樹狀結構的評價法及改進

計算複雜度分析
課堂練習：使用零息利率計算債券價格程式分析及修改
二元樹評價理論
使用二元樹評價選擇權
課堂練習：使用二元樹評價美式賣權
使用組合數學評價選擇權
計算複雜度比較
課堂練習：執行速度比較

以債券價格計算爲例
分析程式執行所需的時間

- 考慮債券價格的計算
  - 假定單期利率爲r
  - 每一期支付coupon c,共付n期
  - 到期日還本100元

債券價格= $c\times(1+r)^{-1}+c\times(1+r)^{-2}+\ldots+c\times(1+r)^{-n}+100\times(1+r)^{-n}$

程式碼分析

見 BondValue project

```
#include <stdio.h>
int main()
{
        int n;
        float c, r, Value=0,Discount;
        scanf("%d",&n);
        scanf("%f",&c);
        scanf("%f",&r);
        for(int i=1;i<=n;i=i+1)
        {
         Discount=1;
         for(int j=1;j<=i;j++)
         {
          Discount=Discount/(1+r);
         }
         Value=Value+Discount*c;
         if(i==n)
         {
          Value=Value+Discount*100;
         }
        }
    printf("BondValue=%f",Value);
return 0;
}
```

只執行一次

計算第i次payoff的現值,並累加到Value中:
這塊程式碼應執行n次

計算 $(1+r)^{-i}$
在計算第i次payoff時,這塊程式碼執行i次
總共需執行1+2+3+4+…+n
共 $n(n+1)/2$ 次

只執行一次

程式執行時間的分析

- 假定每一段程式碼的執行時間
  - 綠色方塊的執行時間=G
  - 藍色方塊的執行時間=B
  - 粉紅色方塊的執行時間=P
  - 其他部分=O
- 整個程式的執行時間估計如下: $\frac{n(n+1)}{2}G+nB+P+O$
- 當n變很大時,整個執行時間主要受 $\frac{n(n+1)}{2}G$ 影響

O-notation

- 程式的計算複雜度可用O-notation表示
- 計算複雜度爲O(f(n))的程式,代表當n夠大時,程式的計算時間小於$cf(n)$
  - c爲一固定常數
- 以上述程式爲例,其計算複雜度爲 $O(n^2)$
- $O(n) \succ O(n^2) \succ O(n^3) \succ ... \succ O(2^n)$
- O-notation的分析,可簡易地估計n很大時,程式需耗費的計算時間的成長速度

債券計算程式化簡 $O(n^2) \rightarrow O(n)$

```
#include <stdio.h>
int main()
{
        int n;
        float c, r, Value=0,Discount;
        scanf("%d",&n);
        scanf("%f",&c);
        scanf("%f",&r);
        Discount=1;
        for(int i=1;i<=n;i=i+1)
        {
         Discount=pow(1+r,-double(i));
         Value=Value+Discount*c;
         if(i==n)
         {
          Value=Value+Discount*100;
         }
        }
  printf("BondValue=%f",Value);

return 0;
}
```

計算第i次payoff的現值,並累加到Value中:
這塊程式碼應執行n次

只需執行n次,就可達到前述程式的效果
使用 pow()函式
pow()複雜度爲 constant time

債券計算程式化簡 $O(n) \rightarrow O(n)$

```
#include <stdio.h>
int main()
{
        int n;
        float c, r, Value=0,Discount;
        scanf("%d",&n);
        scanf("%f",&c);
        scanf("%f",&r);
        Discount=1;
        for(int i=1;i<=n;i=i+1)
        {
         Discount=Discount/(1+r);
         Value=Value+Discount*c;
         if(i==n)
         {
          Value=Value+Discount*100;
         }
        }
  printf("BondValue=%f",Value);

return 0;
}
```

計算第i次payoff的現值,並累加到Value中:
這塊程式碼應執行n次

只需執行n次計算,就可達到前述程式的效果

演算法討論的重點:
如何撰寫結果正確,計算時間短的程式

課堂練習:程式複雜度分析
Zero Rate計算債券價格程式

```
#include<stdio.h>
int main()
{
  int I;
  float ZeroRate[5];
  float C;
  scanf("%f",&C);
  printf("輸入Zero rate:\n");
  for( i=0;i<5;i=i+1)
  {
   scanf("%f",&ZeroRate[i]);
  }
  float BondValue=0;
  for(i=0;i<5;i=i+1)
  {
   float Discount=1;
   for(int j=0;j<=i;j=j+1)
   {
    Discount=Discount/(1+ZeroRate[i]);
   }
   BondValue=BondValue+C*Discount;
   if(i==4)
   {BondValue=BondValue+100*Discount;}
  }
   printf("債券價格=%f", BondValue);
return 0;
}
```

見 ZeroRate project

假定期數改用n來表示

藍色區塊應執行n次

計算此區塊需執行的次數
這一部分的計算是否可化簡?

數值模型

- 假定在市場上無套利機會,且market is complete
  - 存在唯一的風險中立機率 Q
- 在這種市場中,商品的評價可用取期望值的方式化簡
  - 假定有一個商品在到期日T時的payoff爲C(T),利率爲r
  - 在時間t時,該商品的價格爲: $e^{-r(T-t)}\mathrm{E}_{\mathrm{Q}}(\mathrm{C(T)})$
  - 以一個履約價格爲X,標的物價格爲S(t)的賣權爲例
    - 賣權價格= $e^{-r(T-t)}\mathrm{E}_{\mathrm{Q}}(\mathrm{X\text{-}S(T)})^{+}$
- 建立一個數值模型,模擬在風險中立機率下,標的物價格的變化
  - 藉由該數值模型,可求出衍生物payoff的期望值→商品價格

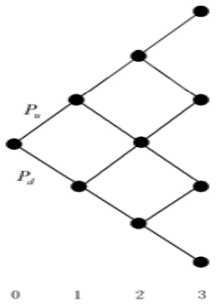
使用二元樹評價選擇權

- 當標的物的價格服從對數常態隨機過程

  $S(t) = S(0)e^{(r-\frac{1}{2}\sigma^2)t+\sigma B(t)}$
  - 可使用n期二元樹來模擬標的物價格
  - $n \to \infty$,二元樹逼近對數常態隨機過程
  - $n \to \infty$,評價結果逼近理論價格

二元樹模型

- 回顧二元樹的架構:
  - 右圖爲三期的模型
  - 每一期價格只能上升或下降
  - 假定上升變 e^u 倍
    - 機率爲 P_u
  - 假定下降變 e^d 倍
    - 機率爲 P_d



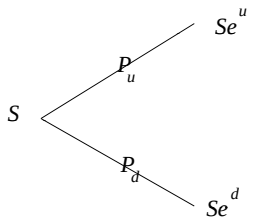
- 第n期的標的物價格可寫成 $S(0)e^{W(n)}$
  - $W(n) = I_1 + I_2 + \ldots + I_n$
  - $I_i = u$ or d

