



Università degli Studi di Firenze

Facoltà di Scienze Matematiche Fisiche e Naturali

# Metodi matematici per la finanza

Parte seconda - I metodi numerici

Docente: Prof. Vespri Vincenzo

Anno Accademico 2007-2008

# Indice

<b>1</b>	<b>I metodi delle differenze finite</b>	<b>1</b>
1.1	Introduzione . . . . .	1
1.2	Approssimazioni del metodo delle differenze finite . . . . .	3
1.3	La Mesh del metodo delle differenze finite . . . . .	6
1.4	Il metodo esplicito delle differenze finite . . . . .	7
1.5	I metodi impliciti delle differenze finite . . . . .	13
1.6	I metodi completamente impliciti . . . . .	14
1.6.1	Considerazioni pratiche . . . . .	16
1.6.2	Il metodo LU . . . . .	17
1.6.3	Il metodo SOR . . . . .	19
1.6.4	L'algoritmo del metodo implicito delle differenze finite . . . . .	25
1.7	Il metodo Crank-Nicolson . . . . .	26
<b>2</b>	<b>Metodi per le opzioni Americane</b>	<b>35</b>
2.1	Introduzione . . . . .	35
2.2	Formulazione del metodo delle differenze finite . . . . .	38
2.3	Il problema della matrice di restrizione . . . . .	40
2.4	Il Projected SOR . . . . .	41
2.4.1	Punto tecnico: il necessario per il Projected SOR . . . . .	44

2.5	L'algoritmo time-stepping . . . . .	45
2.5.1	Punto tecnico: le opzioni Bermudan . . . . .	48
2.6	Esempi numerici . . . . .	49
2.7	La convergenza del metodo . . . . .	49
<b>3</b>	<b>I metodi binomiali</b>	<b>53</b>
3.1	Introduzione . . . . .	53
3.2	Il cammino casuale discreto . . . . .	57
3.2.1	Il caso $u = 1/d$ . . . . .	59
3.2.2	Il caso $p = \frac{1}{2}$ . . . . .	61
3.2.3	L'albero binomiale . . . . .	62
3.3	Valutare le opzioni . . . . .	63
3.4	Opzioni Europee . . . . .	64
3.5	Opzioni Americane . . . . .	67
3.6	Redditi dividendi . . . . .	70

# Capitolo 1

## I metodi delle differenze finite

### 1.1 Introduzione

I metodi delle differenze finite ci permettono di ottenere soluzioni numeriche ad equazioni differenziali parziali e a problemi di complementarità lineare. Essi costituiscono una tecnica molto potente e flessibile che, se applicata correttamente, è in grado di generare soluzioni numeriche molto accurate. Questa tecnica viene applicata sia nelle scienze fisiche che finanziarie.

Come visto nei capitoli precedenti, una volta che l'equazione di Black-Scholes è stata ridotta all'equazione di diffusione diventa relativamente semplice trovare le soluzioni esatte (e convertirle nuovamente in variabili finanziarie). Infatti è molto più semplice trovare le soluzioni numeriche di un'equazione di diffusione e, per mezzo di un cambio di variabili, convertirle in soluzioni numeriche dell'equazione Black-Scholes di partenza piuttosto che risolvere numericamente l'equazione di Black-Scholes stessa. In questo capitolo quindi ci concentreremo nel risolvere le equazioni di diffusione per mezzo del metodo delle differenze finite.

Questo non vuol dire che uno non debba usare il metodo delle differenze finite per risolvere direttamente l'equazione di Black-Scholes (in particolare nei modelli multi-fattoriali). Ci sono molti esempi in cui non è fattibile, se non impossibile, ridurre il problema ad una equazione di diffusione a coefficiente costante (in una o più dimensioni); in questo caso non c'è altra scelta che applicare il metodo delle differenze finite all'equazione di Black-Scholes.

Riassumendo in breve quanto analizzato nei precedenti capitoli, l'equazione Black-Scholes per ogni opzione Europea può essere trasformata nell'equazione di diffusione tramite lo scambio delle variabili nel seguente modo:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}.$$

La funzione payoff per l'opzione determina le condizioni iniziali per  $u(x, \tau)$  e la condizione limite per l'opzione determina le condizioni all'infinito per  $u(x, \tau)$  (che è verificata per  $x \rightarrow \pm\infty$ ). Per una put option queste funzioni sono date da (come già analizzato nel capitolo 3):

$$P(S, T) = \max(E - S, 0) \text{ , } P(0, t) = Ee^{-r(T-t)}$$

$$P(S, t) \rightarrow 0 \text{ per } S \rightarrow \infty$$

Per una call option invece sono date da:

$$C(S, T) = \max(S - E, 0)$$

$$C(0, t) = 0$$

$$C(S, t) \sim S \text{ per } S \rightarrow \infty$$

mentre per una cash-or-nothing call la funzione payoff è data da:

$$g(x, \tau) = be^{\frac{1}{2}(k+1)^2\tau + \frac{1}{2}(k-1)x} H(x).$$

I valori dell'opzione  $V(S, t)$ , in variabili finanziarie, possono essere recuperati dal non-dimensionale  $u(x, \tau)$  usando  $S = Ee^x$ ,  $t = T - \tau/\frac{1}{2}\sigma^2$ ,  $C = Ev(x, \tau)$  producendo:

$$V = E^{\frac{1}{2}(1+k)} S^{\frac{1}{2}(1-k)} e^{\frac{1}{8}(k+1)^2\sigma^2(T-t)} u(\log(S/E), \frac{1}{2}\sigma^2(T-t)),$$

dove  $k = r/\frac{1}{2}\sigma^2$ .

## 1.2 Approssimazioni del metodo delle differenze finite

L'idea alla base dei metodi delle differenze finite è quella di sostituire le occorrenze delle derivate parziali all'interno di equazioni differenziali parziali tramite approssimazioni basate sull'espansione delle funzioni delle serie di Taylor vicino al punto o ai punti di interesse. Ad esempio, la derivata parziale  $\partial u/\partial \tau$  può essere definita come differenza limite:

$$\frac{\partial u}{\partial \tau}(x, \tau) = \lim_{\delta\tau \rightarrow 0} \frac{u(x, \tau + \delta\tau) - u(x, \tau)}{\delta\tau}.$$

Se, anziché prendere il limite  $\delta\tau \rightarrow 0$ , consideriamo  $\delta\tau$  non zero ma piccolo, otteniamo l'approssimazione:

$$(1.1) \quad \frac{\partial u}{\partial \tau}(x, \tau) \approx \frac{u(x, \tau + \delta\tau) - u(x, \tau)}{\delta\tau} + O(\delta\tau),$$

Questa è chiamata **approssimazione del metodo della differenza finita** o **differenza finita** per  $\partial u/\partial \tau$  poichè implica delle piccole, ma non infinitesimali, differenze della variabile dipendente  $u$ . Questa particolare approssimazione del metodo della differenza finita è chiamata **differenza di forward** dato che la differenza si trova più avanti (forward) rispetto a  $\tau$ ;

sono usati soltanto i valori di  $u$  tra  $\tau$  e  $\tau + \delta\tau$ . Come ci suggerisce il termine  $O(\delta\tau)$ , più piccolo è  $\delta\tau$  e più precisa sarà l'approssimazione.

Abbiamo anche:

$$\frac{\partial u}{\partial \tau}(x, \tau) = \lim_{\delta\tau \rightarrow 0} \frac{u(x, \tau) - u(x, \tau - \delta\tau)}{\delta\tau}$$

così che l'approssimazione

$$(1.2) \quad \frac{\partial u}{\partial \tau}(x, \tau) \approx \frac{u(x, \tau) - u(x, \tau - \delta\tau)}{\delta\tau} + O(\delta\tau),$$

è ugualmente un'approssimazione del metodo della differenza finita per  $\partial u / \partial \tau$ . Questa approssimazione del metodo delle differenze finite è chiamata **differenza di backward**.

Possiamo anche definire le differenze centrali notando che

$$\frac{\partial u}{\partial \tau}(x, \tau) = \lim_{\delta\tau \rightarrow 0} \frac{u(x, \tau + \delta\tau) - u(x, \tau - \delta\tau)}{2\delta\tau}.$$

Questo dà un aumento dell'approssimazione:

$$(1.3) \quad \frac{\partial u}{\partial \tau}(x, \tau) \approx \frac{u(x, \tau + \delta\tau) - u(x, \tau - \delta\tau)}{2\delta\tau} + O((\delta\tau)^2).$$

La figura 1.1 mostra un'interpretazione geometrica di questi tre tipi di differenze finite. Da notare che le differenze centrali sono più accurate (per  $\delta\tau$  piccolo) delle altre differenze di backward e forward.

Quando vengono applicate all'equazione di diffusione, le approssimazioni del metodo della differenza di forward e backward per  $\partial u / \partial \tau$  conducono rispettivamente agli schemi della differenza finita esplicita e completamente implicita. Le differenze centrali della forma (1.3) non sono mai usate nella pratica perchè portano sempre a schemi numerici scadenti (in particolare, a schemi che sono inerentemente instabili). Le differenze centrali della forma

$$(1.4) \quad \frac{\partial u}{\partial \tau} \approx \frac{u(x, \tau + \delta\tau/2) - u(x, \tau - \delta\tau/2)}{\delta\tau} + O((\delta\tau)^2)$$

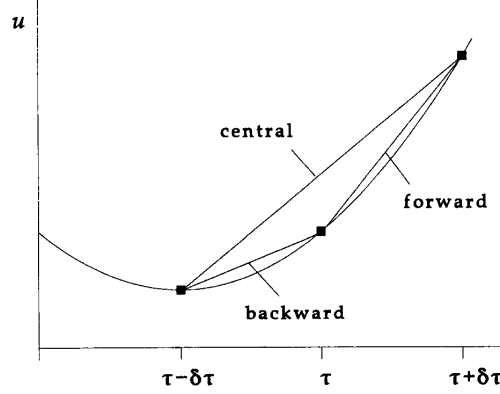


Figura 1.1: *Approssimazioni delle differenze centrali, backward e forward. La pendenza delle linee sono le approssimazioni alla tangente a  $(x, \tau)$ .*

danno origine allo schema del metodo delle differenze finite di **Crank-Nicolson**.

Possiamo definire le approssimazioni del metodo delle differenze finite per la derivata  $x$ -parziale di  $u$  nello stesso identico modo. Per esempio, è facile vedere che l'approssimazione centrale del metodo delle differenze finite è:

$$(1.5) \quad \frac{\partial u}{\partial x}(x, \tau) \approx \frac{u(x + \delta x, \tau) - u(x - \delta x, \tau)}{2\delta x} + O((\delta x)^2).$$

Per le derivate parziali seconde, come  $\partial^2 u / \delta x^2$ , possiamo definire un'approssimazione simmetrica del metodo delle differenze finite come la differenza di forward delle approssimazioni della differenza di backward alla derivata prima o come la differenza di backward delle approssimazioni della differenza di forward alla derivata prima. In entrambi i casi otteniamo l'approssimazione **simmetrica del metodo della differenza finita**

$$(1.6) \quad \frac{\partial^2 u}{\partial x^2}(x, \tau) \approx \frac{u(x + \delta x, \tau) - 2u(x, \tau) + u(x - \delta x, \tau)}{(\delta x)^2} + O((\delta x)^2).$$

Sebbene ci siano altre approssimazioni, questa approssimazione a  $\partial^2 u / \delta x^2$  è quella preferita dato che la sua simmetria preserva la simmetria di riflessione



della derivata parziale seconda; essa viene lasciata invariata dalle riflessioni della forma  $x \mapsto -x$ . E' anche molto più precisa di altre approssimazioni simili.

### 1.3 La Mesh del metodo delle differenze finite

Per continuare con il metodo delle differenze finite applicato all'equazione di diffusione, dividiamo da una parte l'asse  $x$  in **nodi** di uguale distanza  $\delta x$  e da una parte l'asse  $\tau$  in **nodi** di uguale distanza  $\delta \tau$ . Questa divide il piano

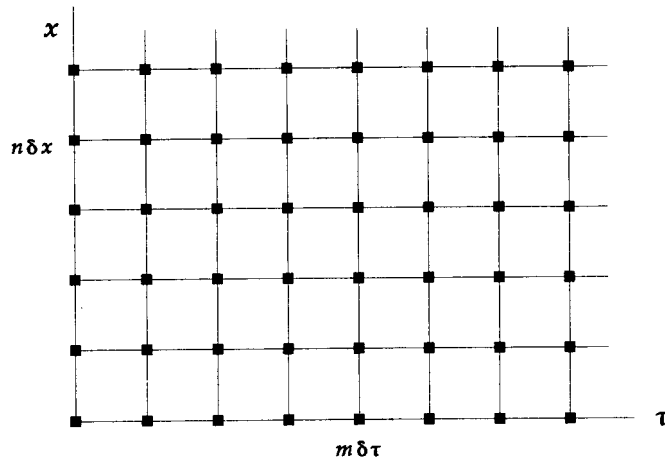


Figura 1.2: La Mesh per il metodo alle differenze finite.

$(x, \tau)$  in una mesh, dove i **punti mesh** hanno la forma  $(n\delta x, m\delta \tau)$ , come si vede nella figura 1.2. Allora ci concentriamo solo sui valori di  $u(x, \tau)$  ai punti mesh  $(n\delta x, m\delta \tau)$ ; vedi figura 8.3. Scriviamo

$$(1.7) \quad u_n^m = u(n\delta x, m\delta \tau)$$

per il valore di  $u(x, \tau)$  al punto mesh  $(n\delta x, m\delta \tau)$ .

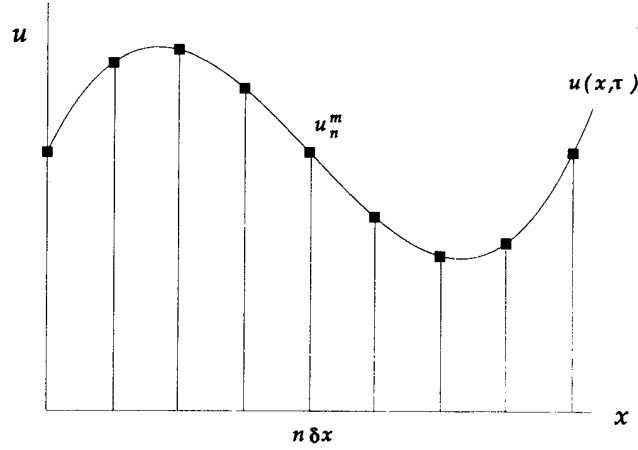


Figura 1.3: *L'approssimazione del metodo della differenza finita ad un punto fissato del tempo.*

## 1.4 Il metodo esplicito delle differenze finite

Consideriamo la forma generale del modello Black-Scholes trasformato per il valore di un'opzione Europea,

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2},$$

con limite e condizioni iniziali

$$(1.8) \quad \begin{aligned} u(x, \tau) &\sim u_{-\infty}(x, \tau), u(x, \tau) \sim u_{\infty}(x, \tau) \text{ per } x \rightarrow \pm\infty, \\ u(x, 0) &= u_0(x). \end{aligned}$$

Usiamo la notazione  $u_{-\infty}(\tau)$ ,  $u_{\infty}(\tau)$  e  $u_0(x)$  per evidenziare che la seguente non dipende in alcun modo dal particolare limite e dalle condizioni iniziali implicate. (Le put, calls e cash-or-nothing sono date come sopra).

Concentrando la nostra attenzione ai valori di  $u$  ai punti mesh, ed usando una differenza di forward per  $\partial u / \partial \tau$ , equazione (1.1), ed una differenza

#### 1.4. IL METODO ESPlicito DELLE DIFFERENZE FINITE

---

simmetrica centrale per  $\partial^2 u / \partial x^2$ , equazione (1.6), troviamo che l'equazione di diffusione (1.8) diventa

$$(1.9) \quad \frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta\tau)^2).$$

Ignorando i termini di  $O(\delta\tau)$  e  $O((\delta\tau)^2)$ , possiamo riadattare questa sopra per ottenere le equazioni di differenza

$$(1.10) \quad u_n^{m+1} = \alpha u_{n+1}^m + (1 - 2\alpha)u_n^m + \alpha u_{n-1}^m$$

dove

$$(1.11) \quad \alpha = \frac{\delta\tau}{(\delta x)^2}.$$

(Da notare che, mentre la (1.9) è esatta, anche se vaga per i termini di errore, la (1.10) è solo approssimata).

