American Option Action Timing

Young Seok Seo

October 29, 2021

After I read the "Evaluating the Longstaff-Schwartz method for pricing of American options" written by William Gustafsson and "Valuing American Options by Simulation: A Simple Least-Squares Approach" by Francis A. Longstaff and Eduardo S. Schwartz, I compare those two paper to see they are consistent or not. I mostly agree with the paper by Gustafsson. However, I think some details need to be added in order to have compelling argument.

This is his LSM algorithm. Gustaffson initiate M stochastic paths for the asset prices with time to maturity N. Then he used dynamic programming to find that discounted payoff is greater than the 0. This indicates that he is trying to find the path that is in the money. Using found the path that is in the money, he calculated the coefficient 'beta'. Once he generates the estimated value of conditional expectation using 'beta', compare the conditional expectation with discounted payoff to find the best optimal time to exercise.

In Gustafsson's paper, he detected the exercise boundary by taking mean of S(t*_i) at each t*_i as decribed in the Equation 2.2. However, in my algorithms, I take maximum S(t*_i) at each t*_i in put option and minimum S(t*_i) at each t*_i in call option to get the exercise boundary. Let's say S refers to the stopping region, and B(t) as a boundary point at t and B_p(t) is the boundary point of put option at t. The paper that I found, "Approximating the Early Exercise Boundary for American-Style Options" by Toshikazu Kimura, argues that

- B_p(t) = sup {S|P(t,s) = max(K-S)}
- B_c(t) = inf {S|P(t,s) = max(K-S)}

So, instead taking mean value of S(t*_i), I will take maximum and minimum for put and call boundary line.

Comparing the result of Gustaffson's

Based on the package, I calculated the option value of American put with following parameters.

Risk_free_rate = 0.03.

Volatility = 0.15.

S(0) = 110 : Initial Price.

Dt = 1/100 : the number of opportunity to exercise the option in a year.

K = 100 : Strike Price.

Then, I would like to compare how the value of put option changes as increasing number of simulations.

```
library(LSMRealOptions)
```

```
## Warning: package 'LSMRealOptions' was built under R version 4.0.5
```

```r
set.seed(1)
paths_1e1 <- GBM_simulate(n = 10, t = 1, mu = 0.03, sigma = 0.15, S0 = 110, dt = 1/100)
put_option_value_1e1 <- LSM_american_option(state_variables = paths_1e1,
                                          payoff = paths_1e1,
                                          K = 100,
                                          dt = 1/100,
                                          rf = 0.03,
                                          verbose = TRUE)
#put_option_value_1e1[1]

paths_1e2 <- GBM_simulate(n = 100, t = 1, mu = 0.03, sigma = 0.15, S0 = 110, dt = 1/100)
put_option_value_1e2 <- LSM_american_option(state_variables = paths_1e2,
                                             payoff = paths_1e2,
                                             K = 100,
                                             dt = 1/100,
                                             rf = 0.03,
                                             verbose = TRUE)
#put_option_value_1e2[1]

paths_1e3 <- GBM_simulate(n = 1e3, t = 1, mu = 0.03, sigma = 0.15, S0 = 110, dt = 1/100)
put_option_value_1e3 <- LSM_american_option(state_variables = paths_1e3,
                                             payoff = paths_1e3,
                                             K = 100,
                                             dt = 1/100,
                                             rf = 0.03,
                                             verbose = TRUE)
#put_option_value_1e3[1]

paths_1e4 <- GBM_simulate(n = 1e4, t = 1, mu = 0.03, sigma = 0.15, S0 = 110, dt = 1/100)
put_option_value_1e4 <- LSM_american_option(state_variables = paths_1e4,
                                               payoff = paths_1e4,
                                               K = 100,
                                               dt = 1/100,
                                               rf = 0.03,
                                               verbose = TRUE)
#put_option_value_1e4[1]

paths_1e5 <- GBM_simulate(n = 1e5, t = 1, mu = 0.03, sigma = 0.15, S0 = 110, dt = 1/100)
put_option_value_1e5 <- LSM_american_option(state_variables = paths_1e5,
                                                payoff = paths_1e5,
                                                K = 100,
                                                dt = 1/100,
                                                rf = 0.03,
                                                verbose = TRUE)
#put_option_value_1e5[1]

paths_1e6 <- GBM_simulate(n = 1e6, t = 1, mu = 0.03, sigma = 0.15, S0 = 110, dt = 1/100)
put_option_value_1e6 <- LSM_american_option(state_variables = paths_1e6,
                                              payoff = paths_1e6,
                                              K = 100,
                                              dt = 1/100,
                                              rf = 0.03,
                                              verbose = TRUE)
```