二元樹模型的建立

- 假定$I_1, I_2, \ldots, I_n$獨立,則根據中央極限定理,當 $n \to \infty$
  - W(n)會逼近常態分配
- 標的物價格過程:$S(T) = S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma B(T)}$
  - $E(W_n) = \left(r-\frac{1}{2}\sigma^2\right)T$
  - $\mathrm{Var}(W_n) = \sigma^2 T$
- 所以 $I_1, I_2, \ldots, I_n$ 要滿足下列條件:
  - $\Delta t \equiv \frac{T}{n}$
  - $E(I_n) = \left(r-\frac{1}{2}\sigma^2\right)\Delta t$
  - $\mathrm{Var}(I_n) = \sigma^2 \Delta t$

二元樹模型的建立

- 考慮單期模型:
  - $P_u \times u + P_d \times d = \left(r - \frac{1}{2}\sigma^2\right)\Delta t \equiv m$
  - $P_u \times (u-m)^2 + P_d \times (d-m)^2 = \sigma^2 \Delta t$
  - $P_u + P_d = 1$

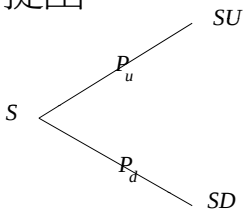


- 三個方程式,但是有四個未知數,可根據對評價的模型的要求加入不同等式
  - CRR model: $e^u \times e^d = 1$
    - Remark:在CRR模型設定中,爲了公式簡化,

$$P_u \times (u-m)^2 + P_d \times (d-m)^2 \to \sigma^2 \Delta t$$

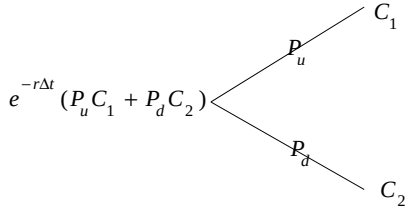
使用二元樹評價選擇權
CRR 二元樹模型

- 由 Cox, Ross, 和Rubinstein提出
- 模型設定如右
  - $U = e^u = e^{\sigma\sqrt{\Delta t}}$ $D = e^d = e^{-\sigma\sqrt{\Delta t}}$



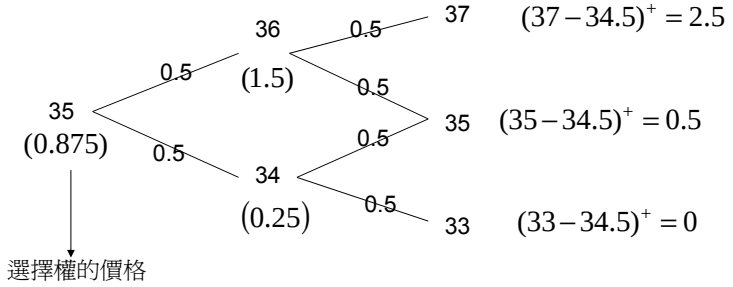
$$P_u = \frac{e^{r\Delta t} - D}{U - D} \qquad P_d = 1 - P_u$$

- 權利金的計算如右:



使用二元樹評價選擇權
Backward Induction

- 以虛擬兩期的模型爲例 (令r=0)
  - 先計算最後一期的價格,然後再往前推



使用二元樹評價選擇權
Backward Induction

- 定義一個陣列,長度>(n+1)
- 計算最後一期的payoff,放入陣列中

2.5	0.5	0

- 利用第二期價格求算第一期價格

2.5	0.5	0
1.5	0.5	0
1.5	0.25	0

使用二元樹評價選擇權
Backward Induction

- 利用第一期的價格求算第零期的價格

1.5	0.25	0
0.875	0.25	0

- 選擇權的價格: 0.875

使用二元樹評價選擇權
程式流程設計

- 程式流程如下:
  - 1. 輸入資料: 標的物價格(S),標的物波動率(Sigma), 無風險利率(r),履約價格(X),時間長度(T),期數(n).
  - 2. 計算CRR二元樹的相關參數
  - 3. 求算選擇權在最後一期的payoff
  - 4. Backward Induction
  - 5. 輸出評價結果

請參見CRR project

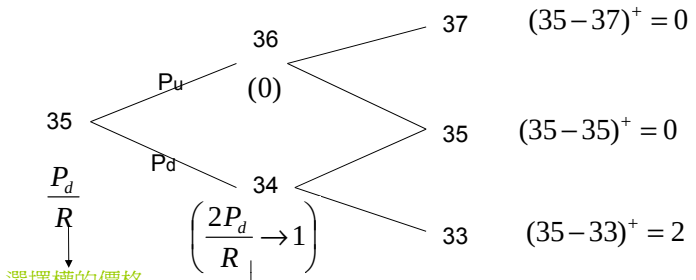
程式架構分析

```
void main()
{
  int i;
  //輸入資料
  float S,T,X,r,Sigma;
  int n;  //期數
  printf("輸入標的物價格:");
  scanf("%f",&S);
  printf("輸入到期日:");
  scanf("%f",&T);
  printf("輸入履約價格:");
  scanf("%f",&X);
  printf("輸入無風險利率:");
  scanf("%f",&r);
  printf("輸入標的物價格波動率:");
  scanf("%f",&Sigma);
  printf("輸入期數:");
  scanf("%d",&n);
  //計算CRR二元樹的相關參數
  double U, D, Pu,Pd,DeltaT;
  DeltaT=T/n;
  U=exp(Sigma*sqrt(DeltaT));
  D=exp(-Sigma*sqrt(DeltaT));
  Pu=(exp(r*DeltaT)-D)/(U-D);
  Pd=1-Pu;
  //計算最後一期的payoff
  double Array[500];
  double CurrentS=S*pow(U,n);
  for(int i=0;i<=n;i=i+1)
  {
   Array[i]=Max(CurrentS-X,0);
   CurrentS=CurrentS*D*D;
  }
  //Backward Induction
  for(i=n-1;i>=0;i=i-1)
  {
   for(int j=0;j<=i;j++)
   {
    Array[j]=exp(-r*DeltaT)*(Array[j]*Pu+Array[j+1]*Pd);
   }
  }
  //輸出評價結果
  printf("選擇權價格=%f",Array[0]);

  return 0;
}
```

課堂練習:美式賣權評價

- 將上述程式改成評價賣權的程式
  - 賣權的payoff: $(X - S(T))^+$
- 考慮選擇權提前履約的問題(令X=35),一期利率爲R>1 (p>0.5)



```
void main()
{
  int i;
  //輸入資料
  float S,T,X,r,Sigma;
  int n;  //期數
  printf("輸入標的物價格:");
  scanf("%f",&S);
  printf("輸入到期日:");
  scanf("%f",&T);
  printf("輸入履約價格:");
  scanf("%f",&X);
  printf("輸入無風險利率:");
  scanf("%f",&r);
  printf("輸入標的物價格波動率:");
  scanf("%f",&Sigma);
  printf("輸入期數:");
  scanf("%d",&n);
  //計算CRR二元樹的相關參數
  double U, D, Pu,Pd,DeltaT;
  DeltaT=T/n;
  U=exp(Sigma*sqrt(DeltaT));
  D=exp(-Sigma*sqrt(DeltaT));
  Pu=(exp(r*DeltaT)-D)/(U-D);
  Pd=1-Pu;
```

