Financial Engineering and Computations

Monte Carlo Methods for Option Pricing

Dai, Tian-Shyr

課程大綱

- 亂數建立及應用
- 用蒙地卡羅法評價陽春選擇權
- 用蒙地卡羅法評價亞式選擇權
- Quasi-Monte Carlo Simulation
- 蒙地卡羅法的改進方法

---Antithetic variates、Control Variant

- 高維度的蒙地卡羅法

亂數定義

- 亂數是一個隨機產生的數字,廣泛的應用在電腦程式中
  - 例如電腦遊戲: 麻將,撲克牌
- 透過亂數所服從的分配,可了解亂數的特性
  - 例如當標的物的價格服從對數常態分配

$$S(t) = S(0)e^{(r-\frac{1}{2}\sigma^2)t+\sigma\sqrt{t}\varepsilon} \qquad \varepsilon \sim N(0,1)$$

- C語言提供函式產生亂數
  - rand()=> ~Uniform(0,RAND_MAX)
  - #include<stdlib.h>

產生不同的亂數序列

- 設定亂數的起始點
  - srand();
  - 起始點不同,就會產生不同的亂數序列
  - 爲了確保每次都有不同的亂數
    - 使用系統時間
    - #include<stdlib.h>

範例程式

參照RandomNumber Project

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

void main( void )
{
  int i;
  //使用系統時間產生亂數序列
  srand( (unsigned)time( NULL ) );
  for( i = 0;   i < 10;i++ )
  {
    printf( "  %d\n", rand() );
  }
}
```

如將本行移除,則每次產生的亂數序列都會相同

列印亂數

常態分配隨機變數建立

- 做財務模擬程式經常需要常態隨機變數

$$S(T) = S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\varepsilon}$$

- 常態分配變數可用下列方式逼近
  - 使用RAND()產生 0~1 的 uniformly distributed 的隨機變數
  - 假定產生的變數爲Wi $E(W_i) = 0.5,\ \mathrm{Var}(W_i) = \dfrac{1}{12}$
  - 根據中央極限定理: $\sum_{i=1}^{12} W_i - 6 \sim N(0,1)$

程式範例:

```
double Normal=0;
        for(int j=0;j<12;j++)
        {
          Normal=Normal+double(rand())/RAND_MAX;
        }
    Normal=Normal-6;
```

課堂演練

- 更改上述程式,輸入M和V,製造平均爲M且變異數爲V的常態隨機變數

Homework

- 課本第九章習題4、5

蒙地卡羅法

- 標的物價格表示: $S(T)=S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\varepsilon}$
- 已知 $\varepsilon \sim N(0,1)$,該隨機變數可用電腦模擬,計算出標的物在到期日的價格
- 以一買權爲例,用蒙地卡羅法模擬到期日payoff可表示如下:

$$\left(S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\varepsilon_i}-X\right)^+ \equiv V_i$$

- $\text{Value}=e^{-rT}\mathrm{E_Q}(\text{Payoff})=e^{-rT}\mathrm{E_Q}(\mathrm{V_i})$

蒙地卡羅法

- 定義 Var(Vi)=U
- 計算 $P_n=\dfrac{\sum_{i=1}^{n}\mathrm{V_i}}{n}$

$$\mathrm{E_Q}(\text{Payoff})=\mathrm{E_Q}(\mathrm{P_n})$$

$$\mathrm{Var}(\mathrm{P_n})=\frac{nU}{n^2}=U/n$$

- 當計算次數夠多時,蒙地卡羅法收斂至正確值

蒙地卡羅法
陽春買權的評價

- 買權的到期日payoff爲

$$(S(T)-X)^+=(S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\varepsilon}-X)^+$$

- 買權的價格估計如下:

$$e^{-rT}\frac{1}{n}\sum_{i=1}^{n}(S(T,i)-X)^+ \rightarrow e^{-rT}E(S(T)-X)^+$$

- 程式設計如下:
  - 輸入相關參數
  - 建立一個副函式計算常態隨機變數
  - 隨機產生標的物到期日的價格,並計算選擇權的報酬
  - 計算報酬的平均值

參照MonteCarlo Project

副程式

