Dipartimento di Impresa e Management

Corso di Laurea Triennale in Economia e Management

Cattedra di Matematica Finanziaria

PRICING DI OPZIONI EUROPEE: SVILUPPO DEL MODELLO DI BLACK-SCHOLES E DEL METODO DI MONTECARLO SU PYTHON

RELATORE:

Emerito Prof. Gennaro OLIVIERI

CANDIDATO:

Leonardo BONIVENTO

Matricola n. 210551

ANNO ACCADEMICO 2018/2019

A mamma, papà e a tutta la mia famiglia

Sommario

Introduzione

Negli ultimi decenni si è verificato un notevole aumento nell'utilizzo di strumenti finanziari derivati, infatti, viene stimato che attualmente ci siano in circolazione all'incirca 637mila miliardi di dollari in derivati. Per farsi un'idea, questo valore è più o meno uguale a nove volte il valore del PIL mondiale, si può così capire l'importanza che hanno assunto questi strumenti finanziari e si può anche capire come mai il numero dei metodi di valutazione di questi strumenti sia aumentato notevolmente nel corso degli anni.

Nello specifico in questo elaborato si andranno ad analizzare tre modelli di calcolo di un particolare strumento derivato: le opzioni (nello specifico opzioni europee) tenendo bene a mente che l'obiettivo finale però non è quello di elencare in modo puntuale i modelli esistenti bensì di osservare come sia possibile trasformare questi modelli in codici Python.

Si è deciso di utilizzare il linguaggio di programmazione Python perché molto intuitivo da utilizzare e per il semplice fatto che, al giorno d'oggi, è uno dei linguaggi più utilizzati nel mondo della finanza e dell'economia in generale.

Python, inoltre, ci permette di trasformare i codici in applicazioni web (web application) ossia trasformare il programma in una vera e propria applicazione eseguibile dal web facendo in modo che non sia visibile il codice Python sottostante ma che venga eseguito ogni qualvolta si inseriscano i dati necessari.

Per fare tutto questo si è deciso di spiegare, come prima cosa, cosa siano i generatori di numeri pseudo-casuali e quali siano le differenze tra i numeri pseudo-causali e i numeri casuali veri e propri, per il semplice fatto che anche il linguaggio di programmazione Python utilizza, come tutti i calcolatori, i numeri pseudo-casuali ogni qualvolta lo si usi per l'implementazione di particolari metodi statistici applicati alla finanza, uno di questi metodi è proprio quello che si andrà ad analizzare; il metodo di Montecarlo.

Il metodo di Montecarlo, infatti, viene definito in finanza come un insieme di algoritmi che sfrutta un campionamento aleatorio per arrivare ad un risultato che rappresenta l'approssimazione di quello che effettivamente è il risultato esatto dato da un altro modello. In questo elaborato si cercherà di mettere a confronto il modello di Black-Scholes con il metodo di Montecarlo per osservare quali effettivamente siano le differenze tra i due diversi metodi e quale sia l'errore di approssimazione del metodo di Montecarlo rapportato con quello di Black-Scholes.

Capitolo 1: I generatori di numeri pseudocasuali

1.1 Sequenze casuali o pseudo-casuali?

Nella vita di tutti i giorni capita spesso di trovarsi davanti a situazioni che la collettività definisce come eventi "casuali" come per esempio nelle lotterie con premio in denaro dove i numeri estratti sono dei veri numeri casuali.

La definizione data dal dizionario è quella di "*un evento che non è voluto o programmato"* o "*ciò che non è controllabile dalla volontà perché imprevedibile*"[1] ciò è quello che succede nell'esempio sopra citato.

Alcuni eventi definiti casuali possono non esserlo del tutto, un caso studiato in letteratura è quello del Paradosso di Monty Hall[2], un famoso problema di teoria della probabilità basato su un vecchio show televisivo americano chiamato "*Let's Make a Deal*".

In questo programma il partecipante deve scegliere una tra tre diverse porte, dietro una di esse c'è una automobile, dietro le altre due ci sono premi di nessun valore. Una volta che il partecipante ha scelto una delle tre porte il conduttore, che sa esattamente dove si trovi la macchina, ne sceglie una delle due rimaste e la apre svelando che dietro a quella porta non c'è la macchina. A questo punto il partecipante deve decidere se mantenere la scelta fatta all'inizio oppure se cambiarla. Questo sembra un gioco totalmente casuale ma in realtà così non è perché cambiando la scelta fatta all'inizio le probabilità di trovare l'automobile raddoppiano. Se si vanno ad esaminare le tre possibilità (ciascuna avente un terzo di possibilità di riuscita):

- il giocatore sceglie la porta n°1. Il conduttore sceglie la n°2 mostrando una capra. Cambiando, il partecipante vince l'auto.
- il giocatore sceglie la porta n°2. Il conduttore sceglie la n°1. Cambiando, il partecipante vince l'auto.
- il giocatore sceglie l'auto. Il conduttore sceglie una capra, non importa quale. Cambiando, il partecipante trova l'altra capra.

Quindi se il partecipante decide di mantenere inalterata la propria scelta iniziale si avrà un terzo di probabilità di vincere mentre cambiando scelta avrà i due terzi di probabilità. Questo ragionamento può sembrare illogico perché si potrebbe pensare che una volta aperta la porta dal conduttore il partecipante abbia il 50% di possibilità di vittoria (come per la probabilità che esca testa o croce nel lancio di una moneta) ma così non è perché, al contrario del lancio della moneta, l'esclusione da parte del conduttore di una delle porte rende la porta non scelta più

[1] http://www.dizionario-italiano.org/Casualità

[2] (Migliorini, 2019)

interessante agli occhi del partecipante rispetto a quando non aveva nessuna conoscenza riguardo le tre porte all'inizio del gioco.

Spiegare quindi cosa effettivamente è la casualità è molto difficile se non quasi impossibile, in questa sede, quello che è veramente importante comprendere è cosa si intenda per *sequenza casuale di numeri*, ossia una sequenza di, *n*, numeri in un intervallo limitato nel quale non è possibile predire n_{k+1} da una qualsiasi combinazione di precedenti valori n_i, i=0,1,…k.[3]

Una sequenza casuale di numeri in senso stretto può essere ottenuta solo se è possibile utilizzare una sorgente casuale esistente in natura, in tutti quei casi, invece, in cui si devono progettare software o programmi con metodi aritmetici è quasi impossibile ottenere sequenze di numeri totalmente casuali privi di errori e correlazioni.

John Von Neumann (1951) affermava che: "*any one who considers arithmetical methods of producing random digits is, of course, in a state of sin*"[4].

Questo concetto è molto significativo perché permette di capire un passaggio molto importante ossia che non è possibile produrre sequenze di numeri puramente casuali con l'utilizzo di metodi matematici perché si cadrebbe in errore.

Un vero generatore è uno strumento capace di ottenere una sequenza di numeri non deterministici (di conseguenza non prevedibile) mentre in tutti quei casi in cui si ricorre all'uso di un calcolatore, quindi un oggetto puramente deterministico, si deve ricorre all'uso di metodi matematici e questo fa si che le sequenze ottenute non siano del tutto causali.

Si può concludere che nessun calcolatore è in grado di produrre una sequenza *puramente* casuale di numeri ma solo sequenze di numeri pseudocasuali (così vengono chiamate tutte le sequenze generate da algoritmi matematici) grazie all'utilizzo di generatori che riescano a formare sequenze di numeri che, almeno all'apparenza, siano casuali[5] e in grado di superare una serie di test statistici.

Queste serie di numeri vengono definite pseudocasuali perché, venendo a conoscenza dell'algoritmo e del primo elemento utilizzato (detto seme), è possibile determinare tutta la sequenza generata. Il seme è l'unico elemento puramente aleatorio dell'intera sequenza di numeri.

Tutti i generatori e i modelli che verranno presentati in questo lavoro saranno comunque da considerarsi generatori e modelli di numeri pseudocasuali e non puramente casuali.

---

[3] (Kneusel, 2018)

[4] (Neumann, 1951)

[5] (Pareschi, 2009)

1.2 Cenni storici

L'uomo fin dall'antichità ha cercato di riprodurre o comprendere eventi casuali che si potevano riscontrare in natura, molto spesso con una certa difficoltà.

Il primo dado fu ritrovato in una tomba in Medio Oriente 2400 anni prima della nascita di Cristo, nel 1890 lo statistico sir Francis Galton scriveva ancora su "*Nature*" che il modo migliore che aveva trovato per generare numeri casuali era con il lancio dei dadi: "*Quando vengono scossi e lanciati in un bussolotto, sbattono in modo così variabile tra di loro e contro le pareti del bussolotto che rimbalzano in modo folle, e le loro posizioni iniziali non danno alcun indizio percettibile su come si troveranno anche dopo una singola bella mescolata e lancio*"[6].

Nel Novecento la domanda di numeri casuali per risolvere problemi di varia natura, principalmente di natura probabilistica, che finiranno per essere chiamati metodi di Monte Carlo aumentò, vertiginosamente, questo perché molte applicazioni richiedevano una fornitura molto elevata di numeri casuali, rendendo insufficiente l'apporto portato dai soli dadi.

Proprio per risolvere questo problema a metà degli anni '40 una società chiamata RAND Corporation scrisse un libro intitolato "*A Million Random Digits with 100.000 Normal Deviates*"[7] nel quale vennero pubblicate delle tabelle con cifre casuali prodotte randomizzando una tavola basica generata da una roulette elettronica.

Per la prima volta lunghe sequenze di numeri casuali furono messe a disposizione di tutti: ricercatori, scienziati e matematici.

Nello stesso periodo una macchina simile fu inventata da un team di decodificatori a Bletchley Park[8], questa macchina fu chiamata ERNIE (Electronic random Number Indicator Equipment) e fu utilizzata per generare numeri casuali per la lotteria dei Premium Bond del Regno Unito, questi numeri erano ottenuti grazie a segnali acustici creati da tubi al neon.

Dopo la Seconda Guerra Mondiale, la randomizzazione di numeri pseudocasuali fu inserita per la prima volta all'interno di un computer vero e proprio che venne chiamato Ferranti Mark 1[9] che fu il primo computer elettronico al mondo ad essere messo in commercio.

In quel periodo si iniziò a formare l'idea di creare dei generatori di numeri pseudocasuali (PRNG) cioè generatori che seguissero una funzione matematica deterministica sottostante.

---

[6] (Galton, 1890)

[7] (Paul Armer, 1955)

[8] Durante la Seconda Guerra Mondiale era diventato il quartier generale per i decodificatori di codici e sistemi tedeschi, italiani e giapponesi (https://www.bletchleypark.org.uk/our-story/why-it-matters/cottage-industry)

[9] https://www.wikizero.com/it/Ferranti_Mark_1

Come si è detto prima, un PRNG può creare una sequenza di numeri pseudocasuali ma solamente dopo aver determinato il primo numero della sequenza (il cosiddetto seme). Per capire meglio cosa si intende per PRNG si può utilizzare la definizione data da Maurizio Codogno "*un PRNG è in pratica una funzione matematica deterministica che viene man mano iterata nel senso che usa il risultato precedente per calcolare quello nuovo. Quindi se si parte con lo stesso valore iniziale ("seme") si otterrà sempre la stessa soluzione*"[10].

Uno dei primi PRNG fu il cosiddetto "*Middle Square Method*" e fu sviluppato da John Von Neumann nel 1946, questo generatore però aveva delle limitazioni molto importanti che verranno spiegate in seguito, in questa sede possiamo dire che questo modello portava a delle piccole sequenze di numeri sempre ripetute.

Agli inizi degli anni '50 furono fatti dei passi in avanti. Il matematico americano D.H. Lehmar iniziò a teorizzare quello che verrà poi chiamato "*Linear Congruential Generator*" (LCG), questo metodo permette di creare sequenze più lunghe e viene identificato come un generatore meno dipendente dal seme iniziale.

Il generatore LCG è stato molto utilizzato nel tempo come generatore di numeri casuali anche da uno dei primi browser pubblici per la navigazione in Internet agli inizi degli anni '90: *Netscape*. Successivamente Phillip Hallam-Baker scoprì che la tecnologia SSL[11] di Netscape non era immune ad attacchi informatici infatti riuscì a risalire al seme del generatore di numeri casuali utilizzato dalla tecnologia SSL, dimostrandone quindi i limiti e la sua poca affidabilità come tecnologia di sicurezza per la trasmissione di dati sensibili.

Si sentì quindi la necessità di passare a generatori ancora più efficaci.

Nel 1998 è stato creato il sito irlandese random.org[12] che genera numeri totalmente casuali, questi nuovi generatori vennero anche chiamati True Random Number Generator (TRNG) proprio per sottolineare il fatto che avevano raggiunto un livello di casualità tale da poter essere classificati proprio come generatori di "veri" numeri casuali.

Nel 1999 fu introdotto un nuovo generatore PRNG sviluppato da Makoto Matsumoto e Takuji Nishimura che è stato chiamato "*Mersenne Twister*". Questo generatore fu chiamato così perché, come vedremo successivamente, si basa su un numero primo detto di Mersenne.

Agli inizi degli anni 2000 sono stati costruiti nuovi generatori chiamati CSPRNG (**C**ryptographically **S**ecure **P**seudo-random **N**umber **G**enerator) ossia un generatore di numeri pseudocasuali le cui proprietà lo rendono adatto all'uso in crittografia.

---

[10] (Codogno, 2018)

[11] Secure socket Layer (SSL) tecnologia che garantisce la sicurezza di una connessione internet, protegge i dati sensibili e permetto lo scambio di questi dati tra due entità

[12] https://www.random.org/

1.3 L'evoluzione dei PRNG

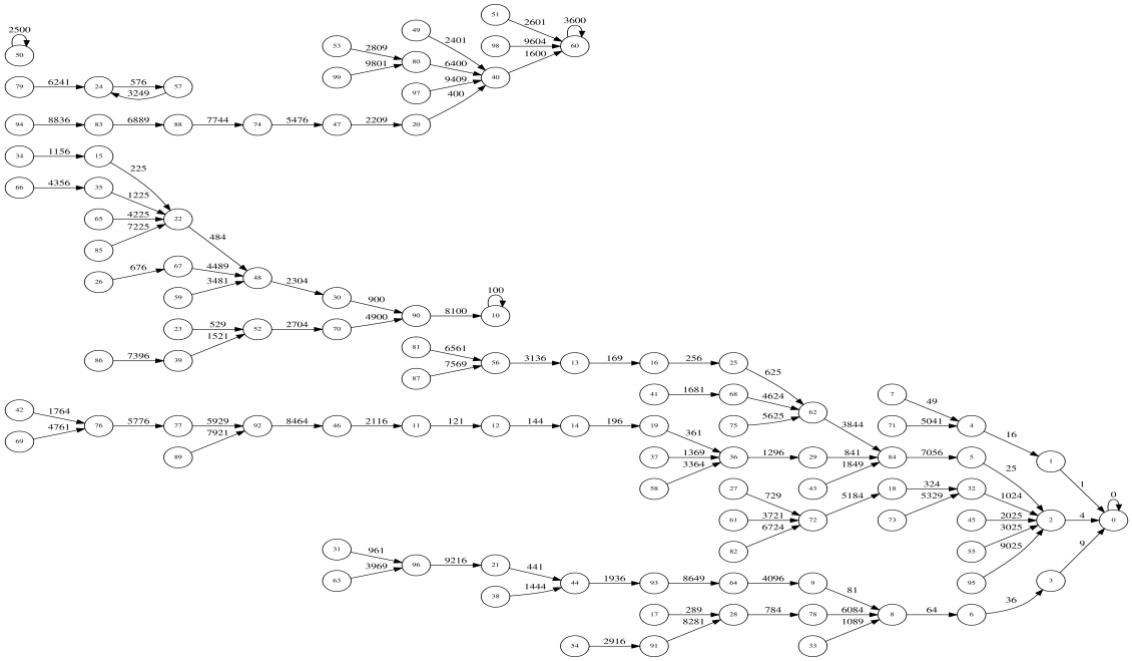
1.3.1 Le origini del PRNG

Come si è visto precedentemente, uno dei primi PRNG fu sviluppato da John Von Neumann nel 1946, la sua idea era molto semplice e prevedeva di prendere un numero per esempio di 4 cifre (seme) lo si eleva al quadrato e si prendono le 4 cifre centrali.
Esempio:

$$4892 \rightarrow 23931664 \rightarrow 9316$$

Il numero così trovato diventa il nostro nuovo seme da cui si potrà ripartire per ottenere una sequenza di numeri pseudocasuali, questo metodo venne chiamato "*Middle Square Method*" proprio per come veniva ottenuta la sequenza.
Tuttavia, questo metodo ha delle limitazioni:

- dipende troppo dal seme iniziale infatti un buon generatore, per essere definito come tale, non dovrebbe dipendere troppo dal seme perché c'è il rischio che venga scoperta poi l'intera sequenza, con questo metodo è molto facile cadere in cicli continui di numeri come vedremo dopo dal grafico.
- la potenza quadrata non è un metodo molto buono a coprire la lunghezza dell'intervallo stesso infatti si potrebbe pensare che dando *n* numeri si dovrebbero avere n^2 soluzioni invece, estraendo unicamente le cifre centrali si hanno solo il 60% dei valori sono in realtà raggiunti.



Nel grafico sono rappresentati i possibili percorsi ottenibili utilizzando il metodo del "the Middle square"[13]

In questo grafico si può vedere come, utilizzando un seme di due cifre, sia possibile incorrere in valori che, una volta elevati alla seconda e isolate le cifre centrali, producano lo stesso valore, per esempio prendendo i numeri 15, 35, 65, 85 si otterrà sempre lo stesso risultato successivo ossia 22 proprio per questo motivo non si avranno n^2 soluzioni ma il numero sarà ridotto notevolmente.
È possibile notare inoltre che le sequenze di numeri hanno dei periodi piuttosto brevi perché è facile arrivare a cicli continui di numeri rendendo questo primo esempio di generatore non adatto per essere implementato su calcolatori.