計算複雜度分析

```
  //計算最後一期的payoff
  double Array[500];
  double CurrentS=S*pow(U,n);
  for(int i=0;i<=n;i=i+1)
  {
   Array[i]=Max(CurrentS-X,0);
   CurrentS=CurrentS*D*D;
  }
  //Backward Induction
  for(i=n-1;i>=0;i=i-1)
  {
   for(int j=0;j<=i;j++)
   {
    Array[j]=exp(-r*DeltaT)*(Array[j]*Pu+Array[j+1]*Pd);
   }
  }
  //輸出評價結果
  printf("選擇權價格=%f",Array[0]);

return 0;
}
```

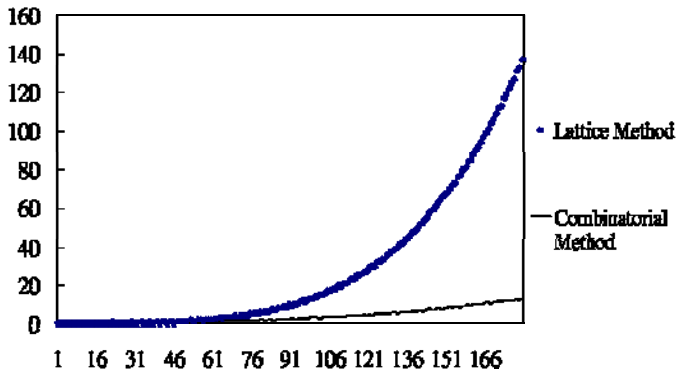
淺藍色部分只需執行一次即可,
淺綠部分需執行n次
淺黃部分需執行n+(n-1)+(n-2)+…+1
共n(n+1)/2次

使用二元樹評價選擇權

計算複雜度分析

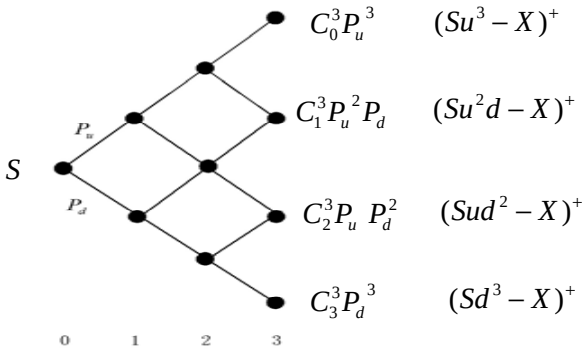
- $n \to \infty$,二元樹逼近對數常態隨機過程,評價結果會接近真實價值
- 分析n增大時,程式的複雜度變得十分重要
- 上述程式的計算時間在n變大時,主要受backward induction的影響
- 程式計算複雜度= $O(n^2)$
- 尋找一個能達到相同功能且計算複雜度較低的程式可提高計算效率

使用二元樹評價選擇權
計算時間的比較



使用組合數學
處理二元樹的評價問題

- 觀察一個三期的二元樹模型:



選擇權價格 $= e^{-rT} \sum_{i=0}^{3} C_i^3 P_u^{3-i} P_d^i (Su^{3-i} d^i - X)^+$

使用組合數學
處理二元樹的評價問題

- 在n期的模型中,選擇權價格可表示如下:
  - $e^{-rT}\sum_{i=0}^{n} C_i^n P_u^{n-i} P_d^i (Su^{n-i}d^i - X)^+$

- 如果 $C_i^n P_u^{n-i} P_d^i (Su^{n-i}d^i - X)^+$ 可在固定時間內算出,則整個程式的計算複雜度可表現成O(n)
- 觀察第i個點和第i+1個點的機率和payoff,發現

$$C_i^n P_u^{n-i} P_d^i \times \left(\frac{n-i}{i+1} \times \frac{P_d}{P_u}\right) = C_{i+1}^n P_u^{n-i-1} P_d^{i+1}$$

$$Su^{n-i}d^i \times \frac{d}{u} = Su^{n-i-1}d^{i+1}$$

兩者皆可在固定時間內求出

使用組合數學
處理二元樹的評價問題

- 原來處理最後一期的報酬和backward induction的程式可縮減如下:

參照Combinatorial Project

```
double CurrentS=S*pow(U,n);
 double CurrentProb=pow(Pu,n);
 double OptionValue=0;
 for(int i=0;i<=n;i=i+1)
 {
  OptionValue=OptionValue+exp(-r*T)*CurrentProb*Max(CurrentS-X,0);
  CurrentS=CurrentS*D/U;
  CurrentProb=CurrentProb*(n-i)/(i+1)*Pd/Pu;
 }
```

- 淺藍區塊的程式只需執行n次,所以整個程式的計算複雜度降爲O(n)

課堂演練

- 討論是否可用組合數學的方式評價美式選擇權

- 考慮期數很大,程式是否能正常運作?
  - Overflow / Underflow problem
  - 使用log 運算

```
double a=pow(0.1,1000)*pow(0.1,-1000);.
printf("%lf",a);
```

換成

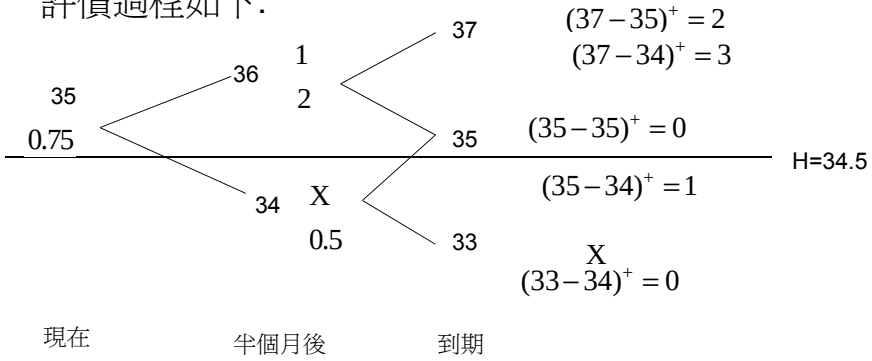
```
double a=exp(1000*log(0.1)-1000*log(0.1));
printf("%lf",a);
```

在CRR樹上評價新奇選擇權

- 新奇選擇權的價格常會受標的物的價格路徑而影響
  - 可在每個節點上加上適當的狀態變數,來記憶不同狀況下的選擇權價格
  - 重設選擇權(Reset option):履約價格會在標的物價格碰到某一界限時重設
    - $$\text{payoff} = \begin{cases} (S(T)-X)^+ & \text{if } S(t)>H \quad \forall t\in(0,T) \\ (S(T)-B)^+ & \text{if } \exists t\in(0,T)\ S(t)\le H \end{cases}$$

重設選擇權評價

- 選擇權在時間t的價格,受到標的物價格在時間t之前是否碰觸預設價格H影響→每個節點最多需放入兩個狀態變數
- 以下圖爲例,假定H=34.5,重設的履約價格爲34,則評價過程如下:

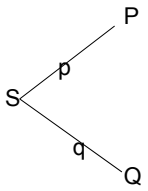


課堂演練

- 修改 CRR1 project,用來評價重設選擇權
  - 增加輸入 B(重設後的履約價格), H(重設界限)
  - Array:改成二維陣列: Array[500][2];
    Array[*][0]→未重設的選擇權價格
    Array[*][1]→重設後的選擇權價格
  - 最後一期的價格
    Array[i][0]=Max(CurrentS-X,0);
    Array[i][1]=Max(CurrentS-B,0);
  - 修改Backward induction
    - 分別考慮節點i的價格大於和小於H的處理 (See next slide)

