ricerca del minimo è spesso nota come metodo *brute force*, anche se non è obiettivo della calibrazione quello di trovare la soluzione migliore in assoluta ma solo l'approssimazione migliore. Le esigenze computazionali dei metodi che verranno presentati in seguito sono inaccettabili per una macchina sequenziale, in termini di tempo. Vengono sfruttate le architetture Hardware e Software parallele in modo da aver un grosso impatto sui tempi.

E' stato possibile accedere a IDRA [16] che è il cluster per calcolo intensivo del MOX. Essa dispone di 128 core disposti su 32 processori Intel Xeon Nehalem 5560 [17]. Ciascuno dei 16 nodi di IDRA contiene 8 core, ovviamente l'ampiezza della banda del canale di comunicazione tra i nodi è inferiore rispetto al bus di sistema del nodo. Quest'osservazione è indispensabile quando si implementa un programma con tanti processi che comunicano tra di loro, perché la scelta del numero di nodi da utilizzare può non essere immediata; bisogna valutare la quantità di informazioni che si passano i processi e successivamente scegliere i nodi da sfruttare. Quest'architettura è gestita da un sistema operativo Linux x86_64, a 64 bit, con versione del kernel 2.6.18-164.2.1.el5.
I programmi sono stati sviluppati in locale nel linguaggio C++ su l'ambiente Scientific Linux CERN 5 in modo da minimizzare l'incompatibilità con la macchina IDRA. C++ è un linguaggio orientato agli oggetti, molto diffuso, e con buone prestazioni in quanto linguaggio compilato. Molte responsabilità sono lasciate al programmatore grazie all'utilizzo dei puntatori, permettendo un accesso diretto all'indirizzo di memoria. (Però a volte si ritorcono contro nel famigerato errore di Segmentation Fault).
Per sfruttare l'architettura parallela è necessario pensare ad un paradigma di programmazione adatto. E' allora indispensabile una libreria che possa permettere la comunicazione tra i processi. Lo standard richiesto è noto come MPI, sta per Message Passing Interface, di cui esistono alcune implementazioni. In questo lavoro sono state impiegate le funzioni di MPICH2 [15] poiché l'unico presente per C++ su IDRA.

La necessita di operare con matrici e vettori ha richiesto alcune funzioni algebriche. Si è scelto di utilizzare le librerie di ALGLIB [14] poiché includevano anche altre funzioni impiegate nella calibrazione come: fft, lm (implementazione di Levenberg-Marquardt), cubicspline, ecc. Sia la libreria MPICH2 che ALGLIB sono open source e quindi accessibili a tutti.

4.2 Paradigma Master-Slave

Lo scopo della tesi è affrontare il problema di calibrazione ricercando tanti minimi a partire da un insieme di punti iniziali. Successivamente le soluzioni devono essere analizzate per poter determinare la soluzione migliore. Nasce così l'esigenza di due tipologie di processi, uno genera i punti iniziali e definisce qual è la soluzione finale e l'altro effettua la ricerca a partire dal punto dato. Praticamente si tratta di una classica applicazione del paradigma Master-Slave, spesso nota anche come Client/Server. Come si vede in figura 4.1 il Master invia i punti della sequenza di Halton allo Slave con la funzione *Send()* e riceve con la funzione *Recv()* il minimo computato. Ovviamente lo Slave impiega le stesse funzioni, prima riceve e poi invia. Si suppone un'architettura con $n+1$ processi e $2*n$ punti di partenza da analizzare. Nella prima fase dello sviluppo il Master invia a tutti gli n Slave i punti di partenza e li attende tutti prima di procedere. Quando il Master ha ricevuto n soluzione passa alla seconda fase ripetendo il procedimento di prima, invia n punti iniziali e si aspetta n soluzioni. Infine analizza le $2*n$ soluzioni e calcola la migliore. Da precisare che lo Slave ritorna sempre una soluzione.
Ma questo modo di effettuare la ricerca non fa pieno utilizzo dell'architettura parallela poiché tutti i processi devono aspettare sempre il processo più lento nella ricerca del minimo. Quindi nella seconda fase di sviluppo è stata modificata la gestione della comunicazione Master-Slave in modo che ogni processo potesse continuare con una nuova ricerca senza aspettare il processo lento. Così ogni processo Slave è sfruttato a pieno.

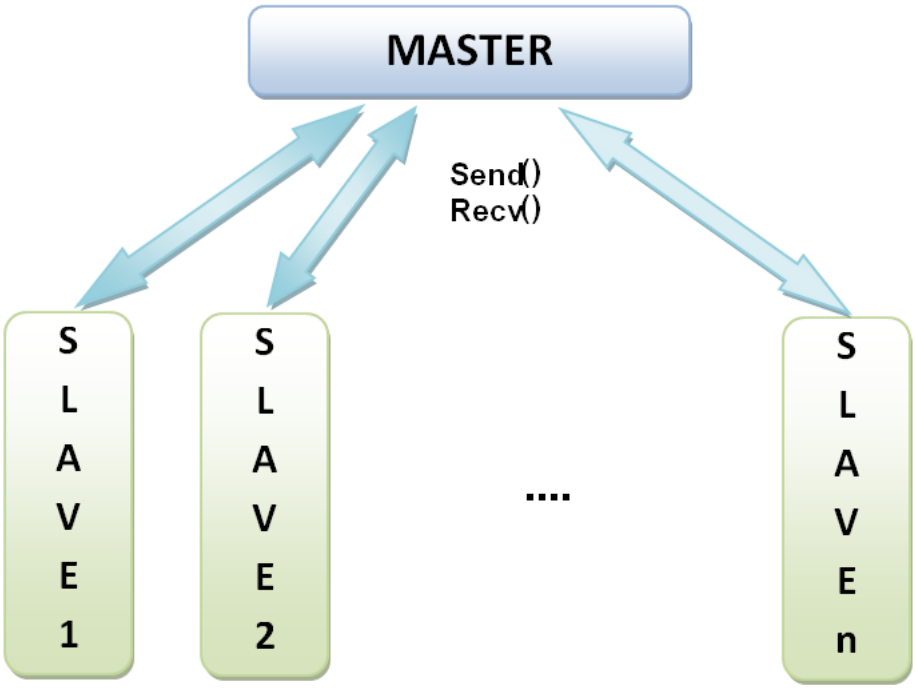

Figura 4.1: Paradigma Master-Slave.

4.3 I metodi implementati

Sono stati sviluppati 3 metodi in C++ per il paradigma descritto. Essi si distinguono fra di loro per il passo di discretizzazione e la logica con cui viene scelta la soluzione migliore. Il primo metodo viene chiamato *Fitta*, presenta un approccio diretto al paradigma Master-Slave; una volta fissato il passo di discretizzazione e il numero di punti di partenza, il Master invia i punti agli Slave e si aspetta le soluzioni. La soluzione migliore è quella per cui il valore della funzione obiettivo è minima.

Il secondo metodo si vuol distinguere dal primo in termini di tempi di esecuzione

poiché una ricerca con passo di discretizzazione piccolo può durare molto. Perciò questo approccio è stato definito *2Passi* ed opera in questo modo:

1. vengono fissati i passi di discretizzazione e i punti di partenza. Si lancia l'esecuzione del programma.

2. il Master invia i punti iniziali e il primo passo di discretizzazione agli Slave e aspetta le soluzioni.

3. il Master analizza tutte le soluzione ricevute e determina le migliori ordinando il corrispondenti vettore delle funzioni obiettivo in modo crescente.

4. invia un predeterminato sottoinsieme delle soluzioni agli Slave con un nuovo passo di discretizzazione.

5. il Master riceve le nuove soluzioni e stabilisce la migliore con il criterio del minimo valore della funzione obiettivo.

L'ultima euristica vuol fare un passo oltre in termini di stabilità della soluzione finale, infatti viene chiamato *Stabile*. Tralasciando la definizione formale di stabilità, si desidera che i parametri ottenuti dalla calibrazione odierna possano essere utilizzati il giorno successivo, eventualmente in due modi; o come parametri stimati della calibrazione oppure come punto di partenza di un'altra calibrazione. Si pensa che in questo modo i tempi della eventuale calibrazione saranno minimi. Va precisato che non esiste nessuna opera teorica a sostegno di quest'idea, è puramente empirica. Nella figura 4.2 è mostrato l'idea della euristica Stabile. Il bacino di sinistra della figura 4.2 è formato da tanti punti di arrivo dell'algoritmo di ricerca. Attraverso questa euristica si vuol scegliere tale bacino (cluster). Anche *Stabile* segue un logica a due fasi ma la seconda fase è differente da *2Passi*. Sostanzialmente cambia il terzo e il quarto punto del metodo precedente. Le soluzione alla fine della prima fase vengono analizzate in termini statistici; poiché le soluzioni sono tante si stabilisce il cluster migliore secondo una cifra di merito e la regione di punti del cluster viene ricoperta con la sequenza di Halton. I punti iniziali così formati vengono trasmessi agli Slave che effettuano la ricerca e ritornano le soluzioni trovate.

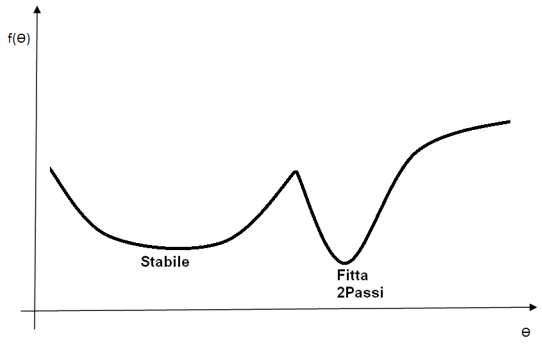

Figura 4.2: L'idea dell'euristica Stabile, profilo della funzione obiettivo.

La scelta del cluster migliore pone un paio di domande; come determinare il numero di cluster e come si valutano i cluster. La tecnica di clustering utilizzata è kmeans [18]. Essa rientra nel clustering partitivo e funziona in questo modo

1. si scelgono i centri dei cluster in modo casuale
2. ogni elemento viene assegnato al centro dei cluster più vicino secondo la distanza Euclidea
3. si sposta il centro alla media degli elementi assegnati ad esso
4. viene ripetuto il passo 2 e 3 fino alla convergenza.

