

Computational Finance

Paolo Foschi

Department of Statistics
University of Bologna
paolo.foschi2@unibo.it

November 2013



Outline



Table of Contents I



Disclaimer



Part I

Introduzione a Matlab

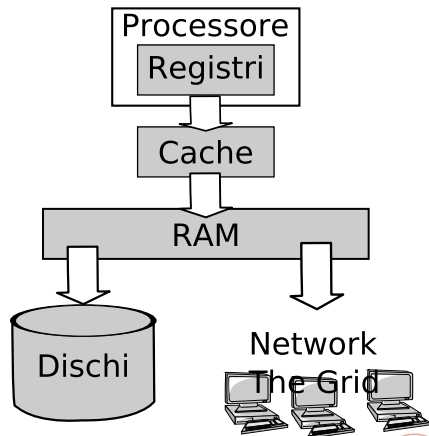
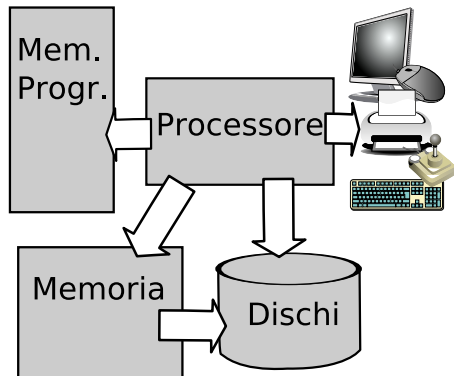


Outline

- 1 Problemi, Algoritmi, Complessità ed Errori
- 2 L'ambiente Matlab
- 3 Programmazione e Matlab
- 4 Grafica in Matlab



Struttura degli elaboratori



Algoritmo

- **Algoritmo**: descrizione dettagliata e chiara dei passi necessari per trasformare un dato insieme di “input” in particolare insieme di “output”.
- Etimologia: Abu Abdullah Muhammad bin Musa “al-Khwarizmi”, Astronomo e matematico, nato a Bagdad nel 780AC. Sviluppo’ metodi per l’aritmetica nel “nuovo” sistema Indo-Arabico
- **Linguaggio**: Permette di rendere non ambiguo un algoritmo.
- **Sintassi**: insieme di regole che definiscono un particolare linguaggio.
- **Programma**: implementazione di un particolare algoritmo in un particolare linguaggio

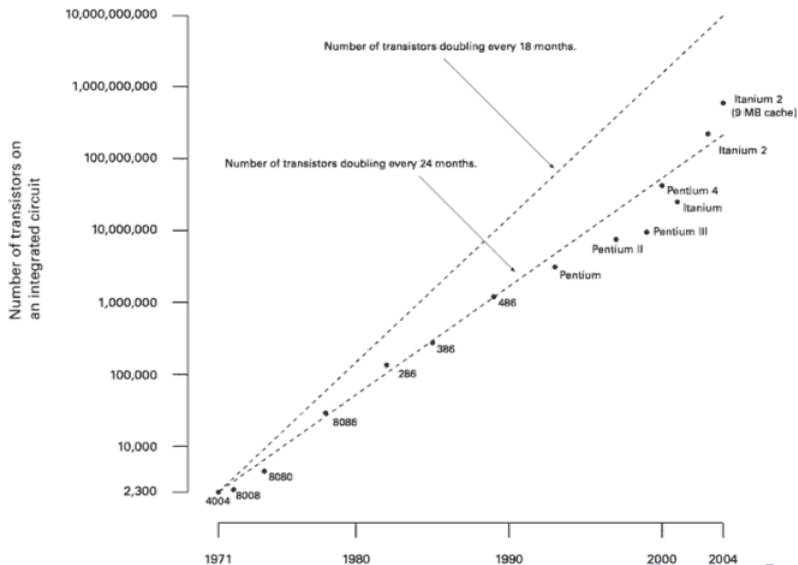


Complessità

- Complessità: Dimensione del problema \rightarrow tempo di esecuzione, (o spazio di memoria utilizzato)
- La complessità viene espressa con valori asintotici:
 - Ordinamento: complessità $O(n \log(n))$
 - Risoluzione sistema di eq. lineari: $\leq O(n^3)$
 - Calcolo del prezzo di un'opzione in un modello binomiale: $O(n^2)$
 - Commesso Viaggiatore: $O(2^n)$: non polinomiale
- Complessità non polinomiali:
 - $n = 20$: $2^{20} \simeq 10^6 \text{ sec} \simeq 300 \text{ ore} \simeq 12 \text{ giorni}$
 - $n = 21$: $2^{21} \simeq 24 \text{ giorni}$
 - $n = 22$: $2^{22} \simeq 48 \text{ giorni}$
 - $n = 25$: $2^{23} \simeq 388 \text{ giorni} \simeq 1 \text{ anno}$
 - E se utilizzassi/aspettassi un computer piú potente.
- Il problema di sapere se un dato programma termina o meno è non decidibile:
 - se termina lo si può sapere in un tempo finito,



Legge di Moore



Errori di arrotondamento

- Ogni numero reale viene rappresentato come una sequenza di cifre di lunghezza fissa.
- Floating point = numero a virgola fissa + esponente (notazione esponenziale).
- Esempi:

| | | | | | | | | |
|-----------|--------------------------|---|---|---|---|---|---|---|
| 12.3 | 0.123×10^2 | + | 1 | 2 | 3 | 0 | + | 2 |
| -0.005279 | -0.5279×10^{-4} | - | 5 | 2 | 7 | 9 | - | 4 |

- $1 + 0.0005 = 1.0005 \simeq 1$, errore del 0.05%

| | | | | | | | | |
|--------|----------------------|---|---|---|---|---|---|---|
| 1.0005 | $.10005 \times 10^1$ | + | 1 | 0 | 0 | 0 | + | 1 |
|--------|----------------------|---|---|---|---|---|---|---|



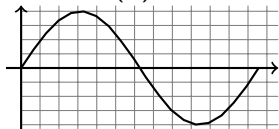
Errori di arrotondamento

- Problema: calcolare la somma $1 + 0.0004 - .9999 (= .5)$
Algoritmo: calcolare la somma $1 + 0.0004$, sommare al risultato $-.9999$
Risultato: $.0001$ Errore: 80%
- Double precision: mantissa 52 cifre binarie (16 decimali) + 9 cifre binarie per l'esponente.
- Può essere un problema se si devono sommare migliaia di numeri.



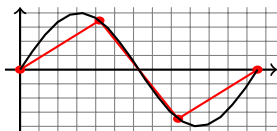
Discretizzazione: errori di approssimazione

- Calcolare la lunghezza della curva $\sin(x)$ da $x = 0$ a $x = 2\pi$

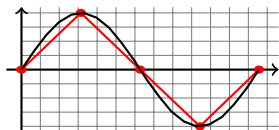


- Approssimazione con una sequenza di corde.

3 corde \rightarrow 4.742



4 corde \rightarrow 5.086



5 corde \rightarrow 5.064



10 corde \rightarrow 5.215

100 corde \rightarrow 5.2698

1000 corde \rightarrow 5.2704



Algoritmi randomizzati

- Molti algoritmi euristici utilizzano numeri casuali per “sondare” lo spazio delle possibili soluzioni
- Esempi: monte carlo, algoritmi genetici, simulated annealing, ant colonies, etc...
- A parità di input, la soluzione trovata è diversa ad ogni esecuzione del programma
- In realtà i computer non generano numeri casuali, ma pseudo casuali:
 - sequenze deterministiche che sembrano casuali (hanno quasi gli stessi momenti)
 - il “seed” è il valore iniziale di una sequenza di numeri (pseudo)-casuali



Outline

- 1 Problemi, Algoritmi, Complessità ed Errori
- 2 L'ambiente Matlab**
- 3 Programmazione e Matlab
- 4 Grafica in Matlab



Cos'è Matlab

Matlab è un ambiente Rapid Application Development (RAD) per il calcolo scientifico, la grafica e visualizzazione.

Matlab fornisce strumenti per:

- Calcolo numerico
- Analisi dei dati e visualizzazione
- Ingegneria e grafica scientifica
- Modellazione e simulazione di processi
- Programmazione e sviluppo di applicazioni
- Altri Toolbox



Perchè Matlab in questo corso

- Pro:
 - ha un linguaggio di programmazione
 - è interpretato ed ha un ambiente per il testing
 - $y = Ax$ si calcola o si risolve scrivendo al più 5 caratteri
 - se dovutamente utilizzato è molto efficiente
 - la finanza computazionale non è altro che calcolo scientifico
- Contro:
 - strutture dati
 - in molte banche italiane si usa Visual Basic/Excel
 - è costoso (per un privato)
- Alternative open source: Scilab, Octave, R, Python, C++, C#
- Per la Finanza: R-metrics e Quantlib



Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- Input/Output
- Controllo dell'esecuzione
- Espressioni
- Cicli
- Funzioni
- Array, Vettori e Matrici

4 Grafica in Matlab



Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- Input/Output
- Controllo dell'esecuzione
- Espressioni
- Cicli
- Funzioni
- Array, Vettori e Matrici

4 Grafica in Matlab



Variabili

- Una variabile associa ad particolare nome un valore.
- Ad ogni variabile è associata una zona di memoria che contiene il valore della variabile.
- Esistono diversi **tipi** di valori: Numeri Interi, Numeri Reali, Numeri Complessi, Vettori di \mathbb{R}^n , sequenze di caratteri, funzioni da \mathbb{R}^n a \mathbb{R}^n , etc...
- **Dichiarare** o **Definire** una variabile equivale a riservare una particolare zona di memoria per il valore che la variabile conterrà.
- **Assegnare** un valore ad una variabile consiste nel porre tale valore nella corrispondente di memoria.
- In “Pascal”, “C/C++” e “Java” le variabili devono essere dichiarate prima di essere utilizzate e non possono cambiare tipo.
- In “Basic”, “Fortran”, “Matlab” ed “R” la dichiarazione è implicita nel primo assegnamento ed il tipo della variabile può variare in base al dato assegnato.



Istruzioni di Assegnamento

- Sintassi: `<nome variabile> = <espressione>`
- Semantica: metti nella variabile `<nome variabile>` il risultato di `<espressione>`
- Esempi:

```
script0.m
x = 3;
y = x + 2;
y = 2*y;
z = 1/(2*pi)*exp(-x^2/2);
```

- l-values: Variabili a SX dell'assegnamento
- r-values: Variabili a DX dell'=
 - `x=expr` è un'azione, non è un'equazione.
 - Gli r-values devono essere già stati inizializzati (devono esistere).



Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- **scripts**
- Input/Output
- Controllo dell'esecuzione
- Espressioni
- Cicli
- Funzioni
- Array, Vettori e Matrici

4 Grafica in Matlab



Scripts

- Uno “script” è un file `.m` contenente una sequenza di comandi `matlab`
- Equivale a digitare gli stessi comandi nella “command window”
- Esempio:

```
script1.m
a=1; b=-3; c=2;
d  = b^2 - 4*a*c;
x1 = (-b-sqrt(d))/(2*a);
x2 = (-b+sqrt(d))/(2*a);
```

- Viene eseguito digitando il nome del file nella command window
- In `matlab` le istruzioni possono essere separate da “;”, “,” o da capo riga



Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- **Input/Output**
- Controllo dell'esecuzione
- Espressioni
- Cicli
- Funzioni
- Array, Vettori e Matrici

4 Grafica in Matlab



Input

- Sintassi: `<var.> = input(<stringa messaggio>)`
- Semantica: visualizza il messaggio, attendi che l'utente inserisca un valore numerico e ritorna tale valore.
- Esempio: `c = input('Inserisci la costante');`
- Esempio script:

```
script2.m
% Risolvi eq. di 2^ grado
a = input('Inserire a: ');
b = input('Inserire b: ');
c = input('Inserire c: ');
d = b^2 - 4*a*c;
x1 = (-b-sqrt(d))/(2*a);
x2 = (-b+sqrt(d))/(2*a);
```

- Input grafico: `x,y = ginput(1);`



Output

- Sintassi: `disp(<expr>)`
- Semantica: visualizza il risultato di `<expr>` nella command window.
- Sintassi: `fprintf(<fmt>, <expr1>, <expr2>, ..)`
- Semantica: visualizza il risultato delle espressioni `<expr1>`, `<expr2>`, etc... secondo quanto prescritto dalla stringa `<fmt>`
- Esempio:

```
script3.m
% Risolvi eq. di 2^ grado
a = input('Inserire a: ');
b = input('Inserire b: ');
c = input('Inserire c: ');
d = b^2 - 4*a*c;
x1 = (-b-sqrt(d))/(2*a);
x2 = (-b+sqrt(d))/(2*a);
fprintf('Le radici sono: %g, %g\n', x1, x2);
```

Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- Input/Output
- **Controllo dell'esecuzione**
- Espressioni
- Cicli
- Funzioni
- Array, Vettori e Matrici

4 Grafica in Matlab



If-then-else

- Sintassi:

```
ifelse.m  
  
if <expr1>  
    <statement 1>  
else  
    <statement 2>  
end
```

- Semantica: se $\langle \text{expr1} \rangle$ è vera esegui $\langle \text{statement 1} \rangle$ altrimenti esegui $\langle \text{statement 2} \rangle$.
- Esempio:

```
ifexample.m  
  
if d<0  
    disp('Radici complesse');  
else  
    x1 = (-d-sqrt(d))/(2*a);  
    x2 = (-d+sqrt(d))/(2*a);  
end
```

Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- Input/Output
- Controllo dell'esecuzione
- **Espressioni**
- Cicli
- Funzioni
- Array, Vettori e Matrici

4 Grafica in Matlab



Operatori e Funzioni Built-in

● Funzioni Built-in

- Funzioni trigonometriche: `sin`, `cos`, `tan`, `asin`, `acos`, `atan`.
- Esponenziali: `exp`, `log`, `log2`, `log10`.
- Arrotondamento: `floor`, `ceil`, `round`, `fix`, `mod`, `rem`
- `max`, `min`, `abs`, `sign`
- Numeri casuali: `rand`, `randn`

● Operatori

- Operatori aritmetici: `+`, `-`, `/`, `\`, `*`
- Operatori relazionali: `==`, `<`, `>`, `<=`, `>=`, `~=`.
- Operatori logici: `~` (not), `&` (and), `|` (or).

| and | T | F |
|-----|---|---|
| T | T | F |
| F | F | F |

| or | T | F |
|----|---|---|
| T | T | T |
| F | F | F |

- Algebra: $\sim(a \ \& \ b) \equiv (\sim a) \ | \ (\sim b)$



Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- Input/Output
- Controllo dell'esecuzione
- Espressioni
- **Cicli**
- Funzioni
- Array, Vettori e Matrici

4 Grafica in Matlab



Cicli for

- Per ripetere qualcosa N volte:

```
forend.m
for i=1:N
    <statement>
end
```

- Semantica:

```
for2.m
i=1, <statement>
i=2, <statement>
...
i=N, <statement>
```

- Il contatore i è una variabile a tutti gli effetti



Cicli for

- Esempio: Somma dei primi $N=100$ numeri:

```
for3.m
N = 100;  somma = 0;
for j=1:N
    somma = somma+j;
end
```

- Ciclo for tipico:

```
for4.m
for i = first:by:last
    <statement>
end
```

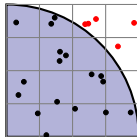
- Assegna ad i in sequenza $first$, $first+by$, $first+2*by$, etc.. fino al raggiungimento del valore $last$.
- Esempio:

● $i=1:4:15$ equivale alla sequenza $[1, 5, 9, 13]$



Esempio: calcolo di π

- $x, y \sim \text{unif}(0, 1)$ e indipendenti.
- $P[x^2 + y^2 \leq 1] = \pi/4$



```
pigreco.m
dentro=0;
for i=1:N
    x = rand;    y = rand;
    if (x^2+y^2<=1)
        dentro=dentro+1;
    end
end
my_pi = dentro/N*4;
```



Il ciclo While

- Per ripetere finché una tale condizione rimane vera

```
while1.m  
  
<init>  
while <expr>  
    <statement>  
end
```

- Semantica:

```
while2.m  
  
Esegui le istruzioni <init>  
Valuta <expr>, se falsa vai ad end  
esegui <statement>  
valuta <expr>, se falsa vai ad end  
esegui <statement>  
...
```

- Il `while` cicla per vero

Cicli for e while

- Ogni ciclo `for` può essere espresso con un ciclo `while`

```
pigreco3.m
dentro=0;
i=1;
while (i<=N)
    x = rand;    y = rand;
    if (x^2+y^2<=1)
        dentro=dentro+1;
    end
    i=i+1;
end
pi_greco = dentro/N*4;
```



Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- Input/Output
- Controllo dell'esecuzione
- Espressioni
- Cicli
- **Funzioni**
- Array, Vettori e Matrici

4 Grafica in Matlab



Analisi top-down

- Spesso conviene un problema complesso in sottoproblemi piú semplici
- Analisi top-down: da un'analisi grossolana si arriva fino ai minimi dettagli
- Problema: ricerca in elenco telefonico.
 - cerca città:
 - scegli una pagina - leggi città - confronta due parole
 - cerca cognome:
 - scegli pagina e riga - leggi cognome - confronta due parole
 - cerca nome:
 - scegli pagina e riga - leggi nome - confronta due parole



Funzioni

- Alcuni sottoproblemi possono presentarsi piú volte o essere parte dell'analisi di altri problemi
- Ogni sottoproblema è risolto da un sottoprogramma: funzione
- Come i programmi i le funzioni associano a particolari input i dovuti output
- Le funzioni devono essere progettate e utilizzate come scatole nere, di cui si conosce il funzionamento e le caratteristiche (specifiche) ma non il contenuto (implementazione).
- Interagiscono con l'esterno solo tramite le variabili di input e quelle di output.
- In questo modo si potrà cambiare l'implementazione di una funzione senza alterare il comportamento del programma che le utilizza.
- Librerie: insieme di funzioni di utilizzo frequente



Funzioni

- Sintassi:

```
function <output> = <name> ( <in1>, <in2>, ...)  
% Post: proprieta'  variabili di output  
% Pre:  proprieta'  che devono avere le  
%      variabili di input  
<statements>
```

fun1.m

- Ogni funzione è contenuta in un file .m dello stesso nome
- Esempio:

```
function y = mynormcdf(x)  
% y = phi(x)  
%      calcola la funzione cumulativa della  
%      distribuzione normale standard  
  
y = erfc( -x/sqrt(2))/2;
```

mynormcdf.m

Exercise (Prezzo Zero Coupon Bond)

Scrivere una funzione 'ZCB' che calcoli il prezzo P al tempo t di uno Zero Coupon Bond con scadenza T e tasso composto R :

$$P = \frac{100}{(1 + R)^{T-t}}$$



Exercise (Prezzo Zero Coupon Bond)

Scrivere una funzione 'ZCB' che calcoli il prezzo P al tempo t di uno Zero Coupon Bond con scadenza T e tasso composto R :

$$P = \frac{100}{(1 + R)^{T-t}}$$

zcb.m

```
function p = zcb(t,T,r)
% p = zcb(t,T,r)
%
% Compute the value at t of a ZCB with maturity T and
% rate (composite) r

p = 100/(1+r)^(T-t);
```



Exercise (Prezzo Coupon Bond)

Scrivere una funzione 'CB' che calcoli il prezzo P al tempo t di un Coupon Bond con scadenza T , tasso composto R , che paga una cedola posticipata c ad intervalli dt

$$B = P(t, T) + \frac{c}{100} P(t, T) + \frac{c}{100} P(t, T - dt) + \dots + \frac{c}{100} P(t, T - ndt)$$

$$n = \lceil \frac{T-t}{dt} \rceil$$



Exercise (Prezzo Coupon Bond)

Scrivere una funzione 'CB' che calcoli il prezzo P al tempo t di un Coupon Bond con scadenza T , tasso composto R , che paga una cedola posticipata c ad intervalli dt

$$B = P(t, T) + \frac{c}{100} P(t, T) + \frac{c}{100} P(t, T - dt) + \dots + \frac{c}{100} P(t, T - ndt)$$

$$n = \lceil \frac{T-t}{dt} \rceil$$

cb.m

```
function p = cb(t,T,r,c,dt)
% p = cb(t,T,r,c,dt)
%
% calcola il prezzo al tempo t di un coupon bond
% con scadenza T, cedola c, rate ad intervalli dt
% e tasso composto r
n = ceil((T-t)/dt);
p = 0;
for i=0:n-1
    p = p + zcb(t,T-i*dt,r)*c/100;
end
p = p + zcb(t,T,r);
```

Exercise (Black and Scholes Formula)

Scrivere una funzione che calcoli il prezzo C di una call utilizzando la formula di Black and Scholes:

$$C = S_0 \Phi(d^+) - e^{-rT} \Phi(d^-), \quad d^\pm = \frac{\log(S_0/K) + (r \pm \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}},$$

$\Phi(x)$ funzione di ripartizione di $N(0, 1)$



Exercise (Black and Scholes Formula)

Scrivere una funzione che calcoli il prezzo C di una call utilizzando la formula di Black and Scholes:

$$C = S_0 \Phi(d^+) - e^{-rT} \Phi(d^-), \quad d^\pm = \frac{\log(S_0/K) + (r \pm \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}},$$

$\Phi(x)$ funzione di ripartizione di $N(0,1)$

bs_call.m

```
function C = bs_call(S, K, T, r, sigma )
% C = bs_call(S, K, T, r, sigma )
%   Calcola il valore di una Call vanilla

d1 = (log(S/K) + (r+sigma^2/2)*T)/(sigma*sqrt(T));
d2 = d1 - sigma*sqrt(T);

C = S*mynormcdf(d1) - exp(-r*T)*K*mynormcdf(d2);
```

Scope, Variabili Locali e Globali

- Le funzioni sono scatole nere, le uniche interazioni con l'esterno avvengono tramite gli input/output.
- Nell'esempio precedente sono state definite le variabili $d1$ e $d2$.
- Variabili locali: tutte le variabili definite ed utilizzate in una funzione non hanno ripercussioni sull'ambiente (workspace) esterno/globale.
- Esiste uno workspace (locale) per ogni chiamata a funzione



Esempio: Stack e Scope

- Consideriamo la funzione

```
fun2.m
1 function y = fun2(x)
2 t = x^2;
3 y = 3+t;
```

Stack:

- E lo script

```
fun3.m
1 >> a = 3;
2 >> b = fun2(a)
3 b =
4     12
```

...



Esempio: Stack e Scope

- Consideriamo la funzione

```
fun2.m
1 function y = fun2(x)
2 t = x^2;
3 y = 3+t;
```

Stack:

- E lo script

```
fun3.m
1 >> a = 3;
2 >> b = fun2(a)
3 b =
4     12
```

a=3
...



Esempio: Stack e Scope

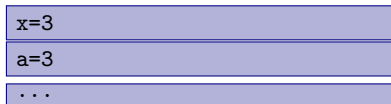
- Consideriamo la funzione

```
fun2.m
1 function y = fun2(x)
2 t = x^2;
3 y = 3+t;
```

- E lo script

```
fun3.m
1 >> a = 3;
2 >> b = fun2(a)
3 b =
4     12
```

Stack:



Esempio: Stack e Scope

- Consideriamo la funzione

```
fun2.m
1 function y = fun2(x)
2 t = x^2;
3 y = 3+t;
```

- E lo script

```
fun3.m
1 >> a = 3;
2 >> b = fun2(a)
3 b =
4     12
```

Stack:



Esempio: Stack e Scope

- Consideriamo la funzione

```
fun2.m
1 function y = fun2(x)
2 t = x^2;
3 y = 3+t;
```

- E lo script

```
fun3.m
1 >> a = 3;
2 >> b = fun2(a)
3 b =
4     12
```

Stack:

| |
|------|
| y=12 |
| t=9 |
| x=3 |
| a=3 |
| ... |



Esempio: Stack e Scope

- Consideriamo la funzione

```

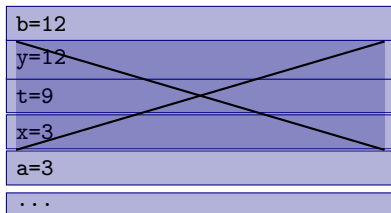
fun2.m
1 function y = fun2(x)
2 t = x^2;
3 y = 3+t;
  
```

- E lo script

```

fun3.m
1 >> a = 3;
2 >> b = fun2(a)
3 b =
4     12
  
```

Stack:



Esempio: Stack e Scope

- Consideriamo la funzione

```
fun2.m
1 function y = fun2(x)
2 t = x^2;
3 y = 3+t;
```

- E lo script

```
fun3.m
1 >> a = 3;
2 >> b = fun2(a)
3 b =
4     12
```

Stack:



Outline

1 Problemi, Algoritmi, Complessità ed Errori

2 L'ambiente Matlab

3 Programmazione e Matlab

- Variabili
- scripts
- Input/Output
- Controllo dell'esecuzione
- Espressioni
- Cicli
- Funzioni
- **Array, Vettori e Matrici**

4 Grafica in Matlab



Array

- Un array è un collezione di variabili dello stesso tipo organizzate in righe, colonne, strati, etc. . .
- La struttura dati fondamentale in `matlab` è la Matrice: array 2-d
- Un particolare tipo di matrice è il vettore: array 1-d, singola riga o colonna.
- Indice: posizione di un valore in un vettore
- Una posizione in una matrice è definita da due indici

| | | | | |
|---|----|---|----|----|
| 1 | 6 | 1 | 3 | -1 |
| 2 | 9 | 2 | 2 | -1 |
| 3 | 0 | 9 | -3 | 4 |
| 4 | 10 | 3 | 2 | 1 |
| | 1 | 2 | 3 | 4 |

| | |
|---|----|
| 1 | 10 |
| 2 | 2 |
| 3 | 6 |
| 4 | 4 |
| 5 | 2 |
| | 1 |

| | | | | | |
|---|----|---|---|---|---|
| 1 | 10 | 2 | 6 | 4 | 2 |
| | 1 | 2 | 3 | 4 | 5 |



Vettori

Creazione

- Separatori di colonna: spazio e “,”; separatore di riga: “;”
- Vettori riga: `vr = [2 3.1 -.01]`
- Vettori colonna: `vc = [2; 3.1; -.01]`
- Usare funzioni per la creazione di matrici:
 - `zeros(m,n)`, Crea una matrice $m \times n$
 - `vr = zeros(1,4)`, Crea una riga con 4 zeri
 - `vc = zeros(1,4)`, Crea una colonna con 4 zeri
- Funzioni analoghe: `ones`, `nan`, `inf`, `rand`, `randn`, `eye`
- `linspace(a,b,n)` crea un vettore riga con n valori equidist. da a a b

Accesso agli elementi:

- `vect(i)` rappresenta l' i -esimo elemento di `vect`.
- `vect(i)` è anche un l-value, cioè può essere utilizzato a sinistra di istruzioni di assegnamento



Vettori

- Esempio:

```
vet1.m
>> vect = [10 2 6 4 2 6]
vect =
    10     2     6     4     2     6
>> vect(5)
ans =
     2
>> k=3;
>> vect(k)
ans =
     6
>> vect(2) = k
vect =
    10     3     6     4     2     6
>> vect(1) = vect(k+2)
vect =
     2     3     6     4     2     6
```

Exercise (Attualizzazione flussi)

Calcolare il valore attuale (in $t=0$) di una serie di flussi c_i ai tempi t_i ($i = 1, \dots, n$).

Esempio di utilizzo

actualvalue_ex.m

```
t = [.5, 1, 1.5, 2, 2.5];  
c = [10, 10, 15, 15, 10];  
p = actualvalue( t, c, 0.04 );
```



Exercise (Attualizzazione flussi)

Calcolare il valore attuale (in $t=0$) di una serie di flussi c_i ai tempi t_i ($i = 1, \dots, n$).