1.3.2 Definizione e caratteristiche generali di un generatore

Una definizione di generatore è stata data da Pierre L'Ecuyer[14] che ha definito un generatore di numeri pseudocasuali come un qualcosa rappresentato con una struttura del tipo
$g = (S, s_0, T, U, G)$ dove:

- S è un numero finito di stati
- $S_0 \in S$ (questo è lo stato iniziale)
- $T: S \rightarrow S$ è la funzione di transazione utilizzata per determinare lo stato al tempo t+1 dato lo stato al tempo t, formalmente $s_{i+1}=T(s_i)$
- U è un insieme finito di simboli di uscita (spazio degli output)
- $G: S \rightarrow U$ è la funzione d'uscita. Dato un qualsiasi stato $s_i, u_i=G(s_i) \in U$

Un generatore inizia dallo stato iniziale (s_0) che viene anche chiamato seme e $u_0=G(s_0)$, poi per i= 1, 2, …., si imposta $s_i=T(s_{i-1})$ e $u_i=G(s_i)$. Si assume che siano disponibili procedure efficienti per T e G. La sequenza di valori $\{u_i\}$ è l'output del generatore e gli elementi u_i sono chiamati *osservazioni*.
Bisogna però ricordare che poiché lo spazio degli stati S è finito, prendendo un qualsiasi seme s_i, esisterà sempre un valore *l* tale per cui $s_{i+1}=s_i$.
Questo significa che dopo un numero di iterazioni *l* il generatore di numeri pseudocasuali tornerà inevitabilmente allo stato iniziale.

[13] (Pigeon, s.d.)

[14] (L'Ecuyer, 1997)

I generatori producono una sequenza fissa di numeri tale per cui i precedenti k numeri (di solito solo il singolo precedente numero) determinano il successivo, proprio perché la sequenza è finita prima o poi, si ripeterà.
Il numero più piccolo di iterazioni necessario perché si riproponga il valore iniziale è chiamato "*periodo*" del PRNG ed è individuato solitamente con la lettera greca ρ.
Lo stato iniziale (s_0) deve essere dato.
Per far si che sia introdotta più casualità all'interno della sequenza si può scegliere questo valore in modo puramente casuale estraendolo da una scatola piena di palline con differenti valori.
La scelta puramente casuale di un piccolo valore come seme fa si che si possa trasformare quel determinato stato iniziale in una sequenza che appaia e si comporti come una vera e propria sequenza casuale di numeri.
I generatori dovrebbero generare sequenze di numeri che siano uniformemente distribuiti (questo significa che devono avere la stessa probabilità di presentarsi) e indipendenti. Dovrebbero quindi riuscire a superare test statistici di uniformità e indipendenza. Questo però, come detto precedentemente, non si verifica al 100% questo in quanto la sequenza di numeri casuali creata è deterministica e periodica e quindi non sarà mai veramente uniforme. L'uniformità rimane comunque ciò a cui si deve tendere quando si progetta un generatore di numeri pseudocasuali.
Questo ragionamento porta a concludere che per capire se un PRNG possa essere considerato un buon generatore o meno si deve ricorrere ad altri parametri.
I parametri che vengono utilizzati per verificare la bontà di un generatore sono i seguenti: la lunghezza del periodo, l'efficienza, la ripetibilità e la portabilità, ora si andranno ad analizzare questi quattro parametri più nel dettaglio.
Il primo di questi parametri è la lunghezza del periodo: soprattutto negli ultimi anni i computer sono diventati via via più veloci con la necessità di un numero sempre più elevato di sequenze di numeri casuali, questo ha fatto si che i periodi delle sequenze che potevano essere sufficienti qualche anno fa non lo siano più, attualmente un generatore per essere considerato accettabile deve avere un periodo di lunghezza almeno pari 2^{60}.
Il secondo parametro che viene osservato è quello dell'efficienza, anche se il potere di calcolo dei computer è notevolmente aumentato, la velocità e l'utilizzo della memoria da parte dei generatori sono ancora considerati fattori importanti nell'osservazione della bontà del generatore stesso, un generatore lento e che utilizza molta memoria potrebbe non essere adeguato alla formazione di sequenze di numeri pseudocasuali.
Un altro aspetto importante è la ripetibilità e cioè la capacità di un dato calcolatore di riprodurre la stessa sequenza di numeri casuali.
È molto importante che un programma di numeri pseudocasuali possa essere ripetuto per due motivi:

- per poter testare e sviluppare ulteriormente il generatore
- per poter studiare meglio delle eccezioni che possono formarsi lungo la sequenza di numeri pseudocasuali.

L'ultimo parametro da osservare è la portabilità del generatore perché il codice del generatore deve poter essere scritto in un linguaggio che possa essere letto da un qualsiasi macchinario o computer, questo significa che differenti macchine devono riuscire a leggere lo stesso codice nello stesso modo, la lettura del codice però può variare un po' a seconda della differente accuratezza aritmetica dei diversi hardware in cui si sta andando ad implementare il generatore. Nei prossimi paragrafi si procederà con l'analisi di alcuni tra i più famosi e importanti generatori di numeri pseudocasuali utilizzati.

1.3.3 Metodo della congruenza lineare (LCG)

Un generatore molto conosciuto ma considerato da alcuni insicuro è il "*Linear Congruential Generator*" (LCG), introdotto da D.H. Lehmer nel 1948.
Questo metodo permette, dato un certo valore iniziale s_0[15] (seme) di ottenere una sequenza di numeri pseudocasuali mediante l'applicazione ripetuta della seguente formula:

$$s_t = (as_{t-1} + c) \bmod m$$

Dove a è un coefficiente strettamente positivo che viene chiamato *moltiplicatore* e compreso tra $0<a<m$, c è un coefficiente intero non negativo detto *incremento*, m è un coefficiente intero strettamente positivo detto *modulo*, s_t è un generico numero della sequenza il quale deve essere $0 \leq s_t \leq m$.
Come si può osservare dalla formula è presente anche l'operazione modulo[16] a(mod B) che rappresenta la parte intera del resto della divisione tra a e b, questa funzione è molto utilizzata nei linguaggi di programmazione e di solito viene rappresentata dal simbolo (%).
Una volta ottenuta la sequenza di numeri pseudocasuali spesso si cerca poi di ottenere una sequenza di numeri pseudocasuali U_n uniformemente distribuita in un intervallo (0;1).
Visto che un computer può rappresentare un numero reale con una accuratezza limitata (per quanto riguarda le cifre che possono essere rappresentate) di solito si generano prima sequenze di numeri interi s_t tra zero e un qualunque numero m e poi si procede prendendo ogni singolo s_t e lo si divide per m:

---

[15] Il seme deve essere un numero intero compreso tra $0 \leq s_0 \leq M - 1$

[16] https://it.wikipedia.org/wiki/Operazione_modulo

$$u_n = \frac{s_t}{m}$$

La frazione così ottenuta sarà compresa tra zero e uno. Solitamente *m* è il numero massimo di bits che il computer può processare in un preciso momento (chiamato anche *word size* del computer).

Se *a* ed *m* sono scelti propriamente la sequenza di numeri u_n sembrerà una sequenza uniformemente distribuita tra 0 e 1.

Lo sviluppo di questo PRNG ha portato a due diverse versioni che differiscono per il valore che ha c, queste due versioni sono:

- Il metodo congruenziale moltiplicativo (Multiplicative congruential generator)
- Il metodo congruenziale misto (Mixed congruential generator)

Il metodo moltiplicativo introdotto per la prima volta da Lehmer fu teorizzato con il valore c=0, mentre il metodo congruenziale misto che fu teorizzato in alcuni scritti di Rotenberg, ha la stessa formula del metodo precedente ma con il valore c≠0.

Le prove empiriche sulla versione mista del metodo congruenziale hanno portato ad affermare che questo metodo può portare ad alcuni vantaggi rispetto al metodo congruenziale moltiplicativo per esempio riesce a produrre sequenze di numeri pseudocasuali con periodi più lunghi, può essere usato qualsiasi valore come seme iniziale (il valore 0 è escluso nel metodo moltiplicativo) e in molte macchine su cui si è adoperato questo tipo di generatore si è riscontata una velocità di esecuzione maggiore rispetto al più veloce generatore moltiplicativo.

Uno dei limiti che è comune a tutte e due le versioni di questo generatore riguarda la lunghezza massima che può raggiungere la sequenza di numeri pseudocasuali generata, in questo generatore la lunghezza massima è rappresentata da *m* nel caso in cui c sia diverso da zero mentre nel caso in cui *m* sia uguale a zero la lunghezza del periodo può essere al massimo pari a m-1.

Bisogna ricordare che il periodo ha lunghezza al massimo pari ad un valore ma è possibile che la sequenza di numeri sia anche inferiore al valore assegnato a *m* e proprio per questo motivo il moltiplicatore è considerato il parametro più importante all'interno della formula dell'LCG.

Solo se il parametro *m*, è un numero primo e il moltiplicatore, *a*, è una radice primitiva del modulo *m* allora il generatore riuscirà a raggiungere la sua lunghezza massima.

Una radice primitiva, *a*, modulo di un numero primo, *m*, è un numero tale per cui il più piccolo valore positivo *k* che soddisfi la seguente formula sia uguale a m-1.

$$a^k = 1 \bmod m$$

Anche se Niederreiter scrisse "*There is no such thing as a universally optimal multiplier*"[17] ossia espose il fatto che non fosse possibile trovare un moltiplicatore (*m*) ottimale, ci sono però alcuni moltiplicatori che vengono utilizzati perché considerati buoni per produrre sequenze di numeri pseudocasuali di una lunghezza accettabile.

I più importanti esempi di LCG vengono elencati qui di seguito con la seguente notazione: LCG(M, a, c, s_0)[18]:

- LCG(2^{31}, 1103515245, 12345, 12345) conosciuto come generatore Ansi-C, impiegato nella funzione rand()
- LCG(2^{31}-1,16807,0,1) questo è definito come generatore "minimal standard" e fu proposto da Lewis nel 1969, questo generatore è usato anche per software ad uso commerciale.
- LCG(2^{31}, 65539,0) questo è definito come RANDU, questo è stato per molti anni il più utilizzato generatore di numeri pseudocasuali al mondo.

1.3.4 Lagged Fibonacci Generator

Un altro generatore molto utilizzato prende spunto da una sequenza di numeri chiamata "*la sequenza di Fibonacci*" ossia una successione di numeri interi che sono collegati tra loro da una semplice relazione del tipo:

$$x_{i+2} = x_{i+1} + x_i$$ [19]

Come si può notare questa sequenza si basa su una stretta relazione che si viene a formare tra i primi due numeri successivi della sequenza che vengono sommati tra di loro per trovare il terzo numero, questo procedimento si può portare avanti all'infinito.

Il primo generatore collegato alla serie di Fibonacci fu analizzato negli anni '50 e prendeva semplicemente la somma dei due numeri successivi della serie e li riduceva per mod m:

$$x_{n+1} = (x_n + x_{n-1}) \bmod m$$

[17] (James, 1990)

[18] (Hellekalek, 1998)

[19] I primi numeri della successione di Fibonacci sono: 1,1,2,3,5,8…

La sequenza che si veniva a formare assomigliava per molti aspetti ad una sequenza di numeri casuali ma non riusciva a soddisfare molti dei più semplici test statistici e quindi non poteva essere considerato un buon generatore di numeri pseudocasuali.
Proprio per questo motivo nel 1958 grazie agli studi di G.P. Mitchell e D.P. Moore[20] (lavoro non pubblicato) si è raggiunta una ulteriore versione di questo generatore che si discosta leggermente da quella precedentemente esposta infatti invece di prendere i due successivi termini della sequenza gli studiosi in questione decisero di combinare due termini che si trovavano ad una maggiore distanza tra di loro, la loro versione era la seguente:

$$x_i = (x_{n-24} + x_{n-55}) \text{mod } m$$

Dove *m* è pari, *n* deve essere un valore superiore o uguale a 55.
I valori costanti di questa sequenza (24 e 55) non sono scelte a caso ma sono valori che definiscono una sequenza che ha un periodo di una lunghezza almeno pari a 2^{55}-1, come si vedrà in modo generalizzato nella tabella successiva, il periodo sarà esattamente uguale a (2^{55}-1) * 2^{l-1} quando *m* è uguale a 2^l.
La novità di questo generatore quindi è il fatto di far affidamento, per la generazione di numeri pseudocasuali, non più ad un solo seme iniziale ma ad una coppia di valori.
La formula che poi venne resa universale è la seguente:

$$x_i = \left(x_{i-j} + x_{i-k}\right) \text{ mod } m$$

Se j, k e m sono scelti in modo appropriato, si può produrre una sequenza più che soddisfacente di numeri pseudocasuali. I valori *j* e *k* vengono comunemente chiamati "*lags*", termie che da il nome al generatore e la x_i si dice appunto successione lagged-Fibonacci (LFG).
Il periodo della sequenza può raggiungere il valore di m^k-1 se m è un numero primo e k>j anche se molte volte per semplicità nell'implementazione di questo generatore viene scelto un $m=2^l$ con un *l* qualsiasi. Il generatore con modulo di valore uguale a 2 alla potenza di un numero qualsiasi produce sequenze con periodi molto più corti rispetto al primo caso con *m* uguale ad un numero primo superiore a due.
È importante ricordare in questa sede che questo particolare tipo di generatore spesso è rappresentato nel seguente modo:

$$x_k = (x_{k-p} \otimes x_{k-p+q}) \text{mod } m$$

[20] (Knuth, 1998)

Questo per indicare che l'operazione eseguita tra i due valori all'interno della parentesi può essere una qualsiasi tra +, -, × nei primi due casi il generatore prende il nome di "*additive lagged Fibonacci generator*" mentre nel terzo caso viene indicato come "*moltiplicative lagged Fibonacci generator*".

Un particolare vantaggio che hanno tutte queste diverse versioni del LFG rispetto al LCG è la maggior lunghezza che il LFG può raggiungere rispetto al LCG. La lunghezza però dipende anche dall'operatore utilizzato nella formula come si può osservare nella tabella seguente[21] nella quale ad ogni operatore corrisponde una lunghezza massima ottenibile:

TABLE I
Maximum Attainable Periods of Lagged Fibonacci Generators $x_k = x_{k-p} \otimes x_{k-p+q} \bmod 2^l$

Operation	Maximum attainable period
Addition, mod 2^l	$(2^p - 1)2^{l-1}$
Subtraction, mod 2^l	$(2^p - 1)2^{l-1}$
Multiplication, mod 2^l	$(2^p - 1)2^{l-3}$

Questa tabella rappresenta i massimi periodi ottenibili con l'utilizzo del generatore "Lagged Fibonacci"[22]

1.3.5 Mersenne – Twister

Mersenne Twister è attualmente il generatore più utilizzato per la formazione di sequenze di numeri pseudocasuali almeno per i tutti quei casi in cui non è richiesto un generatore sicuro crittograficamente. Questo generatore è stato sviluppato nel 1997 da Makoto Matsumoto e Takuji Nishimura, la versione poi originale è stata modificata leggermente dai due autori negli anni successivi ed è stata resa nota con la pubblicazione del testo intitolato "*Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-random Number Generator*"[23]. Proprio questa versione è quella che viene utilizzata ancora oggi.

La prima versione aveva alcuni problemi di casualità infatti produceva sequenze di numeri con troppi zeri che rendevano poi il generatore prevedibile o quanto meno poco casuale.

[21] (Aluru, 1997)

[22] (Aluru, 1997)

[23] (Nishimura, 1998)

La versione definitiva, invece, supera questo problema acquisendo anche alcune caratteristiche che ne dimostrano la bontà:

- la lunghezza del periodo: questo generatore ha un periodo di lunghezza 2_{19937}-1 (anche chiamato numero primo di Mersenne[24]), il numero in questione è composto da 6002 cifre.
- la distribuzione dei valori finali: un generatore per essere definito un "buon generatore" oltre ad avere un periodo molto lungo deve anche rispettare i requisiti di casualità, per verificare ciò basta considerare quanto omogeneamente un generatore riesca a riempire lo spazio. Il Mersenne Twister permette di generare punti equidistribuiti in spazi fino a 623 dimensioni.
- la velocità: è infatti più veloce della maggior parte degli algoritmi di pari prestazioni qualitative
- la casualità: ha passato infatti numerosi test statistici di casualità.

Molti PRNG hanno, al loro intero, il concetto di "stato", per il LCG lo stato è un singolo numero di 32-bit invece per il Mersenne Twister è di 624 parole ed ogni parola ha un valore di 32-bit. Ciò significa che ogni "stato" del generatore produce 624 numeri casuali e questi valori saranno disposti in una sequenza che non si ripeterà mai nel periodo che è ha una lunghezza di 2_{19937}-1 come già detto precedentemente.

Questo generatore è stato scritto in linguaggio di programmazione C, i programmi scritti in questo linguaggio sono principalmente composti da espressioni matematiche e da procedure parametrizzate in grado di manipolare dati molto complesse il che rende molto difficile l'analisi di questo generatore da un punto di vista puramente teorico.

[24] I numeri primi di Mersenne derivano dalla formula $M_p=2^p-1$ dove p deve essere un numero primo

Capitolo 2: Dal modello binomiale al metodo di Montecarlo per la valutazione del prezzo di opzioni europee su azioni

2.1 Opzioni: concetti introduttivi

Il contratto di opzione è un contratto mediante il quale una delle due parti concede all'altra la facoltà, ma non l'obbligo, di concludere in una data futura (maturity o data di scadenza) un contratto di acquisto (**opzione call**), oppure di vendita (**opzione put**), di una certa quantità di una determinata attività sottostante ad un prezzo prefissato (strike price) a fronte del pagamento di un premio[25].

La parte fondamentale, che rende l'opzione uno strumento differente dalle altre tipologie di derivati, è il diritto di scelta (la facoltà) che il detentore del titolo ha, sul venditore, di poter esercitare l'opzione.

L'opzione può essere esercitata solo a scadenza del contratto se si tratta di un'opzione *europea* mentre può essere esercitata in qualsiasi momento se si tratta di un'opzione *americana*.

Una opzione di tipo europeo è più semplice da analizzare rispetto una di tipo americano, per questo motivo, in questa sede, verranno analizzate le opzioni di tipo europeo.

Il prezzo delle opzioni viene calcolato utilizzando modelli matematici che tengano conto di *sei fattori* quantificabili[26]:

- il prezzo del sottostante: se il valore del sottostante aumenta il valore intrinseco di un'opzione call aumenta mentre quello di un'opzione put diminuisce;
- il prezzo di esercizio: è chiamato anche *strike price* e influisce sul valore intrinseco dell'opzione;
- la volatilità: la volatilità è molto importante perché permette di misurare le fluttuazioni del prezzo del sottostante in quanto maggiore è la volatilità più alta è la probabilità che ci siano ampie fluttuazioni del prezzo del titolo;
- il tempo a scadenza: più ci si avvicina a scadenza più il valore temporale dell'opzione si riduce progressivamente fino ad arrivare a zero a scadenza;
- i tassi di interesse: le variazioni del tasso di interesse provocano delle variazioni al prezzo del sottostante;
- i dividendi: la distribuzione di dividendi incide sul prezzo del sottostante e quindi ha un impatto anche sul prezzo dell'opzione.

---

[25] (Fabrizi, 2016)

[26] http://www.borsaitaliana.it/

Come già anticipato, esistono due tipologie di opzioni: le opzioni call e opzioni put.