```
double Max(double a,double b)
{
 if(a>b) return a;
 else return b;
}

double Normal()
{
    double N=0;
    for(int j=0;j<12;j++)
     {
      N=N+double(rand())/RAND_MAX;
     }
    N=N-6;
    return N;
}
```

主程式

```
//輸入資料
  float S,T,X,r,Sigma;
  int n;
  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("輸入模擬的次數:");
  scanf("%d",&n);
  //蒙地卡羅模擬
  srand( (unsigned)time( NULL ) );
  double OptionValue=0;
  double SquareSum=0;
  for(int i=0;i<n;i++)
  {
   double N=Normal();
   double ST=S*exp((r-Sigma*Sigma/2)*T+Sigma*sqrt(T)*N);
   double Payoff=exp(-r*T)*Max(ST-X,0);
   OptionValue=OptionValue+Payoff;
   SquareSum=SquareSum+pow(Payoff,2);
  }
  OptionValue=OptionValue/n;
  double STDERR=sqrt((SquareSum/n-pow(OptionValue,2))/(n-1));
  printf("Value=%f, Standard Err~%f",OptionValue,STDERR);
```

模擬n次,計算每一次payoff

Standard Error的計算

課堂練習

- 修改上述程式,計算賣權的價值
  - 賣權的payoff: $(X - S(T))^+ = (X - S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\varepsilon})^+$

使用Put-Call Parity評價賣權

- 考慮買權和賣權到期日的payoff:
  - 買權: $(S(T)-X)^+$ 賣權: $(X-S(T))^+$
- 如果買一個買權,賣一個賣權,到期日payoff
  - $(S(T)-X)^+ - (X-S(T))^+ = S(T)-X$
  - 其 payoff相當於期初借 Xe^{-rT} ,並購買標的物
- 在無套利的假定之下,假定C,P爲買權和賣權的價格,可得 $C - P = S(0) - Xe^{-rT}$

課堂練習

- 用蒙地卡羅法所得的買權和賣權的價格,驗證put-call parity

$$C - P = e^{-rT}\sum_{1\le i\le n} S(T,i)/n - Xe^{-rT}$$

- 比較使用Monte Carlo Methods和put-call parity計算出的賣權的價值

  ---程式驗證:如何擔保程式碼以及概念推導過程正確(可參考課本C++財務程式設計 pp.9-12.)

亞式選擇權

- 亞式選擇權的payoff和標的物的平均價格有關
  - 假定一個標的物的價格在一段時間內的價格爲S(0),S(1),S(2),S(3), … S(m)
  - 標的物的平均價格 $A = \frac{S(0)+S(1)+S(2)+\ldots+S(m)}{m+1}$
- 以一個亞式買權爲例,到期日的payoff可寫成 $(A-X)^+$
- 亞式選擇權的用途
  - 防止標的物的價格被操縱
  - 規避和平均價格有關的風險

用蒙地卡羅法評價亞式選擇權

- 標的物的價格隨機過程滿足:

$$S(t) = S(t-1)e^{(r-\frac{1}{2}\sigma^2)T/m+\sigma\sqrt{T/m}\varepsilon_t}$$

- 時間t的價格,可用時間t-1的價格和常態隨機變數 ε_t 構成
- 產生 $\varepsilon_1, \varepsilon_2, \varepsilon_3 ..., \varepsilon_m$ 共m個常態隨機變數,利用這些變數產生價格路徑S(0),S(1)… S(m)
- 求出該價格路徑平均值=>求出選擇權報酬
- 模擬n條路徑,並求報酬的平均值

核心程式碼

參照AsianOption Project