Esempio di utilizzo

actualvalue_ex.m

```
t = [.5, 1, 1.5, 2, 2.5];  
c = [10, 10, 15, 15, 10];  
p = actualvalue( t, c, 0.04 );
```

actualvalue.m

```
function p = actualvalue( t, c, r )  
% p = actualvalue( t, c, r )  
%  
% Calcola il valore attuale dei flussi c(i) ai  
% tempi t(i) con tasso di interesse composto r  
n = length(t);  
p = 0;  
for i=1:n  
    p = p + zcb(0,t(i),r)*c(i)/100;
```

Exercise (Simulazione Random Walk)

Simulare n passi di una realizzazione del processo stocastico a tempo discreto:

$$X_t = X_{t-1} + \Delta W_t, \quad \Delta W_t \sim N(0, 1), \quad X_0 = x_0$$

(Specifiche: *function* $x = wiener(n, x_0)$ con $x : 1 \times n$, $x_0 : 1 \times 1$)

Esempio di utilizzo:

wiener2.m

```
z = wiener( 50, 0 );
t = linspace( 1, 50, 50 );
plot( t, z, '-' );
```

Implementazione:



Exercise (Simulazione Random Walk)

Simulare n passi di una realizzazione del processo stocastico a tempo discreto:

$$X_t = X_{t-1} + \Delta W_t, \quad \Delta W_t \sim N(0, 1), \quad X_0 = x_0$$

(Specifiche: *function* $x = wiener(n, x_0)$ con $x : 1 \times n$, $x_0 : 1 \times 1$)

Esempio di utilizzo:

wiener2.m

```
z = wiener( 50, 0 );
t = linspace( 1, 50, 50 );
plot( t, z, '-' );
```

Implementazione:

wiener.m

```
function x = wiener( n, x0 )

x = zeros(n,1);
x(1) = x0;
for i=1:n-1
```

Operazioni su vettori

- Operatori:
 - Aritmetici: +, -, ./, .* , .^
 - Relazionali: ==, ^=, <, >, <=, >=
 - Logici: &, |, ~
- Se applicati a:
 - Due scalari: no problem.
 - Due vettori: elemento per elemento
 - Scalare-Vettore: scalare per ogni elemento del vettore
- Esempi (a=.5; b=2; v=[1 2 3]; w=[4 5 6];):

| | |
|-----------------------|------------------------------|
| ● a ./ b → .25; | ● v > b → [0 0 1]; |
| ● v .* w → [4 10 18]; | ● v < w/3 → [1 0 0]; |
| ● v .^ b → [1 4 9]; | ● (v>b) (v<w/3) → [1 0 1]; |
| ● b .^ v → [2 4 8]; | ● 2 .^ (v>b) → [1 1 2]; |



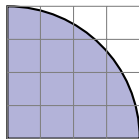
Operazioni su vettori

- $\mathbb{R}^N \rightarrow \mathbb{R}$: `sum`, `prod`, `max`, `min`, `mean`, `std`, `var`, `median`.
Esempio: `sum([1 2 3 4])` \rightarrow 10.
- $\mathbb{R}^N \rightarrow \mathbb{R}^m$: `cumsum`, `cumprod`, `diff`, `gradient`
Esempio: `diff([1 2 4 7 11])` \rightarrow [1 2 3 4]
- $\mathbb{R}^N \rightarrow \mathbb{R}^N$: `abs`, `sign`, `sqrt`, `exp`, `log(10,2)(a)`, `sin(h)`,
`(a)cos(h)`, `(a)tan(h)`, etc. . .
- $\mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^N$ or $\mathbb{R}^N \times \mathbb{R} \rightarrow \mathbb{R}^N$: `max`, `min`;
 - `max(-3:3, 0)` \rightarrow [0 0 0 0 1 2 3];
 - `max(-3:3, -6:2:6)` \rightarrow [-3 -2 -1 0 2 4 6].
- Esempio, calcolo di π

```

pigreco2.m
x = rand(N,1);
y = rand(N,1);
dentro = (x.^2 + y.^2 <= 1);
pi_greco = sum(dentro)/N*4;

```



Colon Notation

- L'espressione $1:5$ è equivalente a $[1 \ 2 \ 3 \ 4 \ 5]$.
- In generale: $a:b:c \equiv [a, a+b, \dots, a+m*b]$, con $m = \lfloor \frac{c-a}{b} \rfloor$.
- è possibile accedere a piú di un elemento di un vettore:
 $a([1,3]) \equiv [a(1) \ a(3)]$ (se a è un vettore riga).
- quindi se $a = [10 \ 2 \ 5 \ 4 \ 2 \ 6]$:
 - $a([1,5]) \rightarrow [10 \ 6]$
 - $a(3:6) \rightarrow [5 \ 4 \ 2 \ 6]$
 - $a(1:2:6) \rightarrow [10 \ 5 \ 2]$
 - $a([6:-1:1]) \rightarrow [6 \ 2 \ 4 \ 5 \ 2 \ 10]$
- Esiste un indice particolare “**end**” tale che $a \equiv a(1:\text{end})$.
 - $a(\text{end})$ è l'ultimo elemento di a
 - $a(1:2:\text{end})$ sono gli elementi di posizione dispari
 - $a(2:2:\text{end})$ quelli di posizione pari
 - $a(\text{end}:-1:1)$ il vettore con posizioni invertite



Matrici

- Ogni elemento è indirizzato da due indici: riga e colonna

| | | | | |
|---|----|---|----|----|
| 1 | 10 | 3 | 2 | 1 |
| 2 | 0 | 9 | -3 | 4 |
| 3 | 9 | 2 | 2 | -1 |
| 4 | 6 | 4 | 3 | 4 |

1 2 3 4

$A(3,2) \rightarrow 2$

- Colon notation:

| | | | | |
|---|----|---|----|----|
| 1 | 10 | 3 | 2 | 1 |
| 2 | 0 | 9 | -3 | 4 |
| 3 | 9 | 2 | 2 | -1 |
| 4 | 6 | 4 | 3 | 4 |

1 2 3 4

$A([1:2:\text{end}], [2:3]) \rightarrow [3 \ 2; 2 \ 2]$



Struttura fondamentale di Matlab

- 1×1 : scalari
- $1 \times N$: vettori riga
- $M \times 1$: vettori colonna
- $M \times N$: matrici

- Comodo per l'algebra lineare.
- Ci sono problemi di coerenza.
- Spesso fanno comodo strutture dati piú sofisticate.



Operazioni fra Matrici

- Tutti gli operatori “elemento per elemento” visti finora vengono utilizzati in maniera analoga fra matrici.
- Gli operatori prodotto e divisione, $*$, \setminus , $/$, rappresentano operazioni fra matrici:
 - $X=A*B$: moltiplicazione riga-per-colonna;
 - $X=A/B$: divisione a destra, X soluzione di $XB = A$;
 - $X=A\setminus B$: divisione a sinistra, X soluzione di $AX = B$;
 - Le dimensioni di A e B devono essere compatibili.
 - $XB = A$ e $AX = B$ possono essere anche sotto/sovra determinati o non-crameriani (soluzione minimi quadrati)
- Trasposizione: A' è la trasposta di A
- Se v e w sono due vettori colonna di uguali dimensioni: $v'*w$ è il loro prodotto scalare



Funzioni su matrici

- $\mathbb{R}^{M \times N} \rightarrow \mathbb{R}^{M \times N}$: (es. `abs`, `sin`) applicate ad ogni elemento;
- $\mathbb{R}^{M \times N} \rightarrow \mathbb{R}^N$: (es. `sum`, `prod`, `mean`) applicate ad ogni colonna;
- $\mathbb{R}^{M \times N} \rightarrow \mathbb{R}^{m \times N}$: (es. `cumsum`) applicate ad ogni colonna;
- `size(A)` restituisce un vettore di due elementi con le dimensioni di `A`
- `length(v)` restituisce il numero di elementi del vettore `v`
- Esempi:
 - `A = ones(3,2); v=1:5;`
 - `size(A)` \rightarrow `[3,2]`
 - `size(v)` \rightarrow `[1,5]`
 - `size(8)` \rightarrow `[1,1]`
 - `length(v)` \rightarrow `5`



Concatenazione di Matrici

- Matrici a blocchi: se $A : m \times n$, $B : m \times q$, $C : p \times n$ e $D : p \times q$

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \equiv \begin{pmatrix} a_{11} \cdots a_{1n} & b_{11} \cdots b_{1q} \\ \vdots & \vdots \\ a_{m1} \cdots a_{mn} & b_{m1} \cdots b_{mq} \\ c_{11} \cdots c_{1n} & d_{11} \cdots d_{1q} \\ \vdots & \vdots \\ c_{p1} \cdots c_{pn} & d_{p1} \cdots d_{pq} \end{pmatrix}$$

- In Matlab: $M = [A, B; C, D]$;
- Esempi: dati $v = [1 \ 2 \ 3]$; $w = [4 \ 5 \ 6]$; $x = [3; 4]$; $y = [2, 3; 4, 5]$;

- $[v, w] \rightarrow (1 \ 2 \ 3 \ 4 \ 5 \ 6)$

- $[v; w] \rightarrow \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

- $[x, y] \rightarrow \begin{pmatrix} 3 & 2 & 3 \\ 4 & 4 & 5 \end{pmatrix}$

- $[v', w'] \rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$

- $[[v; w], x] \rightarrow \begin{pmatrix} 1 & 2 & 3 & 3 \\ 4 & 5 & 6 & 4 \end{pmatrix}$



Indicizzazione con booleani

- Vettori booleani (Vero/Falso) ottenuti da confronti possono essere utilizzati per estrarre elementi da un array.
- Esempio: si vogliono estrarre tutti i valori positivi del vettore
 $x = [2.1 \ 3.4 \ -0.4 \ -1.2 \ 3.2 \ -2.5 \ 2.1]$
 - 1 $\text{pos} = (x >= 0)$; $\rightarrow \text{pos} = [1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1]$
 - 2 $y = x(\text{pos})$; $\rightarrow y = [2.1 \ 3.4 \ 3.2 \ 2.1]$
- In generale se x e pos sono due matrici di pari dimensioni e pos è di tipo booleano, allora $y = x(\text{pos})$ è il vettore che contiene solo i valori di x corrispondenti ai valori “true” (non zero) del vettore pos .



La funzione find

- Un metodo alternativo per estrarre valori che soddisfano una data condizione consiste nell'utilizzare la funzione `find`.
- `find(pos)` restituisce gli indici di tutti i valori "true" nel vettore/matrice `pos`
- Se `x` e `pos` sono vettori/matrici di pari dimensioni, allora `x(find(pos))` è equivalente a `x(pos)`.
- Esempio: dato il vettore `x = [2.1 3.4 -0.4 -1.2 3.2 -2.5 2.1]`
 - 1 `x >= 0` \rightarrow `[1 1 0 0 1 0 1]`
 - 2 `find(x>=0)` \rightarrow `[1 2 5 7]`
 - 3 `x(find(x>=0))` \rightarrow `[2.1 3.4 3.2 2.1]`



Exercise (Vettorializzazione)

Modificare la funzione `zcb` in modo che accetti matrici come argomenti per `t`, `T` e `r`



Exercise (Vettorializzazione)

Modificare la funzione `zcb` in modo che accetti matrici come argomenti per t , T e r

zcb2.m

```
function p = zcb2(t,T,r)
% p = zcb(t,T,r)
%
% Compute the value at t of a ZCB with maturity T and
% rate (composite) r

p = 100./(1+r).^ (T-t);
```



Exercise (Vettorializzazione)

Modificare la funzione `bs_call` in modo che accetti matrici come argomenti per K e T



Exercise (Vettorializzazione)

Modificare la funzione `bs_call` in modo che accetti matrici come argomenti per K e T

bs_call2.m

```
function C = bs_call2(S, K, T, r, sigma )
% C = bs_call(S, K, T, r, sigma )
%   Calcola il valore di una Call vanilla

d1 = (log(S./K) + (r+sigma^2/2).*T)./(sigma.*sqrt(T));
d2 = d1 - sigma.*sqrt(T);

C = S.*mynormcdf(d1) - exp(-r.*T).*K.*mynormcdf(d2);
```



Ciclo For

- Ciclo for: `for i=fist:by:last; ...; end`
- Caso generale: `for i=A; <expr>; end`,
 - se A è una matrice,
ripeti le istruzioni <expr> assegnando ad i ogni colonna di A
 - se A è un vettore riga,
ripeti le istruzioni <expr> assegnando ad i ogni valore di A



Outline

- 1 Problemi, Algoritmi, Complessità ed Errori
- 2 L'ambiente Matlab
- 3 Programmazione e Matlab
- 4 Grafica in Matlab**
 - Grafici di funzioni
 - Grafici di superfici



Outline

- 1 Problemi, Algoritmi, Complessità ed Errori
- 2 L'ambiente Matlab
- 3 Programmazione e Matlab
- 4 Grafica in Matlab**
 - Grafici di funzioni
 - Grafici di superfici



Creare un grafico

Siano x , y vettori di lunghezza n

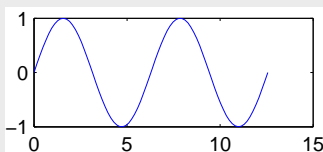
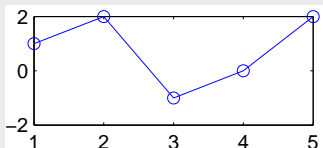
La funzione `plot` ha le seguenti specifiche:

- `plot(y)` produce il grafico della spezzata $(1, y_1), (2, y_2), \dots, (n, y_n)$;
- `plot(x,y)` produce la spezzata $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$;
- `plot(x,y, 'o')` disegna i punti $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$;

Example

```
y = [1 2 -1 0 2];
plot( y );
```

```
x = linspace(0,4*pi,80);
y = sin(x);
plot(x,y);
```

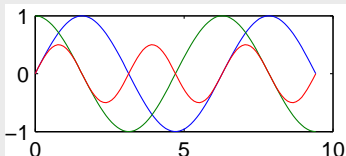


Diverse serie in unico grafico

`plot(x1,y1, x2,y2, x3,y3)`: visualizza in unico grafico tre spezzate.

Example

```
x = linspace(0,3*pi,100);
y1 = sin(x);
y2 = cos(x);
y3 = y1.*y2;
plot(x,y1, x,y2, x,y3 );
```



- `hold on` permette di aggiungere grafici alla figura corrente
- `hold off` fa in modo che la figura venga cancellata prima di ogni plot
- Il comando `subplot(m,n,i)` divide la figura in $m \times n$ sottofigure e si posiziona sulla i -esima.

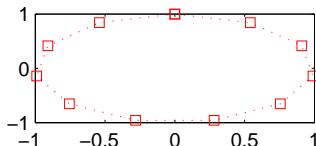


Specificare il tipo di linea

- `plot(x,y,'linestyle')` dove `linestyle` é una stringa composta dai seg. caratteri:

| Colori | | Linee | | Punti | |
|--------|---------|-------|--------------|-------|------------------|
| c | azzurro | - | continua | + | + |
| m | magenta | -- | tratteggiata | o | cerchietto |
| y | giallo | : | punteggiata | * | punto |
| r | rosso | -. | linea-punto | x | x |
| g | verde | | | s | quadrato |
| b | blue | | | ^v<> | triangoli |
| w | bianco | | | p | stella a 5 punte |
| k | nero | | | h | stella a 6 punte |

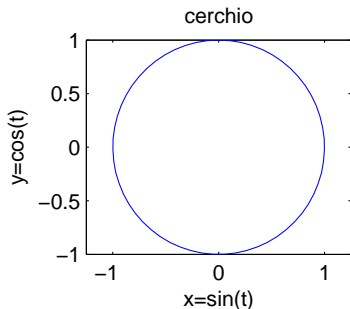
```
t = linspace(0,2*pi,12);
plot( sin(t), cos(t), 'r:s');
```



Annotare i grafici

- `title('titolo')` imposta il titolo della figura
- `xlabel` e `ylabel` impostano le etichette degli assi
- `legend` descrive ogni serie visualizzata
- `axis` imposta gli estremi degli assi

```
t = linspace(0,2*pi,60);  
plot( sin(t), cos(t) );  
axis([-1.25 1.25 -1 1])  
title('cerchio')  
xlabel('x=sin(t)')  
ylabel('y=cos(t)');
```



Outline

- 1 Problemi, Algoritmi, Complessità ed Errori
- 2 L'ambiente Matlab
- 3 Programmazione e Matlab
- 4 Grafica in Matlab**
 - Grafici di funzioni
 - **Grafici di superfici**

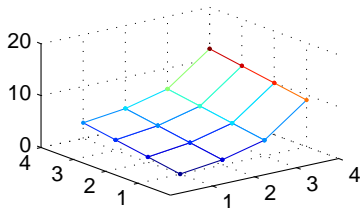


Visualizzare Matrici e Superfici

- `mesh(A)` visualizza una mesh sui punti (i, j, A_{ij})

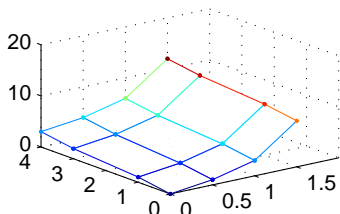
```
A = [0 1 3 9
      1 2 4 10
      2 3 5 11
      3 4 6 12 ];
```

```
mesh(A)
```



- `mesh(x,y,A)` visualizza una mesh con punti (x_j, y_i, A_{ij})

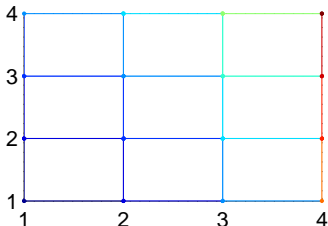
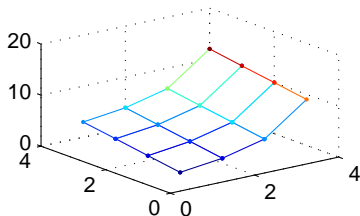
```
x = [0 .5 1 1.5];
y = [0 1 3 4];
mesh(x,y,A)
```



Visualizzare Matrici

- `mesh(X,Y,A)` crea una con punti (X_{ij}, Y_{ij}, A_{ij})

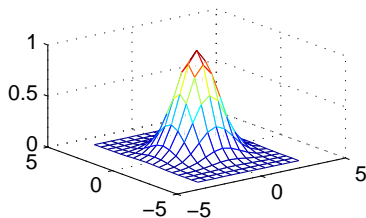
```
A = [0 1 3 9
      1 2 4 10
      2 3 5 11
      3 4 6 12 ] ;
X = [1 2 3 4
      1 2 3 4
      1 2 3 4
      1 2 3 4] ;
Y = [1 1 1 1
      2 2 2 2
      3 3 3 3
      4 4 4 4] ;
mesh(X,Y,A)
```



Funzioni in 2 variabili

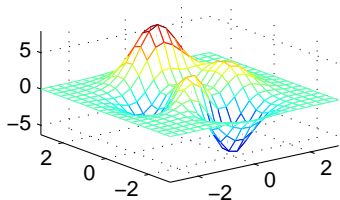
- Per visualizzare funzioni $z = f(x, y)$
 - generare una griglia di punti (x, y) , cioè due matrici X, Y ($m \times n$) contenenti le coordinate dei punti
 - generare la matrice Z in modo che $z(i, j) = f(x(i, j), y(i, j))$.
- Esempio: visualizzare la funzione $z = \exp(-(x^2 + y^2)/2)$

```
xpts = linspace(-3,3,13);  
ypts = linspace(-4,4,17);  
[X,Y] = meshgrid(xpts,ypts);  
Z = exp(-(X.^2+Y.^2)/2);  
mesh(X,Y,Z);
```

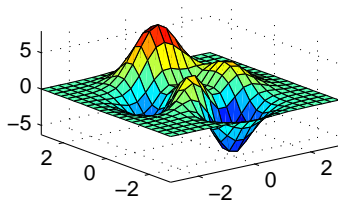


Visualizzazione di superfici

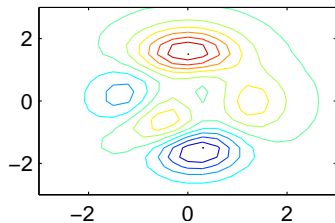
`mesh(X,Y,Z);`



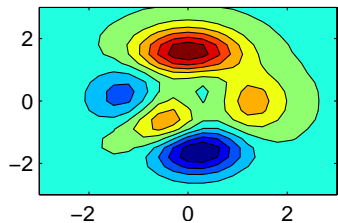
`surf(X,Y,Z);`



`contour(X,Y,Z);`



`contourf(X,Y,Z);`



Payoff: valore di un derivato in funzione di Strike (K) e Scadenza (T)

Exercise (Grafico payoff opzioni europee)

Utilizzare la funzione `bs_call` per visualizzare il Payoff di un'opzione europea

Dati: $S_0 = 1$, $\sigma = 0.2$, $r = 0.03$, $K \in [0.5, 1.4]$ e $T \in [0, 1]$

```
S=1; r=.03; sigma=.2;
m=11; K=linspace(.5,1.5,m);
n=13; T=linspace(0,1,n);
U = zeros(m,n);
for i=1:m
    for j=1:n
        if (T(j)==0)
            U(i,j)=max(S - K(i),0);
        else
            U(i,j)=bs_call(S,K(i),T(j),r,sigma);
        end
    end
end
surf(T,K,U);
```



Esercizio: payoff di opzioni europee

- Oppure...

```
S=1; r=.03; sigma=.2;
```

```
m=11; K=linspace(.5,1.5,m);
```

```
n=13; T=linspace(0,1,n);
```

```
[K,T] = meshgrid(K,T);
```

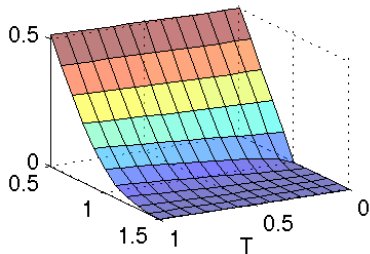
```
U = bs_call2(S,K,T,r,sigma);
```

```
U(1,:) = max(S-K(1,:),0);
```

```
surf(T,K,U);
```

```
xlabel('T'); ylabel('K');
```

```
alpha(0.5); view(150,25);
```



Part II

Computational methods for option pricing



Outline

5 Binomial Model

- Mono-period binomial model
- Parameter Calibration
- Early exercise
- Computational and numerical issues

6 Finite Differences

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Outline

5 Binomial Model

- Mono-period binomial model
- Parameter Calibration
- Early exercise
- Computational and numerical issues

6 Finite Differences

7 Monte Carlo

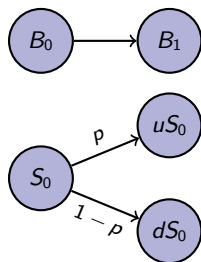
8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Binomial model

- Discrete time: $t_0 = 0$, $t_1 = T$.
- Risk-free asset: $B_1 = B_0(1 + \rho)$.
- Risky asset: $S_1 = \begin{cases} uS_0, & \text{prob. } p, \\ dS_0, & \text{prob. } 1 - p. \end{cases}$
- Portfolio:
 - $V_n = \alpha_n S_n + \beta_n B_n$.
 - Self-financing: $\alpha_{n-1} S_n + \beta_{n-1} B_n = V_n$.
 - Predictable: α_n and β_n only depend on the past.
- The binomial model is:
 - arbitrage free;
 - complete: every european derivative can be replicated by means of a self-financing portfolio.



Arbitrage

- Arbitrage portfolio V
 - 1 V is self-financing and predictable
 - 2 $P[V_0 = 0] = 1$,
 - 3 $P[V_n \geq 0] = 1$ and $P[V_n > 0] > 0$ for some n
- No arbitrage implies $d < 1 + \rho < u$.

- Suppose $1 + \rho \leq d < u$.

Let consider the self-financing portfolio $V_n = S_n - \frac{S_0}{B_0} B_n$.

Then, $V_0 = 0$ and $V_1 = S_1 - \frac{S_0}{B_0} B_1 = S_1 - S_0(1 + \rho) \geq 0$.

Furthermore, with probability $p > 0$, $S_1 = uS_0$ and $V_1 > 0$.

- Assume $d < u \leq 1 + \rho$.

and consider the self-financing portfolio $V_n = \frac{S_0}{B_0} B_n - S_n$.

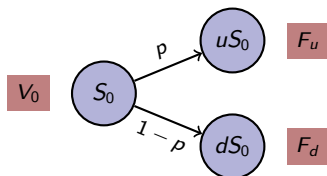
Then, $V_0 = 0$ and $V_1 = S_0(1 + \rho) - S_1 \geq 0$ and $V_1 > 0$ with probability $(1 - p) > 0$.

- When $d < 1 + \rho < u$ the binomial model is arbitrage-free
 - Consider the self-financing portfolio: $V_n = \gamma S_n - \gamma \frac{S_0}{B_0} B_n$ ($V_0 = 0$).



Replicating portfolio

- Portfolio: $V_n = \alpha_n S_n + b_n$, $n = 0, 1$
- Should replicate the payoff $V_1 = F(S_1)$:
 - $\alpha_1 uS_0 + b_1 = F_u \equiv F(uS_0)$
 - $\alpha_1 dS_0 + b_1 = F_d \equiv F(dS_0)$
- the solution of this linear system is



$$\alpha_1 = S_0^{-1} \frac{F_u - F_d}{u - d} \quad b_1 = \frac{uF_d - dF_u}{u - d}$$

- Single period (no rebalancing): $\alpha_0 = \alpha_1 = S_0^{-1} \frac{F_u - F_d}{u - d}$, $b_0 = \frac{1}{1+\rho} b_1$
- $V_0 = \frac{F_u - F_d}{u - d} + \frac{uF_d - dF_u}{(1+\rho)(u-d)} = \frac{(1+\rho-d)F_u - (1+\rho-u)F_d}{(1+\rho)(u-d)}$
- Delta Hedging: $\alpha_0 = \frac{F_u - F_d}{S_u - S_d} = \frac{\Delta F}{\Delta S}$



Equivalent Martingale Measure

- $V_0 = \frac{(1+\rho-d)F_u - (1+\rho-u)F_d}{(1+\rho)(u-d)} = \frac{1}{1+\rho} \left(\frac{1+\rho-d}{u-d} F_u + \frac{u-(1+\rho)}{u-d} F_d \right)$
- $q = \frac{1+\rho-d}{u-d}$, $1-q = -\frac{1+\rho-u}{u-d}$,
- if $d < 1+\rho < u$ then $0 < q < 1$

⇒ q defines a probability measure, let call it Q

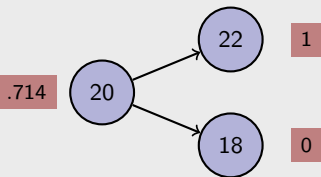
- $V_0 = \frac{1}{1+\rho} (qF_u + (1-q)F_d) = \frac{1}{1+\rho} E^Q[F(S_1)] = \frac{1}{1+\rho} E^Q[V_1]$

here V_1 is the value of the contingent claim at time $t = 1$.

