Marjo Kaci

Luca Regis

MATLAB

8 May 2015

Pricing a European Put option using variance reduction techniques

In this paper, I will describe the simulation process and techniques followed in pricing a European option on an asset evolving according to a Geometric Brownian Motion and some techniques commonly used to reduce the variance of simulated prices of our option price.

First, I will calculate the price of the option using Black and Scholes formula. The price of a Call option using BS formula is:

$$\begin{aligned} C(S,t) &= N(d_1)S - N(d_2)Ke^{-r(T-t)} \\ d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\ d_2 &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)(T-t)\right] \\ &= d_1 - \sigma\sqrt{T-t} \end{aligned}$$

The price of a corresponding put option based on put–call parity is:

$$\begin{aligned} P(S,t) &= Ke^{-r(T-t)} - S + C(S,t) \\ &= N(-d_2)Ke^{-r(T-t)} - N(-d_1)S \end{aligned}$$

S , be the price of the stock,

$C(S,t)$ The price of a European call option and

$P(S,t)$ the price of a European put option.

K, the strike price of the option.

r, the annualized risk-free interest rate, continuously compounded (the force of interest).

σ, the standard deviation of the stock's returns

Finally, we will use N(x) to denote the standard normal cumulative distribution function:

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{z^2}{2}}\, dz$$

The code used to calculate the Put option Black and Scholes price is:

```
d1=(log(S0/K)+(r+1/2*sigma^2)*T)/(sigma*sqrt(T));
d2=(log(S0/K)+(r-1/2*sigma^2)*T)/(sigma*sqrt(T));
P=exp(-r*T)*K*normcdf(-d2)-S0*normcdf(-d1);
```

After we found the Black and Scholes price of the option, we can make a simulation of the price using Monte Carlo simulation. The fact that derivatives prices (which are solution to PDEs) can be written as expectations (via the Feynman-Kac theorem) allows us to state a link between derivatives pricing and Monte Carlo simulation. Moreover, we do that by estimating E[e^rT *(S(T) - K)+] by the following algorithm:

1. For i=1,...n generate asset prices at T, i.e. generate Zi N(0; 1)
2. Compute Si(T) = S0exp((r – ½*sigma^2)*T+sigma*sqrt(T)*Zi)
3. Set Pi = e^rT* (S(T) - K)+
4. Compute Pn* = (P1+:::+Pn)/n

Where Pn* by the LLN and CLT is an unbiased estimator of the price P. Prices here, as we can see from the formula evolve according to a Geometric Brownian Motion. The simulation code used for the Monte Carlo simulation is the following:

```
c=cputime;
mu=(r-1/2*sigma^2)*T;
s=sigma*sqrt(T);
% Preallocate the ST and PT
ST=zeros(N_Sim,1);
PT=zeros(N_Sim,1);


% For loop to generate S(T) and compute price
for i=1:N_Sim
```

```
ST(i)=S0.*exp(mu+s*randn);
PT(i)=exp(-r*T)*max(0,ST(i)-K);
end
P_MC=mean(PT); % Price of the option using Monte Carlo
S_MC=std(PT);  % Standart deviation
time=cputime-c; % time of the simulation
```

Quality of the approximation

We can see from the simulations that, the Black and Scholes price and Monte Carlo price are very similar:

`P = 9.7466 P_MC = 9.8006`

In order to improve the approximation we try to reduce the confidence interval $\bar{\theta}_n \pm z_{d/2}\frac{\bar{\sigma}}{\sqrt{n}}$

We can do so by increase the number of simulations from N_Sim=10000 to N_Sim=1000000 and so the approximation will increase: `P = 9.7466 P_MC =  9.7190`

In order to improve furthermore the approximation we can also use some methods that reduce the variance. As we can see from the above formula, a reduction in the standard deviation will bring a smaller confidence interval and so a better estimator of the Black and Scholes price of the option. There are two techniques for variance reduction that we will analyze in this project, one is the Antithetic variables technique and the second is Control variate technique.

Antithetic variables

Here I will use a different technique in the calculation of the estimator. Instead of the classic estimator, $\bar{\theta}_n(X) := \frac{1}{n}\sum_{i=1}^{n} g(X_i)$ I will use another estimator, which will reduce the variance. In general the antithetic variable estimator is: $\hat{\theta}_{AV} = \frac{g(X)+g(Y)}{2}$

Where Y is identically distributed wrt X and negatively correlated. The algorithm used for this procedure if the following:

1. Generate U1….. Un i.i.d. uniform random variables.
2. Estimate $$\hat{\theta}_{AV} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left[g(F^{-1}(U_i) + g(F^{-1}(1-U_i))\right]$$

The code used for the pricing of the option using this estimator is the following:

```
%                          ANTITHETIC VARIABLES

mu=(r-1/2*sigma^2)*T;
s=sigma*sqrt(T);

% X e Y is the F^-1(U1) and F^-1(U2) which are negatively correlated
X=randn(N_Sim,1);
Y=-X;
% ST1 and ST2 need to be correlated to get a smaller variance and they are
% since X and Y are negatively correlated.
ST1=S0.*exp(mu+s.*X);
ST2=S0.*exp(mu+s.*Y);

% Price of the put option for different simulations using a different
estimator
P=exp(-r*T).*(max(0,K-ST1)+max(0,K-ST2))/2;

% Price of the option using Antithetic variables
P_MC=mean(P);
% Variance using Antithetic variables
S_MC=std(P);
```

What we can observe from the simulation is that the deviation decreased and the time to compute such a calculation reduces. The results of the simulation on the deviation reduction are:

S_MC = 14.8698 to a smaller deviation S_MC_AV = 7.9526

The time for the calculations also reduces from time = 0.031 to almost 0.

