Homework #3

Jingyi Guo
Pittsburgh

1. Pricing an arithmetic Asian option

(a) standard Monte Carlo simulation

Code:

```
r=0.05;
S0=100;
mu=0.1;
sig=0.1;
T=1;
N=52;
K=100;
n=1000;
Sastore=zeros(N+1,n);
Sastore(1,:)=S0*ones(1,n); %initial value
delta=T/N;

for j=2:(N+1)
    Sastore(j,:)=Sastore(j-1,:).*exp((r-
1/2*sig^2)*delta+sig*sqrt(delta).*randn(1,n));
end

Ca=zeros(1,n);
for i=1:n
    Ca(i)=max(1/N*sum(Sastore(2:(N+1),i))-K,0)*exp(-r*T);
end

Cbara=mean(Ca)
stderra=std(Ca)/sqrt(n)
```

Output:

```
Cbara =

    3.7723

stderra =

    0.1358
```

So the estimated price of the option is 3.7723, standard error is 0.1358.

(b) Antithetic path

```
Sbstore1=zeros(N+1,n);
Sbstore1(1,:)=S0*ones(1,n); %initial value
Sbstore2=zeros(N+1,n);
Sbstore2(1,:)=S0*ones(1,n); %initial value
for j=2:(N+1)
    z=randn(1,n);
    Sbstore1(j,:)=Sbstore1(j-1,:).*exp((r-
1/2*sig^2)*delta+sig*sqrt(delta).*z);
    Sbstore2(j,:)=Sbstore2(j-1,:).*exp((r-
1/2*sig^2)*delta+sig*sqrt(delta).*(-z));
end

Cb=zeros(1,n);
for i=1:n
    Cb(i)=1/2*(max(1/N*sum(Sbstore1(2:(N+1),i))-
K,0)+max(1/N*sum(Sbstore2(2:(N+1),i))-K,0))*exp(-r*T);
end
Cbarb=mean(Cb)
stderrb=std(Cb)/sqrt(n)
```

Output:

Cbarb =

3.6859

stderrb =

0.0544

So the estimated price of the option is 3.6859, standard error is 0.0544.

(c) Using S_N as control variable

Code:

```
Y=Ca;
X=Sastore(N+1,:);
a=-corr(X',Y')*std(Y)/std(X);
Cbarc=Cbara+a*(mean(X)-S0*exp(r*T))
stderrc=std(Y)/sqrt(n)*sqrt(1-(corr(X',Y'))^2)
```

Output:

Cbarc =

3.7200

stderrc =

0.0790

So the estimated price of the option is 3.7200, standard error is 0.0790.

(d) Using price of geometric option price as control variable - 1

```
Yd=Ca;
Xd=zeros(1,n);
for i=1:n
    Xd(i)=(max((prod(Sastore(2:(N+1),i)))^(1/N)-K,0))*exp(-
r*T);
end

% check that rho of Xd and Yd is huge
% rho=corr(Xd',Yd')
ad=-corr(Xd',Yd')*std(Yd)/std(Xd);

% calculate real EXd using approximation:
% sig=sig/sqrt(s)
% d=r/2+sig^2/12
EXd=european_call_div(S0, K, r, sig/sqrt(3), T,
r/2+sig^2/12);

Cbard=Cbara+ad*(mean(Xd)-EXd)
stderrd=std(Yd)/sqrt(n)*sqrt(1-(corr(Xd',Yd'))^2)

% define function for BS formula with dividend rate q
function call_price=european_call_div(S0, K, r, sig, T, q)
    dplus=1/(sig*sqrt(T))*(log(S0/K)+(r-q+1/2*sig^2)*T);
    dminus=1/(sig*sqrt(T))*(log(S0/K)+(r-q-1/2*sig^2)*T);

call_price = exp(-q*T)*S0*normcdf(dplus)-
normcdf(dminus)*K*exp(-r*T);
```

Output:

Cbard =

3.6411

stderrd =

0.0021

So the estimated price of the option is 3.6411, standard error is 0.0021.

(e) Using price of geometric option price as control variable – 2

Code:

```
% calculate real EXe using:
sig_e=sig*sqrt((N+1)*(2*N+1)/(6*N^2));
d_e=r*(N-1)/(2*N)+sig^2*((N+1)*(N-1)/(12*N^2));
EXe=european_call_div(S0, K, r, sig_e, T, d_e);
Cbare=Cbara+ad*(mean(Xd)-EXe)
stderre=std(Yd)/sqrt(n)*sqrt(1-(corr(Xd',Yd'))^2)
```

Output:

```
Cbare =

    3.7030

stderre =

    0.0021
```

So the estimated price of the option is 3.7030, standard error is 0.0021.