Le **opzioni call** corrispondono il diritto al compratore (ma non l'obbligo) di comprare il titolo sottostante in una specifica quantità al prezzo concordato nel contratto (strike price). In questa fattispecie l'opzione verrà esercitata solo nel caso in cui il prezzo del titolo sottostante risulterà superiore al prezzo di esercizio perché, in questo modo, il compratore può acquistare al prezzo d'esercizio (inferiore) e rivendere a quello di mercato (superiore) e avere un payoff positivo, in caso contrario il compratore non eserciterà l'opzione e avrà un payoff negativo corrispondente al premio pagato per l'opzione.

Si supponga di identificare con K lo strike price e con S_t il prezzo finale del titolo sottostante il payoff del compratore dell'opzione call potrà allora essere rappresentato nel seguente modo:

$$Max(S_t - K, 0)^{27}$$

Quindi solo nel caso in cui S_t sia strettamente superiore a K l'opzione verrà esercitata e il guadagno sarà uguale a S_t-K, altrimenti non verrà esercitata e il payoff sarà uguale a zero.

Il grafico sottostante permette di capire meglio il funzionamento di un'opzione call:

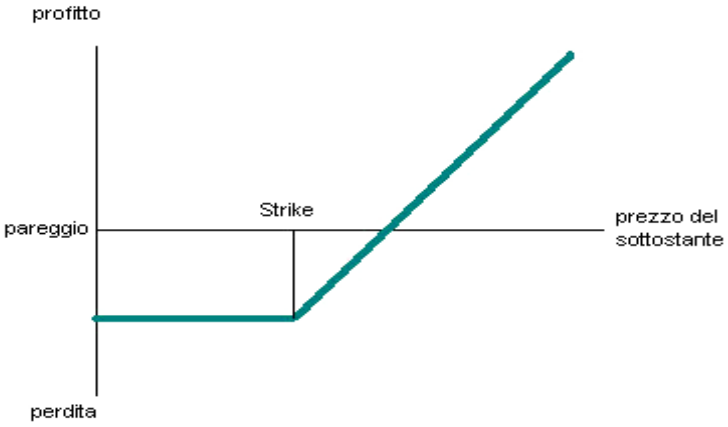


Grafico relativo al profitto di un compratore (buyer) di un'opzione call[28]

Sull'asse delle ascisse si può osservare il prezzo del sottostante mentre sull'asse delle ordinate il profitto. La linea azzurra corrisponde al payoff del compratore dell'opzione. Il payoff corrisponderà ad una perdita finché il prezzo corrente del sottostante sarà inferiore allo strike price, in questo tratto l'opzione call si dice *out of the money*.

---

[27] Quando si sintetizza il profitto di un'opzione call non si tiene conto del premio che va pagato infatti il profitto dovrebbe esser S_t-K-P dove P è il valore del premio pagato; è utile ricordare che la differenza tra prezzo del sottostante e strike price viene anche chiamato **valore intrinseco** di un'opzione call.

[28] (Narciso, 2014)

Quando lo strike price sarà all'incirca uguale al prezzo del sottostante il possessore sarà indifferente tra l'esercitare l'opzione o meno, in questo caso si dice che l'opzione è *at the money.*

La parte della linea azzurra che ha un valore positivo indica la parte in cui il payoff dell'opzione è positivo e quindi al possessore risulterà conveniente esercitarla, in questo caso l'opzione si dice *in the money.*

Dalla figura precedentemente proposta si evince che:

- più il prezzo del titolo sottostante aumenta più le possibilità per il compratore della opzione call di ottenere un payoff positivo aumenteranno poiché se il prezzo del sottostante (S) a scadenza sarà superiore del prezzo strike allora l'opzione verrà esercitata, il payoff positivo ottenibile può essere potenzialmente illimitato;
- più il prezzo del titolo sottostante diminuisce più le possibilità per il compratore di incorrere in un potenziale payoff negativo (perdita) aumenteranno poiché se il prezzo del sottostante, a scadenza, sarà inferiore del prezzo strike l'opzione non verrà esercitata quindi la perdita sarà limitata al premio pagato.

Una opzione call quindi verrà presumibilmente acquistata ogni qual volta si ritiene che il prezzo del titolo sottostante sia destinato a salire.

Per quanto riguarda, invece, colui che scrive l'opzione call (ossia colui che vende l'opzione) riceverà dal compratore un premio (premium) *certo* e *immediato* alla stipulazione del contratto di opzione ma dovrà essere pronto a vendere qualora l'opzione sia esercitata dal compratore.

In termini grafici il venditore di una call avrà il seguente payoff:

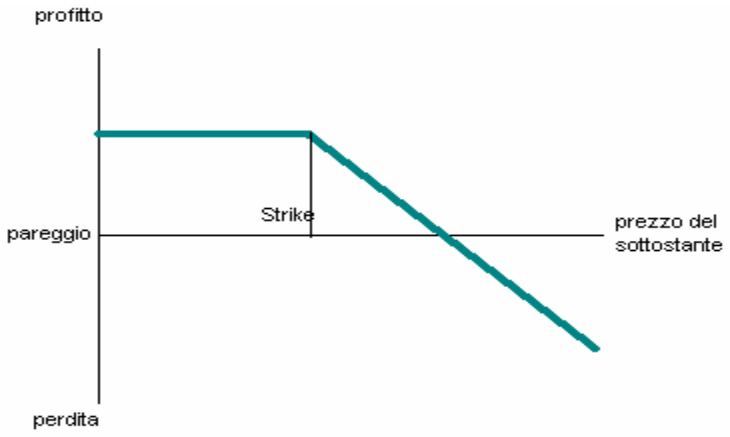


Grafico relativo al payoff di un venditore di un'opzione call[29]

[29] (Narciso, 2014)

Le due posizioni che si possono venire a creare per quanto riguarda il venditore nel caso di una variazione del prezzo del sottostante sono le seguenti:

- più il prezzo del sottostante diminuisce più le possibilità per il venditore di ottenere un payoff positivo (profitto) aumenteranno poiché se il prezzo del sottostante (S) sarà inferiore dello strike price (K) alla data di scadenza dell'opzione allora l'opzione non verrà esercitata dal compratore e il venditore avrà un profitto positivo pari al premio;
- più il prezzo del sottostante aumenta più le possibilità per il venditore di ottenere un payoff negativo (perdita) aumenteranno poiché se il prezzo del sottostante (S) a scadenza sarà superiore allo strike price (K) l'opzione verrà esercitata e il venditore avrà una perdita potenzialmente infinita.

Quindi, ricapitolando, un'opzione call verrà venduta qualora ci si aspetti che il prezzo del sottostante diminuisca.

Le **opzioni put**, invece, sono dei contratti che danno al compratore il diritto, ma non l'obbligo, di vendere un titolo sottostante in una specifica quantità, in un determinato periodo di tempo e ad una specifica data ad un prezzo (strike price) che è fissato nel contratto corrispondendo un premio al venditore dell'opzione alla stipulazione del contratto.
In questo caso il payoff del compratore di una put option sarà il seguente:

$$Max(K - S_t, 0)$$[30]

Il grafico che sintetizza il profitto che ottiene il compratore di un'opzione put è il seguente:

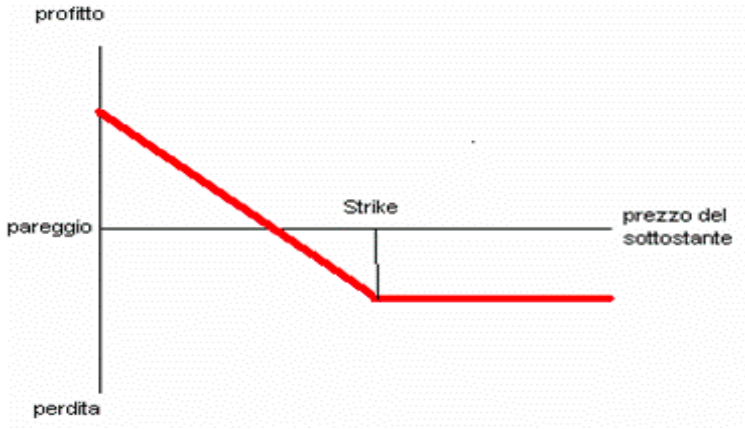


Grafico del profitto di un compratore di un'opzione put[31]

[30] Anche qui non si tiene conto del premio (P) corrisposto al venditore della Put Option, infatti il profitto sarà positivo solo se $K-S_t-P$ è positivo, la differenza tra strike price e prezzo del sottostante viene anche detto **valore intrinseco** di un'opzione put

[31] (Narciso, 2014)

Come per l'opzione call sull'asse delle ascisse si può osservare il prezzo del sottostante mentre sull'asse delle ordinate il payoff dell'opzione. Il possessore dell'opzione deciderà di esercitarla solo nel caso in cui il prezzo del sottostante sia più basso dello lo strike price, in questo caso l'opzione put sarà detta *in the money*. Se il prezzo del sottostante sarà uguale o comunque molto vicino allo strike price si avrà un'opzione put *at the money* e quindi, come spiegato in precedenza, il possessore dell'opzione put è indifferente all'esercizio o meno dell'opzione stessa. Se il prezzo del sottostante sarà superiore allo strike price allora l'opzione sarà detta *out of the money*, la perdita massima che il compratore potrà accumulare sarà uguale al premio pagato (parte orizzontale della linea rossa).
Quanto sopra esposto si può riassumere come segue:

- più il prezzo del titolo sottostante diminuisce più le possibilità di ottenere a scadenza un payoff positivo aumentano visto che se a scadenza il prezzo strike è superiore al prezzo del sottostante l'opzione verrà esercitata e il compratore avrà un profitto potenzialmente illimitato;
- più il prezzo del titolo sottostante aumenta più le possibilità di ottenere a scadenza un payoff negativo aumentano visto che se a scadenza il prezzo strike è inferiore al prezzo del sottostante l'opzione non verrà esercitata e il compratore avrà un payoff negativo pari al premio.

Si sottoscriverà, quindi, una opzione put solo nel caso in cui si ritenga che il titolo sottostante sia destinato a diminuire nel tempo.
Per quanto riguarda chi scrive l'opzione put (colui che la vende) riceverà un premio (premium) ma deve rimanere pronto a comprare allo strike price qualora l'opzione venga esercitata.

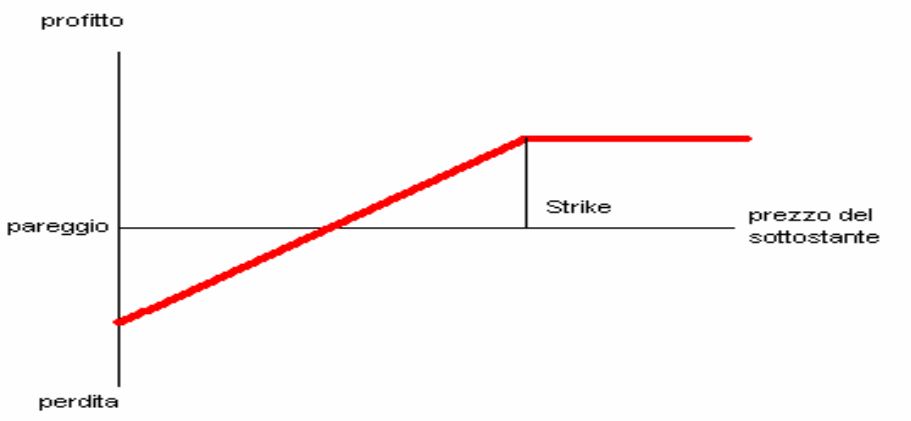


Grafico del profitto di un venditore di un'opzione put[32]

[32] (Narciso, 2014)

Le due posizioni che si possono venire a creare per quanto riguarda il venditore dell'opzione put nel caso di una variazione del prezzo del sottostante sono le seguenti:

- più il prezzo del sottostante aumenta più le possibilità di ottenere, a scadenza, un payoff positivo aumentano visto che se a scadenza il prezzo del sottostante è maggiore dello strike price allora il compratore dell'opzione put non la eserciterà, il profitto massimo del venditore è comunque uguale al premio;
- più il prezzo del sottostante diminuisce più le possibilità di ottenere, a scadenza, un payoff negativo aumentano visto che se a scadenza il prezzo del sottostante è inferiore allo strike price allora il compratore eserciterà l'opzione e il venditore avrà una perdita potenzialmente infinita.

In questa sede è utile ricordare che tutte e due le categorie di opzioni appena descritte possono avere numerose tipologie di sottostante che possono spaziare dai titoli, ai Bond, alle valute, a determinati indici finanziari e anche a diverse classi di derivati.

2.2 Modello binomiale per il calcolo del prezzo di un'opzione europea

Il modello binomiale per il calcolo del prezzo di un'opzione è il modello più utilizzato, questo è dovuto ad alcuni vantaggi che esso possiede: è un modello molto semplice da implementare, è adatto a numerosi (e molto spesso complicati) problemi di calcolo del prezzo di un'opzione e utilizza delle formule matematiche molto più semplici rispetto a quelle utilizzate negli altri modelli.

Questo modello è stato parzialmente sviluppato da William Sharpe[33] nel 1978 nel suo libro "*Investments*", la formulazione definitiva, che ancora oggi si utilizza, è stata formulata da J. Cox, S. Ross e M. Rubinstein i quali scrissero sul Journal of Financial Economics nel 1979 un importante articolo intitolato "*Option pricing: A simplified Approach*"[34].

Il modello che si andrà ad esporre ha delle assunzioni di base:

- mercato perfetto (nessuna tassa o costi di transazione)
- non ci sono pagamenti di dividendi lungo la vita dell'opzione
- arbitraggio non possibile
- tutti gli asset sono infinitamente divisibili

[33] Economista statunitense, premio Nobel per l'economia nel 1990

[34] (Cox, Ross, & Rubinstein, 1979)

Oltre a queste assunzioni di base il modello binomiale deve rispettare l'ipotesi di *neutralità al rischio*[35] che permette di supporre che tutti gli investitori siano indifferenti al rischio e che non chiedano un extra-rendimento per aver investito in un'attività rischiosa.

Questa ipotesi è molto importante perché permette di poter utilizzare il tasso di interesse privo di rischio sia per calcolare il prezzo di un'opzione sia per la valutazione dei futuri flussi di cassa (cash flows).

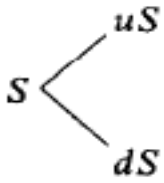
I modelli binomiali sono caratterizzati da una dinamica *discreta* del sottostante in cui quindi si calcola la variazione del valore del titolo solo in un numero limitato di istanti nel tempo di lunghezza δt (questi valori devono essere numerabili).

In questo testo si considereranno opzioni europee derivanti da sottostanti composti da titoli azionari che non distribuiscono dividendi.

Si deve ipotizzare quindi che le contrattazioni avvengano su istanti discreti ed è necessario che il prezzo del sottostante segua un processo binomiale moltiplicativo.

Il primo caso che si andrà a trattare è quello del modello binomiale **uniperiodale**.

Supponiamo di porre il prezzo iniziale del nostro sottostante uguale a S, il prezzo alla fine del primo periodo avrà due possibili valori uS e dS di solito u>1 e d<1, il modello è illustrato qui di seguito:



Questa figura sintetizza i movimenti del prezzo dell'azione al tempo δt

Il movimento da S a uS è un movimento verso l'alto. Il movimento da S a dS viene denominato invece movimento verso il basso. La probabilità con cui si presenterà il movimento verso l'alto viene indicata con la lettera *q*. Il movimento verso il basso avrà probabilità *q-1*.

Ponendo poi *rf* come il tasso privo di rischio, sarà necessario che sia soddisfatta la seguente disuguaglianza:

$$u > (1 + rf) > d$$

[35] È importante sottolineare che questa ipotesi è alquanto irrealistica nella realtà dei mercati finanziari, infatti, gli investitori non saranno mai neutrali al rischio.

Se questa disuguaglianza non fosse rispettata si potrebbero avere delle opportunità di arbitraggio tali da coinvolgere il tasso privo di rischio e il titolo sottostante (questa fattispecie di arbitraggio viene chiamato *arbitraggio privo di rischio*).

Si può iniziare considerando un'opzione call che scade nel periodo successivo (t+1) ossia un modello binomiale uniperiodale, ponendo il suo prezzo corrente uguale a C, C_u sarà il prezzo della call se il prezzo del sottostante avrà un movimento rialzista verso uS e C_d invece se il prezzo del sottostante subirà un movimento ribassista dS.

Per quanto detto prima riguardo al contratto d'opzione si può desumere che:

- $C_u = Max(uS - K, 0)$ con probabilità q
- $C_d = Max(dS - K, 0)$ con probabilità *q-1*

È possibile calcolare il prezzo di una opzione call dimostrando che esiste un portafoglio di bond e azioni che replicano esattamente il *payoff* dell'opzione.

Si supponga quindi di avere un portafoglio con un numero X di titoli e un ammontare pari a B di titoli privi di rischio con un rendimento pari a *r=(1+rf)*. Questo portafoglio all'inizio del periodo costerà XS+B e il valore complessivo del portafoglio alla fine del periodo sarà:

- $XuS + rB$ con probabilità q
- $XdS + rB$ con probabilità *q-1*

Potendo decidere i quantitativi di X e di B sarà possibile porli in modo tale che il valore alla fine del periodo sia uguale al valore garantito dell'opzione call:

$$\begin{cases} XuS + rB = C_u \\ XdS + rB = C_d \end{cases}$$

Risolvendo il sistema di due equazioni in due incognite (X e B) è possibile ottenere il numero di azioni e il numero di titoli risk-free da inserire nel portafoglio per replicare il valore dell'opzioni sia nello scenario di un movimento verso il basso sia in quello verso l'alto:

$$X = \frac{C_u - C_d}{(u - d) * S}$$

$$B = \frac{uC_d - dC_u}{(u - d) * r}$$

In condizioni di non arbitraggio il valore corrente dell'opzione (C) deve coincidere con il prezzo del portafoglio e quindi:

$$C = XS + B = \frac{C_u - C_d}{(u-d)} + \frac{uC_d - dC_u}{(u-d)*r} = \frac{\left[\left(\frac{r-d}{u-d}\right)C_u + \left(\frac{u-r}{u-d}\right)C_d\right]}{r}$$

Per semplificare ulteriormente la formula si pone:

$$p = \frac{r-d}{u-d} \text{ e } 1-p = \frac{u-r}{u-d}$$

Quindi si può riscrivere la formula precedente nel seguente modo:

$$C = \frac{[p * C_u + (1-p) * C_d]}{r}$$

Questa formula ha alcuni aspetti molto importanti:

- la probabilità *q* non appare nella formula e questo significa che anche se differenti investitori hanno diverse aspettative per quanto riguarda movimenti in aumento o in diminuzione del sottostante essi potrebbero comunque essere d'accordo sulla relazione esistente tra C, S, u, d e r;
- il valore dell'opzione call non dipende dalla propensione al rischio ma solo dall'ipotesi che si cerchi di trarre vantaggio dalle possibili opportunità di arbitraggio[36] senza rischio;
- l'unica variabile causale che determina poi il valore dell'opzione è il prezzo del sottostante.