```
//蒙地卡羅模擬
srand( (unsigned)time( NULL ) );
double OptionValue=0;
double SquareSum=0;
double DT=T/m;
for(int i=0;i<n;i++)
{
        double ST=S;
        double Sum=ST;
        for(int j=1;j<=m;j++)
        {
                double N=Normal();
                ST=ST*exp((r-Sigma*Sigma/2)*DT+Sigma*sqrt(DT)*N);
    Sum=Sum+ST;
        }
        double Avg=Sum/(m+1);
        double Payoff=exp(-r*T)*Max(Avg-X,0);
        OptionValue=OptionValue+Payoff;
        SquareSum=SquareSum+pow(Payoff,2);
}
OptionValue=OptionValue/n;
double STDERR=sqrt((SquareSum/n-pow(OptionValue,2))/n);
printf("Value=%f, Standard Err~%f",OptionValue,STDERR);
```

模擬n條價格路徑

模擬該路徑在每個時間點價格

課堂演練

- 用O-notation來估計上述程式的計算複雜度
- 輸入不同的n,驗證standard error隨著n增大而減小,計算時間隨n增大而增加,
  - 驗證當n增加10倍時,standard error大概變成原來1/3,計算時間變10倍
- 要求精確的價格所付出的計算代價頗高

Quasi-Monte Carlo Simulation

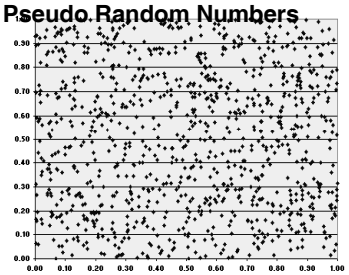
- 蒙地卡羅法有幾個重大缺陷
  - 計算結果是一個隨機變數,
    - 每次計算結果都不相同
    - 當取的隨機變數品質不佳,則計算結果會不正確
  - 收斂速度太慢
    - $O(\frac{1}{\sqrt{n}})$
- Quasi-Monte Carlo simulation採用一些方式改善蒙地卡羅法的缺陷
  - 取的隨機變數品質不佳
  - 收斂速度太慢

Quasi-Monte Carlo Simulation

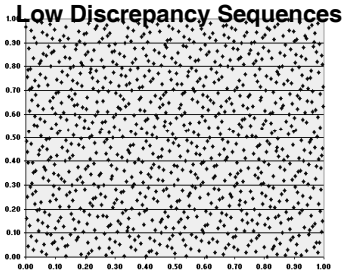
GBM

$$S_i = S_{i-1}\exp\left[\left(r-\frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\,\varepsilon_i\right]$$

Pseudo Random Numbers



Low Discrepancy Sequences



- **Rand()**

- **Halton Sequences**
- **Faure Sequences**
- **Sobol Sequences**

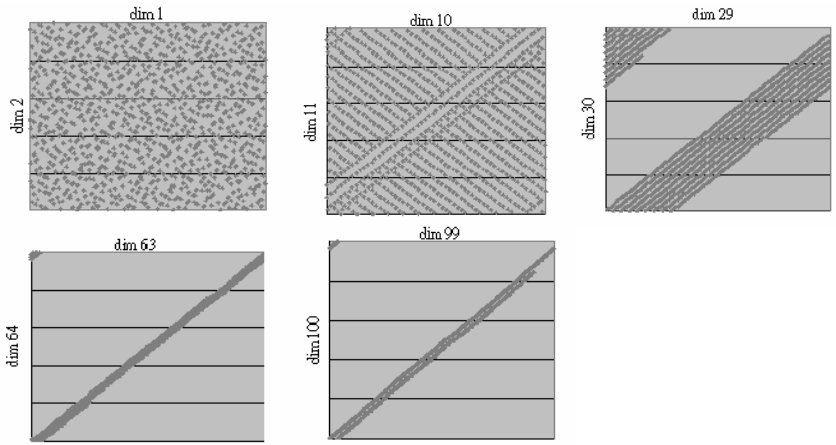
收斂速度 $O(\frac{1}{\sqrt{n}})$ $O(\frac{(\log N)^s}{N})$

Low Discrepancy Sequences

Halton sequence

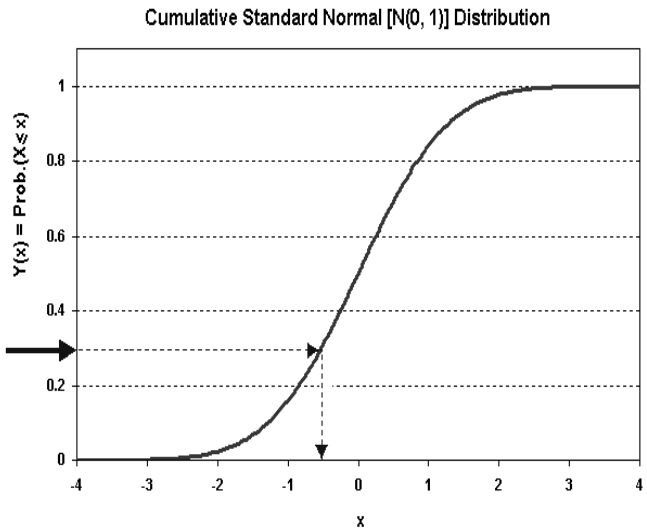
- 每一條 Halton 數列 由一個質數產生
- m1 base=2 $\left(\frac{1}{2},\frac{1}{4},\frac{3}{4},\frac{1}{8},\frac{5}{8},\frac{3}{8},\frac{7}{8},\frac{1}{16},\cdots\right)$