2. Control variables and importance sampling

(a) Using standard Monte Carlo simulation

Code:

```
r=0.05;
S0=100;
mu=0.1;
sig=0.2;
T=1;
N=52;
K=120; % or 140, 160
n=10000;

Z=randn(1,n);
STa=S0*exp((r-1/2*sig^2)*T+sig*sqrt(T).*Z);
Ca=exp(-r*T)*max(STa-K,0);
Cabar=mean(Ca)
stderra=std(Ca)/sqrt(n)
```

Result:

	K=120	K=140	K=160
Estimation	3.2971	0.7751	0.1425
Standard error	0.0871	0.0430	0.0176

(b) Using put-call parity

Idea: first simulate 10,000 paths of put prices, then use Put-Call Parity to compute the call prices of these 10,000 paths.

Code:

```
Pb=exp(-r*T)*max(K-STa,0);
% Put-Call Parity:
% C-P=S0-K*exp(-r*T)
Cb=Pb+S0-K*exp(-r*T);
Cbbar=mean(Cb)
stderrb=std(Cb)/sqrt(n)
```

Result:

	K=120	K=140	K=160
Estimation	3.3165	0.7095	0.2255
Standard error	0.1486	0.1838	0.1956

(c) Put-call Parity + Using S_T as control variable

Idea: first compute call prices of these 10,000 paths in the same manner as in b, then use control variable adjustments

Code:

```
Y=Cb;
X=STa;
a=-corr(X',Y')*std(Y)/std(X);
Ccbar=Cbbar+a*(mean(X)-S0*exp(r*T))
stderrc=std(Y)/sqrt(n)*sqrt(1-(corr(X',Y'))^2)
```

Result:

	K=120	K=140	K=160
Estimation	3.1810	0.7867	0.1977
Standard error	0.0581	0.0368	0.0161

(d) Put-call Parity + Using S_T as control variable

Code:

```
L=(log(K/S0)-(r-0.5*sig^2)*T)/(sig*sqrt(T));
Ud=rand(1,n);
Xd=norminv(Ud.*(1-normcdf(L))+normcdf(L),0,1);
STd=S0*exp((r-1/2*sig^2)*T+sig*sqrt(T).*Xd);
Cd=exp(-r*T)*(STd-K)*(1-normcdf(L));
Cdbar=mean(Cd)
stderrd=std(Cd)/sqrt(n)
```

Result:

	K=120	K=140	K=160
Estimation	3.2702	0.7836	0.1614
Standard error	0.0294	0.0073	0.0016

Compared to (a) and (b), results in (d) have much smaller standard errors while producing approximately similar results.

3. Conditional Monte Carlo and importance sampling: barrier options

(a) standard Monte Carlo

Code:

```
r=0.05;
S0=95;
sig=0.15;
T=0.25;
m=50;
% (H,K)=(94,96) or (90,96) or (85,96) or (90, 106)
H=94;
K=96;
n=100000;
Sastore=zeros(m+1,n);
Sastore(1,:)=S0*ones(1,n); %initial value
delta=T/m;

for j=2:(m+1)
    Sastore(j,:)=Sastore(j-1,:).*exp((r-
1/2*sig^2)*delta+sig*sqrt(delta).*randn(1,n));
end

Ca=zeros(1,n);
for i=1:n
    if (min(Sastore(:,i))<H)
        if (Sastore(m+1,i)-K>0)
            Ca(i)=1*exp(-r*T);
        else
            Ca(i)=0;
        end
    else
        Ca(i)=0;
    end
end
Cbara=mean(Ca)
stderra=std(Ca)/sqrt(n)
```

Result:

	Estimation	Standard error
H=94, K=96	0.2996	0.0014
H=90, K=96	0.0425	6.3381e-04
H=85, K=96	5.5304e-04	7.3883e-05
H=90, K=106	0.0013	1.1339e-04

(b) Conditional Monte Carlo

i. Pricing a digital option

Pricing a digital option at time 0:

$$P(S_0, K, r, \sigma, T) = \tilde{\mathbb{E}}\left(e^{-rT} I\{S(T)>K\}\right)$$
$$= e^{-rT}\tilde{\mathbb{P}}(S(T)>K)$$
$$= e^{-rT}\tilde{\mathbb{P}}\left(S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma(\tilde{W}(T)-\tilde{W}(0))} > K\right)$$

Since $\tilde{W}(T)-\tilde{W}(0) \sim N(0, T-t)$, Let $\tilde{W}(T)-\tilde{W}(0) = \sqrt{T}\, Z$, then $Z \sim N(0,1)$

