周俊宇、鍾熾昌、高春元、黃偉豪
2012/5/2

蒙地卡羅模擬法

一、 *理論*

蒙地卡羅模擬是根據衍生性金融商品標的資產價格其變動透過隨機過程進行模擬，並進而求算出此一衍生性商品的價值。假設選擇權的價值（C）不僅受到到期日資產價格（S_T）的影響，亦受到從發行日至到期日之間資產價格的路徑（S_0 ,...., S_T）影響，即C =f（S_0 ,...., S_T），其中f(・)為新奇選擇權的收益函數。若短期無風險利率為常數，即r(τ) =r，則此一選擇權的現值為$C_0 = e^{-rT} E_r\left(f\left(S_0,...,S_T\right)\right)$，其中$E_r(\bullet)$為風險中立下之期望值。蒙地卡羅模擬法即利用此方程式來進行評價。更明確的說，若資產價格（S）服從幾何GBM(幾何布朗運動)，則資產價格的瞬間變動為:

$$\frac{dS}{S} = \mu dt + \sigma dW$$

設ln S為在一段時間內（如從時點t 至T）的變動，服從常態分配：

$$\ln S_T - \ln S_t \sim N\left[\left(\mu - \frac{\sigma^2}{2}\right)(T-t), \sigma\sqrt{T-t}\right]$$

在風險中立評價法下（Risk-Neutral Valuation），我們以無風險利率r 代替μ，則

$$\ln S_T \sim N\left[\ln S_t + \left(\mu - \frac{\sigma^2}{2}\right)(T-t), \sigma\sqrt{T-t}\right]$$

故在時點T 之資產價格（S_T）可表示為：

$$S_T = S_t \exp\left[\left(r - \frac{\sigma^2}{2}\right)(T-t) + \varepsilon\sigma\sqrt{T-t}\right]$$

其中$\varepsilon \sim N(0,1)$所抽取之隨機樣本。在資產價格模擬時，將選擇權的存續期間分割成N 期，每期皆為Δt（＝T/N），然後自標準常態分配中抽出N 個獨立之隨機樣本後，依據上式即可計算出在時點0，Δt，2Δt，....，T 時之資產價格，如此便產生一條資產價格模擬之路徑，並可以利用V =f(S_0 ,...., S_T)求算新奇選擇權之收益值。每一條資產價格模擬路徑與相應的選擇權收益值之計算，稱為一次的模擬試行（Simulation Trial）。在經過大量的模擬試行之後，譬如100,000 次，可以得到100,000條資產價格模擬路徑及其對應的選擇權收益值，即將模擬所得之100,000 個選擇權收益值予以平均，再以無風險利率折現，便可得出該選擇權價值的估計值。

➢ *信賴區間*

在試行N次的模擬後，可求得其所有估計值的平均數$\bar{\mu}_c$，以及標準差w，以此可建立一信賴區間，投資人可依據自己的容忍誤差程度不大於此信賴區間，以決定蒙地卡羅模擬的試行次數，容忍誤差程度越小，則求得的選擇權價格越精確，但所花費時間也越長。故其估計值的信賴區間為

$$\bar{\mu}_c - Z_{\frac{\alpha}{2}}\frac{w}{\sqrt{n}} \leq C \leq \bar{\mu}_c + Z_{\frac{\alpha}{2}}\frac{w}{\sqrt{n}}$$

1. *Antithetic variate method*

一般來說，Antithetic variate method係用於減低參數估計值的variance。而常見的用法與假設：假設f為對應隨機變數的函數，而 $f(x1)$ 、 $f(x2)$ 為不同函數值，其中 $x1=-x2$ 。取兩函數平均且令為 $\bar{f}=f(x1)+f(x2)/2$ 。則其 $Var(\bar{f})$ 會比原本 $Var(f(X))$ 來的小。故現將BS model對應的股價，套用Antithe tic variate method，假設產生一隨機變數值-x1，其中x1服從標準常態分配。即生成兩個不同模擬股價，分別為：

$$S^{+}_{dt}=S_0\exp\left[\left(r-\frac{\sigma^2}{2}\right)dt+\sigma x1\sqrt{dt}\right]$$

$$S^{-}_{dt}=S_0\exp\left[\left(r-\frac{\sigma^2}{2}\right)dt+\sigma(-x1)\sqrt{dt}\right]$$

由此兩個不同的模擬股價可估計得 $\tilde{C}_1$ 、 $\tilde{C}_2$ 。而選擇權的估計平均值為 $\frac{\tilde{C}_1+\tilde{C}_2}{2}$ ，其對應的variance會比原本只用一個BS model生成的模擬股價對應的估計選擇權值來的小。