Se, al passo di tempo  $m$ , conosciamo  $u_n^m$  per tutti i valori di  $n$  possiamo calcolare esplicitamente  $u_n^{m+1}$ . Questo è il motivo per cui questo metodo è chiamato esplicito. Nota che  $u_n^{m+1}$  dipende solamente da  $u_{n+1}^m$ ,  $u_n^m$  e  $u_{n-1}^m$ , come mostrato in figura 1.4. Questa figura ci suggerisce anche che la (1.10) può essere considerata come un cammino casuale su un reticolo regolare, dove  $u_n^m$  denota la probabilità che un indice sia alla posizione  $n$  al passo di tempo  $m$ ,  $\alpha$  denota la probabilità che esso si sposti a destra o a sinistra di una unità e  $(1 - 2\alpha)$  la probabilità che stia fermo. Se scegliamo una costante  $x$ -spazio  $\delta x$ , non possiamo risolvere il problema per tutto l'intervallo  $-\infty < x < \infty$  senza prendere un numero infinito di  $x$ -passi. Possiamo aggirare questo problema prendendo un numero finito, ma adeguatamente grande, di  $x$ -passi. Restringiamo la nostra attenzione all'intervallo

$$N^- \delta x \leq x \leq N^+ \delta x$$

#### 1.4. IL METODO ESPlicito DELLE DIFFERENZE FINITE

---

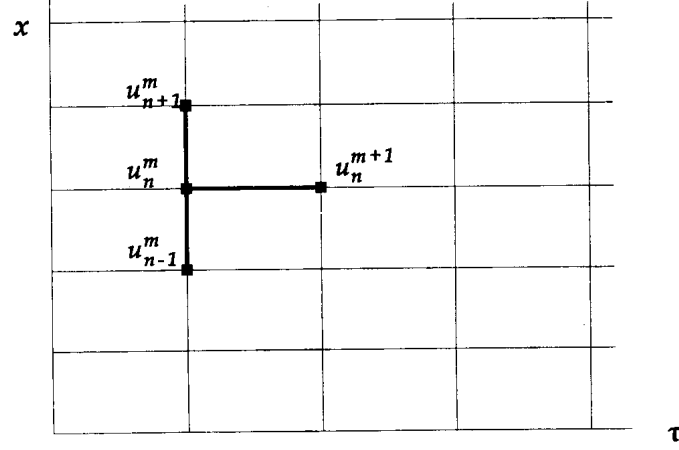


Figura 1.4: *Discretizzazione del metodo esplicito delle differenze finite.*

dove  $N^-$  e  $N^+$  sono interi positivi grandi.

Per ottenere la soluzione del metodo delle differenze finite per il prezzo dell'opzione, dividiamo il tempo di scadenza non-dimensionale dell'opzione,  $\frac{1}{2}\sigma^2 T$ , in  $M$  spazi di tempo uguali così che

$$\delta\tau = \frac{1}{2}\sigma^2 T/M.$$

Quindi risolviamo le equazioni di differenza (1.10) per  $N^- < n < N^+$  e  $0 < m \leq M$ , e usiamo le condizioni di limite dalla (1.8) per determinare  $u_{N^+}^m$  e  $u_{N^-}^m$ :

$$(1.12) \quad \begin{aligned} u_{N^-}^m &= u_{-\infty}(N^- \delta x, m\delta\tau), 0 < m \leq M, \\ u_{N^+}^m &= u_{\infty}(N^+ \delta x, m\delta\tau), 0 < m \leq M. \end{aligned}$$

Per iniziare la procedura iterativa usiamo le condizioni iniziali dalla (1.8):

$$(1.13) \quad u_n^0 = u_0(n\delta x), N^- \leq n \leq N^+.$$

Dato che le equazioni che determinano  $u_n^{m+1}$  in termini di  $u_n^m$  sono esplici-

```

explicit_fd( values,dx,dt,M,Nplus,Nminus )
{
    a = dt/(dx*dx);

    for( n=Nminus; n<=Nplus; ++n )
        oldu[n] = pay_off( n*dx );

    for( m=1; m<=M; ++m )
    {
        tau = m*dt;

        newu[Nminus] = u_m_inf( Nminus*dx,tau );
        newu[ Nplus] = u_p_inf(  Nplus*dx,tau );

        for( n=Nminus+1; n<Nplus; ++n )
            newu[n] = oldu[n]
                + a*(oldu[n-1]-2*oldu[n]+oldu[n+1]);

        for( n=Nminus; n<=Nplus; ++n )
            oldu[n] = newu[n];
    }

    for( n=Nminus; n<=Nplus; ++n )
        values[n] = oldu[n];
}

```

Figura 1.5: Lo pseudo-codice per la soluzione esplicita del metodo delle differenze finite per l'equazione di diffusione. Le variabili sono  $a = \alpha$ ,  $\tau = \tau$ ,  $N_{\text{minus}} = N^-$ ,  $N_{\text{plus}} = N^+$ . I valori di  $u_n^m$  vengono salvati nell'array `newu[]`. Inizialmente i valori di  $u_n^0$  vengono messi in `oldu[]` e il processo viene ripetuto fino a che tutti i passi nel tempo richiesti sono stati completati. La soluzione numerica viene copiata in `values[]` e viene ritornata al programma chiamante.

#### 1.4. IL METODO ESPlicito DELLE DIFFERENZE FINITE

te, questo processo può essere facilmente codificato per essere risolto da un calcolatore; lo pseudo-codice è dato dalla figura 1.5.

In figura 1.6 compariamo le soluzioni esplicite del metodo delle differenze finite per un'opzione put Europea con la formula esatta di Black-Scholes (che è stata ritrasformata nuovamente in variabili finanziarie). Abbiamo deliberatamente scelto di considerare  $\alpha$  e  $\delta\tau$  come variabili, piuttosto che la scelta più ovvia di  $\delta x$  e  $\delta\tau$ , per illustrare un punto di estrema importanza. Quando  $\alpha = 0.25$  e  $\alpha = 0.5$  le soluzioni esatte e quelle elaborate sono molto simili, mentre quando  $\alpha = 0.52$ , la soluzione elaborata non ha senso. Questo sotto illustra il **problema della stabilità** per il metodo le differenze finite esplicite. Il problema della stabilità si presenta perchè stiamo usan-

$S$	$\alpha = 0.25$	$\alpha = 0.50$	$\alpha = 0.52$	exact
0.00	9.7531	9.7531	9.7531	9.7531
2.00	7.7531	7.7531	7.7531	7.7531
4.00	5.7531	5.7531	5.7531	5.7531
6.00	3.7532	3.7532	2.9498	3.7532
7.00	2.7567	2.7567	-17.4192	2.7568
8.00	1.7986	1.7985	95.3210	1.7987
9.00	0.9879	0.9879	350.5603	0.9880
10.00	0.4418	0.4419	625.0347	0.4420
11.00	0.1605	0.1607	-457.3122	0.1606
12.00	0.0483	0.0483	-208.9135	0.0483
13.00	0.0124	0.0123	40.5813	0.0124
14.00	0.0028	0.0027	-15.2150	0.0028
15.00	0.0006	0.0005	-3.1582	0.0006
16.00	0.0001	0.0001	0.7365	0.0001

Figura 1.6: *Confronto tra la soluzione esatta di Black-Scholes e le soluzioni esplicite del metodo delle differenze finite per una put Europea con  $E = 10$ ,  $r = 0.05$ ,  $\lambda = 0.20$  e con sei mesi alla scadenza. Nota l'effetto ottenuto prendendo  $\alpha \geq \frac{1}{2}$*

do un computer aritmetico con precisione finita per risolvere le equazioni di differenza (1.10). Questo introduce errori di arrotondamento alle soluzioni numeriche della (1.10). Il sistema (1.10) viene detto **stabile** se questi errori di arrotondamento non aumentano ad ogni iterazione. Se gli errori di arrotondamento crescono ad ogni iterazione della procedura di soluzione, allora la (1.10) è detta **instabile**.

Può essere dimostrato che il sistema (1.10) è:

- stabile se  $0 < \alpha \leq \frac{1}{2}$  (**condizione di stabilità**);
- instabile se  $\alpha > \frac{1}{2}$  (**condizione di instabilità**).

Se consideriamo le equazioni esplicite del metodo delle differenze finite in termini di un cammino casuale, l'instabilità corrisponde alla presenza di probabilità negative (nello specifico, la probabilità che un indice stia fermo,  $1 - 2\alpha$ , è negativa).

La condizione di stabilità pone delle severe restrizioni alla grandezza dei passi nel tempo. Per la stabilità dobbiamo avere

$$0 < \frac{\delta\tau}{(\delta x)^2} \leq \frac{1}{2}.$$

Perciò se iniziamo con la soluzione stabile su una mesh e raddoppiamo il numero dei punti  $x$ -mesh, per esempio per migliorare l'accuratezza, dobbiamo dividere in quattro parti la misura dei passi nel tempo. Quindi ogni passo nel tempo viene raddoppiato in durata (due volte per ogni distanza  $x$ -mesh) e ce ne sono quattro volte di più. Quindi, raddoppiare il numero dei punti  $x$ -mesh significa che il ritrovamento della soluzione impiega otto volte di più.

Può essere dimostrato che la soluzione numerica delle equazioni del metodo delle differenze finite converge alla soluzione esatta dell'equazione di

diffusione per  $\delta x \rightarrow 0$  e  $\delta \tau \rightarrow 0$ , nel senso che

$$u_n^m \rightarrow u(n\delta x, m\delta \tau),$$

se e solo se il metodo esplicito delle differenze finite è stabile; la dimostrazione di questo va oltre lo scopo di questo libro e per questo la rimandiamo alla lettura di *Option Pricing*.

Da notare che a causa dell'indipendenza del metodo dalle condizioni iniziali e di limite, il metodo esplicito delle differenze finite è facilmente adattabile ad opzioni binarie e di barriera più generali.

## 1.5 I metodi impliciti delle differenze finite

I metodi impliciti delle differenze finite sono usati per superare le limitazioni di stabilità imposte dalla restrizione  $0 < \alpha \leq \frac{1}{2}$  che è applicata al metodo esplicito. I metodi impliciti ci permettono di usare un ampio numero di punti  $x$ -mesh senza dover prendere passi di tempo eccessivamente piccoli.

I metodi impliciti richiedono la risoluzione di equazioni di *sistemi*. Consideriamo le tecniche di decomposizione LU e SOR per risolvere numericamente questi sistemi. Usando queste tecniche i metodi impliciti possono essere svolti in maniera efficiente quasi quanto il metodo esplicito in termini di operazioni aritmetiche per passi di tempo. Dato che viene richiesto un numero minore di passi di tempo, i metodi impliciti di solito sono considerati complessivamente più efficienti dei metodi espliciti. Possiamo considerare sia i metodi completamente impliciti che quelli di Crank-Nicholson.



## 1.6 I metodi completamente impliciti

Lo schema completamente implicito della differenza finita, che di solito è conosciuto come il metodo **implicito delle differenze finite**, usa l'approssimazione della differenza di backward (1.2) per il termine  $\partial u / \partial \tau$  e l'approssimazione simmetrica del metodo delle differenze finite per il termine  $\partial^2 u / \partial x^2$ . Questo porta all'equazione

$$\frac{u_n^m - u_n^{m-1}}{\delta \tau} + O(\delta \tau) = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta x)^2)$$

dove abbiamo usato la stessa notazione della sezione precedente. Di nuovo, trascuriamo i termini di  $O(\delta \tau)$  e  $O((\delta x)^2)$  e, dopo qualche aggiustamento, troviamo le equazioni implicite del metodo delle differenze finite

$$(1.14) \quad -\alpha u_{n-1}^m + (1 + 2\alpha)u_n^m - \alpha u_{n+1}^m = u_n^{m-1}.$$

Come prima, i passi nello spazio e nel tempo sono in relazione tramite il

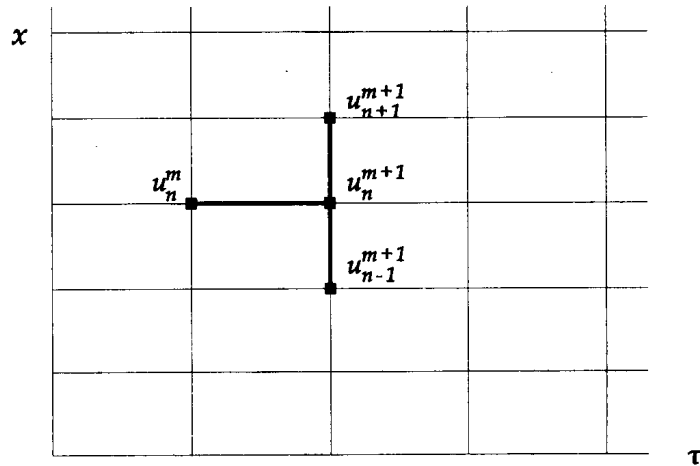


Figura 1.7: Discretizzazione del metodo implicito delle differenze finite

parametro  $\alpha$ , definito dalla (1.11). Nell'equazione implicita del metodo delle differenze finite (1.14),  $u_n^m$ ,  $u_{n-1}^m$  e  $u_{n+1}^m$  dipendono tutti da  $u_n^{m-1}$  in maniera implicita; i nuovi valori non possono essere immediatamente estrapolati e risolti esplicitamente nei termini dei vecchi valori. Lo schema è illustrato nella figura 1.7.

Consideriamo il problema dell'opzione Europea discusso nella precedente sessione. Assumiamo che possiamo troncare la mesh infinita ai valori  $x = N^-\delta x$  ed  $x = N^+\delta x$ , e prendiamo  $N^-$  e  $N^+$  sufficientemente grandi in modo da non introdurre errori significanti. Come prima troviamo  $u_n^0$  usando la (1.13) e  $u_{N^-}^m$  e  $u_{N^+}^m$  usando la (1.12). Il problema è dunque ricavare  $u_n^m$  per  $m \geq 1$  e  $N^- < n < N^+$  dalla (1.14).

Possiamo scrivere la (1.14) come il sistema lineare

$$\begin{aligned}
 (1.15) \quad & \begin{pmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 \\ -\alpha & 1+2\alpha & -\alpha & & 0 \\ 0 & -\alpha & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & -\alpha \\ 0 & 0 & & -\alpha & 1+2\alpha \end{pmatrix} \begin{pmatrix} u_{N^-+1}^m \\ \vdots \\ u_0^m \\ \vdots \\ u_{N^+-1}^m \end{pmatrix} \\
 &= \begin{pmatrix} u_{N^-+1}^{m-1} \\ \vdots \\ u_0^{m-1} \\ \vdots \\ u_{N^+-1}^{m-1} \end{pmatrix} + \alpha \begin{pmatrix} u_{N^-}^m \\ 0 \\ \vdots \\ 0 \\ u_{N^+}^m \end{pmatrix} = \begin{pmatrix} b_{N^-+1}^m \\ \vdots \\ b_0^m \\ \vdots \\ b_{N^+-1}^m \end{pmatrix}.
 \end{aligned}$$

Il vettore di destra della parte centrale in questa equazione deriva dalla parte finale delle equazioni, per esempio

$$(1+2\alpha)u_{N^+-1}^m - \alpha u_{N^+-2}^m = u_{N^+-1}^{m-1} + \alpha u_{N^+}^m.$$

Possiamo scrivere la (1.15) nella forma più compatta

$$(1.16) \quad Mu^m = b^m$$

dove  $u^m$  e  $b^m$  denotano gli  $(N^+ - N^- - 1)$ -vettori dimensionali  $u^m = (u_{N^-+1}^m, \dots, u_{N^+-1}^m)$ ,  $b^m = u^{m-1} + \alpha(u_{N^-}^m, 0, 0, \dots, 0, u_{N^+}^m)$ , e  $M$  è la  $(N^+ - N^- - 1)$ -matrice quadrata simmetrica data dalla (1.15).

Può essere dimostrato che, per  $\alpha \geq 0$ ,  $M$  è invertibile e quindi

$$(1.17) \quad u^m = M^{-1}b^m,$$

dove  $M^{-1}$  è l'inversa di  $M$ . Possiamo a questo punto trovare  $u^m$  dato da  $b^m$ , che a sua volta può essere ricavato da  $u^{m-1}$  e dalle condizioni di limite. Come la condizione iniziale determina  $u^0$ , possiamo trovare  $u^m$  sequenzialmente.

### 1.6.1 Considerazioni pratiche

Nella pratica ci sono tecniche di risoluzione più efficienti rispetto all'inversione della matrice. La matrice  $\mathbf{M}$  ha la proprietà di essere **tridiagonale**; ossia soltanto gli elementi della diagonale, della sopra-diagonale e della sotto-diagonale sono non-zero. Questo ha una serie di importanti conseguenze.

Per prima cosa non dobbiamo salvare tutti i valori zero ma solo i valori diversi da zero. L'inverso di  $M$ ,  $M^{-1}$ , non è tridiagonale e richiede un'operazione di salvataggio più onerosa.

Come seconda cosa, la struttura tridiagonale di  $M$  implica l'esistenza di algoritmi molto più efficienti per risolvere la (1.16) in  $O(N)$  operazioni aritmetiche per soluzione (in particolare circa  $4N$  operazioni). Adesso andremo ad analizzare due di questi algoritmi, la **decomposizione LU** ed il **SOR**.

### 1.6.2 Il metodo LU

Nella decomposizione LU cerchiamo di trovare la decomposizione della matrice  $M$  nel prodotto di una matrice triangolare inferiore  $L$  e di una matrice triangolare superiore  $U$ , che chiamiamo  $M = LU$ , della forma

$$(1.18) \quad \begin{pmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 \\ -\alpha & 1+2\alpha & -\alpha & & \vdots \\ 0 & -\alpha & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -\alpha \\ 0 & \cdots & 0 & -\alpha & 1+2\alpha \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \ell_{N^-+1} & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \ell_{N^+-2} & 1 \end{pmatrix} \begin{pmatrix} y_{N^-+1} & z_{N^-+1} & 0 & \cdots & 0 \\ 0 & y_{N^-+2} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & z_{N^+-2} \\ 0 & \cdots & 0 & 0 & y_{N^+-1} \end{pmatrix}.$$

Al fine di determinare le quantità  $\ell_n$ ,  $y_n$  e  $z_n$  (ed osservare che queste devono essere calcolate solo una volta) semplicemente moltiplichiamo insieme le due matrici della parte destra della (1.18) ed uguagliamo il risultato con la parte sinistra. Dopo alcune semplici manipolazioni troviamo che

$$(1.19) \quad \begin{aligned} y_{N^-+1} &= (1+2\alpha), \\ y_n &= (1+2\alpha) - \alpha^2/y_{n-1} \quad n = N^-+2, \dots, N^+-1, \\ z_n &= -\alpha, \quad \ell_n = -\alpha/y_n, \quad n = N^-+2, \dots, N^+-2. \end{aligned}$$

Questo mostra anche che le sole quantità che dobbiamo calcolare e salvare sono  $y_n$ ,  $n = N^-+1, \dots, N^+-1$ .