The value of the contingent claim is given by the discounted expected value (w.r.t. the measure Q) of its payoff:

$$V_0 = \frac{1}{1+\rho} E^Q[V_1]$$

Example



- $\rho = 0.05$
- $S_0 = 20, u = 1.1, d = .9$
- $F(S_1) = \max(S_1 - 21, 0)$

- The replication portfolio is $V_1 = \alpha_1 S_1 + b_1 \equiv F(S_1)$

$$\begin{cases} \alpha_1 18 + b_1 = 1 \\ \alpha_1 22 + b_1 = 0 \end{cases} \Rightarrow \begin{cases} \alpha_1 = \frac{1-0}{22-18} = 1/4 \\ b_1 = -22\alpha_1 = -22/4 \end{cases}$$

so that

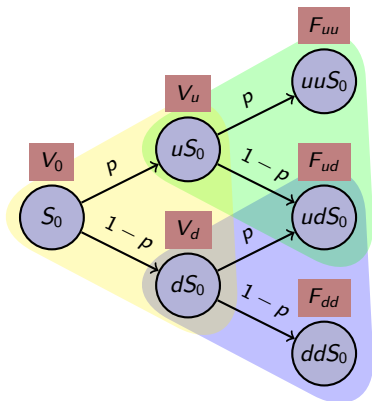
$$V_0 = \alpha_1 S_0 + b_0 = 1/4 \cdot 20 - \frac{1}{1.05} 22/4 = 0.714$$

- Alternatively (martingale approach):

$$\bullet q = \frac{1+\rho-d}{u-d} = \frac{1+.05-.9}{1.1-.9} = \frac{.15}{.2} = 3/4$$

$$\bullet V_0 = \frac{1}{1+\rho} E^Q[V_1] = \frac{1}{1.05} \left(\frac{3}{4} \cdot 1 + \frac{1}{4} \cdot 0 \right) = \frac{3/4}{1.05} = 0.714$$

A two-steps binomial model



Dynamics:

- $S_{n+1} = \begin{cases} uS_n, & p \\ dS_n, & 1-p \end{cases}$
- $B_{n+1} = (1 + \rho)B_n$

At time $t = 1$

The payoff F_{uu}, F_{ud} is replicable by a portfolio with value V_u

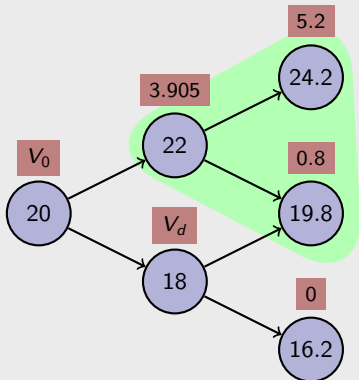
The payoff F_{ud}, F_{dd} is replicable by portfolio with value V_d

At time $t = 0$

The payoff V_u, V_d is replicable by a portfolio with value V_0

Example

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F_T = \max(S_T - 19, 0)$



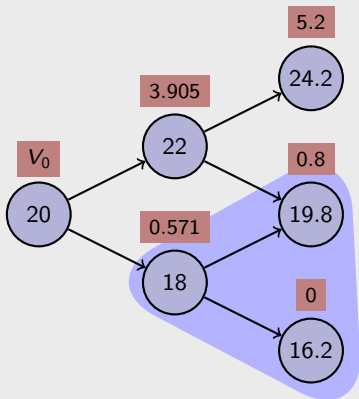
$$\alpha_u = \frac{1}{22} \cdot \frac{5.2 - .8}{.2} = 1$$

$$b_u = \frac{1.1 \cdot 0.8 - 0.9 \cdot 5.2}{.2 \cdot 1.05} = -18.095$$

$$V_u = 22 \cdot 1 - 18.095 = 3.905$$

Example

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F_T = \max(S_T - 19, 0)$



$$\alpha_u = \frac{1}{22} \cdot \frac{5.2 - .8}{.2} = 1$$

$$b_u = \frac{1.1 \cdot 0.8 - 0.9 \cdot 5.2}{.2 \cdot 1.05} = -18.095$$

$$V_u = 22 \cdot 1 - 18.095 = 3.905$$

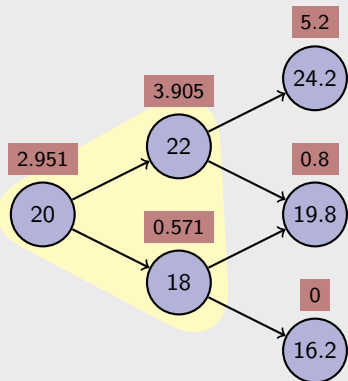
$$\alpha_d = 18^{-1} \frac{0.8 - 0}{0.2} = 0.222$$

$$b_d = \frac{1.1 \cdot 0 - 0.9 \cdot 0.8}{0.2 \cdot 1.05} = -3.429$$

$$V_d = 0.2222 \cdot 18 - 3.429 = 0.571$$

Example

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F_T = \max(S_T - 19, 0)$



$$\alpha_u = \frac{1}{22} \cdot \frac{5.2 - .8}{.2} = 1$$

$$b_u = \frac{1.1 \cdot 0.8 - 0.9 \cdot 5.2}{.2 \cdot 1.05} = -18.095$$

$$V_u = 22 \cdot 1 - 18.095 = 3.905$$

$$\alpha_d = 18^{-1} \frac{0.8 - 0}{0.2} = 0.222$$

$$b_d = \frac{1.1 \cdot 0 - 0.9 \cdot 0.8}{0.2 \cdot 1.05} = -3.429$$

$$V_d = 0.2222 \cdot 18 - 3.429 = 0.571$$

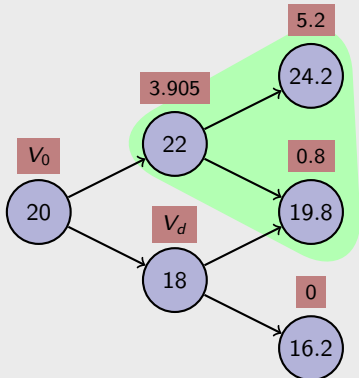
$$\alpha = 20^{-1} \frac{3.905 - .571}{.2} = 0.834$$

$$b = \frac{1.1 \cdot 0.571 - .9 \cdot 3.905}{.2 \cdot 1.05} = -13.729$$

$$V_0 = 20 \cdot 0.834 - 13.729 = 2.951$$

Example

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F_T = \max(S_T - 19, 0)$

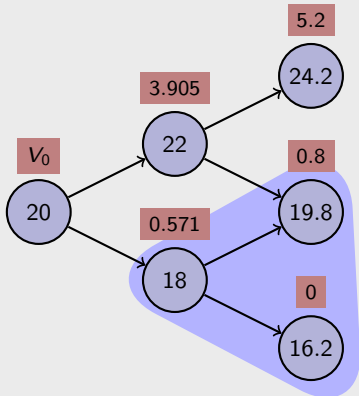


$$q = \frac{1 + \rho - d}{u - d} = \frac{1.05 - 0.9}{1.1 - 0.9} = 0.75$$

$$V_u = \frac{0.75 \cdot 5.2 + 0.25 \cdot 0.8}{1.05} = 3.905$$



Example

 $S_0 = 20, u = 1.1, d = .9, \rho = 0.05, F_T = \max(S_T - 19, 0)$


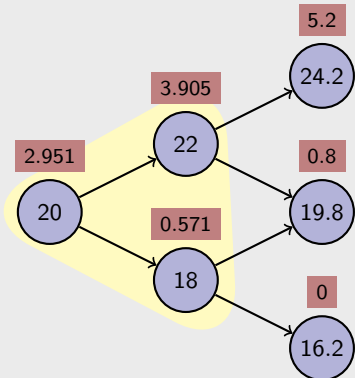
$$q = \frac{1 + \rho - d}{u - d} = \frac{1.05 - 0.9}{1.1 - 0.9} = 0.75$$

$$V_u = \frac{0.75 \cdot 5.2 + 0.25 \cdot 0.8}{1.05} = 3.905$$

$$V_d = \frac{0.75 \cdot 0.8 + 0.25 \cdot 0}{1.05} = 0.571$$



Example

 $S_0 = 20, u = 1.1, d = .9, \rho = 0.05, F_T = \max(S_T - 19, 0)$


$$q = \frac{1 + \rho - d}{u - d} = \frac{1.05 - 0.9}{1.1 - 0.9} = 0.75$$

$$V_u = \frac{0.75 \cdot 5.2 + 0.25 \cdot 0.8}{1.05} = 3.905$$

$$V_d = \frac{0.75 \cdot 0.8 + 0.25 \cdot 0}{1.05} = 0.571$$

$$V = \frac{0.75 \cdot 3.905 + 0.25 \cdot 0.571}{1.05} = 2.951$$



Multiperiod Binomial Model

- Each node corresponds to a specific time (n) and a specific scenario (j)
- At time n in the scenario with j up-movements ($n - j$ down-moves) the underlying is given by
 - $S_{nj} = u^j d^{n-j} S_0, \quad n = 1, \dots, N, j = 0, \dots, n$
- Analogously the values for claim and the portfolio composition are:
 - $V_{nj}, \alpha_{nj}, b_{nj}, \text{ for } n = 1, \dots, N, j = 0, \dots, n$

Algorithm

$$\bullet S_{0,0} = S_0, \quad S_{n+1,j} = dS_{n,j} \quad \text{and} \quad S_{n+1,j+1} = uS_{n,j} \quad (\text{forward})$$

$$\bullet V_{Nj} = F(S_{Nj}), \quad V_{nj} = \frac{qV_{n+1,j+1} + (1-q)V_{n+1,j}}{1+\rho} \quad (\text{backward})$$

$$\bullet \alpha_{nj} = \frac{V_{n+1,j+1} - V_{n+1,j}}{S_{n+1,j+1} - S_{n+1,j}}, \quad b_{nj} = V_{nj} - \alpha_{nj}S_{nj} \quad (\text{backward})$$

Matlab implementation

binomial1.m

```
function [C,V,S] = binomial1(S0,K,u,d,rho,N)

S=nan(N,N);  V=nan(N,N);

% Scenario Generation
S(1,1)=S0;
for n=1:N-1
    for j=1:n; S(n+1,j) = d*S(n,j); end
    S(n+1,n+1) = u*S(n,n);
end

% Compute replicating portfolio
q = (1+rho-d)/(u-d);
V(N,:) = max(S(N,:)-K,0);
for n=N-1:-1:1
    for j=1:n
        V(n,j) = (q*V(n+1,j+1) + (1-q)*V(n+1,j))/(1+rho);
    end
end
C = V(1,1);
```

Matlab implementation

binomial2.m

```

function [C,V,S] = binomial2(S0,K,u,d,rho,N)

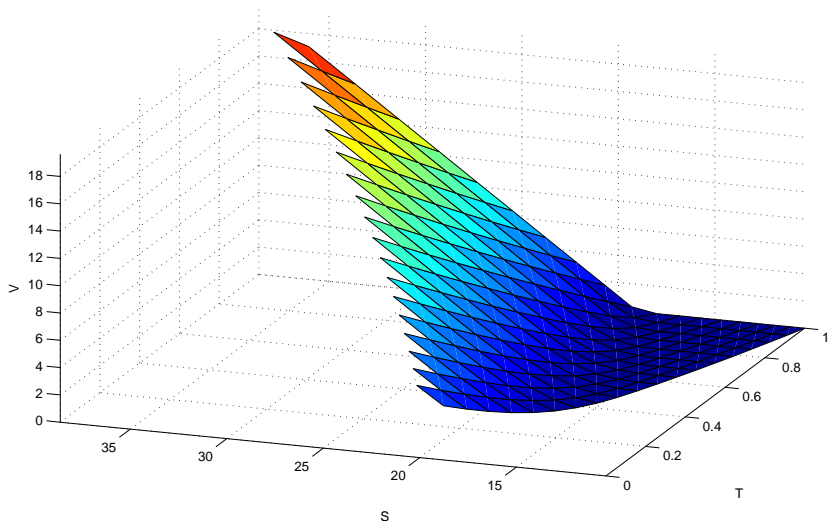
S=nan(N,N);  V=nan(N,N);

% Scenario Generation
S(1,1)=S0;
for n=1:N-1
    S(n+1,1:n) = d*S(n,1:n);
    S(n+1,n+1) = u*S(n,n);
end

% Compute replicating portfolio
q = (1+rho-d)/(u-d);
V(N,:) = max(S(N,:)-K,0);
for n=N-1:-1:1
    V(n,1:n) = (q*V(n+1,2:n+1) + (1-q)*V(n+1,1:n))/(1+rho);
end
C = V(1,1);

```

Payoff computed by the Binomial method



Outline

5 Binomial Model

- Mono-period binomial model
- **Parameter Calibration**
- Early exercise
- Computational and numerical issues

6 Finite Differences

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Calibration

Consider the r.v. S_{n+1} :
$$S_{n+1} = \begin{cases} uS_n, & \text{prob. } p \\ dS_n, & \text{prob. } 1 - p \end{cases}$$

Equivalently: $S_{n+1} = S_n \xi_n$ with $\xi_n = \begin{cases} u, & \text{prob. } p \\ d, & \text{prob. } 1 - p \end{cases}$ (independent)

Thus,

$$\log(S_N/S_0) = \sum_{n=1}^N \log(\xi_n) \sim N \log(\xi_n) \quad (\text{the } \xi_n \text{ are i.i.d.})$$

so that

$$E[\log(S_N/S_0)] = N E[\log(\xi_n)]$$

and

$$\text{Var}[\log(S_N/S_0)] = N^2 \text{Var}[\log(\xi_n)]$$



Theorem (Moments of the binomial distribution)

Let $\xi = \begin{cases} u, & \text{prob. } p, \\ d, & \text{prob. } 1-p. \end{cases}$ then

$$E[\log(\xi)] = p \log(u) + (1-p) \log(d) \quad \text{and} \quad \text{Var}[\log(\xi)] = p(1-p) \log(u/d)^2$$

Proof.

$$\begin{aligned} \text{Var}[\log(\xi)] &= E[\log(\xi)^2] - E[\log(\xi)]^2 \\ &= p \log(u)^2 + (1-p) \log(d)^2 \\ &\quad - p^2 \log(u)^2 - (1-p)^2 \log(d)^2 - 2p(1-p) \log(u) \log(d) \\ &= p(1-p) \log(u)^2 + p(1-p) \log(d)^2 - 2p(1-p) \log(u) \log(d) \\ &= p(1-p) (\log(u) - \log(d))^2 \\ &= p(1-p) \log(u/d)^2 \end{aligned}$$



Binomial model: calibration

Given T (expiration) and N (number of steps) $\Rightarrow \Delta_t = T/N$.

From the continuous risk-free rate r : $\rho = e^{r\Delta_t} - 1$.

The risky asset log-return: $\mu = \frac{1}{T} \log(S_T/S_0) \sim \frac{N}{T} \log(\xi_n) = \frac{1}{\Delta_t} \log(\xi_n)$.

Want to match market expected return m and volatility σ^2 :

$$\begin{cases} \frac{1}{\Delta_t} \mathbb{E}[\log \xi_n] & = m \\ \frac{1}{\Delta_t^2} \text{Var}[\log \xi_n] & = \sigma^2 \end{cases} \Rightarrow \begin{cases} p \log(u/d) + \log(d) & = m\Delta_t \\ \sqrt{p(1-p)} \log(u/d) & = \sigma\Delta_t \end{cases}$$

$$p = 1/2$$

$$\log(u/d) = 2\sigma\Delta_t$$

$$\log(d) = m\Delta_t - \sigma\Delta_t$$

$$\xi_n = \exp((m \pm \sigma)\Delta_t)$$

$$ud = 1$$

$$\log(u/d) = \log(u^2) = 2 \log(u)$$

...

Working in a risk-neutral measure, better to use $m = r$.



Outline

5 Binomial Model

- Mono-period binomial model
- Parameter Calibration
- **Early exercise**
- Computational and numerical issues

6 Finite Differences

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



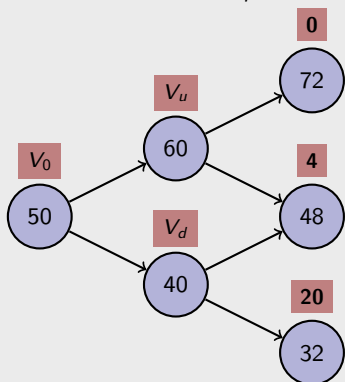
American options

- European options: exercise only at expiration
- Opzioni Americane: can be exercised before or at expiration
- At each time the holder of the contract can choose if
 - exercise the option right
 - hold the contract
- Will choose the maximum between
 - the current payoff
 - the fair value of the option at time t (discounted expected value of the future payoff)
- $V_n = \max(F(S_n), (1 + \rho)^{-1} E^Q[V_{n+1}|S_n])$



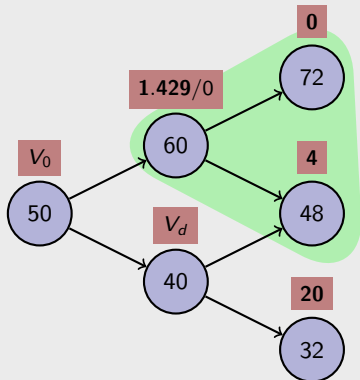
Example (American Options)

$$S_0 = 50, u = 1.2, d = .8, \rho = 0.05, F = \max(52 - S, 0) \Rightarrow q = .625$$



Example (American Options)

$S_0 = 50$, $u = 1.2$, $d = .8$, $\rho = 0.05$, $F = \max(52 - S, 0) \Rightarrow q = .625$



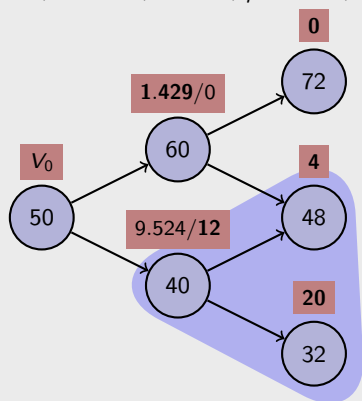
$$V_u^* = \frac{0.625 \cdot 0 + 0.375 \cdot 4}{1.05} = 1.429$$

$$F_u = \max(52 - 60, 0) = 0$$



Example (American Options)

$S_0 = 50$, $u = 1.2$, $d = .8$, $\rho = 0.05$, $F = \max(52 - S, 0) \Rightarrow q = .625$



$$V_u^* = \frac{0.625 \cdot 0 + 0.375 \cdot 4}{1.05} = 1.429$$

$$F_u = \max(52 - 60, 0) = 0$$

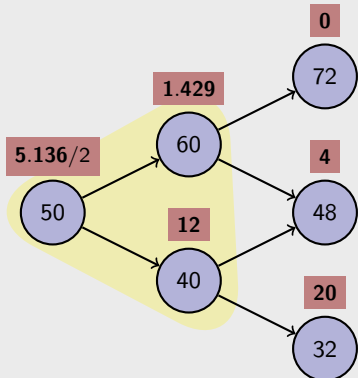
$$V_d = \frac{0.625 \cdot 4 + 0.375 \cdot 20}{1.05} = 9.524$$

$$F_d = \max(52 - 40, 0) = 12$$



Example (American Options)

$$S_0 = 50, u = 1.2, d = .8, \rho = 0.05, F = \max(52 - S, 0) \Rightarrow q = .625$$



$$V_u^* = \frac{0.625 \cdot 0 + 0.375 \cdot 4}{1.05} = 1.429$$

$$F_u = \max(52 - 60, 0) = 0$$

$$V_d = \frac{0.625 \cdot 4 + 0.375 \cdot 20}{1.05} = 9.524$$

$$F_d = \max(52 - 40, 0) = 12$$

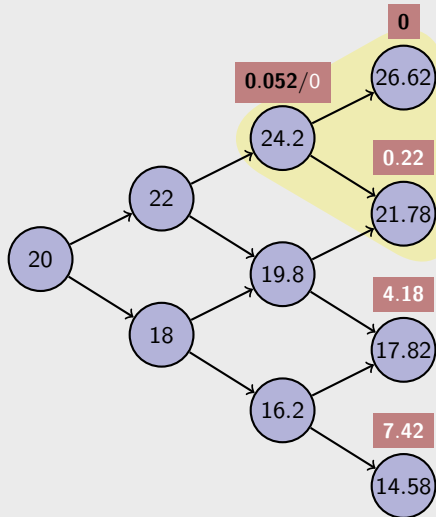
$$V = \frac{0.625 \cdot 1.429 + 0.375 \cdot 12}{1.05} = 5.136$$

$$F = \max(52 - 50, 0) = 2$$



Example (American Options)

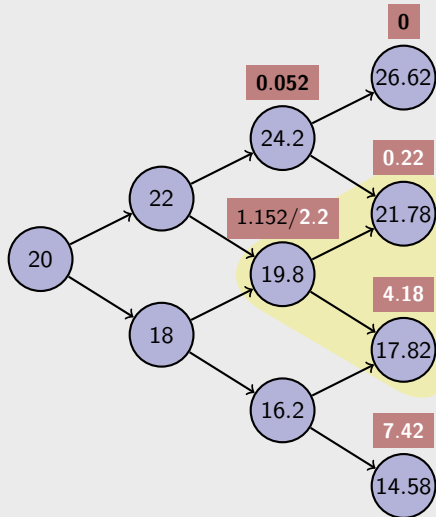
$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F = \max(22 - S, 0) \Rightarrow q = .75$,



$$V_{uu}^* = 0.052, F_{uu} = 0$$

Example (American Options)

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F = \max(22 - S, 0) \Rightarrow q = .75$,

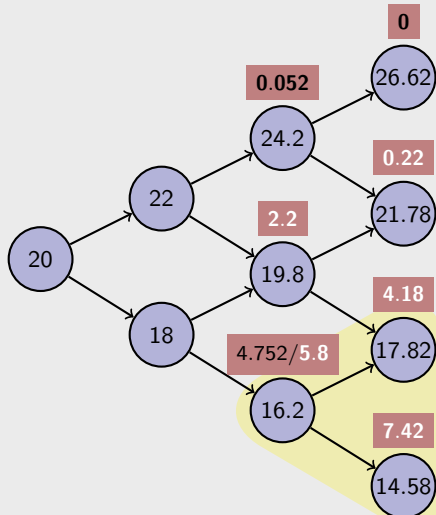


$$V_{uu}^* = 0.052, F_{uu} = 0$$

$$V_{ud}^* = 1.152, F_{ud} = 2.2$$

Example (American Options)

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F = \max(22 - S, 0) \Rightarrow q = .75$,



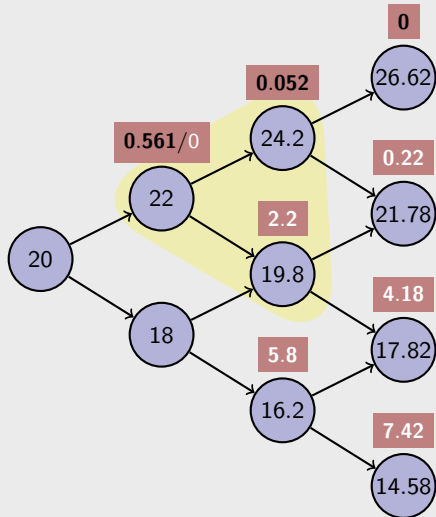
$$V_{uu}^* = 0.052, F_{uu} = 0$$

$$V_{ud}^* = 1.152, F_{ud} = 2.2$$

$$V_{dd}^* = 4.752, F_{dd} = 5.8$$

Example (American Options)

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F = \max(22 - S, 0) \Rightarrow q = .75$,



$$V_{uu}^* = 0.052, F_{uu} = 0$$

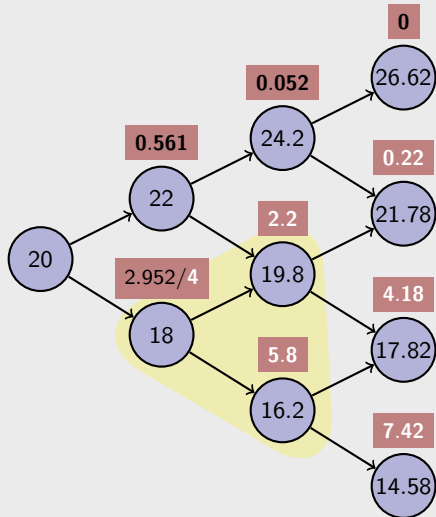
$$V_{ud}^* = 1.152, F_{ud} = 2.2$$

$$V_{dd}^* = 4.752, F_{dd} = 5.8$$

$$V_u^* = 0.561, F_u = 0$$

Example (American Options)

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F = \max(22 - S, 0) \Rightarrow q = .75$,



$$V_{uu}^* = 0.052, F_{uu} = 0$$

$$V_{ud}^* = 1.152, F_{ud} = 2.2$$

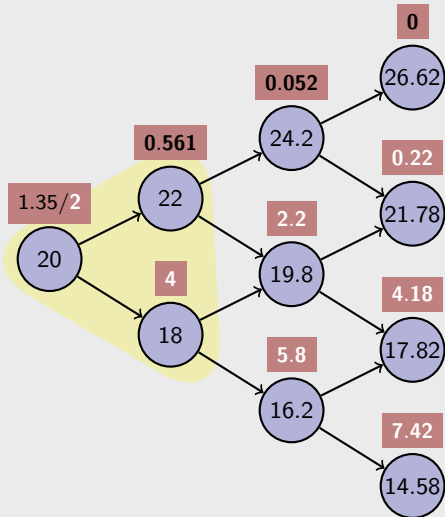
$$V_{dd}^* = 4.752, F_{dd} = 5.8$$

$$V_u^* = 0.561, F_u = 0$$

$$V_d^* = 2.952, F_d = 4$$

Example (American Options)

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F = \max(22 - S, 0) \Rightarrow q = .75$,



$$V_{uu}^* = 0.052, F_{uu} = 0$$

$$V_{ud}^* = 1.152, F_{ud} = 2.2$$

$$V_{dd}^* = 4.752, F_{dd} = 5.8$$

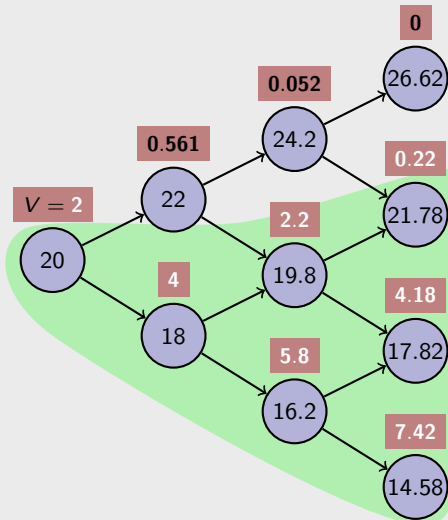
$$V_u^* = 0.561, F_u = 0$$

$$V_d^* = 2.952, F_d = 4$$

$$V^* = 1.35, F = 2$$

Example (American Options)

$S_0 = 20$, $u = 1.1$, $d = .9$, $\rho = 0.05$, $F = \max(22 - S, 0) \Rightarrow q = .75$,



$$V_{uu}^* = 0.052, F_{uu} = 0$$

$$V_{ud}^* = 1.152, F_{ud} = 2.2$$

$$V_{dd}^* = 4.752, F_{dd} = 5.8$$

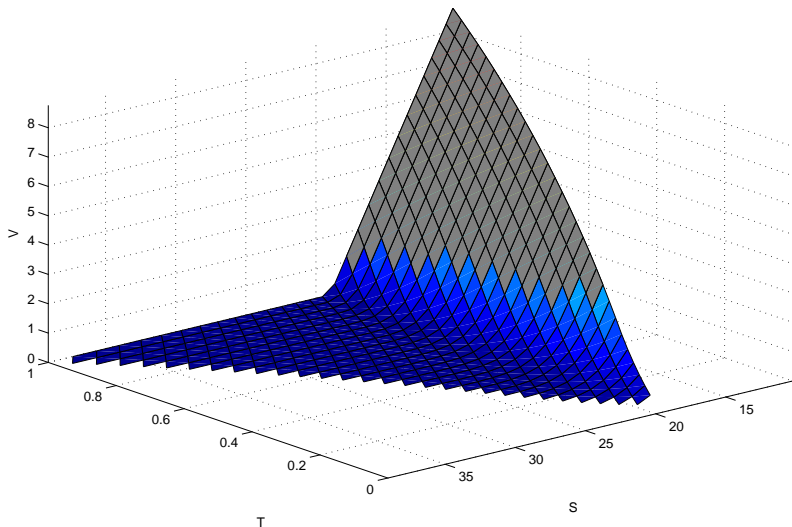
$$V_u^* = 0.561, F_u = 0$$

$$V_d^* = 2.952, F_d = 4$$

$$V^* = 1.35, F = 2$$

\Rightarrow early exercise region

Payoff of an American Put



Outline

5 Binomial Model

- Mono-period binomial model
- Parameter Calibration
- Early exercise
- Computational and numerical issues

