621-Homework 5

Liting Hu

July 5, 2017

1 Problem 1.

1.1

Implement a basic Monte Carlo using m simulation trials for European Call and Put options:

```
function [Price, SD, SE, Time] = Monte_Carlo(iscall, S0, K, Tm, r, sigma, div, n, m)
% iscall=1: european call option; iscall=0: put option
% S0: Initial stock price
% K: Strike price
% Tm: Time to maturity
% r: Interest rate
% sigma: Volatility
% delta: Continuous dividend rate
% n: Time steps
% m: Quantity of trials
tic;
dt = Tm/n;
S0 = ones(1,m).*S0;
lnS_T = log(S0);

nudt = (r-div-sigma^2/2)*dt;
sigsdt = sigma*sqrt(dt);

for i=1:n
    epsilon = symbol*normrnd(0,1,1,m);
    dS = nudt+sigsdt.*epsilon; % evolve the stock price
    lnS_T = lnS_T+dS;
end
S_T = exp(lnS_T);
 % Stock prices at maturity

if iscall==1
    profits = max(S_T-K,0)*exp(-r*Tm);
else
    profits = max(K-S_T,0)*exp(-r*Tm);
end

Price = mean(profits);
SD = std(profits); % Standard deviation
SE = SD/sqrt(m); % Standard error
Time=toc;
```

The answer will be indicated in next subsection.

1.2

Define a Monte Carlo function using the antithetic variates method:

```
function [Price, SD, SE, Time]=Monte_Carlo_AVR(iscall, S0, K, Tm, r, sigma, div, n, m)
% Monte Carlo valuation with Antithetic Variance reduction
% iscall=1: european call option; iscall=0: put option
% S0: Initial stock price
% K: Strike price
% Tm: Time to maturity
% r: Interest rate
% sigma: Volatility
% delta: Continuous dividend rate
% n: Time steps
% m: Quantity of trials
tic;
dt = Tm/n;
S0 = ones(1,m).*S0;
ST1 = Path_to_T(1, S0, r, sigma, div, dt, n, m);
ST2 = Path_to_T(-1, S0, r, sigma, div, dt, n, m);

if iscall==1
    profits = (max(ST1-K,0)+max(ST2-K,0))/2*exp(-r*Tm);
else
    profits = (max(K-ST1,0)+max(K-ST2,0))/2*exp(-r*Tm);
end

Price = mean(profits);
SD = std(profits);
SE = SD/sqrt(m);
Time=toc;
```

Using the delta?based control variate:

```
function [Price, SD, SE, Time]=Monte_Carlo_DC(iscall, S0, K, Tm, r, sigma, div, beta, n, m)
% Monte Carlo valuation with delta control
% iscall=1: european call option; iscall=0: put option
% S0: Initial stock price
% K: Strike price
% Tm: Time to maturity
% r: Interest rate
% sigma: Volatility
% div: Continuous dividend rate
% n: Time steps
% m: Quantity of trials
tic;

dt = Tm/n;
nudt = (r-div-sigma^2/2)*dt;
sigsdt = sigma*sqrt(dt);
erddt = exp((r-div)*dt);

S0 = ones(1,m).*S0;

St = S0;
cv = zeros(1,m);
```

```
for i=1:n
    t=(i-1)*dt;
    d1 = (log(St/K)+(r-div+sigma^2/2)*(Tm-t))/sigma/sqrt(Tm-t);
    if iscall==1 % Black-Scholes Formulas for Delta
        delta = exp(-div*(Tm-t))*normpdf(d1,0,1);
    else
        delta = exp(-div*(Tm-t))*(normpdf(d1,0,1)-1);
    end
    epsilon = normrnd(0,1,1,m);
    Stn = St.*exp(nudt+sigsdt.*epsilon); % evolve the stock price
    cv = cv+delta.*(Stn-St*erddt);
    St = Stn;
end

if iscall==1
    profits = max(St-K,0)+beta*cv;
else
    profits = max(K-St,0)+beta*cv;
end

Price = mean(profits)*exp(-r*Tm);
SD = std(profits);
SE = SD/sqrt(m);
Time=toc;
```