La convergenza è raggiunta quando gli assegnamenti non cambiano significativamente il centro del cluster, cioè non si supera una certa soglia. Tipicamente l'algoritmo si ripete per eliminare l'eventuale partenza sbagliata dovuta alla casualità. Si è scelta questa tecnica perchú; si suppone che i punti di minimo possano formare dei cluster sferici, per le prestazioni in termini di tempo di esecuzione e per il buon comportamento in ambito multidimensionale. In figura 4.3 viene mostrato l'andamento della varianza campionaria generalizzata dei cluster rispetto a tutto l'insieme di dati, costituito dalle soluzione fornite dal metodo dopo la prima fase con Eurostoxx50. La varianza campionaria generalizzata è definita come il determinante della matrice di covarianza [18], rappresenta il volume incluso nelle osservazioni, d'ora in poi la dispersione. Seguendo la figura 4.3, il numero di

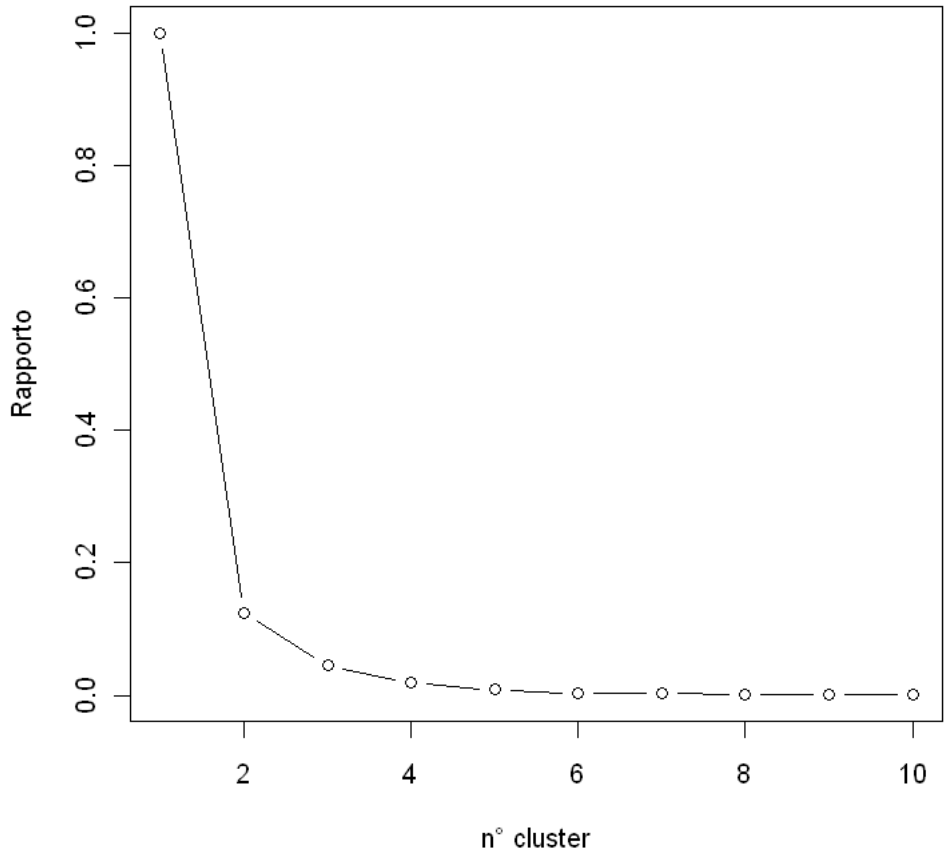

Figura 4.3: Andamento della varianza campionaria generalizzata all'aumentare del numero di cluster.

cluster ritenuti opportuni è il punto di gomito del grafico. Tale punto può essere individuato in corrispondenza del minimo della derivata prima della curva. Può succedere che la curva non presenti un gomito chiaro e quindi non sia evidente il numero di cluster da scegliere. Comunque queste non incide profondamente sulla soluzione, ma può aumentare i tempi di esecuzione.
La cifra di merito è costituita da tre componenti, il minimo in assoluto per il cluster *MinClu*, la differenza tra il minimo e la media dei minimi del cluster*MediaClu* e

la dispersione *DispClu*. La loro composizione fornisce la cifra di merito:

$$Cfr_i = \alpha * MinClu_i + \beta * | MediaClu_i - MinClu_i | + \gamma * | log(1/DispClu) | \; con \; i = 1..k \tag{4.1}$$

dove k il numero di cluster. I parametri α, β, γ non sono nient'altro che i pesi da assegnare a ciascun componente. L'equazione (4.1) associa un valore numerico ad ogni cluster e si definisce cluster migliore quello con la cifra di merito minima. Se i punti di minimo locale non dovessero formare dei cluster ben distinti allora la scelta del numero di cluster con la regola del gomito potrebbe non funzionare bene e considerare tanti cluster. Allora con la cifra di merito si vogliono escludere i cluster derivanti da picchi della funzione obiettivo. L'ultimo componente dovrebbe evitare la scelta di un cluster composto da un minimo isolato, o cluster molto piccoli oppure molto grandi. Il secondo componente cerca di scegliere i cluster che hanno la stessa funzione obiettivo, cioè il minimo piatto nella figura 4.2. Ovviamente si vuole sempre che la soluzione al problema di calibrazione fornisca la migliore approssimazione dei prezzi e questo è garantito dal primo componente, il minimo del cluster.

Capitolo 5

Risultati numerici

Nei capitoli precedenti sono stati illustrati i modelli ritenuti più opportuni dalle osservazioni empiriche e presentati i metodi implementati che li inglobano. Viste le esigenze di calcolo l'implementazione è stata fatta su architettura parallela. E' indispensabile fornire risultati numerici che confermano o smentiscano le attese e possibilmente diano nuovi spunti. A tale scopo sono stati considerati dei data set reali attraverso i quali analizzare il problema di calibrazione nei suoi modelli e nei metodi sviluppati. Sarà quindi necessario concentrarsi su alcune variabili di osservazione come: tempi di esecuzione, funzione obiettivo e differenza di prezzo.
La calibrazione di modelli multidimensionali è un problema time-consuming, molto più del semplice pricing, che può essere implementato su una macchina seriale. L'impostazione del problema effettuata nei capitoli precedenti, ricerca di tanti punti di minimo, suggerisce che ciascun punto può essere cercato in modo indipendente dagli altri. In questo modo la ricerca dei minimi può essere parallelizzata molto. L'architettura parallela hardware e software assicura molti vantaggi in termini computazionali che soddisfano pienamente le esigenze della calibrazione.
Una parte importante del capitolo è occupato dalla validazione perché è indispensabile fornire dei dati che possano dare conferma dell'attendibilità dei risulti e quindi delle conclusioni.

5.1 I data set

I metodi implementati sono stati testati su due differenti insiemi di dati; Eurostoxx50 e ISP IM Equity. Il primo è un indice di titoli dell'eurozona creato dalla Stoxx Limited con l'obiettivo di aver un indicatore rappresentante le 50 maggiori società dell'area. I dati risalgono al 15/02/2008 in cui $S_0 = 3771.1, r = 0.03936, d = 0.0412$ e si considera una Call Europea su tale indice con $T = 1$ e 31 strike comprese $3050 \leq K \leq 4550$.

L'altro insieme di dati è costituito da 20 prezzi di una Call Europea su un titolo azionario della Banca Intesa-San Paolo osservato in diverse date vicine fra loro, a partire dal 01/07/2011 fino al 14/07/2011. La motivazione di questa scelta sarà chiara in seguito, quando si vorranno validare le soluzioni della calibrazione. Come osservato in precedenza, la calibrazione dei modelli di Heston e Bates è

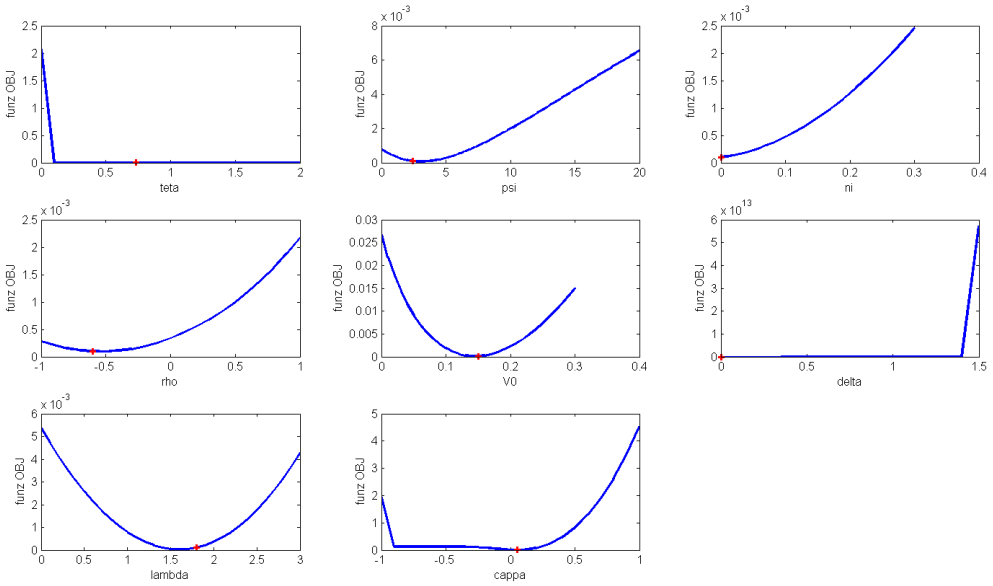

Figura 5.1: Modello di Bates: andamento della funzione obiettivo nell'intorno di una soluzione per ISP IM.

un problema inverso che può presentare molti minimi locali. I metodi sviluppati permettono di analizzare gran parte del dominio e trovare una conferma alla frase precedente. Infatti se si considera una soluzione fornita per il modello di Bates da

uno dei metodi e si osserva l'andamento della funzione obiettivo fissando tutti i parametri tranne uno, si nota che potrebbero esistere tanti minimi locali.

In figura 5.1 è evidenziato in rosso la soluzione trovata da *Fitta* per il data set ISP IM Equity in data 01/07/2011. Come si nota dalla figura, per *teta* e *delta* a grandi variazioni del parametro corrispondono piccole variazioni della funzione obbiettivo. E questo non è facilmente risolvibile dai metodi numerici di ricerca dei minimi, che può portare a grossi errori, introducendo quindi dei "falsi minimi".