- m2 base=3 $\left(\frac{1}{3},\frac{2}{3},\frac{1}{9},\frac{4}{9},\frac{7}{9},\frac{2}{9},\frac{5}{9},\frac{8}{9},\cdots\right)$
- m3 base=5 $\left(\frac{1}{5},\frac{2}{5},\frac{3}{5},\frac{4}{5},\frac{1}{25},\frac{6}{25},\frac{11}{25},\frac{16}{25},\cdots\right)$
- Halton數列的缺點：
  - 數列有循環
  - 數列在每次的循環中單調遞增
  - 若維度高兩數列間呈現高相關性

Halton sequence
Two-Dimensional Projections



若維度高兩數列間呈現高相關性

使用轉換函數將均勻分配轉換常態分配



產生Halton Sequence和常態隨機變數

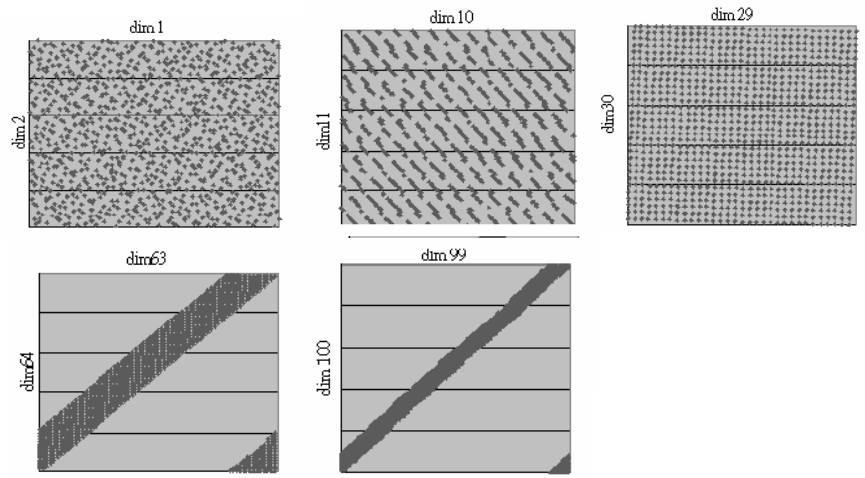
```
double ma[MAX_DIM_NUM][1000];     //存放亂數的容器
int     dim_times=1000;  //每個dimension需要1000個亂數
int     dim_num=20;      //需要20 dimensions
int     step=10;         //每個dimension從第10個值開始
double  r[MAX_DIM_NUM];      //計算亂數的容器
FILE* Write=fopen("T.txt","w");
halton_ndim_set ( dim_num );  //設定 dimension numbers
halton_step_set ( step ); //設定第幾個值開始
for (int i = 0; i <dim_times; i++ )
{
            halton( r );                //產生1~20 dimension 的第10個值
   for(int k=0;k<dim_num;k++)   //將產生的亂數向量存入 自己的容器
   {
                        r[k]=ltqnorm(r[k]);
                        ma[k][i]=r[k];
                        fprintf(Write,"%lf\t",r[k]);
            }
     fprintf(Write,"\n");
}
```

請參見 Halton project

Low Discrepancy Sequences
-Faure sequence-

- 每一條 Faure數列 由一同一個質數P產生,而P要大於 維度 M
- m1 base=3 $\left(\frac{1}{3},\frac{2}{3},\frac{1}{9},\frac{4}{9},\frac{7}{9},\frac{2}{9},\frac{5}{9},\frac{8}{9},\cdots\right)$
- m2 base=3 $\left(\frac{1}{3},\frac{2}{3},\frac{4}{9},\frac{7}{9},\frac{1}{9},\frac{8}{9},\frac{2}{9},\frac{5}{9},\cdots\right)$
- m3 base=3 $\left(\frac{1}{3},\frac{2}{3},\frac{7}{9},\frac{1}{9},\frac{4}{9},\frac{5}{9},\frac{8}{9},\frac{2}{9},\cdots\right)$
- Halton數列的缺點：
  - 數列有循環
  - 若維度高兩數列間呈現高相關性

Faure sequence
Two-Dimensional Projections