È possibile che la propensione al rischio e le caratteristiche di altri titoli possano influenzare il prezzo dell'opzione ma solo indirettamente cioè influenzando le variabili presenti nella formula sopra citata.

Il modello uniperiodale appena descritto è molto semplice per la valutazione di opzioni ma per alcuni aspetti anche irrealistico, infatti è forte l'assunzione fatta in quanto si presume che ci siano solo due scenari possibili che possano essere ottenuti dal titolo sottostante.

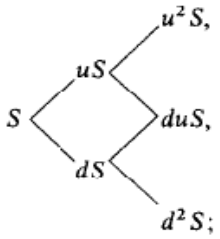
Proprio per questo motivo, il modello uniperiodale è stato usato da J. Cox, S. Ross e M. Rubinstein come base di partenza per l'implementazione di un nuovo modello binomiale più realistico e completo: quello **multiperiodale**.

Si può ora considerare il caso più semplice di modello binomiale multiperiodale ossia quello composto da due periodi.

---

[36] Effettuare un arbitraggio vuol dire mettere in atto una serie di operazioni che permettano di ottenere un guadagno certo e immediato.

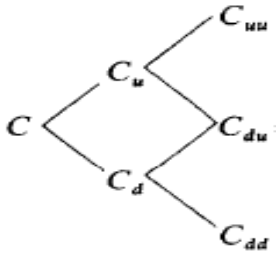
Si deve partire con i valori che può assumere il sottostante, ci sono tre possibilità:



Mentre i valori che può assumere il contratto di opzione, a seconda del valore del sottostante, sono:

- $C_{uu} = \max[u^2S - K, 0]$
- $C_{du} = \max[duS - K, 0]$
- $C_{dd} = \max[d^2S - K, 0]$

Graficamente si ottiene questa struttura ad albero tipica del modello binomiale:



Il primo caso (C_{uu}) rappresenta un'opzione call che dal primo periodo si muove sempre verso l'alto, nel secondo caso (C_{du}) l'opzione subirà un movimento verso il basso ed uno verso l'alto e nell'ultimo caso (C_{dd}) si avranno due movimenti verso il basso.
Nel modello multiperiodale occorrerà procedere partendo dai valori osservati al periodo T (in questo caso 2) e andare indietro fino a trovare il valore corrente dell'opzione. Come già visto nel modello uniperiodale, i valori intermedi C_u e C_d algebricamente possono essere trovati nel seguente modo:

- $C_u = \frac{[p*C_{uu}+(1-p)*C_{ud}]}{r}$
- $C_d = \frac{[p*C_{du}+(1-p)*C_{dd}]}{r}$

Una volta trovati questi valori si deve applicare nuovamente questa formula per trovare il valore attuale dell'opzione (C).

La formula generale che permette di ricavare il valore attuale di un'opzione call con un numero qualsiasi di periodi *n* è la seguente:

$$C = \frac{\left[\sum_{j=0}^{n}\left(\frac{n!}{j!\,(n-j)!}\right) * p^{j} * (1-p)^{n-j} \max\left[u^{j}d^{n-j}S - K, 0\right]\right]}{r^{n}}$$

Ora bisogna capire meglio come determinare alcuni parametri che risultano fondamentali per il calcolo del prezzo delle opzioni utilizzando il modello binomiale, i parametri in questione sono: *p, u,* e *d.*

Questi parametri devono rispecchiare il valore della media e della varianza del cambiamento del prezzo del sottostante lungo un intervallo di tempo di lunghezza δt, come si è visto precedentemente si sta operando nel rispetto dell'ipotesi di neutralità al rischio è quindi necessario che il tasso d'interesse utilizzato sia il tasso privo di rischio *rf.*

Si può imporre che il valore atteso del sottostante alla scadenza dell'intervallo δt sia uguale a $Se^{rf\delta t}$ in altre parole si avrà:

$$Se^{rf\delta t} = puS + (1-p)dS$$

La formula può essere semplificata eliminando sia a destra che a sinistra del simbolo di uguaglianza il valore S:

$$e^{rf\delta t} = pu + (1-p)d$$

Da cui si deduce:

$$p = \frac{e^{rf\delta t} - d}{u - d}$$

Se si vuole misurare le variazioni del sottostante in un dato intervallo di tempo è necessario calcolare la varianza. L'intervallo preso in considerazione è di lunghezza δt e quindi la varianza sarà uguale a $\sigma 2\delta t$ e visto che la varianza di una variabile qualsiasi M è definita come $E(M^2) - [E(M)]^2$ dove E rappresenta il valore atteso allora possiamo riscrivere la forma sopra citata nel seguente modo:

$$pu^2 + (1-p)d^2 - [pu + (1-p) * d]^2 = \sigma^2\delta t$$

Sostituendo il valore p nella formula precedente si ottiene:

$$e^{rf\delta t}(u+d) - ud - e^{2rf\delta t} = \sigma^2 \delta t$$

Nel 1979 Cox, Ross e Rubisten hanno proposto, come soluzione alla formula precedente il seguente risultato:

$$u = e^{\sigma\sqrt{\delta t}} \text{ e } d = \frac{1}{u} = e^{-\sigma\sqrt{\delta t}}$$

Dove il simbolo σ identifica la volatilità del prezzo del sottostante nell'intervallo δt.
In conclusione, quando si vuole utilizzare il modello binomiale sia uniperiodale che multiperiodale per essere in grado di prezzare il sottostante di un'opzione e poi l'opzione stessa è necessario conoscere il valore di suddetti parametri.

2.3 Modello di Black-Scholes

Il modello di Black-Scholes permette di superare uno dei limiti del modello binomiale che, come si è detto prima, considera istanti di tempo discreti nella valutazione del prezzo del sottostante mentre il modello di Black-Scholes fa si che si possano considerare infiniti periodi di tempo nel continuo.
Questo modello è precedente a quello di Cox, Ross e Rubisten, infatti fu implementato nel 1973 da Fisher Black e Myron Scholes che pubblicarono il loro lavoro sul "Journal of Political Economy", il loro testo fu intitolato "*The Pricing of options and corporate liabilities*"[37].
Il modello in questione permette di definire e valutare un'opzione sulla base della conoscenza di 5 fattori:

- S(t)= prezzo del titolo sottostante
- K= strike price definito sull'opzione
- rf =tasso risk-free con la scadenza dell'opzione
- T=scadenza dell'opzione
- σ=volatilità del sottostante

[37] (Scholes & Black, 1973)

Il modello di Black-Scholes permette di valutare propriamente, grazie ad un approccio matematico, il prezzo di una opzione europea[38] su un titolo sottostante che non paga dividendi. Il modello è basato su alcune assunzioni molto importanti:

- il mercato è aperto e continuo (titoli vengono negoziati continuamente);
- il mercato è perfettamente efficiente: si ha quando l'insieme delle informazioni riflesse nei prezzi è costituito sia dalle informazioni passate, dalle informazioni pubblicamente disponibili e da informazioni riservate;
- non c'è nessuna opportunità di arbitraggio senza rischio;
- il tasso privo di rischio, *rf*, è costante ed è identico per qualsiasi scadenza, in altre parole la struttura dei tassi di interesse è piatta e deterministica;
- c'è distribuzione log-normale (con media e varianza costanti) del prezzo dell'azione.

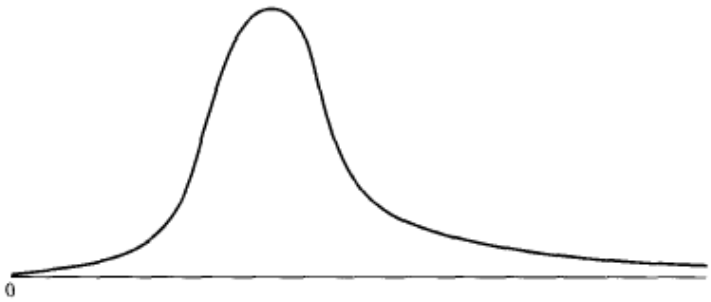
2.3.1 Distribuzione log-normale dei prezzi

L'assunzione di distribuzione log-normale dei prezzi è alla base dell'applicazione del modello di Black-Scholes.

La distribuzione log-normale è una distribuzione statistica non gaussiana di valori ma con proprietà simili alla distribuzione normale gaussiana.

Una variabile che abbia una distribuzione log-normale può assumere qualsiasi valore tra zero e infinito.

Nella figura sottostante è rappresentato un caso classico di distribuzione log-normale[39]:



Definendo:

- μ: il ritorno atteso sul titolo
- σ: la volatilità del prezzo del titolo stesso

[38] Si ricorda che per opzione europea si intende un'opzione che può essere esercitata solo a scadenza

[39] (Hull, 2015)

Si assuma che il prezzo del sottostante (S) sia regolato da un *moto Browniano geometrico* (deve essere seguito dal sottostante perché possa essere utilizzato il modello di Black-Scholes) che viene descritto nel seguente modo:

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

Applicando, poi, il *lemma di Itô* il processo sopra indicato può essere semplificato nel seguente modo:

$$dln(S) = \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dz$$

Visto che μ e σ sono due costanti allora a variazioni del *ln(S)* coincideranno variazioni costanti di $\left(\mu - \frac{\sigma^2}{2}\right)$ e variazioni costanti di σ. Se le variazioni di *ln(S)* fossero tra zero e un tempo futuro uguale a T allora l'equazione sopra descritta avrebbe una distribuzione normale con:

- Media: $\left(\mu - \frac{\sigma^2}{2}\right)T$
- Varianza: $\sigma^2 T$

Si avrà quindi come conseguenza che il logaritmo del sottostante a scadenza sarà normalmente distribuito infatti:

$$lnS_T \sim N[lnS_0 + \left(\mu - \frac{\sigma^2}{2}\right)T, \sigma\sqrt{T}]$$

Dove S_T rappresenta il prezzo futuro del titolo al tempo *T*, S_0 il prezzo del titolo al tempo zero e $N(\mu, \sigma^2)$ indica una distribuzione di tipo normale con media μ e varianza σ^2.

L'equazione sopra citata mostra che $ln(S_T)$ è normalmente distribuito, quindi S_T ha una distribuzione log-normale[40].

Una distribuzione log-normale ha i valori di media, mediana e moda che differiscono tra di loro ma, applicando le proprietà della distribuzione stessa, è possibile osservare che il valore atteso di S_T sarà uguale a:

$$E(S_T) = S_0 e^{\mu T}$$

[40] Questa è una delle caratteristiche della distribuzione log-normale, infatti se il logaritmo di una variabile è distribuito in modo normale allora la variabile stessa presenta una distribuzione log-normale

mentre la varianza di S_t sarà uguale a:

$$var(S_T) = S_0^2 e^{2\mu T}(e^{\sigma^2 T} - 1)$$

Alcune proprietà della distribuzione log-normale ci permettono di trarre delle conclusioni sulla distribuzione del tasso di rendimento composto in modo continuo del titolo che è possibile ottenere tra zero e T.
Si ha

$$S_T = S_0 e^{\eta T}$$

e

$$\eta = \frac{1}{T}\ln\left(\frac{S_T}{S_0}\right)$$

dove η rappresenta il tasso di rendimento annuo composto in modo continuo relativo al periodo tra zero e T il quale tasso è distribuito in forma normale con media uguale a $\mu - \frac{\sigma^2}{2}$ e deviazione standard uguale a $\frac{\sigma}{\sqrt{T}}$.

2.3.2 L'equazione differenziale fondamentale

Il moto che il prezzo S(t) del sottostante deve seguire, come detto prima, è il moto Browniano[41] geometrico (detto *geometrico* perché, quando si calcola la media geometrica, i valori si moltiplicano e non si sommano infatti μ *e* σ sono moltiplicati per S). Il moto Browniano è un modello probabilistico che, in principio, veniva usato per descrive l'evoluzione nel tempo di fenomeni fisici poi è stato anche utilizzato, come per il modello di Black-Scholes anche per molti altri modelli di tipo economico-matematico.
Il moto Browniano, utilizzato per il modello di Black-Scholes, permette di rappresentare l'evoluzione nel tempo del prezzo di azioni e viene descritto dall'equazione differenziale stocastica:

$$dS(t) = \mu S(t)dt + \sigma S(t)dZ(t) \qquad (1)$$

[41] Questo modello fu implementato da Robert Brown (1773-1858)

Nella formula generale i coefficienti μ e σ sono due valori costanti e sono definiti: il primo "*drift*" e il secondo "*volatility*". Il drift può essere anche descritto come il valore istantaneo atteso di variazione in un'unità di tempo.

Il termine *Z(t)* è ciò che rende la nostra equazione *stocastica*, questo temine è molto utilizzato nel calcolo delle probabilità ed ha lo stesso significato dei termini casuale, aleatorio. Un metodo stocastico, quindi, viene sviluppato per cercare di risolvere problemi di natura probabilistica casuale. L'esatto contrario di ciò che succede in un processo *deterministico*.

È possibile semplificare la formula sopra citata dividendo entrambi i membri per S, la formula quindi viene modificata nel seguente modo:

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

Il rapporto presente a sinistra dell'equazione corrisponde al saggio istantaneo di rendimento dell'azione, è utile ricordare che il valore atteso è uguale a:

$$E\left[\frac{dS(t)}{S(t)}\right] = \mu\, dt$$

questo perché la media di Z(t) è nulla.

La varianza invece può essere rappresentata nel seguente modo:

$$V\left[\frac{dS(t)}{S(t)}\right] = \sigma^2 dt$$

Il rendimento atteso è il rendimento che viene richiesto dagli investitori a seconda della rischiosità del titolo, di solito più alto è il rischio più alto è il rendimento richiesto dagli investitori. Fortunatamente non ci si deve preoccupare troppo della determinazione di questo valore perché, una volta che si esprime l'opzione come funzione del valore del titolo sottostante, il valore μ scompare dalla formula.

Il risultato μdt si ottiene solo per intervalli di tempo molto piccoli, quando si passa ad analizzare il rendimento atteso composto in modo continuo su periodi di tempo più lunghi (per esempio T anni) il rendimento atteso è rappresentato dalla formula:

$$\frac{1}{T} * ln\frac{S_t}{S_0}$$

La volatilità del prezzo del titolo, σ, misura l'incertezza che si possa verificare o meno un certo risultato atteso da un particolare titolo. I titoli di solito hanno una volatilità che si va a collocare tra il 20% e il 50%.
Nell'introdurre la formula di Black-Scholes bisogna tenere in considerazione il prezzo C di una opzione call il quale deve essere funzione del prezzo del sottostante S e del tempo t, come rappresentato nella formula seguente:

$$C(t) = C(S_t, t)$$

Si deve presupporre che la funzione sopra esposta sia continua rispetto alle variabili *S* e *t*, supponendo ciò è possibili applicare il *lemma di Itô*[42] a C ottenendo dunque:

$$dC(t) = \left(\frac{\partial C}{\partial S}\mu S + \frac{\partial C}{\partial t} + \frac{1}{2}\frac{\partial^2 C}{\partial S^2}\sigma^2 S^2\right)dt + \frac{\partial C}{\partial S}\sigma S dZ \qquad (2)$$

A questo punto è necessario creare un portafoglio, che verrà chiamato Π tale per cui sia possibile combinare assieme una certa quantità di titoli e opzioni call che permetta di eliminare il termine stocastico *dZ*.
Questo è possibile solamente avendo un portafoglio uguale a:

$$\Pi = -C + \frac{\partial C}{\partial S}S$$

Ossia un portafoglio composto da -1 opzioni (posizione corta) e una porzione uguale a $\frac{\partial C}{\partial S}$ di titoli (posizione lunga) del corrispettivo sottostante. Se si trasforma il portafoglio in modo tale da rappresentare la variazione del valore del portafoglio in un intervallo di tempo *dt* si ottiene il seguente risultato:

$$d\Pi = -dC + \frac{\partial C}{\partial S}dS$$

A questo punto sostituendo l'equazione (1) e l'equazione (2) ai valori di questa equazione è possibile ottenere la seguente formula:

[42] Utilizzato nel calcolo stocastico al fine di computare il differenziale di una funzione

$$d\Pi = \left(-\frac{\partial C}{\partial t} - \frac{1}{2}\frac{\partial^2 C}{\partial S^2}\sigma^2 S^2 \right) dt$$

È possibile osservare come in quest'ultima equazione non sia più presente il termine *dZ*. L'equazione sopra esposta, inoltre, mostra come le variazioni del portafoglio non dipendano più da μ, questo perché la componente che rappresenta l'opzione call è controbilanciata perfettamente dall'effetto che ha μ sulla componente che identifica il sottostante.
L'indipendenza dal termine *dZ* permette di rimuovere dal portafoglio così creatosi il termine stocastico ossia la parte casuale rendendo quindi la formula deterministica.
Questo ha un'altra implicazione molto importante infatti fa si che il portafoglio debba necessariamente offrire, come rendimento, il tasso privo di rischio *rf*.
Infatti, i portafogli che offrono rendimenti maggiori di *rf* sono portafogli portatori di rischio ma, in questo caso, si è in assenza di rischio per l'eliminazione della variabile stocastica.
Questa operazione di eliminazione della variabile casuale (portatrice di rischio) in finanza si chiama *hedging*, più nello specifico *delta hedging* che rappresenta una strategia che riduce al minimo il rischio associato al movimento di prezzo del sottostante (S).
Questo porta alla conclusione che la variazione ottenuta nel portafoglio debba essere uguale all'aumento che potremmo ottenere se prestassimo la stessa quantità iniziale ad una banca.
La variazione del valore del portafoglio deve essere quindi la seguente:

$$d\Pi = rf\Pi dt \tag{3}$$

Quindi la variazione nel portafoglio deve essere uguale al tasso privo di rischio moltiplicato per il valore attuale del portafoglio (Π) e il piccolo intervallo di tempo (dt).
Combinando le tre equazioni 1, 2 e 3 si può ottenere la seguente equazione:

$$\frac{\partial C}{\partial t} + rfS\frac{\partial^2 C}{\partial S^2}\sigma^2 S^2 = rfC \tag{4}$$

L'equazione così ottenuta viene anche definita "*equazione differenziale di Black-Scholes-Merton*" che ha un numero molto elevato di soluzioni corrispondenti alle differenti derivate che possono essere definite a seconda del valore di S e quindi del sottostante. La risoluzione di questa equazione non dura nel tempo proprio perché i valori assunti da S e t possono cambiare alterando anche il valore $\frac{\partial C}{\partial t}$. Per continuare ad essere privo di rischio il portafoglio ha bisogno di un continuo aggiustamento delle rispettive quote.