Il problema originale  $Mu^m = b^m$  può essere scritto come  $L(Uu^m) = b^m$ , che può essere successivamente divisa in due semplici sotto-problemi,

$$Lq^m = b^m, \quad Uu^m = q^m$$

dove  $q^m$  è un vettore intermedio. Quindi, avendo eliminato  $\ell_n$  dalla matrice triangolare inferiore e  $z_n$  dalla matrice triangolare superiore usando la (1.19), la procedura di risoluzione consiste nel risolvere i due sotto-problemi

(1.20)

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\frac{\alpha}{y_{N^-+1}} & 1 & 0 & & \vdots \\ 0 & -\frac{\alpha}{y_{N^-+2}} & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\frac{\alpha}{y_{N^+-2}} & 1 \end{pmatrix} \begin{pmatrix} q_{N^-+1}^m \\ q_{N^-+2}^m \\ \vdots \\ q_{N^+-2}^m \\ q_{N^+-1}^m \end{pmatrix} = \begin{pmatrix} b_{N^-+1}^m \\ b_{N^-+2}^m \\ \vdots \\ b_{N^+-2}^m \\ b_{N^+-1}^m \end{pmatrix}$$

e

(1.21)

$$\begin{pmatrix} y_{N^-+1} & -\alpha & 0 & \cdots & 0 \\ 0 & y_{N^-+2} & -\alpha & & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & & \ddots & y_{N^+-2} & -\alpha \\ 0 & \cdots & 0 & 0 & y_{N^+-1} \end{pmatrix} \begin{pmatrix} u_{N^-+1}^m \\ u_{N^-+2}^m \\ \vdots \\ u_{N^+-2}^m \\ u_{N^+-1}^m \end{pmatrix} = \begin{pmatrix} q_{N^-+1}^m \\ q_{N^-+2}^m \\ \vdots \\ q_{N^+-2}^m \\ q_{N^+-1}^m \end{pmatrix}.$$

Le quantità intermedie  $q_n^m$  sono facilmente ricavate da una sostituzione in avanti. Possiamo leggere il valore  $q_{N^-+1}^m$  direttamente, mentre ogni altra equazione nel sistema si relaziona solo a  $q_n^m$  e  $q_{n-1}^m$ . Se risolviamo il sistema con un ordine incrementale di  $n$  indici, abbiamo disponibile  $q_{n-1}^m$  nel momento in cui risolviamo  $q_n^m$ . Quindi possiamo semplicemente trovare  $q_n^m$ :

$$(1.22) \quad q_{N^-+1}^m = b_{N^-+1}^m, \quad q_n^m = b_n^m + \frac{\alpha q_{n-1}^m}{y_{n-1}}, \quad n = N^- + 2, \dots, N^+ - 1$$

In maniera simile, risolvendo la (1.21) per  $u_n^m$  (dato che abbiamo trovato il vettore intermedio  $q_n^m$ ) è facile applicare la sostituzione all'indietro (backward). Questa volta è  $u_{N^+-1}^m$  che può essere letto direttamente, e se risolveremo decrementando per un grandezza di  $n$  indici possiamo trovare tutte le  $u_n^m$  semplicemente ed in egual modo:

$$(1.23) \quad u_{N^+-1}^m = \frac{q_{N^+-1}^m}{y_{N^+-1}}, \quad u_n^m = \frac{q_n^m + \alpha u_{n+1}^m}{y_n}, \quad n = N^+ - 2, \dots, N^- + 1$$

Le equazioni (1.19), (1.22) e (1.23) definiscono l'algoritmo LU:

- trovare  $y_n$  usando la (1.19);
- dato il vettore  $b^m$ , usare la (1.22) per trovare il vettore  $q^m$ ;
- usare la (1.23) per trovare la soluzione  $u^m$  richiesta.

L'algoritmo è illustrato dallo pseudo-codice in figura 1.8. (Osserva che l'idea sopra può essere applicata anche ai sistemi tridiagonali in generale dove gli elementi superiori, inferiori e diagonali variano con la loro posizione nella matrice. Così si originano le matrici se viene applicato direttamente il metodo implicito all'equazione di Black-Scholes).

### 1.6.3 Il metodo SOR

Il metodo LU discusso nella precedente sezione è un metodo diretto per la soluzione del sistema (1.16) che tenta di trovare in modo esatto quantità non note in un solo passo. Una strategia alternativa è quella di impiegare un metodo *iterativo*. I metodi iterativi differiscono dai metodi diretti: infatti nel primo si inizia con un'ipotesi per la soluzione che viene successivamente migliorata fino a farla convergere alla soluzione esatta (o è abbastanza vicino

```
lu_find_y( y,a,Nminus,Nplus )
{
    asq = a*a;
    y[Nminus+1] = 1+2*a;

    for( n=Nminus+2; n<Nplus; ++n )
    {
        y[n] = 1+2*a - asq/y[n-1];
        if (y[n]==0) return(SINGULAR);
    }
    return( OK );
}

lu_solver( u,b,y,a,Nminus,Nplus )
{
    /* Must call lu_find_y before using this */

    q[Nminus+1] = b[Nminus+1];

    for( n=Nminus+2; n<Nplus; ++n )
        q[n] = b[n]+a*q[n-1]/y[n-1];

    u[Nplus-1] = q[Nplus-1]/y[Nplus-1];

    for( n=Nplus-2; n>Nminus; --n )
        u[n] = (q[n]+a*u[n+1])/y[n];
}
```

Figura 1.8: *Lo pseudo codice per risolvere LU tridiagonale. Le variabili sono  $a = \alpha$ ,  $asq = \alpha^2$ ,  $Nplus = N^+$ ,  $Nminus = N^-$ ,  $b[n] = b_n^m$ ,  $q[n] = q_n^m$ ,  $u[n] = u_n^m$ ,  $y[n] = y_n$ . Il problema viene risolto solo per  $Nminus + 1 \leq n \leq Nplus - 1$ ; si assume che i valori di  $Nplus$  e  $Nminus$  siano settati dal chiamante. Il chiamate deve prima eseguire `lu_find_y` prima di chiamare `lu_solver` per impostare  $y[]$ .*

alla soluzione esatta). Nel metodo diretto si ottiene la soluzione senza alcuna iterazione. Un vantaggio dei metodi iterativi, rispetto a quelli diretti, è quello di generalizzare in modo diretto i problemi delle opzioni americane e dei modelli non lineari che implicano costi di transazione, là dove invece i metodi diretti non riescono. Un altro vantaggio è che sono facili da programmare. Dall'altro lato, uno svantaggio dei metodi iterativi è che, nel contesto dei problemi delle opzioni Europee, sono più lenti rispetto ai metodi diretti.

Il nome *SOR* sta per *Successive Over-Relaxation* e l'algoritmo SOR è l'esempio di un metodo iterativo. Esso è il raffinamento di un altro metodo iterativo conosciuto come il metodo di **Gauss-Seidel**, che a sua volta è uno sviluppo del metodo di **Jacobi**. E' facile spiegare il metodo SOR descrivendo per prima questi altri metodi ad esso collegati ma più semplici. Tutti e tre i metodi relativi sono collegati dal fatto che il sistema (1.14)(o (1.15) oppure (1.16)) può essere scritto nella forma

$$(1.24) \quad u_n^m = \frac{1}{1 + 2\alpha} (b_n^m + \alpha(u_{n-1}^m + u_{n+1}^m)).$$

Questa equazione è un semplice riordinamento della (1.14), o della (1.15) o della (1.16) che isola i termini della diagonale all'interno del problema nel lato sinistro della equazione.

L'idea che stà alla base del metodo di Jacobi è quella di prendere qualche ipotesi iniziale di  $u_n^m$  per  $N^- + 1 \leq n \leq N^+ - 1$  (un'ipotesi iniziale buona è data dai valori di  $u$  recuperati dal passo precedente, cioè  $u_n^{m-1}$ ) e sostituirla nella parte destra della (1.24) per ottenere un nuovo valore ipotetico per  $u_n^m$  (dalla parte sinistra dell'equazione). Il processo viene ripetuto fino a che le approssimazioni smettono di cambiare (o smettono di cambiare significativamente). Quando questo accade abbiamo allora trovato la soluzione.



Formalmente possiamo definire il metodo di Jacobi come segue. Lasciamo che  $u_n^{m,k}$  sia la  $k$ -ima iterazione per  $u_n^m$ . Dunque, il valore iniziale ipotizzato è denotato da  $u_n^{m,0}$  e per  $k \rightarrow \infty$  ci aspettiamo che  $u_n^{m,k} \rightarrow u_n^m$ . Quindi, dato  $u_n^{m,k}$ , calcoliamo  $u_n^{m,k+1}$  usando una forma modificata della (1.24),

$$(1.25) \quad u_n^{m,k+1} = \frac{1}{1+2\alpha}(b_n^m + \alpha(u_{n-1}^{m,k} + u_{n+1}^{m,k})), \quad N^- < n < N^+.$$

L'intero processo viene ripetuto fino a che una misura dell'errore del tipo

$$\|u^{m,k+1} - u^{m,k}\|^2 = \sum_n (u_n^{m,k+1} - u_n^{m,k})^2$$

diventa sufficientemente piccolo per noi da ritenere ogni altra iterazione non necessaria; poi prendiamo  $u_n^{m,k+1}$  come valore di  $u_n^m$ . Il metodo converge alla corretta soluzione per ogni valore di  $\alpha > 0$ , anche se la completa discussione per la convergenza dell'algoritmo va oltre lo scopo di questo libro.

Il metodo **Gauss-Seidel** è uno sviluppo del metodo di Jacobi. Esso conta sul fatto che quando andiamo a calcolare  $u_n^{m,k+1}$  nella (1.25) abbiamo già trovato  $u_{n-1}^{m,k+1}$ . Nel metodo Gauss Seidel usiamo questo valore invece di  $u_{n-1}^{m,k}$ . Dunque, il metodo Gauss-Seidel è identico al metodo di Jacobi eccetto per la (1.25) che viene sostituita da

$$(1.26) \quad u_n^{m,k+1} = \frac{1}{1+2\alpha}(b_n^m + \alpha(u_{n-1}^{m,k+1} + u_{n+1}^{m,k})), \quad N^- < n < N^+.$$

La differenza può essere riassunta dicendo che, nel metodo Gauss-Seidel aggiorniamo immediatamente il valore ipotetico quando esso diventa disponibile mentre nel metodo Jacobi aggiorniamo il valore ipotetico solo quando sono tutti disponibili. Una conseguenza pratica dell'uso dell'informazione più recente (cioè  $u_n^{m,k+1}$  anzichè  $u_n^{m,k}$ ) è che il metodo Gauss-Seidel converge più rapidamente del metodo di Jacobi e quindi è più efficiente. Infatti il metodo

Gauss-Seidel è più efficiente dato che l'aggiornamento è ottenuto dalla sovrascrittura delle iterazioni precedenti mentre nel metodo di Jacobi le vecchie e le nuove iterazioni vengono salvate separatamente fino a che non sono state trovate tutte le nuove iterazioni (e quindi copiate sopra le vecchie iterazioni). Ancora una volta può essere dimostrato che l'algoritmo Gauss-Seidel converge alla soluzione corretta se  $\alpha$  è positivo, per gli altri casi non possiamo dimostrarlo adesso.

L'algoritmo SOR è una rifinitura del metodo Gauss-Seidel. Cominciamo con l'osservazione (apparentemente insignificante) che

$$u_n^{m,k+1} = u_n^{m,k} + (u_n^{m,k+1} - u_n^{m,k}).$$

Dato che la sequenza di iterazioni  $u_n^{m,k}$  converge a  $u_n^m$  per  $k \rightarrow \infty$ , possiamo pensare a  $(u_n^{m,k+1} - u_n^{m,k})$  come un termine di correzione che deve essere aggiunto a  $u_n^{m,k}$  per portarlo il più vicino possibile all'esatto valore di  $u_n^m$ . Siamo dunque in grado di ottenere una sequenza che converge più rapidamente se proseguiamo in maniera corretta; questo è vero se la sequenza di iterazioni  $u_n^{m,k} \rightarrow u_n^m$  è monotona all'incrementare di  $k$  piuttosto che oscillante. Questo è il caso sia del metodo Gauss-Seidel che del metodo SOR. Mettiamo

$$(1.27) \quad \begin{aligned} y_n^{m,k+1} &= \frac{1}{1+2\alpha} (b_n^m + \alpha(u_{n-1}^{m,k+1} + u_{n+1}^{m,k})) \\ u_n^{m,k+1} &= u_n^{m,k} + \omega(y_n^{m,k+1} - u_n^{m,k}), \end{aligned}$$

dove  $\omega > 1$  è il parametro di correzione superiore o **relazione superiore**. (Osserva che il termine  $y_n^{m,k+1}$  è il valore che il metodo Gauss-Seidel può dare per  $u_n^{m,k+1}$ ; nel SOR vediamo  $y_n^{m,k+1} - u_n^{m,k}$  come una correzione che deve essere fatta a  $u_n^{m,k}$  per ottenere  $u_n^{m,k+1}$ ). Può essere dimostrato che l'algoritmo SOR converge alla soluzione corretta della (1.14) o (1.16) se  $\alpha > 0$

```
SOR_solver( u,b,Nminus,Nplus,a,omega,eps,loops )
{
    loops = 0;
    do
    {
        error = 0.0;
        for( n=Nminus+1; n<Nplus; ++n )
        {
            y = ( b[n]+a*(u[n-1]+u[n+1]) )/(1+2*a);
            y = u[n]+omega*(y-u[n]);
            error += (u[n]-y)*(u[n]-y);
            u[n]=y;
        }
        ++loops;
    }
    while ( error > eps );
    return(loops);
}
```

Figura 1.9: *Pseudo codice per l'algoritmo SOR per un problema di opzione Europea. Qui c'è  $a = \alpha$ ,  $Nplus = N^+$ ,  $Nminus = N^-$ ,  $b[n] = b_n^m$ ,  $u[n] = u_n^{m,k}$  o  $u_n^{m,k+1}$ ,  $y = y_n^{m,k+1}$ ,  $omega = \omega$  e  $eps$  è l'errore di tolleranza desiderato. La procedura sovrascrive le vecchie iterazioni fino a che non ne vengono generate di nuove; perciò per un dato valore di  $n$  nel ciclo,  $u[n-1]$ ,  $u[n-2]$ , e così avanti, conterrà  $u_{n-1}^{m,k+1}$ ,  $u_{n-2}^{m,k+1}$ , etc., mentre  $u[n+1]$ ,  $u[n+2]$ , e così avanti, conterrà  $u_{n+1}^{m,k}$ ,  $u_{n+2}^{m,k}$ , etc.. L'algoritmo converge una volta che la somma su  $n$  dei quadrati delle differenze tra  $u_n^{m,k+1}$  e  $u_n^{m,k}$  è più piccola dell'errore di tolleranza  $eps$ . A questo punto l'array  $u[]$  contiene la soluzione SOR per  $u_n^m$ . Osserva che l'algoritmo risolve il problema solo per  $Nminus+1 \leq n \leq Nplus-1$ ; assumiamo che i valori a  $Nplus$  e  $Nminus$  devono essere settati dalla procedura chiamante. La procedura ritorna il numero di iterazioni eseguite nel loop; questo per far sì che la procedura chiamante possa aggiustare  $omega$  per minimizzare il numero di iterazioni.*

e fornisce  $0 < \omega < 2$ . (Quando  $0 < \omega < 1$  l'algoritmo si riferisce alla relazione inferiore piuttosto che a quella superiore, che è usata nei casi in cui  $1 < \omega < 2$ ). Può essere dimostrato che esiste un valore ottimale per  $\omega$ , nell'intervallo  $1 < \omega < 2$ , che conduce ad un valore di convergenza più rapido rispetto ad altri  $\omega$ . Questo valore ottimale di  $\omega$  dipende dalla dimensione delle matrici coinvolte e, in generale, dal dettaglio delle matrici coinvolte. Ci sono mezzi di calcolo o di previsione del valore ottimale  $\omega$  ma tipicamente questi comportano così tanti calcoli che risulta più rapido cambiare  $\omega$  ad ogni passo fino a che non viene trovato il valore che minimizza il numero di iterazioni del loop del metodo SOR. In figura 1.9 è dato l'algoritmo SOR per le equazioni implicite delle differenze finite per l'equazione di diffusione.