若維度高兩數列間呈現高相關性

程式碼請參見
Faure project

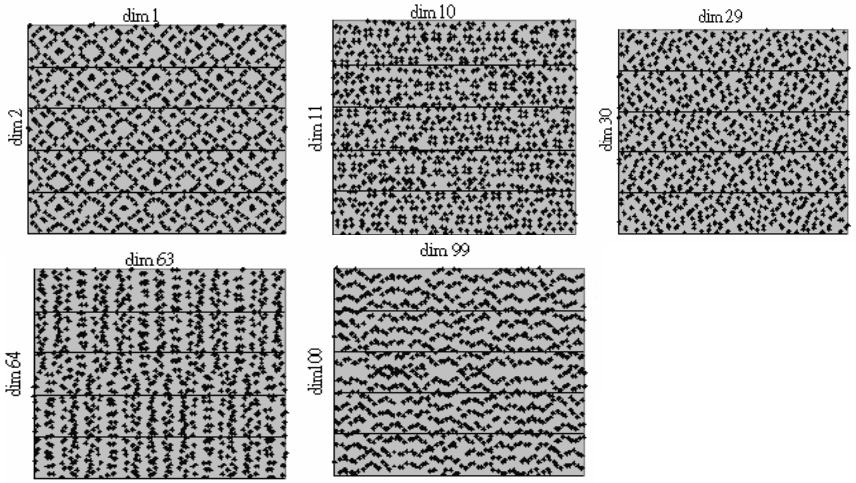
```
double ma[20][1000];     //存放亂數的容器
 int     dim_times=1000;  //每個dimension需要1000個亂數
 int     dim_num=20;      //需要20 dimensions
 int     seed=-1;            //每個dimension從第seed值開始
                        //if  Seed<0.....start (qs)^(4)-1 -->qs   means base
                        //if  Seed>0.....start seed
 double  r[MAX_DIM_NUM];      //計算亂數的容器
 FILE* Write=fopen("T.txt","w");
  for (int i = 0; i <dim_times; i++ )
  {
              faure ( dim_num, &seed, r ); //產生1~20 dimension 的第seed個值
      for(int k=0;k<dim_num;k++)   //將產生的亂數向量存入 自己的容器
              {
                        r[k]=ltqnorm(r[k]);
                        ma[k][i]=r[k];
                        fprintf(Write,"%lf\t",r[k]);
              }
     fprintf(Write,"\n");
  }
```

Sobol sequence

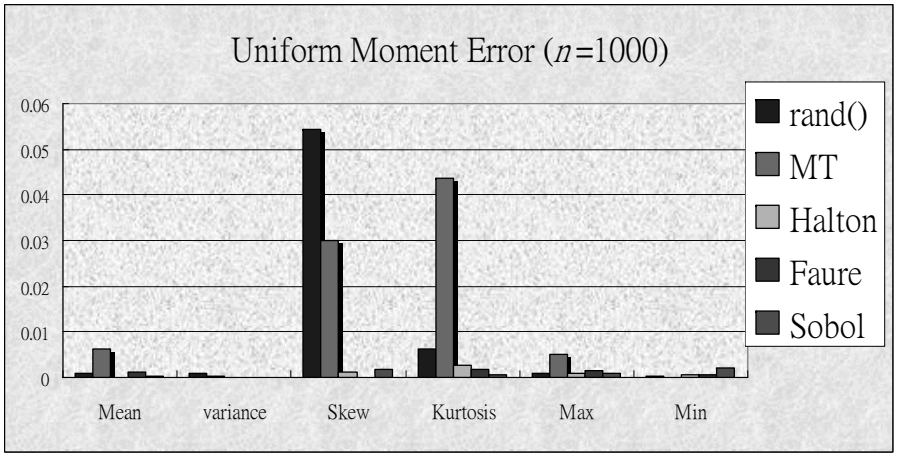
- 每一條 Sobol數列 由同一個質數2產生
- Dim1 base=2 $\left(\frac{1}{2},\frac{3}{4},\frac{1}{4},\frac{3}{8},\frac{7}{8},\frac{5}{8},\frac{1}{8},\frac{3}{16},\cdots\right)$
- Dim2 base=2 $\left(\frac{1}{2},\frac{1}{4},\frac{3}{4},\frac{3}{8},\frac{7}{8},\frac{1}{8},\frac{5}{8},\frac{5}{16},\cdots\right)$
- Dim3 base=2 $\left(\frac{1}{2},\frac{3}{4},\frac{1}{4},\frac{5}{8},\frac{1}{8},\frac{3}{8},\frac{7}{8},\frac{5}{16},\cdots\right)$

沒有上述兩數列的缺點

Sobol sequence Two-Dimensional Projections



比較均勻特性

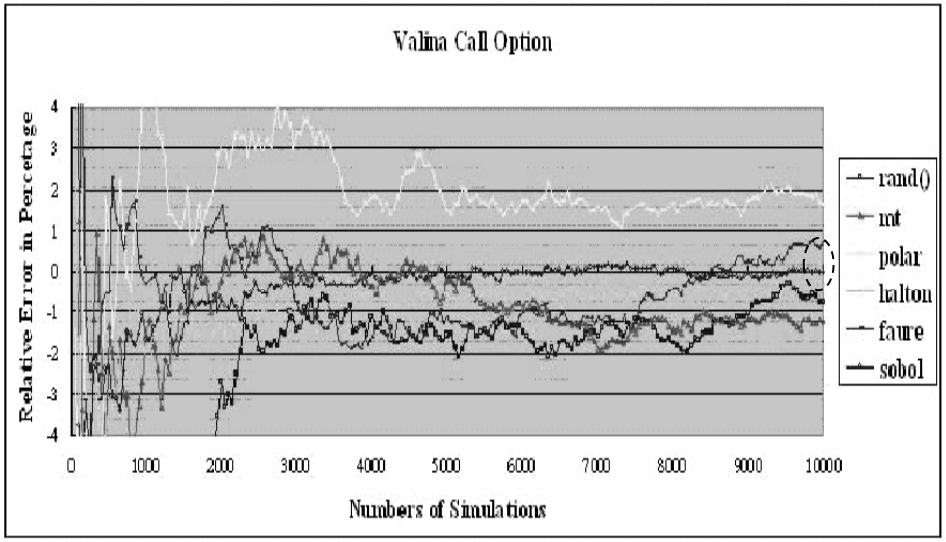


比較產生亂數所需要的時間

Generation time n=100000 $dimension$=20					
Sequence	rand()	MT	Halton	Faure	Sobol
seconds	2.26500	1.53900	2.46900	2.71500	1.21900

•Sobol數列速度顯著的優於其他數列

•以計算買權價格 比較收斂速度



•QMC 確實比傳統 MC 收斂更快 ，其中又以Sobol數列最準確

Summary

- The Sobol sequence can be generated significantly faster than all the other sequences.
