Opzioni Barriera

Marco Evangelista
Donatella Straccamore

1 Opzioni Barriera

Un'opzione è un contratto tra due parti, A e B, che dà il diritto (ma non l'obbligo) a chi lo detiene (supponiamo A) di comprare/vendere (da/a) B un bene ad una data futura (T) e ad un prezzo (K), detto strike, prefissato al momento della stipula del contratto.
Tra le varie opzioni che si possono trovare nel mercato, il programma che è stato implementato si occupa di prezzare un' *Opzione Barriera*.

Definizione: Un' *opzione barriera* è un opzione path dependent (che dipendono dall'intero percorso) ovvero il loro prezzo cambierà a seconda che abbia raggiunto o no un certo livello, definito appunto dalla barriera.
Ci sono vari tipi di opzioni barriera, a seconda che siano up o down e in o out.
I termini *up e down* indicano il livello di barriera, cioè se la barriera sia del tipo superiore o inferiore, *In e Out* si riferiscono invece al fatto che l'opzione si attivi o meno a seconda che si sia toccato o meno il livello sopra indicato. Ovviamente si possono mettere ancora più restrizioni inserendo contemporaneamente entrambe le barriere, superiore e inferiore. Il seguente programma si occuperà di prezzare un'opzione barriera up-and-in, il che significa che è stata inserita una barriera superiore U e che l'opzione si attiva se e solo se la barriera viene toccata entro il tempo finale detto maturità.

2 Il Metodo Monte Carlo

Per calcolare il prezzo di un'opzione call (put) europea esiste una formula deterministica che, però, risulta difficile (e lunga) da implementare quando i beni sottostanti l'opzione sono molti. Ricorrere ad un metodo d'approssimazione, più che ad un calcolo esatto, può risultare utile sia per un'ottimizzazione dei tempi di esecuzione del programma sia per generalizzare opzioni più complicate: la formula deterministica è valida solo per le opzioni call e put europee. Si utilizzerà il metodo MonteCarlo che da, invece, la possibilità di adattare più facilmente il programma ai propri scopi. L'unico inconveniente è che utilizzando un'approssimazione si introduce un errore numerico che può essere trascurato solo aumentando il numero di simulazioni effettuate.

Supponiamo di dover calcolare numericamente una quantità m e poniamo $m = E(X)$ dove X è una v.a. di varianza σ^2. Il metodo MonteCarlo consiste allora nel generare M copie i.i.d. della variabile X e nel considerare la quantità

$$T_M = \frac{1}{M}\sum_{i=1}^{M} X_i$$

come valore di m. Infatti, per il la Legge dei grandi numeri, per $M \to \infty$, $E(X) \to m$. Per calcolare la stima della velocità di convergenza si utilizza il Teorema del Limite Centrale. Infatti

$$\frac{\sum_{i=1}^{M} X_i - Mm}{\sqrt{M\sigma^2}} \to_{M\to\infty} N(0,1).$$

Si ottiene perciò che

$$T_M \simeq \frac{\sigma}{M} Z + m$$

dove $Z \sim N(0,1)$. Ovvero $T_M = m + O(\frac{\sigma}{\sqrt{M}})$, cioè uguale alla quantità m che stiamo cercando più un errore (aleatorio) che dipende da M (cioè dal numero di v.a. simulate) e da σ. E' importante notare che la quantità σ non è marginale: più è alta la varianza più simulazioni bisogna fare per ottenere una buona approssimazione di m.

3 Programma

Per prima cosa bisogna definire tutte le variabili di cui si avrà bisogno nel corso del programma che servirà a simulare il prezzo del sottostante che si sta prendendo in considerazione.
In questo programma è stata assegnata la seguente nomenclatura:
-U=110 è l'altezza della barriera;
-T=1 è il tempo (1 anno);
-K=100 è lo strike, cioè il prezzo che fissiamo oggi e che si il detentore della call riceverà al tempo T;
-R=0.05 è l'interesse annuo fissato;
-sigma=0.2 è la volatilità.

```
/*Calcolo del prezzo di una call con barriera con il metodo Monte
Carlo con intervallo di confidenza al 95%*/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>



main()
{
  int U=110, T=1, K=100, start=100;
  double R=0.05, sigma=0.2;
  double r,a,b,p,ran,z,sn, payoff, media, mediaM, varianza,ICp,ICm;
  double media_quadrati, att, prezzo;
  int N,M,m,n,flag;
```

```
printf("Inserisci il numero M (intero)di simulazioni da effettuare:\n");
scanf("%d",&M);
printf("Inserisci il numero di periodi N di tempo:\n");
scanf("%d",&N);
r=R/double(N); /*definisco "r"*/
a=(1+r)*exp((-sigma)*sqrt(T/double(N)))-1; /*definisco "a"*/
b=(1+r)*exp((sigma)*sqrt(T/double(N)))-1; /*definisco "b"*/
p=(b-r)/(b-a); /*definisco "p"*/
media=0.;
media_quadrati=0.;

for(m=1; m<=M; m++)
   {
       sn=start;
       flag=0;
       for(n=1; n<=N; n++)
       {
   /*procedura per generare una variabile aleatoria "z" uniforme in (0,1)*/
             ran=rand()%32767;
             z=ran/32767;

             if (z<p) sn=sn*(1+a);
             else sn=sn*(1+b);

             if (sn > U) flag=1;
       }
```