課堂演練

- Case 1: S<=H
  - Array[i][0] → Useless
  - Array[i][1]= $\frac{p\times V(P,1)+q\times V(Q,1)}{R}$
- Case 2: Q<=H<S
  - Array[i][0]= $\frac{p\times V(P,0)+q\times V(Q,1)}{R}$
  - Array[i][1]= $\frac{p\times V(P,1)+q\times V(Q,1)}{R}$
- Case 3:Otherwise
  - Array[i][0]= $\frac{p\times V(P,0)+q\times V(Q,0)}{R}$
  - Array[i][1]= $\frac{p\times V(P,1)+q\times V(Q,1)}{R}$



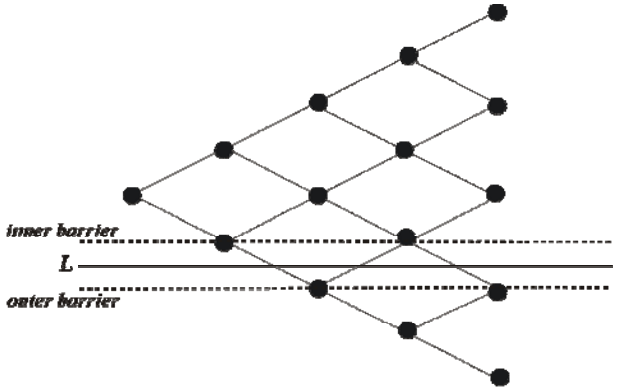
障礙選擇權

- 選擇權是否失效(或生效)視標的物在到期日之前的價格是否曾經到達某障礙價格。
- 依標的物的價格碰到障礙價格而分
  - 出局選擇權(knock-out option)
  - 入局選擇權(knock-in option)
- 依標的物期初價格和障礙價格之關係而分
  - Up option
  - Down option
- 下出局選擇權(down-and-out option)

$$\text{payoff}=\begin{cases}(S(T)-X)^+ & \text{if } S(t)>L \quad \forall t\in(0,T)\\ 0 & \text{if } \exists t\in(0,T)\ S(t)\le L\end{cases}$$

利用CRR樹來評價障礙選擇權

- Backward induction的過程會因爲標的物的價格碰到障礙而影響選擇權價值
- Example figure ：inner barrier vs. outer barrier



使用二元樹評價出局障礙選擇權
程式流程設計

- 程式流程如下:
  - 1. 輸入資料: 標的物價格(S),標的物波動率(Sigma), 無風險利率(r),履約價格(X),時間長度(T),期數(n).
  - 2. 計算CRR二元樹的相關參數
  - 3. 計算有效礙障步數 (走幾步會到有效障礙)
    - Up: inner barrier
    - Down: outer barrier

    u_power= (int)floor(log(H/S)/log(U));
  - 4. 求算選擇權在最後一期的報酬
    - 若有節點落在有效障礙上，該節點的選擇權價值設爲0
  - 5. Backward Induction
    - 檢查是否有節點落在有效障礙上，並將設該節點的價值設爲0
  - 6. 輸出評價結果

請參見BarrierCRR project

CRR barrier：Out option

```
int main()
{
  int i,j;
  //輸入資料
  float S,T,X,r,Sigma;
  int n;  //期數
  printf("輸入標的物價格:");
  scanf("%f",&S);
  printf("輸入到期日:");
  scanf("%f",&T);
  printf("輸入履約價格:");
  scanf("%f",&X);
  printf("輸入無風險利率:");
  scanf("%f",&r);
  printf("輸入標的物價格波動率:");
  scanf("%f",&Sigma);
  printf("輸入期數:");
  scanf("%d",&n);
  //計算CRR二元樹的相關參數
  double U, D, Pu,Pd,DeltaT;
  DeltaT=T/n;
  U=exp(Sigma*sqrt(DeltaT));
  D=exp(-Sigma*sqrt(DeltaT));
  Pu=(exp(r*DeltaT)-D)/(U-D);
  Pd=1-Pu;
  //計算有效障礙步數
  int u_power;
  u_power= (int)floor(log(H/S)/log(U));
  //計算最後一期的payoff
  double CurrentS=S*pow(U,n);
  for(int i=0;i<=n;i=i+1)
  {
  Array[i]=Max(CurrentS-X,0);
   CurrentS=CurrentS*D*D;
  }
  if ( (((n-u_power)%2)==0) &&
        (u_power>=-n) && (u_power<=n)) {
          Array[(n-u_power)/2]=0;
  }
  //Backward Induction
  for(i=n-1;i>=0;i=i-1)
  {
   for(int j=0;j<=i;j++)
   {
   Array[j]=exp(-r*DeltaT)*(Array[j]*Pu+Array[j+1]*Pd
   }
   if ( (((i-u_power)%2)==0) &&
         (u_power>=-i) && (u_power<=i)) {
          Array[(i-u_power)/2]=0;
   }
  }
  //輸出評價結果
  printf("障礙選擇權價格=%f",Array[0]);
return 0;
```

課堂演練

- 輸入不同期數(n)，觀察其價格收斂行爲

- 修改BarrierCRR Project用來評價入局障礙買權:
  - Hint: 入局買權+出局買權=陽春買權
  - 呼叫計算陽春買權的程式:
    - Black-Scholes Formula
    - CRR tree

課堂演練

參見初階班講義Lecture 4
自建函式庫及函式庫的引用

開啓CRR1 project
(內容和CRR porject相同)

加入BarrierCRR project

CRR.h
```
double CRR(float,…);
```

CRR.cpp
```
Int main()
{
  //輸入資料
  float S,T,X,r,Sigma;
  int n;  //期數
  printf("輸入標的物價格:");
  …
}
```

修改

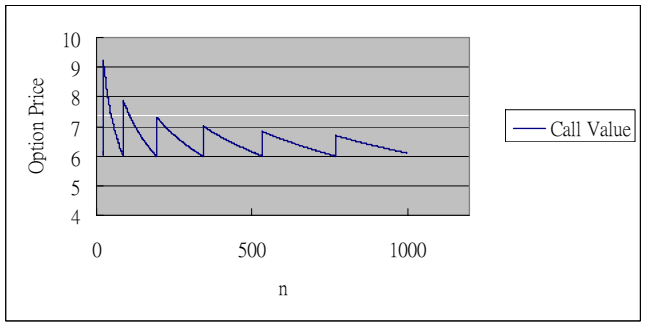
```
double CRR(float S, float T…)
{
 /*  移除輸入的部份,
     輸入部分改用函式呼叫*/
        …
}
```

連結

BarrierCRR.cpp
```
#include "CRR.h"
…
int main()
{
  …
  InOptionValue=
  Array[0]-CRR(S,T…)
}
```