- For high-dimensional integrals, Sobol sequences exhibit better convergence properties than either the Faure or the Halton sequences.
- If the dimension is under 100, QMC using Sobol exhibits better convergence than standard MC.

課堂演練

QmcAsianOption Project程式已使用sobol數列，評價亞式選擇權

- 參數:
  - (標的物價格)S=100,
  - (履約價格)X=100,
  - (無風險利率)r=0.1,
  - (波動率)sigma=0.3,
  - (到期日)T=1,
  - (Sample次數)m=52,
  - (模擬價格路徑數)n=1000

- 使用傳統蒙地卡羅法的AsianOption Project做比較價格與標準差

常見的蒙地卡羅法的改進方法

- Antithetic variates
  - 當產生隨機變數 ε ,同時產生對應的變數 $-\varepsilon$
  - 以陽春買權評價爲例
  - 產生 $(S(T)-X)^+ = (S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\varepsilon} - X)^+$
  - 同時算 $(S(T)-X)^+ = (S(0)e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}(-\varepsilon)} - X)^+$
  - 總共會模擬2n條路徑
  - 取的隨機變數的平均值爲0 (提高隨機變數品質)

$$\varepsilon + (-\varepsilon) = 0$$

課堂演練

- 改寫陽春買權的評價程式,採用Antithetic variates 的方法評價

使用Control Variant評價亞式選擇權

- Control Variant可用來提高蒙地卡羅法的收斂速度
- 考慮幾何平均選擇權
  - 標的物的幾何平均價格 $G = \sqrt[m+1]{S(0)S(1)S(2)...S(m)}$
  - 幾何平均買權的payoff= $(G-X)^+$

$$P_A = (A-X)^+;\quad P_G = (G-X)^+$$

$$V_A = e^{-rT}\mathrm{E}(P_A);\ V_G = e^{-rT}\mathrm{E}(P_G)$$

$$P'_A \equiv P_A + (e^{rT}V_G - P_G) \rightarrow V_A = e^{-rT}\mathrm{E}(P'_A)$$

$$\mathrm{Var}(V_A) = e^{-2rT}\frac{Var(P_A)+Var(P_G)-2Cov(P_A,P_G)}{n}$$

使用Control Variant評價亞式選擇權

- 幾何平均選擇權的價格(V_G)已有公式解
- Black-Scholes 處理dividend-yield(q) 的公式: $SN\left(\frac{\ln\left(S/X\right)+\left(r-q+\sigma^2/2\right)T}{\sigma\sqrt{T}}\right)-e^{-rT}XN\left(\frac{\ln\left(S/X\right)+\left(r-q-\sigma^2/2\right)T}{\sigma\sqrt{T}}\right)$
  - Vol: $\frac{\sigma}{\sqrt{3}}$ dividend yield (q): $\frac{1}{2}(r+\frac{\sigma^2}{6})$
- P_A和P_G有高度正相關,
  - 用P'_A來估計V_A時,其變異數較小
  - 可以用較少的模擬次數求得精確的價值