#### 1.6.4 L'algoritmo del metodo implicito delle differenze finite

Lo schema di soluzione implicita del metodo delle differenze finite è utilizzato per risolvere la (1.16) (oppure la (1.14) o la (1.15)) in ogni passo nel tempo usando o la procedura LU descritta nella sezione (1.6.2) o l'algoritmo SOR descritto nella sezione precedente. Questo ci permette di attraversare i vari passi di tempo e calcolare il valore corrente dell'opzione. L'algoritmo che usa il metodo LU è illustrato in figura 1.10 mentre quello che usa il metodo SOR è illustrato in figura 1.11.

In figura 1.12 mettiamo a confronto le soluzioni implicite delle differenze finite per una put Europea con tre mesi di tempo alla scadenza, prezzo di esercizio  $E = 10$ , volatilità  $\sigma = 0.4$  e tasso di interesse a libero rischio  $r = 0.1$  con la formula esatta di Black-Scholes. Come per il metodo esplicito, viene

prima fissato lo spazio  $x$ -mesh ed in seguito determinati i passi di tempo da  $\alpha$ . Abbiamo scelto, come prima, di considerare  $\alpha$  e  $\delta\tau$  come variabili per evidenziare un punto importante riguardo la stabilità. Se  $\alpha = 0.5$ ,  $\alpha = 1.0$  oppure  $\alpha = 5.0$ , la soluzione elaborata coincide molto con la soluzione esatta e certamente non ci sono evidenti instabilità come quelle viste nella soluzione esplicita del metodo delle differenze finite quando  $\alpha > \frac{1}{2}$ .

Questo illustra il fatto che lo schema implicito è stabile dove lo schema implicito risulta instabile (ossia per  $\alpha > \frac{1}{2}$ ). Infatti possiamo mostrare che il metodo implicito delle differenze finite è stabile per ogni  $\alpha > 0$ . La conseguenza è che possiamo risolvere l'equazione di diffusione con spazi di tempo più grandi ottenuti con l'utilizzo di un algoritmo implicito piuttosto che utilizzando un algoritmo esplicito. Questo porta a soluzioni numeriche più efficienti; anche se nel metodo implicito ogni passo di tempo dura di più, è comunque richiesto un numero minore di passi di tempo che va a compensare la durata. La convergenza dell'approssimazione del metodo implicito delle differenze finite alla soluzione dell'equazione differenziale parziale può essere provata. Ancora una volta, come lo schema esplicito delle differenze finite, esso converge se e solo se è stabile.

## 1.7 Il metodo Crank-Nicolson

Il metodo delle differenze finite di Crank-Nicolson è usato per superare le limitazioni di stabilità imposte dalle restrizioni di stabilità e convergenza del metodo esplicito delle differenze finite ed anche per avere  $O((\delta\tau)^2)$  tassi di convergenza alla soluzione dell'equazione differenziale parziale. (Il tasso di convergenza del metodo implicito ed esplicito è  $O(\delta\tau)$ ).

```
implicit_fdi( values,dx,dt,M,Nminus,Nplus )
{
    a = dt/(dx*dx);

    for( n=Nminus; n<=Nplus; ++n )
        values[n] = pay_off(n*dx);

    lu_find_y( y,a,Nminus,Nplus );

    for( m=1; m<=M; ++m )
    {
        tau = m*dt;

        for( n=Nminus+1; n<Nplus; ++n )
            b[n] = values[n];

        values[Nminus] = u_m_inf( Nminus*dx, tau );
        values[ Nplus] = u_p_inf( Nplus*dx, tau );
        b[Nminus+1] += a*values[Nminus];
        b[ Nplus-1] += a*values[ Nplus];

        lu_solver( values,b,y,a,Nminus,Nplus );
    }
}
```

Figura 1.10: *Pseudo codice per un risolutore implicito utilizzando la decomposizione LU.  $M$  è il numero di passi di tempo e  $a = \alpha$ . Osserva che dobbiamo chiamare `lu_find_y` una (ed una sola) volta prima di usare la procedura `lu_solver`. Salviamo dunque i valori iniziali nell'array `values[]` e chiamiamo ripetutamente `lu_solver` nei vari passi di tempo fino alla scadenza. Osserva la correzione del valore limite applicata ai valori finali  $b[N_{\text{minus}} + 1]$  e  $b[N_{\text{plus}} - 1]$ .*

```
implicit_fd2( values,dx,dt,M,Nminus,Nplus )
{
    a = dt/(dx*dx);
    eps = 1.0e-8;
    omega = 1.0;
    domega = 0.05;
    oldloops = 10000;

    for( n=Nminus; n<=Nplus; ++n )
        values[n] = pay_off(n*dx);

    for( m=1; m<=M; ++m )
    {
        tau = m*dt;

        for( n=Nminus+1; n<Nplus; ++n )
            b[n] = values[n];

        values[Nminus] = u_m_inf( Nminus*dx, tau );
        values[Nplus] = u_p_inf( Nplus*dx, tau );

        SOR_solver( values,b,Nminus,Nplus,a,omega,eps,loops );
        if ( loops > oldloops ) domega *= -1.0;
        omega += domega;
        oldloops = loops;
    }
}
```

Figura 1.11: *Pseudo codice per un risolutore implicito che utilizza il metodo SOR.  $M$  è il numero di passi nel tempo,  $a = \alpha$ ,  $eps$  è l'errore di tolleranza e  $omega$  il parametro SOR. Come nel precedente pseudo codice, la procedura prima inserisce i valori iniziali nell'array `values[]` e poi chiama ripetutamente `SOR_solver` fino alla scadenza. Non c'è bisogno di applicare le correzioni del valore limite ai valori finali `b[Nminus + 1]` e `b[Nplus - 1]` dato che la procedura di SOR lo fa in automatico. La procedura incrementa  $omega$  del valore `domega` ad ogni passo. Se questi risultati nel numero di iterazioni per il passo corrente, `loops`, eccedono le iterazioni del loop precedente, `oldloops`, allora il segno di `domega` viene cambiato così che  $omega$  viene spostata indietro verso il valore che minimizza il numero di iterazioni.*

## 1.7. IL METODO CRANK-NICOLSON

---

Lo schema implicito del metodo delle differenze finite di Crank-Nicolson è essenzialmente un metodo a metà tra uno implicito ed uno esplicito. In parti-

$S$	$\alpha = 0.50$	$\alpha = 1.00$	$\alpha = 5.00$	exact
0.00	9.7531	9.7531	9.7531	9.7531
2.00	7.7531	7.7531	7.7531	7.7531
4.00	5.7531	5.7531	5.7530	5.7531
6.00	3.7569	3.7570	3.7573	3.7569
8.00	1.9025	1.9025	1.9030	1.9024
10.00	0.6690	0.6689	0.6675	0.6694
12.00	0.1674	0.1674	0.1670	0.1675
14.00	0.0327	0.0328	0.0332	0.0326
16.00	0.0054	0.0055	0.0058	0.0054

Figura 1.12: *Confronto tra la soluzione esatta della Black-Scholes e quelle del metodo completamente implicito delle differenze finite per una put Europea con  $E = 10$ ,  $r = 0.1$ ,  $\sigma = 0.4$  e tre mesi alla scadenza. Anche con  $\alpha = 5.0$  i risultati sono accurati a due decimali.*

colare, se usiamo l'approssimazione della differenza di forward alla derivativa parziale di tempo otteniamo lo schema esplicito

$$\frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta x)^2),$$

e se prendiamo una differenza di backward otteniamo lo schema implicito

$$\frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2} + O((\delta x)^2).$$

La media di queste due equazioni è

$$(1.28) \quad \frac{1}{2} \left( \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + \frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2} \right) + O((\delta x)^2).$$



Infatti può essere dimostrato che i termini della (1.28) sono accurati a  $O((\delta\tau)^2)$ , piuttosto che  $O(\delta\tau)$ . Ignorando l'errore i termini ci conducono allo schema Crank-Nicolson

$$(1.29) \quad u_n^{m+1} - \frac{1}{2}\alpha(u_{n-1}^{m+1} - 2u_n^{m+1} + u_{n+1}^{m+1}) = u_n^m + \frac{1}{2}\alpha(u_{n-1}^m - 2u_n^m + u_{n+1}^m)$$

dove, come prima,  $\alpha = \delta\tau/(\delta x)^2$ . Osserva che  $u_n^{m+1}$ ,  $u_{n-1}^{m+1}$  e  $u_{n+1}^{m+1}$  adesso sono determinate implicitamente in termini di  $u_n^m$ ,  $u_{n+1}^m$  e  $u_{n-1}^m$ . Risolvere questo sistema di equazioni, in principio, non differisce dal risolvere le equazioni (1.14) per lo schema implicito. Questo perchè ogni elemento del lato destro della (1.29) può essere valutato esplicitamente se  $u_n^m$  sono conosciute. Il problema quindi si riduce a calcolare per primo

$$(1.30) \quad Z_n^m = (1 - \alpha)u_n^m + \frac{1}{2}\alpha(u_{n-1}^m + u_{n+1}^m),$$

che è la formula esplicita per  $Z_n^m$ , e quindi risolvere

$$(1.31) \quad (1 + \alpha)u_n^{m+1} - \frac{1}{2}\alpha(u_{n-1}^{m+1} + u_{n+1}^{m+1}) = Z_n^m.$$

Questo secondo problema è essenzialmente lo stesso che risolvere la (1.14).

Ancora una volta assumiamo di poter troncare la mesh infinita a  $x = N^-\delta x$  e  $x = N^+\delta x$ , e prendiamo  $N^-$  e  $N^+$  sufficientemente grandi da non introdurre errori significanti. Come prima possiamo calcolare  $u_n^0$  usando la (1.13) e  $u_{N^-}^m$  e  $u_{N^+}^m$  dalle condizioni di limite (1.12).

Rimane il problema di trovare  $u_n^m$  per  $m \geq 1$  e  $N^- < n < N^+$  dalla (1.13). Possiamo scrivere il problema come un sistema lineare

$$(1.32) \quad Cu^{m+1} = b^m$$

dove la matrice  $C$  è data da

$$(1.33) \quad C = \begin{pmatrix} 1 + \alpha & -\frac{1}{2}\alpha & 0 & \cdots & 0 \\ -\frac{1}{2}\alpha & 1 + \alpha & -\frac{1}{2}\alpha & & \vdots \\ 0 & -\frac{1}{2}\alpha & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -\frac{1}{2}\alpha \\ 0 & 0 & & -\frac{1}{2}\alpha & 1 + \alpha \end{pmatrix}$$

e i vettori  $u^{m+1}$  e  $b^m$  sono dati da

$$(1.34) \quad u^{m+1} = \begin{pmatrix} u_{N^-+1}^{m+1} \\ \vdots \\ u_0^{m+1} \\ \vdots \\ u_{N^+-1}^{m+1} \end{pmatrix}, b^m = \begin{pmatrix} Z_{N^-+1}^m \\ \vdots \\ Z_0^m \\ \vdots \\ Z_{N^+-1}^m \end{pmatrix} + \frac{1}{2}\alpha \begin{pmatrix} u_{N^-}^{m+1} \\ 0 \\ \vdots \\ 0 \\ u_{N^+}^{m+1} \end{pmatrix}.$$

Il vettore della parte più a destra dell'equazione (1.34), in  $b^m$ , conduce alle condizioni di limite applicate alla fine, come nel metodo completamente implicito delle differenze finite.

Per implementare lo schema Crank-Nicholson dobbiamo per prima cosa formare il vettore  $b^m$  usando quantità conosciute. Poi usiamo un risolutore LU tridiagonale o un risolutore SOR per risolvere il sistema (1.32). Questo ci permette di giungere alla soluzione. L'unica differenza tra i risolutori LU e SOR per i metodi Crank-Nicholson e quelli pienamente impliciti è che ogni volta che  $\alpha$  appare nell'algoritmo per lo schema implicito, lo sostituiamo con  $\frac{1}{2}\alpha$  nello schema di Crank-Nicholson. Nella figura 1.13 e 1.14 diamo gli pseudo-codici per il metodo di Crank-Nicholson usando rispettivamente i metodi di risoluzione LU e SOR. In figura 1.15 compariamo le soluzioni del metodo delle differenze finite di Crank-Nicholson per una put Europea con

```
crank_fd1( val,dx,dt,M,Nminus,Nplus )
{
    a = dt/(dx*dx);
    a2 = a/2.0;

    for( n=Nminus; n<=Nplus; ++n )
        val[n] = pay_off( n*dx );

    lu_find_y( y,a2,Nminus,Nplus );

    for( m=1; m<=M; ++m )
    {
        tau = m*dt;
        for( n=Nminus+1; n<Nplus; ++n )
            b[n] = (1-a)*val[n]+a2*(val[n+1]+val[n-1]);

        val[Nminus] = u_m_inf( Nminus*dx,tau );
        val[ Nplus] = u_p_inf( Nplus*dx,tau );
        b[Nminus+1] += a2*val[Nminus];
        b[ Nplus-1] += a2*val[ Nplus];

        lu_solver( val,b,y,a2,Nminus,Nplus );
    }
}
```

Figura 1.13: *Pseudo codice per il risolutore Crank-Nicholson usando il metodo LU. La soluzione dopo  $M$  passi nel tempo viene salvata nell'array `val[]`. Qui abbiamo  $a = \alpha$  e  $a2 = \alpha/2$ . Osserva le correzioni limite a  $b[Nminus + 1]$  e  $b[Nplus - 1]$ . Osserva anche che il codice è abbastanza identico al codice in figura 1.10; le differenze principali sono nell'uso di  $\alpha/2$  e nel calcolo di  $b[n]$ .*

```
crank_fd2( val,dx,dt,M, Nminus,Nplus )
{
    a = dt/(dx*dx);
    a2 = a/2.0;
    eps = 1.0e-8;
    omega = 1.0;
    domega = 0.05;
    oldloops = 10000;

    for( n=Nminus; n<=Nplus; ++n )
        val[n] = pay_off( n*dx );

    for( m=1; m<=M; ++m )
    {
        tau = m*dt;

        for( n=Nminus+1; n<Nplus; ++n )
            b[n] = (1-a)*val[n]+a2*(val[n+1]+val[n-1]);

        val[Nminus] = u_m_inf(Nminus*dx,tau);
        val[Nplus] = u_p_inf(Nplus*dx,tau);

        SOR_solver( val,b,Nminus,Nplus,a2,omega,eps,loops );
        if ( loops > oldloops ) domega *= -1.0;
        omega += domega;
        oldloops = loops;
    }
}
```

Figura 1.14: *Pseudo codice per il risolutore Crank-Nicholson usando il metodo SOR. La soluzione dopo  $M$  passi nel tempo viene salvata nell'array  $val[]$ . Qui abbiamo  $a = \alpha$  e  $a2 = \alpha/2$ . Osserva le correzioni limite a  $b[Nminus + 1]$  e  $b[Nplus - 1]$ . Osserva anche che il codice è abbastanza identico al codice in figura 1.11; le differenze sono nell'uso di  $\alpha/2$  anzichè  $\alpha$  e nel calcolo di  $b[n]$ .*

### 1.7. IL METODO CRANK-NICOLSON

---

quattro mesi alla scadenza, prezzo di esercizio 10, volatilità  $\sigma = 0.45$  e tasso di interesse senza rischio  $r = 0.1$  con la formula esatta Black-Scholes. Da notare che con  $\alpha = \frac{1}{2}$ ,  $\alpha = 1.0$  o anche  $\alpha = 10.0$ , la soluzione ottenuta è molto vicina alla soluzione esatta. Questo dimostra che lo schema Crank-Nicholson è stabile dove lo schema esplicito risulta instabile (che è vero per  $\alpha > \frac{1}{2}$ ). Inoltre la sua precisione è molto più grande di quella dello schema completamente implicito.

Possiamo dimostrare che lo schema Crank-Nicholson è sia stabile che convergente per tutti i valori di  $\alpha > 0$ .

$S$	$\alpha = 0.50$	$\alpha = 1.00$	$\alpha = 10.00$	exact
0.00	9.6722	9.6722	9.6722	9.6722
2.00	7.6721	7.6721	7.6721	7.6722
4.00	5.6722	5.6722	5.6723	5.6723
6.00	3.6976	3.6976	3.6975	3.6977
8.00	1.9804	1.9804	1.9804	1.9806
10.00	0.8605	0.8605	0.8566	0.8610
12.00	0.3174	0.3174	0.3174	0.3174
14.00	0.1047	0.1047	0.1046	0.1046
16.00	0.0322	0.0322	0.0321	0.0322

Figura 1.15: *Confronto tra le soluzioni del metodo delle differenze finite di Crank-Nicholson e l'esatta Black-Scholes per una put Europea con  $E = 10$ ,  $r = 0.1$ ,  $\sigma = 0.45$  e quattro mesi alla scadenza. Anche con  $\alpha = 10$  i risultati esatti e numerici differiscono solo marginalmente.*

# Capitolo 2

## Metodi per le opzioni Americane

### 2.1 Introduzione

L'uso dei metodi delle differenze finite per le opzioni Americane è relativamente diretto, dato che non c'è possibilità di esercizio immediato. Come abbiamo visto, la possibilità di esercizio immediato può condurre all'uso di limiti liberi. Il problema principale con i limiti liberi, dal punto di vista dell'analisi numerica, è che non sappiamo dove sono. Questo rende difficile imporre le condizioni dei limiti liberi poichè dobbiamo determinare dove imporre tali condizioni come parte della procedura di soluzione. (Ricordiamo che nel capitolo precedente abbiamo semplicemente imposto le condizioni di limite ai punti fissati della griglia).