La funzione ran ci permette di ottenere un numero casuale tra 0 e il numero massimo che può valutare il calcolatore, e poi con z facciamo in modo che tale numero vari tra 0 e 1. L'utilizzo della bandiera (flag =1) è ciò che fa diventare

la nostra opzione un opzione barriera, in quanto ci permette di vedere se il prezzo del nostro sottostante è raggiunto o meno la barriera prefissata.

```
if (flag==1 && sn>K) payoff=sn-K;
    else payoff=0;

    media = media + payoff;
    media_quadrati = media_quadrati + payoff*payoff;
}
```

Siccome la formula del pay-off nel nostro caso è $h = (S_n - K)_+$ noi lo dovremo quindi prendere in considerazione soltanto se vengono rispettate contemporaneamente entrambi le seguenti condizioni: una data dalla formula stessa , ossia $S_n > k$; e l'altra se la barriera viene toccata o no.

```
mediaM = ( media / double(M) );
att = exp( -N * log(1+r) );
prezzo = att * mediaM;
printf("Il prezzo e':  %f\n",prezzo);

/*Stima con il metodo Monte Carlo*/

varianza = ( media_quadrati / double(M) ) - mediaM*mediaM;
varianza = att * att * varianza;

/*Calcolo dell'Intervallo di Confidenza*/

ICp = prezzo + 1.96 * sqrt( (varianza) / double(M) );
ICm = prezzo - 1.96 * sqrt( ( varianza ) / double(M) );
printf("L'Intervallo di Confidenza al 95perc. e':  %f:\t ", ICm);
printf("e %f\n", ICp);
```

```
 system("PAUSE");
 return 0;
}
```

Infine attualizziamo i prezzi necessari e calcoliamo l'intervallo di confidenza. Quest'ultimo sarà sempre migliore, ossia sempre più piccolo, più sarà grande il numero M di simulazioni che viene assegnato in input e che rappresenta i controlli fatti dall'utente durante l'anno. A causa di questa barriera, le opzioni prese in considerazioni non avranno un prezzo altissimo e conseguentemente non permetteranno di ottenere un alto guadagno per il loro basso rischio.

4 Risultati ottenuti

E' stato compilato il programma per 3 volte assegnando alla barriera 2 diversi valori ($U = 110, 140$) facendo iterare il procedimento per M=100, 10.000 volte ottenendo i seguenti risultati.

U	M	N	Prezzo	Intervallo-	Intervallo+
110	100	365	12.900335	10.042479	15.758190
140	100	365	5.602935	2.847383	8.358487
110	10000	365	10.450542	10.157881	10.743203
140	10000	365	4.628303	4.355203	4.901402

Come risulta evidente dalla tabella che contiene i vari valori dei parametri U(altezza della barriera), M(numero si simulazioni), N(numero di periodi di tempo) il prezzo e l'intervallo non dipendono dagli stessi parametri. Il prezzo varia sensibilmente col variare della barriera, più la barriera è alta minore sarà il prezzo: poichè si sta considerando un'opzione poco rischiosa il guadagno non sarà elevato. L'intervallo di confidenza, invece, raggiunge

valori sensati e utili solo per valori grandi di M e N. Nei primi 2 casi in cui sono state fatte solo 100 simulazioni, entrambi i valori sono piuttosto piccoli e si ottiene un intervallo di ampiezza 6 rispetto al prezzo calcolato e risulta essere troppo grande per poter essere ritenuto accettabile. Negli altri due casi, grazie all'aumento di M, si ha un intervallo che raggiunge valori massimi e minimi molto vicini al prezzo, discostandosi da quest'ultimo non di unità ma di pochi decimi di unità, garantendo una maggiore sicurezza sul valore del prezzo.

Il programma può essere implementato per simulare anche gli altri tre tipi di opzioni, ovvero up-and-out, down-and-in e l'ultima down-and-out. Per una up-and-out bisogna cambiare il controllo della bandiera (flag) nel ciclo if in flag==0. Per il caso down bisogna considerare un aggiornamento del payoff solo se $S_n < K$. Per i casi in o out della down bisogna mettere il controllo della bandiera come nel caso della up.