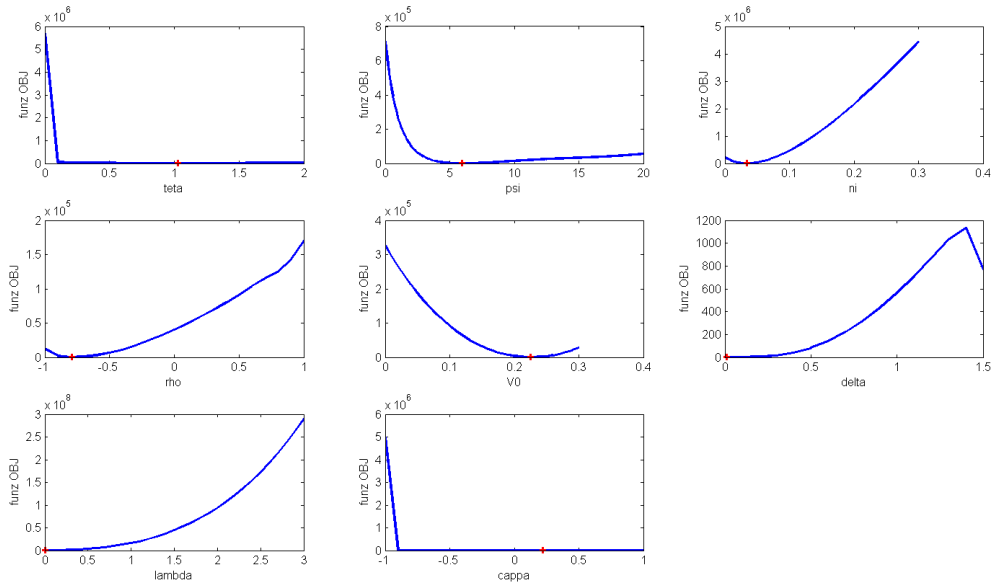

Figura 5.2: Modello di Bates: andamento della funzione obiettivo nell'intorno di una soluzione per Eurostoxx50.

Per il modello di Bates si nota un andamento simile al data set ISP IM anche per Eurostoxx50, mostrato in figura 5.2.
Può essere interessante confrontare le distanze tra i minimi locali per aver un'idea della differenza tra di essi, come mostrato nel grafico della figura 5.3. Fissando l'asse m a 10 e proseguendo verso destra, n cresce, si nota che i primi 17 minimi locali coincidono, la distanza è pressoché zero. In corrispondenza del 18-esimo e del 20-esimo minimo locale la distanza con il decimo sale a 1 per poi scendere di nuovo.

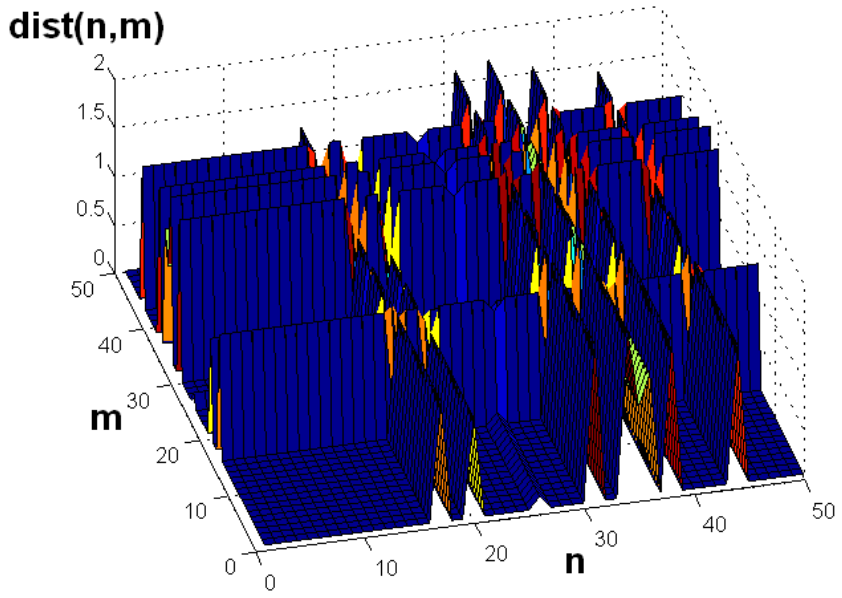

Figura 5.3: Distanza tra i minimi locali.

Tuttavia, per i data set considerati il modello di Heston evidenzia delle caratteristiche differenti da quello di Bates. Per Eurostoxx50 con il modello di Heston si nota un intorno di minimo unico, mostrato nel grafico in figura 5.4. Anche per l'altro data set la funzione obiettivo nel modello di Heston non è costante per nessun parametro.

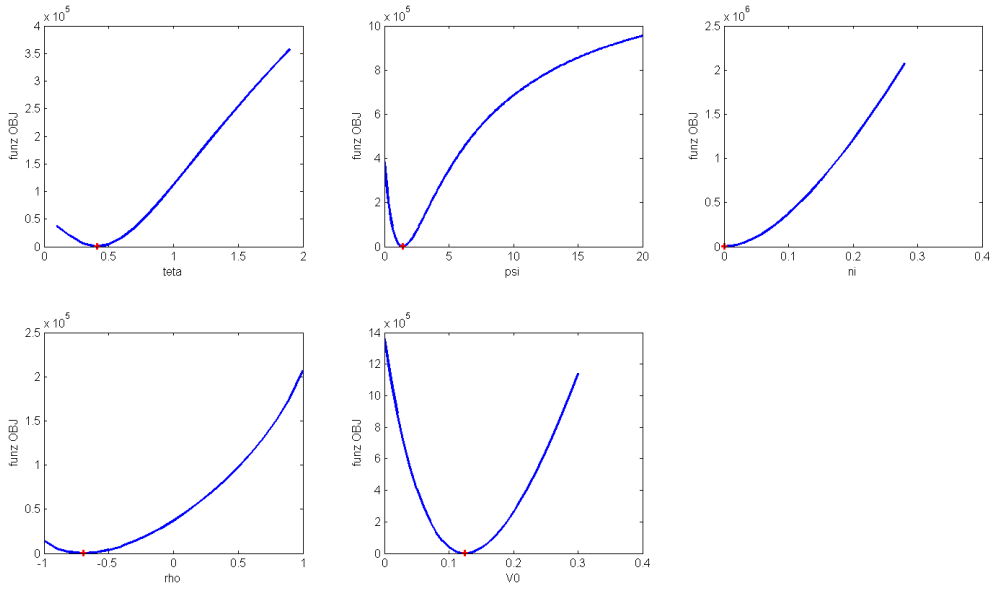

Figura 5.4: Modello di Heston: andamento della funzione obiettivo nell'intorno di una soluzione.

5.2 Confronto dei metodi

La prima distinzione tra i metodi è data dai tempi di esecuzione. Tipicamente in ambito informatico questa variabile presenta molte difficoltà nella misurazione. L'effettivo tempo di esecuzione è composto dal tempo del processore e il tempo di trasferimento dei dati dal processore alla memoria e viceversa. Se si considera che la macchina è parallela vanno aggiunte nuove problematiche e ciò si ripercuote sui tempi. L'ambiente di lavoro è una macchina remota che è collegata con una rete a IDRA. Quindi la valutazione dei tempi può subire le oscillazioni della rete che collega i computer. Inoltre il programma implementa il paradigma Master-Slave in cui ciascun Slave comunica con il Master in modo asincrono attraverso il bus di sistema. Anche se nei calcolatori moderni il bus di sistema è dotato di una banda molto elevata è comunque pensabile che ci sia una ripercussione dal trasferimento sull'incognita tempo di esecuzione.

La tabella 5.1 mostra l'andamento di alcune variabili del metodo *Fitta* al va-

	Heston			Bates		
DiffStep	TmpExec	FObj	LOO	TmpExec	FObj	LOO
1E-1	501.40	4.07227	-0.43454	4435.69	0.46019	-0.09006
1E-2	127.44	0.60685	-0.13949	197.39	0.07058	0.04569
1E-3	123.75	0.59961	-0.13411	953.23	0.07035	0.04631
1E-4	114.18	0.59961	-0.13411	1040.65	0.12913	-0.03441
1E-5	115.59	0.59961	-0.13411	1030.84	0.12913	-0.03441
1E-6	122.91	0.55926	-0.24983	1027.07	0.12913	-0.03441
1E-7	114.80	0.59961	-0.13411	1047.26	0.12913	-0.03441
1E-8	115.01	0.59961	-0.13411	739.42	0.12882	-0.03428
1E-9	115.92	0.59961	-0.13411	734.18	0.12882	-0.03428
1E-10	118.53	0.59961	-0.13411	339.34	0.12884	-0.03428

Tabella 5.1: Andamento di alcune variabili al variare del passo di discretizzazione per *Fitta* con i dati di Eurostoxx50

riare del passo di discretizzazione, da 10^{-1} fino a 10^{-10} considerando 240 punti di partenza e un nodo di IDRA(8 core). La prima colonna di Heston e Bates, TmpExec, denota il tempo di esecuzione in secondi per ogni passo di discretizzazione. Le colonne LOO mostrano le differenze di prezzo tra il prezzo di mercato e il prezzo di Heston e di Bates per il dato accantonato. Nella tabella 5.1 è stato accantonato sempre il quindicesimo dato. Si nota l'andamento decrescente all'inizio, coerente con le attese, e successivamente un valore costante dovuto al troncamento delle cifre decimali per problemi di spazio nella tabella. La seconda colonna di Heston e di Bates, FObj, rappresenta la funzione obiettivo.
Per Heston il tempo è decrescente per i primi passi per poi risalire leggermente. Il tempo di esecuzione per il modello di Bates si presenta oscillante. Si sono però riscontrati dei problemi sulla misurazione dei tempi con IDRA, quindi questi dati potrebbero essere affetti da errori di misurazione. Comunque sembra di poter affermare che, usare un passo troppo piccolo, possa essere svantaggioso, sia in termini di tempi, sia di accuratezza, probabilmente a causa di falsi minimi o minimi

locali determinati dalla procedura numerica.

Heston segue un trend più regolare rispetto ai dati riportati per Bates. Si ricordi quanto detto nelle Figure 5.2 e 5.4 sui problemi di Bates rispetto ad Heston. Quindi si può ipotizzare che un passo troppo piccolo per Bates non sia una scelta opportuna a causa di zone del dominio in cui a grandi variazioni dei parametri corrispondono piccoli variazioni della funzione obiettivo. Si nota che il comportamento di Bates si ripete anche calibrando il modello con leave-one-out, ossia dando in input tutti i dati tranne uno e testare il prezzo con il dato lasciato fuori. Il passo di discretizzazione opportuno per Fitta sembra essere 1E-3, un compromesso tra l'accuratezza della funzione obiettivo e i tempi di esecuzione.