Si può cercare di circoscrivere le soluzioni che si possono ottenere ponendo delle condizioni dette di "*confine*" (boundary conditions) che definiscono il valore del derivato per valori estremi di *S* e *t.*

Per quanto riguarda un'opzione call europea la condizione di confine è la seguente:

$$C(T) = \max [S(T) - K; 0] \quad (5)$$

Questo è il classico caso di un'opzione call a scadenza in cui K rappresenta lo strike price.

2.3.3 La valutazione neutrale del rischio

Si è già parlato in precedenza della valutazione neutrale del rischio parlando del modello binomiale, questo concetto è uno dei più importanti strumenti per l'analisi di derivati.

Questo concetto nasce da una proprietà fondamentale dell'equazione differenziale di Black-Scholes-Merton (4) e dipende dal fatto che nell'equazione non sono coinvolte variabili che siano condizionate dalle preferenze di rischio degli investitori. Le variabili che compaiono sono: il prezzo corrente del sottostante, il tempo, la volatilità e il tasso privo di rischio che sono tutte variabili indipendenti dalle preferenze degli investitori stessi.

Questo non sarebbe più veritiero nel caso in cui fosse incluso il rendimento atteso del sottostante (μ) poiché quest'ultimo dipende dalle preferenze sul rischio infatti più alta è l'avversione al rischio maggiore deve essere il rendimento atteso.

Il fatto che l'equazione differenziale sia priva di rischio fa si che possa essere fatta qualsiasi assunzione sulla propensione al rischio degli investitori, è possibile anche assumere che tutti gli investitori siano neutrali al rischio.

Se si potesse assumere una situazione come quella sopra descritta vorrebbe dire che il tasso di rendimento atteso da tutti i titoli dovrebbe essere uguale al tasso privo di rischio, *rf*, perché gli investitori non avrebbero bisogno di alcun premio per il rischio.

Un ulteriore risvolto che questo potrebbe avere è il fatto che tutte le attualizzazioni per ricavare i valori futuri possono essere fatti prendendo in considerazione il tasso privo di rischio, lo stesso può essere fatto per ricavare i payoff attesi a scadenza.

È utile ricordare, però, che questa è una semplificazione della realtà, l'equazione di Black-Scholes ha soluzioni anche senza questa assunzione. Quando si passa da una situazione in cui gli investitori sono neutrali al rischio ad una situazione in cui gli investitori possono essere avversi al rischio due cose succedono:

- cambia il tasso di rendimento atteso del titolo;
- cambia il tasso di interesse per l'attualizzazione.

I due effetti si compensano esattamente tra loro.

2.3.4 La formula di Black-Scholes

Prendendo a riferimento le formule 4 e 5 è possibile scrivere la formula conclusiva di Black-Scholes cioè quella utilizzata per il calcolo di un'opzione call europea con un sottostante che non paga dividendi:

$$C = SN(d_1) - Ke^{-rfT}N(d_2)$$

Una variante della formula sopra citata può essere utilizzata per calcolare anche il prezzo di un'opzione put, la formula da utilizzare è la seguente:

$$P = Ke^{-rfT}N(-d_2) - SN(-d_1)$$

dove S è il prezzo dell'opzione al tempo zero e K è lo strike price, rf il tasso privo di rischio composto in modo continuo e T è la vita residua, i valori d_1 ed_2 sono uguali a:

$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

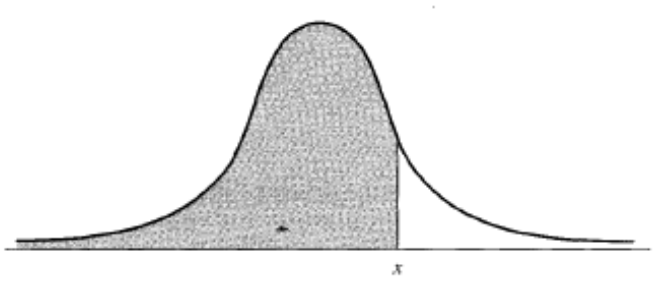
$$d_2 = \frac{\ln\left(\frac{S}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

e l'espressione N(x)[43] rappresenta la funzione della distribuzione normale standard (con media nulla e deviazione standard unitaria) definita come N(0,1), la sua funzione di densità è la seguente:

$$N(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$$

Ed è rappresentata nel grafico seguente:

[43] La funzione N(x) che rappresenta una funzione di distribuzione cumulativa spesso viene determinata utilizzando o delle tabelle che approssimano il valore oppure con la funzione DISTRIB.NORM in Excel.



L'area evidenziata rappresenta il fattore N(x) della formula sopra esposta

Nello specifico però $N(d_1)$ rappresenta la *pendenza* della curva di Black-Scholes in funzione di S che rappresenta la sensibilità del prezzo dell'opzione rispetto a movimenti di prezzo del sottostante. $N(d_2)$, invece, rappresenta la *probabilità* con la quale l'opzione verrà esercitata a scadenza ossia la probabilità neutrale al rischio sotto le assunzioni fatte col modello di Black-Scholes.

Sia $N(d_1)$ che $N(d_2)$ devono assumere valori che siano positivi e compresi tra 0 e 1.

La formula di Black-Scholes permette di identificare il valore di una opzione call o di una opzione put di tipo europeo in funzione di dati contrattuali come sono quelli inerenti al tempo (T) e allo strike price (K) e in funzioni a dati osservabili sul mercato come quelli del prezzo del titolo sottostante (S) e il tasso privo di rischio (rf). Infine, è necessario ricavare il valore della volatilità del sottostante (σ) che di solito è stimata utilizzando tecniche statistiche su serie storiche o mediante l'inversione numerica della funzione prezzo. Proprio per come è osservata la volatilità quest'ultima è spesso definita come *volatilità implicita*.

Esempio

Ora verrà proposto un esempio utilizzando il modello di Black-Scholes per il calcolo di un'opzione call europea su un titolo azionario che non corrisponde dividendi.

In t=0 si ha un titolo con payoff in T[44], si hanno i seguenti dati:

- prezzo del sottostante (S) in t=0 è uguale a 100;
- la durata (T) è di un anno ossia 365 giorni;
- strike price (K) è uguale a 100;
- volatilità (σ) è uguale a 20% su base annua;
- tasso privo di rischio (r) è uguale a 6%.

$$C(0) = S(0)N(d_1) - Ke^{-rfT}N(d_2) = 100N(d_1) - 100e^{-0.06*1}N(d_2)$$

[44] Si ricorda che la struttura del payoff di un'opzione call europea è la seguente: $\text{Max}[S(T) - K, 0]$

$$Ke^{-rfT} = 100 * e^{-0,06*1} = 94,1764$$

$$d_1 = \frac{\ln\left(\frac{S}{K}\right)+\left(rf+\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} = \frac{\ln\left(\frac{100}{100}\right)+\left(0,06+\frac{0,2^2}{2}\right)}{0,2\sqrt{1}} = 0,4$$

$$d_2 = d_1 - \sigma\sqrt{T} = 0,2$$

$$N(d_1) = 0,65542$$

$$N(d_2) = 0,5792$$

$$C(0) = 100 * 0,65542 - 94,1764 * 0,5792 = 10,9895$$

Il valore della call europea è di 10,9895

Con gli stessi dati è possibile ricavare anche il valore di una opzione put europea:

$$P = Ke^{-rfT}N(-d_2) - SN(-d_1) = 100e^{-0.06*1}N(-d_2) - 100N(-d_1)$$

$$Ke^{-rfT} = 94,1764$$

$$N(-d_1) = 0,34458$$

$$N(-d_2) = 0,4207$$

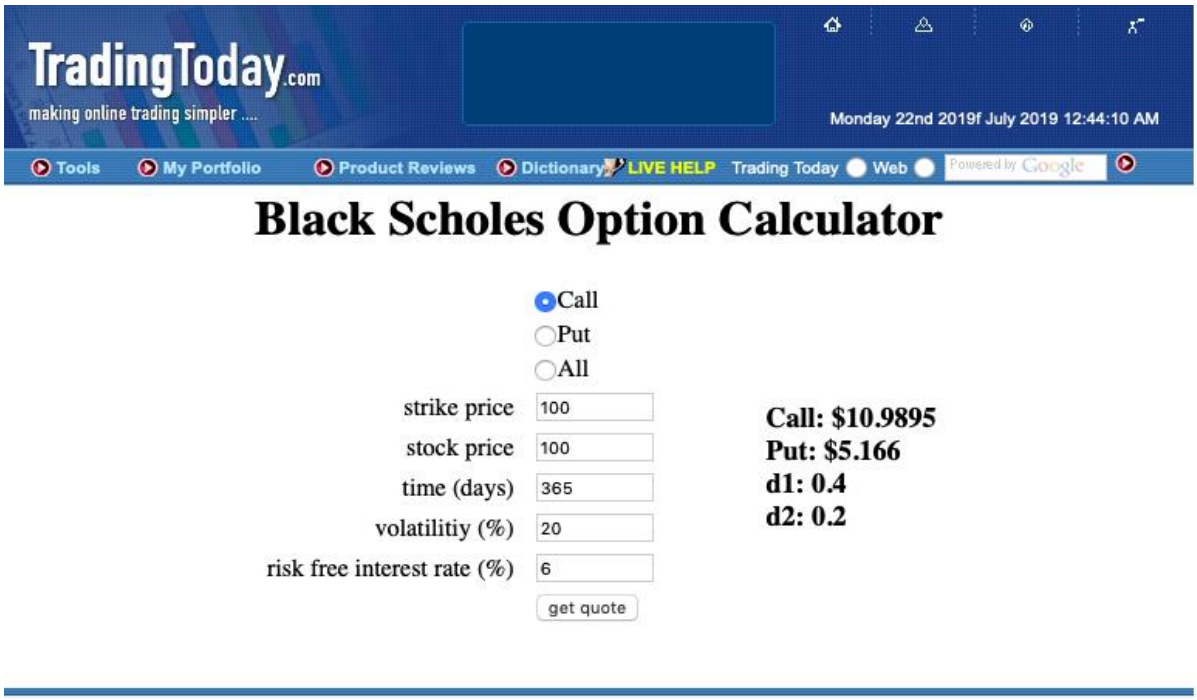
$$P = 94,1764 * 0,4207 - 100 * 0,34458 = 5,162$$

Il valore di una put europea è di 5,1620

Per controllare che i valori ottenuti siano corretti è possibile trovare in rete dei siti che, una volta inserite le informazioni di base (prezzo del sottostante, strike price, volatilità, media, tasso privo di rischio e durata), permettono di calcolare immediatamente e in modo preciso il valore di un'opzione utilizzando la formula di Black-Scholes, uno di questi siti è chiamato "*TradingToday.com*"[45]. Per eseguire tale controllo, di seguito vengono inseriti i dati utilizzati dell'esercizio precedente.

Il risultato è il seguente:

[45] http://tradingtoday.com/black-scholes



2.3.5 Put-Call Parity

Il principio definito come *put-call parity* è molto importante per il pricing delle opzioni europee. Questo principio definisce la relazione esistente tra il prezzo di una put europea e di una opzione call sempre europea dello stesso tipo[46] con lo stesso strike price e la stessa scadenza, il risultato di questa relazione è espresso dalla seguente formula:

$$c + Ke^{-rfT} = p + S_0$$[47]

Questo perché partendo da due portafogli ipotetici il primo formato da una opzione call europea (c) più il valore attuale dello strike price (K) che viene attualizzato al tasso privo di rischio e il secondo costituito da una opzione put (p) più il valore del sottostante (S_0) si può arrivare facilmente alla conclusione che a scadenza i due portafogli avranno lo stesso valore infatti al tempo T varranno entrambi K se $S_T \leq K$ e varranno entrambi S_T se $S_T \geq K$.
In particolare, proprio questa relazione viene chiamata *put-call parity* e permette di poter risalire, in qualsiasi momento, ad una tra l'opzione put o call conoscendo l'altro tipo di opzione. Nel caso in cui questa relazione non fosse rispettata allora esisterebbe una opportunità di arbitraggio che porterebbe alcuni investitori ad ottenere dei guadagni certi ad un tasso privo di rischio (profitti certi ed immediati).

[46] Sia la Put che la Call sono scritte con riferimento allo stesso titolo sottostante

[47] Questa relazione è valida solamente per opzioni di tipo europeo

2.3.6 Le greche

In questo paragrafo verrà esaminata la sensitività che la formula di Black-Scholes ha nei confronti dei suoi parametri: le "*greche*". Le greche, chiamate così perché identificate con lettere greche, misurano ciascuna una diversa dimensione del rischio in una determinata posizione su opzione.
L'obiettivo ultimo degli investitori è quindi quello di gestirle facendo si che tutte rappresentino un livello accettabile di rischio.
La prima di queste lettere è il ***delta*** di un'opzione, identificato col simbolo greco "Δ", definisce la variazione del prezzo di un'opzione alla variazione del prezzo del sottostante. La strategia che si può attuare con il delta è una strategia detta *delta hedging* che consiste nell'acquistare un numero di azioni tali da controbilanciare la perdita derivante dalla vendita delle opzioni: una posizione con delta uguale a zero è detta "neutrale al delta".
È molto importante sottolineare che il delta rimane invariato solamente per brevi istanti di tempo per mantenere inalterata la strategia di delta hedging nel lungo periodo è necessario fare ribilanciamenti periodici delle quote di azioni necessarie perché il rischio sia coperto.
Il delta è espresso dalla derivata parziale del prezzo dell'opzione rispetto al prezzo del sottostante ed è quindi uguale alla pendenza della curva che lega il prezzo dell'opzione al prezzo del sottostante:

$$\Delta = \frac{\partial C}{\partial S}$$

La lettera greca delta è strettamente collegata all'analisi di Black-Scholes, i due studiosi sono riusciti a dimostrare che era possibile formare un portafoglio privo di rischio utilizzando una posizione per l'opzione e un'altra per il titolo sottostante, il portafoglio di Black-Scholes espresso in termini di Δ è il seguente:

$$-1: Opzioni$$
$$+\Delta: azioni\ del\ sottostante$$

Così facendo sono riusciti a creare una posizione neutrale al delta e arrivare alla conclusione che il rendimento atteso di questa posizione dovesse essere il tasso privo di rischio.

Per un'opzione call europea su un sottostante che non paga dividendi delta può essere definito come:

$$\Delta_{call} = N(d_1)$$

$$\Delta_{put} = N(d_1) - 1$$

Il valore $N(d_1)$ è lo stesso utilizzato nella formula di Black-Scholes e rappresenta il numero di azioni del sottostante che andrebbero comprate/vendute a seconda della posizione (lunga o corta) che si vuole assumere per le opzioni call, la stessa cosa vale per le opzioni put solo che il valore in questione del delta è cambiato.

La seconda lettera che si andrà ad esaminare è ***gamma*** che rappresenta la derivata seconda del valore del portafoglio (Π) rispetto al prezzo del sottostante:

$$\Gamma = \frac{\partial^2 \Pi}{\partial^2 S}$$

Se il valore di gamma è piccolo allora delta cambierà lentamente e di conseguenza le modifiche che si dovranno fare al portafoglio perché rimanga neutrale al delta (*delta neutral*) potranno essere meno frequenti. Al contrario, se gamma è grande in valore assoluto allora delta sarà molto sensibile alle variazioni del prezzo del sottostante.

Per un'opzione put o una call su azioni che non pagano dividendi il valore gamma sarà uguale a:

$$\Gamma = \frac{N'(d_1)}{S\sigma\sqrt{T}}$$

La terza lettera greca è ***theta*** (θ) che rappresenta la variazione del valore del portafoglio di derivati (Π) al ridursi della vita residua delle opzioni (detto anche *time decay* o "declino temporale):

$$\theta = \frac{\partial \Pi}{\partial T}$$

Per un'opzione call su un titolo che non paga dividendi si avrà la seguente formula:

$$\theta = -\frac{SN'(d_1)\sigma}{2\sqrt{T}} - rfKe^{-rT}N(d_2)$$

Mentre per un'opzione put su un titolo che non paga dividendi si avrà:

$$\theta = -\frac{SN'(d_1)\sigma}{2\sqrt{T}} + rfKe^{-rT}N(-d_2)$$

Sia per l'opzione put che per l'opzione call il tempo è al denominatore ed è un evento certo che aumenta quindi questo non può che far diminuire il valore di theta più ci si avvicina a scadenza.
La quarta lettera greca che si prenderà in considerazione è ***vega*** (Λ). Si è dato per scontato fino a qui che la volatilità dell'asset sottostante rimanesse costante, nella realtà questo valore cambia col passare del tempo. Questo vuol dire che il valore del derivato è soggetto a variazioni dovute ai movimenti della volatilità cosi come a variazioni del prezzo del sottostante e del passare del tempo.
In altre parole, il vega di un portafoglio di derivati misura il valore del portafoglio stesso rispetto alla volatilità del titolo sottostante:

$$\Lambda = \frac{\partial \Pi}{\partial \sigma}$$

Se il valore di vega è molto alto in termini assoluti allora il valore del portafoglio è molto sensibile a piccole variazioni nella volatilità, se il vega invece è basso in valore assoluto la volatilità del sottostante ha un impatto basso sul valore del portafoglio.
Sia per un'opzione europea call che per una opzione put su un titolo che non paga dividendi il vega è dato dalla seguente formula:

$$\Lambda = S\sqrt{T}N'(d_1)$$

L'ultima lettera greca che si andrà ad analizzare è ***rho*** (ρ) che valuta la sensibilità del valore di un portafoglio di derivati alle variazioni del tasso di interesse privo di rischio (rf):

$$\rho = \frac{\partial \Pi}{\partial rf}$$

Per quanto riguarda una opzione call europea su un titolo che non paga dividendo il valore rho è uguale a:

$$\rho = KTe^{-rfT}N(d_2)$$

Per una opzione put dello stesso tipo invece:

$$\rho = -KTe^{-rfT}N(-d_2)$$

Dove l'espressione N(d_2) fa sempre riferimento alla formula di Black-Scholes.