Ci sono due strategie distinte per la soluzione numerica dei problemi dei limiti liberi. Una è tentare di tenere traccia del limite libero come parte del processo di time-step. Nel contesto di valutazione delle opzioni Americane

questo non è un metodo particolarmente attraente dato che le condizioni dei limiti liberi sono entrambe implicite - ossia non forniscono un'espressione diretta per il limite libero o per la sua derivata nel tempo. Qui osserviamo semplicemente l'esistenza di tali metodi e rimandiamo il lettore ad altre letture per una discussione delle varie strategie che analizzano più a fondo i problemi dei limiti liberi impliciti.

L'altra strategia è quella di tentare di trovare una trasformazione che riduce il problema ad un problema di limite fissato dal quale il limite libero può essere dedotto in seguito. Ci sono varie trasformazioni che fanno questo ma noi considereremo solo i metodi più eleganti che portano all'uso della formulazione della complementarità lineare.

Come già analizzato nei capitoli precedenti, possiamo scrivere il problema di valutazione delle opzioni Americane nella forma di complementarità lineare compatta

$$(2.1) \quad \begin{aligned} \left( \frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2} \right) &\geq 0, & (u(x, \tau) - g(x, \tau)) &\geq 0, \\ \left( \frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2} \right) \cdot (u(x, \tau) - g(x, \tau)) &= 0, \end{aligned}$$

dove utilizziamo il cambio di variabili. La funzione di restrizione della payoff trasformata,  $g(x, \tau)$ , è data da

$$g(x, \tau) = e^{\frac{1}{4}(k+1)^2\tau} \max(e^{\frac{1}{2}(k-1)x} - e^{\frac{1}{2}(k+1)x}, 0)$$

per la put,

$$g(x, \tau) = e^{\frac{1}{4}(k+1)^2\tau} \max(e^{\frac{1}{2}(k+1)x} - e^{\frac{1}{2}(k-1)x}, 0)$$

per la call e

$$g(x, \tau) = e^{\frac{1}{4}(k+1)^2\tau} \times \begin{cases} 0 & x < 0 \\ be^{\frac{1}{2}(k-1)x} & x \geq 0 \end{cases}$$

per la cash-or-nothing call e dove  $k = r/\frac{1}{2}\sigma^2$ . Le condizioni iniziali fissate del limite diventano

$$\begin{aligned}
 (2.2) \quad & u(x, 0) = g(x, 0), \\
 & u(x, \tau) \text{ è continua,} \\
 & \frac{\partial u}{\partial x}(x, \tau) \text{ è continua come } g(x, \tau), \\
 & \lim_{x \rightarrow \pm\infty} u(x, \tau) = \lim_{x \rightarrow \pm\infty} g(x, \tau).
 \end{aligned}$$

Questa struttura conduce in maniera piuttosto ovvia a funzioni di payoff più generali rispetto alle tre forme per  $g(x, \tau)$  date sopra.

Come già fatto notare, uno dei vantaggi principali della formulazione di complementarità lineare della (1.1) è che non c'è nessun accenno esplicito del limite libero. Se possiamo risolvere il problema della complementarità lineare allora troviamo il limite libero  $x = x_f(\tau)$  a posteriori dalla condizione che lo definisce, cioè

$$u(x_f(\tau), \tau) = g(x_f(\tau), \tau), \quad \text{ma} \quad u(x, \tau) > g(x, \tau) \text{ per } x > x_f(\tau)$$

per la put, e

$$u(x_f(\tau), \tau) = g(x_f(\tau), \tau), \quad \text{ma} \quad u(x, \tau) > g(x, \tau) \text{ per } x < x_f(\tau)$$

per le call. La formulazione rimane valida se esistono diversi limiti liberi o al contrario se non ne è presente neanche uno; i limiti liberi sono definiti nei punti dove  $u(x, \tau)$  incontra la prima volta  $g(x, \tau)$ .



## 2.2 Formulazione del metodo delle differenze finite

Dividiamo il piano  $(x, \tau)$  in una mesh finita regolare nella solita maniera e prendiamo un'approssimazione del metodo delle differenze finite per le equazioni (2.1) di complementarità lineare. Dato che la maggior parte di queste discretizzazioni è una semplice estensione delle formulazioni del metodo delle differenze finite data nel precedente capitolo, diamo qui solo un piccolo resoconto di esse.

Approssimiamo i termini della forma  $\partial u / \partial \tau - \partial^2 u / \partial x^2$  per il metodo delle differenze finite su una mesh regolare con grandezze di step  $\delta \tau$  e  $\delta x$ , e troncando in modo che  $x$  menta nei valori tra  $N^- \delta x$  e  $N^+ \delta x$ ,

$$N^- \delta x \leq x = n \delta x \leq N^+ \delta x,$$

dove  $N^-$  e  $N^+$  sono preferibilmente numeri grandi.

Piuttosto che proseguire separatamente attraverso i vari casi dei metodi esplicito, implicito e di Crank-Nicholson, consideriamo solamente il metodo di Crank-Nicholson in ogni dettaglio e lasciamo le altre formulazioni come esercizio. Quindi prendiamo

$$\frac{\partial u}{\partial \tau}(x, \tau + \delta \tau / 2) = \frac{u_n^{m+1} - u_n^m}{\delta \tau} + O((\delta \tau)^2)$$

e

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2}(x, \tau + \delta \tau / 2) &= \frac{1}{2} \left( \frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2} \right) \\ &\quad + \frac{1}{2} \left( \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} \right) + O((\delta x)^2), \end{aligned}$$

## 2.2. FORMULAZIONE DEL METODO DELLE DIFFERENZE FINITE

---

dove  $u_n^m = u(n\delta x, m\delta\tau)$ . Escludendo i termini  $O((\delta\tau)^2)$  e  $O((\delta x)^2)$ , la disuguaglianza  $\partial u / \partial \tau - \partial^2 u / \partial x^2 \geq 0$  è approssimata da

$$(2.3) \quad \begin{aligned} u_n^{m+1} - \frac{1}{2}\alpha(u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}) \\ \geq u_n^m + \frac{1}{2}\alpha(u_{n+1}^m - 2u_n^m + u_{n-1}^m), \end{aligned}$$

dove  $\alpha$  è dato dalla (1.11).

Scriviamo

$$(2.4) \quad g_n^m = g(n\delta x, m\delta\tau)$$

per la funzione di payoff discretizzata. La condizione  $u(x, \tau) \geq g(x, \tau)$  è approssimata da

$$(2.5) \quad u_n^m \geq g_n^m \quad \text{per } m \geq 1,$$

e le condizioni (2.2) iniziali e di limite implicano che

$$(2.6) \quad u_{N-}^m = g_{N-}^m, \quad u_{N+}^m = g_{N+}^m, \quad u_n^0 = g_n^0.$$

Se definiamo  $Z_n^m$  da

$$(2.7) \quad Z_n^m = (1 - \alpha)u_n^m + \frac{1}{2}\alpha(u_{n+1}^m + u_{n-1}^m),$$

dunque la (2.3) diventa

$$(2.8) \quad (1 + \alpha)u_n^{m+1} - \frac{1}{2}\alpha(u_{n+1}^{m+1} + u_{n-1}^{m+1}) \geq Z_n^m.$$

Osserva che al tempo  $(m + 1)\delta\tau$  possiamo trovare  $Z_n^m$  esplicitamente visto che conosciamo i valori di  $u_n^m$ . La condizione di complementarità lineare

$$\left(\frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2}\right) \cdot (u(x, \tau) - g(x, \tau)) = 0$$

è approssimata da

$$(2.9) \quad ((1 + \alpha)u_n^{m+1} - \frac{1}{2}\alpha(u_{n+1}^{m+1} + u_{n-1}^{m+1}) - Z_n^m)(u_n^{m+1} - g_n^{m+1}) = 0.$$

## 2.3 Il problema della matrice di restrizione

Adesso formuliamo l'approssimazione (2.5)-(2.9) del metodo delle differenze finite come un problema di matrice di restrizione e, nella sezione successiva, descriviamo come risolvere questo problema usando il metodo projected SOR.

Dato  $u^m$  che denota il vettore di valori approssimati al passo  $m\delta\tau$  e  $g^m$  il vettore che rappresenta la restrizione allo stesso tempo:

$$(2.10) \quad u^m = \begin{pmatrix} u_{N^-+1}^m \\ \vdots \\ u_{N^+-1}^m \end{pmatrix}, \quad g^m = \begin{pmatrix} g_{N^-+1}^m \\ \vdots \\ g_{N^+-1}^m \end{pmatrix}.$$

Non includiamo i termini  $u_{N^+}^m$  e  $u_{N^-}^m$  dato che sono determinati esplicitamente dalle condizioni di limite (2.6). Il vettore  $b^m$  è dato da

$$(2.11) \quad b^m = \begin{pmatrix} b_{N^-+1}^m \\ \vdots \\ b_0^m \\ \vdots \\ b_{N^+-1}^m \end{pmatrix} = \begin{pmatrix} Z_{N^-+1}^m \\ \vdots \\ Z_0^m \\ \vdots \\ Z_{N^+-1}^m \end{pmatrix} + \frac{1}{2}\alpha \begin{pmatrix} g_{N^-}^{m+1} \\ 0 \\ \vdots \\ 0 \\ g_{N^+}^{m+1} \end{pmatrix},$$

dove le quantità  $Z_n^m$  sono determinate dalla (2.7) e il vettore più a destra della (2.11) include il risultato delle condizioni di limite a  $n = N^-$  e  $n = N^+$ .

Se introduciamo nuovamente le  $(N^+ - N^- - 1)$ -matrici quadrate, tridiagonali e simmetriche (che abbiamo usato nel precedente capitolo per lo schema

Crank-Nicholson)

$$(2.12) \quad C = \begin{pmatrix} 1 + \alpha & -\frac{1}{2}\alpha & 0 & \cdots & 0 \\ -\frac{1}{2}\alpha & 1 + \alpha & -\frac{1}{2}\alpha & & \vdots \\ 0 & -\frac{1}{2}\alpha & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 + \alpha & -\frac{1}{2}\alpha \\ 0 & \cdots & 0 & -\frac{1}{2}\alpha & 1 + \alpha \end{pmatrix},$$

possiamo riscrivere la nostra approssimazione discreta (2.5)-(2.9) al problema di complementarità lineare (2.1)-(2.2) sotto forma di matrice nel seguente modo

$$(2.13) \quad \begin{aligned} Cu^{m+1} &\geq b^m, \quad u^{m+1} \geq g^{m+1} \\ (u^{m+1} - g^{m+1}) \cdot (Cu^{m+1} - b^m) &= 0. \end{aligned}$$

Prendiamo l'espressione  $a \geq b$ , dove  $a$  e  $b$  sono vettori, a significare che ogni componente di  $a$  è più grande o uguale al corrispondente componente di  $b$ , ossia  $a_n \geq b_n$  per ogni  $n$ .

Nello schema il passare del tempo è immanente: il vettore  $b^m$  contiene l'informazione relativa al passo  $m\delta\tau$  che determina il valore di  $v^{m+1}$  al passo  $(m+1)\delta\tau$ . Ad ogni passo possiamo calcolare  $b^m$  dai valori già noti di  $v^m$ . Possiamo calcolare  $g^m$  per ogni  $m\delta\tau$ , perciò per portare avanti il sistema dobbiamo solamente risolvere il problema (2.13). Questo può essere fatto usando una forma modificata del SOR conosciuta come projected SOR.

## 2.4 Il Projected SOR

Il projected SOR è una variante del metodo SOR descritto nella sezione (1.6.3). Consideriamo l'algoritmo (1.27). Se adattiamo questo ad una for-

mulazione del problema delle differenze finite di Crank-Nicholson otteniamo le equazioni

$$\begin{aligned} y_n^{m+1,k+1} &= \frac{1}{1+\alpha} (b_n^m + \frac{1}{2}\alpha(u_{n-1}^{m+1,k+1} + u_{n+1}^{m+1,k})) \\ u_n^{m+1,k+1} &= u_n^{m+1,k} + \omega(y_n^{m+1,k+1} - u_n^{m+1,k}). \end{aligned}$$

Se queste equazioni vengono iterate fino a che  $u_n^{m+1,k}$  non converge a  $u_n^{m+1}$ , otteniamo la soluzione di  $Cu^{m+1} = b^m$ . Per soddisfare la restrizione  $u^{m+1} \geq g^{m+1}$  modifichiamo semplicemente la seconda di queste equazioni per forzare il tutto al seguente risultato:

$$u_n^{m+1,k+1} = \max(u_n^{m+1,k} + \omega(y_n^{m+1,k+1} - u_n^{m+1,k}), g_n^{m+1}).$$

Da notare che la restrizione viene forzata allo stesso tempo in cui l'iterazione  $u_n^{m+1,k+1}$  viene calcolata; l'effetto della restrizione viene immediatamente risentito nel calcolo di  $u_{n+1}^{m+1,k+1}$ ,  $u_{n+2}^{m+1,k+1}$ , e così via. Perciò l'algoritmo projected SOR si basa sull'iterare (su  $k$ ) le equazioni

$$\begin{aligned} (2.14) \quad y_n^{m+1,k+1} &= \frac{1}{1+\alpha} (b_n^m + \frac{1}{2}\alpha(u_{n-1}^{m+1,k+1} + u_{n+1}^{m+1,k})) \\ u_n^{m+1,k+1} &= \max(u_n^{m+1,k} + \omega(y_n^{m+1,k+1} - u_n^{m+1,k}), g_n^{m+1}) \end{aligned}$$

fino a che la differenza  $\|u^{m+1,k+1} - u^{m+1,k}\|$  è abbastanza piccola da essere considerata insignificante. Infine uno può mettere  $u^{m+1} = u^{m+1,k+1}$ .

Dato che l'algoritmo è iterativo, ogni soluzione generata è consistente visto che non viola alcuna restrizione (guarda sotto la sezione dedicata al punto tecnico). Inoltre tale soluzione ha la proprietà di far scomparire o le n-componenti di  $Cu^{m+1} - b^m$  o  $u_n^{m+1} = g_n^{m+1}$ . Perciò l'algoritmo garantisce che  $u^{m+1} \geq g^{m+1}$  e  $(Cu^{m+1} - b^m) \cdot (u^{m+1} - b^m) = 0$ . La condizione che  $Cu^{m+1} \geq b^m$  nasce come conseguenza della struttura di  $C$  (in modo particolare dal fatto

```
PSOR_solver( u,b,g,Nminus,Nplus,a,omega,eps,loops )
{
    loops = 0;
    a2 = a/2.0;
    u[Nminus] = g[Nminus];
    u[ Nplus] = g[ Nplus];

    do
    {
        error = 0.0;

        for( n=Nminus+1; n<Nplus; ++n )
        {
            y = (b[n]+a2*(u[n-1]+u[n+1]))/(1+a);
            y = max( g[n], u[n]+omega*(y-u[n]) );
            error += (u[n]-y)*(u[n]-y);
            u[n] = y;
        }
        ++loops;
    } while ( error > eps );
    return(loops);
}
```

Figura 2.1: *Pseudo codice per l'algoritmo projected SOR per il problema dell'opzione Americana usando il metodo delle differenze finite di Crank-Nicholson. Abbiamo  $a = \alpha$ ,  $a2 = \frac{1}{2}\alpha$ ,  $Nplus=N^+$ ,  $Nminus=N^-$ ,  $u[n]=u_n^{m+1,k}$  o  $u_n^{m+1,k+1}$ ,  $b[n]=b_n^m$ ,  $g[n]=g_n^{m+1}$ ,  $y=y_n^{m+1,k+1}$ ,  $omega=\omega$  e  $eps$  è l'errore di tolleranza desiderato. La procedura conta il numero di iterazioni richieste e ritorna il valore dentro `loops` così che il programma chiamante possa ottimizzare la distensione del parametro  $omega$ . Compara l'algoritmo con la figura (1.9).*

che è definita positiva). L'algoritmo è illustrato in figura (2.1) e discusso più avanti nella sezione successiva.

