Q3

Chayapon Lee-Isranukul

2024-10-30

Valuing American Options by Simulation: A Simple Least-Squares Approach

```
S0 = K = 100
t = 1/12
r = 0.04
q = 0.02
sigma = 0.2

S0
```

```
## [1] 100
```

```
K
```

```
## [1] 100
```

```
m = 10
# m = c(10,20,30,40,50)
n = 10
# n = c(1000,1000*4, 100*4**2, 1000*4**3, 1000*4**4)
delta_t = t/m

S_df = data.frame(matrix(ncol = 0, nrow = n))
S = rep(S0,n)
#S_df[[paste0("S",0)]] = S
for (i in 1:m) {

  delta_S =  S * r * delta_t + sigma * S * sqrt(delta_t) * rnorm(n)
  S = S + delta_S
  S_df[[paste0("S",i)]] = S
}

#write.csv(S_df, file = "S_df.csv", row.names = FALSE)
```

```
#S_df = read.csv("S_df.csv", header = TRUE)
exercise_times = rep(m,n)
payoff_df = data.frame(apply(K - S_df, c(1, 2), function(x) max(x, 0)))
exercise_times
```

```
##  [1] 10 10 10 10 10 10 10 10 10 10
```

```r
payoff_df
```

```
##           S1        S2       S3       S4        S5        S6       S7       S8
## 1  4.4919286 5.7893305 6.190921 4.631088 5.1341445 3.3017949 3.688937 5.386751
## 2  0.0000000 0.0000000 0.000000 0.000000 0.0000000 0.0000000 0.000000 0.000000
## 3  0.4880020 1.1196107 3.229358 2.581068 0.7069338 5.7755974 8.811743 5.498695
## 4  2.3800135 2.4124723 3.042444 4.416562 2.0597036 0.8791276 3.011384 4.830824
## 5  0.0000000 0.0000000 0.000000 0.000000 0.0000000 0.0000000 0.000000 0.000000
## 6  0.0000000 0.0000000 0.000000 0.000000 0.0000000 0.0000000 0.000000 0.000000
## 7  0.0000000 0.0000000 0.000000 0.000000 0.0000000 0.0000000 0.000000 0.000000
## 8  0.7641161 0.4060501 0.000000 3.210298 4.6562621 4.2668184 4.642930 4.675247
## 9  1.8662425 0.0000000 0.000000 0.000000 0.0000000 0.0000000 0.000000 0.000000
## 10 2.6124749 3.7578507 3.708880 4.959248 4.7599083 4.9141822 5.167215 7.035441
##           S9      S10
## 1  3.0020918 5.117076
## 2  0.0000000 0.000000
## 3  8.1706085 8.252637
## 4  3.1050497 4.068907
## 5  0.0000000 0.000000
## 6  0.1871352 0.000000
## 7  0.0000000 0.000000
## 8  4.9502175 4.089909
## 9  0.0000000 0.000000
## 10 8.2803096 6.162956
```

```r
payoff_scaled_df = payoff_df/K
S_scaled_df = S_df/K
```

```r
simulate_S_paths <- function(S0,r,q,sigma,n,m,delta_t) {
  S = rep(S0,n)
  S_df = data.frame(matrix(ncol = 0, nrow = n))
  for (i in 1:m) {

  delta_S =  S * (r-q) * delta_t + sigma * S * sqrt(delta_t) * rnorm(n)
  S = S + delta_S
  S_df[[paste0("S",i)]] = S
  }
  return(S_df)
}

laguerre_poly <- function(X, i) {
  # Ensure X is a vector
  if (!is.vector(X)) stop("X must be a vector")

  # Calculate Laguerre polynomial L_i(X)
  result <- 0
  for (n in 0:i) {
    result <- result + ((-1)^n * choose(i, n) * X^n / factorial(n))
  }

  return(result * exp(-X / 2))  # Apply exponential decay for the generalized form
```

```r
}

monte_carlo <- function(sample) {

    x_bar = sample[1]
    y_bar = x_bar ** 2

    for  (i in 2:n) {
      x = sample[i]
      x_bar = (1-1/i) * x_bar + (1/i) * x
      y_bar = (1-1/i) * y_bar + (1/i) * (x ** 2)
    }
    return(list(x_bar = x_bar, y_bar=y_bar))
}

#LSM_American_put <- function(S, )
```

```r
m = 20
# m = c(10,20,30,40,50)
n = 10
# n = c(1000,1000*4, 100*4**2, 1000*4**3, 1000*4**4)
delta_t = t/m
k_regressors = 3

S_df = simulate_S_paths(S0,r,q,sigma,n,m,delta_t)

exercise_times = rep(m,n)
payoff_df = data.frame(apply(K - S_df, c(1, 2), function(x) max(x, 0)))

payoff_scaled_df = payoff_df/K
S_scaled_df = S_df/K


for (i in (m-1):1) {
  itm_idx = payoff_df[,i] > 0
  future_cashflows = mapply(function(row, col) payoff_df[row, col], row = 1:nrow(payoff_scaled_df), col