Come descritto in precedenza i metodi 2Passi e Stabile sono stati sviluppati per poter ridurre i tempi di esecuzione di Fitta. Si sono incontrati alcuni problemi nella misurazione di tale variabile sulla macchina parallela e i tempi ottenuti su essa nell'ultimo periodo di lavoro sono ritenuti poco affidabili. Quindi si è preferito confrontare i tempi di esecuzione dei metodi su un PC; con processore AMD Athlon a 64 bit, frequenza 3200 Mhz e 3GB di RAM. Essi vengono mostrati nella tabella 5.2, fissando il passo di discretizzazione a 1E-10 per Fitta e come secondo passo per 2Passi e Stabile. Mentre il primo passo per quest'ultimi si fissa a 1E-2. E' evidente la riduzione dei tempi per Bates, 2Passi e Stabile ottengono più rapidamente il minimo. Invece per Heston, all'aumentare dei punti di partenza (colonna Nr Pti) la differenza dei tempi si assottiglia rendendo l'uso di Stabile e 2Passi meno vantaggioso. Questa conclusione ha suggerito di verificare i tempi di esecuzione su un altro data set mantenendo i passi discretizzazione. I tempi ottenuti con ISP IM Equity mostrano risultati comparabili con 5.2.
E' naturale chiedersi quanto abbia contribuito la macchina parallela nella riduzione dei tempi di esecuzione. Viene confrontato il metodo Fitta con passo di discretizzazione 1E-4, suggerito da un analisi fatta sul data set ISP IM Equity identica alla tabella 5.1. Si alloca un nodo di IDRA con 8 core e si creano 8 processi, uno per il master e gli altri 7 per gli Slave. Mentre è PC è dotato di un core. Allora ci si aspetta che lo *speedup* sia determinato dal numero di Slave se il Master occupi

	HESTON			BATES		
Nr Pti	Fitta	2Passi	Stabile	Fitta	2Passi	Stabile
10	143.58	104.02	65.35	380.82	270.84	277.01
20	909.07	536.92	535.3	630.77	460.49	474.56
30	1272.22	926.71	903.67	675.48	544.45	414.22
40	1334.79	1334.89	1330.78	1428.5	573.13	430.18
50	1434.7	1493.4	1387.44	1528.03	602.03	434.85
60	1626.14	1577.33	1500.99	1873.3	650.74	569.06

Tabella 5.2: Tempi di esecuzione in secondi per Eurostoxx50 alla data 01072011.

il processore per un tempo molto basso. Infatti, come si vede nella tabella 5.3, il rapporto tra i tempi del PC con IDRA si aggirano intorno a 7 all'aumentare del numero di punti iniziali. Si nota che gli ultimi tempi del PC nella tabella superano di 7 volte quelli di IDRA, è probabile che sia dovuto alla gestione dei processi del sistema operativo; è pensabile che IDRA non abbia allocato i core con processi in background mentre è naturale che ciò avvenga in un PC. E' chiaro che i tempi del PC possono essere aumentati a causa di eventuali variabili come processi in background, collegamento a internet ed eventuale aggiornamento automatico oppure semplicemente il movimento del mouse.

Nr Pti	IDRA	PC
40	37.39	230.89
80	70.91	480.23
160	127.70	871.61
240	180.34	1555.97
480	327.50	2333.08
800	557.38	4956.18

Tabella 5.3: Confronto dei tempi di esecuzione tra un PC e IDRA con Fitta e il modello di Heston sul data set ISP IM Equity alla data 01072011.

5.3 Validazione

Per poter apprezzare la bontà dei modelli implementati e quindi conoscere l'entità dell'errore ottenuto da essi è necessario valutarli con una tecnica di validazione. Un metodo utilizzato in statistica e machine learning [22] è Leave-One-Out, esso permette di misurare l'errore che commette il modello considerato iterando il processo di calibrazione sull'insieme dei dati modificato. Per ciascuna iterazione si accantona un dato e si calibra il modello con i dati restanti. Si calcola l'errore che commette il modello come differenza tra il prezzo del dato lasciato fuori dal data set per la calibrazione e quello di mercato. Il procedimento si ripete per tutto l'insieme di dati e alla fine la media degli errori costituisce una stima finale dell'errore. In Tabella 5.4 sono esposti i risultati di Leave-One-Out per il data set ISP IM Equity La stima dell'errore con leave-one-out per il modello di Heston è $3.6938e-004$ mentre per Bates è $-6.3264e-006$. Questi valori sono stati ottenuti effettuando la media campionaria della differenza tra i prezzi di mercato (Prz Mkt) e i prezzi di Heston (Prz Heston) della tabella 5.4.

Siccome il problema di calibrazione può presentare tanti minimi locali è utile affrontare il punto di minimo con gli indicatori statistici classici. Nella tabella 5.5 vengono esposti la media e la varianza di ciascun parametro del modello di Heston. Sono stati considerati 240 punti iniziali e il passo di discretizzazione è stato variato tra 1E-1 e 1E-10. Gli indicatori statistici sono calcolati su un insieme di 20 campioni. I risultati confermano le attese della figura 5.4. La variabilità dei parametri è molto bassa perché il minimo osservato in figura 5.4 è unico. Da precisare che esso è stato ottenuto con tutti i dati. La media delle soluzioni nella tabella 5.5 si discosta dai punti rossi. Questo comportamento è imputabile alla dipendenza sensibile del modello dai dati di input poiché la media e la varianza sono state calcolate sulle soluzioni fornite da leave-one-out. Si raccolgono risultati analoghi anche per Bates nella tabella 5.6. Sono stati considerati 42 campioni di soluzioni partendo da 240 punti iniziali e con passo di discretizzazione simile a Heston. Chiaramente la variabilità sarà maggiore nei parametri che presentavano un andamento costante nella figura 5.2, vedi θ,ξ e κ.

Iterazione	Prz Mkt	Prz Heston	Prz Bates	FOBJ Heston	FOBJ Bates
1	0.6718	0.6671	0.6714	5.6216e-05	1.8856e-06
2	0.6226	0.6188	0.6227	4.2707e-05	1.8616e-06
3	0.526	0.5234	0.5260	4.9840e-05	1.8850e-06
4	0.4781	0.4764	0.4783	5.3643e-05	1.8509e-06
5	0.431	0.4301	0.4311	5.5579-05	1.8564e-06
6	0.3851	0.3847	0.3848	5.6101e-05	1.8497e-06
7	0.3395	0.3405	0.3399	5.5284e-05	1.7402e-06
8	0.2967	0.2974	0.2962	5.5707e-05	1.6836e-06
9	0.2542	0.2562	0.2547	5.2515e-05	1.6649e-06
10	0.2157	0.2167	0.2152	5.5187e-05	1.6803e-06
11	0.1785	0.1799	0.1787	5.4377e-05	1.8354e-06
12	0.1456	0.1458	0.1452	5.6166e-05	1.7702e-06
13	0.1157	0.1151	0.1153	5.5971e-05	1.7840e-06
14	0.0886	0.0885	0.0895	5.6214e-05	1.1563e-06
15	0.0675	0.0655	0.0671	5.3422e-05	1.7861e-06
16	0.0345	0.0334	0.0347	5.5438e-05	1.8425e-06
17	0.016	0.0157	0.0154	5.6179e-05	1.7049e-06
18	0.0061	0.0075	0.0063	5.4655e-05	1.8524e-06
19	0.0022	0.0034	0.0022	5.4948-05	1.8830e-06
20	0.0007	0.0015	0.0007	5.5560e-05	1.8851e-06

Tabella 5.4: Leave-One-Out per il data set ISP IM Equity in data 01072011.

	θ	ξ	η	ρ	V_0
Media	0.4011	1.2784	0.0004	-0.6988	0.1188
Varianza	0.0012	0.1912	1.66E-06	2.01E-06	0.0003

Tabella 5.5: Media e varianza dei parametri di Heston ottenuti sul data set Eurostoxx50

	θ	ξ	η	ρ	V_0	δ	κ	λ
Media	0.867	4.889	0.020	-0.740	0.199	0.094	-0.121	0.180
Varianza	0.131	1.406	0.001	0.012	0.007	0.033	0.150	0.043

Tabella 5.6: Media e varianza dei parametri di Bates ottenuti sul data set Eurostoxx50

Un altro modo per validare i modelli è chiedersi se i parametri ottenuti in una determinata data rappresentano bene il fenomeno anche nei giorni successivi alla data di calibrazione. Sostanzialmente ci si pone la domanda su quanto siano stabili i parametri ricavati dai metodi. La stabilità intesa in senso formale si esprime chiedendo che per piccole oscillazioni in input al problema di calibrazione ci siano variazioni ridotte dei parametri. Per come è stato descritto il problema e per la sua natura, ossia la dipendenza sensibile dai dati, è difficile quantificare l'entità di queste variazioni dal punto di vista teorico. Allora si può affrontare la questione con dati di mercato per vedere il comportamento della calibrazione.
Quindi si vogliono valutare le soluzioni del problema di calibrazione con i dati delle giornate successive, che ovviamente contengono delle oscillazioni rispetto alla giornata fissata. Per fare ciò è necessario disporre dei dati di mercato della Call Europea nelle date di interesse. Nella tabella 5.7 sono mostrati i prezzi osservati nel mercato che verranno messi a confronto con i prezzi forniti dal modello di Heston e quello di Bates.

Nella tabella 5.8 sono contenuti i prezzi di una Call Europea con Heston su ISP IM Equity utilizzando i parametri della calibrazione alla data 01072011. Nella figura 5.5 sono rappresentate le curve delle differenze tra i prezzi di mercato e i prezzi ottenuti dalla calibrazione di Heston. Come ci si può aspettare, per date vicine alla data di calibrazione, i parametri rappresentano bene l'evoluzione dei prezzi. Si nota che le curve dei punti rossi, blu e magenta sono a tratti sovrapposte. Mentre per le date 06072011 e 14072011 l'errore nei prezzi è notevole. Si può pensare di sfruttare la calibrazione del 01072011 per effettuare un'unica calibrazione nelle giornate successive dando come punto di partenza la soluzione

	Prezzo osservato in data				
Indice	01072011	04072011	05072011	06072011	14072011
1	0.6718	0.6599	0.6206	0.5423	0.3832
2	0.6226	0.6104	0.5713	0.494	0.339
3	0.5260	0.513	0.4747	0.3998	0.2554
4	0.4781	0.4647	0.4267	0.3538	0.2172
5	0.4310	0.4172	0.3802	0.3096	0.1813
6	0.3851	0.3709	0.3343	0.2667	0.1485
7	0.3395	0.3248	0.29	0.2264	0.12
8	0.2967	0.2816	0.2476	0.188	0.0936
9	0.2542	0.2388	0.2076	0.1537	0.073
10	0.2157	0.2001	0.1696	0.1218	0.0549
11	0.1785	0.1633	0.1368	0.0947	0.04
12	0.1456	0.1304	0.1068	0.0719	0.0292
13	0.1157	0.1019	0.0805	0.0523	0.0204
14	0.0886	0.077	0.0596	0.037	0.014
15	0.0675	0.0563	0.0426	0.0258	0.0097
16	0.0345	0.0282	0.0193	0.011	0.0042
17	0.0160	0.0125	0.0082	0.0044	0.0017
18	0.0061	0.0052	0.0029	0.0016	0.0007
19	0.0022	0.0021	0.001	0.0005	0.0002
20	0.0007	0.0008	0.0003	0.0001	0.0001

Tabella 5.7: Prezzi di una Call Europea su ISP IM Equity osservate in 5 date diverse