2. *Control variate method*

Control variate method係為估計 θ ，假設X為的 θ 不偏估計量($\mathrm{E}(X)=\theta$)，且有另一變數Y，其期望值為 $E(y)=\mu_y$ 。

現設Z=X+c $(Y-\mu_y)$ ，E(Z)=E $\left(\mathrm{X}+\mathrm{c}(Y-\mu_y)\right)$ =E $(X)=\theta$ (Z同樣為 θ 的不偏估計量)，

$$Var(Z)=Var\left(X+c(Y-\mu_y)\right)=Var(X)+c^2Var(Y)+2cCov(X,Y)$$

(當c= $-\frac{Cov(X,Y)}{Var(Y)}$ 時，可使 $Var(Z)$ 變小)

故採用Control variate method，套用在選擇權價格模型中，通常會先假設兩衍生性金融商品A、W，其中A與W具有較高相關性。且透過Monte Carlo simulation用A與W生成兩模擬金融性商品價格分別為 A^* 、 W^* 。

令 $A=k(A^*-W^*)+W$

則 $Var(A)=Var\left(k(A^*-W^*)+W\right)=Var(W)+k^2Var(A^*-W^*)+2kCov(W,A^*-W^*)$

就上述過程， $k=-\frac{Cov(W,A^*-W^*)}{Var(W)}$ 時，變異數會變小。

3. *最小平方蒙地卡羅模擬*

最小平方蒙地卡羅模擬是由Longstaff和Schwartz在2000發表，其所使用蒙地卡羅模擬法配合最小平方法(Linear Square Monte Carlo Simulation,簡稱LSM)可用來估計未來現金流量，以計算多因子的模型。最小平方蒙地卡羅法是由到期日以倒推的方式，利用最小平方迴歸計算標的資產的繼續持有條件期望價值。一旦每個時間點上不同路徑的繼續持有價值確認之後，即可直接直接比較每個時間點上選擇權的「馬上履約價值」與「繼續持有價值」，進而做出最適提前履約的決策。在所有價內路徑都決定出最適執行策略後，所有路徑之現金流量折現值取平均即美式選擇權價值。

舉例說明最小平方蒙地卡羅模擬：

茲引用 Longstaff 與 Schwartz(2001)上例子以說明最小平方蒙地卡羅法之建構如下：假設一個不發放現金股利的美式賣權，執行價格為1.10，可在 t1、t2 及 t3 三個時點之下執行，t3 為選擇權之到期日，無風險利率為 6%，在風險測度下模擬出八個股價路徑，如【表一】。

【表一】股價路徑

Path	t=0	t=1	t=2	t=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

為了使選擇權價值最大，因此使用後推的方式。在 t=3 選擇權到期前尚未執行選擇權，選擇權持有者因符合最佳策略而實現的現金流量如【表二】所示。

【表二】在 t=3 時的現金流量

Path	t=0	t=1	t=2	t=3
1				.00
2				.00
3				.07
4				.18
5				.00
6				.20
7				.09
8				.00

若賣權在 t=2 為價內時，則選擇權持有者需考慮是否要立刻執行選擇權。從股價矩陣中得知，在 t=2 有五個路徑的選擇權處於價內。X 代表在 t=2 時五個價內路徑的股價，Y 代表假設賣權在 t=2 時不執行下，從 t=3 所能獲得的對應現 金流量折現值。在這僅使用價內的路徑，是因為價內的路徑與選擇權在決定是否執行較為攸關並且可以用更少的自變數項估計出近似於正確的條件期望函數。將如【表三】所示。

【表三】在 t=2 時的迴歸資料

路徑	Y	X
1	**0.00*0.94176**	**1.08**
2		
3	**0.07*0.94176**	**1.07**
4	**0.18*0.94176**	**0.97**
5		
6	**0.20*0.94176**	**0.77**
7	**0.09*0.94176**	**0.84**
8		

為推論在 t=2 時的股價下繼續持有之期望現金流量，Longstaff 和 Schwartz（2001）使用常數和 X^2 來迴歸 Y。推論出來的條期望函數為：

$$E(Y/X) = -1.070 + 2.983X - 1.813X^2$$

將股價代入條件期望函數中的 X 求得配適值。在 $t=2$ 時的立即執行價值及繼續持有價值將如【表四】所示。

【表四】在 t=2 時的選擇權提早執行策略

路徑	立刻執行	繼續持有
1	0.02	0.0369
2		
3	0.03	0.0461
4	0.13	0.1176
5		
6	0.33	0.1520
7	0.26	0.1565
8		