6 Finite Differences

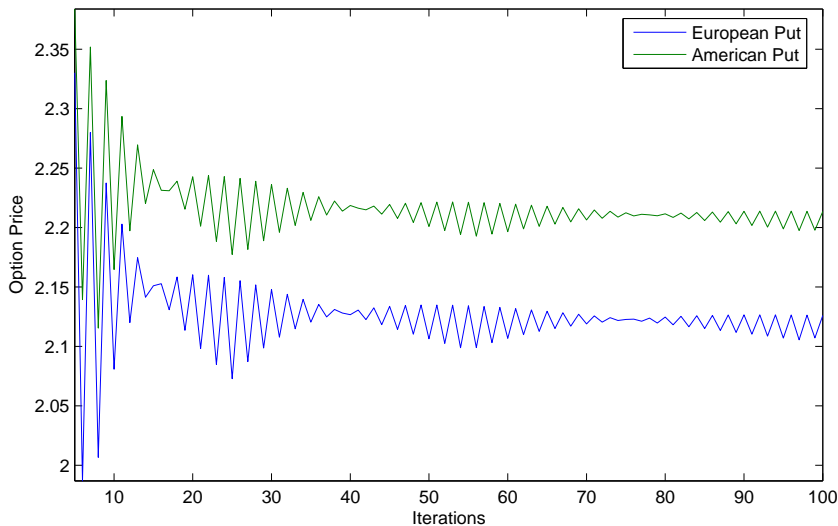
7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Convergence



Computational efficiency

To save memory: vectors instead of matrices (more in [10])

binomial4.m

```
function P = binomial4(S0,K,u,d,rho,N)

S=nan(N,1);  V=nan(N,1);
% Scenario Generation
S(1) = S0*d^(N-1);
for j=2:N
    S(j) = S(j-1)*u/d;
end

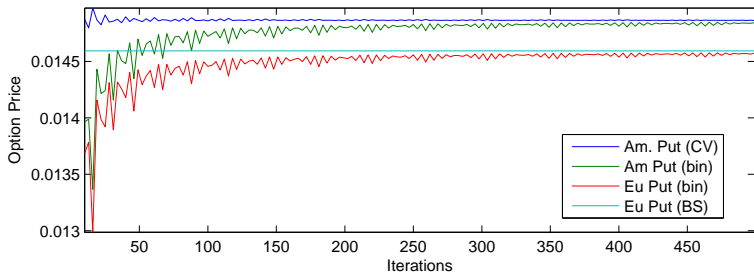
% Claim price computation
q = (1+rho-d)/(u-d);
V(:) = max(K-S(:),0);
for n=N-1:-1:1
    S(1:n) = S(1:n)/d;
    V(1:n) = max( (q*V(2:n+1) + (1-q)*V(1:n))/(1+rho), K-S(1:n) );
end
P = V(1,1);
```

Control Variates

Let

- P^A : American Put price (unknown)
- P^E : European Put price (known by B&S)
- P_n^A : American Put price computed by an n -steps binomial method
- P_n^E : European Put price computed by an n -steps binomial method

Heuristics: $P^A - P_n^A \simeq P^E - P_n^E$ da cui $P^A \simeq P_n^A - (P^E - P_n^E) =: P_n^{CV}$



binomial_cv.m bin_convergence2.m

Control Variates

binomial_cv.m

```

function [P,PA,PE,PBS] = binomial_cv(S0,K,T,r,sigma,N)

u=exp(sigma*sqrt(T/N)); d=1/u;
rho=exp(r*T/N)-1;
q = (1+rho-d)/(u-d);

% Scenario Generation
S=nan(N,1); S(1)=S0*d^(N-1);
for j=2:N; S(j)=S(j-1)*u/d; end

% Option pricing
VE = max(K-S(:),0); V=VE;
for n=N-1:-1:1
    S(1:n) = S(1:n)/d;
    VE(1:n) = (q*VE(2:n+1) + (1-q)*VE(1:n))/(1+rho);
    V(1:n) = max( (q*V(2:n+1) + (1-q)*V(1:n))/(1+rho), K-S(1:n) );
end
PE=VE(1,1); PA=V(1,1); PBS = bs_put(S0,K,T,r,sigma);
P = PA + PBS-PE;

```

Outline

5 Binomial Model

6 Finite Differences

- Discretization
- Explicit Method
- Implicit Method
- The Black and Scholes PDE
- Bi-dimensional equations

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



The Heat Equation

- Usually, partial differential equations arising in option pricing are
 - of second order
 - parabolic (or semi-parabolic)

- A prototype equation is the Heat Equation:

$$\partial_{xx} u(x, t) - \partial_t u(x, t) = 0 \quad \text{on } [a, b] \times [0, T]$$

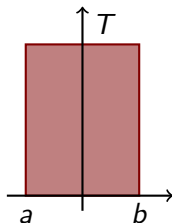
- Evolution equation with initial conditions

$$u(x, 0) = \phi(x), \quad x \in [a, b]$$

- A finite domain $a \neq -\infty$ e $b \neq \infty$ needs border conditions:

$$u(a, t) = \phi_a(t), \quad u(b, t) = \phi_b(t) \quad t \in]0, T]$$

- The Black & Scholes equations can be reformulated as a Heat Eqn.



Outline

5 Binomial Model

6 Finite Differences

- Discretization
- Explicit Method
- Implicit Method
- The Black and Scholes PDE
- Bi-dimensional equations

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



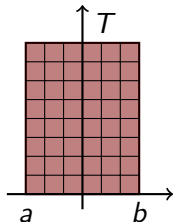
Discretization

- Continuous domain.
- Memory and computing time are finite.
- Discretization: approximate u on a finite grid
- Uniform grid:

- $(i, j) \longrightarrow (x_i, t_j)$,
- $x_i - x_{i-1} = \Delta_x$, $x_1 = a$, $x_M = b$,
- $t_n - t_{n-1} = \Delta_t$, $t_1 = 0$, $t_N = T$,
- $u_{i,n} = u(x_i, t_n)$

- thus

- $\Delta_x = \frac{b-a}{M-1}$ and $x_i = a + (i-1)\Delta_x$
- $\Delta_t = \frac{T}{N-1}$ and $t_n = (n-1)\Delta_t$



Problem

Approximate $\partial_{xx}u$ and $\partial_t u = 0$

Taylor Expansion

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2f''(x) + \frac{1}{6}h^3f'''(x) + \frac{1}{24}h^4f^{IV}(x) + O(h^5)$$

First order finite differences

$$\text{Forward: } \delta_{x,h}^+ f \equiv \frac{f(x+h) - f(x)}{h} = f'(x) + O(h)$$

$$\text{Backward: } \delta_{x,h}^- f \equiv \frac{f(x) - f(x-h)}{h} = f'(x) + O(h)$$

$$\text{Centered: } \delta_{x,h}^o f \equiv \frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2)$$

$$f(x+h) = f(x) + hf'(x) + h^2f''(x)/2 + h^3f'''(x)/6 + o(h^4)$$

$$f(x-h) = f(x) - hf'(x) + h^2f''(x)/2 - h^3f'''(x)/6 + o(h^4)$$



Second order finite differences

$$\text{Centered: } \delta_{xx,h}^o f \equiv \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = f''(x) + O(h^2)$$

Proof:

$$\begin{aligned} f(x+h) - f(x) &= +hf'(x) + \frac{1}{2}h^2f''(x) + \frac{1}{6}f'''(x) + O(h^4) \\ -f(x) + f(x-h) &= -hf'(x) + \frac{1}{2}h^2f''(x) - \frac{1}{6}f'''(x) + O(h^4) \\ \hline f(x+h) - 2f(x) + f(x-h) &= +h^2f''(x) + O(h^4) \end{aligned}$$

$$\text{Note that: } \delta_{xx,h}^o f = \frac{1}{h}(\delta_{x,h}^+ f - \delta_{x,h}^- f)$$



Outline

5 Binomial Model

6 Finite Differences

- Discretization
- **Explicit Method**
- Implicit Method
- The Black and Scholes PDE
- Bi-dimensional equations

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Explicit Method

Solve $\delta_{xx,h}^{\circ} u - \delta_{t,k}^{+} u = 0$ with initial condition $u(0, x) = \phi(x)$

- Approximate ∂_{xx} with $\delta_{xx,h}^{\circ}$ and ∂_t with $\delta_{t,k}^{+}$
- Using $h = \Delta_x$ and $k = \Delta_t$ the method can be computed on the grid points:

$$\delta_{xx,\Delta_x}^{\circ} u(x_i, t_n) = \frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{\Delta_x^2}$$

$$\delta_{t,\Delta_t}^{+} u(x_i, t_n) = \frac{u(x_i, t_{n+1}) - u(x_i, t_n)}{\Delta_t}$$

- or, equivalently,

$$\delta_{xx}^{\circ} u_{i,n} = \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{\Delta_x^2}$$

$$\delta_t^{+} u_{i,n} = \frac{u_{i,n+1} - u_{i,n}}{\Delta_t}$$

$$u_{i,n+1} = u_{i,n} + \frac{\Delta_t}{\Delta_x^2} (u_{i+1,n} - 2u_{i,n} + u_{i-1,n})$$

Border conditions

$$u_{i,n+1} = u_{i,n} + \frac{\Delta t}{\Delta x^2} (u_{i+1,n} - 2u_{i,n} + u_{i-1,n}) \quad (*)$$

- Eq. (*) is not defined for $i = 1$ or $i = M$ and for $n = 0$

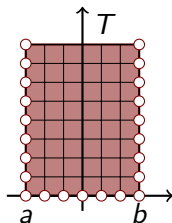
- Initial Conditions: $u_{i,1} = \phi(x_i)$, $i = 1, \dots, M$

- Border Conditions:

$$u_{1,n+1} = \phi_a(t_{n+1}),$$

$$u_{M,n+1} = \phi_b(t_{n+1}), \quad n = 1, \dots, N-1$$

- Note: (*) has to be computed in the correct order: Firstly, all the u in the second row ($n = 1$), then in the third ($n = 2$) and so on.



Exercise (Forward Euler FD method)

Using the explicit method approximate the Cauchy-Dirichlet problem

$$\begin{aligned} \partial_{xx} u - \partial_t u &= 0, & (x, t) &\in (-a, a) \times (0, T] & (*) \\ u(x, 0) &= (e^x - 1)^+, & x &\in [-a, a] \\ u(x, t) &= (e^x - 1)^+, & (x, t) &\in -a, a \times [0, T] \end{aligned}$$

and write a matlab function

function `U=heat(a,T,M,N)`

where $U_{i,n} \simeq u(x_i, t_n)$ and

$$\begin{aligned} x_i &= -a + (i-1)\Delta_x, & \Delta_x &= \frac{2a}{M-1}, & i &= 1, \dots, M \\ t_n &= (n-1)\Delta_t, & \Delta_t &= \frac{T}{N-1}, & n &= 1, \dots, N \end{aligned}$$

Test the function with $a = 1$, $T = 0.5$, $M = 21$, $N = 151, 101, 91$ and draw U as a surface (see `mesh` and `surf`)



heat.m

```
function V = heat( a,T, M,N )

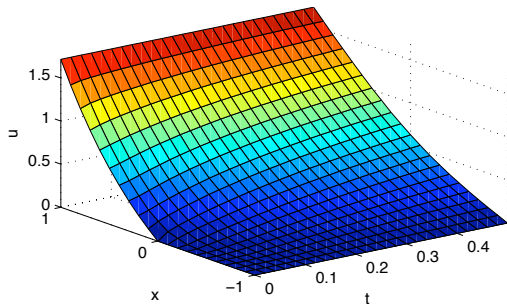
%% Initializations
dx = (2*a)/(M-1);      dt = T/(N-1);
V = zeros(M,N);
nu = dt/(dx*dx);

x = linspace(-a,a,M);
V(:,1) = max(exp(x)-1,0);

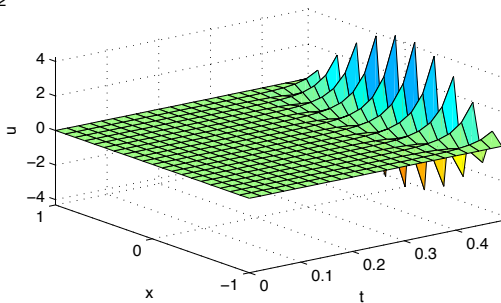
%% Compute V(:,2), ..., V(:,N)
for n=2:N
    V(1,n) = V(1,1);
    V(M,n) = V(M,1);
    V(2:M-1,n) = (1-2*nu)*V(2:M-1,n-1) + nu*V(1:M-2,n-1) ...
                + nu*V(3:M,n-1);
end
```



$$M=21, N=101, \Delta_t / \Delta_x^2 = 1/2$$



$$M=21, N=91, \Delta_t / \Delta_x^2 = .556$$



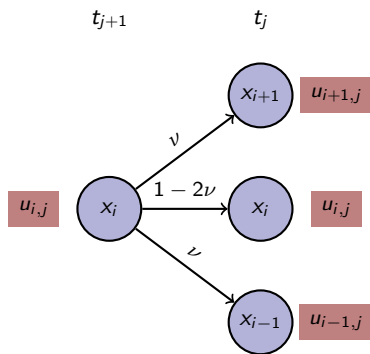
Finite differences - trinomial connection

Finite Differences ($\nu = \Delta_t / \Delta_x^2$) :

- $u_{i,n+1} = \nu u_{i-1,n} + (1 - 2\nu)u_{i,n} + \nu u_{i+1,n}$
- $x_i = a + (i - 1)\Delta_x$, $t_n = (n - 1)\Delta_t$
- Usually domain is a rectangle
- Border conditions can be imposed
- $\{\nu, 1 - 2\nu, \nu\}$ probability when $2\nu < 1$

Trinomial Tree:

- $v_{i,m-1} = q_1 v_{i-1,m} + q_2 v_{i,m} + q_3 v_{i+1,m}$
- Probability: $0 \leq q_i \leq 1$ e $\sum_i q_i = 1$
- Backward: $m = N + 1 - n$
- Domain is triangular



Consistency

What is the error at each step?

- Let U be the solution to the PDE $\partial_{xx} U - \partial_t U = 0$ with the appropriate border conditions
- $\delta_{xx}^o - \delta_t^+$ discrete operator
- Truncation error: $\mathcal{T}(x, t) = \delta_{xx}^o U(x, t) - \delta_t^+ U(x, t)$
- Facts: $\delta_{xx}^o U = \partial_{xx} U + O(\Delta_x^2)$ and $\delta_t^+ U = \partial_t U + O(\Delta_t)$
- Thus, $\mathcal{T}(x, t) = O(\Delta_t) + O(\Delta_x^2)$

Consistency: $\mathcal{T}(x, t) \rightarrow 0$ for $\Delta_t \rightarrow 0$ and $\Delta_x \rightarrow 0$

- But $\Delta_t \rightarrow 0$ implies $N \rightarrow \infty$: consistency is not sufficient, errors may accumulate



Stability

- U and u solve $(\partial_{xx} - \partial_t)U = 0$ and $(\delta_t^+ - \delta_{xx}^o)u = 0$, resp.
- Error: $e_{i,n} = u_{i,n} - U(x_i, t_n)$

Convergence: $e_{i,n} \rightarrow 0$ when $\Delta_t \rightarrow 0$ and $\Delta_x \rightarrow 0$.

- Recall $\mathcal{T} = (\delta_t^+ - \delta_{xx}^o)U(x_i, t_n) = O(\Delta_t + \Delta_x^2)$
- Thus

$$(\delta_t^+ - \delta_{xx}^o)e_{i,n} = (\delta_t^+ - \delta_{xx}^o)u_{i,n} + (\delta_t^+ - \delta_{xx}^o)U(x_i, t_n) = \mathcal{T}$$

$$\frac{e_{i,n+1} - e_{i,n}}{\Delta_t} = \frac{e_{i+1,n} - 2e_{i,n} + e_{i-1,n}}{\Delta_x^2} + \mathcal{T}$$

$$e_{i,n+1} = \nu e_{i-1,n} + (1 - 2\nu)e_{i,n} + \nu e_{i+1,n} + \Delta_t \mathcal{T}$$

$$|e_{i,n+1}| \leq \nu |e_{i-1,n}| + |(1 - 2\nu)e_{i,n}| + \nu |e_{i+1,n}| + \Delta_t |\mathcal{T}|$$



$$|e_{i,n+1}| \leq \nu |e_{i-1,n}| + |(1 - 2\nu)e_{i,n}| + \nu |e_{i-1,n}| + \Delta_t \mathcal{T}$$

- If $\nu \leq 1/2$ then $|(1 - 2\nu)e_{i,n}| = (1 - 2\nu)|e_{i,n}|$ so that

$$|e_{i,n+1}| \leq \nu |e_{i-1,n}| + (1 - 2\nu)|e_{i,n}| + \nu |e_{i-1,n}| + \Delta_t \mathcal{T}$$

- Define $E_n = \max_i |e_{i,n}|$, thus

$$|e_{i,n+1}| \leq \nu E_n + (1 - 2\nu)E_n + \nu E_n + \Delta_t \mathcal{T} = E_n + \Delta_t \mathcal{T}$$

$$E_{n+1} \leq E_n + \Delta_t \mathcal{T}$$

- Starting $E_0 = 0$, after n steps $E_n \leq n\Delta_t \mathcal{T}$
- Fixing the time t^* , $n = t^*/\Delta_t$ and

$$E_{t^*} \leq t^* \mathcal{T} = t^* O(\Delta_t + \Delta_x^2) \rightarrow 0 \quad \text{for } \Delta_t \rightarrow 0 \text{ and } \Delta_x \rightarrow 0$$

- Remark: Δ_t and Δ_x converge to 0 conditionally on $\Delta_t/\Delta_x^2 < 1/2$.



Matrix Form

The explicit method recurs:

$$u_{1,n+1} = u_{1,n} + b_1,$$

$$u_{i,n+1} = u_{i,n} + \nu u_{i-1,n} - 2\nu u_{i,n} + \nu u_{i+1,n},$$

$$u_{M,n+1} = u_{M,n} + b_M,$$

$$b_1 = (\phi_a(t_{n+1}) - \phi_a(t_n))$$

$$n = 2, \dots, M$$

$$b_M = (\phi_b(t_{n+1}) - \phi_b(t_n))$$

Example, for $M = 5$:

$$u_{1,n+1} = u_{1,n} + b_1$$

$$u_{2,n+1} = u_{2,n} + \nu u_{1,n} - 2\nu u_{2,n} + \nu u_{3,n}$$

$$u_{3,n+1} = u_{3,n} + \nu u_{2,n} - 2\nu u_{3,n} + \nu u_{4,n}$$

$$u_{4,n+1} = u_{4,n} + \nu u_{3,n} - 2\nu u_{4,n} + \nu u_{5,n}$$

$$u_{5,n+1} = u_{5,n} + b_5$$

In matrix form

$$\begin{pmatrix} u_{1,n+1} \\ u_{2,n+1} \\ u_{3,n+1} \\ u_{4,n+1} \\ u_{5,n+1} \end{pmatrix} = \begin{pmatrix} u_{1,n} \\ u_{2,n} \\ u_{3,n} \\ u_{4,n} \\ u_{5,n} \end{pmatrix} + \nu \begin{pmatrix} 0 & 0 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 0 & 0 \end{pmatrix} \begin{pmatrix} u_{1,n} \\ u_{2,n} \\ u_{3,n} \\ u_{4,n} \\ u_{5,n} \end{pmatrix} + \begin{pmatrix} b_1 \\ 0 \\ 0 \\ 0 \\ b_5 \end{pmatrix}$$



$$\begin{pmatrix} u_{1,n+1} \\ u_{2,n+1} \\ u_{3,n+1} \\ u_{4,n+1} \\ u_{5,n+1} \end{pmatrix} = \begin{pmatrix} u_{1,n} \\ u_{2,n} \\ u_{3,n} \\ u_{4,n} \\ u_{5,n} \end{pmatrix} + \nu \begin{pmatrix} 0 & 0 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 0 & 0 \end{pmatrix} \begin{pmatrix} u_{1,n} \\ u_{2,n} \\ u_{3,n} \\ u_{4,n} \\ u_{5,n} \end{pmatrix} + \begin{pmatrix} b_1 \\ 0 \\ 0 \\ 0 \\ b_5 \end{pmatrix}$$

with the appropriate definitions it can be written as

$$u_{n+1} = u_n + \nu \tilde{A} u_n + b = (I + \nu \tilde{A}) u_n + b$$

where

- $u_n : M \times 1$ contains the solution at time t_n
- $\tilde{A} : M \times M$ is a tridiagonal matrix
- $b : M \times 1$ contains the contribution from the border conditions
- $A = I + \nu \tilde{A}$ is the tridiagonal matrix:

$$A = \begin{pmatrix} 1 & 0 & & & \\ \nu & 1 - 2\nu & \nu & & \\ & \nu & 1 - 2\nu & \nu & \\ & & \nu & 1 - 2\nu & \nu \\ & & & 0 & 1 \end{pmatrix}$$



Outline

5 Binomial Model

6 Finite Differences

- Discretization
- Explicit Method
- **Implicit Method**
- The Black and Scholes PDE
- Bi-dimensional equations

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Implicit method

- $\partial_{xx}u - \partial_t u = 0$ is approximated by $\partial_{xx}^o u - \partial_t^- u = 0$
- Specifically

$$\frac{u(x, t) - u(x, t - \Delta_t)}{\Delta_t} - \frac{u(x + \Delta_x, t) - 2u(x, t) + u(x - \Delta_x, t)}{\Delta_x^2} = 0$$

- which on the grid points (x_i, t_n) can be written as

$$\frac{u_{i,n} - u_{i,n-1}}{\Delta_t} - \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{\Delta_x^2} = 0$$

- or

$$-\nu u_{i-1,n} + (1 + 2\nu)u_{i,n} - \nu u_{i+1,n} = u_{i,n-1}$$

where $\nu = \Delta_t / \Delta_x^2$



- Linear system with M equations and M unknowns $\{u_{i,n}, i = 1, \dots, M\}$

$$\begin{cases} u_{1,n} & = u_{1,n-1} + b_1 \\ -\nu u_{i-1,n} + (1 + 2\nu)u_{i,n} - \nu u_{i+1,n} & = u_{i,n-1} & i = 2, \dots, \\ u_{M,n} & = u_{M,n-1} + b_M \end{cases}$$

where $b_1 = \phi_a(t_n) - \phi_a(t_{n-1})$ and $b_M = \phi(t_n) - \phi(t_{n-1})$

- or $Au_n = u_{n-1} + b$
- A is tridiagonal
- Solving an $M \times M$ linear system requires $O(M^3)$
- Solving Tridiagonal systems requires $O(M)$ (see Golub and Van Loan)



- Example ($M = 5$):

$$\begin{array}{rcccccc}
 u_{1,n} & & & & & & = u_{1,n-1} + \\
 u_{2,n} & -\nu u_{1,n} & +2\nu u_{2,n} & -\nu u_{3,n} & & & = u_{2,n-1} \\
 u_{3,n} & & -\nu u_{2,n} & +2\nu u_{3,n} & -\nu u_{4,n} & & = u_{3,n-1} \\
 u_{4,n} & & & -\nu u_{3,n} & +2\nu u_{4,n} & -\nu u_{5,n} & = u_{4,n-1} \\
 u_{5,n} & & & & & & = u_{5,n-1} +
 \end{array}$$

- $u_n - \nu \tilde{A} u_n = u_{n-1} + b$ where

$$\tilde{A} = \begin{pmatrix} 0 & 0 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & 0 & 0 \end{pmatrix}$$

- or: $Au_j = u_{j-1} + b$, where $A = I - \nu \hat{A}$
- Implicit method:
 - 1 Put initial conditions in u_1
 - 2 For $j = 2, \dots, N$ solve $Au_j = u_{j-1} + b$



Stability of implicit method

- Truncation error: $\mathcal{T} = O(\Delta_x^2 + \Delta_t)$
- The error satisfy

$$(1 + 2\nu)e_{i,n} = e_{i,n-1} + \nu e_{i-1,n} + \nu e_{i+1,n} + \Delta_t \mathcal{T}$$

$$(1 + 2\nu)|e_{i,n}| \leq |e_{i,n-1}| + \nu|e_{i-1,n}| + \nu|e_{i+1,n}| + \Delta_t \mathcal{T},$$

setting $E_n = \max_i e_{i,n}$,

$$(1 + 2\nu)E_j \leq E_{n-1} + \nu E_{n-1} + \nu E_{n-1} + \Delta_t \mathcal{T}$$

$$E_n \leq E_{n-1} + \mathcal{T}$$

$$E_n \leq E_1 + (n - 1)\Delta_t \mathcal{T}$$

- For a fixed time t^* ($n - 1 = t^*/\Delta_t$) and for exact initial conditions

$$E_n \leq t^* \mathcal{T} \rightarrow 0$$

- No restriction on Δ_t and Δ_x : the method is **unconditionally stable**



```
function V = heatbw( a,T, M,N )

%% Initializations:
dx = (2*a)/(M-1);      dt = T/(N-1);
V = zeros(M,N);
nu = dt/(dx*dx);

x = linspace(-a,a,M);
V(:,1) = max(exp(x)-1,0);

A = -diag(ones(M-1,1),1) + 2*eye(M) - diag(ones(M-1,1),-1);
A(1,:) = 0;
A(end,:) = 0;

A = eye(M) - nu*A;

%% Compute V(:,2), ..., V(:,N)
for n=2:N
    V(:,n) = A\V(:,n-1);
end
```

```
function v = heatbw2( a,T, M,N )

% Initializations
dx = (2*a)/(M-1);      dt = T/(N-1);
nu = dt/(dx*dx);

x = linspace(-a,a,M);
v = max(exp(x)-1,0);

A = -diag(ones(M-1,1),1) + 2*eye(M) - diag(ones(M-1,1),-1);
A(1,:) = 0;
A(end,:) = 0;

A = eye(M) - nu*A;

% Compute V(:,2), ..., V(:,N)
for n=2:N
    v = A\v;
end
```

Border conditions

- Until now, only explicit bc have been used: $u(x, t) = \phi(t)$
- Let consider now the bc $\partial_x u(x, t) = c$ on the left border $x = a$
- At the border it holds

$$\begin{aligned}\partial_{xx}u(a, t) &= \frac{\partial_x u(a + \Delta_x, t) - \partial_x u(a, t)}{\Delta_x} + O(\Delta_x) \\ &= \frac{u(a + \Delta_x, t) - u(a, t)}{\Delta_x^2} - \frac{c}{\Delta_x} + O(\Delta_x)\end{aligned}$$

- thus $\partial_t u - \partial_{xx}u$ can be approximated at (a, t_j) as

$$u_{1,j+1} = u_{1,j} + \nu u_{2,j} - \nu u_{1,j} - \frac{\Delta_t}{\Delta_x} c \quad \text{explicit method}$$

$$u_{1,j} - \nu u_{2,j} + \nu u_{1,j} = u_{1,j-1} - \frac{\Delta_t}{\Delta_x} c \quad \text{implicit method}$$



Outline

5 Binomial Model

6 Finite Differences

- Discretization
- Explicit Method
- Implicit Method
- **The Black and Scholes PDE**
- Bi-dimensional equations

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Black and Scholes

Black & Scholes: the price of an European option solves

$$\begin{cases} \partial_t V + \frac{\sigma^2}{2} S^2 \partial_{SS} V + (r - q) S \partial_S V - rV = 0 \\ V(S, T) = \phi(S) \end{cases}$$

- t time
- T expiry dell'opzione
- S value of the underlying
- σ volatility
- r risk-free interest rate
- q dividend rate
- $\phi(S)$ option payoff $\phi(S) = \begin{cases} \max(S - K, 0), & \text{call} \\ \max(K - S, 0), & \text{put} \end{cases}$
- K exercise price or strike



B&S and Change of Variables

$$\partial_t V + \frac{\sigma^2}{2} S^2 \partial_{SS} V + (r - q) S \partial_S V - rV = 0$$

First approach: $r = q = 0$

- $x = \log(S)$,
- $\tau = T - t$,
- $V(S, t) = u(\log(S), T - t)$
- $\partial_t V = -\partial_\tau u$, $\partial_S V = \frac{1}{S} \partial_x u$, $\partial_{SS} V = \frac{1}{S^2} (\partial_{xx} u - \partial_x u)$

$$\partial_\tau u - \frac{\sigma^2}{2} (\partial_{xx} u - \partial_x u) = 0$$

- Work even when σ is non-constant



B&S and Change of Variables

$$\partial_t V + \frac{\sigma^2}{2} S^2 \partial_{SS} V + (r - q) S \partial_S V - rV = 0$$

Second approach ($r = q = 0$)

- $x = \log(S) - \frac{\sigma^2}{2}(T - t)$,
- $\tau = T - t$,
- $V(S, t) = u(x, \tau)$
- $\partial_t V = -\partial_\tau u + \frac{\sigma^2}{2} \partial_x u$, $\partial_{SS} V = \frac{1}{S^2} (\partial_{xx} u - \partial_x u)$

$$\partial_\tau u - \frac{\sigma^2}{2} \partial_{xx} u = 0$$



Black and Scholes: Change of variables

$$\partial_t V + \frac{\sigma^2}{2} S^2 \partial_{SS} V + (r - q) S \partial_S V - rV = 0$$

- General case: $r \neq 0$ e $q \neq 0$

- $x = \log(S) + (r - q - \frac{\sigma^2}{2})(T - t)$, $\tau = T - t$, $V(S, t) = e^{-r\tau} u(x, \tau)$

- $\partial_t V = e^{-r\tau} (-\partial_\tau u - (r - q - \frac{\sigma^2}{2}) \partial_x u) + r e^{-r\tau} u$,

- $S \partial_S V = e^{-r\tau} (\partial_x u)$, $S^2 \partial_{SS} V = e^{-r\tau} (\partial_{xx} u - \partial_x u)$

$$\Rightarrow \partial_\tau u - \frac{\sigma^2}{2} \partial_{xx} u = 0$$

- Backward/forward prices:

$$x = \log(e^{(r-q)(T-t)} S) + \text{adj}, \quad V = e^{-r(T-t)} u$$

- Homogeneity: using $\log(S/K)$ give same results
- Exercise: by solving a single Cauchy problem compute prices of a set of call options with different exercise prices and different maturities.