鋸齒狀的收斂行爲

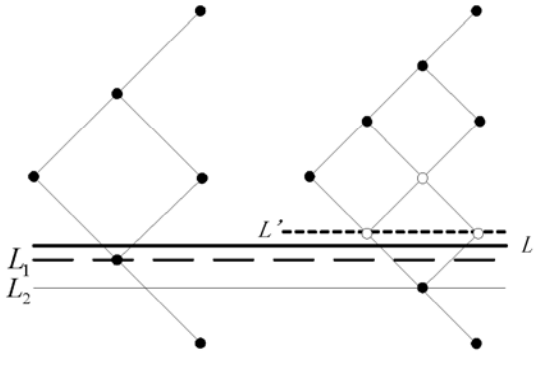
- 以CRR來評價障礙選擇權
(S=95,X=100,T=1,volitility=25%,r=10%,H=90)



- 如何改善？
  - 誤差來自何處？

使用二元樹模型評價造成誤差原因

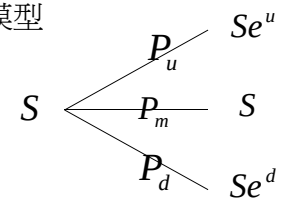
- 有效障礙隨n而改變



如何建構一個樹狀結構,使得有效障礙不隨n而改變?

使用Kamrad & Ritchken 三元樹模型評價障礙選擇權

- 由Kamrad and Ritchken [1991] 提出
- 是一個擁有延伸參數 $\lambda(\geq 1)$ 的三元樹模型
- 模型的設定：
  - $e^u = e^{\lambda\sigma\sqrt{\Delta t}}$
    $e^m = e^0 = 1$
    $e^d = e^{-\lambda\sigma\sqrt{\Delta t}}$



- 標的物價格過程: $S(T) = S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma B(T)}$
  - 由三元樹來模擬標的物的價格→三元樹的期望值和變異數要符合標的物價格過程的期望值和變異數
  - $p_u(\lambda\sigma\sqrt{\Delta t}) + p_d(-\lambda\sigma\sqrt{\Delta t}) = (r - 1/2\sigma^2)\Delta t$
    $p_u(\lambda\sigma\sqrt{\Delta t})^2 + p_d(-\lambda\sigma\sqrt{\Delta t})^2 = \sigma^2\Delta t$

兩個變數 p_u, p_d ,兩個方程式

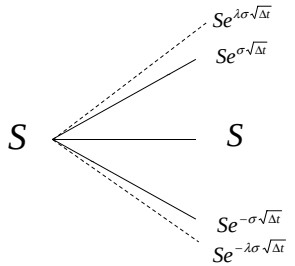
使用Kamrad & Ritchken 三元樹模型評價障礙選擇權

- 再加上 $p_u + p_m + p_d = 1$ 條件
  - 可得到 $(\alpha = r - \frac{1}{2}\sigma^2)$

$$P_u = \frac{1}{2\lambda^2} + \frac{\alpha\sqrt{\Delta t}}{2\lambda\sigma}$$

$$P_m = 1 - \frac{1}{\lambda^2}$$

$$P_d = \frac{1}{2\lambda^2} - \frac{\alpha\sqrt{\Delta t}}{2\lambda\sigma}$$



- Ritchken[1995] 透過Kamrad & Ritchken三元樹模型中參數 λ 的適當選取，使三元樹模擬的標的物價格S(t)能恰恰好落在障礙價格H上

使用Rtichken三元樹評價下出局障礙買權程式流程設計

- 程式流程如下:
  - 1. 輸入資料: 標的物價格(S),標的物波動率(Sigma), 無風險利率(r),履約價格(X),時間長度(T),期數(n).
  - 2. 計算Ritchken三元樹的相關參數
    - Lambda：延伸參數
    - Lindex：從標的物初使價格往下走幾步到障礙
  - 3. 求算選擇權在最後一期的payoff
    - 計算至落在障礙的節點(第n+Lindex個)即可停止
  - 4. Backward Induction
    - 每次均計算至落在障礙的節點
  - 5. 輸出評價結果

請參見Ritchken project

Ritchken project
下出局障礙買權

```
int main()
{
  int i,j;
 //輸入資料
  float S,T,X,r,Sigma;
  int n;  //期數
  printf("輸入標的物價格:");
  scanf("%f",&S);
  printf("輸入到期日:");
  scanf("%f",&T);
  printf("輸入履約價格:");
  scanf("%f",&X);
  printf("輸入無風險利率:");
  scanf("%f",&r);
  printf("輸入標的物價格波動率:");
  scanf("%f",&Sigma);
  printf("輸入期數:");
  scanf("%d",&n);

//計算相關參數
  double dt=T/n;
  double tmp = log(S/L)/(Sigma*sqrt(dt));
  int Lindex = (int)floor(tmp);
  double lambda=tmp/Lindex;

  double R=exp(r*dt);
  double U=exp(lambda*Sigma*sqrt(dt));
  double D=1/U;

double pu=1/(2*lambda*lambda)+
(r-Sigma*Sigma/2.0)*sqrt(dt)/(2*lambda*Sigma);
double pd=1/(2*lambda*lambda)-
(r-Sigma*Sigma/2.0)*sqrt(dt)/(2*lambda*Sigma);
double pm=1.0-pu-pd;

//計算最後一期的payoff
double Array[1000];
double CurrentS=S*pow(U,n);
for(int i=0;i<n+Lindex;i++)
{
   Array[i]=Max(CurrentS-X,0);
   CurrentS=CurrentS*D;
}
Array[n+Lindex]=0;

  //Backward Induction
for(int j=n-1;j>=0;j--)
{   int i=0;
    for(i=0;i<Min(j+Lindex,2*j+1);i++)
    {
     Array[i]=(pu*Array[i]+pm*Array[i+1]+pd*Array[i+2])/R
    }
   if(i==j+Lindex)      Array[i] = 0;
}
 //輸出評價結果
printf("障礙選擇權價格=%f",Array[0]);
}
```

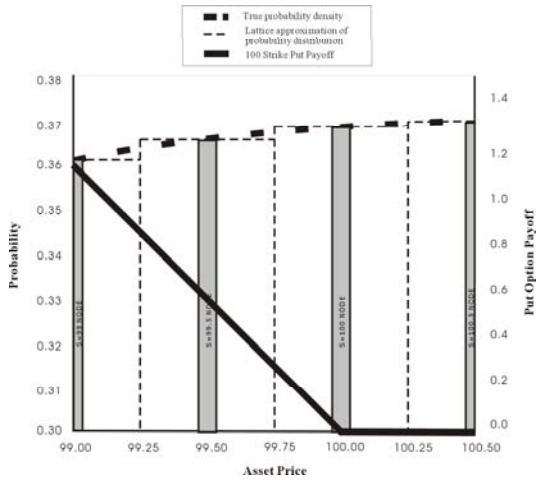
課堂練習

- 輸入不同的n,判斷本程式的收斂
- 修改本程式來評價一上出局障礙買權
- 如果股價的起始價格與障礙價格太接近，會發生什麼現象？(barrier-too-close problem)

模型誤差的來源

- 分佈型誤差(Distribution Error)
- 非線性誤差(Non-linearity Error)