2.4 Metodo di Montecarlo

La teoria del cosiddetto "*option pricing*" è in continua evoluzione infatti dalla teorizzazione della formula di Black-Scholes e il modello binomiale sono stati teorizzati molti altri modelli per cercare di calcolare il prezzo di derivati e nello specifico di opzioni in modo sempre più flessibile e preciso.
I due modelli che sono stati trattati nei paragrafi precedenti hanno delle limitazioni.
La formula di Black-Scholes porta ad un risultato corretto solamente quando il titolo sottostante segue una distribuzione log-normale questa distribuzione, però, nella vita reale non viene sempre seguita dai titoli sottostanti.
Il modello binomiale permette di superare la limitazione del modello di Black-Scholes per quanto riguarda la distribuzione log-normale però necessita di un movimento, da parte del sottostante, discreto e binomiale.
Il metodo di Montecarlo, invece, permette di valutare un'opzione anche quando il movimento stocastico del sottostante non è continuo, non segue la distribuzione log-normale oppure non ha un movimento discreto e binomiale.
La denominazione *Montecarlo* fu coniata durante la Seconda Guerra Mondiale da J. Von Neumann e S. Ulam mentre lavoravano al *Progetto Manhattan* per la costruzione della bomba atomica.
I due studiosi si ispirarono al famoso casinò di Monte Carlo come nome in codice per il loro progetto, nome che tuttora viene utilizzato.
Von Neumann e Ulam utilizzarono simulazioni di numeri casuali (di cui si è già parlato nel primo capitolo) per generare i parametri delle equazioni che descrivevano la dinamica delle esplosioni nucleari, in questo modo era possibile ottenere soluzioni sempre diverse senza dover

inserire parametri fissi provenienti da dati sperimentali facendo si che il processo potesse essere il più aleatorio possibile.
Il termine "Montecarlo" oggi viene utilizzato come sinonimo di metodo stocastico ossia di un processo casuale e aleatorio, in realtà i numeri *casuali* utilizzati nel metodo di Montecarlo sono comunque da considerarsi pseudocasuali perché generati attraverso il computer con metodi deterministici e quindi non puramente casuali come discusso ampiamente nel primo capitolo.

2.4.1 Valutazione di una opzione call con l'utilizzo del metodo di Montecarlo

Il metodo di Montecarlo viene implementato partendo dalla supposizione che il titolo sottostante segua un *moto Browniano geometrico* quindi:

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

Che poi applicando il *lemma di Itô*, come fatto precedentemente per il modello di Black-Scholes si ottiene:

$$dln(S) = \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dz$$

Dove ln(S) è distribuito in modo normale con media $\left(\mu - \frac{\sigma^2}{2}\right)$ e varianza σ^2 .
È possibile costruire un portafoglio di copertura (hedged) utilizzando una posizione lunga su un'unità di sottostante e una posizione corta di un numero *n* di opzioni call in modo che l'investitore sia neutrale al rischio. Questo fa si che il rendimento atteso del portafoglio sia uguale al tasso privo di rischio (rf) allora è possibile assumere che l'equazione sopra utilizzata sia distribuita in modo normale con una media di $\left(rf - \frac{\sigma^2}{2}\right)$ e una varianza uguale a σ^2.
L'equazione sopra descritta può essere riproposta quindi nel seguente modo:

$$dln(S) = ln\frac{S_T}{S} = \left(rf - \frac{\sigma^2}{2}\right)dt + \sigma dz$$

La parte stocastica (dz) è possibile sostituirla con il valore $\varepsilon\sqrt{dt}$, inoltre è possibile eliminare il logaritmo naturale utilizzando in entrambi i lati dell'equazione il numero di Nepero che modifica la precedente formula in:

$$S_T = S * e^{\left[\left(rf - \frac{\sigma^2}{2}\right)dt + \sigma\varepsilon\sqrt{dt}\right]}$$

dove ε rappresenta una variabile casuale con una distribuzione normale e con media uguale a zero e varianza unitaria.

L'utilizzo del metodo di Montecarlo per la valutazione del prezzo di un'opzione consiste nell'individuare i possibili scenari che potrebbero profilarsi riguardo il prezzo dell'attività sottostante sfruttando l'assunzione di operare in un ambiente neutrale al rischio per poi ricavare il valore dell'opzione come media attualizzata dei payoff.

Il valore fondamentale per l'utilizzo di questo metodo è ε che è estratto casualmente (con una distribuzione normale standardizzata) è intuibile quindi che sarà necessario individuare delle tecniche che permettano di generare tali valori casuali con criteri opportuni. Affinché il risultato sia attendibile è necessario che i valori generati per l'evoluzione del sottostante siano *verosimili.*

Il metodo di Montecarlo è usato maggiormente per il calcolo di opzioni di tipo europeo (esercitabili solo a scadenza) e consiste in una serie di passaggi che ora si andranno a spiegare.

Prima di tutto è necessario generare *n* possibili scenari di prezzo del sottostante a scadenza utilizzando la seguente formula:

$$S_T = S * e^{\left[\left(rf - \frac{\sigma^2}{2}\right)(T) + \sigma\varepsilon\sqrt{(T)}\right]}$$

in cui S è il valore iniziale del titolo e S_T è il prezzo del sottostante a scadenza.

In secondo luogo, è necessario calcolare i payoff delle opzioni ottenuti utilizzando gli *n* scenari del sottostante e sottraendoli per lo strike price (K) con la nota formula:

$$C_T = \max[S_T - K, 0]$$

Poi bisogna procedere con l'attualizzazione, utilizzando il tasso privo di rischio (rf), della media campionaria dei payoff così ottenuti:

$$C_{MC} = e^{-rf(T)} E(C_T)$$

Dove la media campionaria è uguale a:

$$E(C_T) = \frac{1}{M} \sum_{i=1}^{M} \max[S_T - K, 0]$$

In altre parole, il metodo di Montecarlo stima il valore di un'opzione calcolando la media campionaria degli M payoff attualizzati (C_{MC}) questo perché grazie al il *teorema del limite centrale*[48] è possibile affermare che, per un numero di osservazioni sufficientemente grande[49] di variabili casuali indipendenti, la media campionaria, $E(C_T)$, se attualizzata ha una distribuzione che tende a quella di una distribuzione normale con media C_t e varianza $\frac{\eta^2}{M}$ in cui η indica la varianza della variabile casuale payoff, è possibile notare facilmente che avendo al denominatore M (il numero dei payoff) è possibile ridurre sempre di più il valore della varianza aumentando il numero degli M payoff.
Il metodo di Montecarlo è quindi un modo accurato di calcolare il prezzo di un'opzione quando si produce un gran numero di scenari, l'errore a cui questo metodo può portare è chiamato *errore standard* ed è dato dalla seguente formula:

$$errore\ standard = \frac{\sigma_c}{\sqrt{M}}$$

in cui M è il numero degli scenari (o campioni) e σ_c è la deviazione standard di C_i, questo valore in statistica misura la stima della deviazione standard dello stimatore.
È possibile seguire lo stesso procedimento utilizzato per il calcolo di una call europea per il calcolo del valore di una put europea con il metodo di Montecarlo variando solamente il payoff dell'opzione e quindi si deve passare dalla differenza tra il prezzo del sottostante e lo strike price alla differenza tra lo strike price e il prezzo del sottostante.

Nel prossimo capitolo verranno creati due programmi, uno che rispetti le regole del modello di Black-Scholes e uno che segua il metodo di Montecarlo per calcolare il prezzo di un'opzione call e di un'opzione put europea con un linguaggio di programmazione chiamato Python.
Una volta spiegati i vari passaggi per arrivare alla creazione dei due codici si procederà alla trasformazione di questi ultimi in una vera *web application*, con questo termine si identificano tutti i programmi distribuiti e eseguibili dal web.

[48] (Monti, 2008)

[49] In statistica si considerata tale quando il numero dei campioni supera le 30-50 unità

Capitolo 3: Python come strumento di calcolo del prezzo delle opzioni utilizzando Black-Scholes e Montecarlo

Python è un linguaggio di programmazione dinamico ad alto livello, è stato rilasciato per la prima volta nel 1991 dal suo creatore Guido Van Rossum, un informatico olandese.

Il nome Python deriva da una commedia britannica trasmessa dalla BBC tra il 1969 e il 1974, questa commedia era intitolata "*Monty Python's Flying Circus*" (il circo volante dei Monty Python), gli autori di questa commedia facevano parte di un gruppo che si chiamava proprio Monty Python.

Oggi lo sviluppo della piattaforma viene gestito da un'organizzazione no-profit chiamata *Python Software Foundation*.

Python è un linguaggio di programmazione semplice da imparare, da leggere e da scrivere ma è anche uno dei linguaggi più ricchi e comodi da utilizzare. Proprio per tutti questi motivi, secondo alcuni sondaggi, l'84% di coloro che programmano utilizzano Python come primo linguaggio di programmazione[50].

Gli usi che Python può avere sono molteplici ma più del 50% di chi utilizza Python lo utilizza per analizzare dati, che in finanza può tradursi in analisi sia quantitative che qualitative come per esempio analisi dei mercati finanziari e predizioni future sui mercati.

Python comunque rimane un linguaggio open-source e scaricabile gratuitamente dal sito ufficiale[51].

Python permette di poter istallare una vasta gamma di librerie, le librerie sono un insieme di funzioni o strutture dati predefinite e predisposte per essere collegate ad un programma software, in questo caso Python attraverso un opportuno collegamento.[52]

Le più importanti librerie per quanto riguarda Python applicato alla finanza sono tre:

- Numpy: la libreria Numpy è una delle librerie fondamentali per lo sviluppo di Python è una libreria numerica che permette la gestione di operazioni di algebra lineare e funzioni di numeri casuali.
- Pandas: la libreria Pandas è una libreria molto potente ed è uno dei più importanti strumenti per l'analisi di dati, uno degli attributi più importanti di questa libreria è il fatto che riesca a lavorare con dati provenienti da moltissime diverse base dati come HTML, JSON e Excel.

---

[50] https://www.jetbrains.com/research/python-developers-survey-2018/

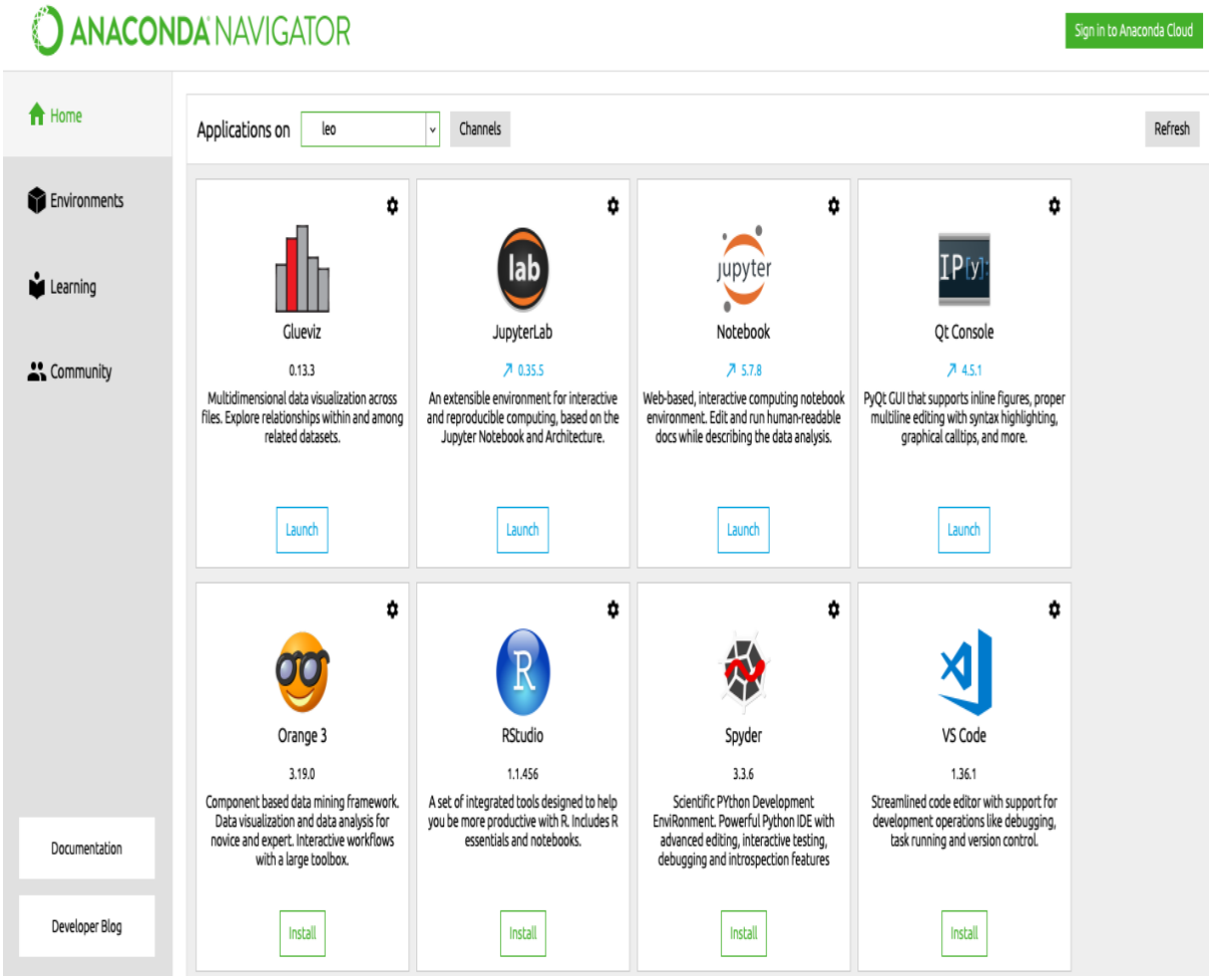
[51] https://www.python.org/

[52] https://it.wikipedia.org/wiki/Libreria_%28software%29

- Scipy: è una collezione di software open-source che permette di utilizzare funzioni matematiche in Python,

Per la creazione dei programmi su Python in questa sede si utilizzerà Anaconda ossia un pacchetto open-source che contiene al suo interno una serie di piattaforme sulle quali è possibile utilizzare una grande varietà di versioni sia di Python che del linguaggio di programmazione R.

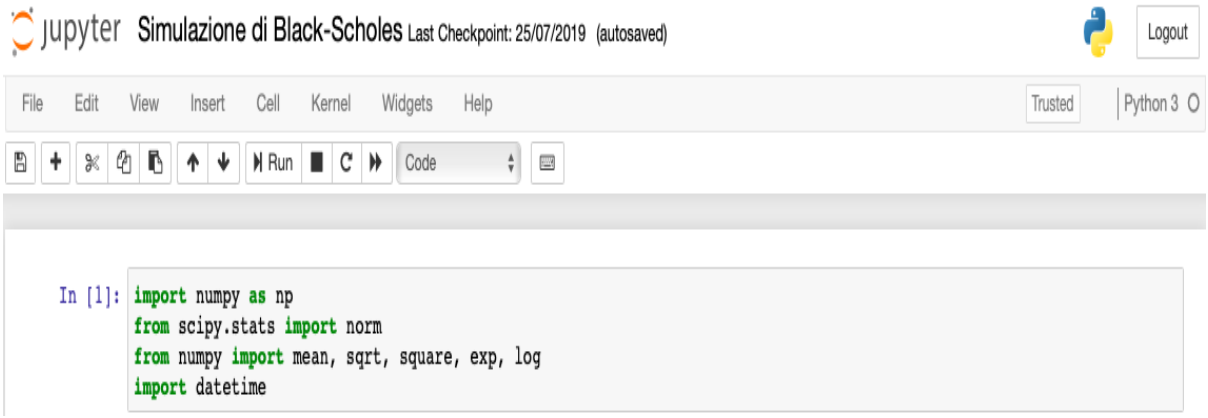
Questa è la schermata iniziale una volta avviato il programma Anaconda:



Come ambiente di programmazione si utilizzerà Jupyter Notebook ossia un'ambiente virtuale in cui è possibile programmare con Python, Anaconda permette di poter scegliere la piattaforma adeguata a seconda delle esigenze del programmatore.

3.1 Simulare Black-Scholes con Python

La prima cosa da fare è importare le librerie necessarie per la creazione dell'algoritmo.



Come si può vedere la funzione *import* permette di aggiungere alla piattaforma virtuale la libreria Numpy e, grazie a questa libreria, di poter inserire nel programma anche alcune funzioni matematiche di base come la media (mean), la radice quadrata (sqrt), il quadrato di un numero (square), il numero di Nepero (exp) e la funzione logaritmo naturale (log), dalla libreria Scipy si può importare la funzione "norm" che permette di poter ottenere la distribuzione normale cumulativa di un determinato valore.
Infine, è possibile importare il modulo "datetime" che permette di calcolare il numero effettivo di giorni passati tra due una data e un'altra.
Poi si procede dando dei valori alle variabili fondamentali del modello ossia al prezzo del sottostante (S), allo strike price (K), al tempo (T)[53], al tasso privo di rischio (rf) e alla volatilità (sigma), nell'esempio si sono riportati i seguenti valori:

```
In [2]: S=100
        E=90
        T=(datetime.date(2015,9,21)-datetime.date(2014,9,21)).days/365.0
        rf=0.05
        sigma=0.2
```

Dopo di che si deve procedere definendo la formula di Black-Scholes che, per un'opzione call, si ricorda essere la seguente:

$$C = SN(d_1) - Ke^{-rfT}N(d_2)$$

[53] Nell'esempio si è deciso di assegnare come intervallo di tempo l'anno che intercorre tra il 21 settembre 2014 e il 21 settembre 2015.

con

$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(rf + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \quad e \quad d_2 = d_1 - \sigma\sqrt{T}$$

Il codice Python può essere così scritto per definire un'opzione call:

```
In [3]: def blackscholes_call(S,E,T,rf,sigma):
            d1=(log(S/E)+(rf+sigma*sigma/2.0)*T)/(sigma*sqrt(T))
            d2=d1-sigma*sqrt(T)
            return S*norm.cdf(d1)-E*exp(-rf*T)*norm.cdf(d2)
```

La parola *def* all'inizio del programma, in Python, permette di "insegnare" al programma una determinata funzione, con questo termine si intende un blocco organizzato di codici che viene utilizzato per eseguire un mini-programma, in questo caso ogni volta che la funzione *balckscholes_call* verrà lanciata specificando le variabili tra parentesi verrà dato come risultato il valore dell'opzione call.

Dopo aver definito il valore di d_1 e d_2 è necessario scrivere la parola *return* prima di scrivere la formula di Black-Scholes, questo perché così facendo, una volta lanciato il programma verrà riportato solo il risultato della formula senza riportare anche i valori di d_1 e d_2.

Nella formula finale è stato inserita la parola *norm.cdf* che permette di ottenere la distribuzione normale cumulativa dei valori tra parentesi proprio come nella formula sopra citata.