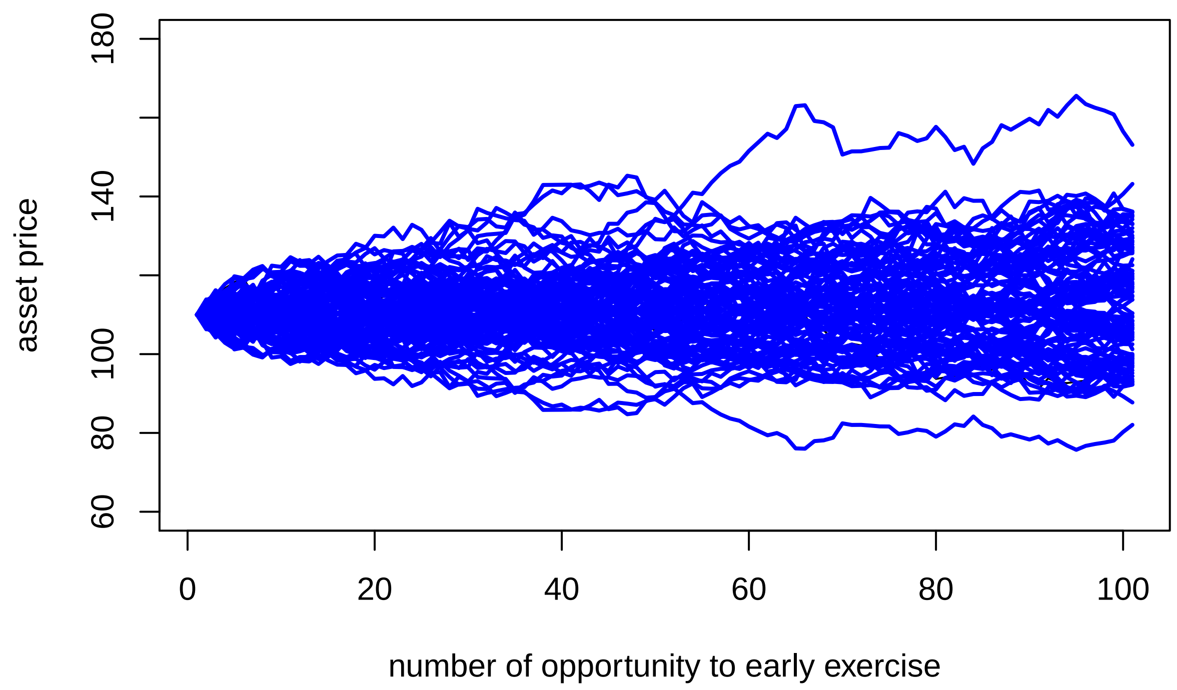
```
#put_option_value_1e6[1]

option_value <- c(put_option_value_1e1[1],put_option_value_1e2[1],put_option_value_1e3[1],
                  put_option_value_1e4[1],put_option_value_1e5[1],put_option_value_1e6[1])


option_value
```

```
## $Value
## [1] 2.689772
##
## $Value
## [1] 2.387453
##
## $Value
## [1] 1.853244
##
## $Value
## [1] 1.842551
##
## $Value
## [1] 1.824808
##
## $Value
## [1] 1.822409
```

As results indicated, the value of option with large number of simulations is getting closer to the result of Gustaffson's. Gustaffson got 1.8282 of option value when the initial price is 110. According to my result, I got 1.822409 of option value with the same initial price.