So $$P(S_0, K, r, \sigma, T) = e^{-rT}\mathbb{P}\left(S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}Z} > K\right)$$
$$= e^{-rT}\mathbb{P}\left(Z > \frac{1}{\sigma\sqrt{T}}\left[\log\left(\frac{K}{S_0}\right)-(r-\tfrac{1}{2}\sigma^2)T\right]\right)$$
$$= e^{-rT}N\left(\frac{1}{\sigma\sqrt{T}}\left[\log\left(\frac{S_0}{K}\right)+(r-\tfrac{1}{2}\sigma^2)T\right]\right)$$
$$= e^{-rT}N(d_-)$$

where $d_- = \frac{1}{\sigma\sqrt{T}}\left[\log\left(\frac{S_0}{K}\right)+(r-\frac{1}{2}\sigma^2)T\right]$, $N(\cdot)$ is normal dist. cdf

```
% define function for digital call
function call_price=digital_call(S0, K, r, sig, T)
    dminus=1/(sig*sqrt(T))*(log(S0/K)+(r-1/2*sig^2)*T);
call_price = exp(-r*T)*normcdf(dminus);
```

ii.

Code:

```
% (H,K)=(94,96) or (90,96) or (85,96) or (90, 106)
H=94;
K=96;
Sbstore=zeros(m+1,n);
Sbstore(1,:)=S0*ones(1,n); %initial value
Cb=zeros(1,n);

for i=1:n
```

```
    for j=2:(m+1)
        Sbstore(j,i)=Sbstore(j-1,i).*exp((r-
1/2*sig^2)*delta+sig*sqrt(delta).*randn(1));
        if (Sbstore(j,i)<H)
            Cb(i)=exp(-delta*(j-1))*digital_call(Sbstore(j,i),
K, r, sig, T-(j-1)*delta);
            break
        end
    end
end
Cbarb=mean(Cb)
stderra=std(Cb)/sqrt(n)
```

Result:

	Estimation	Standard error
H=94, K=96	0.2919	4.9723e-04
H=90, K=96	0.0390	1.9680e-04
H=85, K=96	5.0940e-04	9.1640e-06
H=90, K=106	0.0012	8.8262e-06

iii. Multiply our estimations by 10,000 to compare with the table in Glasserman's book, note that there is no need to alter the calculation of variance ratio

	Estimation	Variance Ratio
H=94, K=96	2919.4	8.3371
H=90, K=96	390.3872	10.3716
H=85, K=96	5.0940	65.0007
H=90, K=106	12.4109	165.0432

This table has similar results with Glasserman's.

4. Discrete versus continuous pricing

(a) standard Monte Carlo

Code:

```
r=0.10;
S0=100;
sig=0.30;
T=0.2;
N=25; % or 50
H=95;
K=100;
n=100000;
Sastore=zeros(N+1,n);
```

```
Sastore(1,:)=S0*ones(1,n); %initial value
delta=T/N;

for j=2:(N+1)
    Sastore(j,:)=Sastore(j-1,:).*exp((r-
1/2*sig^2)*delta+sig*sqrt(delta).*randn(1,n));
end

Ca=zeros(1,n);
for i=1:n
    if (min(Sastore(:,i))<H)
        Ca(i)=exp(-r*T)*max(Sastore(j,i)-K,0);
    end
end
Cbara=mean(Ca)
stderra=std(Ca)/sqrt(n)
```

Result:

	N=25	N=50
Estimation	1.2578	1.4381
Standard error	0.0125	0.0134

(b) conditional Monte Carlo

Code:

```
Sbstore=zeros(N+1,n);
Sbstore(1,:)=S0*ones(1,n); %initial value
Cb=zeros(1,n);

for i=1:n
    for j=2:(N+1)
        Sbstore(j,i)=Sbstore(j-1,i).*exp((r-
1/2*sig^2)*delta+sig*sqrt(delta).*randn(1));
        if (Sbstore(j,i)<H)
            Cb(i)=exp(-delta*(j-
1))*european_call_div(Sbstore(j,i), K, r, sig, T-(j-1)*delta,
0);
            break
        end
    end
end
Cbarb=mean(Cb)
stderrb=std(Cb)/sqrt(n)
```

Result:

	N=25	N=50
Estimation	1.2450	1.4413
Standard error	0.0124	0.0135

In the case of N approaching infinity, we have Hull formula:

$$c_{\text{di}} = S_0 e^{-qT}(H/S_0)^{2\lambda}N(y) - Ke^{-rT}(H/S_0)^{2\lambda-2}N(y - \sigma\sqrt{T})$$

where

$$\lambda = \frac{r - q + \sigma^2/2}{\sigma^2}$$

$$y = \frac{\ln[H^2/(S_0K)]}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}$$

So we have: the continuous pricing result of this option is 1.9466. The results in (a)(b) are all smaller than in continuous case, which is reasonable.

(c)

Code:

Result:

	N=25	N=50
Estimation	1.2147	1.3851
Standard error	0.0016	0.0014

The standard errors in (c) are much smaller than in (a) and (b).