Outline

5 Binomial Model

6 Finite Differences

- Discretization
- Explicit Method
- Implicit Method
- The Black and Scholes PDE
- Bi-dimensional equations

7 Monte Carlo

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Two dimensional heat equation

- 2D: $\partial_t u - \partial_{xx} u - \partial_{yy} u = 0$
- Domain: $[x_a, x_b] \times [y_a, y_b] \times [0, T]$

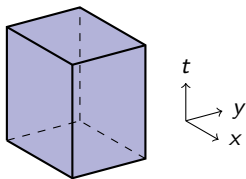
- Discretization:

$$x_i = x_a + (i - 1)\Delta_x, \quad i = 1, \dots, I,$$

$$y_j = y_a + (j - 1)\Delta_y, \quad j = 1, \dots, J,$$

$$t_n = (n - 1)\Delta_t, \quad n = 1, \dots, N,$$

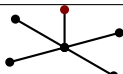
$$u_{ij}^n = u(x_i, y_j, t_n)$$



- Numerical methods ($\nu = \Delta_t / \Delta_x^2$ and $\Delta_x = \Delta_y$):

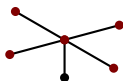
Explicit

$$u_{ij}^{n+1} - u_{ij}^n = \nu(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n)$$



Implicit:

$$u_{ij}^n - u_{ij}^{n-1} = \nu(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n)$$



2D Heat: matlab implementation

- u_{ij}^n needs an $I \times J \times N$ array
- otherwise, only the current step has to be stored: $U(i,j) = u_{ij}^n$
- The algorithm core is:

```
heat2da.m
for n=2:N
    for i=2:I-1
        for j=2:J-1
            Unew(i,j) = (1-4*nu)*U(i,j) + ...
                nu*( U(i-1,j)+U(i+1,j)+U(i,j-1)+U(i,j+1) );
        end
    end
    %% Add border conditions

    U = Unew;
end
```

Exercise

Approximate numerically the Cauchy problem

$$\begin{cases} \partial_t u - \partial_{xx} u - \partial_{yy} u = 0 & \text{in } \Omega \times [0, T] \\ u(x, y, 0) = \max(x + y - 2, 0), & \text{per } (x, y) \in \Omega \\ u(x, y, t) = \max(x + y - 2, 0), & \text{per } (x, y) \in \partial\Omega \text{ e } t \in [0, T] \end{cases}$$

where $\Omega = [-a, a] \times [-a, a]$.

Remark

The stability condition is now $\Delta_t \leq \frac{1}{4} \Delta_x^2$



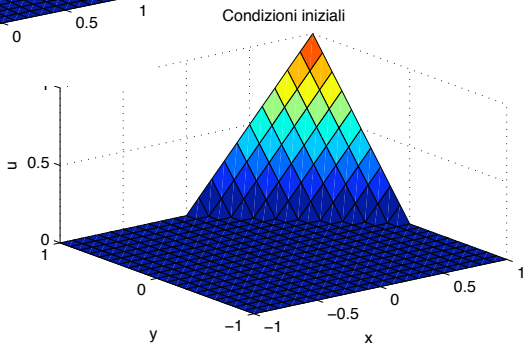
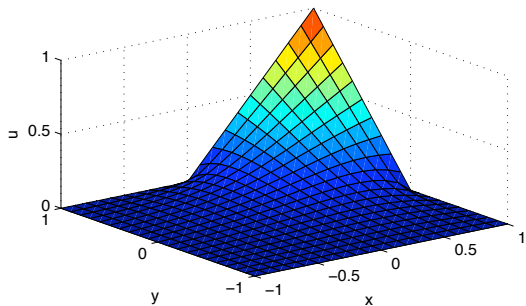
```
function U = heat2d(a,T, I, N)

dx = 2*a/(I-1);    dt = T/(N-1);    ni = dt/(dx*dx);

x = linspace(-a,a,I);
[X,Y] = meshgrid(x,x);
U = max(X+Y-1,0);
Unew = U;

for n=2:N
    for i=2:I-1
        for j=2:I-1
            Unew(i,j) = (1-4*ni)*U(i,j) + ...
                ni * (U(i-1,j)+U(i+1,j)+U(i,j-1)+U(i,j+1));
        end
    end
    U = Unew;
end
```


$T=0.5, M=21, N=201, \Delta_t / \Delta_x^2 = 1/4$



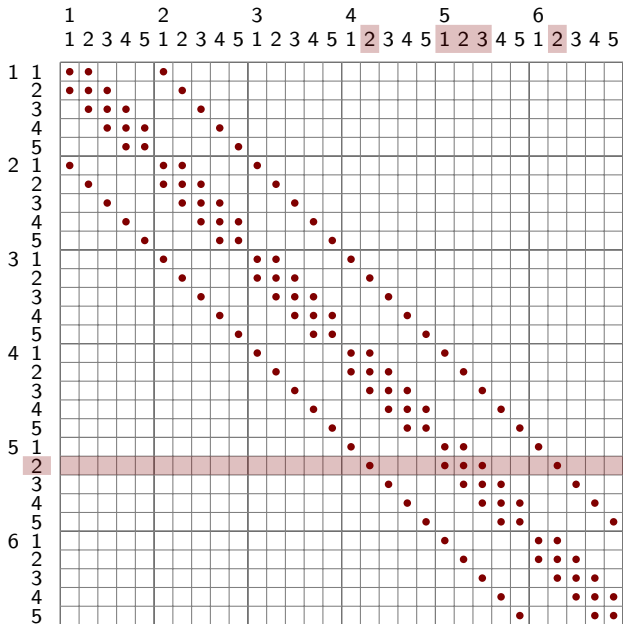
2d heat: implicit method

- Problem: rewrite in matrix form the following system

$$u_{ij}^n - u_{ij}^{n-1} = \nu(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{ij}^n)$$

- The $I \times J$ matrix u_{ij} ($i = 1, \dots, I, j = 1, \dots, J$) needs to be transformed into a $IJ \times 1$ vector





Exercise

Approximate and numerically solve the Cauchy problem

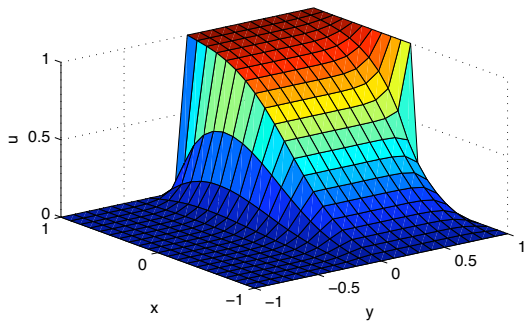
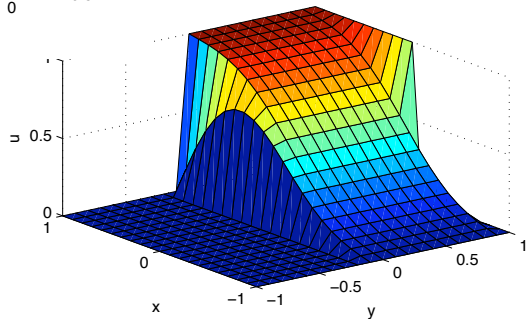
$$\begin{cases} \partial_t u - \partial_{xx} u - \partial_y u = 0 & \text{on } \Omega \times [0, T] \\ u(x, y, 0) = 1 & \text{for } x \geq 0 \text{ and } y \geq 0 \\ u(x, y, 0) = 0 & \text{for } x < 0 \text{ or } y < 0 \\ u(x, y, t) = u(x, y, 0) & \text{for } (x, y) \in \partial\Omega \text{ and } t \in [0, T] \end{cases}$$

where $\Omega = [-a, a] \times [-a, a]$.

Remark

The terms $\partial_t u - \partial_y u$ can be approximated:

- Independently: $\partial_t u \simeq \delta_t^\pm u$ and $\partial_y u \simeq \delta_y^\pm u$
In that case the direction of the discretization is critical for stability.
- Jointly:
let $f(s) = u(x, y - s, t + s)$ then $f'(0) = \partial_t u(x, y, t) - \partial_y u(x, y, t)$,
so that $\partial_t u - \partial_y u \simeq \Delta_t^{-1}(u(x, y - \Delta_t, t + \Delta_t) - u(x, y, t))$

Primo approccio ($T=0.1, l=21, N=201$)Secondo approccio ($T=0.1, l=21, N=31$)

Outline

5 Binomial Model

6 Finite Differences

7 Monte Carlo

- Numerical Integration
- Monte Carlo Method
- Generating Random Variables
- Variance Reduction Techniques
- SDE Integration
- American Options

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Outline

5 Binomial Model

6 Finite Differences

7 Monte Carlo

- Numerical Integration
- Monte Carlo Method
- Generating Random Variables
- Variance Reduction Techniques
- SDE Integration
- American Options

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Expected Values and Numerical Integration

Problem

Compute the expected value

$$E[f(X)], \quad \text{where } X \sim P$$

or equivalently

$$\int_{\Omega} f(x) dP(x)$$

Can be computed

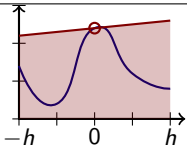
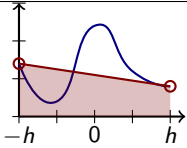
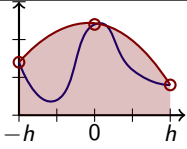
- explicitly
- numerically
- numerically approximated



Problem

Approximate $I(h) = \int_{-h}^h f(x) = \tilde{I}(h) + E$

Newton-Côtes approximation methods: polynomial interpolation of $f(x)$

| Method | | Approximation | Bound $ E $ |
|-------------|---|-------------------------------------|---------------------------------|
| Mid Point |  | $2hf(0)$ | $\frac{1}{3}h^3 \ f''\ _\infty$ |
| Trapezoidal |  | $h(f(-h) + f(h))$ | $\frac{2}{3}h^3 \ f''\ _\infty$ |
| Simpson |  | $\frac{h}{3}(f(-h) + 4f(0) + f(h))$ | $O(h^5 \ f^{(4)}\ _\infty)$ |



- Newton-Cotes: good for small h , but for large h ?
- Solution:

Uniformly partition the domain:

$$[a, b] = [x_0, x_1] \cup [x_1, x_2] \cup \dots \cup [x_{M-1}, x_M],$$

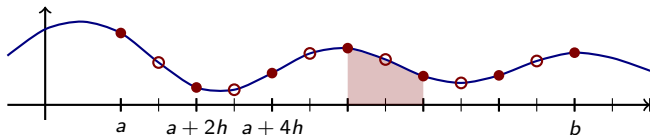
where $x_i = a + ih$, $h = (b - a)/M$

Then, apply a Newton Cotes integration to each interval

- Notice,
 - there may be more sampling points than intervals (e.g. Simpson)
 - the points may have different weights

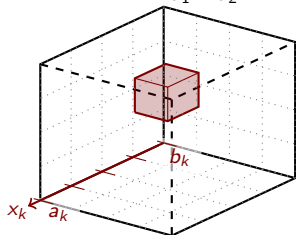
$$\int_a^b f(x) = \sum_i \int_{x_{2i}}^{x_{2i+2}} f(x) = \sum_i w_i f_i + o(h^\alpha),$$

$$x_i = a + ih, \\ h = (b - a)/M$$



Integration in \mathbb{R}^n

- Compute $\int_{a_1}^{b_1} \int_{a_2}^{b_2} \dots \int_{a_n}^{b_n} f(\mathbf{x}) d\mathbf{x}$, with $\mathbf{x} = (x_1 \ x_2 \ \dots \ x_n)$



- Partition each $[a_k, b_k]$ into M intervals:
 $\Rightarrow \sum_{i_1} \sum_{i_2} \dots \sum_{i_n} w_I f(\mathbf{x}_I), \quad I = (i_1, \dots, i_n)$.
- The domains is partitioned into M^n cubes
- Need to compute f for $O(M^n)$ points

- Precision: $O(M^{-\alpha})$
- Computing time: $O(M^n)$
- The precision is $O(t_c^{-\alpha/n})$ for given computing time t_c
- The computational complexity is $O(e^{np/\alpha})$ for a given precision 10^{-p}



Mid point rule

By approximating $f(x)$ with a constant $p(x) = f(0)$, it results

$$f(x) = p(x) + \epsilon, \quad \epsilon = x f'(0) + \frac{1}{2}x^2 f''(\xi)$$

$$I(h) = 2h f(0) + E, \quad |E| \leq \frac{1}{3}h^3 \|f''\|_\infty$$



Trapezoidal rule

Linearly approximate $f(x)$: $p(x) = \frac{h-x}{2h}f(-h) + \frac{h+x}{2h}f(h)$:

$$f(x) = p(x) + \epsilon, \quad \epsilon = \frac{h^2 - x^2}{2}x^2 f''(\xi)$$

$$I(h) = h(f(-h) + f(h)) + E, \quad |E| \leq \frac{2}{3}h^3 \|f''\|_\infty$$

Proof.

By Taylor:

$$f(-h) = f(x) - (h+x)f'(x) + \epsilon_1, \quad \epsilon_1 = \frac{1}{2}(h+x)^2 f''(\xi_1)$$

$$f(h) = f(x) + (h-x)f'(x) + \epsilon_2, \quad \epsilon_2 = \frac{1}{2}(h-x)^2 f''(\xi_2)$$

so that

$$p(x) := \frac{h-x}{2h}f(-h) + \frac{h+x}{2h}f(h) = f(x) + 0 \cdot f'(x) + \epsilon,$$

$$\epsilon = \frac{h^2 - x^2}{2} \left(\frac{h+x}{2h}f''(\xi_1) + \frac{h-x}{2h}f''(\xi_2) \right) = \frac{h^2 - x^2}{2} f''(\xi), \quad \xi \in [-h, h].$$

Furthermore $|E| = \frac{1}{2} \left| \int_{-h}^h (h^2 - x^2) f''(\xi) dx \right| \leq \frac{2}{3} h^3 \|f''\|_\infty$ □

Let consider now the Taylor expansion of the $I(h) := \int_{-h}^h f(x)$ near $h = 0$

$$I(h) = I(0) + hI'(0) + \frac{1}{2}h^2I''(0) + \frac{1}{3!}h^3I^{(3)}(0) + \dots + \frac{1}{n!}h^nI^{(n)}(0) + R_n$$

where $R_n = \frac{1}{(n+1)!}h^{n+1}I^{(n+1)}(\xi)$, for $\xi \in [0, h]$,

In order The derivatives of $I(h)$ are given by

| | |
|---|--|
| | $I(0) = 0$ |
| $I'(h) = f(h) + f(-h)$ | $I'(0) = 2f(0)$ |
| $I''(h) = f'(h) - f'(-h)$ | $I''(0) = 0$ |
| $I^{(3)}(h) = f''(h) + f''(-h)$ | $I^{(3)}(0) = 2f''(0)$ |
| $I^{(4)}(h) = f^{(3)}(h) - f^{(3)}(-h)$ | $I^{(4)}(0) = 0$ |
| \vdots | \vdots |
| $I^{(k)}(h) = f^{(k-1)}(h) + f^{(k-1)}(-h)$ | $I^{(k)}(0) = 2f^{(k-1)}(0)$ k dispari |
| $I^{(k)}(h) = f^{(k-1)}(h) - f^{(k-1)}(-h)$ | $I^{(k)}(0) = 0$ k pari |



So that the Taylor expansion of $I(h) = \int_{-h}^h f(x)$ becomes

$$\begin{aligned} I(h) &= hf'(0) + \frac{h^3}{3!} f^{(3)}(0) + \frac{h^5}{5!} f^{(5)}(0) + \dots + \frac{h^{2k+1}}{(2k+1)!} f^{(2k+1)}(0) + R_{2k} \\ &= 2hf(0) + \frac{h^3}{3} f''(0) + \frac{2h^5}{5!} f^{(4)}(0) + \dots + \frac{2h^{2k+1}}{(2k+1)!} f^{(2k)}(0) + R_{2k} \end{aligned}$$

where $R_{2k+1} = \frac{2h^{2k+1}}{(2k+1)!} f^{(2k)}(\xi)$ and $\xi \in [0, h]$.

$k = 1$ Mid point rule: $I(h) = 2hf(0) + \frac{h^3}{3} f''(\xi)$

$k = 2$ Simpson rule: $I(h) = 2hf(0) + \frac{h^3}{3} f''(0) + \frac{2h^5}{5!} f^{(4)}(\xi)$

- Need $f''(0)$
- Can be approximated by differences



Simpson rule

$$I(h) = 2hf(0) + \frac{h^3}{3}f''(0) + O(h^5\|f^{(4)}\|_\infty) \quad (1)$$

From Taylor:

$$f(h) = f(0) + hf'(0) + \frac{h^2}{2}f''(0) + \frac{h^3}{3!}f^{(3)}(0) + O(h^4\|f^{(4)}\|_\infty)$$

$$f(-h) = f(0) - hf'(0) + \frac{h^2}{2}f''(0) - \frac{h^3}{3!}f^{(3)}(0) + O(h^4\|f^{(4)}\|_\infty)$$

whose sum gives

$$h^2f''(0) = f(-h) - 2f(0) + f(h) + O(h^4\|f^{(4)}\|_\infty) \quad (2)$$

By (1) and (2):

$$I(h) = \frac{h}{3} \left(f(-h) + 4f(0) + f(h) \right) + O(h^5\|f^{(4)}\|_\infty)$$



Trapezoidal Rule

Approximate $f(x)$ by a linear function going through $f(-h)$ e $f(h)$

$$g(x) = \alpha f(-h) + (1 - \alpha)f(h), \quad \alpha = \frac{h - x}{2h} \quad (3)$$

A Taylor expansion of $f(h)$ e $f(-h)$ on $h = 0$ gives

$$(h+x) \cdot f(h) = f(x) + (h-x)f'(x) + \frac{1}{2}(h-x)^2 f''(x) + O((h-x)^3 \|f'''\|_\infty)$$

$$(h-x) \cdot f(-h) = f(x) - (h+x)f'(x) + \frac{1}{2}(h+x)^2 f''(x) + O((h+x)^3 \|f'''\|_\infty)$$

whose sum gives

$$2h g(x) = 2h f(x) + 0f'(x) + \frac{1}{2}(h^2 - x^2)2h f''(x) + O(h^4 \|f'''\|_\infty)$$

$$g(x) = f(x) + \frac{1}{2}(h^2 - x^2)f''(x) + O(h^3 \|f'''\|_\infty)$$



$$g(x) = f(x) + \frac{1}{2}(h^2 - x^2)f''(x) + O(h^3\|f'''\|_\infty)$$

rearranging and integrating

$$I(h) = \int_{-h}^h g(x)dx + O(\|f'''\|_\infty) \int_{-h}^h (h^2 - x^2)dx + \text{smaller terms}$$

and, from

$$\int_{-h}^h g(x)dx = h(f(-h) + f(h)) \quad \text{and} \quad \int_{-h}^h (h^2 - x^2)dx = 2h^3 - \frac{2}{3}h^3 = \frac{4}{3}h^3$$

it follows that

$$I(h) = h\left(f(h) + f(-h)\right) + O(h^3\|f'''\|_\infty)$$



Outline

5 Binomial Model

6 Finite Differences

7 Monte Carlo

- Numerical Integration
- **Monte Carlo Method**
- Generating Random Variables
- Variance Reduction Techniques
- SDE Integration
- American Options

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Monte Carlo Method

Problem

Given

- $X : \Omega \rightarrow \mathbb{R}^N$ r.v. on a probability space (Ω, \mathcal{F}, P) and
- $g : \mathbb{R}^N \rightarrow \mathbb{R}$ such that $\text{Var}[g(X)] = \sigma^2 < \infty$

Compute $\theta = E[g(X)] = \int_{\Omega} g(X) dP$

Monte Carlo Method

The Monte Carlo estimate of θ is given by

$$\hat{\theta}_m = \frac{1}{m} \sum_{i=1}^m g(x_i), \quad (4)$$

where x_1, x_2, \dots, x_m are m independent occurrences of X

Monte Carlo

$$\hat{\theta}_m = \frac{1}{m} \sum_{i=1}^m g(X_i), \quad X_i \sim X \text{ and independent} \quad (5)$$

Unbiased: $E[\hat{\theta}_m] = \frac{1}{m} \sum_i E[g(X_i)] = E[g(X)] = \theta$

Efficiency: $\text{Var}[\hat{\theta}_m] = \frac{1}{m^2} \sum_i \text{Var}[g(X_i)] = \frac{1}{m} \text{Var}[g(X)] = \frac{1}{m} \sigma^2$

The error $\hat{\theta}_m - \theta$ can be bounded by

- Chebichev Theorem: $P[|\hat{\theta}_m - \theta| \geq \varepsilon] \leq \frac{\sigma}{\varepsilon^2 m}$

!check!

- Central Limit Theorem: $\frac{\hat{\theta}_m - \theta}{\sigma/\sqrt{m}} \xrightarrow{d} N(0, 1)$

When unknown σ^2 can be estimated as $S^2 = \frac{1}{m-1} \sum_i (g(x_i) - \hat{\theta}_m)^2$



Pro and Cons

- Pro: Require $g \in \mathcal{L}^2$ not necessarily $g \in \mathcal{C}$
- Pro: Error of order $m^{-1/2}$, not dep. on the domain dimension
- Cons: probabilistic bound, the worst case error is ∞
- Cons: the estimates depends on the generated random sequence
 - initial guess
 - “randomness” of the sequence



Outline

5 Binomial Model

6 Finite Differences

7 Monte Carlo

- Numerical Integration
- Monte Carlo Method
- **Generating Random Variables**
- Variance Reduction Techniques
- SDE Integration
- American Options

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Low-discrepancy sequences

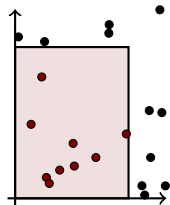
- Consider $P = \text{unif}([0, 1]^N)$.
- Let $Q \subset \mathbb{R}^N$ be a N -dimensional rectangle parallel to the axes and with a vertex on the origin
- Discrepancy: $D_m^* \equiv \sup_Q \left| \frac{\#\{x_i \in Q\}}{m} - P[Q] \right|$
- Low discrepancy: uniformly distributed points
- Koksma-Hlawka:

$$|\theta_m - \theta| \leq V(g)D_m^*, \quad (6)$$

where $V(g) = \sup_s \sum_i |f(x_{i+1}) - f(x_i)|$,
 over all the sequences $s = \{0 \equiv x_1 < \dots < x_n \equiv 1\}$.

- Conjecture: $D_m^* \leq C \log(m)^N / m$
- Optimal Sequences:

Faure, Halton, Hammersley, Sobol, Niederreiter and Van der Corput.



Simulating a R.V. from a uniformly distributed R.V.

Lemma

Let $X \sim D$ (with cdf D) and let $U \sim \text{Unif}(0, 1)$,
if $Y = D^{-1}(U)$ then $X \sim Y$ equal in distribution

Proof.

$$P[Y < y] = P[D^{-1}(U) < y] = P[U < D(y)] = \int_0^{D(y)} dx = D(y)$$

since $0 \leq D(y) \leq 1$



- Work even if D is not continuous
- Often D is difficult to invert



Outline

5 Binomial Model

6 Finite Differences

7 Monte Carlo

- Numerical Integration
- Monte Carlo Method
- Generating Random Variables
- **Variance Reduction Techniques**
- SDE Integration
- American Options

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Antithetic Variables

Rationale

- 1 $\text{Var}[Y_1 + Y_2] = \text{Var}[Y_1] + \text{Var}[Y_2] + 2 \text{Cov}[Y_1, Y_2]$
- 2 g monotone then $\text{Cov}[g(X), g(-X)] < 0$

Example

Want to compute $\theta = E[g(X)]$ where $X \sim N(0, \sigma^2)$ and g is increasing

- Generate $X_1, \dots, X_{m/2} \sim N(0, \omega^2)$ i.i.d.
- Set $X_{m/2+i} = -X_i$ for $i = 1, \dots, m/2$
- Estimate $\hat{\theta}_m = \frac{1}{m} \sum_{i=1}^m g(X_i)$

Properties:

- $\hat{\theta}_m$ is unbiased
- $\text{Var}[\hat{\theta}_m] = \frac{1}{m} \text{Var}[g(X)] + \frac{1}{m} \text{Cov}[g(X), g(-X)] \leq \frac{1}{m} \text{Var}[g(X)]$

Lemma

When g and h are increasing then $E[g(X)h(X)] \geq E[g(X)]E[h(X)]$

Proof.

For any x, y : $(g(x) - g(y))(h(x) - h(y)) \geq 0$, thus

for X, Y i.i.d.: $(g(X) - g(Y))(h(X) - h(Y)) \geq 0$ and

$$E[(g(X) - g(Y))(h(X) - h(Y))] \geq 0$$

$$E[g(X)h(X)] + E[g(Y)h(Y)] \geq E[g(Y)h(X)] + E[g(X)h(Y)]$$

$2E[g(X)h(X)] \geq 2E[g(Y)h(X)]$ and from the independence of X and Y ,

$$E[g(X)h(X)] \geq E[g(X)]E[h(X)]$$



Setting $h(x) = -g(-x)$ and using the property $\text{Cov}[g, h] = E[gh] - E[g]E[h]$ gives the following corollary.

Corollary

If g is monotone then $\text{Cov}[g(X), g(-X)] < 0$

Monte Carlo with Antithetic Variables (MC-AV)

- The Antithetic Variable method is a variance reduction technique
- Consists on generating the opposite samples X_i and $-X_i$.
- $\hat{\theta}_m$ is unbiased when X **symmetrically distributed** (with 0 mean)

But, when X is not symmetrically distributed?

- Let $X \sim P$ and g monotone
- Estimate the expected value of $h(U) = g(P^{-1}(U))$, $U \sim \text{Unif}(0, 1)$
- Generate opposite samples U_i and $1 - U_i$
- The MC-AV estimate is unbiased: $E[h(1 - U)] = E[h(U)] = E[g(X)]$
- When g is monotone so is h and $\text{Cov}[h(U), h(1 - U)] < 0$.
- Analogously: simulate $X_i^+ = P^{-1}(U_i)$ and $X_i^- = P^{-1}(1 - U_i)$



Exercise

Compute the price $C_{0,T}$ at time 0 of an European Call expiring at time T :

$$C_{0,T} = e^{-rT} E[(S_T - K)^+]$$

where $S_T = S_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}Z\right)$, $Z \sim N(0, 1)$.