### 2.4.1 Punto tecnico: il necessario per il Projected SOR

La (2.13) mostra che per ogni componente  $u_n^{m+1}$  del vettore  $u^{m+1}$  ci sono due e soltanto due possibilità:

$$(1) \quad u_n^{m+1} > g_n^{m+1};$$

$$(2) \quad u_n^{m+1} = g_n^{m+1}.$$

Il caso 1 corrisponde al caso ottimale per l'holding dell'opzione mentre il secondo corrisponde al caso ottimale per l'esercizio dell'opzione. Inoltre dalla condizione di complementarità lineare della (2.13) vediamo che, rispettivamente per i casi 1 e 2, dobbiamo avere:

$$(1) \quad (Cu^{m+1})_n = b_n^m;$$

$$(2) \quad (Cu^{m+1})_n > b_n^m.$$

Nota che qui è richiesta una consistenza interna. Non è molto semplice risolvere  $Cu^{m+1} = b^m$  e poi applicare la condizione di limite estremo che ogni componente  $u_n^{m+1}$  di  $u^{m+1}$  che è più piccola della corrispondente componente  $g_n^{m+1}$  di  $g^{m+1}$  venga sostituita dalla componente più recente. Questa strategia viene frequentemente usata (ed è infatti valida per la discretizzazione del metodo delle differenze finite relativa alla formulazione della complementarità lineare del problema - l'algoritmo projected SOR conduce a questa strategia nel caso del metodo delle differenze esplicite). Il motivo per cui non risulta valida per le formulazioni del problema del metodo delle differenze finite

implicite è semplicemente perchè le componenti  $u_n^{m+1}$  di  $u^{m+1}$  sono tutte in relazione implicita tra di loro e non possiamo modificarne una in modo isolato senza influire sulle altre. Se modifichiamo  $u^{m+1}$  nel modo suggerito sopra, non c'è garanzia che sia  $Cu^{m+1} \geq b^m$  o che  $(Cu^{m+1} - b^m) \cdot (u^{m+1} - g^{m+1}) = 0$ . L'effetto è quello di produrre 'soluzioni' spurie che né falliscono nell'andare incontro alle condizioni di limite libero (cioè producono valori sotto ottimali per l'opzione) e né falliscono nel soddisfare la disuguaglianza Black-Scholes (cioè produce valori per cui sono presenti possibilità arbitrarie). L'algoritmo projected-SOR è costruito in modo da garantire una soluzione interna consistente della (2.13) (cioè una che soddisfi tutte le restrizioni contemporaneamente). Questa soluzione interna consistente è anche l'unica soluzione della (2.13), anche se non lo dimostriamo in questo libro.

## 2.5 L'algoritmo time-stepping

L'algoritmo per il time-stepping da  $u^m$  a  $u^{m+1}$  è una semplice trasformazione della soluzione SOR per le opzioni Europee. In particolare, come sopra, il vettore

$$u^{m+1,k} = (u_{N+1}^{m+1,k}, \dots, u_{N-1}^{m+1,k})$$

denota la  $k$ -esima iterazione dell'algoritmo al passo  $(m+1)$ . (Perciò, al passo  $m+1$  cominciamo con l'ipotesi iniziale  $u^{m+1,0}$  e, applicando l'algoritmo SOR, generiamo  $u^{m+1,k+1}$  da  $u^{m+1,k}$ . Sappiamo che  $u^{m+1,k} \rightarrow u^{m+1}$  per  $k \rightarrow \infty$ .)

- (i) Dato  $u^m$ , prima si forma il vettore  $b^m$  usando le formule (2.7) e (2.11), e calcoliamo il vettore di restrizione  $g^{m+1}$  usando la (2.4) e la (2.10).



(ii) Cominciamo con l'ipotesi iniziale  $u^{m+1,0} = \max(u^m, g^{m+1})$ , che è  $u_n^{m+1,0} = \max(u_n^m, g_n^{m+1})$ .

(iii) Incrementando nell'ordine di n-indici, per primo formiamo la quantità  $y_n^{k+1}$  come segue

$$y_n^{k+1} = \frac{1}{1+\alpha}(b_n^m + \frac{1}{2}\alpha(u_{n-1}^{m+1,k+1} + u_{n+1}^{m+1,k}))$$

(osserva che usiamo gli aggiornamenti  $u_{n-1}^{m+1,k+1}$  piuttosto che i vecchi  $u_{n-1}^{m+1,k}$ ). Poi genera  $u_n^{m+1,k+1}$  usando

$$u_n^{m+1,k+1} = \max(g_n^{m+1}, u_n^{m+1,k} + \omega(y_n^{k+1} - u_n^{m+1,k})),$$

dove  $1 < \omega < 2$  è il parametro di relazione superiore.

(iv) Testa se  $\|u^{m+1,k+1} - u^{m+1,k}\|$  è più piccolo oppure no di un valore di tolleranza  $\epsilon$ , cioè testa

$$\sum_n (u_n^{m+1,k+1} - u_n^{m+1,k})^2 \leq \epsilon^2.$$

Se è così allora procede al passo (v), altrimenti torna al passo (iii) e ripete il processo sostituendo a  $k$  il valore  $k+1$ .

(v) Quando i vettori  $u^{m+1,k}$  convergono alla tolleranza richiesta, mette  $u^{m+1} = u^{m+1,k+1}$ .

(vi) Ritorna al passo (i) fino a che non sono stati completati tutti i passi necessari.

In figura 2.2 riportiamo lo pseudo codice per l'implementazione di questo algoritmo.

```
American( u,dt,dx,Nminus,Nplus,M,omega,eps )
{
    a = dt/(dx*dx);
    a2 = alpha/2.0;
    eps = 1.0e-8;
    omega = 1.0;
    domega = 0.05;
    oldloops = 10000;

    for( n=Nminus; n<=Nplus; ++n )
    {
        x[n] = n*dx;
        u[n] = pay_off( x[n], 0.0 );
    }

    for( m=1; m<=M; ++m )
    {
        tau = m*dt;

        for( n=Nminus+1; n<Nplus; ++n )
        {
            g[n] = payoff( x[n],tau );
            b[n] = u[n] + a2*(u[n+1]-2*u[n]+u[n-1]);
        }
        g[Nminus] = pay_off( x[Nminus],tau );
        g[ Nplus] = pay_off( x[ Nplus],tau );
        u[Nminus] = g[Nminus];
        u[ Nplus] = g[ Nplus];

        PSOR_solver( u,b,g,Nminus,Nplus,a,omega,eps,loops );
        if ( loops > oldloops ) domega *= -1.0;
        omega += domega;
        oldloops = loops;
    }
}
```

Figura 2.2: *Pseudo codice per la risoluzione del problema delle opzioni Americane. Da confrontare con le figure 1.11 e 1.14*

### 2.5.1 Punto tecnico: le opzioni Bermudan

Il metodo projected SOR è una generalizzazione del metodo SOR. Infatti i due metodi sono identici tranne che per il passo

$$u_n^{m+1,k+1} = u_n^{m+1,k} + \omega(y_n^{k+1} - u_n^{m+1,k})$$

che occorre nell'algoritmo SOR e che diventa

$$u_n^{m+1,k+1} = \max(g_n^{m+1}, u_n^{m+1,k} + \omega(y_n^{k+1} - u_n^{m+1,k}))$$

nell'algoritmo projected SOR.

Un vantaggio dell'algoritmo SOR rispetto all'algoritmo LU nella valutazione delle opzioni Europee è che il codice per il computer è identico al codice per la versione Americana eccetto per una singola linea. Perciò, utilizzando il SOR o il projected SOR, risulta di poco conto modificare il codice per un'opzione Europea o Americana per valutare un'opzione che può essere esercitata immediatamente ma solo su date prestabilite. Ci riferiamo a queste opzioni con il termine di opzioni Bermudan. Durante il periodo in cui le opzioni possono essere esercitate immediatamente, la payoff applica la restrizione che il suo valore aumenti. Questo implica che utilizziamo il projected SOR per un determinato periodo. Quando non è permessa l'esercitazione immediata, non viene applicata la restrizione di payoff, ed usiamo dunque il SOR ordinario. La differenza tra i due algoritmi è nell'usare oppure no la funzione di massimo. Per completezza, in figura 2.3 diamo un algoritmo generale di SOR projected che permette ad ogni passo di utilizzare o no la restrizione di esercitazione immediata. Questo può essere utilizzato per valutare le opzioni Bermudan.

## 2.6 Esempi numerici

In figura 2.4 diamo i valori per una put Americana con tasso di interesse  $r = 0.10$ , volatilità  $\sigma = 0.4$  e prezzo di esercizio  $E = 10$ . Il calcolo viene eseguito con  $\alpha = 1$ .

In figura 2.5 mostriamo una soluzione numerica elaborata per un problema di call Americana con  $E = 10$ ,  $r = 0.25$ ,  $\sigma = 0.8$ ,  $D_0 = 0.2$  e un anno di vita, entrambe con il corrispondente valore Europeo. La raffinata separazione del valore dell'opzione dalla funzione di payoff può essere vista al punto  $S_f$ . (Determiniamo la posizione del limite libero a posteriori trovando l'x-nodo  $x_n = n\delta x$  dove  $u_n^m > g_n^m$  ma per cui  $u_{n-1}^m \leq g_{n-1}^m$ . La successiva risoluzione è possibile assumendo le variazioni lineari per  $u(x, m\delta\tau)$  e  $g(x, m\delta\tau)$  tra i punti  $n\delta x$  della griglia; possiamo dunque approssimare la posizione del limite libero dall'intersezione dei segmenti di linea retta che uniscono  $g_{n-1}^m$ ,  $g_n^m$  e  $u_{n-1}^m$ ,  $u_n^m$ .)

## 2.7 La convergenza del metodo

Un'analisi dettagliata della convergenza dell'approssimazione del metodo delle differenze finite alla forma di complementarità lineare per il problema delle opzioni Americane è qualcosa che va oltre l'obiettivo di questo libro. Una prova rigorosa della convergenza implica l'uso di un'analisi astratta funzionale ed è presentato in maniera più semplice dentro la struttura della soluzione degli elementi finiti per la formulazione della disuguaglianza variazionale del problema. (La formulazione della disuguaglianza variazionale può essere derivata ed è equivalente alla formulazione della complementarità lineare). Dato

```

GSOR_solver( u,b,g,Nminus,Nplus,a,omega,eps,early,loops )
{
    loops = 0;
    a2 = a/2.0;
    u[Nminus] = g[Nminus];
    u[ Nplus] = g[ Nplus];

    do
    {
        error = 0.0;

        for( n=Nminus+1; n<Nplus; ++n )
        {
            y = (b[n]+a2*(u[n-1]+u[n+1]))/(1+a);

            if( early_exercise == TRUE)
                y = max( g[n], u[n]+omega*(y-u[n]) );
            else
                y = u[n]+omega*(y-u[n]);

            error += (u[n]-y)*(u[n]-y);
            u[n]=y;
        }
        ++loops;
    } while ( error>eps );
}

```

Figura 2.3: *Pseudo codice per il SOR generalizzato per i problemi delle opzioni Americane, Europee e Bermudan usando il metodo delle differenze finite di Crank-Nicholson. Abbiamo  $a = \alpha$ ,  $a2 = \frac{1}{2}\alpha$ ,  $Nplus = N^+$ ,  $Nminus = N^-$ ,  $u[n] = u_n^{m+1,k}$  o  $u_n^{m+1,k+1}$ ,  $b[n] = b_n^m$ ,  $g[n] = g_n^{m+1}$ ,  $y = y_n^{m+1,k+1}$ ,  $omega = \omega$  e  $eps$  è l'errore di tolleranza desiderato. Il parametro *early* determina se è permesso oppure no l'esercizio immediato. Se è permesso, *early* deve essere settata a *TRUE* dalla procedura; viene quindi usato il projected SOR. Se l'esercizio immediato non è permesso, *early* deve essere settata a *FALSE*; viene quindi usato il SOR. Al richiamo della procedura, per ogni passo, un'opzione Bermudan può essere valutata chiamando la procedura con *early* = *TRUE* quando l'opzione può essere esercitata immediatamente e *early* = *FALSE* quando invece non può. La procedura ritorna il numero di iterazioni richieste in *loops* così che il programma chiamante possa ottimizzare il valore di *omega*.*

## 2.7. LA CONVERGENZA DEL METODO

Asset Price	Payoff Value	3 months		6 months	
		Amer.	Euro.	Amer.	Euro.
0.00	10.0000	10.0000	9.7531	10.0000	9.5123
2.00	8.0000	8.0000	7.7531	8.0000	7.5123
4.00	6.0000	6.0000	5.7531	6.0000	5.5128
6.00	4.0000	4.0000	3.7569	4.0000	3.5583
8.00	2.0000	2.0200	1.9024	2.0951	1.9181
10.00	0.0000	0.6913	0.6694	0.9211	0.8703
12.00	0.0000	0.1711	0.1675	0.3622	0.3477
14.00	0.0000	0.0332	0.0326	0.1320	0.1279
16.00	0.0000	0.0055	0.0054	0.0460	0.0448

Figura 2.4: *Soluzione Crank-Nicholson per una put Americana con  $E = 10$ ,  $r = 0.1$ ,  $\sigma = 0.4$  e con tre e sei mesi di tempo alla scadenza.*

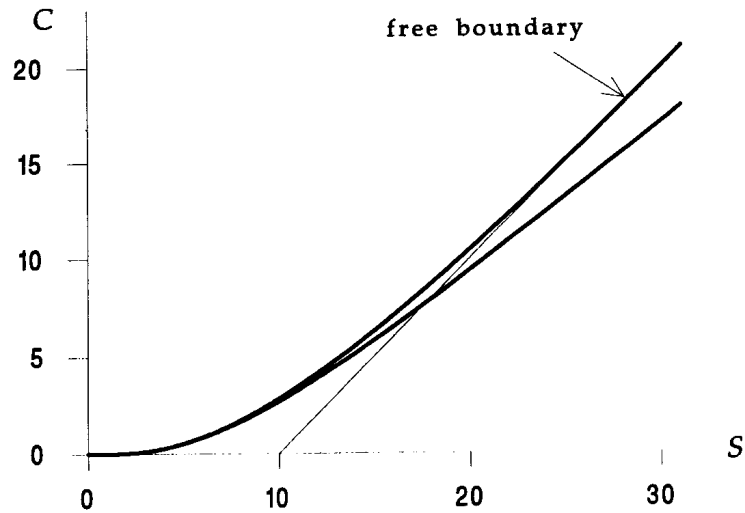


Figura 2.5: *Soluzione numerica calcolata di un problema di call Americana con  $E = 10$ ,  $r = 0.25$ ,  $D_0 = 0.2$ ,  $\sigma = 0.8$  e un anno di tempo alla scadenza. I valori dei parametri sono stati scelti per esagerare la differenza tra l'opzione Americana e la sua controparte Europea.*

## 2.7. LA CONVERGENZA DEL METODO

---

che gli algoritmi numerici per la risoluzione della formulazione della complementarità lineare per mezzo del metodo delle differenze finite e per la risoluzione della formulazione della disuguaglianza variazionale per mezzo degli elementi finiti sono identici, la convergenza dell'algoritmo presentato qui può essere stabilito in questa maniera. Un'analisi dettagliata della convergenza può essere trovata nella letteratura.

# Capitolo 3

## I metodi binomiali

### 3.1 Introduzione

I metodi binomiali per la valutazione delle opzioni e di altri titoli derivati nascono dai modelli di cammino casuale del titolo fondamentale. Essi contano solo indirettamente sull'analisi Black-Sholes per mezzo dell'assunzione della neutralità di rischio; la loro relazione con l'equazione parziale differenziale ed i modelli di disuguaglianza descritti e quello che deriveremo in queste dispense diventa evidente solo quando avremo analizzato che i metodi binomiali sono casi particolari del metodo esplicito delle differenze finite visto nel capitolo 1.

Ci sono due idee principali evidenziate dai metodi binomiali. La prima di questa è che il cammino casuale continuo può essere modellato da un cammino casuale discreto con le seguenti proprietà:

- Il prezzo  $S$  dell'azione cambia solo ai tempi discreti  $\delta t, 2\delta t, 3\delta t, \dots$ , fino a  $M\delta t = T$ , la data di scadenza del titolo derivato. Usiamo  $\delta t$  invece



di  $dt$  per denotare il piccolo ma non-infinitesimale passo di tempo tra i movimenti nel prezzo dell'azione.

- Se il prezzo dell'azione è  $S^m$  al tempo  $m\delta t$  allora al tempo  $(m+1)\delta t$  può assumere solamente uno o due valori possibili,  $uS^m > S^m$  oppure  $dS^m < S^m$ . (In altre parole, durante un singolo passo di tempo, il prezzo del titolo può salire da  $S$  a  $uS$  oppure scendere verso  $dS$ ; vedi figura 3.1). Osserva che questo è equivalente ad assumere che ci siano solo due risultati  $\delta S/S$  possibili ad ogni passo nel tempo,  $u - 1 > 0$  e  $d - 1 < 0$ , e che questi due risultati sono gli stessi per ogni passo nel tempo.
- La probabilità,  $p$ , che  $S$  salga verso  $uS$  è conosciuta (dato che  $(1 - p)$  è la probabilità che  $S$  scenda verso  $dS$ ).

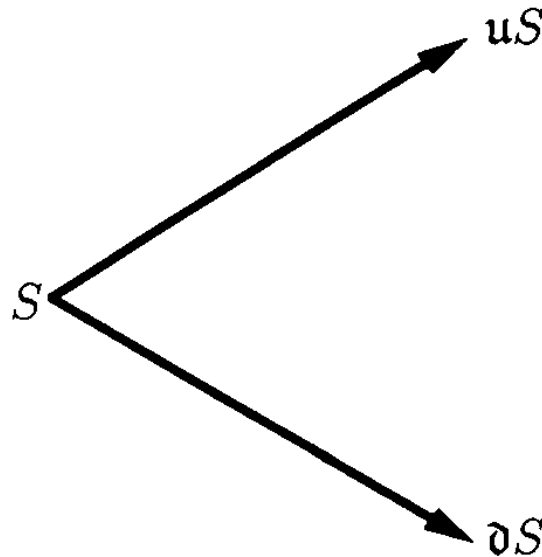


Figura 3.1: *Movimento del prezzo dell'azione nel metodo binomiale*

Come possiamo vedere, scegliamo i parametri  $u$ ,  $d$  e  $p$  in modo tale che le importanti proprietà statistiche del cammino casuale discreto descritto sopra coincidano con quelle di un cammino casuale continuo.