	Prezzo previsto in data				
Indice	01072011	04072011	05072011	06072011	14072011
1	0.66719	0.653935	0.61646	0.534963	0.365139
2	0.618968	0.60558	0.568183	0.487026	0.318032
3	0.523543	0.509852	0.472765	0.392801	0.227422
4	0.476556	0.462703	0.425886	0.346898	0.184909
5	0.430241	0.416228	0.379794	0.302143	0.145164
6	0.384782	0.370621	0.334718	0.258876	0.10908
7	0.340399	0.32612	0.290941	0.217512	0.0776779
8	0.297362	0.283015	0.24881	0.178546	0.0519492
9	0.255986	0.241656	0.208735	0.142561	0.0325086
10	0.216644	0.202452	0.171201	0.110193	0.0191635
11	0.179761	0.165878	0.136751	0.0820805	0.0108213
12	0.145804	0.132453	0.10596	0.0587364	0.00596763
13	0.115255	0.102713	0.0793666	0.0403794	0.0032624
14	0.0885617	0.0771323	0.0573575	0.0267854	0.0017842
15	0.066052	0.0560232	0.0400314	0.0172863	0.0009808
16	0.0337421	0.0269752	0.0179264	0.0068883	0.0003035
17	0.0158427	0.0119611	0.0075527	0.0027127	9.7424e-05
18	0.0071674	0.0051512	0.0031480	0.0010858	3.2464e-05
19	0.0032352	0.0022277	0.0013302	0.0004458	1.1222e-05
20	0.0014816	0.0009811	0.0005749	0.0001881	4.0180e-06

Tabella 5.8: Prezzi di una Call Europea con Heston su ISP IM Equity con i parametri calibrati sui prezzi del 01072011

precedente. Le prove effettuate vengono mostrate in tabella 5.9 in cui gli acronimi CI e CR indicano, Calibrazione Intera e Calibrazione Ridotta rispettivamente; ossia calibrazione con 240 punti di partenza e calibrazione con 1 punto di partenza. E' evidente il risparmio di tempo che si ha per questo procedimento, mentre

	Tmp CI	Tmp CR	Diff CI	Diff CR	FObj CI	FObj CR
Modello	Calibrazione in data 04072011					
Heston	344.62	6.83	-0.002478	-0.002478	8.550e-05	8.550e-05
Bates	2667.27	11.56	0.000215	0.000206	1.334e-06	1.353e-06
Modello	Calibrazione in data 05072011					
Heston	355.11	3.98	-0.000287	-0.000287	4.410e-05	4.410e-05
Bates	1651.96	9.1	0.000184	0.000184	1.142e-06	1.142e-06
Modello	Calibrazione in data 06072011					
Heston	438.52	7.46	0.001030	0.001030	5.855e-05	5.855e-05
Bates	2386.39	136.59	-0.0001009	-0.0001009	6.252e-07	6.2529e-07
Modello	Calibrazione in data 14072011					
Heston	589.53	4.78	0.002141	0.002141	0.000431	0.000431
Bates	2676.33	45.07	-1.518e-06	-1.519e-06	1.106e-06	1.106e-06

Tabella 5.9: Confronto di alcune variabili su date successive per ISP IM Equity per entrambi i modelli

non si verifica nessuna differenza significativa in termini di prezzo(Diff CI e Diff CR) tra essi. Si espongono anche i dati dell'andamento della funzione obiettivo nei due casi (FObj CI e FObj CR), notando un comportamento simile ai prezzi. Un'altra conferma della bontà di questo ragionamento data dalle tabelle 5.10 per Bates e 5.11 per Heston. Nella prima colonna della tabella 5.10 viene rappresentato il giorno di osservazione e la modalità di calibrazione, cioè 04-CI indica la data 04072011 e CI che la calibrazione è stata effettuata con 240 punti iniziali e passo di discretizzazione 1E-4. Allo stesso modo vengono fissati i parametri del modello di Heston. I risultati sono mostrati nella tabella 5.11.

	θ	ξ	η	ρ	V_0	δ	κ	λ
04-CI	0.5378	0	0	-0.6622	0.1041	0.0734	0.0095	3
04-CR	0.5756	0	0	-0.5919	0.1157	0	0.1907	0.1496
05-CI	0.6836	0	0	-0.5441	0.1253	0	0.0977	0.2335
05-CR	0.6853	0	1.5e-11	-0.5415	0.1254	0	0.0936	0.2436
06-CI	1.6944	20	0.1323	-0.5780	0.1967	0	0.0968	0.3882
06-CR	1.6944	20	0.1323	-0.5780	0.1967	0	0.0968	0.3881
14-CI	2	10.835	0.3	-0.5529	0.2851	0	0.0791	0.8716
14-CR	2	10.835	0.3	-0.5529	0.2851	0	0.0791	0.8716

Tabella 5.10: Confronto delle soluzioni su date successive dell'ISP IM Equity per il modello di Bates

	θ	ξ	η	ρ	V_0
04072011-CI	2	4.1433093	0.3	-0.4554531	0.1053855
04072011-CR	2	4.1439268	0.3	-0.4554780	0.1053773
05072011-CI	2	6.3680201	0.3	-0.4792057	0.0815591
05072011-CR	2	6.3682023	0.3	-0.4792120	0.0815567
06072011-CI	2	3.0500734	0.3	-0.5347261	0.1757867
06072011-CR	2	3.0502046	0.3	-0.5347308	0.1757856
14072011-CI	0.7695502	0	0.3	-1	0.3
14072011-CR	0.7695492	0	0.3	-1	0.3

Tabella 5.11: Confronto delle soluzioni su date successive dell'ISP IM Equity per il modello di Heston

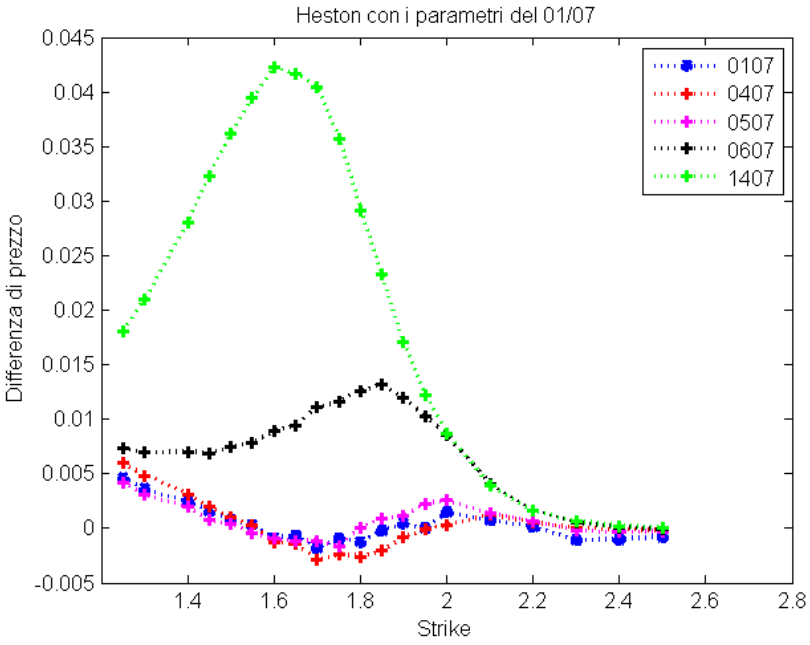

Figura 5.5: Differenza tra i prezzi della tabella 5.7 e la 5.8 alla stessa data.

Per completezza viene mostrato anche la differenza tra i prezzi di mercato e quelli forniti dal modello di Bates alla figura 5.6. Si denota un errore inferiore rispetto al modello di Heston nelle prime tre giornate.

5.4 Confronto dei modelli

Il modello di Bates si distingue da quello di Heston unicamente per i salti. Essi richiedono 3 dimensioni in più e ciò si ripercuote nettamente nei tempi di esecuzione della calibrazione, vedi tabella 5.1. I tempi sono un indice del tasso di convergenza. Siccome lo spazio di ricerca dei parametri di Bates è maggiore esso richiede più iterazioni rispetto a Heston. Però, allo stesso tempo, più parametri gli forniscono maggiore flessibilità ottenendo una funzione obiettivo più bassa.
Nonostante il modello di Heston abbia andamento più regolare per i parametri poiché diversamente dai parametri di Bates non presenta andamento costante in nessuno di essi, entrambi i modelli mostrano una dipendenza sensibile dai punti

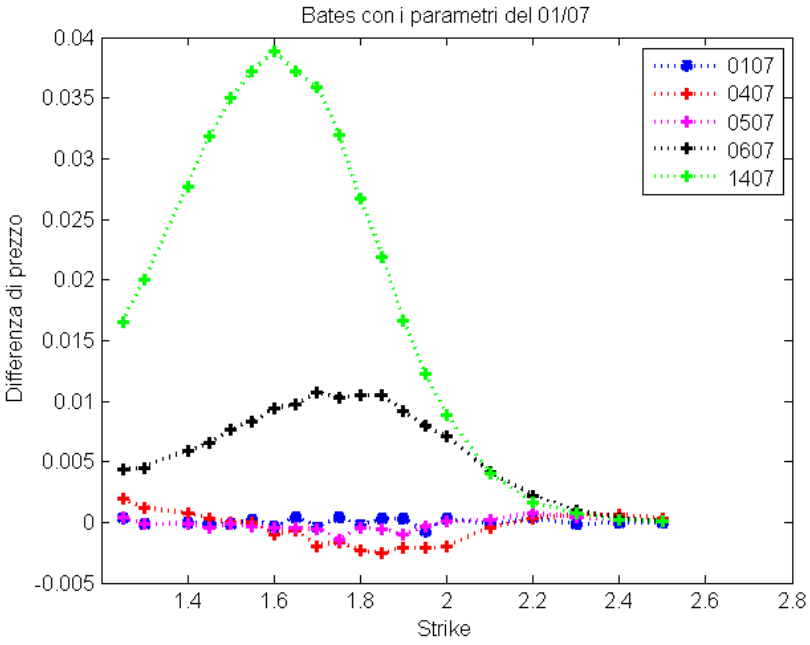

Figura 5.6: Differenza tra i prezzi della tabella 5.7 e quelli ottenuti con il modello di Bates alla stessa data.

di partenza dell'algoritmo deducibile dalla varianza delle soluzioni mostrate nelle tabelle 5.5 e 5.6. La perturbazione considerata è minima in quanto riguarda lo scarto di un elemento dall'insieme di dati e non l'aggiunta di un disturbo a tutti i dati. Si nota che per alcuni punti di partenza e per entrambi i modelli, il valore della funzione obiettivo è molto elevato. Questo può essere interpretato come l'assenza di una soluzione nell'intorno del punto considerato.
Tuttavia i modelli rappresentano bene il fenomeno sottostante poiché mostrano un buon comportamento anche con i dati delle giornata successive.