Control Variates

Consider the couple of random variables (X, Y)


- Known how to simulate (X, Y)
 - $a = E[X]$, $\text{Var}[X] = \sigma_X^2$ and $\text{Cov}[X, Y] = \sigma_{XY}$ are known
- ⇒ Want to compute $b = E[Y]$

Monte Carlo:

- Simulate by MC $(x_i, y_i) \sim (X, Y)$ i.i.d.
- $\hat{a} = \frac{1}{m} \sum_i x_i$ $\hat{b} = \frac{1}{m} \sum_i y_i$ $\text{Cov}[\hat{a}, \hat{b}] = \frac{1}{m} \sigma_{XY}$

The error $\hat{a} - a$ is known, use it to correct \hat{b} : $\hat{b}_{CV} = \hat{b} - \beta(\hat{a} - a)$

- $E[\hat{b}_{CV}] = b$,
- $\text{Var}[\hat{b}_{CV}] = \frac{1}{m} \sigma_Y^2 + \beta^2 \frac{1}{m} \sigma_X^2 - 2\beta \frac{1}{m} \sigma_{XY}$
- Variance is minimized by $\beta = \frac{\sigma_{XY}}{\sigma_X^2}$:

$$\text{Var}[\hat{b}_{CV}] = \text{Var}[\hat{b}] - \frac{1}{m} \frac{\sigma_{XY}^2}{\sigma_X^2}$$


$$\hat{b}_{CV} = \hat{b} - \beta(\hat{a} - a), \quad \beta = \frac{\sigma_{XY}}{\sigma_X^2}, \quad \text{Var}[\hat{b}_{CV}] = \frac{1}{m} \left(\sigma_Y^2 - \frac{\sigma_{XY}^2}{\sigma_X^2} \right)$$

- Only correlation matters:

$$\frac{\text{Var}[\hat{b}_{CV}]}{\text{Var}[\hat{b}]} = \left(1 - \frac{\sigma_{XY}^2}{\sigma_X^2 \sigma_Y^2} \right) = 1 - \rho_{XY}^2$$

- Usually, σ_X^2 and σ_{XY} are unknown
- Solution: estimate σ_X^2 and σ_{XY} from the generated sample:
 - $\hat{\sigma}_X^2 = \frac{1}{m-1} \sum_i (x_i - \hat{a})^2$
 - $\hat{\sigma}_{XY} = \frac{1}{m-1} \sum_i (x_i - \hat{a})(y_i - \hat{b})$
- β is no longer constant but depends on the sample $\{(x_i, y_i), i = 1, \dots, m\}$.
- \hat{b}_{CV} is unbiased only asymptotically.



Example (Asian Options)

- Dynamics: $dS_t = rS_t dt + \sigma S_t dW_t$ Geometric Brownian Motion
- n observation dates: $t_i = i\Delta t$, $\Delta t = T/n$

| | Average | Fixed Strike | Floating Strike |
|------------|--|--------------|-----------------|
| Arithmetic | $A = \frac{1}{n} \sum_{i=1}^n S_{t_i}$ | $(A - K)^+$ | $(S_T - A)^+$ |
| Geometric | $G = \left(\prod_{i=1}^n S_{t_i} \right)^{\frac{1}{n}}$ | $(G - K)^+$ | $(S_T - G)^+$ |

- Closed form expression for the fixed strike geometric average call:

$$C^G(S_0, K, r, T, \sigma) = C^{BS}(S_0, K, r, \tilde{T}, \tilde{\sigma}, \tilde{d}), \quad (7)$$

$$\text{where } \tilde{T} = \frac{n+1}{2n} T, \quad \tilde{\sigma}^2 = \frac{2n+1}{3n} \sigma^2, \quad \tilde{d} = \frac{n-1}{6n} \sigma^2.$$

- Known how to simulate $X = (G_T - K)^+$ and $Y = (A_T - K)^+$
Known $C^G = e^{-rT} E[X]$ compute $C^A = e^{-rT} E[Y]$

- Problem: simulate $S_{t_1}, S_{t_2}, \dots, S_{t_n}$:

$$\Rightarrow S_{t_{i+1}} = S_{t_i} \exp \left(\left(r - \frac{1}{2} \sigma^2 \right) \Delta t + \sigma (\Delta W_i) \right), \quad \Delta W_i \sim N(0, \Delta t), \text{ i.i.d.}$$

```

asian_mc.m

function [CArith,CGeom] = ...
    asian_mc( S0, K, sigma, r, T, n, m )

dt = T/n;
S = S0 * ones(m,1);  G = ones(m,1);  A = zeros(m,1);
for i=1:n
    dW = randn(m,1)*sqrt(dt);
    S = S .* exp( (r - .5*sigma^2)*dt + sigma *dW );
    G = G .* S;
    A = A + S;
end
G = G.^(1/n);      A = A/n;

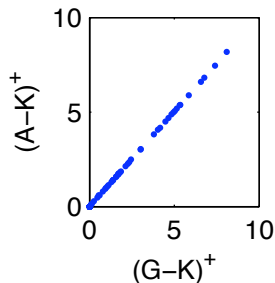
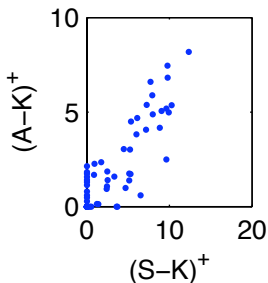
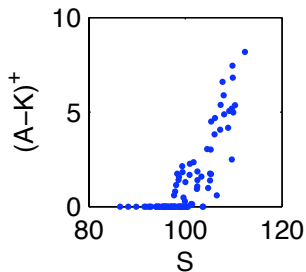
CArith = exp(-r*T)*mean( max(A-K,0) );
CGeom  = exp(-r*T)*mean( max(G-K,0) );

```

- Let consider three choices for the control variate:

| | | | |
|----------------|---------------|--------------------------|-------------------|
| Underlying | S_T | $e^{-rT} E[S_T] = S_0$ | |
| European Call | $(S_T - K)^+$ | $e^{-rT} E[(S_T - K)^+]$ | Black & Scholes |
| Geometric Avg. | $(G - K)^+$ | $e^{-rT} E[(G - K)^+]$ | Closed Form expr. |

- Look at the relationships:



- The Geom. Avg. payoff has the strongest dependence



Algorithm:

- 1 Compute the Geometric Average Call by mean the closed form expression
- 2 Simulate m replications of S_{t_i} , A and G
- 3 Compute the Arithmetic and Geometric Average payoffs
- 4 Compute the two option prices as discounted expected values of the payoffs
- 5 Compute the variance of the discounted Geometric Avg. payoff
- 6 Compute the covariance of the two discounted payoffs
- 7 Correct the Arithmetic option price by using the approximation error of the Geometric Call.



```
function [CArithCV,CGeom,CArithMC,CGeomMC] = ...
    asian_mccv( S0, K, sigma, r, T, n, m )

% Compute exact value of Geometric call
TT = (n+1)/(2*n) * T;
sigmaa = sigma * sqrt( (2*n+1)/(3*n) );
d = (n-1)/(6*n)*sigma^2;
CGeom = blsprice( S0, K, r, TT, sigmaa, d );

% Compute Arith and Geom calls with MC
dt = T/n;
S = S0*ones(m,1);
G = ones(m,1);
A = zeros(m,1);
for i=1:n
    dW = randn(m,1)*sqrt(dt);
    S = S .*exp( (r - .5*sigma^2)*dt + sigma *dW );
    G = G .* S;    A = A + S;
end
```

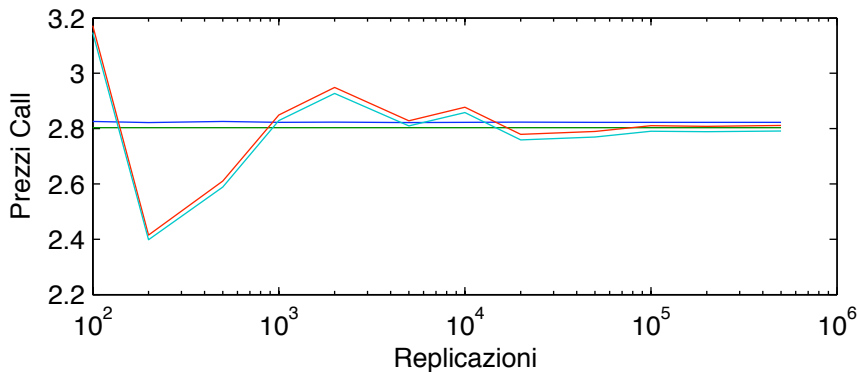
```
%% Compute Payoffs;
ArithPayoff = exp(-r*T)*max(A-K,0);
GeomPayoff = exp(-r*T)*max(G-K,0);

%% Compute sample mean, variance and covar.
CArithMC = mean(ArithPayoff);
CGeomMC = mean(GeomPayoff);
CovAG = cov( ArithPayoff, GeomPayoff );
VarG = CovAG(2,2);
CovAG = CovAG(1,2);

if (VarG>1e-7)
    CArithCV = CArithMC + CovAG/VarG *(CGeom - CGeomMC);
else
    CArithCV = CArithMC;
end
```



Control Variates Convergence



Multivariate CV

- X has values in \mathbb{R}^n , $E[X] = a$
- Let $\text{Cov}(X) = \Sigma_X : n \times n$, $\text{Cov}(X, Y) = \sigma_{XY} : n \times 1$

$$\hat{b}_{CV} = \hat{b} - (\hat{a} - a)^T \beta, \quad \beta = ?$$

- $\text{Var}(\hat{b}_{CV}) = \text{Var}(\hat{b}) + \beta^T \text{Cov}(\hat{a})\beta - 2\beta^T \text{Cov}(\hat{a}, \hat{b})$
- $E[\hat{a}] = a$,
- $\text{Cov}(\hat{a}) = \text{Cov}(\frac{1}{m} \sum_i x_i) = \frac{1}{m} \text{Cov}(x) = \frac{1}{m} \Sigma_X \quad (x_i \sim \text{i.i.d. } X)$
- $\text{Cov}(\hat{a}, \hat{b}) = \frac{1}{m} \sigma_{XY}$
- We want to find the value β that minimize:

$$\phi(\beta) = \beta^T \Sigma_X \beta - 2\beta^T \sigma_{XY}$$



Multivariate CV

Problem

Minimize $\phi(\beta) = \beta^T \Sigma_X \beta - 2\beta^T \sigma_{XY}$

- Σ_X is a covariance matrix, it is positive semidefinite and thus ϕ is convex
- $\partial_\beta \phi = 2\Sigma_X \beta - \sigma_{XY}$
- $\partial_\beta \phi = 0 \Rightarrow \beta = \Sigma_X^{-1} \sigma_{XY}$ minimize ϕ
- With this choice of β ,

$$\text{Var}(\hat{b}_{CV}) = \text{Var}(\hat{b}) - \frac{1}{m} \sigma_{XY}^T \Sigma_X^{-1} \sigma_{XY}$$

- Let c_{XY} be the vector of correlations between X and Y and C_X be the correlation matrix of X , then

$$\text{Var}(\hat{b}_{CV}) = \text{Var}(\hat{b}) - \frac{1}{m} \text{Var}(y) c_{XY}^T C_X^{-1} c_{XY}$$

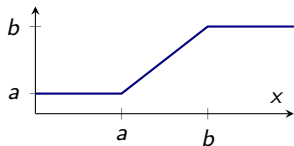
- Σ_X need to be positive definite (non-singular),



Exercise: Cliquet Option

- Define the truncating function:

$$(x)_{[a,b]} = \min(\max(x, a), b)$$



- Given the monitoring dates $0 = t_0 < t_1 < \dots < t_N = T$ consider the returns

$$R_n = \frac{S_{t_n}}{S_{t_{n-1}}} - 1 \quad n = 1, \dots, N$$

- Compute the truncated returns (Cap/Floor)

$$R_n^* = (R_n)_{[F,C]} \quad F < C, \quad n = 1, \dots, N$$

- Accumulate the returns and truncate

$$R_g = R_1^* + R_2^* + \dots + R_N^*, \quad R_g^* = (R_g)_{[F_g, C_g]} \quad F_g < C_g$$

- The **Cliquet payoff** is given by R_g^*



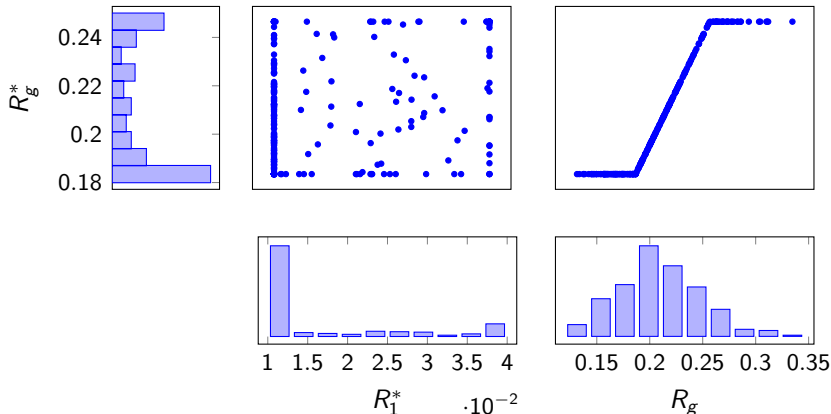
cliquet_mc.m

```
function [res,R,Rg] = cliquet_mc( r,vol,t, C,F,Cg,Fg, m )

if( t(1) ~= 0);    t = [0;t(:)];    end
dt = diff(t);
n = length(dt);
R = nan(m,n);
for i=1:n
    z = randn(m,1);
    R(:,i) = exp( (r-.5*vol^2)*dt(i) + vol*sqrt(dt(i))*z) - 1;
end
R = min(max(R,F),C);
Rg = sum(R,2);
Rg = min(max(Rg,Fg),Cg);
res = exp(-r*t(end))*mean(Rg);
```



Cliquet Options (cont.)



- Both R_1^* and R_g are candidates as control variates.
- R_g much better than R_1^* :

$$\text{corr}(R_1^*, R_g^*) = .315$$

\ll

$$\text{corr}(R_g, R_g^*) = .946$$



Cliquet Options

- It is necessary to analytically compute the expected value of R_1 or R_g .
- To this aim notice that, the truncating function can be rewritten as

$$(x)_{[a,b]} = a + (x - a)^+ - (x - b)^+, \quad b > a$$

- And R_n^* consists on the difference of two call payoffs on R_n
- Assume a B&S dynamics for S_t :

$$dS_t = rS_t dt + \sigma dW_t$$

then

$$R_n = \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(t_n - t_{n-1}) + \sigma(W_{t_n} - W_{t_{n-1}})\right) - 1$$

- $X_n = R_n + 1$, ($n = 1, \dots, N$) are independent and log-normally distributed.
- It follows that R_n^* is given by

$$R_n^* = F + (X_n - (F + 1))^+ - (X_n - (C + 1))^+, \quad n = 1, \dots, N$$



- R_n^* can be decomposed on a constant plus the difference of two calls:

$$R_n^* = F + (X_n - (F + 1))^+ - (X_n - (C + 1))^+,$$

$$X_n = \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(t_n - t_{n-1}) + \sigma(W_{t_n} - W_{t_{n-1}})\right)$$

- Thus, $E[R_n^*]$ and $E[R_g] = \sum_n E[R_n^*]$ are given by B&S formulae:

$$E[R_n^*] = F + e^{\Delta t_n} \Phi(d_+^{F,n}) - (F + 1)\Phi(d_-^{F,n}) \\ - e^{\Delta t_n} \Phi(d_+^{C,n}) + (C + 1)\Phi(d_-^{C,n})$$

where $\Delta t_n = t_n - t_{n-1}$,

$$d_{\pm}^{F,n} = \frac{-\log(F + 1) + (r \pm \sigma^2/2)\Delta t_n}{\sqrt{\sigma^2 \Delta t_n}} \quad \text{and}$$

$$d_{\pm}^{C,n} = \frac{-\log(C + 1) + (r \pm \sigma^2/2)\Delta t_n}{\sqrt{\sigma^2 \Delta t_n}}.$$

- All the R_n^* and the R_g can be used as control variables.



cliquet_mccv.m

```
function res = cliquet_mccv( r,vol,t, C,F,Cg,Fg, m )

dt = diff(t);
n = length(dt);
R = nan(m,n);
for i=1:n
    z = randn(m,1);
    R(:,i) = exp( (r-.5*vol^2)*dt(i) + vol*sqrt(dt(i))*z) - 1;
end
R = min(max(R,F),C);
Rg = sum(R,2);                                clear R;
RgStar = min(max(Rg,Fg),Cg);

mcRg = exp(-r*t(end))*mean(Rg);
mcRgStar = exp(-r*t(end))*mean(RgStar);
```



cliquet_mccv.m

```
%% Compute the B&S prices
```

```
dfplus = (- log(F+1) + (r + .5*vol^2)*dt) ./ (vol*sqrt(dt));
dfminus = dfplus - vol*sqrt(dt);
dcplus = (- log(C+1) + (r + .5*vol^2)*dt) ./ (vol*sqrt(dt));
dcminus = dcplus - vol*sqrt(dt);
```

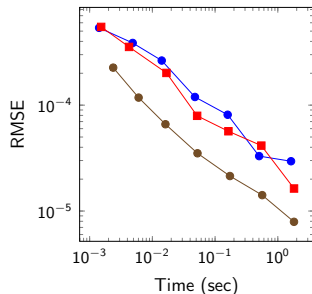
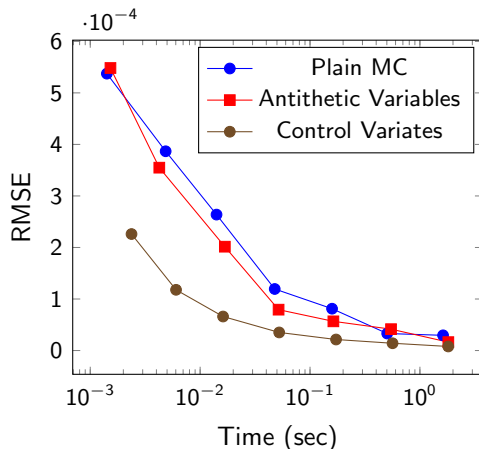
```
ExpRStar = F + exp(r*dt) .* (normcdf(dfplus) - normcdf(dcplus))
          - (F+1) * normcdf(dfminus) + (C+1) * normcdf(dcminus);
ExpR = exp(-r*t(end))*sum(ExpRStar);
```

```
%% Compute the CV approximation
```

```
covMatrix = cov(Rg,RgStar);
beta = covMatrix(1,2)/covMatrix(1,1);
res = mcRgStar - beta*(mcRg - ExpR);
```



Cliquet Options: Comparison of Methods



- $RMSE = \sqrt{\text{mean}(C_{MC} - C_{best})^2}$
- Here computed over 20 different sets of simulations



Stratified Sampling

- Partition the domain Ω :

$$\Omega = \bigcup_{k=1}^K W_k, \quad W_k \cap W_j = \emptyset, \quad \text{for } k \neq j = 1, \dots, K.$$

- $p_k = P[W_k]$ is known

- Bayes rule:

$$\theta = E[g(X)] = \sum_k P[W_k] E[g(X)|W_k] = \sum_k p_k \theta_k.$$

- For each partition W_k , approximate θ_k by MC:

- Simulate $x_1^k, x_2^k, \dots, x_{m_k}^k \sim \text{i.i.d. } P|_{W_k}$

- $\hat{\theta}^k \equiv \frac{1}{m_k} \sum_i g(x_i^k) \sim N(\theta_k, \frac{1}{m_k} \sigma_k^2)$

- $\hat{\theta}^K = \sum_k p_k \hat{\theta}^k \sim N(\theta, \sum_k \frac{p_k}{m_k} \sigma_k^2)$

- Increase m_k where σ_k^2 is large



Example

Add examples



Importance Sampling

- Let consider the change of variables

$$\int_{\Omega} g(X) dP = \int_{\Omega} \frac{g(X)}{f(X)} dF, \quad E^P[g] = E^F[g/f],$$

where $E^P[f] = \int_{\Omega} f(X) dP = 1$ and $f dP = dF$.

- Use MC to compute the estimate $\hat{\theta}_{IS}$ of $\theta = E^F[g/f]$
 - Error has variance: $\text{Var}[\hat{\theta}_{IS}] = \frac{1}{m} \text{Var}^F[g/f]$
- The optimal choice is $f(x) = cg(x)$ (utopia)
- In practice f is chosen so that
 - f approximate $|g|$
 - F easy to simulate
 - f should not have a too fast decay



Example

Add examples



Outline

5 Binomial Model

6 Finite Differences

7 Monte Carlo

- Numerical Integration
- Monte Carlo Method
- Generating Random Variables
- Variance Reduction Techniques
- **SDE Integration**
- American Options

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



Solutions to SDE

- Consider the Stochastic Differential Equation (SDE):

$$dX_t = a(X_t, t)dt + b(X_t, t)dW_t, \quad X_0 = x_0$$

- We know the exact solution if we know a f such that

$$X_t = f(W_t, t), \quad t \in [0, T]$$

- or if we know the transition density function (TDF)

$$p(y, s; x, t)$$

that is, the conditional density of X_s given $X_t = x$.



Euler Discretization

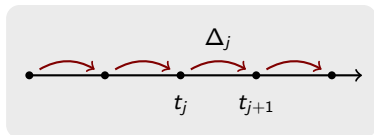
$$\text{SDE:} \quad dX_t = a(X_t)dt + b(X_t)dW_t, \quad X_0 = x_0$$

- Time discretization: $0 = t_0 < t_1 < \dots < t_j < \dots < t_N = T$

$$\Delta_j = t_{j+1} - t_j,$$

$$\Delta W_j = W_{t_{j+1}} - W_{t_j},$$

$$\Delta X_j = X_{t_{j+1}} - X_{t_j}$$



- Then $\Delta W_j \sim N(0, \Delta_j)$ iid.
- Euler method: at each step freeze a and b at (X_{t_j}) :

$$\text{Euler:} \quad \Delta Y_j = a(Y_j, t_j)\Delta_j + b(Y_j, t_j)\Delta W_j \quad Y_0 = x_0$$

$$\text{or} \quad Y_{j+1} = Y_j + a(Y_j, t_j)\Delta_j + b(Y_j, t_j)\Delta W_j$$

where Y_j is the approximation to X_{t_j} .



Example (Geometric Brownian Motion)

$$dS_t = \mu S_t dt + \sigma S_t dW_t \quad S_0 = s_0$$

Exact solution by Itô Lemma on $X_t = \log(S_t)$:

$$\begin{aligned} dX_t &= \frac{\partial \log(S_t)}{\partial S_t} dS_t + \frac{1}{2} \frac{\partial^2 \log(S_t)}{\partial S_t^2} \sigma^2 S_t^2 dt \\ &= \frac{1}{S_t} (\mu S_t dt + \sigma S_t dW_t) - \frac{1}{2} \frac{1}{S_t^2} \sigma^2 S_t^2 dt = \left(\mu - \frac{1}{2}\sigma^2\right) dt + \sigma dW_t \end{aligned}$$

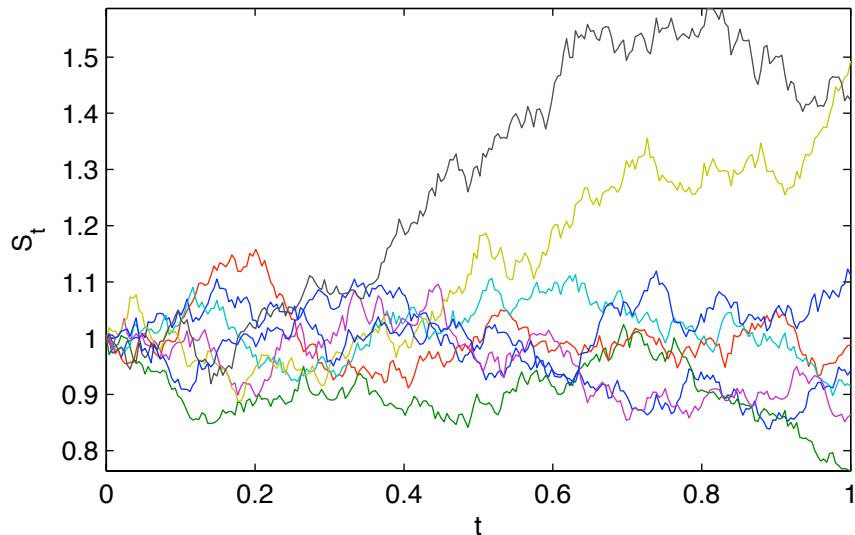
$$X_t - X_0 = \left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t$$

$$S_t = s_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t\right)$$

Euler discretization:

$$Y_{n+1} - Y_n = \mu Y_n \Delta t + \sigma Y_n \Delta W_t, \quad Y_0 = s_0$$

Trajectories of Brownian motions



Exercise (European Call)

Compute $C = e^{-rT} E^Q[(S_T - K)^+]$ where $dS_t = rS_t dt + \sigma S_t dW_t^Q$ and S_0 is given.

bs_euler.m

```
function [c,v] = bs_euler( S0, K, T, r, sigma, M, N )

dt = T/(N-1);
C = zeros(M,1);

for i=1:M
    %% simulation of the i-th scenario
    S = S0;
    for n=1:N-1
        dW = randn()*sqrt(dt);
        S = S*exp( (r - .5*sigma^2)*dt + sigma*dW );
    end
    %% payoff in the i-th scenario
    C(i) = max(S-K,0);
end

c = mean(C);    v = var(C);
```

Convergence

- Weak Convergence of order γ :

$$|E[p(Y_n)] - E[p(X_{t_n})]| \leq O(\Delta_t^\gamma)$$

for all p smooth and with polynomial growth

- Strong Convergence of order γ :

$$E[|Y_n - X_{t_n}|] \leq O(\Delta_t^\gamma)$$

- Strong convergence is important if trajectories are of interest
- Weak convergence is important when expected values of functions of X are required



Taylor formula and ODEs

- Consider the ODE: $dX_t = a(X_t)dt$ or $X_t - X_0 = \int_0^t a(X_s)ds$
- Consider $f(X_t)$ with $f \in \mathcal{C}^1(\mathbb{R})$. Thus:

$$df(X_t) = \partial_x f(X_t)dX_t = a(X_t)\partial_x f(X_t)dt$$

or, defining $Lf(x) = a(x)\partial_x f(x)$,

$$f(X_t) = f(X_0) + \int_0^t Lf(X_s)ds \quad (8)$$

- Now, applying (8) to the integrand in $f^{(1)} \equiv Lf$, it holds¹

$$\begin{aligned} f(X_t) &= f(X_0) + \int_0^t f^{(1)}(X_0)ds + \int_0^t \int_0^s Lf^{(1)}(X_z)dz ds \\ &= f(X_0) + tf^{(1)}(X_0) + \int_0^t \int_0^s f^{(2)}(X_z)dz ds \end{aligned}$$

- and so on:

$$f(X_t) = f(X_0) + tf^{(1)}(X_0) + \frac{t^2}{2}f^{(2)}(X_0) + \int_0^t \int_0^s \int_0^z \dots$$



Milstein method

- Itô formula for $f(X_t)$, where $dX_t = a(X_t)dt + b(X_t)dW_t$:

$$df(X_t) = \underbrace{\left(a_t \partial_x f(X_t) + \frac{1}{2} b_t^2 \partial_{xx} f(X_t) \right)}_{L^0 f(X_t)} dt + \underbrace{b \partial_x f(X_t)}_{L^1 f(X_t)} dW_t$$

$$f(X_{t+h}) = f(X_t) + \int_t^{t+h} L^0 f(X_s) ds + \int_t^{t+h} L^1 f(X_s) dW_s \quad (9)$$

- Consider $X_{t+h} = X_t + \int_t^{t+h} a(X_s) ds + \int_t^{t+h} b(X_s) dW_s$
- Applying Itô to $a_s \equiv a(X_s)$ and $b_s \equiv b(X_s)$, the latter becomes

$$X_{t+h} = X_t + \underbrace{\int_t^{t+h} \left(a_t + \int_t^s L^0 a_z dz + \int_t^s L^1 a_z dW_z \right)}_{a_s} ds + \int_t^{t+h} \underbrace{\left(b_t + \int_t^s L^0 b_z dz + \int_t^s L^1 b_z dW_z \right)}_{b_s} dW_s$$