CONTROL VARIATES

It is the most effective and broadly applicable technique for improving MC efficiency.

We use two outputs to calculate the estimator, one if the price of the derivative and the other output has known mean and is used to reduce the variance of the estimator. The second output is the price of the underlying. Consider the i.i.d. pairs (Xi; Yi). For any fixed b we can compute

Yi(b) = Yi - b (Xi - E[X]) whose sample mean is

$$\text{Y\_avg (b)} = \text{Y\_avg} - \text{b*1/n*} \sum_{k=0}^{n} (\text{Yi} - \text{b}(\text{X} - \text{E}[\text{X}]))$$

The Y_abg(b) is our control variate estimator which is unbiased and consistent.

We can find the b such that it minimize the variance of each Yi(b) and that is :

$$b^* = \frac{\sigma_Y}{\sigma_X}\rho_{XY} = \frac{Cov[X,Y]}{Var[X]}$$

As in principle Cov(X; Y) and V ar(X) may be unkown, we substitute b by its population estimate: $\hat{b}_n = \frac{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}$

The code used for such simulation is the following:

```
CONTROL VARIATES

% We could  use PT and ST generated when we use  Eur_C function or
% generate them with the following code:
disp('Price of the PUT option using CONTROL VARIATES estimator ')

%{ %}
z=cputime;
N_Sim1=1000000;



mu=(r-1/2*sigma^2)*T;
s=sigma*sqrt(T);
ST=zeros(N_Sim1,1); % the price of the underlying
PT=zeros(N_Sim1,1); % the option price of different simulations
for i=1:N_Sim1
ST(i)=S0.*exp(mu+s*randn);
PT(i)=exp(-r*T)*max(0,K-ST(i));
end




% Choose two outpus of N_Sim simulations
OP1=PT; % First output: prices of the dericative - vettore con tt i prezzi
dell N_sim simulazioni
OP2=ST; % Second output is the price of the underlying with known mean.
% Calculate the covariance of the two inputs
VarOP2=S0^2*exp(2*mu*T)*(exp(sigma^2*T)-1); %Variance of a log-normal
distributed random variable
Cov_matrix=cov(OP1,OP2);
COV=Cov_matrix(1,2);
% Calculate the mean of the second output
OP2_avg=S0*exp(r*T);

% This is the b for such the variance is minimal
b_min=COV/VarOP2;



% Define the estimator that will be used to reduce the variance
P_b=zeros(N_Sim,1);
ST=zeros(N_Sim,1);
```

```
PT=zeros(N_Sim,1);
for i=1:N_Sim
ST(i)=S0.*exp(mu+s*randn);
PT(i)=exp(-r*T)*max(0,K-ST(i));
P_b(i)=PT(i) - b_min*(ST(i) - OP2_avg);
end

P_MC_CV=mean(P_b);
S_MC_CV=std(P_b);


%Calculate time
time_CV=cputime-z;
```

From the results we can see that the standard deviation reduces to `S_MC_cv =12.6420`.

However though the computation time increases significantly and for a higher number of simulation the computation time of the code might take a while.

Sensitivity analysis

Here we fix the strike price to K=75 and change the standard deviation of the sock return from 0.6 and see how the effectiveness of the control variates variance reduction technique changes.
The nr of simulation: `N_Sim=1000000`.
By making such a change we can see that:

Results for sigma=0.6 on the control variate estimator are:

- Black and Scholes price of the option and the Monte Carlo simulated price

P = 9.7466

P_MC = 9.7190

S_MC =14.8698

time = 0.2969

Price of the PUT option using CONTROL VARIATES estimator

P_MC_CV = 9.7296

S_MC_CV = 12.6420

time_CV = 0.4219

- Results for sigma=0.7 on the control variate estimator are:

Black and Scholes price of the option and the Monte Carlo simulated price

P = 12.6451

P_MC =12.6561

S_MC = 17.4170

time = 0.1563

Price of the PUT option using CONTROL VARIATES estimator

P_MC_CV = 12.6349

S_MC_CV = 15.5771

time_CV = 0.4375

- Results for sigma=0.2 on the control variate estimator are:

Black and Scholes price of the option and the Monte Carlo simulated price

P = 0.4585

P_MC = 0.4587

S_MC = 2.1219

time =0.1719

Price of the PUT option using CONTROL VARIATES estimator

P_MC_CV = 0.4569

S_MC_CV = 1.9434

time_CV = 0.4219

What we can derive from this analysis is that when we increase the sigma to 0.7 we can see that the variance increase and that the variance of our control variates estimator is lower then the variance of the option price when we do the classic Monte Carlo simulation. Also the approximation of the Black and Scholes price decreases as a consequence of the increase in the variance.

When we reduce the sigma to 0.2 we can see that the standard deviation of the prices will decrease and the standard deviation of the control variate estimator will still remain lower than that of the Monte Carlo simulation. In this case the approximation of using the control variate is much better then before since the variance is also reduced.