  discount_times = delta_t * (exercise_times - i)
  Y = future_cashflows * exp(-r*discount_times[itm_idx])
  # X = S_df[,i][itm_idx]
  S_itm = S_scaled_df[,i][itm_idx]

  # Laguerre polynomial
  X = data.frame(matrix(ncol = 0, nrow = length(S_itm)))
  for (o_i in 0:(k_regressors-1)) {

    X[[paste0("L",o_i)]] = laguerre_poly(S_itm, o_i)
  }

  model = lm(Y ~ ., data=X)
  #summary(model)
  cond_exp_Y = predict(model, newdata = data.frame(X))
  names(cond_exp_Y) = NULL
  current_itm_payoff = payoff_scaled_df[,i][itm_idx]
```

```
  exercise_times[itm_idx] = ifelse(current_itm_payoff > cond_exp_Y, i, exercise_times[itm_idx])
}
```

```
payoff_decisions = mapply(function(row, col) payoff_df[row, col], row = 1:nrow(payoff_df), col = exerci
discount_times = delta_t * (exercise_times - i)
option_path_values = payoff_decisions * exp(-r*discount_times)
option_expected_value = mean(option_path_values)
option_expected_value
```

```
## [1] 6.227917
```

```
exercise_times
```

```
##  [1] 20 20 20 20 20 20 20 20 20 20
```

```
as.matrix(payoff_df)
```

```
##              S1        S2        S3        S4        S5         S6       S7
##  [1,] 0.0000000 1.6185843 1.9752744 3.1491036 2.4530771 1.78626680 6.829579
##  [2,] 1.2120658 1.6726615 2.3095217 1.5058542 2.8733731 3.94699673 5.396752
##  [3,] 0.0000000 0.2960233 0.6700930 0.9175563 0.0000000 0.00000000 0.000000
##  [4,] 1.8558197 1.7919563 1.2070475 0.7892984 0.8598143 2.22231008 1.476780
##  [5,] 0.1600337 1.1600964 0.0000000 0.0000000 0.0000000 1.73062785 3.341959
##  [6,] 1.8133342 1.7307656 2.0454264 2.7884850 2.0280337 3.06846725 2.498444
##  [7,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000 0.000000
##  [8,] 1.5774478 0.9957723 0.0000000 0.0000000 0.4857740 0.06593954 1.138612
##  [9,] 0.2183538 1.0582985 2.0251480 3.2301869 4.5529313 4.26291385 4.435853
## [10,] 1.2685098 0.4447025 0.1487807 0.0000000 0.0000000 0.00000000 0.000000
##              S8        S9      S10       S11        S12       S13       S14
##  [1,] 7.1245611 8.5857549 9.053808 9.1027552 10.2645001 10.068183 8.7799308
##  [2,] 7.1159722 6.3865199 5.074140 3.8001742  2.0741812  2.901491 0.8518748
##  [3,] 0.4558399 0.0000000 0.000000 0.0000000  0.0000000  0.000000 0.0000000
##  [4,] 2.0526488 2.2207098 2.191176 1.8234082  1.1360416  1.671447 2.9061580
##  [5,] 4.9821641 5.6280244 5.895603 4.4931042  4.2591952  5.064937 4.9399564
##  [6,] 3.2730909 3.0353581 3.877378 2.2045540  2.4838635  1.657700 0.4504748
##  [7,] 0.0000000 0.0000000 0.000000 0.9224321  1.8710530  3.697547 3.6978670
##  [8,] 0.5968398 0.2290923 1.155489 0.9887132  0.0000000  1.775845 1.6864910
##  [9,] 4.6392999 6.3693901 7.616563 5.9059088  7.3415684  7.889491 7.4392901
## [10,] 0.0000000 0.0000000 0.000000 0.0000000  0.7425103  2.556468 1.1315271
##             S15       S16       S17       S18       S19       S20
##  [1,] 8.7142107 8.2096590  9.445902  9.688632 10.416872 11.480838
##  [2,] 1.4387611 2.8980337  3.677462  3.940941  3.372824  4.350477
##  [3,] 0.0000000 0.0000000  0.000000  0.000000  0.000000  0.000000
##  [4,] 3.2781886 2.2057692  1.004103  2.174961  0.775070  1.611929
##  [5,] 6.6720058 6.3271580  8.060362 10.182532 10.602401  9.176789
##  [6,] 0.4937271 0.7946802  0.000000  0.000000  2.388905  5.475518
##  [7,] 5.0762722 5.7595807  5.708501  5.120408  6.059503  6.582621
##  [8,] 1.5245055 3.1698284  2.836767  4.609294  5.769435  5.540855
##  [9,] 7.8413655 9.2760326 10.757954 10.269168  9.510245 12.380255
## [10,] 4.7736016 3.8003061  4.596368  5.328324  6.267857  5.877419
```