當把 X 的值代入條件期望函數所得的繼續持有價值時，立刻執行的價值將 會等於原本執行價 1.10-X。這個比較將說明在 t=2 時有三條路徑為最佳執行點， 分別為第四、第六及第七路徑。進而推論出選擇權持有者將收到在 t=2 時所不執行的現金流量將如【表五】所示。

【表五】在 t=2 時的現金流量

Path	t=1	t=2	t=3
1		0.00	.00
2		0.00	.00
3		0.00	.07
4		0.13	.00
5		0.00	.00
6		0.33	.00
7		0.26	.00
8		0.00	.00

由於選擇權只能執行一次，所以在 t=2 時的路徑在 t=3 時則不會有現金流量。將在 t=2 時所收到的現金流量折現一期到 t=1。重複上述方法得到 t=1 時的現金流量。配合終止法則的規定，它將以符合終止法則來明確的決定是否實現現金流量。經由執行策略所決定出執行時點將以 1 標示之, 0 則代表繼續持有。如【表六】所示。

【表六】終止法則

Path	t=1	t=2	t=3
1	0	0	0
2	0	0	0
3	0	0	1
4	1	0	0
5	0	0	0
6	1	0	0
7	1	0	0
8	1	0	0

從最適 執行時點中可得選擇權每一路徑的現金流量，如【表七】所示。

【表七】選擇權現金流量

Path	t=1	t=2	t=3
1	.00	.00	.00
2	.00	.00	.00
3	.00	.00	.07
4	.17	.00	.00
5	.00	.00	.00
6	.34	.00	.00
7	.18	.00	.00
8	.22	.00	.00

因此，透過上面各表的步驟確定各路徑美式選擇權之現金流量，選擇權之價值等於各路徑之現金流量折現至時點零，並將其值取平均值為此選擇權之價值。運用此方法得此美式選擇權之價值等於 0.1144。

二、 結論

以容忍區間為0.1為例，試行蒙地卡羅基本型、Antithetic variate method及Control variate method，在Control variate method的模擬中，以波動率為0.4的買權做為高度相關之衍生性商品，當信賴區間小於等於容忍區間時，即停止模擬。由以下結果可以發現在收斂的效率性上，Control variate method明顯優於Antithetic variate method，而基本型的效率則最差。

最後運用 matlab 之 B-S model 函數計算出的歐式買權價格為 4.3468。而用最小平方蒙地卡羅模擬法模擬美式選擇權，總共模擬 4600000 次，其美式買權價格為 4.3596，標準誤為 0.0303，在 95%的信賴區間則為[4.3002 4.4190]。