Partendo con un dato valore del prezzo dell'azione (per esempio il prezzo attuale) il tempo di vita rimanente del titolo derivato viene diviso in  $M$  passi di grandezza  $\delta t = (T - t)/M$ . Assumiamo che il prezzo  $S$  dell'azione si muova solo ai passi  $m\delta t$  per  $m = 1, 2, \dots, M$ . Dunque viene creato un albero di tutti i possibili prezzi dell'azione. Questo albero viene costruito partendo dal valore dato  $S$  e vengono generati i due possibili prezzi di azione ( $uS$  e  $dS$ ) al primo passo, poi i tre possibili prezzi di azione ( $u^2S$ ,  $udS = duS$  e  $d^2S$ ) al secondo passo, e così via fino a che non viene raggiunto il tempo di scadenza; vedi figura 3.2. Osserva che un aumento seguito da un ribasso conduce allo stesso valore di azione e che un ribasso seguito da un aumento riconnette l'albero binomiale e dopo  $m$  passi esistono solo  $m + 1$  possibili prezzi di azione.

La seconda congettura è quella di un rischio neutrale a livello mondiale, ossia una dove le preferenze di rischio dell'investitore sono irrilevanti rispetto alla valutazione di sicurezza derivata. Questa congettura può essere fatta ogni volta in cui è possibile proteggere perfettamente un portfolio e renderlo meno rischioso. Sotto queste circostanze possiamo assumere che gli investitori sono risk-neutral (anche se molti non lo sono) e che il risultato è il tasso di interesse di rischio libero. La  $u$  nell'equazione differenziale stocastica  $dS = \sigma S dX + uS dt$  è una misura del tasso di crescita aspettata dell'azione e, come abbiamo visto, non rientra nell'equazione Black-Scholes. In un rischio neutrale a livello

mondiale possiamo sostituire l'equazione differenziale stocastica con

$$(3.1) \quad \frac{dS}{S} = \sigma dX + r dt.$$

Scegliamo i valori  $u$ ,  $d$  e  $p$  nel nostro cammino casuale discreto per riflettere l'importante proprietà statistica del cammino casuale continuo, cioè con un tasso di crescita  $r$  anziché  $u$ .

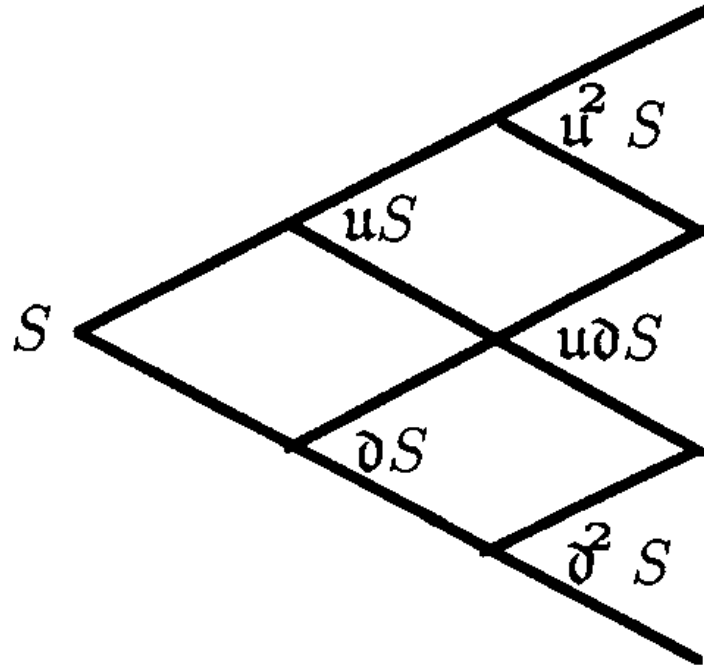


Figura 3.2: *L'albero binomiale per i prezzi possibili del titolo*

Supponendo un rischio neutrale mondiale osserviamo che il valore  $V^m$  del titolo derivato al passo  $m\delta t$  è il valore atteso del titolo al passo  $(m+1)\delta t$  scontato del tasso di interesse  $r$  di rischio libero:

$$(3.2) \quad V^m = \varepsilon[e^{-r\delta t}V^{m+1}];$$

questo è un altro modo per interpretare la formula Black-Scholes.

In un metodo binomiale per prima cosa costruiamo un albero dei possibili valori del prezzo dell'azione e le loro probabilità, dato un prezzo iniziale dell'azione; quindi usiamo questo albero per determinare il possibile prezzo di azione alla scadenza e le probabilità che questo prezzo d'azione si realizzi. I possibili valori del titolo alla scadenza possono essere calcolati ed il titolo può essere valutato. Una conseguenza utile è che possiamo piuttosto facilmente avere a che fare con la possibilità di esercizio immediato e di pagamenti dividendi.

## 3.2 Il cammino casuale discreto

La probabilità  $p$  di un aumento e le ampiezze  $u$  e  $d$  dell'aumento vengo scelte così che il cammino casuale discreto rappresentato dall'albero ed il cammino casuale continuo (3.1) abbiano lo stesso significato e la stessa varianza. In altre parole, dato che il valore del titolo è  $S^m$  al tempo  $m\delta t$ , mettiamo in relazione i valori attesi e le variazioni di  $S^{m+1}$  (dove  $S^{m+1}$  è il valore del titolo al passo  $(m+1)\delta t$ ) sotto il cammino casuale continuo di un rischio neutrale ed il modello binomiale discreto.

Dato che il valore del titolo è  $S^m$  al tempo  $m\delta t$ , il valore atteso di  $S^{m+1}$  sotto il modello di cammino casuale continuo (3.1) è

$$\varepsilon_c[S^{m+1}|S^m] = \int_0^\infty S' p(S^m, m\delta t; S', (m+1)\delta t) dS' = e^{r\delta t} S^m$$

dove  $p(S, t; S', t')$  è la funzione di probabilità di densità

$$(3.3) \quad p(S, t; S', t') = \frac{1}{\sigma S' \sqrt{2\pi(t' - t)}} e^{-(\log(S'/S) - (r - \frac{1}{2}\sigma^2)(t' - t))^2 / 2\sigma^2(t' - t)}$$

### 3.2. IL CAMMINO CASUALE DISCRETO

---

per il cammino casuale di rischio neutrale. Il valore atteso di  $S^{m+1}$ , dato  $S^m$ , sotto il cammino casuale binomiale discreto è

$$\varepsilon_b[S^{m+1}|S^m] = (pu + (1-p)o)S^m.$$

Eguagliando questi due valori attesi otteniamo

$$(3.4) \quad pu + (1-p)d = e^{r\delta t}.$$

La varianza di  $S^{m+1}$ , dato  $S^m$ , è definita come

$$\text{var}[S^{m+1}|S^m] = \varepsilon[(S^{m+1})^2|S^m] - \varepsilon[S^{m+1}|S^m]^2.$$

Sotto il cammino casuale continuo (3.1) abbiamo

$$\begin{aligned} \varepsilon_c[(S^{m+1})^2|S^m] &= \int_0^\infty (S')^2 p(S^m, m\delta t; S', (m+1)\delta t) dS' \\ &= e^{(2r+\sigma^2)\delta t} (S^m)^2, \end{aligned}$$

dove  $p(S, t; S', t')$  è la funzione di densità (3.3). Dunque la varianza sotto il processo continuo (3.1) è

$$\text{var}_c[S^{m+1}|S^m] = e^{2r\delta t} (e^{\sigma^2\delta t} - 1) (S^m)^2.$$

Sotto il processo binomiale discreto abbiamo

$$\varepsilon_b[(S^{m+1})^2|S^m] = (pu^2 + (1-p)d^2)(S^m)^2$$

e usando la (3.4),  $\varepsilon_b[S^{m+1}|S^m] = S^m e^{r\delta t}$ . Abbiamo quindi

$$\text{var}_b[S^{m+1}|S^m] = (pu^2 + (1-p)d^2 - e^{2r\delta t})(S^m)^2.$$

Uguagliando queste due varianze troviamo che

$$(3.5) \quad pu^2 + (1-p)d^2 = e^{(2r+\sigma^2)\delta t}.$$

Le equazioni (3.4) e (3.5) sono due equazioni per i tre valori sconosciuti  $u$ ,  $d$  e  $p$ . Per determinare unicamente questi tre valori sconosciuti abbiamo bisogno di un'altra equazione. Dato che la (3.4) e la (3.5) determinano tutte le importanti proprietà statistiche del cammino casuale discreto (eccetto i requisiti  $u > 0$ ,  $d > 0$  e  $0 \leq p \leq 1$ ), la scelta della terza equazione è in qualche modo arbitraria. Due sono le scelte più utilizzate,

$$(3.6) \quad u = \frac{1}{d},$$

e

$$(3.7) \quad p = \frac{1}{2}.$$

### 3.2.1 Il caso $u = 1/d$

In questo caso otteniamo un metodo binomiale dove  $u$ ,  $d$  e  $p$  sono determinate dalla (3.4), (3.5) e (3.6). Dalla (3.4) e (3.5) troviamo che

$$(3.8) \quad p = \frac{e^{r\delta t} - d}{u - d} = \frac{e^{(2r+\sigma^2)\delta t} - d^2}{u^2 - d^2},$$

così che

$$u + d = \frac{e^{(2r+\sigma^2)\delta t} - d^2}{e^{r\delta t} - d}.$$

Usando la (3.6) per eliminare  $u$  otteniamo l'equazione quadratica

$$d^2 - 2Ad + 1 = 0$$

dove

$$(3.9) \quad A = \frac{1}{2}(e^{-r\delta t} + e^{(r+\sigma^2)\delta t}).$$

Risolvendo per  $d$  e usando la (3.6) per determinare  $u$  e la (3.8) per trovare  $p$ , otteniamo

$$(3.10) \quad d = A - \sqrt{A^2 - 1}, \quad u = A + \sqrt{A^2 - 1}, \quad p = \frac{e^{r\delta t} - d}{u - d}.$$

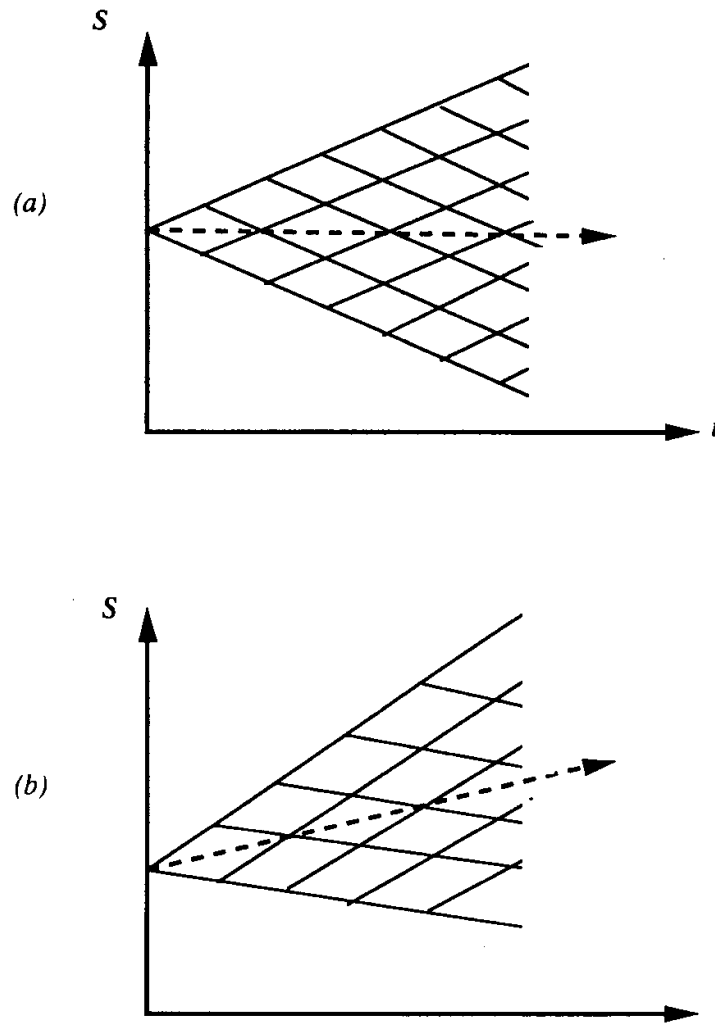


Figura 3.3: Alberi binomiali quando (a)  $u = 1/d$  e (b)  $p = \frac{1}{2}$ .

Se il passo di tempo preso è troppo grande allora  $p$  o  $1 - p$  possono diventare negative; in tal caso il metodo binomiale fallisce.

La scelta (3.6) porta ad un albero in cui il prezzo di partenza del titolo viene riconsiderato ad ogni passo del tempo e che è simmetrico rispetto a questo prezzo; vedi la figura (3.3a). Il prezzo derivato del titolo, causato dal termine  $r dt$  della (3.1), è riflesso nel fatto che la probabilità di un aumento differisce dalla probabilità di un calo;  $p \neq 1 - p$ .

### 3.2.2 Il caso $p = \frac{1}{2}$

In questo caso le costanti  $u$  e  $d$  sono determinate dalla (3.4) e (3.5) e  $p$  è data dalla (3.7). Quindi troviamo che

$$(3.11) \quad u + d = 2e^{r\delta t}, \quad u^2 + d^2 = 2e^{(2r+\sigma^2)\delta t}.$$

Le equazioni sono chiaramente invariate sotto lo scambio di  $u$  e  $d$ , così cerchiamo soluzioni della forma  $u = B + C$ ,  $d = B - C$ , per trovare che

$$(3.12) \quad \begin{aligned} d &= e^{r\delta t}(1 - \sqrt{e^{\sigma^2\delta t} - 1}), \\ u &= e^{r\delta t}(1 + \sqrt{e^{\sigma^2\delta t} - 1}), \\ p &= \frac{1}{2} \end{aligned}$$

In questo caso le probabilità di un aumento e di una diminuzione sono uguali e troviamo che  $ud > 1$  (assumendo  $r > 1$  e  $\delta t$  non troppo grande) e l'albero è orientato nella direzione dello spostamento (vedi figura 3.3b). Se il passo di tempo viene preso troppo grande  $d$  può diventare negativo ed in questo caso il metodo binomiale fallisce.



### 3.2.3 L'albero binomiale

Usando anche la (3.10) o la (3.12) possiamo costruire un albero di possibili prezzi di titolo. Cominciamo al tempo  $t = 0$ . Assumiamo che a questo tempo conosciamo il prezzo del titolo,  $S_0^0$ . Quindi al passo successivo  $\delta t$  ci sono due possibili prezzi di azione,  $S_1^1 = uS_0^0$  e  $S_0^1 = dS_0^0$ . Al seguente passo  $2\delta t$  ci sono tre possibili prezzi di azione  $S_2^2 = u^2S_0^0$ ,  $S_1^2 = udS_0^0 = ouS_0^0$  e  $S_0^2 = d^2S_0^0$ . Al terzo passo  $3\delta t$  ci sono quattro possibili prezzi e così via. Al passo  $m\delta t$  ci sono  $m + 1$  possibili valori del prezzo di azione,

$$S_n^m = d^{m-n}u^nS_0^0, \quad n = 0, 1, \dots, m.$$

(Osserva che qui  $S_n^m$  denota l'ennesimo possibile valore di  $S$  al tempo  $m\delta t$  dove  $d^n$  e  $u^n$  denotano  $d$  e  $u$  elevati all'ennesima potenza). Al passo finale  $M\delta t$  abbiamo  $M + 1$  possibili valori dell'azione. In questo caso  $u = 1/d$  vediamo che

$$S_n^m = u^{2n-m}S_0^0 = d^{m-2n}S_0^0, \quad n = 0, 1, 2, \dots, m$$

così che in ogni momento  $m$  è uguale,  $S_{m/2}^m = S_0^0$ .

Osserva che gli alberi in figura 3.3 ricollegano. Questo ha due conseguenze che sono di interesse immediato. Il primo è che la storia di un particolare prezzo del titolo è perduta dato che ci sono più di un cammino per ogni punto dato. Dunque, in generale, le opzioni dipendenti dal path non possono essere valutate usando questi alberi di riconnessione. La seconda è che il numero totale di punti del reticolo aumentano solo in maniera quadratica con il numero di passi nel tempo. Questo significa che può essere preso un numero elevato di passi nel tempo.

### 3.3 Valutare le opzioni

Assumendo che conosciamo la funzione di payoff per il nostro titolo derivato, e che essa dipende solo dai valori dell'azione al suo scadere, siamo in grado di valutarla allo scadere, cioè al tempo  $M\delta t$ . Se stiamo considerando una put option, per esempio, troviamo che

$$(3.13) \quad V_n^M = \max(E - S_n^M, 0), \quad n = 0, 1, \dots, M,$$

dove  $E$  è il prezzo di esercizio e  $V_n^M$  denota l'ennesimo possibile valore della put al tempo  $M$  e l'ennesimo possibile valore  $S_n^M$  dell'azione. Per una call troviamo che

$$(3.14) \quad V_n^M = \max(S_n^M - E, 0), \quad n = 0, 1, \dots, M,$$

e, in modo simile, per una cash or nothing con prezzo di esercizio  $E$  e payoff

$$V(S, T) = \begin{cases} 0 & S < E, \\ B & S \geq E, \end{cases}$$

abbiamo

$$(3.15) \quad V_n^M = \begin{cases} 0 & S_n^M < E, \\ B & S_n^M \geq E, \end{cases} \quad n = 0, 1, \dots, M.$$

Possiamo trovare il valore atteso dell'azione derivata al passo che precede la scadenza,  $(M-1)\delta t$ , e per il possibile prezzo di azione  $S_n^{M-1}$ ,  $n = 0, 1, \dots, M-1$ , poichè conosciamo che la probabilità che un'azione prezzata al tempo  $S_n^{M-1}$  si muova verso  $S_{n+1}^M$  durante un passo di tempo è  $p$  e la probabilità che essa si muova verso  $S_n^M$  è  $(1-p)$ . Usando l'argomento del rischio neutrale possiamo calcolare il valore dell'azione ad ogni possibile prezzo del titolo per il passo  $(M-1)$ . In maniera simile, questo ci permette di trovare

il valore dell'azione al passo  $(M - 2)$  e così oltre fino ad arrivare al passo 0. Questo ci fornisce il valore della nostra azione al tempo attuale.