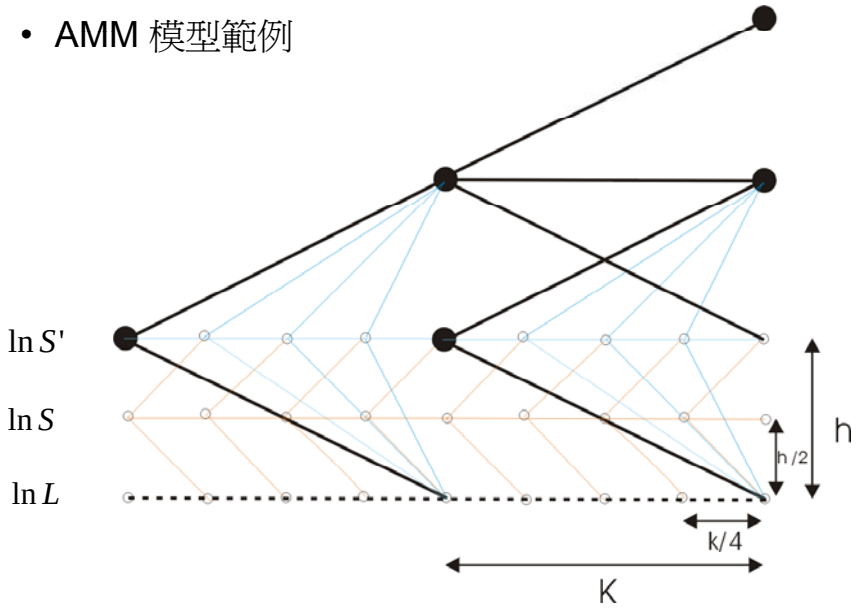
以三元樹模型來評價歐式賣權 (S=100, X=100, r=10%, sigma=25%)，到期日各節點值與所對應機率分佈之示意圖。



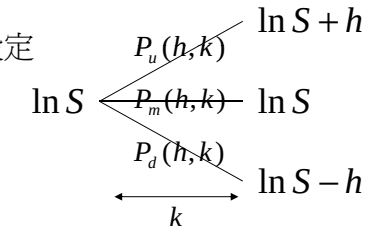
利用Adaptive Mesh Model (AMM模型)評價障礙選擇權

- 由Stephen Figlewski和Bin Gao於1999年提出
- 是一個可局部增加網格密度的樹狀結構，可用較高密度之網格來降低non-linearity error
- 可正確而有效率地評價標的物起始價格和障礙價格過近之障礙選擇權。(可解決barrier-too-close problem)

- AMM 模型範例



- 模型的設定



- 符合標的物價格過程的期望值和變異數
  - 令 $\alpha = r - \frac{1}{2}\sigma^2$

$$P_u(h,k)\times h + P_d(h,k)\times(-h) = \alpha k$$
$$P_u(h,k)\times h^2 + P_d(h,k)\times h^2 = \sigma^2 k + \alpha^2 k^2$$
$$P_u(h,k) + P_m(h,k) + P_d(h,k) = 1$$

- 可解得

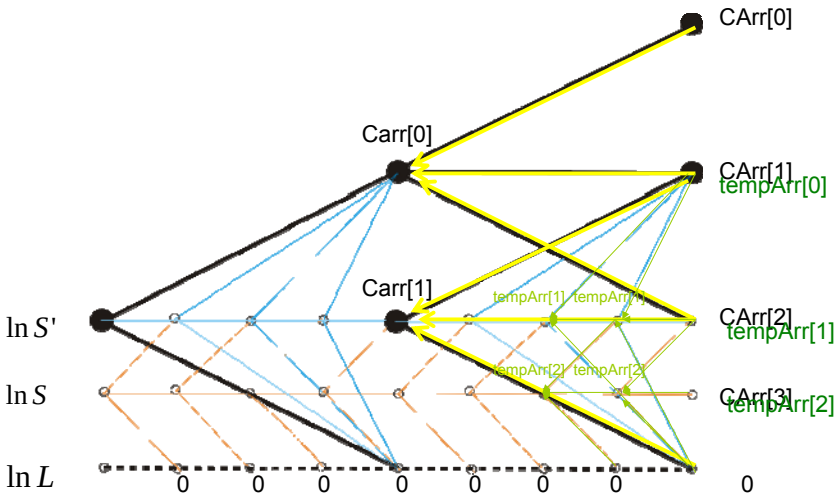
$$P_u = \frac{1}{2}\left[\sigma^2\left(k/h^2\right) + \alpha^2\left(k^2/h^2\right) + \alpha(k/h)\right]$$
$$P_d = \frac{1}{2}\left[\sigma^2\left(k/h^2\right) + \alpha^2\left(k^2/h^2\right) - \alpha(k/h)\right]$$
$$P_m = 1 - P_u - P_d$$

- 一般而言，h和k會遠小於1→ $P_m \approx 1-\frac{\sigma^2 k}{h^2}$
- 爲了確保 $0<P_m<1$
  - 設輸入參數 $\lambda=\frac{h^2}{\sigma^2 k}$ → $k=\frac{h^2}{\lambda\sigma^2}$,規定 $\lambda>1$
- λ 的初始值爲3
- AMM模型建構過程
  1. 建構於標的物於 $S'=\ln L+h$ 起始之最粗三元樹
     - ✓ h = 2 (ln(S)-ln(L))
     - ✓ h已決定，接下來要透過 λ 的調整選擇一個可整除T的k
     - ✓ $n=\text{int}[T/k]=\text{int}[(3\sigma^2/h^2)T]\Rightarrow k=T/n$
  2. 根據粗的三元樹節點計算較細的三元樹節點值且backward induction.

AMM project 資料結構

- CArr：儲存backward induction的過程中，以k爲時間單位之節點值(包含粗三元樹節點、細三元樹節點)
- tempArr：儲存時間單位k內，計算各細節點值過程中所需之節點值。

