Tarification par Monte Carlo d’une Option Put européenne de type Up-and-Out

Marcel Héritier K.

2023-12-21

## à propos

Dans ce tutoriel, nous voyons comment utiliser la méthode de simulation de Monte Carlo pour évaluer les produits financiers dérivés. La notation mathématique et les exemples sont tirés du livre Implementing Derivatives Models de Les Clewlow et Chris Strickland.

La valorisation des produits financiers dérivés par des simulations de Monte Carlo n’est possible qu’en utilisant les mathématiques financières de l’évaluation neutre du risque et en simulant des trajectoires d’actifs neutres du point de vue du risque. La formule de prix de l’espérance risque-neutre en temps continu est donnée par :

# Dynamique d’un MBG

Ici, nous disposons d’un taux d’intérêt constant, le facteur d’actualisation est donc , et la dynamique des actions est modélisée par le mouvement brownien géométrique (GBM).

La solution suivante s’applique au prix de l’action S dans le cadre d’une dynamique neutre à l’égard du risque :

où

## Particularités des options barrières

Lors de la détermination du prix d’options complexes ou exotiques dépendant d’un scénario, un produit très répandu est l’option à barrière. Il s’agit d’options européennes standard à expiration, mais elles cessent d’exister ou n’existent que si le prix du sous-jacent franchit une barrière prédéterminée. Ce niveau de barrière peut avoir un seuil de déclenchement continu ou discret .

Pour une option *barrière* du type *Up-and-out*, nous avons :

Pour autant que, pour tout ,

* Si et , alors
* Sinon et , alors

S0 = 100 # initial stock price  
K = 100 # strike price  
T = 1 # time to maturity in years  
H = 125 # up-and-out barrier price/value  
r = 0.01 # annual risk-free rate  
vol = 0.2 # volatility (%)  
  
N = 100 # number of time steps  
M = 1000 # number of simulations

## Simulation

Ici, nous simulons le prix de l’action directement, car nous avons besoin de cette valeur lors du calcul pour la comparer à la barrière .

#------------------------#  
# Slow Solution - Steps #  
#------------------------#  
# Initialize variables  
  
dt <- T/N  
nudt <- (r - 0.5 \* vol^2) \* dt  
volsdt <- vol \* sqrt(dt)  
erdt <- exp(r \* dt)  
  
# Standard Error Placeholders  
sum\_CT <- 0  
sum\_CT2 <- 0  
  
# Monte Carlo Method  
for (i in 1:M) {  
 # Barrier Crossed Flag  
 BARRIER <- FALSE  
 St <- S0  
   
 for (j in 1:N) {  
 epsilon <- rnorm(1)  
 Stn <- St \* exp(nudt + volsdt \* epsilon)  
 St <- Stn  
   
 if (St >= H) {  
 BARRIER <- TRUE  
 break  
 }  
 }  
   
 if (BARRIER) {  
 CT <- 0  
 } else {  
 CT <- max(0, K - St)  
 }  
   
 sum\_CT <- sum\_CT + CT  
 sum\_CT2 <- sum\_CT2 + CT^2  
}  
  
# Compute Expectation and SE  
C0 <- exp(-r \* T) \* sum\_CT/M  
sigma <- sqrt((sum\_CT2 - sum\_CT^2/M) \* exp(-2 \* r \* T) / (M - 1))  
SE <- sigma / sqrt(M)  
  
cat("Call value is $", round(C0, 2), " with SE +/- ", round(SE, 3), "\n")

## Call value is $ 7.62 with SE +/- 0.335

## Vectorisation

#-----------------------------#  
# Vectorized Implementation #  
#-----------------------------#  
  
# Set seed for reproducibility  
set.seed(689)  
  
# Precompute constants  
dt <- T/N  
nudt <- (r - 0.5 \* vol^2) \* dt  
volsdt <- vol \* sqrt(dt)  
erdt <- exp(r \* dt)  
  
# Monte Carlo Method  
Z <- matrix(rnorm(N \* M), nrow = N, ncol = M)  
delta\_St <- nudt + volsdt \* Z  
ST <- matrix(S0, nrow = N + 1, ncol = M)  
for (i in 2:(N + 1)) {  
 ST[i,] <- ST[i - 1,] \* exp(delta\_St[i - 1,])  
}  
  
# Apply Barrier Condition to ST matrix  
mask <- apply(ST, 2, function(x) any(x >= H))  
ST[, mask] <- 0  
  
CT <- pmax(0, K - ST[N + 1, ST[N + 1,] != 0])  
C0 <- exp(-r \* T) \* sum(CT) / M  
  
sigma <- sqrt(sum((exp(-r \* T) \* CT - C0)^2) / (M - 1))  
SE <- sigma / sqrt(M)  
  
cat("Call value is $", round(C0, 2), " with SE +/- ", round(SE, 3), "\n")

## Call value is $ 7.53 with SE +/- 0.309