Sample Path for asset price

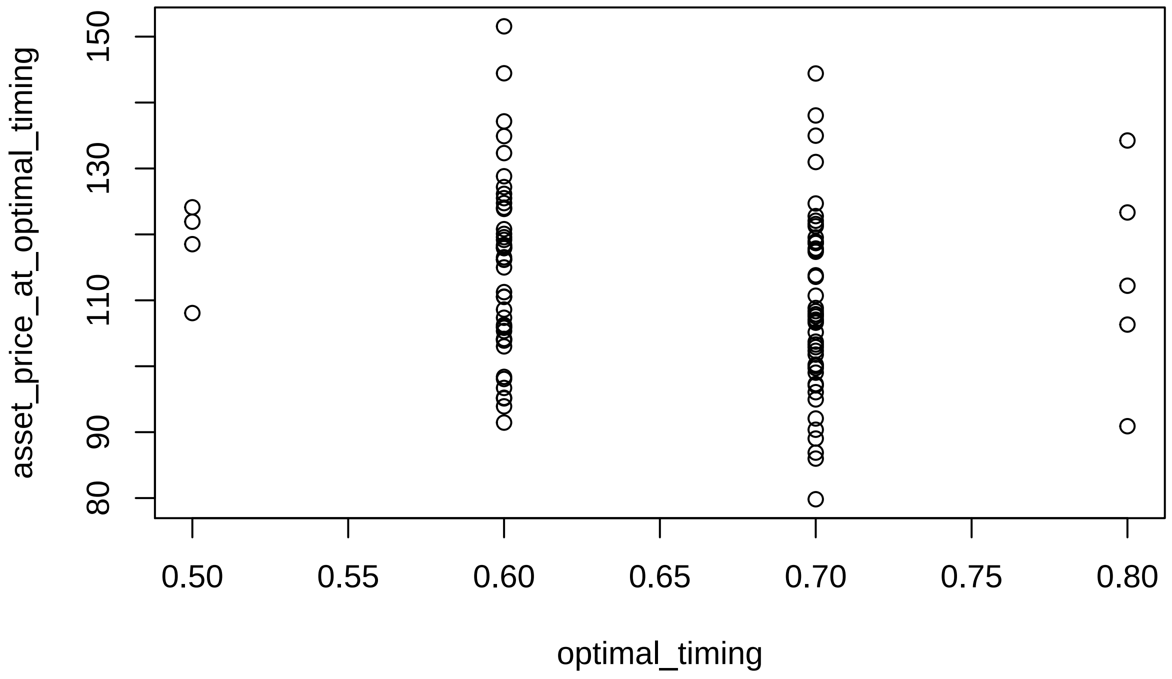
```
plot(paths_1e6[,1], type = 'l',ylim= c(60,180), xlab = 'number of opportunity to early exercise', ylab 
for (i in 2:100) {
  points(paths_1e6[,i] , lwd=2, col = 'blue',type = 'l')
}
```



Although I agree with Gustaffson's algorithm, I do not agree the part that he takes mean value of S(t*_i) at each t*_i as I mentioned above. So I implemented my own algorithms using LSMRealOptions package in R. In my algorithm, I used GBM_simulate function to generate the 100 paths then calculated the option value and optimal time to exercise at each path with K=100, S_0 = 110, volatility = 0.15, risk_free rate = 3% with 100 opportunity to exercise the option in a year. Then, I was able to calculate the asset price at optimal time for each path. Once I get 100 optimal time and corresponding asset price, I take max value of asset price that is grouped by each opportunity time.

Put option

```
dt <- 1/100 # Number of opportunity to exercise early during time 0 to time to maturity
K <- 100 # Strike Price
rf <- 0.03 # risk_free
n_simulations <- 100
sigma <- 0.15
S0 <- 110 # Initial Prices
TTM <- rep(1,n_simulations)
value_timing_matrix <- matrix(0, 100, 2,
   dimnames = list(NULL, c("option_value", "Optimal_timing")))

set.seed(2)
for(i in 1:100) {
  paths <- GBM_simulate(n = n_simulations, t = TTM[i],mu = rf,
                        sigma = sigma, S0 = S0, dt = dt)
  output <- LSM_american_option(state_variables = paths,
                        payoff = paths,call = FALSE,K = K,
                        dt = dt,rf = rf, verbose = TRUE,
                        orthogonal = "Laguerre",
                        degree = 3
)
value_timing_matrix[i,1] <- output$Value
value_timing_matrix[i,2]  <- output$`Expected Timing`
}

optimal_timing<-round(value_timing_matrix[,2],1)
#Asset_price_at_optimal_timing
asset_price_at_optimal_timing<-c()
for(i in 1:100) {
  asset_price_at_optimal_timing<-
    c(asset_price_at_optimal_timing,paths[(value_timing_matrix[i,2] * (1/dt)), i])
}
table_2 <- cbind.data.frame(optimal_timing,asset_price_at_optimal_timing)
table_2<- table_2[order(table_2$optimal_timing),c(1,2)]
plot(optimal_timing,asset_price_at_optimal_timing)
```