Using the antithetic variates method and delta?based control variate:

```
function [Price, SD, SE, Time]=Monte_Carlo_AVRDC(iscall, S0, K, Tm, r, sigma, div, beta, n, m)
% Monte Carlo valuation with Antithetic Variance reduction and delta control
% iscall=1: european call option; iscall=0: put option
% S0: Initial stock price
% K: Strike price
% Tm: Time to maturity
% r: Interest rate
% sigma: Volatility
% div: Continuous dividend rate
% n: Time steps
% m: Quantity of trials
tic;

dt = Tm/n;
nudt = (r-div-sigma^2/2)*dt;
sigsdt = sigma*sqrt(dt);
erddt = exp((r-div)*dt);

S0 = ones(1,m).*S0;

St1 = S0;
St2 = S0;
cv1 = zeros(1,m);
cv2 = zeros(1,m);

for i=1:n
```

```
    t=(i-1)*dt;
    d1 = (log(St1/K)+(r-div+sigma^2/2)*(Tm-t))/sigma/sqrt(Tm-t);
    d2 = (log(St2/K)+(r-div+sigma^2/2)*(Tm-t))/sigma/sqrt(Tm-t);
    if iscall==1 % Black-Scholes Formulas for Delta
        delta1 = exp(-div*(Tm-t))*normpdf(d1,0,1);
        delta2 = exp(-div*(Tm-t))*normpdf(d2,0,1);
    else
        delta1 = exp(-div*(Tm-t))*(normpdf(d1,0,1)-1);
        delta2 = exp(-div*(Tm-t))*(normpdf(d2,0,1)-1);
    end

    epsilon = normrnd(0,1,1,m);
    Stn1 = St1.*exp(nudt+sigsdt.*epsilon); % evolve the stock price
    Stn2 = St2.*exp(nudt-sigsdt.*epsilon);
    cv1 = cv1+delta1.*(Stn1-St1*erddt);
    cv2 = cv2+delta2.*(Stn2-St2*erddt);
    St1 = Stn1;
    St2 = Stn2;
end

if iscall==1
    profits = (max(St1-K,0)+beta*cv1 + max(St2-K,0)+beta*cv2)/2;
else
    profits = (max(K-St1,0)+beta*cv1 + max(K-St2,0)+beta*cv2)/2;
end

Price = mean(profits)*exp(-r*Tm);
SD = std(profits);
SE = SD/sqrt(m);
Time=toc;
```

All the results are showed in the table below:

	MC	MC.AVC	MC.DC	MC.AVCDC
Price	9.1349	9.1385	9.1296	9.1334
SD	13.6937	9.6744	11.0993	7.8494
SE	0.0137	0.0097	0.0111	0.0078
Time(s)	6.8481	13.8185	13.9291	21.9580

2 Problem 2.

2.1

Apply the Euler discretization schemes and implement all the five schemes listed.

```
function [Price, Bias, RMSE, Time]=MC_Heston(Scheme, S0, K, V0, Tm, r, sigma, kappa, theta, rho, n
% For european call option
% S0: Initial stock price
% K: Strike price
% V0: Initial volatility
% Tm: Time to maturity
% r: Interest rate
% sigma: Volatility variance
% kappa: speed of mean-reversion of the variance
```

```
% theta: the long-term average variance
% rho: Correlation coefficient between two Brownian motions
% n: Time steps
% m: Quantity of trials

tic;
dt = Tm/n;
St = ones(1,m).*S0;
lnSt = log(St);
Vt1 = ones(1,m).*V0;
dWv = normrnd(0,1,n,m)*sqrt(dt);
dZ = normrnd(0,1,n,m)*sqrt(dt);
dWs = rho*dWv+sqrt(1-rho^2)*dZ;

for i=1:n
    if Scheme==1 || Scheme==4 || Scheme==5
        Vt2 = max(Vt1,0); % Effective variance
    else
        Vt2 = abs(Vt1);
    end

    lnSt = lnSt+(r-Vt2/2)*dt+sqrt(Vt2).*dWs(i,:);
    if Scheme==1
        f1=max(Vt1,0);
        f2=f1;
        f3=f1;
    elseif Scheme==2
        f1=abs(Vt1);
        f2=f1;
        f3=f1;
    elseif Scheme==3
        f1=Vt1;
        f2=f1;
        f3=abs(Vt1);
    elseif Scheme==4
        f1=Vt1;
        f2=f1;
        f3=max(Vt1,0);
    else
        f1=Vt1;
        f2=max(Vt1,0);
        f3=f2;
    end

    Vt1 = f1+kappa*(theta-f2)*dt+sigma.*f3.^(1/2).*dWv(i,:);



end
St = exp(lnSt);

Calls = max(St-K, 0);
Price = mean(Calls)*exp(-r*Tm);
Bias = abs(Price-6.8061);
```

```
RMSE = rms(Calls-6.8061);

Time = toc;
```

All the results are showed in the table below: (Quantity of trials is 1e6.)

Scheme	Absorption	Reflection	Higham and Mao	Partial truncation	Full truncation
Price	6.8812	6.9201	6.8567	6.8285	6.7791
Bias	0.0751	0.1140	0.0506	0.0224	0.0270
RMSE	7.8474	7.9933	7.7839	7.6809	7.6613
Time(s)	2.6324	2.7505	2.8136	2.7355	2.6148

3 Problem 3.

3.1

By function chol in matlab, the lower triangular matrix L is easy to calculate.

```
A=[1,0.5,0.2;0.5,1,-0.4;0.2,-0.4,1];
L = chol(A,'lower');
```