Capitolo 6

Conclusioni

In questa tesi si è studiato il problema di calibrazione di processi a volatilità stocastica per prezzare derivati finanziari. Essendo il problema time-consuming si è mostrato come un'architettura parallela e un metodi di ricerca dei minimi in cui i punti di partenza si ottengono da una sequenza di Halton, siano in grado di rendere più veloce la risoluzione del problema.

Il problema di calibrazione presentato non soddisfa le condizioni di Hadamard e i risultati numerici mostrano che si presentato tutte le difficoltà di un problema mal posto. Tuttavia è stato possibile ottenere delle soluzioni e validarle sia con la tecnica leave-one-out che testando le soluzioni su date successive. Il secondo metodo suggerisce un possibile impiego importante in finanza in quanto abbatte i tempi della calibrazione.
Nel lavoro sono stati considerati due modelli in cui la volatilità è una variabile non osservabile nel mercato. Il modello di Heston evidenzia un andamento più regolare e tempi minori rispetto a Bates ma la complessità di quest'ultimo fornisce prezzi migliori. La scelta del modello più opportuno può essere guidata dalle osservazioni empiriche.
Sulla base dei risultati e dei grafici esposti nel quinto capitolo si possono attendere andamenti simili a Eurostoxx50 e a ISP IM Equity, per Heston e Bates, anche in altri data set. Quindi si può affermare che tali modelli possono presentare un mi-

nimo (Heston) o tanti minimi locali (Bates), ma che sono sensibili ai dati in input e ciò si ripercuote nella scelta del passo di discretizzazione e sulle soluzioni. Si ottengono quindi tante soluzioni che possono generare un'elevata variabilità nell'intorno del punto di minimo.

Bibliografia

[1] Heinz W. Engl,Martin Hanke,Andreas Neubauer
Regularization of inverse problems
Kluwer Academic Publishers 2000

[2] Vittorino Pata
Appunti del Corso di Analisi reale e funzionale
http://www.mate.polimi.it/utenti/Pata

[3] Tomas Bjork
Arbitrage Theory in Continuous Time, Second Edition
Oxford Finance 2004

[4] Baldi Paolo
Equazioni differenziali stocastiche e applicazioni
Pitagora 2000

[5] Giovanni Maienza
Option pricing con il modello di Heston
Tesi di Laurea 2007

[6] Rama Cont, Peter Tankov
Financial modelling with jump processes
Chapman & Hall/CRC Financial Mathematics Series 2004e

[7] Steven L. Heston
A closed-form solution for options with stochastic volatility with applica-

tions to bond and currency options
The Review of Financial Studies 1993

[8] Peter Carr, Dilip B. Madan
Option valuation using the Fast Fourier Transform
The Journal of Computational Finance 1999

[9] David S. Bates
Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in The-phlx Deutschemark Options
The Review of Financial Studies 1996

[10] Rudiger Seydel
Tools for computational finance, Third Edition
Springer 2006

[11] K. Levenberg
A method for the solution of certain problems in least-squares
Quarterly of Applied Mathematics 1944

[12] D. Marquardt
An algorithm for least-squares estimation of nonlinear parameters
SIAM Journal 1963

[13] J. Nocedal , S.J. Wright
Numerical Optimization
Springer 1999

[14] Libreria Alglib
http://www.alglib.net/

[15] MPICH2
http://www.mcs.anl.gov/research/projects/mpich2/

[16] IDRA: un cluster per applicazioni parallele
http://mox.polimi.it/it/progetti/unix/idra.php

[17] Intel Xeon Nehalem 5560
http://ark.intel.com/products/37109

[18] Richard A. Johnson, Dean W. Wichern
Applied Multivariate Statistical Analysis, Sixth Edition
Pearson International Edition 2007

[19] Hansjorg Albrecher
The liltle Heston trap
Wilmott Magazine 2007

[20] Bart Vandewoestyne,Ronald Cools
On the Halton sequence and its scramblings
KULeuven 2005

[21] Barucci, Marsala, Nencini, Sgarra
Ingegneria finanziaria - Un'introduzione quantitativa
Egea 2009

[22] I. Witten, E. Frank
Data Mining: Practical Machine Learning Tools and Techniques
Morgan Kaufmann 2005

Appendice A

Codici

Elenco delle funzioni ausiliari implementate per tutti i metodi

```
//Restituisce il valore della funzione caratteristica di Heston
complex<double> heston(const double parametri[], complex<double> u,
 double t, double x0, double r, double q);
//Restituisce il valore della funzione caratteristica di Bates
complex<double> bates(const double parametri[], complex<double> u,
 double t, double x0, double r, double q);
//Restituisce il valore della funzione caratteristica del MBG
complex<double> gbm(complex<double> u,double t,double x0,double r,
 double vol, double q, double alpha);
//Restituisce il valore della funzione di Kronecker
double kronecker(double num);
//Legge da un file di input
int letturaFileInput(double *strike,double *r,double *q,double *T,
 double *S0, double *vol);
//Restituisce i prezzi di Black-Scholes
double blackScholes(double S0, double k, double t, double r,
double dividendi, double sigma);
//Fornisce i prezzi metodo di Carr-Madan
double* pricingCarrMadan(const alglib::real_1d_array &par, double s,
```

```
 double strike[], double t, double r, double q);

//Caricamento dei dati originali
void caricaDati(double x0,double strk[],double t,double r,
double div,double prz[])

//Valorizzazione dei dati previo controllo di LOO
void valorizzazioneLOO(int loo)

//Estrazione di un numero casuale nell'intorno di ATM
int estraiLOO()

//Distanza tra i prezzi di B&S e Heston
void distanza(const alglib::real_1d_array &parametri,
 alglib::real_1d_array &funzione, void *ptr)

//Numero di cluster ritenuti corretti per le soluzioni trovate
int numeroDiCluster(alglib::real_2d_array array2d, int size)

//Rappresenta la cifra decimale num in una base
definita dal numero primo
string toBase(int primo, int num)

//Permuta la sequenza delle cifra dopo la virgola
double permuta(double valore)

//Calcola la sequenza di Halton
double haltonSequence(int primo, int num)

/*Permette di scegliere il cluster ritenuto migliore secondo
una cifra di merito
```

```
int qualeCluster(alglib::real_2d_array array2d,int k,
 alglib::integer_1d_array xyc, double funzObj[])

//Imposta il numero di dati letti in input
void setNumDati(int n);
//Restituisce il numero di dati letti in input
int getNumDati();
//Imposta il numero accantonato
void setNumLOO(int n);
//Restituisce il numero accantonato
int getNumLOO();
//Ordina la matrice secondo il valore della funzione obiettivo
alglib::real_2d_array sortMatrixByLastCol(alglib::real_2d_array m)
```