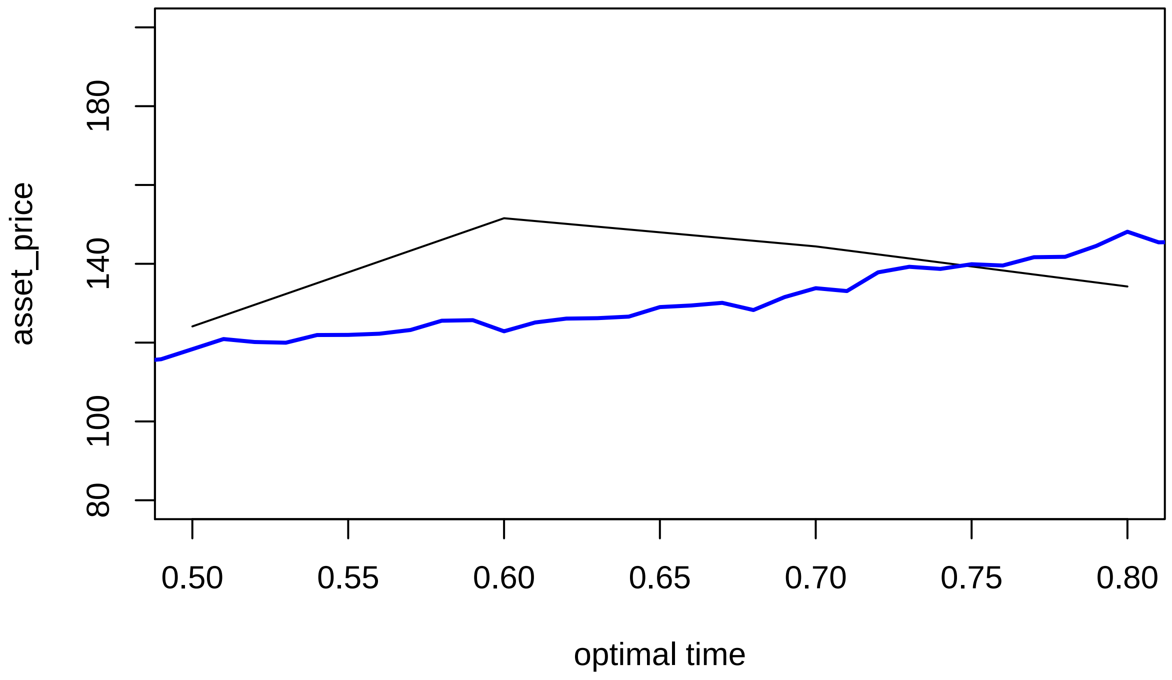
Maximum of Asset price at corresponding timing

```
max_price_by_timing<-
  aggregate(table_2$asset_price_at_optimal_timing, list(table_2$optimal_timing), FUN=max)
colnames(max_price_by_timing) <- c('timing', 'asset_price_at_optimal_timing')
max_price_by_timing
```

```
##   timing asset_price_at_optimal_timing
## 1    0.5                      124.1119
## 2    0.6                      151.5615
## 3    0.7                      144.4099
## 4    0.8                      134.2436
```