3.2

To calculate all m paths, define function Correlated_BM:

```
function [St]=Correlated_BM(S0,Tm,dt,A,sigma,MU,m)
% S0: Initial stock prices
% Tm: Time to maturity
% dt: Time intervel
% A: Correlation matrix
% sigma: Volatilities
% MU: Risk free rates
% m: Quantity of trials

L = chol(A,'lower'); % Cholesky factorization

n = Tm/dt;
nudt = (MU-0.5*sigma.^2)'*dt*ones(1,m);
St = zeros(n+1,m,3);
St(1,:,1)=S0(1);
St(1,:,2)=S0(2);
St(1,:,3)=S0(3);
lnSt = log(St);

for i=1:n
    Zt = normrnd(0,1,3,m)*sqrt(dt);
    Wt = L*Zt;
    for j=1:3
        lnSt(i+1,:,j)=lnSt(i,:,j)+nudt(j,:)+Wt(j,:).*sigma(j);
    end
end

St = exp(lnSt);
```

And then, draw a plot:

```
S0 = [100,101,98];
MU = [0.03,0.06,0.02];
sigma = [0.05,0.2,0.15];

Tm = 100/365;
m = 1000;
dt = 1/365;

St = Correlated_BM(S0,Tm,dt,A,sigma,MU,m);
Onepath = St(:,1,:);
Onepath = reshape(Onepath,(Tm/dt+1)*1,3);
plot(Onepath)
legend('Stock1','Stock2','Stock3')
```

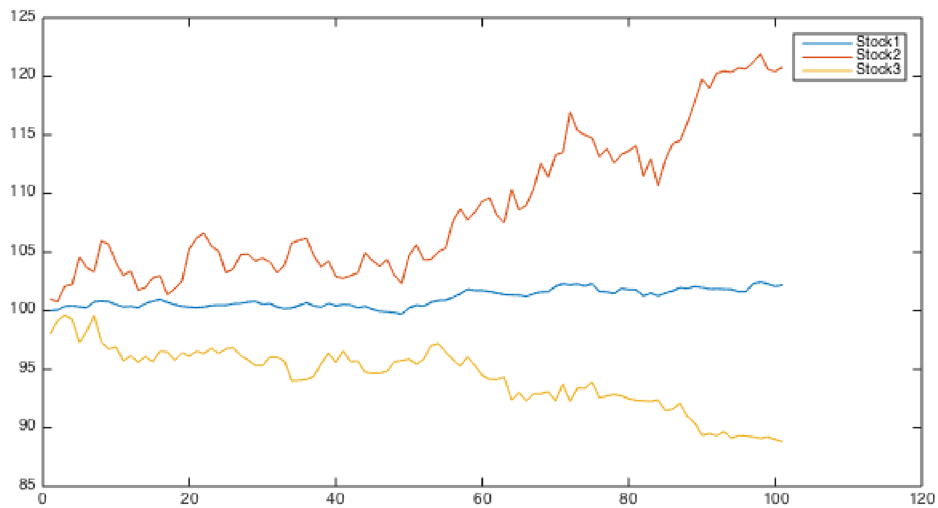

Figure 1:

3.3

For a simple average basket:

```
function [Price]=Basket_option(iscall,S0,K,Tm,dt,A,sigma,MU,m)
% iscall=1: european call option; iscall=0: put option
% S0: Initial stock prices
% K: Strike price
% Tm: Time to maturity
% dt: Time intervel
% A: Correlation matrix
% sigma: Volatilities
% MU: Risk free rates
% m: Quantity of trials

St = Correlated_BM(S0,Tm,dt,A,sigma,MU,m);
```

```
Ut = mean(St,3); % a simple average basket
UT = Ut(end,:);
Price = mean(max((-1)^iscall*(K-UT),0));
```

The price of this basket call option is 2.0076.

3.4

For this exotic option:

```
function [Price]=Basket_exotic_option(S0,K,B,Tm,dt,A,sigma,MU,m)
% For european call option
% S0: Initial stock prices
% K: Strike price
% B: Barrier
% Tm: Time to maturity
% dt: Time intervel
% A: Correlation matrix
% sigma: Volatilities
% MU: Risk free rates
% m: Quantity of trials

St = Correlated_BM(S0,Tm,dt,A,sigma,MU,m);
UT = zeros(1,m); % Define option price matrix
for i=1:m
    pathi = St(:,i,:);
    pathi = reshape(pathi,(Tm/dt+1)*1,3);

    if max(pathi(:,2)) > B % condition (i)
        UT(i) = max(pathi(end,2)-K,0);
        continue;
    elseif max(pathi(:,2)) > max(pathi(:,3)) % condition (ii)
        UT(i) = max(pathi(end,2)-K,0)^2;
        continue;
    elseif mean(pathi(:,2)) > mean(pathi(:,3)) % condition (iii)
        UT(i) = max(mean(pathi(:,2))-K,0);
        continue;
    else % condition (iv)
        ST = mean(pathi(end,:));
        UT(i) = max(ST-K,0);
    end
end

Price = mean(UT)*exp(-mean(MU)*Tm);
```

Price of this exotic option is 5.8203.

All the matlab code can be found in zip file.