European Option Simulation

Sascha Strobl

7/15/2018

Introduction

Our project is to simulate the price of a European call and put option and compare our results with the prices according to the Black-Scholes formula. To demontsrate that our simulaion program gives a good approximation we chose the following example: the initial stock price is $ 124, the strike price of the option is $ 142, the volatility is 30%, the risk free rate is 2% and the option will expire in 5 years. We implement two variance reduction techniques: the antithetic variable method and the control variate method. For the former method, we use one minus the uniform random variable generated for the direct method applied to the inverse of the standard normal cdf . For the latter method, we use the sum of the z-scores multiplied with the volatility and the square root of the time step as control variate. This time step is user defined as proporation of a year the time advances until the next stock price is simulated.

R code

The first section of the code initializes all the variables.

```
Stock_price = 124
Strike_price = 142
Volatility = 0.3
Risk_free_rate = 0.02
Maturity_in_years = 5

n = 10000
P = 20
Time_step = 0.05
t = Maturity_in_years/Time_step

End_Stock_price = numeric(n)
End_Stock_price_antithetic = numeric(n)

OptionValue_discounted = numeric(n)
OptionValue_discounted_antithetic = numeric(n)

Put_OptionValue_discounted = numeric(n)
Put_OptionValue_discounted_antithetic = numeric(n)

OptionPrice = numeric(P)
Put_OptionPrice = numeric(P)
OptionPrice_antithetic = numeric(P)
Put_OptionPrice_antithetic = numeric(P)
OptionPrice_cont = numeric(P)
Put_OptionPrice_cont = numeric(P)

h = numeric(n)

options(warn=-1)
```

The following section contains the core code with which simulates the option price and includes the variance reduction. It contains two loops. The outer loop has 20 iterations and the inner 10,000 but these are only arbitrary choices. Then, t uniform random variables (u) are created, where t is the number of time steps and defined as maturity devided by time step (both of which are user defined). Next, u is used to get t z-scores, which in turn is used to calculate the log stock price according to following formula:

$$ln(S_{t+1}) = ln(S_t) + (r_f - \frac{1}{2}\sigma^2) * \tau + \sigma^2\sqrt{\tau}\Phi^{-1}(u)$$

The stock price at the option maturity is then $ln(S_T)$ after t steps. Finally, the value of the call or put option is evaluated and discounted: Call: $max\{e^{ln(S_T)} - K, 0\}e^{-r_f T}$ and Put: $max\{K - e^{ln(S_T)}, 0\}e^{-r_f T}$.
Next, the mean to all 10,000 generated discounted option values is applied.

```
for (k in 1:P){
  for (j in 1:n){
    u = runif(t)
    r = qnorm(u)
    r_anti = qnorm(1-u)
    h[j] = Volatility*sqrt(Time_step)*sum(r) #our control variate
    log_End_Stock_price = log(Stock_price)+  #log of stock price at maturity of option
      (Risk_free_rate-0.5*Volatility^2)*Time_step*t+h[j]
    #log of stock price at maturity of option of the antithetic variable
    log_End_Stock_price_anti = log(Stock_price)+
      (Risk_free_rate-0.5*Volatility^2)*Time_step*t+
      Volatility*sqrt(Time_step)*sum(r_anti)
    #discounted option value, ie. maximum of stock price minus strike price and zero
    OptionValue_discounted[j] = max(exp(log_End_Stock_price)-
      Strike_price,0)*exp(-t*Time_step*Risk_free_rate)
    #discounted average of 'regular' option value and antithetic option value
    OptionValue_discounted_antithetic[j] = (max(exp(log_End_Stock_price)-
      Strike_price,0)*exp(-t*Time_step*Risk_free_rate)+max(exp(log_End_Stock_price_anti)-
      Strike_price,0)*exp(-t*Time_step*Risk_free_rate))/2
    #discounted put option value, i.e. maximum of strike price minus stock price and zero
    Put_OptionValue_discounted[j] = max(Strike_price -
      exp(log_End_Stock_price),0)*exp(-t*Time_step*Risk_free_rate)
    #discounted average of 'regular' option value and antithetic option value (put)
    Put_OptionValue_discounted_antithetic[j] = (max(Strike_price -
      exp(log_End_Stock_price),0)*exp(-t*Time_step*Risk_free_rate)+max(Strike_price -
      exp(log_End_Stock_price_anti),0)*exp(-t*Time_step*Risk_free_rate))/2
  }
  c_star = -cov(h,OptionValue_discounted)/var(h)
  c_star_Put =  -cov(h,Put_OptionValue_discounted)/var(h)
  OptionPrice[k] = mean(OptionValue_discounted)
  OptionPrice_antithetic[k] = mean(OptionValue_discounted_antithetic)
  OptionPrice_cont[k] = mean(OptionValue_discounted + c_star*(h))
  Put_OptionPrice[k] = mean(Put_OptionValue_discounted)
  Put_OptionPrice_antithetic[k] = mean(Put_OptionValue_discounted_antithetic)
  Put_OptionPrice_cont[k] = mean(Put_OptionValue_discounted + c_star_Put*(h))
}
```