AMM project 節點計算示意圖



請參照AMM project

```
//定義副函式
double Max(double a,double b){
  if(a>b)  return a;
  else return b;
}

double Round(double a){ 處理四捨五入
   if(a-floor(a)>=0.5)  return ceil(a);
   else      return floor(a);
}

double Pu(double h,double k,
                double Sigma,double r){
   double alpha = r - Sigma*Sigma/2;
   return     (Sigma*Sigma*(k/(h*h))+
          alpha*alpha*(k*k/(h*h))+alpha*(k/h))/2;
}

double Pd(double h,double k,
                double Sigma,double r){
   double alpha = r - Sigma*Sigma/2;
   return (Sigma*Sigma*(k/(h*h))+
          alpha*alpha*(k*k/(h*h))-alpha*(k/h))/2;
}
```

```
int main()
{int i,j;
  //輸入資料
   float S,T,X,L,r,Sigma;
  printf("輸入標的物價格:");
  scanf("%f",&S);
  printf("輸入到期日:");
  scanf("%f",&T);
  printf("輸入履約價格:");
  scanf("%f",&X);
  printf("輸入障礙價格:");
  scanf("%f",&L);
  printf("輸入無風險利率:");
  scanf("%f",&r);
  printf("輸入標的物價格波動率:");
  scanf("%f",&Sigma);
//計算AMM樹的相關參數
  double h = 2*(log(S/L));
  double lambda = 3.;
  int n= (int)Round((lambda*Sigma*Sigma*T)/(h*h
  double k = T/n;
  lambda = (h*h)/(Sigma*Sigma*k);
  S = L*exp(h);

  double R=exp(r*k);
  double pu=Pu(h,k,Sigma,r);
  double pd=Pd(h,k,Sigma,r);
```

```
//配置所需陣列空間
   double CArr[30000];
   double tempArr[3];
   //計算最後一期的payoff
     //計算粗三元樹在起始價之上之節點值(不含lnS')
   double CurrentS=S*exp(n*h);
   for(int i=0;i<n;i++)
   {
    CArr[i]=Max(0,CurrentS-X);
    CurrentS = CurrentS*exp(-h);
   }
     //計算細三元樹的節點值(含lnS')
   CurrentS = S;
   for(int i=n;i<n+2;i++)
   {
     CArr[i] = Max(0,CurrentS-X);
     CurrentS = CurrentS*exp(-h/pow(2,i-n+1));
   }
   double pu2,pd2,pm2;

   //backward induction
   for(int j=n-1;j>=0;j--){
      //計算粗三元樹(包含lnS')的節點值
      for(int i=0;i<j;i++){
          CArr[i]=(pu*CArr[i]+pm*CArr[i+1]+pd*CArr[i+2])/R;
      }
   }
```

```
   //暫存計算細三元樹節點所需資訊
    for(int i=0;i<3;i++)
    {
        tempArr[i] = CArr[j+i];
    }
    //用appraochArr來算粗三元樹在lnS'之下(不含l
   for(int jj=0;jj<4;jj++)
   {
       pu2 = Pu(h/2,k/4,Sigma,r);
       pd2 = Pd(h/2,k/4,Sigma,r);
       pm2 = 1.0 - pu2 - pd2;
       tempArr[2] = (pu2*tempArr[1]+pm2*tempAr
                     pd2*0)/exp(r*k/4);
       pu2 = Pu(h,k*(jj+1)/4,Sigma,r);
       pd2 = Pd(h,k*(jj+1)/4,Sigma,r);
       pm2 = 1.0 - pu2 - pd2;
       tempArr[1] = (pu2*tempArr[0]+pm2*CArr[j+
                     pd2*0)/exp(r*k*(jj+1)/4);
   }
    //複製細三元樹的計算結果至粗三元樹的相對應
   for(int i=1;i<3;i++)
   { //tempArr[0]不需複製
     CArr[j+(i-1)] = tempArr[i];
   }
}
   //輸出評價結果
   printf("下障礙出局選擇權價格=%f",CArr[1]);
```

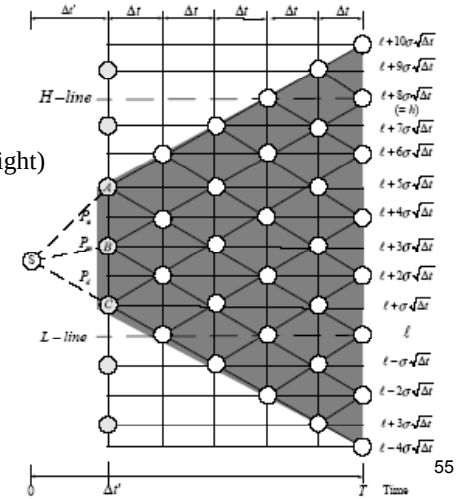
課堂練習

- 請試當修改AMM project的AMM程式，使其所評價之障礙選擇權，在障礙條件成立時(即被knock-out)，能獲得Rebate r。

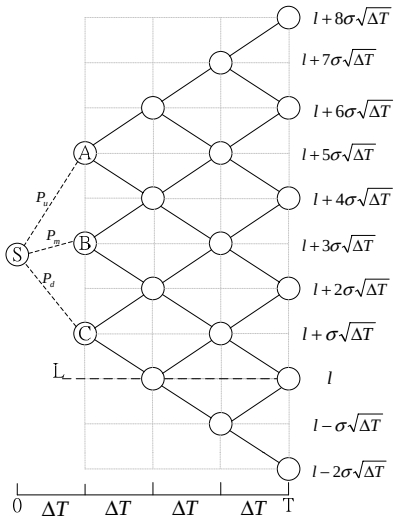
An Overview of bBTT

- Simple: (2 parts)
  - 1-time-step trinomial tree
  - truncated CRR tree
- Flexible:
  - adjustment of cell width (height)
  - position of the grid
  - Hit both barriers

A double barrier options with barriers H and L



To Deal with Single-Barrier Options

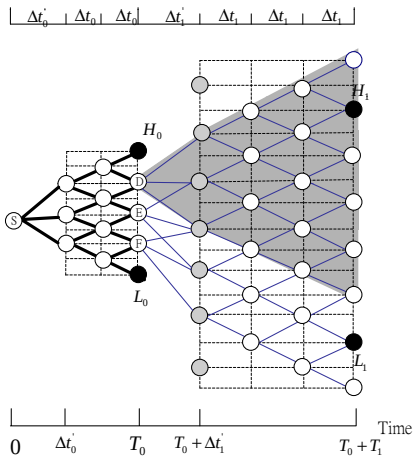


Use a degree of freedom

An Overview of BTT

- To match more critical lcoations:
  - Integrate bBTT
    - In thick edge
    - in shadow
- Four critical locations: H0, H1, L0 and L1

A discrete barrier option



Construct bBTT

The Underlying Grid

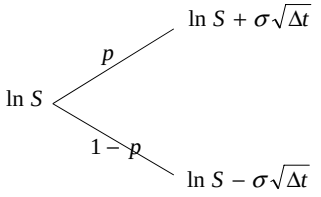
Step 1:

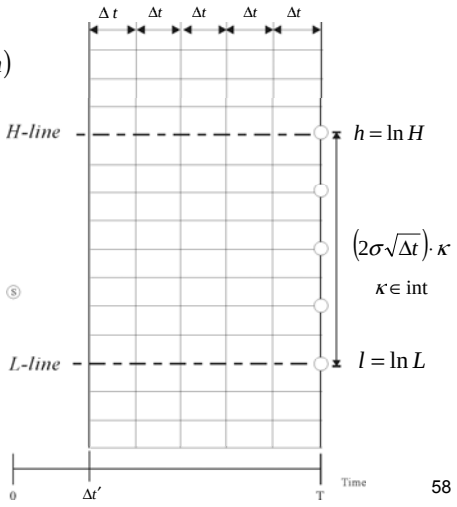
a. If we want to build up an n-step bBTT, find out Δt that is close to, but does not exceed $\Delta\tau(=T/n)$ that makes $\frac{h-l}{2\sigma\sqrt{\Delta t}}$ an integer.

$$\kappa \equiv \left\lceil \frac{h-l}{2\sigma\sqrt{\Delta\tau}} \right\rceil \qquad \Delta t = \left(\frac{h-l}{2\kappa\sigma}\right)^2$$

$$\Delta t \le \Delta t' < 2\Delta t$$

b. properly layout the grid.