È possibile poi utilizzare lo stesso procedimento per il calcolo di un'opzione put europea ricordando che è necessario cambiare la formula di Black-Scholes nella seguente:

$$P = Ke^{-rfT}N(-d_2) - SN(-d_1)$$

I valori di d_1 e d_2 sono uguali a quelli della formula per l'opzione call e quindi il programma sarà il seguente:

```
In [4]: def blackscholes_put(S,E,T,rf,sigma):
            d1=(log(S/E)+(rf+sigma*sigma/2.0)*T)/(sigma*sqrt(T))
            d2=d1-sigma*sqrt(T)
            return E*exp(-rf*T)*norm.cdf(-d2)-S*norm.cdf(-d1)
```

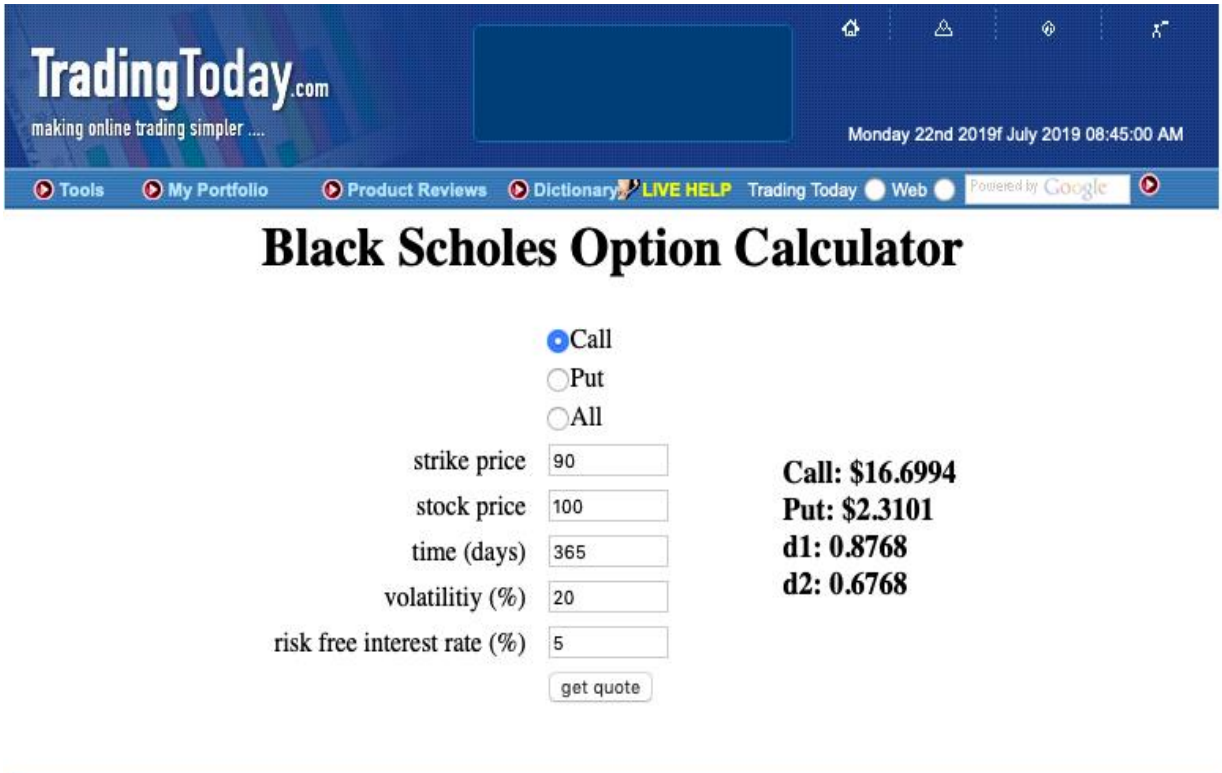
A questo punto per poter lanciare il programma è necessario comporre l'ultima riga del programma scrivendo la parola *print* seguita dalla descrizione del risultato che si otterrà dai due programmi, così facendo si otterrà la parte finale del programma:

```
In [5]: print ("il prezzo di un'opzione call con la formula di Black-Scholes", blackscholes_call(S,E,T,rf,sigma))
        print ("il prezzo di un'opzione put con la formula di Black-Scholes:", blackscholes_put(S,E,T,rf,sigma))

il prezzo di un'opzione call con la formula di Black-Scholes 16.699448408416004
il prezzo di un'opzione put con la formula di Black-Scholes: 2.3100966134802654
```

Ottenendo i seguenti risultati:

- Per l'opzione call il valore 16,69945 (approssimato)
- Per l'opzione put il valore 2,310097 (approssimato)

A questo punto è possibile cercare un riscontro sulla piattaforma di *TradingToday* come fatto in precedenza per osservare se i valori coincidono con quelli ottenuti in Python:



Come si può notare i valori ottenuti coincidono, di seguito si propone l'intero programma costruito in Python.

```
import numpy as np
from scipy.stats import norm
from numpy import mean, sqrt, square, exp, log
import datetime

S=100
E=90
T=(datetime.date(2015,9,21)-datetime.date(2014,9,21)).days/365.0
rf=0.05
sigma=0.2

def blackscholes_call(S,E,T,rf,sigma):
    d1=(log(S/E)+(rf+sigma*sigma/2.0)*T)/(sigma*sqrt(T))
    d2=d1-sigma*sqrt(T)
    return S*norm.cdf(d1)-E*exp(-rf*T)*norm.cdf(d2)

def blackscholes_put(S,E,T,rf,sigma):
    d1=(log(S/E)+(rf+sigma*sigma/2.0)*T)/(sigma*sqrt(T))
    d2=d1-sigma*sqrt(T)
    return E*exp(-rf*T)*norm.cdf(-d2)-S*norm.cdf(-d1)

print ("il prezzo di un'opzione call con la formula di Black-Scholes", blackscholes_call(S,E,T,rf,sigma))
print ("il prezzo di un'opzione put con la formula di Black-Scholes:", blackscholes_put(S,E,T,rf,sigma))
```

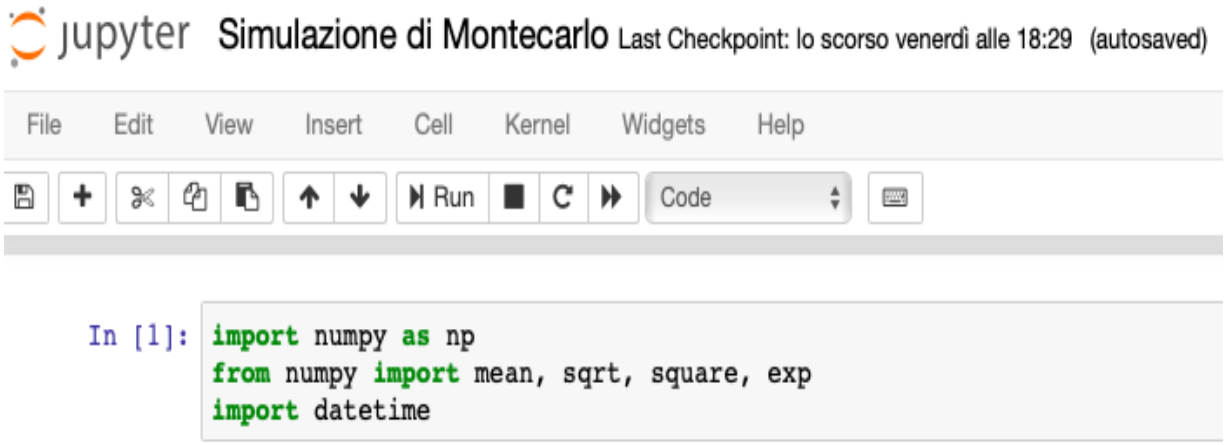
```
il prezzo di un'opzione call con la formula di Black-Scholes 16.699448408416004
il prezzo di un'opzione put con la formula di Black-Scholes: 2.3100966134802654
```

3.2 Simulazione con il metodo di Montecarlo

Il modello di Black-Scholes permette di poter raggiungere un valore esatto dell'opzione sia nel caso in cui si tratti di un'opzione put sia nel caso di opzione call. Il metodo di Montecarlo, invece, ci permette di ottenere il valore centrale di una distribuzione producendo dei percorsi "casuali" che potrebbe assumere il sottostante.

Prima di tutto è necessario importare le librerie richieste, come nel caso del modello di Black-Scholes, per l'implementazione del programma.

Le librerie richieste sono le seguenti:



Dopo di che è necessario inserire alcuni valori per identificare le variabili fondamentali del programma

```
S=100
E=90
T=(datetime.date(2015,9,21)-datetime.date(2014,9,21)).days/365.0
rf=0.05
sigma=0.2
iterazioni=100000000
```

Dopo di che si deve definire la funzione che si vuole andare a creare, in questo caso la funzione sarà chiamata "*call_option_simulation*" per indicare che si andrà ad analizzare il caso di una opzione call europea, in seguito verrà mostrato il procedimento anche per una opzione put.
Il metodo di Montecarlo, come detto precedentemente, ha bisogno di un numero molto elevato di simulazioni del sottostante. Il sottostante si calcola con la formula ricavata già in precedenza:

$$S_T = S * e^{\left[\left(rf-\frac{\sigma^2}{2}\right)(T)+\sigma\varepsilon\sqrt{(T)}\right]}$$

La prima cosa da fare per calcolare il prezzo del sottostante a scadenza T è di definire il valore di ε utilizzando il seguente codice:

$$Z = np.random.standard\_normal(iterazioni)$$

Questo codice ci permette di produrre una quantità di numeri casuali con distribuzione normale uguale al numero di iterazioni che si decide di impostare, qui sotto si produrrà un esempio con un numero di iterazioni uguale a 100:

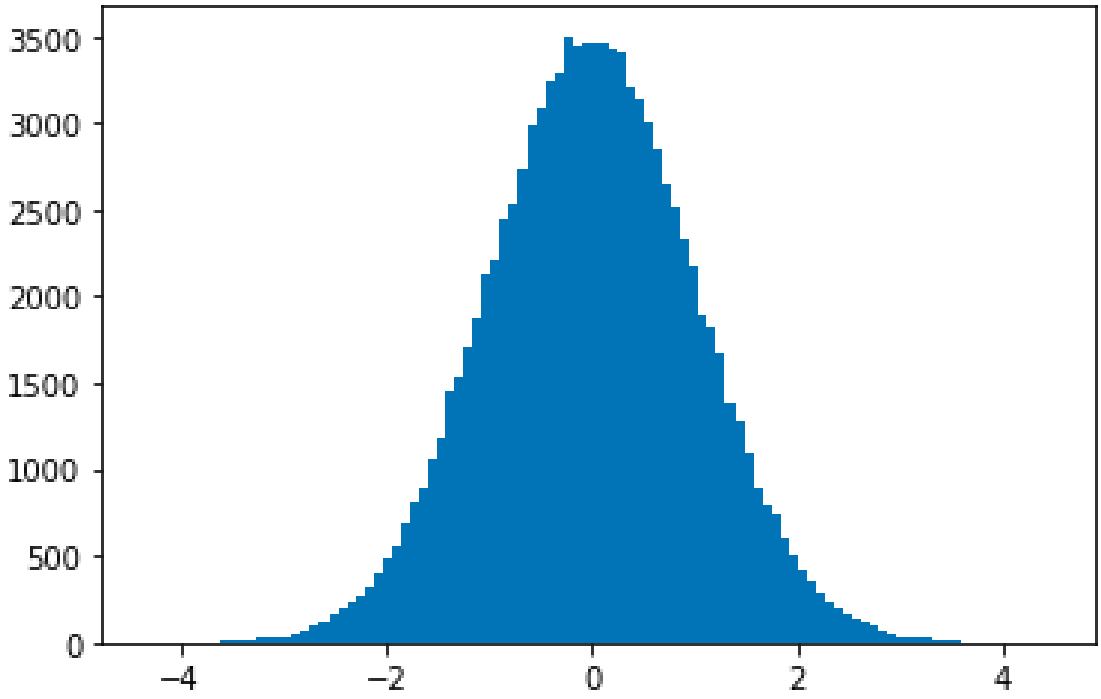
```
iterazioni=100
Z=np.random.standard_normal(iterazioni)

print(Z)
```

```
[ 1.17306614  0.44688316  1.1860826   0.43504133 -0.9165392  -1.14073291
  0.80557536  0.81620563  0.70873097  0.29406729 -0.14918961 -0.56075092
  2.00429329 -1.95244216  0.23454971  0.95779937 -0.91723408  0.55917432
  0.92272528 -0.07815386  0.4294416   0.28425146  0.50963807  0.35414713
  0.64287321  2.25350455  0.62014495  0.58686274 -0.26265103  0.4875567
 -0.96241719  0.5490869   0.08589978  0.38073619  0.3703561   1.55758901
 -0.2120609   0.87742391 -1.17268515  2.00292806 -0.2700313  -0.1768041
  0.20000598 -1.52774676  0.60635338  0.61937763  0.17976111 -0.45350499
 -1.1509341   0.56119449 -0.63557645 -1.29797885 -1.53031521 -1.39855451
 -0.16813727  1.72249068  0.10505624  1.03125362  0.0795908  -1.66310845
 -0.14149758  0.3438523  -0.0874093   1.55014781  1.65312574 -0.3648019
 -1.27133885  0.33143593  0.81864435 -0.83625496  2.43872263 -0.73034119
 -0.32864872 -0.97734658 -1.85511918  1.39534012 -0.21608026  0.70235879
 -0.16112668 -1.85900531 -0.22157533 -1.02868625  1.16042725  0.07504518
  1.79418698 -1.51952189 -0.48107387 -0.20682804 -1.40769662  0.36934294
  0.88647414 -0.66312499 -1.00075489 -0.44589485 -0.36214256  0.9666032
 -0.041662   -0.32225722  0.62117861 -0.47851902]
```

Si può dimostrare che i numeri generati siano disposti con una distribuzione normale con media zero e varianza uno producendo il grafico[54], ovviamente per produrre un grafico accettabile è necessaria una quantità di iterazioni ben più alta di 100, qui di seguito si esporrà il codice e il grafico con un numero di iterazioni pari a 100000.

[54] È necessario introdurre una nuova libreria per poter disegnare un grafico su Python, questa libreria si chiama *matplotlib.pyplot.*

```
iterazioni=100000
Z=np.random.standard_normal(iterazioni)
plt.hist(Z, bins=100)
plt.show()
print(stock_price)
```



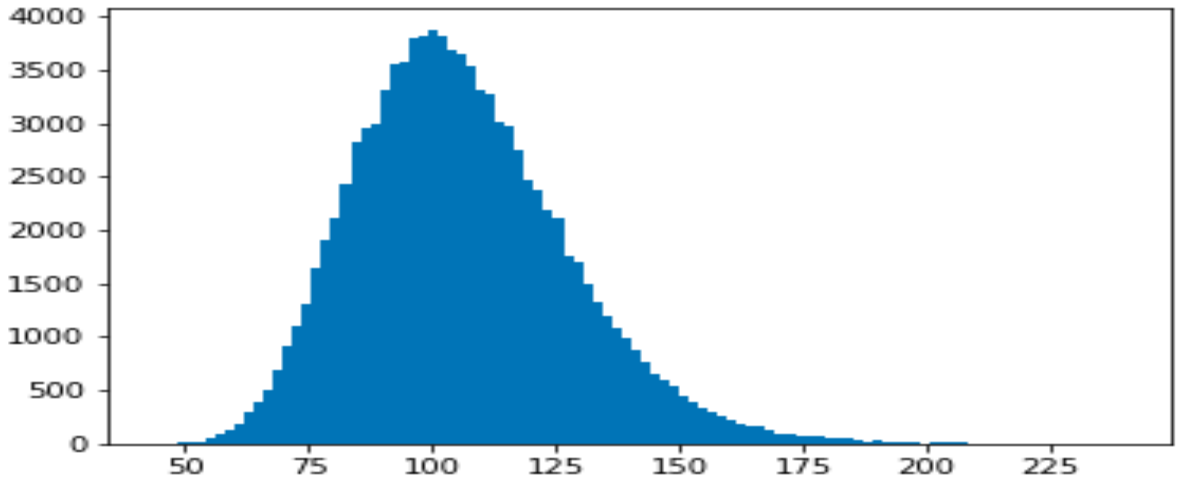
A questo punto si può scrivere il codice per creare i possibili scenari che può assumere il sottostante con il seguente codice:

```
S=100
E=90
T=(datetime.date(2015,9,21)-datetime.date(2014,9,21)).days/365.0
rf=0.05
sigma=0.2
iterazioni=100


Z=np.random.standard_normal(iterazioni)
stock_price=S*np.exp(T*(rf-0.5*sigma**2)+sigma*sqrt(T)*Z)
print(stock_price)
```

Stampando il codice si ottiene una lista di valori casuali che rispondono alla formula dello *stock price* sopra esposta, questo è un esempio di valori con un numero di iterazioni pari a 100:

```
[ 88.01552667  74.87955314 130.42593321  90.02089248  96.8938655
 132.31311628 128.21936673 118.77733185  77.03268303  91.24757894
 104.59355943  97.45766833  83.34117522 136.85516526 129.4119065
 120.03246003  74.4038346  104.54102928 151.66270903  99.86303827
 127.45563631 107.97516417  91.98044345  97.9862738   85.23423491
 115.90022471  95.99423207  95.69873147  84.62473646 105.60245992
 145.90677225  85.472826    92.07823938 112.28543475  89.54628815
  83.69099809  92.9517283  110.94530599 167.04209422 115.24391004
 120.37667176  88.99126218 101.5303397  123.41725145  94.76463098
 104.96175707 170.60783294 100.92429598  80.4347658  109.32858197
 118.23911802  79.68108009  77.86630515 144.30421829  89.11258333
 118.59130345 102.37451893  88.83598838  91.29333408  82.10421615
 121.00830605  79.2009385   76.1476487   86.37544709 107.88245179
  74.08337265 173.71249417 115.63906604 124.87682837 113.33259739
 104.78439498 128.31570149 146.10716496  74.93897686 118.91284399
 175.07309003 108.75735086 107.55809624 101.39578506 102.65631804
 160.42100642 149.19282921  91.82858868  77.19004754  90.53316335
  73.87999492 110.10121364 129.59206535  86.03821184 123.28466169
 109.43549342 120.72626317  78.07646661  78.4536004  108.56536323
  96.4712445  108.07265992 102.77167485  95.53869838 143.94673984]
```

È possibile anche rappresentare i valori dei prezzi del sottostante con un grafico simile a quello utilizzato per la variabile Z (anche in questo caso il numero di iterazioni equivale a 100000), in questo caso si può notare come la distribuzione dei prezzi assomigli molto ad una distribuzione log-normale, ossia la stessa distribuzione che assumono i prezzi del sottostante nella formula di Black-Scholes.