Boundary Line of the Put option

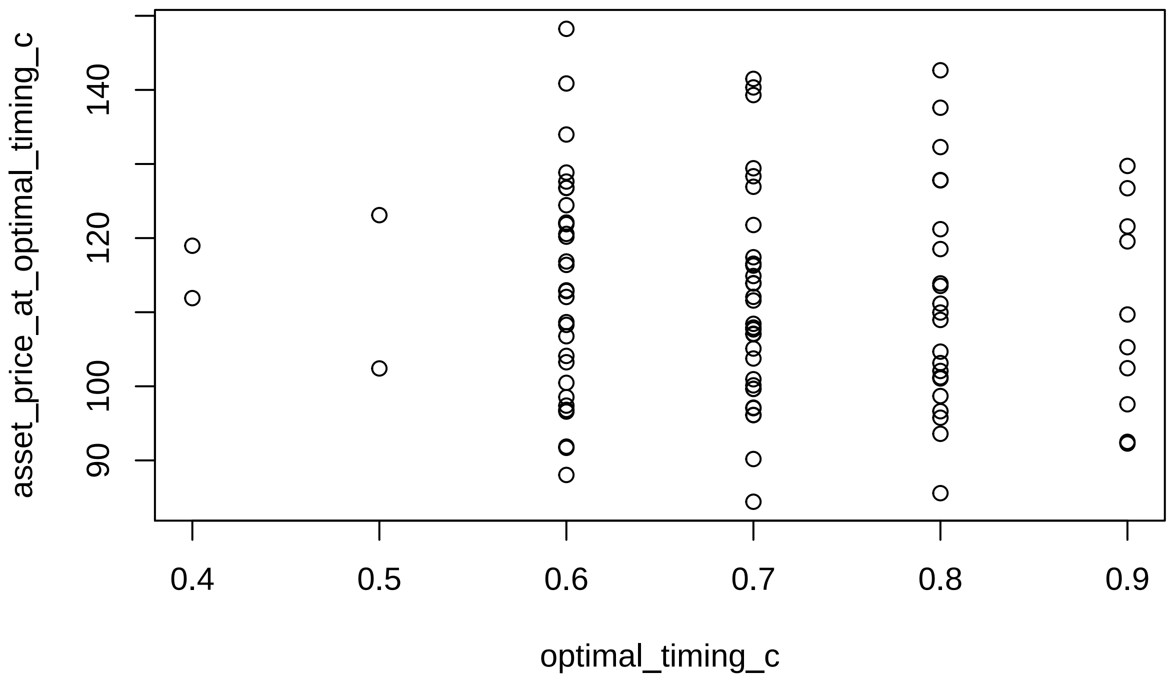
```
index <- 1
for (i in 1:101) {
  index[i] <- (dt)*i
}

plot(max_price_by_timing$asset_price_at_optimal_timing ~ max_price_by_timing$timing,
     xlab='optimal time', ylab='asset_price',type = 'l', ylim= c(80,200))
points(paths[,19]~index , lwd=2, col="blue", type = 'l')
```



Call option

For the call option, the basic algorithms are exactly same as put option. Only thing that I change from put option is take min value of asset price at each time instead of taking max. However, as you may know, early exercise is never optimal for American call option theoretically. Here, I assumed that I do not pay dividends of stock and the interest rate. But in real life, I need to pay Strike price K at time of exercise. However, option holder can save their interest.

```r
dt <- 1/100
K <- 100
rf <- 0.03
n_simulations <- 100
sigma <- 0.15
S0 <- 110
TTM <- rep(1,n_simulations)
value_timing_matrix_c <-
  matrix(0, 100, 2, dimnames = list(NULL, c("option_value", "Optimal_timing")))

set.seed(1)
for(i in 1:100) {
  path_c <- GBM_simulate(n = n_simulations, t = TTM[i],
                                   mu = rf, sigma = sigma, S0 = S0, dt = dt)
  output_c <- LSM_american_option(state_variables = path_c,
                                payoff = path_c, call = TRUE,
                                K = K,dt = dt,rf = rf,
                                verbose = TRUE,
                                orthogonal = "Laguerre",degree = 3)
  value_timing_matrix_c[i,1] <- output_c$Value
  value_timing_matrix_c[i,2]  <- output_c$`Expected Timing`
}

optimal_timing_c<-round(value_timing_matrix_c[,2],1)
#Asset_price_at_optimal_timing
asset_price_at_optimal_timing_c<-c()
for(i in 1:100) {
  asset_price_at_optimal_timing_c<-
    c(asset_price_at_optimal_timing_c,path_c[(value_timing_matrix_c[i,2] * (1/dt)), i])
}

table_2_c <- cbind.data.frame(optimal_timing_c,asset_price_at_optimal_timing_c)
table_2_c<- table_2_c[order(table_2_c$optimal_timing_c),c(1,2)]
plot(optimal_timing_c,asset_price_at_optimal_timing_c)
```