Construct bBBT

The Trinomial Structure

Step 2:

a. Choose node A, B, C at time $\Delta t'$ to make P_u , P_m , and P_d valid.

b. The mean and variance of log-stock price from S at $\Delta t'$

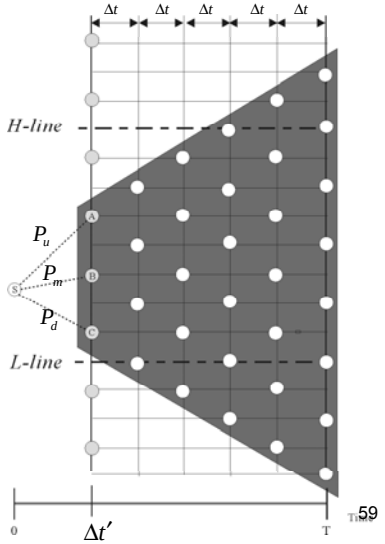
$$\ln\frac{S_{\Delta t'}}{S_0} = \left(r-0.5\sigma^2\right)\Delta t' + \sigma dW_{\Delta t'}$$

Let $\mu \equiv (r-\sigma^2/2)\Delta t'$

c. We choose the only node lies in $\left(\mu-\sigma\sqrt{\Delta t}, \mu+\sigma\sqrt{\Delta t}\right)$ as B

$$\beta = \hat{\mu} - \mu \qquad \alpha = \beta + 2\sigma\sqrt{\Delta t}$$

$$\gamma = \beta - 2\sigma\sqrt{\Delta t}$$



Construct the bBTT

Derive the Branch Probabilities

By matching mean and variance and summing 3 probabilities to be one, we can solve P_u , P_m , and P_d .

$$P_u\alpha + P_m\beta + P_d\gamma = 0$$

$$P_u\alpha^2 + P_m\beta^2 + P_d\gamma^2 = Var$$

$$P_u + P_m + P_d = 1$$

Construct the bBTT

Branch Probabilities are Valid!

$$\det = \begin{bmatrix} \alpha & \beta & \gamma \\ \alpha^2 & \beta^2 & \gamma^2 \\ 1 & 1 & 1 \end{bmatrix} = (\beta-\alpha)(\gamma-\alpha)(\gamma-\beta) < 0,$$

$$\det_u = (\beta\gamma + \mathrm{Var})(\gamma-\beta),$$

$$\det_m = (\alpha\gamma + \mathrm{Var})(\alpha-\gamma),$$

$$\det_d = (\alpha\beta + \mathrm{Var})(\beta-\alpha).$$

Thus, $P_u = \det_u/\det$, $P_m = \det_m/\det$, and $P_d = \det_d/\det$

Verify that $P_u, P_m, P_d > 0$.

Construct the bBTT

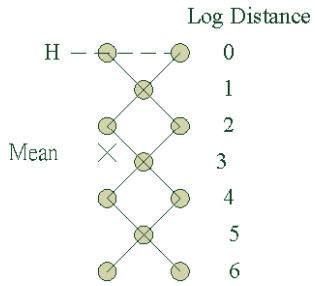
Branch Probabilities are Valid!

Show that $\det_u$, $\det_m$, and $\det_d$ <0 instead.

$$\begin{aligned}
\beta\gamma + \mathrm{Var}(\Delta t') &= \beta^2 - 2\beta\sigma\sqrt{\Delta t} + \sigma^2\Delta t' \geq \beta^2 - 2\beta\sigma\sqrt{\Delta t} + \sigma^2\Delta t = (\beta - \sigma\sqrt{\Delta t})^2 \geq 0, \\
\alpha\gamma + \mathrm{Var}(\Delta t') &= \beta^2 - 4\sigma^2\Delta t + \sigma^2\Delta t' \leq \beta^2 - 4\sigma^2\Delta t + 2\sigma^2\Delta t = \beta^2 - 2\sigma^2\Delta t \leq 0, \\
\alpha\beta + \mathrm{Var}(\Delta t') &= \beta^2 + 2\beta\sigma\sqrt{\Delta t} + \sigma^2\Delta t' \geq \beta^2 + 2\beta\sigma\sqrt{\Delta t} + \sigma^2\Delta t = (\beta + \sigma\sqrt{\Delta t})^2 \geq 0,
\end{aligned}$$

See BTTDB Project

```
double C=ceil((LogH-LogL)/(2*v*sqrt(t)));
 t=pow((LogH-LogL)/(2*v*C),2);  調整後的期數長度
 n=floor(T/t)-1;                計算期數(扣除第一期)
 double TriDeltaT=T-t*n;        計算第一期的時間長度
 double Mean=log(S)+(r-0.5*v*v)*TriDeltaT;   計算第一期的 Log(MeanPrice)
 double LogDistance=(LogH-Mean)/(v*sqrt(t));  計算 H 和 mean 的 log distance
 int Nstep;    將 n整數化成 Nstep
 if(n-int(n)>0.5)
            NStep=int(n)+1;
 else NStep=int(n);
 double Shift;
 double LogSPrice;
 if(NStep%2==0)
 {  //even step
   if(fmod(LogDistance,2.0)>1)   Middle node距離 H int(LogDistance)+Shift 格
                    Shift=1;
            else Shift=0;
   LogSPrice=int(LogDistance)+Shift-2;   -2 後算出upnode 位置
 }
 else
 {   // odd steps
    if(fmod(LogDistance-1,2)>1)
                    Shift=1;
            else Shift=0;
   LogSPrice=int(LogDistance)+Shift-2;
 }
 S=H/exp(v*sqrt(t)*LogSPrice);          計算upper node 的值
```




```
Mean=(r-0.5*v*v)*TriDeltaT;
 double Var=v*v*TriDeltaT;
 //Compute Alpha, beta, gamma and the branching probabilities
 S=H/exp(v*sqrt(t)*LogSPrice);      計算upper node的值
 double Alpha=log(S/OriginalS)-Mean;
 S=S*pow(d,2.0);
 double Beta=log(S/OriginalS)-Mean;
 S=S*pow(d,2.0);
 double Gamma=log(S/OriginalS)-Mean;
 //Compute branching probabilities
 double Det=(Beta-Alpha)*(Gamma-Alpha)*(Gamma-Beta);
 double Pu=(Beta*Gamma+Var)*(Gamma-Beta)/Det;   計算到upper node機率
 double Pm=(Alpha*Gamma+Var)*(Alpha-Gamma)/Det; ;   計算到middle node機率
 double Pd=(Alpha*Beta+Var)*(Beta-Alpha)/Det; ;          計算到lower node機率
 double DBOptionValue=exp(-
r*TriDeltaT)*(Pu*OptionValue[0]+Pm*OptionValue[1]+Pd*OptionValue[2]);
 double DBOutOptionValue=exp(-
r*TriDeltaT)*(Pu*OutOptionValue[0]+Pm*OutOptionValue[1]+Pd*OutOptionValue[2]);
```

Homework

- 上述程式中有一段沒有介紹
  - 該程式使用組合數學求 upper middle 和lower node 的option value
  - 見 C++ 財務程式設計 Chap. 8 中double barrier option 評價
- 請將combinatorial 改成 backward induction (可和原來程式對答案)
- 處理 single barrier option 評價，同時 match barrier 和 stikr price