In seguito viene esposto il codice della funzione *main* del metodo Stabile. Gli altri metodi cambiano leggermente la parte della seconda fase. Nel 2Passi si chiama la funzione *sortMatrixByLastCol()* che ordina le soluzioni in base alla funzione obiettivo e restituisce le migliori mentre per Fitta tale parte del codice termina l'esecuzione.

```
int main(int argc, char *argv[])
{
int rank, size;
char inmsg, outmsg = 'c';
MPI_Status status;
int dest, source, tag = 1;
//Sincronizzazione iniziale dei processi
MPI_Init(&argc, &argv);
//Progressivo che identifica ogni processo
rank = MPI::COMM_WORLD.Get_rank();
//Totale di processi nel gruppo
     size = MPI::COMM_WORLD.Get_size();
//Vettore delle richieste in attesa
```

```
MPI_Request arrayReq[size-1];

double diffstep = DIFFSTEP1;
int loo=0, contaLOO=0, numLOO=0;

//Processo in esecuzione
processo = rank;
//Avvia la sequenza casuale
srand(time(NULL)+clock());

//Operazione che svolge il Master
if(rank == MASTER){
cout << "MASTER!" << endl;
time_t start, end;
//Avvia il tempo
start = clock();
//Variabili locali
double strike[NumMaxDati];
double r;
double t;
double S0;
double prz[NumMaxDati];
double q;
//Vettore dei punti di partenza
double ptoPar[8];

//Caricamento dati da file
NumDati = letturaFileInput(strike, &r, &q,&t, &S0, prz);

double z0 = log(S0);
```

```
double sol[8],valFunz, vetFunzObj[NUMITER*(size-1)];
int contatore=0;
alglib::real_1d_array temp = alglib::real_1d_array();
alglib::real_1d_array paramLOO = alglib::real_1d_array();
alglib::real_2d_array soluzioni = alglib::real_2d_array();
string str = "[";
  double tempObj=10e10;
double *solFin = new double[8];
int index,flag, contaRicevuti=0,contaRicevutiFase2=0
, contaInvii=0,contaInviiFase2=0;
int *destinazione = new int[NUMITER*(size-1)];
double *prezzi = new double[NumDati];;
bool fase2 = false;

//Creazione del file di output con la data di esecuzione
time_t rawtime;
time(&rawtime);
char* datetime = ctime(&rawtime);

string nomeFileSolRad,nomeFileSolFit;
string nomeFilePto;
string nomeFileLOO;
string sottostr;
sottostr.append(datetime);

nomeFileSolRad.append("Soluzione_Rada_Bates_");
nomeFileSolFit.append("Soluzione_Fitta_Bates_");
nomeFilePto.append("ptoPartenza_Bates_");
nomeFileLOO.append("LOO_Bates_");
nomeFileSolRad.append(sottostr.substr(4,20));
nomeFileSolFit.append(sottostr.substr(4,20));
```

```
nomeFilePto.append(sottostr.substr(4,20));
nomeFileLOO.append(sottostr.substr(4,20));
nomeFileSolRad.append(".txt");
nomeFileSolFit.append(".txt");
nomeFilePto.append(".txt");
nomeFileLOO.append(".txt");
//Definizione dei flussi
ofstream outfile1,outfileRad,outfileFit, outfileLOO;
//Apertura e scrittura su file
  outfile1.open(nomeFilePto.c_str());
outfile1 << "Par1;Par2;Par3;Par4;Par5;Par6;Par7;Par8" << endl;
outfileRad.open(nomeFileSolRad.c_str());
outfileRad << "Griglia Rada: " << DIFFSTEP1 << endl;
outfileRad << "Par1;Par2;Par3;Par4;Par5;Par6;Par7;Par8
;valFunzObj" << endl;
outfileFit.open(nomeFileSolFit.c_str());
outfileFit << "Griglia fitta: " << DIFFSTEP2 <<endl;
outfileFit << "Par1;Par2;Par3;Par4;Par5;Par6;Par7;Par8
;valFunzObj" << endl;
outfileLOO.open(nomeFileLOO.c_str());
outfileLOO << "PrezzoMercato;PrezzoHeston;Differenza;SolPar0;
SolPar1;SolPar2;SolPar3;SolPar4;SolPar5;SolPar6;SolPar7;
ValoreFunzObj" << endl;


//Creazione della lista dei punti che si vuol indagare
double listPto [NUMITER*(size-1)][8];

//Caricamento della lista con la sequenza di Halton
for(int l=0;l< NUMITER*(size-1);l++){
listPto[l][0] = haltonSequence(2,l)*2;
```

```
listPto[l][1] = haltonSequence(3,l)*20;
listPto[l][2] = haltonSequence(5,l)*0.3;
listPto[l][3] = haltonSequence(7,l)*pow(-1.0,double(l));
listPto[l][4] = haltonSequence(11,l)*0.3;
listPto[l][5] = haltonSequence(13,l)*1.5;
listPto[l][6] = haltonSequence(17,l)*pow(-1.0,double(l));
listPto[l][7] = haltonSequence(19,l)*3;
//Memorizzazione del punto su file
outfile1<<listPto[l][0]<<";"<<listPto[l][1]<<";"<<listPto[l][2]
   <<";"<<listPto[l][3]<<";"<<listPto[l][4]<<";"<<listPto[l][5]
   <<";"<<listPto[l][6]<<";"<<listPto[l][7]<<endl;
}
//Chiusura file
outfile1.close();

//Invio dei parametri globali ai processi
for(int i = 1;i < size;i++){
dest = i;
MPI_Send(&NumDati, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(&z0, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(&r, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(&t, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(strike, NumDati, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(prz, NumDati, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(&q, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
}
cout << "MASTER - Parametri inviati agli SLAVE" << endl;


//Estrazione casuale
if(LeaveOneOut_ON == true)
```

```
loo = estraiLOO();
for(int a = 1;a < size;a++){
dest = a;
//Caricamento del vettore con il punto di Halton
ptoPar[0] = listPto[a-1][0];
ptoPar[1] = listPto[a-1][1];
ptoPar[2] = listPto[a-1][2];
ptoPar[3] = listPto[a-1][3];
ptoPar[4] = listPto[a-1][4];
ptoPar[5] = listPto[a-1][5];
ptoPar[6] = listPto[a-1][6];
ptoPar[7] = listPto[a-1][7];
//Invio del punto all'a-esimo Slave
MPI_Send(ptoPar, 8, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
//invio passo di ricerca
MPI_Send(&diffstep, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(&loo, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
}
contaInvii=size-1;
//Si mettono in attesa tante funzioni di ricezione
// non bloccante quanti sono gli Slave
for(int ind = 1;ind < size;ind++){
dest = ind;
MPI_Irecv(&destinazione[ind-1], 1, MPI_INT, dest, tag,
MPI_COMM_WORLD, &arrayReq[ind-1]);
}
//Index fornisce il NUMERO dell'operazione non bloccante
index =0;
//Ciclo in cui vengono gestiti tutti gli Slave
while(1){
//Verifica se qualche Slave ha terminato la ricerca
```

```
MPI_Testany(size-1, arrayReq, &index, &flag, &status);
//Uno Slave \‘e pronto per trasmettere la soluzione
if(flag == true){
cout<<"TestiAny:"<<index<< " rank: "<<destinazione[index]<< endl;
tag = 3;
//Ricezione della soluzione e del valore della funzione obiettivo
MPI_Recv(sol, 8, MPI_DOUBLE, destinazione[index], tag
, MPI_COMM_WORLD, &status);
MPI_Recv(&valFunz, 1, MPI_DOUBLE, destinazione[index], tag
, MPI_COMM_WORLD, &status);
//Per la seconda fase si scrive su altro file
if(fase2 == true){
contaRicevutiFase2 ++;
outfileFit   <<sol[0]<<";"<<sol[1]<<";"<<sol[2]<<";"
<<sol[3]<<";"<<sol[4]<<";"<<sol[5]<<";"<<sol[6]<<";"
<<sol[7]<<";"<< valFunz << endl;
}
else{
outfileRad   <<sol[0]<<";"<<sol[1]<<";"<<sol[2]<<";"
<<sol[3]<<";"<<sol[4]<<";"<<sol[5]<<";"<<sol[6]<<";"
<<sol[7]<<";"<< valFunz << endl;
contaRicevuti ++;
}
cout << "Ricevuto dal "<< destinazione[index]
<< " valore funzione obiettivo : " << valFunz << endl;

//Viene accettata la soluzione con funzione obiettivo
// minimo sotto la soglia
if(valFunz < SOGLIA_FUNZ_OBJ){
vetFunzObj[contatore] = valFunz;
//costruzione dell’array per alglib
```

```
temp.setcontent(8, sol);
if(contatore != 0)
str.append(",");
str.append(temp.tostring(15));
//Conta le soluzioni sotto la soglia
contatore ++;
}
//Memorizza la solozione migliore
if(valFunz < tempObj){
tempObj = valFunz;
solFin[0] = sol[0];
solFin[1] = sol[1];
solFin[2] = sol[2];
solFin[3] = sol[3];
solFin[4] = sol[4];
solFin[5] = sol[5];
soFin[6] = sol[6];
solFin[7] = sol[7];
}
tag = 2;

//I punti iniziali sono stati ricevuti tutti
if(contaRicevuti == (NUMITER*(size-1))){
if(contaRicevutiFase2==0){
end = clock();
cout << "\n Fine prima fase: CPU time in secondi: "
<< (double)(end-start)/CLOCKS_PER_SEC << "\n"<<endl;
cout.precision(15);
cout << "Soluzione MIGLIORE: " << solFin[0] << " : "
<< solFin[1] << " : "<< solFin[2]<< " : "<< solFin[3]<<" : "
<< solFin[4] << " : "<< solFin[5]<< " : "<< solFin[6]<<" : "
```

```
<< solFin[7]<< endl;
cout << "Funzione obiettivo: " << tempObj << endl;
cout << "\n" << endl;
}
cout << "\nSECONDA FASE.\n" <<endl;
//Impongo lo step di differenzzazione alla seconda fase
diffstep = DIFFSTEP2;
//Se la seconda fase non \‘e ancora iniziata
if(contatore > 0 && fase2 == false){
str.append("]");
//Costruisce l’array per alglib
  soluzioni = alglib::real_2d_array::real_2d_array(str.c_str());
//Svuota la stringa
str.clear();
cout << "Array di soluzioni: "<< soluzioni.tostring(5) << endl;
int cluster;
alglib::integer_1d_array xyc;
//Ha senso di fare clustering solo se ci sono pi\‘u di una soluzione
if(soluzioni.rows() > 1){
//Analisi delle soluzioni fornite da Levenberg-Marquadart
alglib::ae_int_t npoints = soluzioni.rows();
alglib::ae_int_t nvars = 5;
//Euristica per il calcolo del numero di cluster significativi
alglib::ae_int_t k = numeroDiCluster(soluzioni,soluzioni.rows());
//Numero di ripartenze per kmeans
alglib::ae_int_t restarts = 3;
alglib::ae_int_t info;
alglib::real_2d_array c;

//Impongo il secondo passo di discretizzazione
diffstep = DIFFSTEP2;
```

```
//k-means clustering con il numero di cluster
 determinato precedentemente
alglib::kmeansgenerate(soluzioni, npoints, nvars, k, restarts,
 info, c, xyc);

//Determina il cluster migliore
cluster = qualeCluster(soluzioni,k, xyc, vetFunzObj);
cout << "Cluster migliore: " << cluster << endl;
}
else{
//Valorizzazione dummy
int uni[3]={1,1,1};
xyc=alglib::integer_1d_array::integer_1d_array();
xyc.setcontent(3,uni);
cluster =1;
}
//Array per la nuova regione
double estInf[8], estSup[8];
//Creazione della nuova regione di interesse
estInf[0] = 2;
estInf[1] = 20;
estInf[2] = 0.3;
estInf[3] = 1;
estInf[4] = 0.3;
estInf[5] = 1.5;
estInf[6] = 1;
estInf[7] = 3;
estSup[0] = 0;
estSup[1] = 0;
estSup[2] = 0;
```

```
estSup[3] = -1;
estSup[4] = 0;
estSup[5] = 0;
estSup[6] = -1;
estSup[7] = 0;
//Calcolo dei valori massimali e minimali per ogni dimensione
for(int i=0;i<soluzioni.rows();i++){
if(int(xyc[i]) == cluster){
if(soluzioni[i][0] < estInf[0])
estInf[0] =soluzioni[i][0];