課堂練習

- 給定幾何平均買權的評價程式
  - 見函式 GeometricCall()
  - 請修改蒙地卡羅評價亞式選擇權的程式,改用control variants 的方式來計算亞式選擇權的價格
  - 驗證在相同的模擬次數(n)下,變異數縮小

高維度的蒙地卡羅法

- 彩虹選擇權的報酬受多種標的物影響
- 假定有k種標的物的價格

$$dS_1 = rdt + \sigma_1 dW_1$$
$$dS_2 = rdt + \sigma_2 dW_2$$
$$\ldots$$
$$dS_k = rdt + \sigma_k dW_k$$

- 其中$W_1, W_s, W_3 \ldots W_k$分別代表標準布朗運動 W_i 和的相關W_j係數爲 ρ_{ij}。
- k個標的物價格不獨立，所以無法各自去模擬標的物價格的變化

多維常態分配

- 假定常態隨機變數向量$(a_1, a_2, a_3 \ldots a_k)$
- a_i 和a_j 的相關係數可用ρ_{ij} 表示
- 取k個獨立標準常態隨機變數構成的向量w
- 帶入$a = Bw^T$
  - 其中 B爲一k*k的二維下三角矩陣
  - 令Σ 爲a的共變異數矩陣 $\Sigma = BB^T$
  - B可用Cholesky分解定理求得

見 C++財務程式設計 pp.9-26.

矩陣B元素的決定

- B(i,j)=0 i<j，
- B(i,1)= ρ_{i1}，
- $B(i,j) = \dfrac{\rho_{ij} - \sum_{l=1}^{j-1} B(i,l) \times B(j,l)}{B(j,j)}$， if j<i，
- $B(j,j) = \sqrt{1 - \sum_{l=1}^{j-1} B(j,l)^2}$ 。
- 利用a 帶入 $S_i(T) = S_i(0) e^{(r - 0.5\sigma_i^2)T + \sigma_i \sqrt{T} a_i}$,就可以模擬出k種標的物價格變動的連動性

Variance 計算

$$Var(a_i) = Var\left(\sum_{j=1}^{i-1} B(i,j) w_j + \sqrt{1 - \sum_{j=1}^{i-1} (B(i,j))^2} w_i \right)$$

$$= \sum_{j=1}^{i-1} B(i,j)^2 Var(w_j) + \left(1 - \sum_{j=1}^{i-1} B(i,j)^2 \right) Var(w_i)$$

$$= \sum_{j=1}^{i-1} B(i,j)^2 + 1 - \sum_{j=1}^{i-1} B(i,j)^2 = 1$$

Covariance 計算

$$\frac{Cov(a_i,a_j)}{\sqrt{Var(a_i)Var(a_j)}} = E\big((a_i-0)(a_j-0)\big) = E\left(\sum_{l=1}^{i} B(i,l)w_l \times \sum_{h=1}^{j} B(j,h)w_h\right)$$

$$= \sum_{l=1}^{i} B(i,l)\times B(j,l)E(w_l^{\,2}) = \sum_{l=1}^{i-1} B(i,l)\times B(j,l) + B(i,i)\times B(j,i)$$

Let i<j

$$= \sum_{l=1}^{i-1} B(i,l)\times B(j,l) + B(i,i)\times \frac{\rho_{ij} - \sum_{l=1}^{i-1} B(i,l)\times B(j,l)}{B(i,i)}$$

$$= \rho_{ij}$$

$$B(j,i)$$

程式撰寫

請參閱Rainbow project

- 程式包含三個部分:
  - 文字檔(檔名TestData.txt)，包含標的物初始價格，波動率和跟其他標的物價格的相關係數，
  - 利用Cholesky分解定理計算矩陣B
  - 蒙地卡羅程式的主體。

文字檔