〔附件〕程式碼

```
clear all;


clear;clc
S=100;
K=100;
r=0.02;
vol=0.3;
T=4/12;
nsteps=4;
dt=T/nsteps;
dividend=0;
D=6;
r=r-dividend;
CallPutFlag='c';
nsimulations=4600000;
inter=0.1;
vol2=0.4;
[Call, Put] = blsprice(S, K, r, T, vol2);
G=Call;


for i=1:inf;


    MC_ST_posi(1,i)=S;
    MC_ST_negi(1,i)=S;
    MC_ST_hat(1,i)=S;



for j=2:nsteps;
        RAN=randn(1,1);



MC_ST_posi(j,i)=MC_ST_posi(j-1,i)*exp((r-0.5*vol^2)*dt+vol*RAN*dt^0.5
);
```

```
MC_ST_negi(j,i)=MC_ST_negi(j-1,i)*exp((r-0.5*vol^2)*dt+vol*(-RAN)*dt^
0.5);


MC_ST_hat(j,i)=MC_ST_hat(j-1,i)*exp((r-0.5*vol2^2)*dt+vol2*RAN*dt^0.5
);
end


       MS_CT_posi=MC_ST_posi(end,i)-K;
    if MS_CT_posi<0
      MS_CT_posi=0;
    end

    MS_CT_negi=MC_ST_negi(end,i)-K;
    if MS_CT_negi<0
      MS_CT_negi=0;
    end

    MS_CT_hat=MC_ST_hat(end,i)-K;
     if MS_CT_hat<0
        MS_CT_hat=0;
     end

    CALL_simple(i,1)=MS_CT_posi*exp(-r*T);
    meanCALL_simple=mean(CALL_simple);
    sdCALL_simple=std(CALL_simple);

    CALL_antithetic(i,1)=((MS_CT_posi+MS_CT_negi)/2)*exp(-r*T);
meanCALL_antithetic=mean(CALL_antithetic);
sdCALL_antithetic=std(CALL_antithetic);
     CALL_hat(i,1)=MS_CT_hat*exp(-r*T);
     CALL_control=G+CALL_simple-CALL_hat;
     meanCALL_control=mean(CALL_control);
```

```
sdCALL_control=std(CALL_control);

      if i>1

       if inter>2*1.96*sdCALL_simple/i^0.5
          break
       end
    end
    clc,process=i
end


S=S-D*exp(-r*2/12);
nsteps=5;
dt=T/(nsteps-1);

if CallPutFlag=='c',
    z=1;
else
    z=-1;
end;

smat=zeros(nsimulations,nsteps);
CC=zeros(nsimulations,nsteps);
CE=zeros(nsimulations,nsteps);
EF=zeros(nsimulations,nsteps);

smat(:,1)=S;

drift=(r-vol^2/2)*dt;
vsqrdt=vol*dt^0.5;
for q=1:nsimulations,
    st=S;
    curtime=0;
    for k=2:nsteps,
```

```
        curtime=curtime+dt;
        st=st*exp(drift+vsqrdt*randn);
        smat(q,k)=st;
    end
end


CC=smat*0;
CE=smat*0;
EF=smat*0;
st=smat(:,nsteps);
CE(:,nsteps)=max(z*(st-K),0);
CC(:,nsteps)=CE(:,nsteps);
EF(:,nsteps)=(CE(:,nsteps)>0);

paramat=zeros(3,nsteps);

for k=nsteps-1:-1:2,
    st=smat(:,k);
    if k==3
    CE(:,k)=max(z*(st+D-K),0);
    else
    CE(:,k)=max(z*(st-K),0);
    end
      idx=find(CE(:,k)>0);
    Xvec=smat(idx,k);
    Yvec=CC(idx,k+1)*exp(-r*dt);
    p=polyfit(Yvec,Xvec,2);
    CC(idx,k)=polyval(p,Xvec);
    EF(idx,k)=CE(idx,k) > CC(idx,k);
    EF(find(EF(:,k)),k+1:nsteps)=0;
    paramat(:,k)=p;
    idx=find(EF(:,k) == 0);
    CC(idx,k)=CC(idx,k+1)*exp(-r*dt);
    idx=find(EF(:,k) == 1);
```

```
    CC(idx,k)=CE(idx,k);
end

payoff_sum=0;
sdMCAmericanPrice=0;

for v=2:nsteps,
    idx=find(EF(:,v) == 1);
    st=smat(idx,v);
    payoffvec=exp(-r*(v-1)*dt)*max(z*(st-K),0);
    payoff_sum=payoff_sum+sum(payoffvec);
    MCAmericanPrice=payoff_sum/nsimulations;
sdMCAmericanPrice=sdMCAmericanPrice+sum((payoffvec-MCAmericanPrice).^
2)/(nsimulations-1);
end

ierror=2*1.96*sdMCAmericanPrice/q^0.5


[Call, Put] = blsprice(S, K, r, T, vol);
bs_call=Call;
g=MCAmericanPrice-1.96*sdMCAmericanPrice/q^0.5;
h=MCAmericanPrice+1.96*sdMCAmericanPrice/q^0.5;
fprintf('CALL PRICE=%4.4f\n',MCAmericanPrice)
fprintf('std. error=%4.4f\n',sdMCAmericanPrice/q^0.5)
fprintf('[%4.4f %4.4f]\n',g,h)
fprintf('\n')
e=meanCALL_control-1.96*sdCALL_control/i^0.5;
f=meanCALL_control+1.96*sdCALL_control/i^0.5;
fprintf('CALL PRICE=%4.4f\n',meanCALL_control)
fprintf('std. error=%4.4f\n',sdCALL_control/i^0.5)
fprintf('[%4.4f %4.4f]\n',e,f)
fprintf('\n')
c=meanCALL_antithetic-1.96*sdCALL_antithetic/i^0.5;
d=meanCALL_antithetic+1.96*sdCALL_antithetic/i^0.5;
```

```
fprintf('CALL PRICE=%4.4f\n',meanCALL_antithetic)
fprintf('std. error=%4.4f\n',sdCALL_antithetic/i^0.5)
fprintf('[%4.4f %4.4f]\n',c,d)
fprintf('\n')
a=meanCALL_simple-1.96*sdCALL_simple/i^0.5;
b=meanCALL_simple+1.96*sdCALL_simple/i^0.5;
fprintf('CALL PRICE=%4.4f\n',meanCALL_simple)
fprintf('std. error=%4.4f\n',sdCALL_simple/i^0.5)
fprintf('[%4.4f %4.4f]\n',a,b)
```