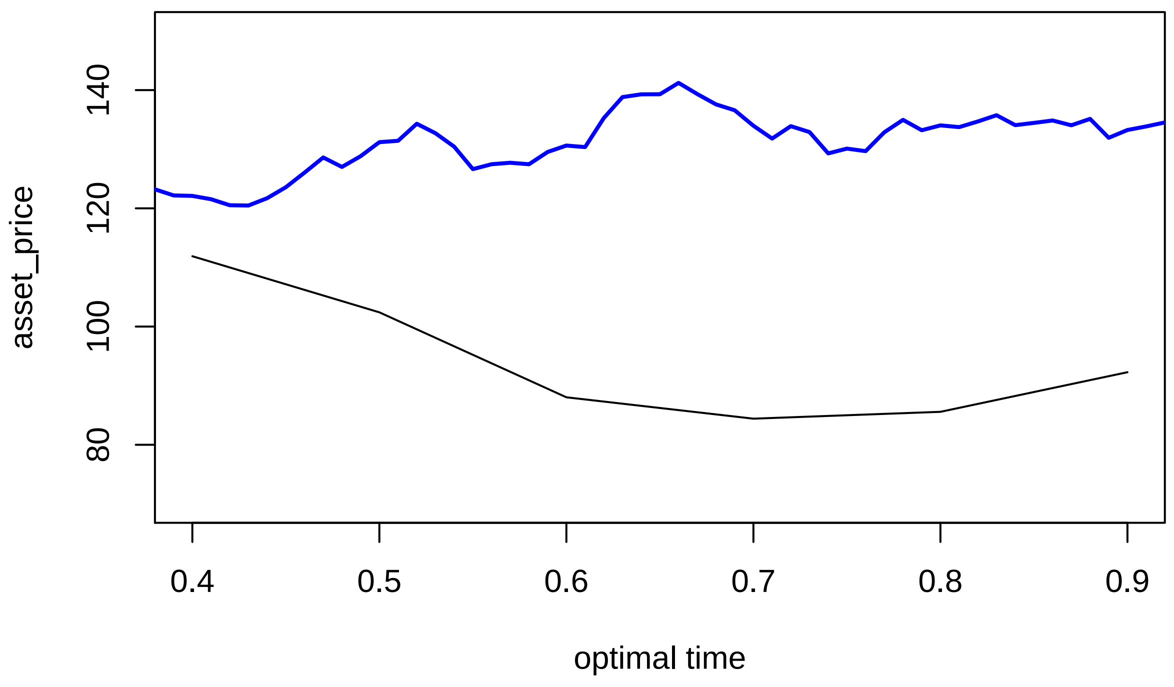
Minimum of Asset price at corresponding timing

```
min_price_by_timing_c<-
  aggregate(table_2_c$asset_price_at_optimal_timing_c, list(table_2_c$optimal_timing_c), FUN=min)
colnames(min_price_by_timing_c) <- c('timing_c', 'asset_price_at_optimal_timing_c')
min_price_by_timing_c
```

```
##   timing_c asset_price_at_optimal_timing_c
## 1      0.4                       111.89916
## 2      0.5                       102.40785
## 3      0.6                        88.03346
## 4      0.7                        84.41846
## 5      0.8                        85.57531
## 6      0.9                        92.27419
```

Boundary Line of the call option

```
index_c <- 1
for (i in 1:101) {
  index_c[i] <- (dt)*i
}
plot(min_price_by_timing_c$asset_price_at_optimal_timing_c ~ min_price_by_timing_c$timing_c,
         type = 'l',xlab='optimal time', ylab='asset_price',ylim= c(70,150))
points(path_c[,3]~index_c , lwd=2, col="blue", type = 'l')
```


asset_price
optimal time