- Recall

$$X_{t+h} = X_t + \int_t^{t+h} \left(a_t + \int_t^s L^0 a_z dz + \int_t^s L^1 a_z dW_z \right) ds \\ + \int_t^{t+h} \left(b_t + \int_t^s L^0 b_z dz + \int_t^s L^1 b_z dW_z \right) dW_s$$

where $L^0 = a \partial_x + \frac{1}{2} b^2 \partial_{xx}$ and $L^1 = b \partial_x$

- Thus $X_{t+h} = X_t + a_t h + b_t \Delta W_t + \int_t^{t+h} \int_t^s b_z \partial_x b_z dW_z dW_s + R$

$$R = \int_t^{t+h} \int_t^s L^0 a_z dz ds + \int_t^{t+h} \int_t^s L^1 a_z dW_z ds + \int_t^{t+h} \int_t^s L^0 b_z dz dW_s$$

- From Itô expansion of $b_z \partial_x b_z$ and $\int_t^{t+h} \int_t^s dW_z dW_s = \frac{1}{2} ((\Delta W_t)^2 - h)$:

$$X_{t+h} = X_t + a_t h + b_t \Delta W_t + \frac{1}{2} b_t \partial_x b_t ((\Delta W_t)^2 - h) + \tilde{R}$$

$$\tilde{R} = R + \int_t^{t+h} \int_t^s \left(\int_t^z L^0 L^1 b_r dr + \int_t^z L^1 L^1 b_r dW_r \right) dz dW_s$$



$$X_{t+h} = X_t + a(X_t)h + b(X_t)\Delta W_t + \frac{1}{2}b(X_t)b'(X_t)((\Delta W_t)^2 - h) + \tilde{R}$$

- Euler:
 - Strong conv. of order 1/2
 - Weak conv. of order 1
- Milstein:
 - Strong conv. of order 1
 - Weak conv. of order 1



Outline

5 Binomial Model

6 Finite Differences

7 Monte Carlo

- Numerical Integration
- Monte Carlo Method
- Generating Random Variables
- Variance Reduction Techniques
- SDE Integration
- American Options

8 Compound Poisson Processes Simulation

9 Fourier Transform methods



A simple case

- Underlying is S_t and the risk-neutral interest rate is null: $r = 0$
- Choose when to get a payoff $(1 - S_t)^+$: now or at $t = T$
- The contract value is

$$V_T = (1 - S_T)^+ \qquad C_0 = E[V_T | S_0],$$

$$V_0 = \max((1 - S_0)^+, C_0)$$

- Monte Carlo:

$$V_T^\omega = (1 - S_T^\omega)^+ \qquad \omega = 1, \dots, m$$

$$\hat{C}_0 = \frac{1}{m} \sum_{\omega} \hat{V}_T^\omega, \quad \hat{V}_0 = \max((1 - S_0)^+, \hat{C}_0),$$

- \hat{C}_0 is unbiased: $E[\hat{C}_0] = C_0$
- \hat{V}_0 is biased high:

$$E[\hat{V}_0] = E[\max((1 - S_0)^+, \hat{C}_0)]$$

$$> \max((1 - S_0)^+, E[\hat{C}_0]) = \max((1 - S_0)^+, C_0) = V_0$$



Regression

- Consider the r.v.s. $Y \in \mathbb{R}$ and $X \in \mathbb{R}^k$
- We want to approximate Y by means of $X^T \beta$, with $\beta \in \mathbb{R}^k$ constant

$$Y = X^T \beta + U$$

- The Least Squares approximation $\tilde{\beta}$ minimizes $E[U^2]$:

$$E[U^2] = E[Y^2] + \beta^T E[XX^T] \beta - 2\beta^T E[XY]$$

- Thus,

$$\tilde{\beta} = S^{-1} q, \quad \text{where} \quad S = E[XX^T], \quad q = E[XY].$$

- MC: Given $(x_\omega, y_\omega) \sim \text{iid}(X, Y)$, $\omega = 1, \dots, m$

$$\hat{\beta} = \hat{S}^{-1} \hat{q} \quad \text{where} \quad \hat{S} = \frac{1}{m} \sum_{\omega=1}^m x_\omega x_\omega^T, \quad \hat{q} = \frac{1}{m} \sum_{\omega=1}^m x_\omega y_\omega.$$

- $\hat{S} \rightarrow S$, $\hat{q} \rightarrow q$ and, thus, $\hat{\beta} \rightarrow \tilde{\beta}$



Regression (cont.)

- MC approximation:

$$\hat{\beta} = \hat{S}^{-1} \hat{q} \quad \text{where} \quad \hat{S} = \frac{1}{m} \sum_{\omega=1}^m x_{\omega} x_{\omega}^T, \quad \hat{q} = \frac{1}{m} \sum_{\omega=1}^m x_{\omega} y_{\omega}.$$

- With abuse of notation define

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_m^T \end{pmatrix} : m \times k \quad \text{and} \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} : m \times 1$$

- then: $\hat{S} = \frac{1}{m} X^T X$, $\hat{q} = \frac{1}{m} X^T y$ and $\hat{\beta} = (X^T X)^{-1} X^T y$
- that is $\hat{\beta}$ is the OLS estimator of the linear regression

$$y = X\beta + u, \quad u \sim (0, \sigma^2 I_m).$$

- Recall the original model was $Y = X^T \beta + U$



Regression and Approximation

Problem

Approximate $c(\omega) : \Omega \rightarrow \mathbb{R}$ by means of $\phi_i(\omega) : \Omega \rightarrow \mathbb{R}$, $i = 1, \dots, k$

$$c(\omega) = \sum_{i=1}^k \phi_i(\omega) \beta_i + \varepsilon(\omega)$$

- ϕ_i basis functions, ε error function
- We need a norm to evaluate the error: $\|\varepsilon\|$
- Given a probability measure $P(\omega)$, define the scalar product

$$\langle u, v \rangle = E[uv] = \int_{\Omega} u(\omega)v(\omega)dP(\omega)$$

- $\|\varepsilon\| = \langle \varepsilon, \varepsilon \rangle^{1/2} = E[\varepsilon^2]^{1/2}$ is a semi-norm
- Back to the regression: $Y = X^T \beta + U$,

$$Y(\omega) = c(\omega), \quad X(\omega) = (\phi_1(\omega) \quad \phi_2(\omega) \quad \dots \quad \phi_k(\omega))^T$$



Outline

- 5 Binomial Model
- 6 Finite Differences
- 7 Monte Carlo
- 8 Compound Poisson Processes Simulation**
 - Poisson Processes
 - Algorithms
 - American Options
- 9 Fourier Transform methods



Outline

- 5 Binomial Model
- 6 Finite Differences
- 7 Monte Carlo
- 8 Compound Poisson Processes Simulation**
 - Poisson Processes
 - Algorithms
 - American Options
- 9 Fourier Transform methods



Exponential R.V.

$$Y \sim \text{Exp}(\lambda)$$

Domain \mathbb{R}^+ .

CDF $F_Y(y) = 1 - e^{-\lambda y}$, for $y \geq 0$.

PDF $f_Y(y) = \lambda e^{-\lambda y}$, for $y \geq 0$.

Inv. CDF $F_Y^{-1}(u) = -\lambda^{-1} \log(1 - u)$.

Moments $E[Y] = \lambda^{-1}$, $\text{Var}(Y) = \lambda^{-2}$.

Since the inverse CDF is explicitly known an exponential r.v. can be easily simulated:

$$U \sim \text{Unif}([0, 1]), \quad \Rightarrow \quad F_Y^{-1}(U) \sim \text{Exp}(\lambda)$$

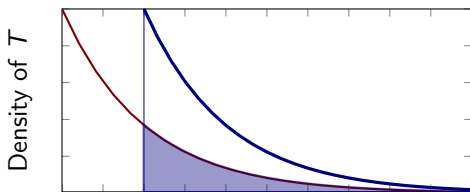


Absence of Memory

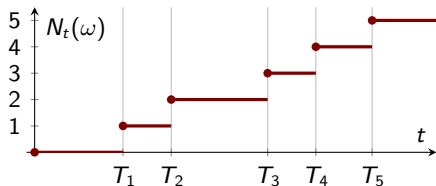
Absence of Memory

$$T \sim \text{Exp}(\lambda) \quad \Rightarrow \quad P[T > t + x | T > t] = P[T > x]$$

- Let $T \sim \text{Exp}(\lambda)$ be the time when a specific event occurs. The AoM states that if we are at time t and that event didn't occur then $T - t$ will have the same density that T had at time 0. I.e. the distribution of the waiting time is the same we had at time 0.



The poisson process



For $i = 1, 2, \dots$

- $\tau_i \sim \text{Exp}(\lambda)$, indep.
- $T_i = T_{i-1} + \tau_i$

- T_1 is the time of the first occurrence of a specific event
- T_2 that of the second occurrence
- \vdots
- T_n the time of the n -th occurrence

Poisson Process

N_t counts how many of such events occurs up to time t :

$$N_t(\omega) = \sum_{n=1}^{\infty} 1\{t \geq T_n(\omega)\}, \quad \omega \in \Omega, t \geq 0.$$

Poisson Distribution

$$N \sim \text{Pois}(\lambda)$$

Domain Non-negative integers: $k = 0, 1, 2, \dots$

$$\text{PMF } P_N[k] = \frac{\lambda^k}{k!} e^{-\lambda}$$

$$\text{CDF } F_N(k) = \frac{\Gamma(k+1, \lambda)}{\Gamma(k+1)} = \sum_{i=1}^k P_N[i]$$

$$\text{Moments } E[N] = \lambda, \text{Var}(N) = \lambda$$

That is,

- $N_t \sim \text{Pois}(\lambda t)$
- in an interval $[0, t]$ we expect an average of λt jumps

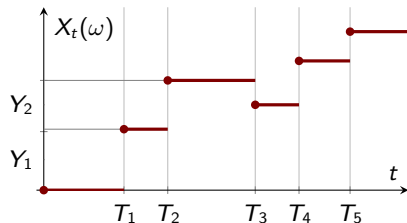
PS. $\Gamma(x)$ and $\Gamma(x, \lambda)$ are the complete and incomplete Gamma functions.



Compound Poisson Process

Idea

Instead of jumps of unit magnitude, use iid random jump sizes.



- $\tau_i \sim \text{i.i.d Exp}(\lambda)$
- $Y_i \sim \text{i.i.d}$
- $\{Y_i\}, \{\tau_i\}$ independent.
- $T_i = T_{i-1} + \tau_i$
- $X_{T_i} = X_{T_i^-} + Y_i$

Compound poisson process:

$$X_t = \sum_{i=1}^{N_t} Y_i = \sum_{i=1}^{\infty} Y_i 1\{t \geq T_i\}.$$



Compound Poisson Processes

- X_t is Compound Poisson Process if and only if it is a Levy process with piecewise constant sample paths (see Cont and Tankov 2004, Prop. 3.3).



Outline

- 5 Binomial Model
- 6 Finite Differences
- 7 Monte Carlo
- 8 Compound Poisson Processes Simulation**
 - Poisson Processes
 - **Algorithms**
 - American Options
- 9 Fourier Transform methods



The basic scheme

$$X_t = \sum_{i=1}^{N_t} Y_i = \sum_{i=1}^{\infty} Y_i 1\{t > T_i\}, \quad Y_i \sim N(\mu, \sigma^2)$$

```

cpois1.m

function x=cpois1(lambda,mu,sigma,t,m)

n = poissrnd(lambda*t,m,1);
x = zeros(m,1);
for w=1:m
    for j=1:n(w)
        y = mu + sigma*randn;
        x(w) = x(w) + y;
    end
end
end

```

- We can **swap** the two loops and/or **vectorize** one of them.



Second Algorithm: foreach sample path

$$X_t = \sum_{i=1}^{N_t} Y_i$$

$$Y_i \sim N(\mu, \sigma^2)$$

```

function x=cpois2(lambda,mu,sigma,t,m)

n = poissrnd(lambda*t,m,1);
x = zeros(m,1);
for w=1:m
    y = mu + sigma*randn(n(w),1);
    x(w) = sum(y);
end
  
```

cpois2.m



Third Algorithm: for each jump

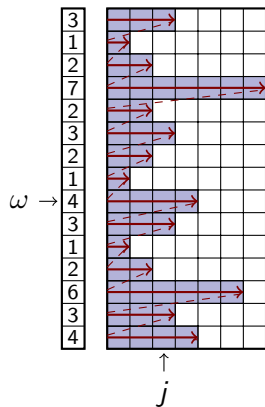
- $X_t = \sum_{i=1}^{N_t} Y_i \quad Y_i \sim N(\mu, \sigma^2).$

```
cpois3.m  
  
function x=cpois3(lambda,mu,sigma,t,m)  
  
n = poissrnd(lambda*t,m,1);  
nMax = max(n);  
x = zeros(m,1);  
for j=1:nMax  
    y = mu + sigma*randn(m,1);  
    x = x + y .* (j<=n);  
end
```

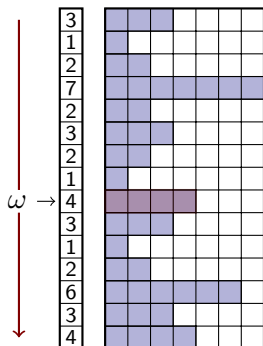


In pictures

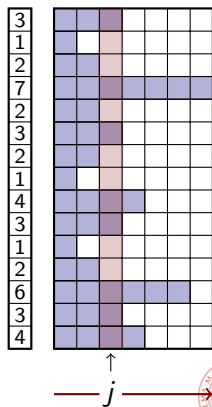
Alg. 1

 $N(\omega)$ $Y_j(\omega)$ 

Alg. 2

 $N(\omega)$ $Y_j(\omega)$ 

Alg. 3

 $N(\omega)$ $Y_j(\omega)$ 

Fourth algorithm

Assumption: we know the distribution $\sum_{i=1}^n Y_i \sim D_n$.

Example:

$$\sum_{i=1}^n Y_i \sim N(n\mu, n\sigma^2)$$

cpois4.m

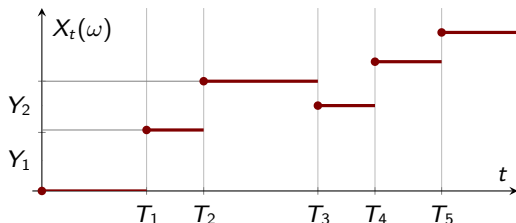
```
function x=cpois4(lambda,mu,sigma,t,m)

n = poisrnd(lambda*t,m,1);
x = n*mu + sigma*sqrt(n).*randn(m,1);
```



Simulating Sample Paths

- Sample paths are described by means of (T_i, X_{T_i}) , $i = 1, \dots, N_T$.



Data Structures:

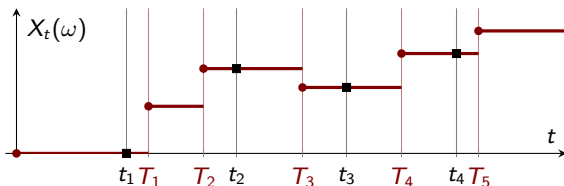
- Store T_i , Y_i and X_{T_i} in matrices: $m \times nMax$.
 - $X = \text{cumsum}(Y, 2)$;
 - We don't know $nMax$.
- Cell Arrays
- Linearize the data-structure



Sample paths on a grid

$$X_{t_i}(\omega) = \sum_j Y_j(\omega) 1\{t_i \geq T_j(\omega)\}$$

- We may need to evaluate the paths on a grid (t_1, t_2, \dots, t_n) .



- It's an approximation
 - two jump may occur between 2 grid points
 - jump's exact location is lost
- Need to iterate on ω, i, j



Algorithm

cpois5.m

```

function X=cpois5(lambda,mu,sigma,t,m)

n=length(t);
X = zeros(m,n);
for w=1:m                                     % forall (parallel)
    % Simulate the poisson trajectory
    Tmax=0;    T=[];
    while Tmax<t(n)
        Tmax = Tmax + exprnd(lambda);
        T = [T, Tmax];
    end

    % Simulate the comp. poisson
    N = lenth(T);
    y = mu + sigma*randn(1,N);
    for i=1:n
        for j=1:N
            X(w,i) = X(w,i) + y(j) * (t(i) >= T(j) );
        end
    end
end

```

- Vectorize the j loop

```

function X=cpois6(lambda,mu,sigma,t,m)
n=length(t);
X = zeros(m,n);
for w=1:m                                     % forall
    Tmax=0;    T=[];
    while Tmax<t(n)
        Tmax = Tmax + exprnd(lambda);
        T = [T, Tmax];
    end

    N = lenth(T);
    Y = mu + sigma*randn(1,N);                 % parallel wrt j
    for i=1:n
        X(w,i) = X(w,i) + sum( Y .* (t(i) >= T) );
    end
end

```



- Vectorize the i loop

cpois7.m

```
function X=cpois7(lambda,mu,sigma,t,m)

n=length(t);
X = zeros(m,n);

for w=1:m
% forall (parallel)
    Tmax=0;    T=[];
    while Tmax<t(n)
        Tmax = Tmax + exprnd(lambda);
        T = [T, Tmax];
    end

    N = lenth(T);
    y = mu + sigma*randn(1,N); % parallel wrt j
    for j=1:N
        X(w,:) = X(w,:) + y(j) * (t >= T(j) ); % parallel wrt i
    end
end
```


- Vectorize the i loop and perform some optimisation.

```

cpois7b.m
function X=cpois7b(lambda,mu,sigma,t,m)

n=length(t);
X = zeros(m,n);

for w=1:m                                % forall
    % Simulate jump times
    T=0;
    while T<t(n)
        T = T +exprnd(lambda);
        if (T>t(n));
            break;
        end
        y = mu + sigma* randn();          % y: scalar
        X(w,:) = X(w,:) + y * (t>=T);    % parallel wrt i
    end
end

```



- Notice that: $X_{t_i} = X_{t_{i-1}} + \sum_j Y_j \times 1\{T_j \in (t_{i-1}, t_i]\}$.

cpois8.m

```
function X=cpois8(lambda,mu,sigma,t,m)

n=length(t);
X = zeros(m,n+1);
t = [0,t];

for w=1:m
    % Simulate jump times
    Tmax=0;    T=[];
    while Tmax<t(n)
        Tmax = Tmax + exprnd(lambda);
        T = [T, Tmax];
    end

    N = length(T);
    y = mu + sigma*randn(1,N);
    for i=2:(n+1)
        X(w,i) = X(w,i-1) + sum(y .* (t(i)>=T) .* (t(i-1)<T));
    end
end
```

cpois9.m

```
function X=cpois9(lambda,mu,sigma,t,m)

n=length(t);
X = zeros(m,n);

T = zeros(m,1);
while min(T)<t(n) % for j
    T = T + exprnd(lambda,m,1);
    y = mu + sigma*randn(1,N);
    for i=1:n % forall (parallel)
        X(:,i) = X(:,i) + y .* (t(i)>=T);
    end
end
end
```



- Truncate the poisson distribution: $X_{t_i} = \sum_{j=1}^{N_{max}} Y_j 1\{t_i \geq T_j\}$.

cpois10.m

```
function X=cpois10(lambda,mu,sigma,t,m,Nmax)

n=length(t);
U = exprnd(lambda,m,Nmax);
T = cumsum(U,2);
Y = mu + sigma*randn(m,Nmax);

X = zeros(m,n);
for i=1:n                                % forall (parallel)
    for j=1:Nmax
        X(:,i) = X(:,i) + Y(:,j) .* (t(i)>=T(:,j));
    end
end
end
```



- Truncate the poisson distribution: $X_{t_i} = \sum_{j=1}^{N_{max}} Y_j 1\{t_i \geq T_j\}$.

cpois10b.m

```
function X=cpois10b(lambda,mu,sigma,t,m,Nmax)

n=length(t);
U = exprnd(lambda,m,Nmax);
T = cumsum(U,2);
Y = mu + sigma*randn(m,Nmax);

X = zeros(m,n);
for i=1:n % forall (parallel)
    X(:,i) = X(:,i) + sum( Y .* (t(i)>=T), 2 );
end
```



Outline

- 5 Binomial Model
- 6 Finite Differences
- 7 Monte Carlo
- 8 Compound Poisson Processes Simulation**
 - Poisson Processes
 - Algorithms
 - American Options
- 9 Fourier Transform methods



Bermudan Options

Notation

- X_t state variable
- V_t value of the contract at time t
- D_t stoch. discount factor from t to $t + 1$
- $t = 1, 2, \dots, T$ exercise dates

At each exercise date the holder can choose between

- keep the contract: $C_t = E[D_t V_{t+1} | X_t]$
- exercise and obtain $h_t(X_t)$

The value of the contract at time t is

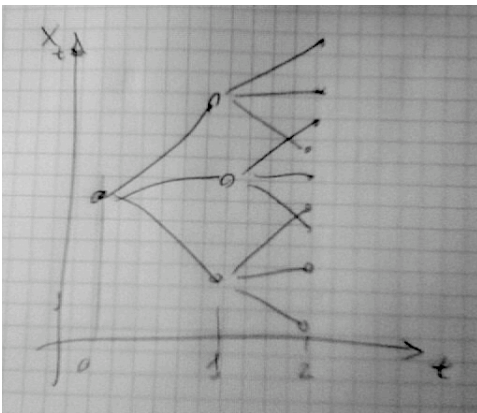
$$V_t = \max(h_t(X_t), C_t)$$



Problem: the conditional expectation.

$$V_t = \max \left(h_t(X_t), E[D_t V_{t+1} | X_t] \right)$$

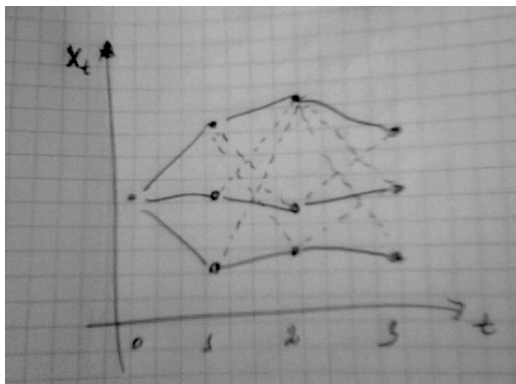
Solution 1: Random Trees



Problem: the conditional expectation.

$$V_t = \max \left(h_t(X_t), E[D_t V_{t+1} | X_t] \right)$$

Solution 2: Stochastic Mesh (recombines)



Problem: the conditional expectation.

$$V_t = \max \left(h_t(X_t), E[D_t V_{t+1} | X_t] \right)$$

Solution 3: Use Regression

Idea

$C_t(x) = E[D_t V_{t+1} | X_t = x]$ is a function of x .

Approximate C_t as a linear combination of functions of x



Regression Based MC

We want to approximate the function:

$$c(x) = E[D_t V_{t+1} | X_t = x]$$

Note that $C = c(X_t) = E[D_t V_{t+1} | X_t]$ is a r.v.

Use a set of functions of x to approximate c :

$$Z_1 = \phi_1(X_t), Z_2 = \phi_2(X_t), \dots, Z_k = \phi_k(X_t)$$

Approximation:

$$C = Z_1\beta_1 + Z_2\beta_2 + \dots + Z_k\beta_k + U$$

In vector form:

$$C = Z^T \beta + U, \quad \text{where} \quad Z = \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_k \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix}.$$

Here, U is the error term



$$C = Z^T \beta + U,$$

$$C = E[D_t V_{t+1} | X_t]$$

- Find β that minimize $E[U^2]$

$$E[U^2] = E[C^2] - 2 E[CZ^T] \beta + \beta^T E[ZZ^T] \beta$$

$$\frac{\partial E[U^2]}{\partial \beta^T} = -2 E[CZ^T] + 2 \beta^T E[ZZ^T]$$

- The gradient is null for

$$\beta = E[ZZ^T]^{-1} E[ZC]$$

- $E[ZZ^T]$ depends on the basis functions
- $E[ZC]$ depends also on the continuation value C

$$E[ZC] = E[Z E[D_t V_{t+1} | X_t]] = E[Z D_t V_{t+1}]$$

- The conditional expectation is no longer necessary!!



$$C = Z^T \beta + U,$$

$$\beta = E[ZZ^T]^{-1} E[ZC]$$

The error is orthogonal to each basis function

$$E[ZU] = E[ZZ^T]\beta - E[ZC] = 0.$$

When $Z_1 = 1$, **the approximation does not introduce a bias.**

Indeed, since the first component of $E[ZU]$ is zero,

$$0 = E[U] = E[C] - E[Z^T \beta].$$



$$C = Z^T \beta + U, \quad \beta = E[ZZ^T]^{-1} E[ZD_t V_{t+1}]$$

Need to compute $E[ZZ^T]$ and $E[ZC]$

Idea: Use MonteCarlo:

$$E[ZZ^T] = \frac{1}{M} \sum_{\omega} z_{\omega} z_{\omega}^T, \quad E[ZD_t V_{t+1}] = \frac{1}{M} \sum_{\omega} z_{\omega} d_{t,\omega} v_{t+1,\omega}$$

where

- z_{ω} , $d_{t,\omega}$ and $v_{t+1,\omega}$ are the ω -th realizations of the r.v. Z_{ω} , $D_{t,\omega}$, $V_{t+1,\omega}$

Any conditional expectation have been used



A simple case

- Underlying is S_t and the risk-neutral interest rate is null: $r = 0$
- Choose when to get a payoff $(1 - S_t)^+$: now or at $t = T$
- The contract value is

$$V_T = (1 - S_T)^+ \qquad C_0 = E[V_T | S_0],$$

$$V_0 = \max((1 - S_0)^+, C_0)$$

- Monte Carlo:

$$V_T^\omega = (1 - S_T^\omega)^+ \qquad \omega = 1, \dots, m$$

$$\hat{C}_0 = \frac{1}{m} \sum_{\omega} \hat{V}_T^\omega, \quad \hat{V}_0 = \max((1 - S_0)^+, \hat{C}_0),$$

- \hat{C}_0 is unbiased: $E[\hat{C}_0] = C_0$
- \hat{V}_0 is biased high:

$$E[\hat{V}_0] = E[\max((1 - S_0)^+, \hat{C}_0)]$$

$$> \max((1 - S_0)^+, E[\hat{C}_0]) = \max((1 - S_0)^+, C_0) = V_0$$



Regression

- Consider the r.vs. $Y \in \mathbb{R}$ and $X \in \mathbb{R}^k$
- We want to approximate Y by means of $X^T \beta$, with $\beta \in \mathbb{R}^k$ constant

$$Y = X^T \beta + U$$

- The Least Squares approximation $\tilde{\beta}$ minimizes $E[U^2]$:

$$E[U^2] = E[Y^2] + \beta^T E[XX^T] \beta - 2\beta^T E[XY]$$

- Thus,

$$\tilde{\beta} = S^{-1} q, \quad \text{where} \quad S = E[XX^T], \quad q = E[XY].$$

- MC: Given $(x_\omega, y_\omega) \sim \text{iid}(X, Y)$, $\omega = 1, \dots, m$

$$\hat{\beta} = \hat{S}^{-1} \hat{q} \quad \text{where} \quad \hat{S} = \frac{1}{m} \sum_{\omega=1}^m x_\omega x_\omega^T, \quad \hat{q} = \frac{1}{m} \sum_{\omega=1}^m x_\omega y_\omega.$$

- $\hat{S} \rightarrow S$, $\hat{q} \rightarrow q$ and, thus, $\hat{\beta} \rightarrow \tilde{\beta}$



Regression (cont.)