### 3.4 Opzioni Europee

Abbiamo dunque  $V_n^m$  che denota il valore dell'opzione al tempo  $m\delta t$  e con prezzo di azione  $S_n^m$  (dove  $0 \leq n \leq m$ ). Calcoliamo il valore atteso dell'opzione al passo  $m\delta t$  dai valori al passo  $(m + 1)\delta t$  e scontiamo questi per ottenere il valore attuale usando il tasso  $r$  di interesse a rischio libero,

$$e^{r\delta t}V_n^m = pV_{n+1}^{m+1} + (1 - p)V_n^{m+1}.$$

Con questo otteniamo

$$(3.16) \quad V_n^m = e^{-r\delta t}(pV_{n+1}^{m+1} + (1 - p)V_n^{m+1}), \quad n = 0, 1, \dots, m.$$

Dato che conosciamo il valore di  $V_n^m$ ,  $n = 0, 1, \dots, M$  dalla funzione di payoff possiamo determinare ricorsivamente i valori di  $V_n^m$  per ogni  $n = 0, 1, \dots, m$  per  $m < M$  per arrivare al valore corrente dell'opzione,  $V_0^0$ .

Non abbiamo bisogno dei prezzi  $S_n^m$  dell'azione durante la valutazione dei prezzi dell'opzione Europea ma solo di  $S_n^M$  quando troviamo  $V_n^M$ . Ad ogni passo possiamo eliminare il vecchio  $S_n^m$  non appena abbiamo calcolato  $S_n^{m+1}$ . Una volta che  $V_n^M$  è stato trovato, possiamo eliminare anche  $S_n^M$ .

Questa osservazione ci conduce ad un algoritmo estremamente efficiente per l'utilizzo della memoria; la memoria richiesta varia linearmente con il numero dei passi di tempo ed il tempo di esecuzione varia quadraticamente con il numero di passi. Uno pseudo codice per l'algoritmo è dato dalla figura 3.4.

```

euro_option( array,S0,u,d,p,r,dt,M )
{
    discount = exp(-r*dt);

    array[0] = S0;
    for( m=1; m<=M; ++m )
    {
        for( n=m; n>0; --n )
            array[n] = u*array[n-1];
        array[0] = d*array[0];
    }

    for( n=0; n<=M; ++n )
        array[n] = pay_off( array[n] );

    for( m=M; m>0; --m )
    {
        for( n=0; n<m; ++n )
        {
            tmp = p*array[n+1] + (1-p)*array[n];
            array[n] = discount*tmp;
        }
    }
}

```

Figura 3.4: *Pseudo codice per il modello binomiale per una semplice opzione Europea.* `array[]` è un array usato per salvare i prezzi di azione durante la costruzione dell'albero (il primo loop for) ed i valori dell'opzione (l'ultimo loop for).  $S_0$  è il valore corrente e le tre lettere  $u$ ,  $d$  e  $p$  possono essere calcolate usando la (3.9) e la (3.10) dalla sezione 3.2.1 oppure la (3.12) dalla sezione 3.2.2.  $r$  è il tasso di interesse,  $dt$  è il passo di tempo e  $M$  il numero di passi di tempo. La procedura costruisce un albero di possibili prezzi del titolo, poi trova i valori delle opzioni al tempo di scadenza usando la funzione `pay-off`. Infine calcola gli attuali valori dei valori attesi del prezzo di opzione, sotto un cammino casuale con rischio neutrale, procedendo a ritroso dal tempo di scadenza fino al valore attuale. Tale valore viene riportato in `array[0]`.

$T$	Black-Scholes	Binomial Method ( $u = 1/d$ )				
		$M = 16$	32	64	128	256
0.25	4.8511	4.8511	4.8511	4.8511	4.8511	4.8511
0.50	4.7048	4.7046	4.7047	4.7047	4.7048	4.7048
0.75	4.5636	4.5626	4.5632	4.5634	4.5634	4.5636
1.00	4.4304	4.4292	4.4300	4.4300	4.4300	4.4304

Figura 3.5: Confronto tra i valori del metodo binomiale (con  $u = 1/d$ ) e dell'equazione di Black-Scholes per una put Europea con  $E = 10$ ,  $S = 5$ ,  $r = 0,06$  e  $\sigma = 0,3$ . Il tempo di scadenza  $T$  è misurato in anni.

$T$	Black-Scholes	Binomial Method ( $p = \frac{1}{2}$ )				
		$M = 16$	32	64	128	256
0.25	4.8511	4.8511	4.8511	4.8511	4.8511	4.8511
0.50	4.7048	4.7046	4.7047	4.7047	4.7048	4.7048
0.75	4.5636	4.5625	4.5632	4.5634	4.5635	4.5635
1.00	4.4304	4.4293	4.4300	4.4300	4.4302	4.4303

Figura 3.6: Confronto tra i valori del metodo binomiale (con  $p = \frac{1}{2}$ ) e l'equazione Black-Scholes per una put Europea con  $E = 10$ ,  $S = 5$ ,  $r = 0,12$  e  $\sigma = 0,5$ . Il tempo di scadenza  $T$  è misurato in anni.

In figura 3.5 confrontiamo i valori di una put Europea calcolati usando il metodo binomiale, con  $u = 1/d$  con  $M = 16, 32, 64, 128$  e  $256$  passi di tempo, con il valore Black-Scholes.

In figura 3.6 confrontiamo i valori di una put Europea calcolati usando il metodo binomiale, con  $p = \frac{1}{2}$ , con  $M = 16, 32, 64, 128$  e  $256$  passi di tempo, con il valore Black-Scholes.

### 3.5 Opzioni Americane

Possiamo facilmente integrare la possibilità di esercizio immediato di un'opzione all'interno del modello binomiale. Come prima, dividiamo il tempo di scadenza in  $M$  passi di tempo uguali  $\delta t = T/M$  e costruiamo il nostro albero dei possibili prezzi di azioni,

$$S_n^m, \quad n = 0, 1, \dots, m,$$

dove  $S_{0,0}$  è il valore corrente e  $S_n^m$  è un possibile valore al tempo  $m\delta t$ . Come nella sezione precedente possiamo calcolare  $u$ ,  $d$  e  $p$  anche usando la (3.9) e la (3.10) o usando la (3.12). Al tempo  $M\delta t$  possiamo calcolare i valori possibili dell'opzione dalla funzione di payoff, per esempio, di una put usando la (3.13), per le call usando la (3.14) e per una cash-or-nothing usando la (3.15).

Considera la situazione al tempo  $m$  e al prezzo di azione  $S_n^m$ .

L'opzione può essere esercitata prima che scada per produrre un profitto determinato dalla funzione di payoff; per le put, call e cash-or-nothing, queste sono rispettivamente:

$$\max(E - S_n^m, 0), \quad (S_n^m - E, 0), \quad \begin{cases} 0 & S_n^m < E, \\ B & S_n^m \geq E \end{cases}.$$

```

amer_option( s,v,S0,u,d,p,r,dt,M )
{
    discount = exp(-r*dt);

    s[0][0] = S0;
    for( m=1; m<=M; ++m )
    {
        for( n=m+1; n>0; --n )
            s[m][n] = u*s[m-1][n-1];
        s[m][0] = d*s[m-1][0];
    }

    for( n=0; n<=M; ++n )
        v[M][n] = pay_off(s[M][n]);

    for( m=M; m>0; --m )
    {
        for( n=0; n<=m; ++n )
        {
            hold = (1-p)*v[m+1][n] + p*v[m+1][n+1];
            hold *= discount;
            v[m][n] = max( hold, pay_off( s[m][n] ) );
        }
    }
}

```

Figura 3.7: *Pseudo codice per il modello binomiale per un'opzione Americana.* Gli array  $s[][]$  e  $v[][]$  sono usati per salvare gli alberi dei valori dell'azione ed i valori dell'opzione, rispettivamente.  $S0$  è il valore corrente dell'azione e i valori  $u$ ,  $d$  e  $p$  possono essere calcolati usando la (3.10) e la (3.11) oppure la (3.12).  $r$  è il tasso di interesse,  $dt$  è il passo di tempo e  $M$  il numero di passi. La procedura prima costruisce e salva l'albero dei prezzi di azione, poi calcola la payoff alla scadenza usando la funzione `pay_off` ed infine valuta l'opzione prendendo il massimo del valore atteso e la funzione di payoff per un esercizio immediato ad ogni passo di tempo e prezzo di azione.

Se l'opzione viene conservata, il suo valore  $V_n^m$  è, come nel caso dell'opzione Europea,

$$V_n^m = e^{-r\delta t}(pV_{n+1}^{m+1} + (1-p)V_n^{m+1}).$$

Il valore dell'opzione è il massimo tra queste due possibilità, o in altre parole

$$(3.17) \quad V_n^m = \max(\text{pay\_off}(S_n^m), e^{-r\delta t}(pV_{n+1}^{m+1} + (1-p)V_n^{m+1})).$$

Per esempio, per una put option abbiamo

$$V_n^m = \max(\max(E - S_n^m, 0), e^{-r\delta t}(pV_{n+1}^{m+1} + (1-p)V_n^{m+1}))$$

e per una call

$$V_n^m = \max(\max(S_n^m - E, 0), e^{-r\delta t}(pV_{n+1}^{m+1} + (1-p)V_n^{m+1})).$$

L'implementazione dello schema è semplice come lo è per le opzioni Europee. Per primo viene costruito l'albero  $S_n^m$  dei valori dell'azione e viene salvato, diversamente dal caso Europeo. Poi valutiamo  $V_n^m$  dalla funzione di payoff ed analizziamo l'albero dall'alto verso il basso per trovare il valore dell'opzione. L'unica complicazione è che è necessario testarlo per decidere quale tra i due possibili valori (esercizio immediato o conservazione dell'opzione) è il più grande. Questa è la ragione per cui salviamo i valori  $S_n^m$  visto che ci permettono di implementare il test in maniera efficiente. Uno pseudo codice che implementa questo algoritmo è mostrato in figura 3.7. La richiesta di salvataggio di  $S_n^m$  implica che la memoria richiesta abbia una variazione quadratica con il numero di passi di tempo, come fa anche il tempo di esecuzione.

In figura 3.8 e 3.9 mostriamo delle approssimazioni binomiali per una put Americana con prezzo di esercizio 10, prezzo di azione corrente 9, tasso di



interesse  $r = 0,06$ , volatilità  $\sigma = 0,3$ , per  $M = 16$ ,  $M = 32$ ,  $M = 64$ ,  $M = 128$  e  $M = 256$  passi di tempo. In figura 3.8 abbiamo usato la (3.9) e la (3.10) per calcolare  $u$ ,  $d$  e  $p$  mentre in figura 3.9 abbiamo usato la (3.12).

### 3.6 Redditi dividendi

Il metodo binomiale può fornire un reddito dividendo costante  $D_0$  pagato nell'opzione. L'effettivo tasso di crescita di rischio libero per l'azione diventa  $r - D_0$  piuttosto che  $r$ ,

$$\frac{dS}{S} = (r - D_0)dt + \sigma dX.$$

Quindi sostituiamo  $r$  con  $r - D_0$  nella fase di costruzione dell'albero quando calcoliamo  $u$ ,  $d$  e  $p$ . Perciò la (3.9) e la (3.10), per il caso  $u = 1/d$ , diventano

$$(3.18) \quad A = \frac{1}{2}(e^{-(r-D_0)\delta t} + e^{(r-D_0+\sigma^2)\delta t})$$

e

$$(3.19) \quad d = A - \sqrt{A^2 - 1}, \quad u = A + \sqrt{A^2 - 1}, \quad p = \frac{e^{(r-D_0)\delta t} - d}{u - d},$$

e la (3.12) per il caso  $p = \frac{1}{2}$ , diventa

$$(3.20) \quad \begin{aligned} d &= e^{(r-D_0)\delta t}(1 - \sqrt{e^{\sigma^2\delta t} - 1}), \\ u &= e^{(r-D_0)\delta t}(1 + \sqrt{e^{\sigma^2\delta t} - 1}), \\ p &= \frac{1}{2}. \end{aligned}$$

Il valore attuale di un'azione è ancora determinato dalla riduzione di prezzo tramite l'utilizzo del tasso di interesse a rischio libero  $r$  così che la (3.16) per le opzioni Europee e la (3.17) per le opzioni Americane sono ancora

$T$	Binomial Method ( $u = 1/d$ )				
	$M = 16$	32	64	128	256
0.25	1.1316	1.1240	1.1261	1.1253	1.1260
0.50	1.2509	1.2598	1.2539	1.2553	1.2546
0.75	1.3579	1.3568	1.3569	1.3555	1.3547
1.00	1.4444	1.4332	1.4384	1.4342	1.4349

Figura 3.8: Confronto dei valori binomiali (con  $u = 1/d$ ) per una put Americana con  $E = 10$ ,  $S = 9$ ,  $r = 0,06$  e  $\sigma = 0,3$ . Il tempo alla scadenza  $T$  è misurato in anni.

$T$	Binomial Method ( $p = \frac{1}{2}$ )				
	$M = 16$	32	64	128	256
0.25	1.1311	1.1249	1.1266	1.1258	1.1260
0.50	1.2526	1.2590	1.2553	1.2559	1.2553
0.75	1.3609	1.3530	1.3573	1.3540	1.3541
1.00	1.4473	1.4358	1.4369	1.4358	1.4354

Figura 3.9: Confronto dei valori binomiali (con  $p = \frac{1}{2}$ ) per una put Americana con  $E = 10$ ,  $S = 9$ ,  $r = 0,06$  e  $\sigma = 0,3$ . Il tempo alla scadenza  $T$  è misurato in anni.

$T$	Black-Scholes	Binomial Method ( $u = 1/d$ )				
		$M = 16$	32	64	128	256
0.25	2.1116	2.1138	2.1128	2.1111	2.1113	2.1117
0.50	2.2820	2.2874	2.2856	2.2839	2.2818	2.2819
0.75	2.4374	2.4479	2.4316	2.4381	2.4361	2.4380
1.00	2.5752	2.5795	2.5806	2.5789	2.5771	2.5752

Figura 3.10: Confronto tra i valori del metodo binomiale (con  $u = u/d$ ) e quelli di Black-Scholes per una call Europea con  $E = 10$ ,  $S = 12$ ,  $r = 0,06$ ,  $D_0 = 0,04$  e  $\sigma = 0,3$ . Il tempo alla scadenza  $T$  è misurato in anni.

$T$	Black-Scholes	Binomial Method ( $p = \frac{1}{2}$ )				
		$M = 16$	32	64	128	256
0.25	2.1116	2.1132	2.1123	2.1121	2.1119	2.1118
0.50	2.2820	2.2906	2.2839	2.2824	2.2831	2.2825
0.75	2.4374	2.4436	2.4406	2.4407	2.4390	2.4368
1.00	2.5752	2.5667	2.5840	2.5745	2.5741	2.5756

Figura 3.11: Confronto tra i valori del metodo binomiale (con  $p = \frac{1}{2}$ ) e quelli di Black-Scholes per una call Europea con  $E = 10$ ,  $S = 12$ ,  $r = 0,12$ ,  $D_0 = 0,04$  e  $\sigma = 0,5$ . Il tempo alla scadenza  $T$  è misurato in anni.

valide quando calcoliamo il valore attuale atteso dell'opzione. Quindi l'unico effetto dei redditi dividendi continui è quello di modificare la probabilità  $p$  e le ampiezze di salto  $u$  e  $d$ . Perciò possiamo usare gli stessi codici dati sopra per valutare tali opzioni semplicemente modificando i parametri  $u$ ,  $d$  e  $p$  in accordo con la (3.18), la (3.19) e la (3.20). Questo lo applichiamo sia per le opzioni Europee che per quelle Americane.

Nelle figure 3.10 e 3.11 confrontiamo i valori dell'esatta Black-Scholes con le approssimazioni binomiali per una call Europea con prezzo di esercizio 10, prezzo di azione corrente 12, tasso di interesse  $r = 0,06$ , reddito dividendo  $D_0 = 0,03$  e volatilità  $\sigma = 0,03$ , per  $M = 16$ ,  $M = 32$ ,  $M = 64$ ,  $M = 128$  e  $M = 256$  passi di tempo. In figura 3.10 abbiamo usato la (3.18) e la (3.19) per calcolare  $u$ ,  $d$  e  $p$  mentre in figura 3.11 abbiamo usato la (3.20).