```
100 110 120 130
0.3 0.4 0.5 0.6
0.5
0.4 0.3
0.5 0.4 0.6
```

- 第一行的四個數字(100, 110, 120 130)代表四個標的物的初始價格，第二行的四個數字(0.3, 0.4, 0.5, 0.6)代表四個標的物的價格波動率，第三行的0.5代表第一個和第二個標的物價格報酬率的相關係數 ρ_{12} ，第四行兩個數字分別代表 ρ_{13}和 ρ_{23} ，最後一行三個數字別代表 ρ_{14}、ρ_{24} 和 ρ_{34} 。

Cholesky分解定理計算矩陣B

- 放在ComputeB()這個副函式中
- 求出矩陣B後，就可利用$a = Bw^T$ 產生向量a，這一部份的程式碼放在副函式GenNormalVector()中
  - 計算結果則擺在矩陣a[]中

蒙地卡羅程式的主體

- 可分成三個部分
  - 輸入相關的參數，如無風險利率，
  - 呼叫ComputeB()計算矩陣B，
  - 利用GenNormalVector()模擬標的物的價格和選擇權的報酬
- 程式中採用Maximum call option定義
- $Max(Max(S_1(T), S_2(T), ..., S_k(T)) - X, 0)$

程式主體

```
ComputeB(k);
 //蒙地卡羅模擬
  srand( (unsigned)time( NULL ) );
  double SumU=0;
  double SquareSum=0;
  for( int i=0;i<n;i++)
  {
   GenNormalVector(k);
            double MaxST=0,ST;
            for(int j=0;j<k;j++)
            {
              ST=InitPrice[j]*
   exp((r-Vol[j]*Vol[j]/2)*T+Vol[j]*sqrt(T)*a[j]);
              MaxST=Max(MaxST,ST);
            }
            double V=exp(-r*T)*Max(MaxST-X,0);
   SumU=SumU+V;
  }
  double U=SumU/n;
  printf("Value=%f\n",U);
```

計算矩陣B

計算n次

每次模擬k個標的物

計算Maximum option payoff

課堂練習

- 請修改上述程式來評價Minimum put

$$Max(X - Min(S_1(T), S_2(T), ..., S_k(T)), 0)$$