if(soluzioni[i][0] > estSup[0])
estSup[0] =soluzioni[i][0];
if(soluzioni[i][1] < estInf[1])
estInf[1] =soluzioni[i][1];
if(soluzioni[i][1] > estSup[1])
estSup[1] =soluzioni[i][1];
if(soluzioni[i][2] < estInf[2])
estInf[2] =soluzioni[i][2];
if(soluzioni[i][2] > estSup[2])
estSup[2] =soluzioni[i][2];
if(soluzioni[i][3] < estInf[3])
estInf[3] =soluzioni[i][3];
if(soluzioni[i][3] > estSup[3])
estSup[3] =soluzioni[i][3];
if(soluzioni[i][4] < estInf[4])
estInf[4] =soluzioni[i][4];
if(soluzioni[i][4] > estSup[4])
estSup[4] =soluzioni[i][4];
if(soluzioni[i][5] < estInf[5])
estInf[5] =soluzioni[i][5];
if(soluzioni[i][5] > estSup[5])
```

```
estSup[5] =soluzioni[i][5];
if(soluzioni[i][6] < estInf[6])
estInf[6] =soluzioni[i][6];
if(soluzioni[i][6] > estSup[6])
estSup[6] =soluzioni[i][6];
if(soluzioni[i][7] < estInf[7])
estInf[7] =soluzioni[i][7];
if(soluzioni[i][7] > estSup[7])
estSup[7] =soluzioni[i][7];
cout <<soluzioni[i][0]<<"; "<<soluzioni[i][1]<<";"<<soluzioni[i][2]
<<";"<<soluzioni[i][3]<<"; "<<soluzioni[i][4]<<";"<<soluzioni[i][5]
<<";"<<soluzioni[i][6]<<"; "<<soluzioni[i][7]<< endl;
}
}
cout << "Fase 2: regione di interesse" << endl;
cout<<"SUP:"<< estSup[0]<<"; "<<estSup[1]<<"; "<<estSup[2]<<"; "
<< estSup[3]<<"; "<<estSup[4]<<"; "<<estSup[5]<<"; "
<< estSup[6]<<"; "<<estSup[7]<< endl;
cout <<"INF:" <<estInf[0]<<"; "<<estInf[1]<<"; "<<estInf[2]<<"; "
             <<estInf[3]<<"; "<<estInf[4]<<"; "<<estInf[5]<<"; "
             <<estInf[6]<<"; "<<estInf[7]<< endl;
//Costruzione e successivamente invio della nuova
// sequenza di partenza
cout << "Costruzione della sequenza finale." << endl;
tag = 2;
for(int i = 1;i < size;i++){
dest = i;
ptoPar[0] = haltonSequence(2,i)*(estSup[0]-estInf[0])+ estInf[0];
ptoPar[1] = haltonSequence(3,i)*(estSup[1]-estInf[1])+ estInf[1];
ptoPar[2] = haltonSequence(5,i)*(estSup[2]-estInf[2])+ estInf[2];
ptoPar[3] = haltonSequence(7,i)*(fabs(estSup[3]) - fabs(estInf[3]))
```

```
 + estInf[3];
ptoPar[4] = haltonSequence(11,i)*(estSup[4]-estInf[4])+ estInf[4];
ptoPar[5] = haltonSequence(13,i)*(estSup[5]-estInf[5])+ estInf[5];
ptoPar[6] = haltonSequence(17,i)*(fabs(estSup[6])-fabs(estInf[6]))
 + estInf[6];
ptoPar[7] = haltonSequence(19,i)*(estSup[7]-estInf[7])+estInf[7];
cout<<"Ultimi parametri da inviare: "<<ptoPar[0]<<" : "
<<ptoPar[1]<<" :" <<ptoPar[2]<<" : " <<ptoPar[3]<<" : "
<<ptoPar[4]<<" : "<<ptoPar[5]<<" : " <<ptoPar[6]<<" : "
<<ptoPar[7]<< endl;
MPI_Send(ptoPar, 8, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
//invio passo di ricerca
MPI_Send(&diffstep, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(&loo, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
}
//La seconda fase \'e partita
fase2 = true;

}
else
cout << "ATTENZIONE! NON E' STATA TROVATA NESSUNA SOLUZIONE
SOTTO LA SOGLIA IMPOSTATA : " << SOGLIA_FUNZ_OBJ<< ". \n" << endl;

//I punti minimi della seconda fase sono stati ricevuti tutti
if(contaRicevutiFase2==NumIterFase2*(size-1) || contatore==0){
//Soluzione MIGLIORE
cout << "\n\n" << endl;
cout.precision(15);
cout << "Soluzione MIGLIORE: " << solFin[0] << " : "<< solFin[1]
 << " : "<< solFin[2] << " : " << solFin[3] << " : "<< solFin[4]
 << " : "<< solFin[5] << " : "  << solFin[6]<< " : "<< solFin[7]
```

```
 << endl;
cout << "Funzione obiettivo: " << tempObj << endl;
//Se LOO \'e attiva si possono fare pi\'u cicli sugli stessi dati
if(LeaveOneOut_ON == true){
cout << "\nLOO ON - step " << contaLOO << "\n"<<endl;
paramLOO.setcontent(8,solFin);
setNumLOO(0);
setNumDati(NumDati);
prezzi = pricingCarrMadan(paramLOO, z0, strike,t, r, q);
setNumLOO(1);
//Estraggo i prezzi senza LOO
outfileLOO<<prz[loo]<<";"<<prezzi[loo]<<";"<<prz[loo]-prezzi[loo]
<<";"<<solFin[0]<<";"<<solFin[1]<<";"<<solFin[2]<<";"<<solFin[3]
<<";"<<solFin[4]<<";"<<solFin[5]<<";"<<solFin[6]<<";"<<solFin[7]
<<";"<<tempObj  << endl;
contaLOO ++;
//Valorizzazioni opportune per ripartire
contaRicevuti=0;
contaRicevutiFase2 = 0;
contaInvii =0;
contaInviiFase2 = 0;
contatore=0;
fase2 = false;
str = "[";

//Estrazione del nuovo dato data accantonare
loo = estraiLOO();
}
  //Terminazione del ciclo
if(contaLOO == ITER_LOO || LeaveOneOut_ON != true)
{
```

```
cout << "Tutti i minimi richiesti sono stati cercati,
invio segnale di terminazione ai processi." <<endl;
//Invia del segnale di terminazione ai processi in attesa
tag = 2;
for(int i = 1;i < size;i++){
dest = i;
ptoPar[0] = -1;
ptoPar[1] = -1;
ptoPar[2] = -1;
ptoPar[3] = -1;
ptoPar[4] = -1;
ptoPar[5] = -1;
ptoPar[6] = -1;
ptoPar[7] = -1;
MPI_Send(ptoPar, 8, MPI_DOUBLE, dest, tag,
 MPI_COMM_WORLD);
}
break;
}
}
}
// Se \'e stato ricevuto una soluzione allora se possibile
//assegnare nuovo punto da cercare allo Slave
if(contaInvii < (NUMITER*(size-1))){
ptoPar[0] = listPto[contaInvii][0];
ptoPar[1] = listPto[contaInvii][1];
ptoPar[2] = listPto[contaInvii][2];
ptoPar[3] = listPto[contaInvii][3];
ptoPar[4] = listPto[contaInvii][4];
ptoPar[5] = listPto[contaInvii][5];
ptoPar[6] = listPto[contaInvii][6];
```

```
ptoPar[7] = listPto[contaInvii][7];
//Il processo continua con la nuova ricerca
MPI_Send(ptoPar, 8, MPI_DOUBLE, destinazione[index],
 tag, MPI_COMM_WORLD);
//invio passo di ricerca
MPI_Send(&diffstep, 1, MPI_DOUBLE, destinazione[index],
 tag, MPI_COMM_WORLD);
MPI_Send(&loo, 1, MPI_INT, destinazione[index], tag,
 MPI_COMM_WORLD);
contaInvii++;
}
else
cout << "MASTER attendo la fine dei processi." <<endl;
//Per ogni ricezione si mette in attesa un'altra da qualsiasi Slave
MPI_Irecv(&destinazione[index], 1, MPI_INT, MPI_ANY_SOURCE, tag,
MPI_COMM_WORLD, &arrayReq[index]);
}
}
//Chiusura dei file
outfileLOO.close();
outfileRad.close();
outfileFit.close();


//Misurazione del tempo di esecuzione del programma
end = clock();
cout << rank << "CPU time in secondi: "
<< (double)(end-start)/CLOCKS_PER_SEC << endl;

  }else{
/*******************SLAVES*******************/
```

```
cout << "SLAVE " << rank << "!" << endl;
time_t start, end;
//avvio del tempo di esecuzione
start = clock();
//Inizializzazione delle variabili dello Slave
double z0, r,t,q;
double epsg = 0.0000000001;
double epsf = 0.0;
double epsx = 0.0;
alglib::ae_int_t maxits = 0;
alglib::minlmstate state;
alglib::minlmreport rep;
alglib::real_1d_array x0, x;
double *soluzione;
int contaMinimi =0;

//Il Master invia i parametri
source = 0;

//Ricezione dei parametri globali
MPI_Recv(&NumDati, 1, MPI_DOUBLE, source, tag,
MPI_COMM_WORLD, &status);

double arrayStrike[NumDati], arrayPrz[NumDati], pto[8];
setNumDati(NumDati);

MPI_Recv(&z0, 1, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status);
MPI_Recv(&r, 1, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status);
MPI_Recv(&t, 1, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status);

MPI_Recv(arrayStrike, NumDati, MPI_DOUBLE, source, tag,
```

```
 MPI_COMM_WORLD, &status);
MPI_Recv(arrayPrz, NumDati, MPI_DOUBLE, source, tag,
 MPI_COMM_WORLD, &status);
MPI_Recv(&q, 1, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status);

//Carica i dati originali nella struttura
caricaDati(z0,arrayStrike,t,r,q,arrayPrz);

//Ciclo in cui lo Slave ricerca tutti i minimi richiesti dal Master
while(1){
//Punto di partenza
tag=2;
//Ricezione del punto di partenza
MPI_Recv(pto, 8, MPI_DOUBLE, source, tag, MPI_COMM_WORLD, &status);

//Verifica se il master ha chiesto la terminazione
if((pto[0]==-1) && (pto[1]==-1) && (pto[2]==-1) && (pto[3]==-1)
 &&(pto[4]==-1) && (pto[5]==-1) && (pto[6]==-1) && (pto[7]==-1)){
cout << "SLAVE " << rank << " - NOP END! " << endl;
cout << "SLAVE " << rank << " Numero di MINIMI cercati: "
<< contaMinimi << endl;
break;
}
else{
contaMinimi ++;
//Ricezione del passo di discretizzazione
MPI_Recv(&diffstep, 1, MPI_DOUBLE, source, tag,
 MPI_COMM_WORLD, &status);
//Ricezione dell'indice in corrispondenza del dato da accantonare
MPI_Recv(&loo, 1, MPI_INT, source, tag, MPI_COMM_WORLD, &status);
//Caricamento della struttura dati accantonando loo-esimo dato
```

```
valorizzazioneLOO(loo);
//Se il Leave-One-Out \'e attivo
if(loo > 0)
numLOO = 1;
//Passa un flag alle altre funzione per avvisarle
// dell'eventuale LOO on
setNumLOO(numLOO);

cout << "SLAVE " << rank << " Cerco minimo..."
 << contaMinimi << endl;

//Conversione del tipo per alglib
x0.setcontent(8, pto);

//Condizioni sui parametri - per poter aver un significato
alglib::real_1d_array bndl = "[0.0,0.0,0.0,-1.0,0.0,0.0,-1,0]";
alglib::real_1d_array bndu = "[2.0,20.0,0.3,1.0,0.3,1.5,1,3]";
//Gestion dell'eventuale errore
try{
//Inizializzazione dell'algoritmo
alglib::minlmcreatev(8,NumDati-numLOO,x0,diffstep,state);
//Si impongono i vincoli
alglib::minlmsetbc(state, bndl, bndu);
//Condizioni di terminazione
alglib::minlmsetcond( state, epsg, epsf, epsx, maxits);
//Minimizzazione con LM
alglib::minlmoptimize(state, distanza);
//Lettura dei risultati
alglib::minlmresults(state, x, rep);
}catch(alglib::ap_error err){
cout<< processo << " - Si \'e verificato un'eccezione: "
```

```
<< err.msg << endl;
}
//Motivo per cui \'e terminata la ricerca
cout << "\nSlave " << rank << " - Termination type: " <<
 int(rep.terminationtype) << endl;

//conversione del tipo di alglib
soluzione = x.getcontent();
//Il destinatario \'e il processo Master
dest = 0 ;
//Invio del numero di processo per identificarsi col master
MPI_Send(&rank, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
tag =3;
//Invio della soluzione del valore della funzione obiettivo
MPI_Send(soluzione, 8, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
MPI_Send(&valFunzObj, 1, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
cout << "SLAVE " << rank << " - Soluzione inviata" << endl;
}
}

end = clock();
cout << rank << "CPU time in secondi: "
<< (double)(end-start)/CLOCKS_PER_SEC << endl;
}
//Sincronizzazione finale tra processi
MPI_Finalize();
return EXIT_SUCCESS;
}
```