The following section calculates the three different call and put option prices as well as 95% confidence intervals for the prices. The mean price for the 20 outer loops as well as the standard deviation of this prices is calculated. Lastly, the fOptions library is called to calculate the call and put prices according to the Black-Scholes formula: Call : $C(S_t, t) = N(d_1)S_t - N(d_2)Ke^{-r_f(T)}$ and Put: $P(S_t, t) = N(-d_2)Ke^{-r_f(T)} - N(-d_1)S_t$, where N(.) is the cumulative distribution function of the standard normal distribution, T is the time to maturity, S_t is the current stock price, K is the strike price, r_f is the risk free rate, σ, the volatility of the stock returns, $d_1 = \frac{1}{\sigma\sqrt{T}}[ln(\frac{S_t}{K}) + (r_f + \frac{1}{2}\sigma^2)T]$ and $d_2 = d_1 - \sigma\sqrt{T}$

```
Option_Price = mean(OptionPrice)
s = sd(OptionPrice)
```

```
upper = Option_Price + qnorm(.975)*s/sqrt(n)
lower = Option_Price - qnorm(.975)*s/sqrt(n)

Option_Price_antithetic = mean(OptionPrice_antithetic)
s_2 = sd(OptionPrice_antithetic)
upper_antithetic = Option_Price_antithetic + qnorm(.975)*s_2/sqrt(n)
lower_antithetic = Option_Price_antithetic - qnorm(.975)*s_2/sqrt(n)

Option_Price_cont = mean(OptionPrice_cont)
s_3 = sd(OptionPrice_cont)
upper_cont = Option_Price_cont + qnorm(.975)*s_3/sqrt(n)
lower_cont = Option_Price_cont - qnorm(.975)*s_3/sqrt(n)

Put_Option_Price = mean(Put_OptionPrice)
s_Put = sd(Put_OptionPrice)
upper_Put = Put_Option_Price + qnorm(.975)*s_Put/sqrt(n)
lower_Put = Put_Option_Price - qnorm(.975)*s_Put/sqrt(n)

Put_Option_Price_antithetic = mean(Put_OptionPrice_antithetic)
s_Put2 = sd(Put_OptionPrice_antithetic)
upper_Put_antithetic = Put_Option_Price_antithetic + qnorm(.975)*s_Put2/sqrt(n)
lower_Put_antithetic = Put_Option_Price_antithetic - qnorm(.975)*s_Put2/sqrt(n)

Put_Option_Price_cont = mean(Put_OptionPrice_cont)
s_Put3 = sd(Put_OptionPrice_cont)
upper_Put_cont = Put_Option_Price_cont + qnorm(.975)*s_Put3/sqrt(n)
lower_Put_cont = Put_Option_Price_cont - qnorm(.975)*s_Put3/sqrt(n)

suppressMessages(library(fOptions))
BlackScholesOption_Price=BlackScholesOption("c",Stock_price,Strike_price,
  Maturity_in_years,Risk_free_rate,Risk_free_rate,Volatility)@price
BlackScholesOption_Price_put=BlackScholesOption("p",Stock_price,Strike_price,
  Maturity_in_years,Risk_free_rate,Risk_free_rate,Volatility)@price
```

This section outputs the simulated call option price with confidence interval, then the same after applying the anithethic variable variance reduction method, and thirdly after applying the control variate method. Lastly, the option price according to Black-Scholes is shown.

```
Option_Price
```

```
## [1] 31.11864
```

```
upper; lower
```

```
## [1] 31.13186
```

```
## [1] 31.10542
```

```
Option_Price_antithetic
```

```
## [1] 30.99845
```

```
upper_antithetic; lower_antithetic
```

```
## [1] 31.0066
```

```
## [1] 30.99029
```

```
Option_Price_cont
```

```
## [1] 30.99589
```

```
upper_cont; lower_cont
```

```
## [1] 31.00584
```

```
## [1] 30.98593
```

```
BlackScholesOption_Price
```

```
## [1] 30.96353
```

This section outputs the simulated put option price with confidence interval, then the same after applying the anithethic variable variance reduction method, and thirdly after applying the control variate method. Lastly, the option price according to Black-Scholes is shown.

```
Put_Option_Price
```

```
## [1] 35.37357
```

```
upper_Put; lower_Put
```

```
## [1] 35.38139
```

```
## [1] 35.36575
```

```
Put_Option_Price_antithetic
```

```
## [1] 35.45342
```

```
upper_Put_antithetic; lower_Put_antithetic
```

```
## [1] 35.4548
```

```
## [1] 35.45205
```

```
Put_Option_Price_cont
```

```
## [1] 35.44585
```

```
upper_Put_cont; lower_Put_cont
```

```
## [1] 35.44824
```

```
## [1] 35.44347
```

```
BlackScholesOption_Price_put
```

```
## [1] 35.45044
```

This section compares the standard deviation without variance reduction, with antithetic variable method and control variate method for call and put options.

```
s
```

```
## [1] 0.6746164
```

```
s_2
```

```
## [1] 0.4162349
```

```
s_3
```

```
## [1] 0.5079902
```

```
# percentage reduction of antithetic variable method for the call option
(1-s_2/s)*100
```

```
## [1] 38.3005
```

```
# percentage reduction of control variate method for the call option
(1-s_3/s)*100
```

```
## [1] 24.6994
```

```
s_Put
```

```
## [1] 0.3991017
```

```
s_Put2
```

```
## [1] 0.07006349
```

```
s_Put3
```

```
## [1] 0.1216265
```

```
# percentage reduction of antithetic variable method for the put option
(1-s_Put2/s_Put)*100
```

```
## [1] 82.4447
```

```
# percentage reduction of control variate method for the put option
(1-s_Put3/s_Put)*100
```

```
## [1] 69.52493
```

Conclusion

This project demonstrates the benefit of variance reduction techniques when estimating European option prices using Monte Carlo simulations. Using an example we show that the variance of the call option can be reduced by about 40% and the variance of the put option even further. The anithetic variable method reduces the variance often (but not always) more compared to the control variate method. The simulated prices are all close to the prices according to Black-Scholes. As we increase the number of iterations and decrease the time between stock price updates, the difference between our estimates and the Black-Scholes prices decreases.