- MC approximation:

$$\hat{\beta} = \hat{S}^{-1} \hat{q} \quad \text{where} \quad \hat{S} = \frac{1}{m} \sum_{\omega=1}^m x_{\omega} x_{\omega}^T, \quad \hat{q} = \frac{1}{m} \sum_{\omega=1}^m x_{\omega} y_{\omega}.$$

- With abuse of notation define

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_m^T \end{pmatrix} : m \times k \quad \text{and} \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} : m \times 1$$

- then: $\hat{S} = \frac{1}{m} X^T X$, $\hat{q} = \frac{1}{m} X^T y$ and $\hat{\beta} = (X^T X)^{-1} X^T y$
- that is $\hat{\beta}$ is the OLS estimator of the linear regression

$$y = X\beta + u, \quad u \sim (0, \sigma^2 I_m).$$

- Recall the original model was $Y = X^T \beta + U$



Regression and Approximation

Problem

Approximate $c(\omega) : \Omega \rightarrow \mathbb{R}$ by means of $\phi_i(\omega) : \Omega \rightarrow \mathbb{R}$, $i = 1, \dots, k$

$$c(\omega) = \sum_{i=1}^k \phi_i(\omega) \beta_i + \varepsilon(\omega)$$

- ϕ_i basis functions, ε error function
- We need a norm to evaluate the error: $\|\varepsilon\|$
- Given a probability measure $P(\omega)$, define the scalar product

$$\langle u, v \rangle = E[uv] = \int_{\Omega} u(\omega)v(\omega)dP(\omega)$$

- $\|\varepsilon\| = \langle \varepsilon, \varepsilon \rangle^{1/2} = E[\varepsilon^2]^{1/2}$ is a semi-norm
- Back to the regression: $Y = X^T \beta + U$,

$$Y(\omega) = c(\omega), \quad X(\omega) = (\phi_1(\omega) \quad \phi_2(\omega) \quad \dots \quad \phi_k(\omega))^T$$



American (Bermudan) options

- Assume a flat market: $r = 0$ (i.e. work with forward prices).
- State vector $S_i = S_{t_i}$, $0 = t_0 \leq t_1 \leq \dots \leq t_n$.
- $h_i(s)$ payoff at the i th step (time t_i)
- $V_i(s)$ contract value at the i th step

Dynamic programming

$$V_n(s) = h_n(s)$$

$$V_i(s) = \max\left(h_i(s), E[V_{i+1}(S_{i+1})|S_i = s]\right), \quad i = n-1, n-2, \dots, 1.$$

- $C_i(s) = E[V_{i+1}(S_{i+1})|S_i = s]$: continuation value



American Options (cont.)

- Let define the r.v.s. $V_i = V_i(S_i)$, $C_i = C_i(S_i)$ and $h_i = h_i(S_i)$, thus

Dynamic programming

$$V_n = h_n, \quad C_i = E[V_{i+1}|S_i], \quad \text{and} \quad V_i = \max(h_i, C_i(S_i))$$

Regression Based Methods

Linearly approximate C_i by means of the regressors

$$X_i^T = (\phi_1(S_i) \quad \dots \quad \phi_k(S_i))$$

That is, find a β_i such that $C_i \simeq X_i^T \beta_i$.

- ϕ_1, \dots, ϕ_k are a set of properly chosen “basis functions”
- Note: $X_i = (X_{i1} \cdots X_{ik})$ is adapted to the filtration (This is what we really need)



Regression based methods

- Recall $C_i = E[V_{i+1}|S_i]$
- Regression problem: $C_i = X_i^T \beta_i + U$
- Least Squares: $E[C_i X_i]$ is needed
- The conditional expectation disappears, indeed

$$E[C_i X_i] = E[X_i E[V_{i+1}|S_i]] = E[X_i V_{i+1}]$$

- Suppose that MC provides a set of scenarios $S_i^\omega, V_{i+1}^\omega, \omega = 1, \dots, m$
- Set:

$$v_{i+1} = \begin{pmatrix} V_{i+1}^1 \\ V_{i+1}^2 \\ \vdots \\ V_{i+1}^m \end{pmatrix} \quad Z_i = \begin{pmatrix} \phi_1(S_i^1) & \phi_2(S_i^1) & \cdots & \phi_k(S_i^1) \\ \phi_1(S_i^2) & \phi_2(S_i^2) & \cdots & \phi_k(S_i^2) \\ \vdots & \vdots & & \vdots \\ \phi_1(S_i^m) & \phi_2(S_i^m) & \cdots & \phi_k(S_i^m) \end{pmatrix}$$

- Compute the OLS $\hat{\beta}_i$ of $v_{i+1} = Z_i \beta_i + u_i$ then $\hat{\beta}_i \rightarrow \beta_i$



Algorithm

- Compute the scenarios (forward)

$$S_i^\omega, \quad i = 1, \dots, n, \omega = 1, \dots, m.$$

- Compute the payoff at T :

$$V_n^\omega = h_n(S_n^\omega), \quad \omega = 1, \dots, m$$

- Compute the contract value going backward ($i = n - 1, n - 2, \dots, 1$):

- Compute Z_i and form v_{i+1}
- Compute $\hat{\beta}_i = (Z_i^T Z_i)^{-1} Z_i^T v_{i+1}$
- Compute $V_i^\omega = \max(h_i(S_i^\omega), \beta_i^T X_i^\omega)$



Algorithm (vector-matrix form)

Data structures (each row a scenario):

- $\mathbf{s}_i : m \times 1$, all the scenarios at step i
- $\mathbf{v}_i : m \times 1$, the value of the contract at step i at all the states
- $\mathbf{h}_i = h_i(\mathbf{s}_i) : m \times 1$, exercise values
- $\mathbf{Z}_i = (\phi_k(\mathcal{S}_i^\omega))_{\omega,k} : m \times k$ regressor matrix

Algorithm:

- Compute the scenarios (forward): $\mathbf{s}_{i+1} = \mathbf{s}_i + \dots$
- Compute the payoff at T : $\mathbf{v}_n = \mathbf{h}_n$
- Going Backward:
- Compute \mathbf{Z}_i
- Least Squares Approximation: $\hat{\beta}_i = (\mathbf{Z}_i^T \mathbf{Z}_i)^{-1} \mathbf{Z}_i^T \mathbf{v}_{i+1}$
- Compute $\mathbf{v}_i = \max(\mathbf{h}_i, \mathbf{Z}_i \hat{\beta}_i)$



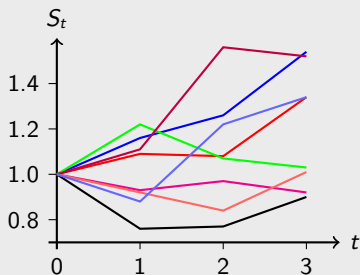
Longstaff & Schwartz Monte Carlo method

Example (Longstaff and Schwartz 2001)

Consider an American Put Option on a non-dividend-paying stock S_t with strike $K = 1.10$. The option is exercisable at times 1, 2 and 3. The riskless rate is $r = 6\%$.

Suppose to approximate (by MC) the dynamics with the 8 paths:

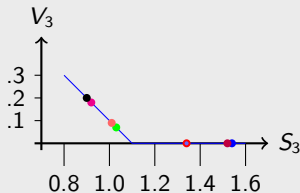
| ω | S_0 | S_1 | S_2 | S_3 |
|----------|-------|-------|-------|-------|
| 1 | 1.00 | 1.09 | 1.08 | 1.34 |
| 2 | 1.00 | 1.16 | 1.26 | 1.54 |
| 3 | 1.00 | 1.22 | 1.07 | 1.03 |
| 4 | 1.00 | 0.93 | 0.97 | 0.92 |
| 5 | 1.00 | 1.11 | 1.56 | 1.52 |
| 6 | 1.00 | 0.76 | 0.77 | 0.90 |
| 7 | 1.00 | 0.92 | 0.84 | 1.01 |
| 8 | 1.00 | 0.88 | 1.22 | 1.34 |



Example (cont.)

At time $t = 3$ the value of the contract is (= European): $V_3 = (K - S_3)^+$

| ω | S_0 | S_1 | S_2 | S_3 | V_3 |
|----------|-------|-------|-------|-------|-------|
| 1 | 1.00 | 1.09 | 1.08 | 1.34 | 0 |
| 2 | 1.00 | 1.16 | 1.26 | 1.54 | 0 |
| 3 | 1.00 | 1.22 | 1.07 | 1.03 | 0.07 |
| 4 | 1.00 | 0.93 | 0.97 | 0.92 | 0.18 |
| 5 | 1.00 | 1.11 | 1.56 | 1.52 | 0 |
| 6 | 1.00 | 0.76 | 0.77 | 0.90 | 0.20 |
| 7 | 1.00 | 0.92 | 0.84 | 1.01 | 0.09 |
| 8 | 1.00 | 0.88 | 1.22 | 1.34 | 0 |



- At time $t = 2$: $V_2 = \max((K - S_2)^+, E^Q[e^{-r} V_3 | S_2])$
- The problem is how to compute $E^Q[e^{-r} V_3 | S_2]$:
continuing to go forward, for each S_2 a new MC is needed
- LSM: $E^Q[e^{-r} V_3 | S_2] = \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$, with $X_i = f_i(S_2)$
Compute by regressing $e^{-r} V_3$ on X_1, X_2, \dots, X_k



Example (cont.)

At time $t = 2$.

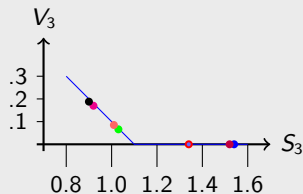
Estimate β_1, \dots, β_k in the model

$$E^Q[e^{-r} V_3 | S_2] = \beta_1 + S_2 \beta_2 + S_2^2 \beta_3$$

$$E^Q[Y | X] = X_1 \beta_1 + X_2 \beta_2 + X_3 \beta_3$$

| ω | Y | X_1 | X_2 | X_3 |
|----------|-------|-------|-------|-------|
| 1 | 0 | 1 | 1.08 | 1.166 |
| 2 | 0 | 1 | 1.26 | 1.588 |
| 3 | 0.066 | 1 | 1.07 | 1.145 |
| 4 | 0.170 | 1 | 0.97 | 0.941 |
| 5 | 0 | 1 | 1.56 | 2.434 |
| 6 | 0.188 | 1 | 0.77 | 0.593 |
| 7 | 0.085 | 1 | 0.84 | 0.706 |
| 8 | 0 | 1 | 1.22 | 1.489 |

| $\hat{\beta}_1$ | $\hat{\beta}_2$ | $\hat{\beta}_3$ |
|-----------------|-----------------|-----------------|
| 0.821 | -1.138 | 0.389 |



Outline

- 5 Binomial Model
- 6 Finite Differences
- 7 Monte Carlo
- 8 Compound Poisson Processes Simulation
- 9 Fourier Transform methods**
 - Exponential Damping
 - Time Value Approach



Fourier transforms and option pricing

European pricing formula is a convolution

European Call:

$$C_T(k) = e^{-rT} \int_{\mathbb{R}} (e^x - e^k)^+ q_T(x) dx = e^{k-rT} \int_{\mathbb{R}} \phi(x - k) q_T(x) dx$$

where $x = \log(S_T)$, $k = \log(K)$, $\phi(m) = (e^m - 1)^+$, q_T risk-neutral pdf

- Often only the characteristic function (cf) ϕ_T is known
- A convolution becomes a product in the Fourier space

$$c(y) = \int_{\mathbb{R}} a(x - y) b(x) dx \quad \xleftrightarrow{\mathcal{F}} \quad \hat{c}(\omega) = \hat{a}(\omega) \hat{b}(\omega)$$

- Problem: the Fourier transform of common payoffs does not exist



Definition (Fourier Transform)

$$\hat{g} = (\mathcal{F}g)(\omega) = \int_{\mathbb{R}} e^{i\omega x} g(x) dx \quad g = (\mathcal{F}^{-1}\hat{g})(x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-i\omega x} \hat{g}(\omega) d\omega$$

- Linearity
- Parseval: $\int_{\mathbb{R}} g(x) \overline{h(x)} dx = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\omega) \overline{\hat{h}(\omega)} d\omega$
- Convolution: $\mathcal{F}(g * h) = (\mathcal{F}g)(\mathcal{F}h)$, $(g * h)(x) = \int_{\mathbb{R}} g(s)h(x - s) ds$
- Scaling: $(\mathcal{F}g(ax))(\omega) = |a|^{-1}(\mathcal{F}g)(a^{-1}\omega)$
 - g even $\Leftrightarrow \hat{g}$ even
- Conjugation: $(\mathcal{F}\overline{g})(\omega) = \overline{(\mathcal{F}g)(-\omega)} = \overline{\hat{g}(-\omega)}$
 - g real $\Leftrightarrow \text{Re}(\hat{g})$ even, $\text{Im}(\hat{g})$ odd
 - g real and even $\Leftrightarrow \hat{g}$ real and even



Fourier Transform and Characteristic Functions

Definition (Characteristic Function)

Let X be an \mathbb{R} -valued r.v. with pdf $p(x)$, its characteristic function is

$$\phi_X(\omega) = \mathbb{E}[e^{i\omega X}] = \int_{\mathbb{R}} e^{i\omega x} p(x) dx, \quad \omega \in \mathbb{R}.$$

- Characteristic Function = FT of the pdf: $\phi_X = \mathcal{F}p$
- The expected value of $g(X)$ can be computed as

$$\mathbb{E}[g(X)] = \int_{\mathbb{R}} g(x)p(x)dx = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(-\omega)\phi_X(\omega)d\omega$$

- ◆ g should be square integrable



Option Pricing in Fourier Space

problem

Given ϕ_T the CF of X , compute the price of a Call ^a

$$C_T(k) = E[(e^x - e^k)^+] = \int_{\mathbb{R}} (e^x - e^k)^+ q_T(x) dx$$

^aassume for simplicity $r = 0$

Problem: C_T is not L^2 , it does not have a FT

Solutions:

- Exponential Damping ([4])
 - Numerically unstable when far from the money or near to expiration
- Time Value ([4])
- Other approaches: Chen and Scott
- In the following we Follow [4]



Outline

- 5 Binomial Model
- 6 Finite Differences
- 7 Monte Carlo
- 8 Compound Poisson Processes Simulation
- 9 Fourier Transform methods**
 - Exponential Damping
 - Time Value Approach



Exponential Damping

$$C_T(k) = \int_{\mathbb{R}} (e^x - e^k)^+ q_T(x) dx$$

- $C_T(k) \notin L^2$, the FT is not defined
- Damping: consider the FT of $c_T(k) = e^{\alpha k} C_T(k)$

$$\psi_T(\omega) = \int_{\mathbb{R}} e^{i\omega k} c_T(k) dk$$

- so that

$$C_T(k) = e^{-\alpha k} \frac{1}{2\pi} \int_{\mathbb{R}} e^{-i\omega k} \psi_T(\omega) d\omega$$

- Let consider $\psi_T(\omega)$



$$\begin{aligned}
\psi_T(\omega) &= \int_{-\infty}^{+\infty} e^{i\omega k} c_T(k) dk \\
&= \int_{-\infty}^{+\infty} e^{i\omega k} \int_{-\infty}^{+\infty} e^{\alpha k} (e^x - e^k)^+ q_T(x) dx dk \\
&= \int_{-\infty}^{+\infty} e^{i\omega k} \int_k^{+\infty} (e^{x+\alpha k} - e^{(1+\alpha)k})^+ q_T(x) dx dk \\
&= \int_{-\infty}^{+\infty} q_T(x) \int_{-\infty}^x e^{i\omega k} (e^{x+\alpha k} - e^{(1+\alpha)k})^+ dk dx \\
&= \int_{-\infty}^{+\infty} q_T(x) \left(\frac{e^{(\alpha+1+i\omega)x}}{\alpha+i\omega} - \frac{e^{(\alpha+1+i\omega)x}}{\alpha+1+i\omega} \right) dx \\
&= \frac{1}{(\alpha+i\omega)(\alpha+1+i\omega)} \int_{-\infty}^{+\infty} q_T(x) e^{(\alpha+1+i\omega)x} dx \\
&= \frac{1}{(\alpha+i\omega)(\alpha+1+i\omega)} \phi_T(\omega - i(\alpha+1))
\end{aligned}$$



Choosing the dumping factor α

$$\psi_T(\omega) = \frac{1}{(\alpha + i\omega)(\alpha + 1 + i\omega)} \phi_T(\omega - i(\alpha + 1))$$

- $\psi_T(\omega)$ is the FT of $c_T(k)$
- Sufficient condition for $c_T(k)$ to be L^2 is: $\psi_T(0) < \infty$

$$\psi_T(0) = \phi_T(-i(\alpha + 1)) = E[\exp(i \overbrace{(-i)(\alpha + 1)}^{\omega} X)] = E[S_T^{\alpha+1}]$$

- Thus, it is sufficient that $E[S_T^{\alpha+1}] < \infty$
- Choose α to a quarter of the value that guarantee the bound
- Furthermore, the domain needs to be truncated

$$C_T(k) = e^{-\alpha k} \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-i\omega k} \psi_T(\omega) d\omega$$

more on Carr and Madan (1999).



Outline

- 5 Binomial Model
- 6 Finite Differences
- 7 Monte Carlo
- 8 Compound Poisson Processes Simulation
- 9 Fourier Transform methods**
 - Exponential Damping
 - Time Value Approach



Time Value Approach

- Assume for simplicity: $r = 0$ and $S_0 = 0$
- Let $C_T = E[(e^x - e^k)^+]$, $P_T = E[(e^k - e^x)^+]$ and

$$\begin{aligned} z_T(k) &= P_T(k)\chi_{\{k < 0\}} + C_T(k)\chi_{\{k \geq 0\}} \\ &= \chi_{\{k < 0\}} \int_{-\infty}^k (e^k - e^x)q_T(x)dx + \chi_{\{k \geq 0\}} \int_k^{\infty} (e^x - e^k)q_T(x)dx \end{aligned}$$

- Note that: $z_T = C_T(k) - (e^x - e^k)^+$ is the “time value” of the Call
- Consider the FT of z_T :

$$\zeta_T(\omega) = \int_{\mathbb{R}} e^{i\omega k} z_T(k) dk, \quad z_T(k) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-i\omega k} \zeta_T(\omega) d\omega$$



$$z_T(k) = P_T(k)\chi_{\{k<0\}} + C_T(k)\chi_{\{k\geq 0\}}$$

- Let consider the FT of the first term:

$$\begin{aligned} \int_{\mathbb{R}} e^{i\omega k} P_T(k)\chi_{\{k<0\}} dk &= \int_{-\infty}^0 e^{i\omega k} \int_{-\infty}^k (e^k - e^x) q_T(x) dx dk \\ &= \int_{-\infty}^0 q_T(x) \int_x^0 (e^{(1+i\omega)k} - e^{x+i\omega k}) dk dx \\ &= \int_{-\infty}^0 q_T(x) \int_x^0 (e^{(1+i\omega)k} - e^{x+i\omega k}) dk dx \\ &= \int_{-\infty}^0 q_T(x) \left(\frac{1}{1+i\omega} - \frac{e^x}{i\omega} + \frac{e^{(1+i\omega)x}}{\omega^2 - i\omega} \right) dx \end{aligned}$$

analogously

$$\int_{\mathbb{R}} e^{i\omega k} P_T(k)\chi_{\{k<0\}} dk = \int_0^{+\infty} q_T(x) \left(\frac{1}{1+i\omega} - \frac{e^x}{i\omega} + \frac{e^{(1+i\omega)x}}{\omega^2 - i\omega} \right) dx$$



- Thus:

$$\zeta_T(\omega) = \int_{-\infty}^{+\infty} q_T(x) \left(\frac{1}{1+i\omega} - \frac{e^x}{i\omega} + \frac{e^{(1+i\omega)x}}{\omega^2 - i\omega} \right) dx$$

since $E[S_T] = E[e^x] = 1$

$$\zeta_T(\omega) = \frac{1}{1+i\omega} - \frac{1}{i\omega} + \frac{\phi_T(\omega - i)}{\omega^2 - i\omega} = \frac{1}{\omega^2 - i\omega} (1 + \phi_T(\omega - i))$$

- so that

$$z_T(k) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-i\omega k} \zeta_T(\omega) d\omega = \frac{1}{2\pi} \int_{\mathbb{R}} \frac{1 + \phi_T(\omega - i)}{\omega^2 - i\omega} e^{-i\omega k} d\omega$$

- z_T computed by the FFT
- Unstable at the money (near $k = 0$) when $T \rightarrow 0$



Fixing instabilities in Time-Value approach

- $z_T(k)$ unstable for $k \simeq 0$
- Consider the FT of $\sinh(\alpha k)z_T(k)$ ($\sinh(x)$ is null at $x = 0$)

$$\begin{aligned}
 \gamma_T(\omega) &= \int_{\mathbb{R}} e^{i\omega k} \sinh(\alpha k) z_T(k) dk \\
 &= \int_{\mathbb{R}} e^{i\omega k} \frac{e^{\alpha k} - e^{-\alpha k}}{2} z_T(k) dk \\
 &= \frac{1}{2} \int_{\mathbb{R}} e^{(i\omega + \alpha)k} z_T(k) dk - \frac{1}{2} \int_{\mathbb{R}} e^{(i\omega - \alpha)k} z_T(k) dk \\
 &= \frac{\zeta_T(\omega - i\alpha) - \zeta_T(\omega + i\alpha)}{2}
 \end{aligned}$$

- Then $z_T(k)$ is computed as

$$z_T(k) = \frac{1}{\sinh(\alpha k)} \frac{1}{2\pi} \int_{\mathbb{R}} e^{i\omega k} \gamma_T(\omega) d\omega$$



Discrete Fourier Transforms

- Consider the sequence u_j , $j = 1, \dots, N$
- The Discrete Fourier Transform (DFT) of u is the sequence

$$w_s = \sum_{j=1}^N e^{-ih(j-1)(s-1)} u_j \quad s = 1, \dots, N$$

where $h = 2\pi/N$

- Briefly: $w = DFT(u)$
- Typically $N = 2^n$
- FFT computes the whole sequence w in $O(N \log N)$



From FTs to DFTs

- Let consider the inverse FT:

$$g(k) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-i\omega k} \psi(\omega) d\omega$$

- Assuming $g(k) \in \mathbb{R}$ implies $\text{Re}(\psi(\omega))$ even and $\text{Im}(\psi(\omega))$ odd:

$$\begin{aligned} g(k) &= \frac{1}{\pi} \int_0^{\infty} e^{-i\omega k} \psi(\omega) d\omega \simeq \frac{1}{\pi} \int_0^a e^{-i\omega k} \psi(\omega) d\omega \\ &\simeq \frac{1}{\pi} \sum_{j=1}^N e^{-i\omega_j k} \psi(\omega_j) \eta \end{aligned}$$

where $\eta = a/N$ and $\omega_j = \eta(j-1)$

- Set $k = -\frac{N}{2}\lambda + \lambda(s-1)$

$$g(k) \simeq g_s = \frac{1}{\pi} \sum_{j=1}^N e^{-i\eta\lambda(j-1)(s-1)} e^{i\frac{N}{2}\lambda\eta(j-1)} \psi(\omega_j) \eta$$



- Finally:

$$\begin{aligned}
 g(k) &\simeq g_s = \frac{1}{\pi} \sum_{j=1}^N e^{-i\eta\lambda(j-1)(s-1)} e^{i\frac{N}{2}\lambda\eta(j-1)} \psi(\omega_j) \eta \\
 &= \sum_{j=1}^N e^{-ih(j-1)(s-1)} \frac{\eta}{\pi} e^{i\pi(j-1)} \psi(\omega_j)
 \end{aligned}$$

when $\eta\lambda = \frac{2\pi}{N} = h$

- thus

$$g = DFT(u), \quad \text{where} \quad u_j = \frac{\eta}{\pi} e^{i\pi(j-1)} \psi(\omega_j) = \frac{\eta}{\pi} (-1)^{j-1} \psi(\omega_j).$$

- Notice:

$$\begin{aligned}
 k &\in \left[-\frac{N}{2}\lambda, \left(\frac{N}{2} - 1\right)\lambda\right] = \left[-\frac{\pi}{\eta}, \frac{\pi}{\eta}\left(1 - \frac{2}{N}\right)\right] \quad \text{step } \lambda \\
 \omega_j &\in \left[0, \frac{2\pi}{\lambda}\left(1 - \frac{1}{N}\right)\right] = [0, (N-1)\eta] \quad \text{step } \eta
 \end{aligned}$$



Exercise

Compute the price of a Call under the CGMY or tempered Lévy model. The CF of log-returns in the CGMY model is given by

$$\frac{1}{T} \log \phi_T(\omega) = -C \Gamma(-Y) \left((M - i\omega)^Y - M^Y + (G + i\omega)^Y - G^Y \right) - i\omega \kappa$$

$$\kappa = C \left(M(\Gamma(-Y)Y + \Gamma(1 - Y, M)) - G(\Gamma(-Y)Y + \Gamma(1 - Y, G)) \right),$$

where $\Gamma(a, b) = \int_b^\infty x^{a-1} e^{-x} dx$ and $\Gamma(a) = \Gamma(a, 0)$

Implement it as the matlab function

```
function Call = cgmy_fft(S0,K, C,G,M,Y, T, N)
```




Characteristic exponents for Lèvy models


| Model | Characteristic Exponent ($\frac{1}{T} \log \phi_T(\omega)$) |
|-----------|--|
| GBM | $i(\mu - \frac{1}{2}\sigma^2)\omega - \frac{1}{2}\sigma^2\omega^2$ |
| Merton JD | $i(\mu - \frac{1}{2}\sigma^2)\omega - \frac{1}{2}\sigma^2\omega^2 + \lambda(e^{i\bar{\mu}\omega - \bar{\sigma}^2\omega^2/2} - 1)$ |
| Kou JD | $i(\mu - \frac{1}{2}\sigma^2)\omega - \frac{1}{2}\sigma^2\omega^2 + i\omega\lambda(\frac{p}{\eta_+ - i\omega} - \frac{1-p}{\eta_- - i\omega})$ |
| VG | $\kappa^{-1} \log(1 - i\mu\kappa\omega + \frac{1}{2}\sigma^2\kappa\omega^2)$ |
| NIG | $\kappa^{-1}(1 - \sqrt{1 - 2i\mu\kappa\omega + \sigma^2\kappa\omega^2})$ |



Bibliography I

 *Getting Started with MATLAB.*
The shortest path to learn and use Matlab.

 Thomas Bjork.
Arbitrage theory in continuous time.
Oxford University Press, 2004.

 Paolo Brandimarte.
Numerical methods in finance and economics.
Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, second edition, 2006.
A MATLAB-based introduction.
The new edition contains added material on Optimizations and Stochastic Programming.



Bibliography II



P. Carr and D. Madan.

Option pricing and the fast fourier transform.

Journal of Computational Finance, 2(4):61–73, 1999.

This paper introduces the two reference methods (for the financial community) for pricing European Options in the frequency space



Gerard Cornuejols and Reha Tütüncü.

Optimization methods in finance.

Mathematics, Finance and Risk. Cambridge University Press, Cambridge, 2007.

A good book on optimization methods applied to finance. Treats topics like static replication, arbitrage bounds and robust portfolio selection which are usually not considered in courses on quantitative finance.



Bibliography III



Manfred Gilli, Dietmar Maringer, and Enrico Schumann.

Numerical methods and optimization in finance.

Amsterdam: Elsevier/Academic Press. xv, 584 p. \$ 99.95 , 2011.

A reference book for those that need to solve non-standard optimization problems arising in Finance and are interested into heuristics.



Paul Glasserman.

Monte Carlo Methods in Finance.

Springer, 2004.

The reference book on Monte Carlo Methods for Finance



Bibliography IV



Gene H. Golub and Charles F. Van Loan.

Matrix computations.

Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

The bible of Numerical Linear Algebraists



Desmond J. Higham.

An algorithmic introduction to numerical simulation of stochastic differential equations.

SIAM Rev., 43(3):525–546 (electronic), 2001.



Bibliography V



Desmond J. Higham.

Nine ways to implement the binomial method for option valuation in MATLAB.

SIAM Rev., 44(4):661–677 (electronic) (2003), 2002.

The paper shows that explicit formulae are not so fast (or may be not so explicit).



John C. Hull.

Option Futures and Other Derivatives.

Prentice Hall, 1993-2011.

A must read



Bibliography VI



Andrea Pascucci.

PDE and martingale methods in option pricing, volume 2 of *Bocconi & Springer Series*.

Springer, Milan, 2011.

I think that the notation used in these slides is (almost) consistent with the one in this book.



Rüdiger U. Seydel.

Tools for Computational Finance.