Il passaggio successivo è calcolare i payoff per il calcolo dell'opzione quindi è necessario sottrarre dai vari prezzi assunti dal sottostante lo strike price:

$$C_T = \max[S_T - K, 0]$$

Per fare questo bisogna creare delle matrici con primo valore uguale alla differenza tra prezzo del sottostante e strike price e secondo valore uguale a zero, è possibile creare una matrice di soli zeri con lunghezza pari al numero delle iterazioni eseguite utilizzando il seguente codice:

```
iterazioni=100
Z=np.random.standard_normal(iterazioni)
option_data=np.zeros_like(Z)
print(option_data)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.]
```

Ora si deve scrivere un codice che dia come risultato il valore massimo della differenza sopra citata e i valori uguali a zero.

```
iterazioni=100
Z=np.random.standard_normal(iterazioni)
stock_price=S*np.exp(T*(rf-0.5*sigma**2)+sigma*sqrt(T)*Z)
option_data=np.zeros_like(Z)
p=np.maximum(stock_price-E,option_data)
print(p)
```

```
[41.29475825  1.84456144 13.92119316  5.67436151 14.53016458  6.305
92035
 12.4814425  30.43656309 15.20916503 14.2441163  49.51999375 39.670
48912
 50.50566069  8.50329523  0.         31.03052829 42.77649697  0.
  0.         30.68765105  0.         32.78479769 40.12776296  1.901
29235
 73.85464169 34.98193006 46.74140464 37.86744878 12.92315791 60.183
16181
  0.          0.         16.71287764 39.51628173 34.75938478  0.
  0.         39.03629071 25.22637103 22.80779777 30.38210478 51.026
46093
 34.73205496  0.         18.47455073 15.98584563 16.93405046  0.
  8.38322387  0.          0.          0.         29.36654238  2.775
25671
  0.14677088 37.38040317 10.69605391  4.00380131 52.19503549 26.662
90887
  0.         10.69703003 10.97398527 29.86670332 17.51095884  0.
  0.         26.88052846 12.1895376  27.29207185 13.51346146  6.296
38871
 28.98718223  0.86253632 10.74296061  0.          0.          5.328
51386
 41.81459218 23.60475968  0.          0.         46.73994987  8.977
32156
  0.         85.33571857  0.         16.04850007 15.1326187  29.397
34678
 12.55161363  0.         14.23387699  8.29857169 19.91610898  0.
  9.60152729  0.         46.82361487 12.35641974]
```

Questo è il risultato che si ottiene con 100 iterazioni.

A questo punto si deve introdurre un nuovo codice per creare la media campionaria della sommatoria dei valori del titolo sottostante, la funzione sommatoria può essere scritta nel seguente modo:

```
media_campionaria=np.sum(p)/iterazioni
```

Ora si hanno tutti i codici per poter calcolare il prezzo dell'opzione call con il metodo di Montecarlo, questo è il codice completo:

```
def call_option_simulations(S,E,T,rf,sigma,iterazioni):
    Z=np.random.standard_normal(iterazioni)
    option_data=np.zeros_like(Z)
    stock_price=S*np.exp(T*(rf-0.5*(sigma**2))+sigma*sqrt(T)*Z)
    p=np.maximum(stock_price-E,option_data)
    media_campionaria=np.sum(p)/iterazioni
    C=np.exp(-rf*T)*media_campionaria
    return C
```

L'opzione put si può scrivere sotto forma di codici in modo molto simile, l'unica cosa che bisogna ricordarsi di cambiare è il valore del payoff che per l'opzione put è il seguente:

$$C_T = \max[K - S_T, 0]$$

Quindi il codice finale di una opzione put sarà il seguente:

```
def put_option_simulations(S,E,T,rf,sigma,iterazioni):
    Z=np.random.standard_normal(iterazioni)
    stock_price=S*np.exp(T*(rf-0.5*(sigma**2))+sigma*sqrt(T)*Z)
    option_data=np.zeros_like(Z)
    p=np.maximum(E-stock_price,option_data)
    media_campionaria=np.sum(p)/iterazioni
    P=np.exp(-rf*T)*media_campionaria
    return P
```

Prima di far partire il programma è possibile aggiungere una riga di codice per vedere quale sia l'errore standard del programma appena creato, si ricorda che la formula dell'errore standard è la seguente:

$$errore\ standard = \frac{\sigma_c}{\sqrt{M}}$$

Ora si farà partire il programma completo con un numero di iterazioni pari a 100:

```
import numpy as np
from numpy import mean, sqrt, square, exp
import datetime

S=100
E=90
T=(datetime.date(2015,9,21)-datetime.date(2014,9,21)).days/365.0
rf=0.05
sigma=0.2
iterazioni=100

def call_option_simulations(S,E,T,rf,sigma,iterazioni):
    Z=np.random.standard_normal(iterazioni)
    stock_price=S*np.exp(T*(rf-0.5*(sigma**2))+sigma*sqrt(T)*Z)
    option_data=np.zeros_like(Z)
    p=np.maximum(stock_price-E,option_data)
    media_campionaria=np.sum(p)/iterazioni
    C=np.exp(-rf*T)*media_campionaria
    return C

def put_option_simulations(S,E,T,rf,sigma,iterazioni):
    Z=np.random.standard_normal(iterazioni)
    stock_price=S*np.exp(T*(rf-0.5*(sigma**2))+sigma*sqrt(T)*Z)
    option_data=np.zeros_like(Z)
    p=np.maximum(E-stock_price,option_data)
    media_campionaria=np.sum(p)/iterazioni
    P=np.exp(-rf*T)*media_campionaria
    return P

standard_error=sigma/sqrt(iterazioni)


print("il prezzo di un'opzione europea call è:", call_option_simulations(S,E,T,rf,sigma,iterazioni))
print("il prezzo di un'opzione put europea è:", put_option_simulations(S,E,T,rf,sigma,iterazioni))
print("l'errore del programma è:", standard_error)
```

```
il prezzo di un'opzione europea call è: 15.239501190113788
il prezzo di un'opzione put europea è: 2.1464633259382238
l'errore del programma è: 0.02
```

Con un valore di iterazioni pari a 100 si può notare come l'errore si attesti su un valore pari a: 0,02 un valore abbastanza elevato, questo lo si può notare anche sui valori assunti dai prezzi sia dell'opzione call che dell'opzione put che discostano abbastanza dal valore finale che si era trovato utilizzando la formula di Black-Scholes.

Si prova adesso ad aumentare il numero delle iterazioni a 100000000 per vedere come varia il risultato all'aumentare del numero delle iterazioni.

Il risultato ottenuto è il seguente:

```
import numpy as np
from numpy import mean, sqrt, square, exp
import datetime

S=100
E=90
T=(datetime.date(2015,9,21)-datetime.date(2014,9,21)).days/365.0
rf=0.05
sigma=0.2
iterazioni=100000000

def call_option_simulations(S,E,T,rf,sigma,iterazioni):
    Z=np.random.standard_normal(iterazioni)
    stock_price=S*np.exp(T*(rf-0.5*(sigma**2))+sigma*sqrt(T)*Z)
    option_data=np.zeros_like(Z)
    p=np.maximum(stock_price-E,option_data)
    media_campionaria=np.sum(p)/iterazioni
    C=np.exp(-rf*T)*media_campionaria
    return C

def put_option_simulations(S,E,T,rf,sigma,iterazioni):
    Z=np.random.standard_normal(iterazioni)
    stock_price=S*np.exp(T*(rf-0.5*(sigma**2))+sigma*sqrt(T)*Z)
    option_data=np.zeros_like(Z)
    p=np.maximum(E-stock_price,option_data)
    media_campionaria=np.sum(p)/iterazioni
    P=np.exp(-rf*T)*media_campionaria
    return P

standard_error=sigma/sqrt(iterazioni)


print("il prezzo di un'opzione europea call è:", call_option_simulations(S,E,T,rf,sigma,iterazioni))
print("il prezzo di un'opzione put europea è:", put_option_simulations(S,E,T,rf,sigma,iterazioni))
print("l'errore del programma è:", standard_error)
```

```
il prezzo di un'opzione europea call è: 16.70037639099834
il prezzo di un'opzione put europea è: 2.3094636587166386
l'errore del programma è: 2e-05
```

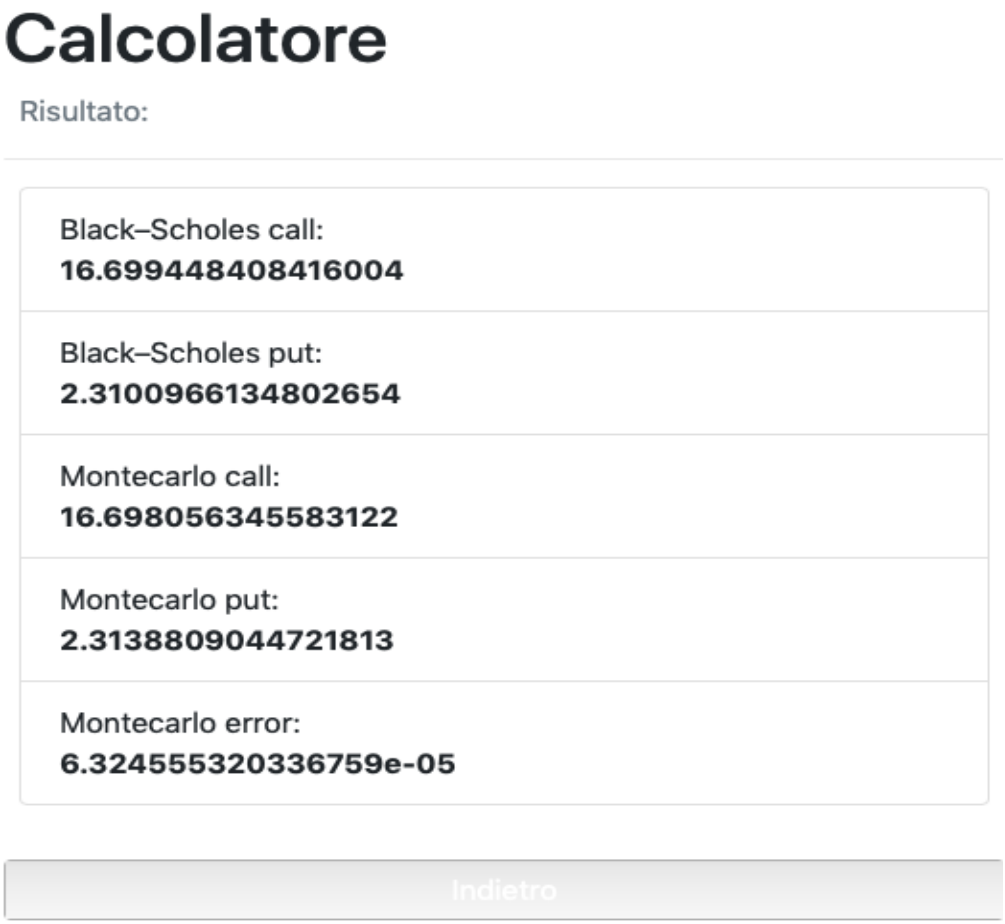
Come si può osservare l'errore è diminuito notevolmente e il valore delle opzioni così ottenuto si avvicina molto al valore reale delle opzioni calcolato mediante la formula di Black-Scholes, il numero delle iterazioni si può aumentare sempre di più ottenendo un risultato sempre più preciso.

È necessario tenere a mente che più si aumenta il numero delle iterazioni più tempo impiegherà il programma a calcolare il valore dell'opzione, un valore troppo alto di iterazioni potrebbe addirittura portare al blocco del programma.

3.3 Web Application dei codici Python

Una delle librerie utilizzate per la creazione di pagine e programmi web scritti in Python è chiamata *Flask* che, nello specifico, viene individuato come un web framework ossia una libreria di codici che possa consentire uno sviluppo facile e veloce di applicazioni web.
Una volta che il codice Python è stato adattato per la poter essere eseguito da Flask bisogna farlo partire dal terminale del computer, dopo di che comparirà un indirizzo http a cui è possibile accedere e trovare il calcolatore.
Questo è la schermata che si otterrà non appena si apre la pagina http:

Una volta inseriti tutti i dati richiesti il programma calcolerà i risultati e farà vedere in un'altra pagina i risultati ottenuti sia con il metodo di Montecarlo sia con il metodo di Black-Scholes, questi sono i risultati ottenuti utilizzando i dati usati anche in precedenza nei due esercizi sopra citati:



È utile ricordare che il programma appena esposto è eseguibile solamente avendo i codici di esecuzione del programma in una cartella situata sul proprio computer e solamente eseguendo i suddetti codici da terminale. Questo perché il terminale genera un cosiddetto *localhost* del tipo (127.0.0.1) ossia un indirizzo IP che possa essere usato dalle applicazioni per comunicare con il calcolatore su cui sono in esecuzione, l'indirizzo IP di localhost permette di eseguire un servizio network senza aver bisogno di una fisica interfaccia network.

In altre parole, l'indirizzo che comincia con 127.0.0.1 è l'indirizzo che identifica la rete interna del computer stesso.

Il sistema di localhost permette quindi di poter testare applicazioni web senza aver bisogno di possedere un proprio server o dover pagare per ottenerne uno.

Conclusioni

Nei tre capitoli precedenti si sono esposti alcuni metodi di valutazione del prezzo delle opzioni, in particolare sono stati analizzati il modello binomiale, il modello di Black-Scholes e il metodo di Montecarlo, quest'ultimo è, come si è visto, molto utile perché permette di superare alcuni limiti dei modelli precedentemente trattati e riesce ad approssimare in modo fedele il reale prezzo di un'opzione nel caso in cui si facciano delle simulazioni con un numero molto elevato di iterazioni.

Prima dell'introduzione dei calcolatori e dei linguaggi di programmazione, il valore del prezzo di un'opzione risultava molto più complesso da ottenere con l'utilizzo del metodo di Montecarlo proprio perché era molto più difficile riuscire a produrre un numero abbastanza alto di iterazioni, questo portava ad un risultato che discostava parecchio dal reale valore dell'opzione, proprio per questo motivo non poteva essere considerato un metodo efficace per calcolare il prezzo di un'opzione.

Una volta introdotti i vari calcolatori e linguaggi di programmazione, l'utilizzo di metodi che richiedevano l'uso di un alto numero di iterazioni e di lunghe sequenze di numeri casuali è cresciuto notevolmente permettendo di ottenere risultati sempre più accurati.

È importante sottolineare, inoltre, come ogni qualvolta si utilizzino linguaggi di programmazione o calcolatori per l'implementazione del metodo di Montecarlo o altri modelli di natura statistica che richiedono sequenze casuali di numeri vengano utilizzate serie di numeri pseudo-casuali e non serie di numeri casuali veri e propri proprio per la natura deterministica degli strumenti utilizzati.

Questo, però, comporta che anche i valori scelti che, in linea teorica, dovrebbero essere completamente aleatori nella pratica non lo siano del tutto, anche se è comunque corretto specificare che gli attuali generatori di numeri pseudo-casuali hanno raggiunto un livello tale di casualità da poter quasi essere classificati come veri e propri generatori di numeri casuali.

In sintesi, si può affermare che il metodo di Montecarlo sia, al giorno d'oggi, un buon metodo di calcolo del prezzo delle opzioni se implementato con un linguaggio di programmazione adeguato e abbastanza potente da poter supportare la creazione di un numero di iterazioni abbastanza grande da diminuire il più possibile l'errore con l'aumento delle iterazioni come osservato nell'esempio sopra citato che riporta un errore veramente basso e quasi trascurabile.

Bibliografia

Aluru, S. (1997). Lagged Fibonacci Random Generators for Distributed Memory Parallel Computers. *Journal of Parallel and Distributed Computing vol.45*, p. 1-12.

Codogno, M. (2018, Giugno 26). *Il Post*. Tratto da Numeri pseudocasuali e il ritorno dei TRNG: https://www.ilpost.it/mauriziocodogno/2018/06/26/numeri-pseudocasuali-e-il-ritorno-dei-trng/

Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of Financial Economics, vol. 7*, p. 229-263.

Fabrizi, P. L. (2016). *Economia del Mercato Mobiliare.* Milano: Egea.

Galton, F. (1890, maggio 1). Dice for Statistical Experiments. *Nature*, p. 13-14.

Häcker, J., & Ernst, D. (2017). *Financial Modeling: An Introductory Guide to Excel and VBA Applications in Finance.* United Kingdom: Palgrave macmillan.

Hellekalek, P. (1998). Good random number generators are (not so) easy to find. *Mathematics and Computers in Simulation*, p. 485-505.

Hull, J. C. (2015). *Options, Futures, and Other Derivatives (Ninth ed.) .* Boston: Pearson.

James, F. (1990). A review of pseudorandom number generators. *Computer Physics Communications*, p. 329-344.

Kneusel, T. R. (2018). *Random numbers and computers.* Thornton: Springer International Publishing .

Knuth, D. E. (1998). *The Art of Computer Programming Volume 2.* Addison-Wesley.

L'Ecuyer, P. (1997). Uniform Random Number Generators: A Review. *Winter Simulation Conference Proceedings* (p. 127-134). IEEE.

Migliorini, F. (2019, Marzo 1). *Il "Dilemma di Monty Hall": il cambio di variabile*. Tratto da Fiscomania.com: https://fiscomania.com/dilemma-di-monty-hall/

Monti, A. C. (2008). *Introduzione alla Statistica.* Edizioni scientifiche italiane.

Narciso, A. (2014, Maggio 24). *Opzioni e Futures*. Tratto da Statistica Apllicata: https://narcisoabigail.wordpress.com/2014/05/24/dividendi-opzioni-futures/

Neumann, J. v. (1951). *Various Techniques Used in Connection With Random Digits.*

Nishimura, M. M. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS) vol. 8*, p. 3-30.

Pareschi, L. (2009, Agosto 1). *I numeri (scientificamente) casuali*. Tratto da Maddmaths!: http://maddmaths.simai.eu/divulgazione/i-numeri-scientificamente-casuali/

Paul Armer, E. B. (1955). *A Million Random Digits with 100.000 Normal Deviates.* The Free Press.

Pigeon, S. (s.d.). *The Middle Square Method (Generating Random Sequences VIII)* . Tratto da Harder, Better, Faster, Stronger: https://hbfs.wordpress.com/2017/11/21/the-middle-square-method-generating-random-sequences-viii/

Scholes, M., & Black, F. (1973). The Pricing of Options and Corporate Liabilities,. *Journal of Political Economy, vol.81*, p. 637-654.

Stoll, H. R. (1969, Dicembre). The Relationship Between Put and Call Option Prices. *The Journal of Finance, Vol. 24, No. 5*, p. 801-824.